Kommunikationsnetzwerkplanung unter Kosten- und Zuverlässigkeitsgesichtspunkten mit Hilfe von evolutionären Algorithmen

Dissertation

zur Erlangung des akademischen Grades

doctor rerum politicarum (Dr. rer. pol.)

eingereicht an der Fakultät für Wirtschaftswissenschaften
an der Technischen Universität Ilmenau

vorgelegt von **Dipl.-Wirtsch.-Inf. Dirk Reichelt**

1. Gutachter: Univ.-Prof. Dr.-Ing. habil. Peter Gmilkowsky, TU Ilmenau
2. Gutachter: Univ.-Prof. Dr. rer. pol. habil. Armin Heinzl, Universität Mannheim

Tag der Einreichung: 6.12.2005
Tag der wissenschaftlichen Aussprache: 31.3.2006

urn:nbn:de:gbv:ilm1-2006000022

Für Katrin und Lena

Danksagung

Diese Arbeit entstand im Rahmen meiner Tätigkeit am Lehrstuhl für Wirtschaftsinformatik I an der TU Ilmenau. Ich möchte mich an dieser Stelle zunächst bei Professor Gmilkowsky für ein Unterstützung und die gewährten Freiräume bedanken. Mein Dank gebührt ebenfalls Dr. Franz Rothlauf, der trotz der großen Entfernung zwischen Ilmenau und Mannheim stets ein wachsames Auge auf den Fortschritt der Arbeit hatte und mit vielen konstruktiven Kommentaren und einer Vielzahl von Diskussionen maßgeblich die Qualität der vorliegenden Arbeit beeinflußt hat. Ebenfalls bedanken möchte ich mich bei Prof. Heinzl für die Erstellung der Zweigutachtens und die großartige Unterstützung der Mitarbeiter seines Lehrstuhls bei gemeinsamen Forschungsprojekten, welche im Rahmen meines Dissertationsvorhabens bearbeitet wurden. Nicht unerwähnt bleiben sollen meine ehemaligen Kollegen Torsten Munkelt, Dr. Sven Völker und Dr. Thomas Döring, die mir bei der Findung der zur bearbeitenden Thematik unter die Arme griffen. Mein Dank gebührt ebenfalls PD Dr. Lars Mönch für viele interessante und anregende Diskusionen, die mich in meinem eingeschlagenen Weg bestärkten.
Vielen Dank an meine Korrekturleser Annekatrin Krieg, Juliane und Arne Borsum sowie meine Eltern.
Für die Unterstützung bei Programmieraufgaben und der Durchführung der Experimente bedanke ich mich bei Daniel Heddergott, Mark Knauf, Stefan Thor, Stefan Gold, Enrico von Otte, Eric Große und Matthias Wimmer. Ebenfalls bedanken möchte ich mich bei Klaus Held, Oliver Fischer, Ilona Cialla und Klaus-Dieter Seeber, welche mich bei der Durchführung der Experimente für diese Arbeit stets mit der notwendigen Rechentechnik unterstützten.
Mein besonderer Dank gilt meiner Frau Katrin, welche mir auch in den schwierigsten Situationen den Rücken frei hielt und als unermüdlicher Korrekturleser jedes einzelne Kapitel akribisch inspizierte.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

Symbolverzeichnis

$\cup_{nondom}$	Fügt eine Lösung/Menge von Lösungen zu einer pareto-optimalen Lösungsmenge hinzu
Δ_j	Differenz zwischen dem maximalen und minimalen Zielfunktionswert der Zielfunktion j
$\geq_n$	Crowding-Comparator des NSGA-II-Verfahrens
$ans(prob)$	Antworten für $prob$
aus	Menge der Ausgaben eines Automaten
$C_{AVG}(P_t)$	Mittelwert der Kosten sämtlicher Lösungen einer Population P_t
C_B	Budgetschranke
C_{Beste}	Kosten der besten Lösung aus allen Durchläufen
$C_{BestEver}$	Kosten der besten bisher bekannten Lösung
C_B^{Relax}	Adaptiver Budgetparameter für SAGA
$c(e_i)$	Kosten der Kante e_i
$C(G_N)$	Summe der Kosten der Kanten E_N in $G_N(V_N, E_N)$
$c(l_k(e_{v_i,v_j}))$	Kosten einer Kante e_{v_i,v_j} für die gewählte Option l_k
$C_{\mathcal{Y}}^{max}$	Maximale Kosten einer Lösung in der Lösungsmenge $\mathcal{Y}$
C_{max}	Kosten der bisher schlechtesten gefundenen gültigen Lösung
C_{min}	Kosten der bisher besten gefundenen gültigen Lösung
$C_{\mathcal{Y}}^{min}$	Minimale Kosten einer Lösung in der Lösungsmenge $\mathcal{Y}$
C_{opt}	Kosten der besten Lösung
$C_{R_{All}^{opt}}$	Kosten der Lösung mit der besten All-Terminal-Zuverlässigkeit (R_{All}^{opt}) in allen GA-Läufen
D	Gradmatrix
D'	Reduzierte Gradmatrix

$deg(v_i)$	Grad eines Knotens v_i
$det(D)$	Determinante einer Matrix D
$dist$	Distanzkennzahl für den Abstand von $\mathcal{Y}_{known}$ zu $\mathcal{Y}_{true}$
$dist_1$	Distanzkennzahl für den mittleren Abstand von $\mathcal{Y}_{true}$ zu $\mathcal{Y}_{known}$
$dist_2$	Distanzkennzahl für den maximalen Abstand von $\mathcal{Y}_{known}$ zu $\mathcal{Y}_{true}$
$dist_3$	Verhältnis von $dist_2$ zu $dist_1$
Dom	Vektor aller Domänen Dom_i für die Entscheidungsvariablen
Dom_i	Domäne für eine Entscheidungsvariable x_i
$d(y', y)$	Distanz zwischen den Lösungen y' und y im Lösungsraum
$\|E\|$	Anzahl der Kanten eines Graphen
E	Menge der Kanten eines Graphen
e_i	Beliebige Kante
E_C	Menge der Kanten eines Schnittes V_C
$\underline{e_i}$	Markiert eine ausgefallene Verbindung e_i
e_{v_i,v_j}	Kante zwischen den Knoten v_i und v_j
ein	Menge der Eingabesymbole
E_N	Kantenmenge eines Netzwerkes N
$\#Eval$	Anzahl der Fitnessbewertungen
$\mathcal{F}_i$	Lösungsfront
$f_{prop}(y)$	Proportionale Fitness einer Lösung y
$FSUM$	Summe aller Fitnesswerte
$f(y)$	Zielfunktionswert einer Lösung y
$f_i(y)$	Zielfunktionswert der Zielfunktion i einer Lösung
G	Beliebiger Graph
g	Genom
$g(e_i)$	Gewicht einer Kante e_i
gen	Anzahl der Generationen

$G(V, E)$	Graph mit der Menge von Knoten V und der Menge von Kanten E
g_i	Ausprägung eines Gens an Position i
G_N	Repräsentation des Netzwerkes N als Graph G
$g(v)$	Gewicht eines Knotens v
$\mathcal{H}$	Lösungsmenge zur Bildung der NSGA-II-Fronten
Ha	Menge der Handlungsalternativen eines Entscheiders
ha_i	Handlungsalternative eines Entscheiders
$\mathcal{I}$	Lösungsmenge bei der Bestimmung der Crowding-Distanz
$\mathcal{I}[y_i]_{Distanz}$	Crowding-Distanz der Lösung y_i
$K_{Eingabe}$	Menge der zu verbindenden Netzwerkknoten
k_{max}	Maximale Anzahl der Technologieoptionen l
l	Technologieoption für eine Verbindung
λ_{ES}	Nachkommenpopulation einer Evolutionsstrategie
$L_{Eingabe}$	Menge der zur Verfügung stehenden Verbindungen für $K_{Eingabe}$
l_k	$k$te Technologieoption für eine Verbindung
$L_{Options}$	Menge der Technologieoptionen
LS	Anzahl der lokalen Suchschritte für COMNETEA
M	Stichprobenumfang für Monte-Carlo-Simulation
μ	Mittelwert
μ_{ES}	Elternpopulation einer Evolutionsstrategie
N	Netzwerk(topologie)
$Nach(y)$	Nachbarschaftslösungsmenge einer Lösung y
NB	Vektor aller Nebenbedingungen nb_i
nb_i	Nebenbedingung
n_{y_i}	Anzahl der von y_i dominierten Lösungen
ω	Entscheidungsvariable für die Anwendung eines Strafterms
P	Beliebige Population

p_{cross}	Crossoverwahrscheinlichkeit
$p_{fail}(e)$	Ausfallwahrscheinlichkeit einer Kante e ($p_{fail}(e) = 1 - r(e)$)
$\Phi(s_i)$	Zusammenhangsfunktion für s_i
p_{init}	Wahrscheinlichkeit für die Erstellung einer Kante
p_{init}^{opt}	Wahrscheinlichkeit zur Auswahl einer Technologieoption
p_{mut}	Mutationswahrscheinlichkeit
pop	Populationsgröße
$p_{Rang}(i)$	Auswahlwahrscheinlichkeit für die Lösung mit dem Rang i unter Verwendung eines linearen Rangschemas
$prob$	Instanz eines berechenbaren Problems $Prob$
$Prob$	Berechenbares Problem
$P(s)$	Eintrittswahrscheinlichkeit für die Realisierung der Kantenzustände s
p_s	Verhältnis zwischen der Anzahl der erfolgreichen Mutationen und Anzahl der durchgeführten Mutationen
P_t	Population zum Zeitpunkt t
P'_t	Menge von Lösungen, welche durch die Rekombination erzeugt wurden
P''_t	Menge von Lösungen, nach der Anwendung des Mutationsoperators
P^*_t	Menge von Lösungen, welche für die Rekombination ausgewählt wurden
r	Gewichtsfunktion
R_0	Minimal geforderte All-Terminal-Zuverlässigkeit
R_2	Source-Terminal-Zuverlässigkeit
R_{All}	All-Terminal-Zuverlässigkeit
$R_{All}^{AVG}(P_t)$	Durchschnittliche All-Terminal-Zuverlässigkeit der Lösungen einer Population P_t
R_{All}^{Beste}	All-Terminal-Zuverlässigkeit der besten Lösung aus allen Durchläufen
$R_{All}^{C_{Beste}}$	All-Terminal-Zuverlässigkeit der Lösung mit den Kosten C_{Beste}
$R_{All}(G_N)$	All-Terminal-Zuverlässigkeit des Graphen G_N
$\hat{R}_{All}$	Schätzwert für die All-Terminal-Zuverlässigkeit

R_{All}^{max}	Maximale All-Terminal-Zuverlässigkeit der Lösungen einer Population am Ende eines GA-Laufes oder in einer Menge von Lösungen
$R_{All}^{min}(P_t)$	Minimale All-Terminal-Zuverlässigkeit der Lösungen einer Population P_t
R_{All}^{opt}	All-Terminal-Zuverlässigkeit der besten Lösung aus allen Experimenten
$R_{All\mathcal{Y}}^{max}$	Maximale All-Terminal-Zuverlässigkeit der Lösungen einer Lösungsmenge $\mathcal{Y}$
$R_{All\mathcal{Y}}^{min}$	Minimale All-Terminal-Zuverlässigkeit der Lösungen einer Lösungsmenge $\mathcal{Y}$
$rang_{y_i}$	Rang einer Lösung y_i
$r(e_i)$	Zuverlässigkeit einer Kante e_i
R_k	k-Terminal-Zuverlässigkeit
$r(l_k(e_{v_i,v_j}))$	Zuverlässigkeit einer Kante e_{v_i,v_j} bei Wahl der Technologieoption l_k
$R_{obereSchranke}$	Maximale All-Terminal-Zuverlässigkeit
S	Menge aller Realisierungen s
sel	Parameter zur Steuerung des Selektionsdruckes bei rangbasierter Selektion
s_i	Vektor mit einer konkreten Realisierung sämtlicher Kantenzustände $state_e$ eines Netzwerkes
S_i	Zustand eines Systems/Graphen
σ	Standardabweichung
σ_i	Strategievariable, Anpassungsparameter für die Evolutionstrategie
$\overline{\sigma}$	Vektor der Strategievariablen für die Evolutionstrategie
$sol(prob)$	Menge von Lösungen für $prob$
$S(prob)$	Suchraum für $prob$
$state_{e_i}$	Zustand einer Kante e_i
$\#Steps$	Durchschnittliche Anzahl der lokalen Suchschritte
$\mathcal{S}_{y_i}$	Menge der von y_i dominierten Lösungen
T	Spannbaum

t_{conv}	Laufzeit
ϑ_i	Zielfunktionsgewicht für die Zielfunktion f_i
Tr	Menge der Transition
tr_i	Transition für einen Zustandsübergang
$\underline{T}$	Disjunkte Komplementärereignisse zu T
U	Menge der Umweltzustände
$\|V\|$	Anzahl der Knoten eines Graphen
V	Menge der Knoten eines Graphen
v	Knoten
V_C	Schnitt eines Graphen
v_i	Beliebiger Knoten
V_N	Menge des Knoten eines Netzwerkes N
$w_{C(G_N)}^{y_i}$	Gewicht für die Kostenzielfunktion einer Lösung y_i bei der Berechnung des skalaren Zielfunktionswertes
w_C	Gewicht für die Kostenzielfunktion bei der Berechnung des skalaren Zielfunktionswertes
w_j	Gewichte zur Normalisierung der Zielfunktionswerte f_j bei der Berechnung von $d(y', y)$
$w_{R_{All}(G_N)^{y_i}}$	Gewicht für den All-Terminal-Zuverlässigkeits-Zielfunktionswert einer Lösung y_i zur Berechnung eines skalaren Zielfunktionswertes
$w_{R_{All}}$	Gewicht für den All-Terminal-Zuverlässigkeits-Zielfunktionswert zur Berechnung eines skalaren Zielfunktionswertes
X	Konfiguration mit einer konkreten Realisierung sämtlicher x_i
x_i	Entscheidungsvariable
x_i^t	Entscheidungsvariable x_i zum Zeitpunkt t
$\mathcal{Y}$	Menge pareto-optimaler Lösungen
$\|\mathcal{Y}_i\|$	Anzahl der Lösungen einer Pareto-Front $\mathcal{Y}_i$
y	Lösung (Konfiguration) einer Instanz eines berechenbaren Problems
$y(x_i)$	Entscheidungsvariable x_i einer Lösung y

y_{best}	Lösung mit dem besten Fitnesswert
y_i	Lösung i aus einer Lösungsmenge
$\mathcal{Y}_{known}$	Pareto-Lösungsfront eines multikriteriellen Verfahrens
$\mathcal{Y}^*_{known}$	Aggregierte Pareto-Front eines multikriteriellen Verfahrens
y_{Nach}	Durch den Zug $z(y)$ generierte Lösung in der Nachbarschaft von y
y_{N_i}	Durch die Rekombination erzeugte Lösung
$\hat{y}^t$	In einer Iteration einer Evolutionsstrategie generierte neue Lösung
y^*	Optimale Lösung einer Instanz eines berechenbaren Problems
y^t_n	Lösung n einer Instanz eines berechenbaren Problems zum Zeitpunkt t
$\mathcal{Y}_{true}$	Wahre/optimale Pareto-Front
y_{work}	Aktuelle Lösung bei der lokalen Suche in COMNETEA und TPNDA
$\mathcal{Z}(y, Nach(y))$	Menge aller möglichen Züge zur Erreichung einer Lösung in $Nach(y)$
Z	Zustand eines Knotens
z	Zug aus $\mathcal{Z}(y, Nach(y))$
z_i	Normalverteilte Zufallszahl zur Generierung neuer Lösung einer Evolutionsstrategie
$Ziele$	Menge der Zielgrößen zur Beurteilung einer Entscheidung
$ziel_i$	Zielgröße zur Beurteilung einer Entscheidung

Abkürzungsverzeichnis

BaBORNet	Baldwinian Based Optimizer for Reliable Network Design Problems
LaBORNet	Lamarckian Based Optimizer for Reliable Network Design Problems
AS	Anwendungssystem
AVG	Average, Mittelwert
COMNETEA	COMmunication NEtwork Topology dEsign Algorithm
CURE	Cut based Repair Heuristic
EA	Evolutionäre Algorithmen
EDI	Electronic Data Interchange
EP	Evolutionäres Programmieren
ES	Evolutionsstrategie
GA	Genetischer Algorithmus
GALib	Genetic Algorithm Library
GP	Genetisches Programmieren
IT	Informationstechnologie
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
IV	Informationsverarbeitung
M-PAES	Memetic-Pareto Archived Evolution Strategy
MC	Monte-Carlo
MOGA	Multi-objective Genetic Algorithm
MOGLS	Multi-objective Genetic Local Search
MOKP	Multikriterielles kombinatorisches Optimierungsproblem
MOMHLib++	Multiple Objective MetaHeuristics Library in C++
MOSA	Multiple-objective Simulated Annealing

MPI	Message Passing Interface
MTBF	Mean Time Between Failures
MTTR	Mean Time To Repair
NFL	No-Free Lunch
NPGA	Niched pareto Genetic Algorithm
NSGA-II	Non-dominated Sorting Genetic Algortihm II
ONVG	Overall Non-dominated Vector Generation
ONVGR	Overall Non-dominated Vector Generation Ratio
OTNVG	Overall True Non-dominated Vector Generation
PPS	Produktionsplanung- und Steuerung
PSA	Pareto Simulated Annealing
SA	Simulated Annealing
SAGA	Self-Adapting Genetic Algorithm
SMOSA	Serafinis Multiple-objective Simulated Annealing
SPEA	Strength Pareto Evolutionary Algorithm
SPEA-2	Strength Pareto Evolutionary Algorithm 2
STC	Spanning Tree Counting
Std.Abw.	Standardabweichung
TPNDA	Two Phase Network Design Algorithm
VEGA	Vector Evaluated Genetic Algorithms
XML	eXtensible Markup Language

1 Einleitung

1.1 Gegenstand der Arbeit

In den letzten 10 Jahren hat insbesondere der Bereich des elektronischen, zwischenbetrieblichen Datenaustauschs zunehmend an Bedeutung gewonnen [175]. Durch den Einsatz von neuen Datenübertragungsformaten und -standards gehen die Möglichkeiten der Integration und Automatisierung bei der Kopplung verschiedener Informationssysteme heute weit über die bisheriger EDI-Ansätze hinaus [5]. Zentrale Forschungsfragen der Wirtschaftsinformatik sind heute unter anderem der Einsatz elektronischer Marktplätze ([153, 42, 105]) sowie Supply-Chain-Management Konzepte ([16, 183, 28]) und die Vorteile, die sich aus der Kombination beider Konzepte im Rahmen der zwischenbetrieblichen Kommunikation ergeben. Der Einsatz dieser Konzepte setzt die Installation und den Betrieb von hochgradig vernetzten Informationssystemen für einen inner- und überbetrieblichen Datenaustausch voraus. Bei der Planung von geeigneten Kommunikationsinfrastrukturen rückt damit unter anderem die Forderung nach Kommunikationsnetzwerken mit einer sehr hohen Verfügbarkeit und damit verbundenen geringen Ausfallzeiten in den Vordergrund. Parallel dazu werden an den Netzwerkplaner die Forderungen nach wirtschaftlichen Netzwerktopologien gestellt. Aufgrund der Komplexität der zu planenden Netzwerke ist der Einsatz von Entscheidungsunterstützungsystemen notwendig.
Im Rahmen dieser Arbeit werden Methoden für die Entscheidungsunterstützung zur Planung von Kommunikationsnetzwerken unter Kosten- und Zuverlässigkeitsaspekten untersucht. Als Planungsziele werden die Fixkosten, die mit der Installation einer Netzwerktopologie entstehen und die Zuverlässigkeit der entworfenen Kommunikationsinfrastruktur betrachtet. Als Maß für die Gesamtzuverlässigkeit eines Kommunikationsnetzwerkes wird die All-Terminal-Zuverlässigkeit verwendet. Dieses Maß berechnet sich aus den vorgegebenen Zuverlässigkeiten der einzelnen Verbindungen, welche im Netzwerk eingesetzt werden, und gibt die Wahrscheinlichkeit dafür an, dass sämtliche Knoten des Netzwerkes miteinander kommunizieren können. Die Arbeit betrachtet und erweitert bestehende Verfahren zur Berechnung der All-Terminal-Zuverlässigkeit und untersucht diese in einer empirischen Studie.
Im Rahmen der Arbeit werden Planungsprobleme betrachtet, bei denen der Entscheider für eine Menge an fest vorgegebenen Kommunikationspunkten aus einer Menge an zur Verfügung stehenden Kommunikationsverbindungen eine Teilmenge an Verbindungen auswählt, die sämtliche Kommunikationspunkte miteinander verbindet. Es werden zum einen Planungsprobleme betrachtet, bei denen sämtliche zur Verfügung stehenden Verbindungen die gleiche Zuverlässigkeit und entferungsabhängigen Kosten besitzen. Zum

anderen werden mit der Arbeit Planungsprobleme untersucht, bei denen der Entscheider pro Verbindung aus einer Menge unterschiedlicher Technologieoptionen, welche sich in der Zuverlässigkeit und den Kosten unterscheiden, auswählen kann.
Die vorliegende Arbeit untersucht Methoden zur Entscheidungsunterstützung,

- die eine Planung von kostenminimalen Netzwerktopologien unter Beachtung einer Zuverlässigkeitsanforderung ermöglichen.

- die die All-Terminal-Zuverlässigkeit einer Netzwerktopologie maximieren und dabei ein maximal zur Verfügung stehendes Budget für die Installationkosten nicht überschreiten.

- die eine parallele Planung mit den Zielen Zuverlässigkeit und Kosten in Form eines multikriteriellen Planungsproblems ermöglichen.

1.2 Motivation und Zielsetzung

Das hier untersuchte Planungsproblem wurde in der Vergangenheit häufig mit dem Ziel des Findens einer Netzwerktopologie mit minimalen Kosten bei Einhaltung einer geforderten Zuverlässigkeit als Nebenbedingung betrachtet. Als Planungsverfahren wurden hierfür häufig genetische Algorithmen vorgeschlagen. Defizite bisheriger Ansätze bestehen in der Berücksichtigung der Zuverlässigkeitsnebenbedingung im Planungsverfahren. Bisherige Arbeiten verwenden entweder nur sehr einfache Zuverlässigkeitsmaße [69] oder propagieren den Einsatz von Straftermen [176, 52, 50]. Es konnte in [159] gezeigt werden, dass der Einsatz von Straftermen einen genetischen Algorithmus mit unzulässigen Lösungen konvergieren lässt. Diese Arbeit schlägt daher den Einsatz anderer Methoden (insbesondere Reparaturheuristiken) für evolutionäre Planungsverfahren vor, die mittels einer problemspezifischen Heuristik ungültige Lösungen während des Suchprozesses in gültige Lösungen überführen und dabei zielgerichtet das Planungsverfahren in Bereiche des Lösungsraums mit einer hohen Lösungsgüte steuern. Unter Verwendung unterschiedlicher Reparaturkonzepte werden neue Planungsverfahren entwickelt, welche als Verfahren für die Entscheidungsunterstützung bei der Planung von Kommunikationsnetzwerken bisherigen Ansätzen hinsichtlich der Lösungsqualität überlegen sind.
Während das Problem als monokriterielles Optimierungsproblem mit verschiedenen Variationen bereits seit vielen Jahren intensiv erforscht wird, liegen für die Betrachtung als multikriterielles Optimierungsproblem erst wenige Arbeiten [63, 14, 112] vor. Durch die Anwendung und Anpassung von State-of-the-Art-Verfahren auf das im Rahmen der Arbeit untersuchte Entscheidungsproblem mit seinen Spezifika leistet die Arbeit deshalb einen Beitrag dazu, die Forschungsarbeiten auf dem Gebiet der multikriteriellen Planung von Kommunikationsnetzwerken mit den Zielen Kosten und Zuverlässigkeit voranzubringen und für die Praxis Verfahren bereitzustellen, die bisherigen Ansätzen in der Qualität der Lösung überlegen sind.

1.3 Aufbau der Arbeit

Die vorliegende Arbeit ist in acht Kapitel unterteilt. Mit Kapitel 2 erfolgt eine Einordnung des im Rahmen der Arbeit untersuchten Entscheidungsproblems als Gegenstand der Betriebswirtschaftslehre und als Untersuchungsgegenstand der Wirtschaftsinformatik. Hierfür werden die Grundlagen der betriebswirtschaftlichen Entscheidungsprobleme eingeführt und aufbauend darauf Lösungen für die Entscheidungsunterstützung betrachtet. Das Kapitel stellt dabei insbesondere die Planung von ausfallsicheren und ökonomischen Netzwerken als Aufgabe des IT-Sicherheitsmanagements und damit als Aufgabe der Wirtschaftsinformatik heraus.
Die Grundlagen kombinatorischer Optimierungsprobleme mit einer sowie mehreren Zielgrößen, als welches das im Rahmen der Arbeit untersuchte Entscheidungsproblem modelliert wird, werden in Kapitel 3 betrachtet. Es erfolgt zunächst eine allgemeine Einführung in die Problemklasse der kombinatorischen Optimierungsprobleme, gefolgt von einer Übersicht zu unterschiedlichen Lösungsansätzen. Das Kapitel stellt detailliert unterschiedliche evolutionäre Verfahren mit ihren charakteristischen Eigenschaften vor. Das Kapitel schließt mit einer Betrachtung zu bisherigen Arbeiten auf dem Gebiet der Planung von ausfallsicheren und zuverlässigen Kommunikationsnetzwerken.
Unterschiedliche Zuverlässigkeitsmaße und Verfahren zu deren Berechnung werden mit Kapitel 4 eingeführt. Im Rahmen einer empirischen Studie werden ausgewählte Verfahren gegenübergestellt und anhand der dabei gewonnenen Ergebnisse Schlussfolgerungen für den Einsatz der untersuchten Berechnungsmethoden in den zu entwerfenden Planungsverfahren zur Entscheidungsunterstützung abgeleitet. Zusätzlich führt Kapitel 4 sämtliche graphentheoretischen Grundlagen, die für das Verständnis der Arbeit notwendig sind, ein.
Im Mittelpunkt von Kapitel 5 steht die Planung von zuverlässigen und ökonomischen Netzwerktopologien unter Beachtung von Zuverlässigkeitsrestriktionen. Hier werden Ansätze vorgestellt, die in Problemstellungen Anwendung finden, in denen eine Netzwerktopologie mit minimalen Kosten zu finden ist, deren Zuverlässigkeit eine zuvor aufgestellte, minimale Zuverlässigkeitsanforderung nicht unterschreitet. Mit dem Kapitel werden Topologieentwurfsverfahren für die Planung bei identischen Zuverlässigkeiten für sämtliche Verbindungen im Netzwerk sowie Topologieentwurfsverfahren für die Planung mit unterschiedlichen Technologieoptionen je Netzwerkverbindung vorgeschlagen.
Planungsverfahren für den Entwurf von zuverlässigen Kommunikationsnetzwerken unter Budgetvorgaben stellt Kapitel 6 vor. Hier werden Verfahren vorgeschlagen, die es erlauben, Kommunikationsnetzwerke mit einer maximalen Zuverlässigkeit unter Berücksichtigung eines zur Verfügung stehenden Budgets zu entwerfen. Das Kapitel betrachtet dabei wie zuvor auch Kapitel 5 Problemstellungen mit identischen Zuverlässigkeiten für sämtliche Kommunikationsverbindungen sowie Verfahren für die Planung bei unterschiedlichen Technologieoptionen für eine Kommunikationsverbindung, welche sich hinsichtlich Zuverlässigkeit und Kosten unterscheiden.

Eine parallele Betrachtung der Planungsziele Kosten und Zuverlässigkeit als multikriterielles Optimierungsproblem erfolgt mit Kapitel 7. Aufbauend auf den in der Literatur vorgeschlagenen Verfahren für die multikriterielle Optimierung werden zwei Planungsverfahren, welche eine parallele Planung mit beiden Zielen ermöglichen, eingeführt und in einer experimentellen Studie gegenübergestellt.
Die Arbeit schließt mit einer Zusammenfassung und einem Ausblick.

2 Kommunikationsnetzwerkplanung als Planungsproblem der Wirtschaftsinformatik

Mit diesem Kapitel erfolgt eine Einordnung der Kommunikationsnetzwerkplanung als Betrachtungsgegenstand der Wirtschaftsinformatik. Eingangs findet eine Betrachtung des Untersuchungsgegenstandes der Arbeit aus betriebswirtschaftlicher Sicht statt. Hierfür wird eine Übersicht zur Entscheidungstheorie gegeben. Anschließend erfolgt eine Einordnung der im Rahmen der Arbeit untersuchten Problemstellung. Im Anschluss daran wird das Modell der integrierten Informationsverarbeitung nach Mertens [138, 139] vorgestellt, das die wesentliche Grundlage für die Gestaltung von Informationssystemen auf operativer und strategischer Ebene bildet. Die Umsetzung dieses Modells der integrierten Informationsverarbeitung erfordert ein gesamtheitliches Informationsmanagement über alle betrieblichen Aufgabenbereiche. Die Arbeit fokussiert dabei auf den Bereich des Kommunikationsmanagements als strategisches Element eines gesamtheitlichen Informationsmanagements. Das Kapitel führt in die Aufgaben des Kommunikationsmanagements ein. Insbesondere wird auf das IT-Sicherheitsmanagement als eine Teilaufgabe des Kommunikationsmanagements eingegangen, welches eine Schlüsselrolle bei der Gestaltung inner- und überbetrieblicher, elektronischer Geschäftsprozesse einnimmt. Mit Abschnitt 2.3.3 werden die Anforderungen an das IT-Sicherheitsmanagement dargestellt und der Teilaspekt Ausfallsicherheit als ein Planungsziel herausgestellt. Abschließend ordnet Abschnitt 2.3.4 die Kommunikationsnetzwerkplanung mit den Zielen Kosten und Zuverlässigkeit als Teilaufgabe eines ganzheitlichen Informationsmanagements ein. Das Kapitel schließt mit einer Zusammenfassung.

2.1 Entscheidungstheorie im Kontext der Betriebswirtschaftslehre

Die Wissenschaft der Wirtschaftsinformatik beschäftigt sich mit der Gestaltung rechnergestützter Informationssysteme in der Wirtschaft. Sie versteht sich als interdisziplinäres Fach zwischen Betriebswirtschaftslehre und Informatik [79, S. 22]. Mit diesem Abschnitt wird ein Überblick zur untersuchten Problemstellung im Kontext der betriebswirtschaftlichen Entscheidungstheorie gegeben.
In vielen alltäglichen Situationen muss ein Mensch individuell oder als Mitglied einer Gruppe Entscheidungen treffen. Die Folgen einer solchen Entscheidung können nachhaltig die strategische Ausrichtung von betrieblichen Abläufen sowie privaten Belangen beeinflussen. Viele Entscheidungen wie z. B. die Errichtung eines neuen Betriebsstandortes oder die Anschaffung einer neuen Maschine müssen sorgfältig geplant und ausgeführt werden. Die Entscheidungstheorie stellt Methoden und Werkzeuge bereit, um interdisziplinär Entscheidungsprobleme zu erfassen und diese zu lösen.

2.1.1 Gegenstand der Entscheidungstheorie

Als Entscheidung bezeichnet man die bewusste Auswahl einer von mehreren möglichen Handlungsalternativen [126, S. 1]. Ein Entscheidungsproblem liegt vor, wenn in einer Situation mindestens zwei unterschiedliche Handlungsalternativen zur Auswahl stehen, die unterschiedliche Auswirkungen auf das betrachtete System haben [13, S. 22]. Die Aufgaben der Entscheidungstheorie sind nach Bamberg und Coenenberg [13, S. 1] logische und empirische Analysen über rationales oder intendiert rationales Entscheidungsverhalten. Aus dieser Definition ergeben sich die beiden zentralen Forschungsgebiete der Entscheidungstheorie. Zum einen ist dies die normative (oder präskriptive) Entscheidungstheorie, die Methoden bereitstellt, mit deren Hilfe Entscheidungen rational getroffen werden können. Dadurch werden dem Entscheider Ratschläge gegeben, welche Entscheidung er in bestimmten Situationen treffen soll. Dem gegenüber steht die deskriptive Entscheidungstheorie, welche erklärt, wie eine Entscheidung getroffen wurde. Durch die Beobachtung des Entscheidungsprozesses versucht die deskriptive Entscheidungstheorie eine Entscheidung zu beschreiben und auf Basis der Beobachtungen Prognosen für künftige, ähnlich gelagerte Entscheidungen zu ermöglichen.

2.1.2 Entscheidungstheorie als wesentliche Grundlage der Betriebswirtschaftslehre

Die Betriebswirtschaftslehre wird heutzutage als angewandte Wissenschaft betrachtet, deren Aufgabe es ist, die in einer Organisation tätigen Menschen bei ihren Entscheidungen sowie den Gesetzgeber bei der Konzeption unternehmensrelevanter Gesetze beratend zu unterstützen [13, S. 11]. Die Betriebswirtschaftslehre wird daher in der Literatur (vgl. [13, 126]) häufig als spezielle Entscheidungstheorie bezeichnet. Das Ziel der Betriebswirtschaftslehre ist es, mit Hilfe von normativen Entscheidungsmodellen eine rationale Beurteilung von Handlungsalternativen für praktische Problemstellungen zu ermöglichen. In der betriebswirtschaftlichen Entscheidungstheorie werden beide Teilbereiche der Entscheidungstheorie vereint. Die präskriptive Entscheidungstheorie stellt die Grundlagen bereit, um einen betriebswirtschaftlichen Entscheidungsprozess zu analysieren und basierend auf den Beobachtungen Prognosemodelle zu entwickeln. Auf Basis der hierdurch gewonnenen Erkenntnisse ist die Erstellung von normativen Entscheidungsmodellen möglich, die eine rationale Bewertung von Handlungsalternativen zur Zielgrößenmaximierung/-minimierung ermöglichen gestatten.

2.1.3 Das Grundmodell der betriebswirtschaftlichen Entscheidungstheorie

Das Grundmodell der betriebswirtschaftlichen Entscheidungstheorie, im Folgenden als Entscheidungsmodell bezeichnet, führt unabhängig von der Besonderheit des einzelnen Entscheidungsproblems alle Entscheidungstatbestände prinzipiell auf eine gemeinsame Grundstruktur zurück. Bretzke [24, S. 8] beschreibt das Entscheidungsmodell als „... das Ergebnis eines Versuches, die für wesentlich gehaltenen Elemente und Beziehungen einer als Problem empfundenen Handlungssituation in einer formalisierten Sprache so

zu definieren, dass aus dem resultierenden Strukturkomplex die Problemlösung als logische Implikation abgeleitet werden kann“. Das Entscheidungsmodell setzt sich dabei aus den in Abbildung 2.1 dargestellten Teilelementen Zielsystem und Entscheidungsfeld zusammen.
Unter dem Entscheidungsfeld werden die Handlungsalternativen des Entscheiders, die Ergebnisse als Konsequenzen der Alternativen sowie die vom Entscheider nicht beeinflussbaren Umweltzustände zusammengefasst.

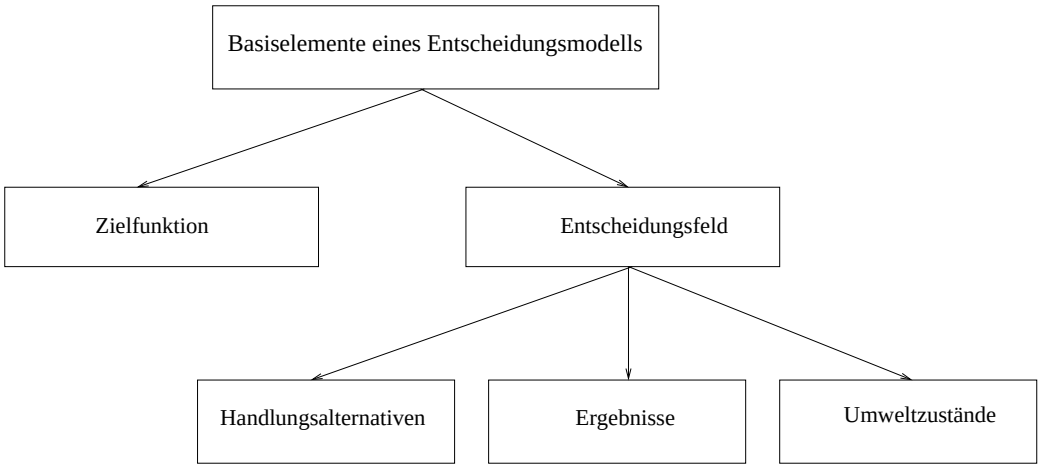


Abbildung 2.1: Basiselemente des normativen Entscheidungsmodells nach [126, S. 20]

Jede der Handlungsalternativen $HA = \{ha_1, \ldots, ha_n\}$ besitzt eine Menge von Merkmalen, auf deren Ausprägung der Entscheider direkt Einfluss nehmen kann. Die Handlungsalternativen unterscheiden sich hinsichtlich der Ausprägung der einzelnen Merkmale. Jede Handlungsalternative ha_i hat direkten Einfluss auf die Zielgrößen des Entscheidungsproblems. Zur objektiven Beurteilung einer Entscheidung ist die Identifikation der Zielgrößen $Ziele = \{ziel_1, \ldots, ziel_n\}$ für das Entscheidungsproblem notwendig. Über die Ereignisse werden die Zustände dieser Zielgrößen im Entscheidungsmodell abgebildet. Die Wahl einer Handlungsalternative ha_i resultiert in einer konkreten Realisierung der Zielgrößen $Ziele$. Zusätzlich zu den durch den Entscheider direkt beeinflussbaren Zuständen des Entscheidungsproblems (abgebildet durch die Handlungsalternativen) werden die Zielgrößen bei der Wahl einer Handlungsalternative durch für den Entscheider nicht steuerbare Merkmale (wie z. B. die Anzahl der Sonnentage, Marktsituation) beeinflusst. All diese Merkmale werden im Entscheidungsmodell mittels der Umweltzustände U abgebildet.
Mit der Zielfunktion wird jeder Handlungsalternative und den daraus resultierenden Zielgrößenzuständen ein konkreter Zielfunktionswert zugewiesen, so dass eine rationale Entscheidungsfindung möglich ist [126, S.23ff]. Die Zielfunktion spiegelt dabei die individuellen Präferenzen des Entscheiders wider. Wird jede Handlungsalternative lediglich hinsichtlich eines Zieles bewertet, so ist das Finden einer Zielfunktion häufig einfach, da es hier lediglich gilt, den Wert des Zieles zu minimieren bzw. zu maximieren. Betriebswirtschaftliche Entscheidungsprobleme weisen jedoch häufig eine Vielzahl von Zielen auf, die mitunter zueinander konfliktär sind [160, S.133]. Zum Finden einer rationalen

Entscheidung ist es daher notwendig, dass der Entscheider jedem einzelnen Ziel eine individuelle Präferenz zuordnet oder dieses als multikriterielles Entscheidungsproblem löst [13, S. 45ff].

2.1.4 Einordnung der Kommunikationsnetzwerkplanung als betriebswirtschaftliches Entscheidungsproblem

Kommunikationsnetzwerke bilden das Rückgrat für eine Vielzahl von elektronischen Geschäftsprozessen. Ohne den Einsatz von vernetzten Systemen wäre die Umsetzung von unternehmensübergreifenden Planungsprozessen, wie sie beispielsweise im Rahmen des Supply-Chain-Managements notwendig sind, nicht denkbar. Die Planung und Erweiterung von Kommunikationsnetzwerken lässt sich anhand der zuvor eingeführten Grundlagen der Entscheidungstheorie als normatives Entscheidungsproblem darstellen. Aufgabe eines Netzwerkplaners in der Praxis ist es, eine vorgegebene Menge von Kommunikationsknoten (wie z. B. Unternehmensstandorte, zentrale Rechenzentren oder kooperierende Unternehmen) mit einer geeigneten Netzinfrastruktur zu verbinden.
Im Rahmen der Kommunikationsnetzwerkplanung muss der Planer dabei Entscheidungen über:

- Standorte (Konzentratoren, zentrale Server),
- Einsatz geeigneter Technologien,
- zeitliche Weiterentwicklungen oder
- Routingverfahren

treffen. Die hier aufgezählten Kriterien stellen dabei die für den Entscheidungsprozess zur Verfügung stehenden Handlungsalternativen dar.
Ziele der Kommunikationsnetzwerkplanung sind (vgl. [49, 135, 136]):

- geringe Installations- und Betriebskosten,
- hohe Zuverlässigkeit,
- geringe Verzögerung sowie
- ausreichende Übertragungskapazitäten für die auftretenden Kommunikationsanforderungen.

Die Aufzählung dieser Planungsziele macht deutlich, dass einige der hier genannten Ziele, wie z. B. geringe Kosten und hohe Zuverlässigkeit zueinander konfliktär sind. Für den Aufbau eines Entscheidungsmodells für die Kommunikationsnetzwerkplanung ist es deshalb notwendig, dass der Entscheider anhand individueller Präferenzen sämtliche Ziele in einer gemeinsamen Zielfunktion zusammenfasst (vgl. [126, S. 25ff]) oder das Entscheidungsproblem mit mehreren Zielen[1] (vgl. [13, S. 48ff]) löst.

[1] Im Folgenden als multikriterielles Entscheidungsproblem bezeichnet.

Wie in Abbildung 2.1 gezeigt, setzt sich das Entscheidungsfeld aus den Handlungsalternativen, den daraus resultierenden Ergebnissen sowie Umweltzuständen zusammen. Das Entscheidungsfeld der Kommunikationsnetzwerkplanung wird, wie viele andere Entscheidungsprobleme auch, durch eine Vielzahl von externen Umweltzuständen begrenzt, die durch den Entscheider nicht beeinflussbar sind. Zu diesen Umweltzuständen zählen unter anderem:

- die technologisch bedingten Übertragungseigenschaften der Verbindungen,
- bautechnische, organisatorische sowie rechtliche Beschränkungen bei der Auswahl von Standorten und
- Ausfälle und Störungen in Verbindungen durch externe Einflüsse (z. B. Zerstörung einer Leitung durch Bauarbeiten).

Im Rahmen dieser Arbeit wird das Kommunikationsnetzwerkplanungsproblem unter Verwendung der Planungsziele Zuverlässigkeit und Kosten untersucht. Mit Kapitel 5 und Kapitel 6 werden Verfahren zur Planung mit einer Zielfunktion vorgestellt. In Kapitel 7 erfolgt eine Betrachtung des Entscheidungsproblems als multikriterielles Planungsproblem. Dabei werden in dieser Arbeit in Anlehnung an andere Arbeiten auf diesem Gebiet [39, S. 17ff][63, 14, 50, 112] die Annahmen getroffen, dass:

- der Planungsprozess eines Backbone-Kommunikationsnetzwerkes mit den Zielen Kosten und Zuverlässigkeit erfolgt,
- die zu verbindenden Standorte fix vorgegeben sind,
- aus einer Menge von zur Verfügung stehenden Verbindungen zur Vernetzung der Standorte eine Teilmenge von Verbindungen ausgewählt wird und
- für die Verbindungen zwischen den Standorten unterschiedliche Technologieoptionen[2] zur Verfügung stehen, die sich hinsichtlich der Kosten und der Zuverlässigkeit unterscheiden.

2.2 Grundlagen der integrierten Informationsverarbeitung

Während Abschnitt 2.1 einen Überblick über das im Rahmen der Arbeit untersuchte Kommunikationsnetzwerkplanungsproblem aus Sicht der betriebswirtschaftlichen Entscheidungstheorie gegeben hat, erfolgt mit diesem Abschnitt eine Betrachtung aus Sicht der Wirtschaftsinformatik. Als Basis hierfür wird das Grundmodell der integrierten Informationsverarbeitung nach Mertens [138, 139] herangezogen und mit Abschnitt 2.2.4 eine Einordnung der Kommunikationsnetzwerkplanung in die integrierte Informationsverarbeitung vorgenommen.

[2] Unterschiede in den Technologieoptionen ergeben sich z. B. durch unterschiedliche Telekommunikationsanbieter, die eine Verbindung zwischen zwei Standorten zu unterschiedlichen Konditionen anbieten.

Viele betriebliche Entscheidungen werden in der heutigen Zeit durch den Einsatz von Anwendungssystemen (AS) unterstützt. Aufgabe der AS ist es dabei, dem Entscheider die Möglichkeit zu geben, den Entscheidungsprozess abzubilden und unterschiedliche Szenarien, die mit der Wahl verschiedener Handlungsalternativen entstehen, objektiv zu bewerten. Erst durch den Einsatz von AS ist es möglich, eine Vielzahl von Handlungsalternativen in kurzer Zeit bezüglich ihrer Auswirkungen auf die Zielfunktion zu bewerten und auf diese Weise dem Entscheider ein geeignetes Werkzeug zur Unterstützung eines Entscheidungsprozesses im Sinne der normativen Entscheidungstheorie bereitzustellen. Auf Basis des von Mertens [138, 139] eingeführten Modells zur integrierten Informationsverarbeitung wird die Unterstützung betriebswirtschaftlicher Entscheidungen im Folgenden betrachtet. Insbesondere erfolgt dabei eine Einordnung von Entscheidungsunterstützungswerkzeugen für das in Abschnitt 2.1.4 eingeführte betriebswirtschaftliche Entscheidungsproblem der Kommunikationsnetzwerkplanung.

2.2.1 Das Grundmodell der integrierten Informationsverarbeitung

Der Begriff Integration lässt sich auf das Lateinische integrale (wiederherstellen, erneuern), integratio (erneuern) sowie integer (ganz, unberührt) zurückführen [77, S.770]. Er ist Ausdruck dafür, einzelne Bestandteile eines komplexen Systems oder Prozesses, die aus sich gegenseitig ergänzenden Teilen bestehen, zu einer ganzheitlichen Einheit zusammenzuführen [84, S. 333]. Heilman [83] stellt den Integrationsbegriff im Kontext der Wirtschaftsinformatik als Zusammenführung von Menschen, Aufgaben und Technik zu einer Einheit heraus. Lehner u. a. [128, S. 133] beschreiben die Wirtschaftsinformatik als Integrationswissenschaft und weisen insbesondere auf die Bedeutung des Themas Integration für die Wirtschaftsinformatik hin. Anhand dieser Aussagen wird deutlich, dass der Integrationsgedanke im Rahmen der Forschungsaufgaben der Wirtschaftsinformatik eine zentrale Stellung einnimmt.
Die Herausforderung in modernen AS besteht darin, Systeme aus unterschiedlichen Planungs- und Steuerungsebenen der betrieblichen Entscheidungsfindung miteinander zu verbinden. Nach Mertens [139, S. 1] sind für eine integrierte Informationsverarbeitung insbesondere die folgenden Anwendungssysteme miteinander zu verbinden:

- Administrationssysteme
- Dispositionssysteme
- Planungssysteme
- Kontrollsysteme

Die Gruppe der Administrations- und Dispositionssysteme wird dabei unter dem Begriff der operativen Systeme zusammengefasst. Eine detaillierte Beschreibung unterschiedlicher Integrationsmethoden findet der interessierte Leser in [139, S. 1ff]. Diese Arbeit gibt lediglich einen Überblick über die Ziele und das Wesen der unterschiedlichen Anwendungssysteme. Abbildung 2.2 stellt das Gesamtkonzept der integrierten Informationsverarbeitung dar.

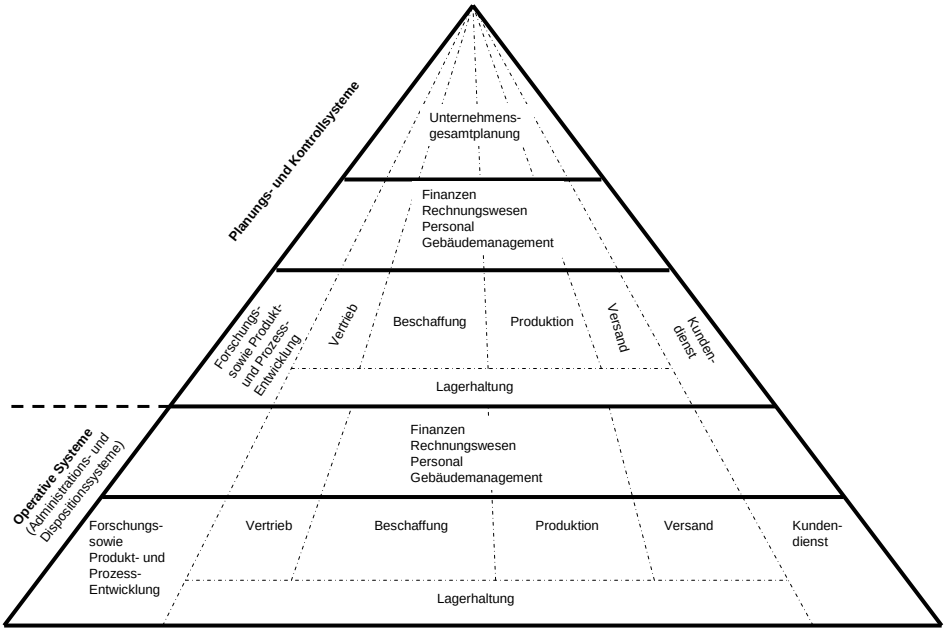


Abbildung 2.2: Gesamtkonzept der integrierten Informationsverarbeitung nach [138, S. 5]

Anhand der Darstellung wird der horizontale Integrationsgedanke der AS entlang der Wertschöpfungskette sowie der vertikale Integrationsgedanke der AS der unteren Ebene (als Datenlieferanten der höheren Ebenen) gut sichtbar. Im Folgenden werden die unterschiedlichen Ebenen hinsichtlich des zeitlichen Horizontes sowie die Komplexität des Entscheidungsproblems betrachtet.

2.2.2 Operative Systeme

Unter den operativen Systemen werden in der Fachsprache der Wirtschaftsinformatik die AS zum Administrieren und Disponieren subsumiert. Einsatzbereich der Administrationssysteme ist die Massendatenverarbeitung, um hierdurch die Rationalisierung der Prozesse, eine Kostensenkung sowie eine Vereinfachung der Arbeitsabläufe durchzuführen [139, S. 12]. Weitreichender als die Administrationssysteme haben Dispositionssysteme die Aufgabe, einen Entscheidungsprozess zu unterstützen oder sogar gänzlich automatisiert durchzuführen. Insbesondere durch den Einsatz von Methoden des Operation Research sowie Verfahren der künstlichen Intelligenz sind die Dispositionssysteme in der Lage, den Entscheidungsprozess unter Verwendung komplexer Optimierungsmodelle zu unterstützen. Die Grundlagen für die Entscheidungen – die Informationen – werden direkt in den operativen Systemen (z. B. PPS-System) erhoben und durch die Administrations- und Dispositionssysteme verarbeitet.

2.2.3 Planungs- und Kontrollsysteme zur Unterstützung von mittel- und langfristigen betriebswirtschaftlichen Entscheidungen

Erst mit dem Einsatz von Administrations- und Dispositionssystemen für die operative Steuerung stehen für die mittelfristige und langfristige Planung Informationen zur Verfügung, die im Rahmen einer integrierten Informationsverarbeitung für den Planungsprozess nutzbar sind. Aufbauend auf den von der operativen Ebene zur Verfügung gestellten Daten ist mit Hilfe der Planungssysteme die Erstellung von Planungsmodellen auf mittel- und langfristiger Ebene möglich. Als strategisches Planungswerkzeug lassen sich die Planungssysteme als Fortführung der in die Dispositionssysteme eingebetteten Modelle auffassen [139, S. 12]. Aufgabe der Kontrollsysteme ist es, die Einhaltung der gesteckten Ziele zu überwachen und gegebenenfalls Vorschläge zur Korrektur bei Nichteinhaltung der Planungsziele zu machen.

2.2.4 Einordnung der Kommunikationsnetzwerkplanung in die integrierte Informationsverarbeitung

Für die in den letzten beiden Abschnitten vorgenommene Untergliederung der AS, die im Rahmen eines integrierten Informationsmanagements zum Einsatz kommen, werden im Folgenden charakteristische Merkmale und deren Ausprägung dargestellt. Die Ge-

Tabelle 2.1: Charakteristika betrieblicher Anwendungssysteme (in Anlehnung an [139, S. 13])

	Operative Systeme	*Planungs- und Kontrollsysteme*
Struktur der Probleme	gut strukturiert	schlecht strukturiert
Häufigkeit der Entscheidungsfindung	in kurzen Abständen	in größeren Zeitabständen
Entscheidungsunterstützung für	mittleres Management	Unternehmensführung
Qualität der Daten für die Planung	zeitnahe, detaillierte Daten aus operativen Datenbanken	verdichtete Informationen (z. B. aus Data Warehouses)
Automatisierungsgrad	Vollautomatisierung möglich	häufig Mensch-Maschine-Dialog notwendig

genüberstellung der operativen Systeme mit den Planungs- und Kontrollsystemen in Tabelle 2.1 macht deutlich, dass beide Systemklassen auf unterschiedlichen zeitlichen Horizonten und mit unterschiedlich verdichteten Informationen arbeiten.
Die vorliegende Arbeit leistet einen Beitrag zur Planung von ökonomischen und zuverlässigen Kommunikationsnetzwerken. Aus der Umsetzung einer ganzheitlichen integrierten Informationsverarbeitung im Unternehmen ergibt sich die Forderung nach der Installation und dem Betrieb eines leistungsfähigen Kommunikationsnetzwerkes. Die Integration verschiedener IT-Systeme aus unterschiedlichen Bereichen der betrieblichen Aktivitäten einer Unternehmung macht den Einsatz von Kommunikationsnetzwerken,

welche einen einfachen und schnellen elektronischen Datenaustausch ermöglichen, notwendig. Gleichzeitig hat in den letzten Jahren eine voranschreitende Diensteintegration (auch als Netzkonvergenz bezeichnet) bei der Verschmelzung von Kommunikations- und Datennetzen stattgefunden, aus welcher sich neue Anwendungsmöglichkeiten für die betriebliche Informationsverarbeitung (IV) ergeben. Die Planung, die Installation und der Betrieb von Kommunikationsnetzwerken für die inner-, zwischen- und überbetriebliche Daten- und Sprachkommunikation muss diesen neuen „Herausforderungen" gerecht werden. In heutigen, modernen IT-Landschaften besteht dabei insbesondere die Forderung nach hochverfügbaren Kommunikationsnetzwerken, durch die eine uneingeschränkte Kommunikation zwischen allen Komponenten (IT-Systeme, End-User-PCs usw.) der betrieblichen Informationsverarbeitung möglich ist. Gleichzeitig unterliegt die Planung eines Kommunikationsnetzwerkes ökonomischen Zwängen, so dass wichtige Bestandteile eines Kommunikationsnetzwerkes (wie z. B. Server oder Konzentratoren) zur Unterstützung einer ganzheitlichen integrierten Informationsverarbeitung nur begrenzt zur Erhöhung der Verfügbarkeit redundant einsetzbar sind. Für den Entscheider (Netzwerkplaner) gilt es, eine Kommunikationsinfrastruktur zu finden, welche die integrierte Informationsverarbeitung ausreichend unterstützt und gleichzeitig ökonomische Restriktionen beim Entwurf und dem Betrieb der Kommunikationsinfrastruktur berücksichtigt. Anhand der in Tabelle 2.1 eingeführten Kriterien erfolgt mit Tabelle 2.2 eine Einordnung des im Rahmen dieser Arbeit untersuchten Netzwerkplanungsproblems (vgl. Abschnitt 2.1.4). Es ist unstrittig, dass es sich beim Entwurf eines Kommunikationsnetz-

Tabelle 2.2: Klassifizierung der Kommunikationsnetzwerkplanung

	Kommunikationsnetzwerkplanung
Struktur der Probleme	in der Regel gut strukturiert
Häufigkeit der Entscheidungsfindung	in größeren Abständen
Entscheidungsunterstützung für	mittleres Management
Qualität der Daten für die Planung	meist liegen detaillierte Daten sowie langfristige konstante und aggregierte Informationen vor
Automatisierungsgrad	Vollautomatisierung möglich

werkes um eine Entscheidung von strategischer Reichweite handelt (vgl. [149, S. 31],[18, S. 75]), die auf viele Bereiche eines Unternehmens Einfluss hat. Im Gegensatz zu anderen strategischen Entscheidungen lässt sich für das Kommunikationsnetzwerkdesign-Entscheidungsproblem feststellen, dass es sich sehr gut strukturieren lässt und somit die Bildung eines auf den Computer übertragbaren Optimierungsmodells gut möglich ist. Bei der Kommunikationsnetzwerkplanung greift der Entscheider auf eine Vielzahl von gut strukturierten Informationen (wie z. B. Preis für Kommunikationsverbindungen, Daten über benötigte Bandbreiten, Verfügbarkeitsanforderungen) zurück. Der Entscheidungshorizont für die Kommunikationsnetzwerkplanung ist als langfristig einzuschätzen. Mit dem Entwurf einer Kommunikationsinfrastruktur legt der Entscheider langfristig die Basis für die Unternehmenskommunikation. Veränderungen dieser Struktur aufgrund

von technologischen Fortschritten, inner- und außerbetrieblichen organisatorischen Veränderungen oder einem wachsenden Bedarf an Information, der mit der bereitgestellten Infrastruktur nicht abgedeckt werden kann, machen eine Erweiterung oder Neuplanung des Kommunikationsnetzwerkes notwendig. Ein aufgrund der zuvor genannten Kriterien angestoßener Entscheidungsprozess ist jedoch ebenfalls als strategische Planungsaufgabe anzusehen. Der Entscheidungsfindungsprozess über die Kommunikationsinfrastruktur ist innerhalb eines Unternehmens in der IT-Abteilung angesiedelt. Als mittel- bis langfristige Planungsaufgabe wird die Kommunikationsnetzwerkplanung als Entscheidungsaufgabe für das mittlere Management (wie z. B. IT-Abteilungsleiter) angesehen. Das Modell der integrierten Informationsverarbeitung macht die breite Durchdringung der elektronischen Datenverarbeitung in fast allen betrieblichen Aufgabenbereichen deutlich. Auf Basis des Informations- und Kommunikationsbedarfs der hier eingesetzten AS stehen dem Entscheider eine Vielzahl von Daten über Kommunikationsflüsse, erforderliche Bandbreiten, maximale Signallaufzeiten und Verfügbarkeitsanforderungen zur Verfügung. All diese Daten können als „Roh"-Daten oder in aggregierter Form in den Entscheidungsprozess einfließen. Aufgrund der guten Strukturierbarkeit und den zur Verfügung stehenden Eingabedaten kann das mit Abschnitt 2.1.4 eingeführte Entscheidungsproblem mit Hilfe eines Entscheidungsunterstützungssystems sehr gut abgebildet und gelöst werden.

2.3 Ganzheitliches Informationsmanagement zur Unterstützung der integrierten Informationsverarbeitung

Anhand des in Abschnitt 2.2.1 vorgestellten Modells der integrierten Informationsverarbeitung wird deutlich, dass der Einsatz von Anwendungssystemen auf operativer sowie taktischer und strategischer Ebene in der Unternehmensplanung sowohl eine horizontale sowie vertikale Integration von Anwendungssystemen und damit verbunden einen Informationsfluss zwischen den unterschiedlichen Anwendungssystemen erfordert. Die Unterstützung dieser Informationsflüsse und -bedürfnisse durch ein ganzheitliches Informationsmanagement wird mit diesem Abschnitt vorgestellt. Es werden zunächst die Aufgaben des Informationsmanagements dargestellt. Für die im Rahmen der Arbeit untersuchten Planungsziele Kosten und Zuverlässigkeit sind insbesondere die Aufgaben des IT-Sicherheitsmanagements von Interesse. Abschnitt 2.3.3 betrachtet dessen Aufgabenbereich.

2.3.1 Die Aufgabenbereiche eines ganzheitlichen Informationsmanagements

Pietsch u. a. [149, S. 30] weisen auf die Bedeutung der Information als geschäftskritischen Faktor hin. Picot u. a. [148, S. 60] schreiben: „Die systematische Planung der Unternehmensressource Information ist damit mindestens ebenso bedeutsam wie die Planung der menschlichen, finanziellen oder materiellen Ressourcen". Es wird deutlich, dass in der Bereitstellung, Verarbeitung sowie Speicherung von Informationen zur Erfüllung der betrieblichen Aufgaben heutzutage eine der großen Herausforderungen in der betrieblichen Informationsverarbeitung zu sehen ist.

Im Allgemeinen werden diese Aufgaben als Untersuchungsgegenstand der Wirtschaftsinformatik unter dem Begriff Informationsmanagement zusammengefasst. Der Begriff des Informationsmanagements wurde in der Literatur in den letzten Jahren sehr intensiv diskutiert und dabei unterschiedlich abgegrenzt (vgl. u. a. [23, 85, 149]). Eine ausführliche Übersicht zu den unterschiedlichen Definitionen bieten Biethahn u. a. [18, S. 18ff]. Im Rahmen dieser Arbeit wird unter dem Begriff des Informationsmanagements (in Anlehnung an Biethahn u. a. [18, S. 18]) „das systematische, methodengestützte Planen, Steuern, Kontrollieren, Koordinieren und Führen der aufeinander abgestimmten Sammlung, Erfassung, Be- und Verarbeitung, Aufbewahrung und Bereitstellung von Informationen sowie die erforderliche Organisation“ verstanden. Im Rahmen eines ganzheitlichen Informationsmanagements stellen Biethahn u. a. [18, S. 28] die Forderung nach einem Informationsmanagement auf, das sich an allen Zielen des Unternehmens orientiert und dabei sämtliche Informationsflüsse organisiert. Pietsch u. a. [149, S.89] stellen das Informationsmanagement gleichfalls als ganzheitliche Managementaufgabe dar, die in allen wesentlichen Bereichen des Unternehmens einen effektiven und effizienten Einsatz des Produktionsfaktors Information ermöglicht. Das gesamtheitliche Konzept des Informationsmanagements erstreckt sich auf die in Pietsch u. a. [149, S. 82] genannten Aufgabenbereiche der betrieblichen IV:

- IV-Planung
- IV-Organisation
- IV-Controlling
- IV-Qualifizierung
- IV-Sicherheit

Aufgabe der *IV-Planung* ist es, Informationssysteme für das Unternehmen effektiv und effizient zu planen und dabei deren optimale Nutzung unter der Beachtung der betrieblichen Anforderungen zu sichern.
Ausgehend von der Komplexität der IV sowie der Vielzahl von AS im Unternehmen ergibt sich die Forderung nach einem eigenständigen IV-Controlling. Das *IV-Controlling* deckt dabei strategische, funktionale, kosten- und zeitmäßige Fragestellungen bei der Planung und dem Einsatz der IV ab ([149, S. 85], [119]).
Die Umsetzung eines ganzheitlichen Informationsmanagements macht die Einbeziehung sämtlicher organisatorischer Einheiten im Unternehmen erforderlich. Zu den Aufgaben der *IV-Organisation* zählt die Aufgabenzuordnung im Rahmen der IV zwischen Fachabteilungen und der zentralen IV-Abteilung sowie die Einbeziehung der Unternehmensführung in IV-Entscheidungen.
Einher mit dem Einsatz der IV geht die Forderung nach der Qualifizierung der Mitarbeiter für den Einsatz der ausgewählten Anwendungssysteme. Es obliegt der *IV-Qualifizierung*, die Aufgabenträger mit einer adäquaten Qualifizierung auszustatten, um eine optimale Nutzung der Anwendungssysteme zu gewährleisten.
In den letzten Jahren hat die Sicherheit der IV immer mehr an Bedeutung gewonnen. Ziel der *IV-Sicherheit* ist es, einen störungsfreien Betrieb der IV zu gewährleisten und

damit u. a. die Akzeptanz der Nutzer während Einführung und Betrieb von Informationssystemen zu erhöhen [118, S. 224].
Die Beschreibung der einzelnen IV-Aufgabenbereiche macht die Interdependenzen zwischen den einzelnen IV-Aufgabenbereichen deutlich. Erst durch die Schaffung eines ganzheitlichen Informationsmanagements, das die Zusammenhänge zwischen den einzelnen Aufgabenbereichen berücksichtigt, kann man der Aufgabe einer optimalen IV-Unterstützung im Unternehmen gerecht werden. Alle fünf hier genannten Aufgabenbereiche bilden gemeinsam den Gegenstandsbereich des Informationsmanagements.

2.3.2 Kommunikationsmanagement als strategische Aufgabe eines ganzheitlichen Informationsmanagements

Die Kommunikation zwischen verschiedenen Anwendungssystemen impliziert den Informationsaustausch zwischen diesen Systemen. Biethahn u. a. [18, S. 75ff] weisen das Kommunikationsmanagement als ein strategisches Element im Rahmen eines ganzheitlichen Informationsmanagements aus. Unter dem Begriff Kommunikationsmanagement wird dabei die Gesamtheit aller organisatorischen und IV-technischen Maßnahmen zur Planung, Steuerung und Kontrolle von Kommunikationsprozessen und -infrastrukturen verstanden. Diese Definition macht deutlich, dass eine exakte Einordnung des Kommunikationsmanagements in eine der fünf in Abschnitt 2.3.1 genannten Aufgabenbereiche der Informationsverarbeitung nicht möglich ist. Vielmehr stellt das Kommunikationsmanagement eine Querschnittsfunktion über alle fünf IV-Aufgabenbereiche dar. Biethahn u. a. nennen als Aufgabenbereiche des Kommunikationsmanagements [18, S. 76]:

- die Identifikation und die Analyse des Kommunikationsbedarfs,
- die Überwachung der Wirtschaftlichkeit der Kommunikationsinfrastruktur,
- die Konzeption und die Bereitstellung der IV-technischen und organisatorischen Kommunikationsinfrastruktur für eine sichere und effiziente Unternehmenskommunikation sowie
- den Betrieb, die Wartung und die Weiterentwicklung der Kommunikationsinfrastruktur.

Mit Tabelle 2.3 erfolgt eine Zuordnung der Aufgabenbereiche des Kommmunikationsmanagements zu den Aufgabenbereichen der betrieblichen IV. Es wird deutlich, dass alle von Biethahn u. a. genannten Aufgabenbereiche des Kommunikationsmanagements sämtliche der von Pietsch u. a. genannten Aufgabenbereiche der IV im Unternehmen abdecken. Für die im Rahmen des Kommunikationsmanagements untersuchten Informationsflüsse und Geschäftsprozesse unterscheidet man die Kategorien (vgl. [18, S. 77ff]):

- innerbetriebliche Kommunikation,
- zwischenbetriebliche Kommunikation und
- kundenorientierte Kommunikation.

Tabelle 2.3: Einordnung der Aufgaben des Kommunikationsmanagements zu den Aufgabenbereichen der Informationsverarbeitung

Aufgaben des Kommunikationsmanagements	*IV-Aufgabenbereich*
Identifikation und Analyse des Bedarfs	IV-Planung
Überwachung der Wirtschaftlichkeit	IV-Controlling
Konzeption und Bereitstellung	IV-Planung, IV-Organisation, IV-Qualifizierung, IV-Sicherheit
Betrieb, Wartung und Weiterentwicklung	IV-Planung, IV-Sicherheit, IV-Controlling

Mit dem bereits eingeführten Modell der integrierten Informationsverarbeitung wird die Notwendigkeit der Planung, Steuerung, Kontrolle und Befriedigung der innerbetrieblichen Informationsbedarfe deutlich. Im Rahmen der zwischenbetrieblichen Kommunikation wurden durch die Wirtschaftsinformatik bereits in der Vergangenheit eine Vielzahl von Konzepten für deren Unterstützung untersucht. In den letzten zehn Jahren hat insbesondere der Bereich des elektronischen zwischenbetrieblichen Datenaustauschs zunehmend an Bedeutung gewonnen [175]. Durch den Einsatz der XML-Technologie gehen die Möglichkeiten der Integration und Automatisierung bei der Kopplung verschiedener Informationssysteme heute weit über die bisheriger EDI-Ansätze hinaus [5]. Zusätzlich hat die rasante Verbreitung der Internettechnologie und deren fast flächendeckende Verfügbarkeit den kostengünstigen Einsatz des elektronischen Datenaustauschs begünstigt. Zentrale Forschungsfragen der Wirtschaftsinformatik sind heute u. a. der Einsatz elektronischer Marktplätze ([42, 105, 153]) sowie Supply-Chain-Management Konzepte ([16, 28, 183]) und die Vorteile, die sich aus der Kombination beider Konzepte (siehe [29]) für die zwischenbetrieblichen Kommunikation ergeben. Mit der direkten Einbeziehung des Kunden in die Informationsverarbeitung des Unternehmens eröffnen sich neue und kostengünstige Absatzwege für Produkte und Dienstleistungen. Im Rahmen einer kundenorientierte Kommunikation bieten Unternehmen ihre Produkte mittels Online-Shops oder über Portalen an. Aufgrund der 7x24 Stunden Verfügbarkeit solcher Angebote eröffnen sich hierdurch neue und flexible Vertriebswege, auf denen neue (bisher vielleicht nicht erreichbare) Kunden angesprochen werden. Gleichzeitig profitiert der Kunde durch den Vertrieb von Produkten über die Internetplattform von kurzen Warendistributionswegen, die in Form eines Preisvorteils bei Online-Kauf an den Kunden weitergegeben werden können. Im Rahmen der kundenorientierten Kommunikation hat zusätzlich das Customer Relationship Management an hoher Bedeutung gewonnen [67]. Unternehmen treten heutzutage auf vielen Wegen (wie z. B. Telefon, E-Mail, Vertreterbesuch etc.) mit ihren Kunden in Kontakt. Durch eine konsequente und vollständige Verarbeitung der im Kundenkontakt gewonnenen Informationen ist es möglich, langfristige und damit profitable Kundenbeziehungen aufzubauen und dabei gezielt Produktentwicklung, Marketing und Vertrieb auf die Kunden (individuell oder als Gruppe) auszurichten.
Mit Hilfe des Kommunikationsmanagements gilt es, eine geeignete Kommunikationsinfrastruktur zu schaffen, die die Anforderungen aller, mittels des Kommunikationsmanagements untersuchten, Informationsflüsse und Geschäftsprozesse abdeckt.

2.3.3 IT-Sicherheitsmanagement im Rahmen eines ganzheitlichen Informationsmanagements

Für viele Unternehmen besteht heute eine große Abhängigkeit von der Zuverlässigkeit und Funktionsfähigkeit der eingesetzten Informations- und Kommunikationssysteme [87, S. 135]. Bereits der Ausfall nur eines Teils der betrieblichen IV kann dazu führen, dass logistische Prozesse nicht mehr abgewickelt werden können, keine Auftragsannahme möglich ist oder ein Stillstand der Produktion eintritt. Neben dem finanziellen Schaden, der durch einen solchen Ausfall entsteht, verursacht ein Ausfall zusätzlich einen nicht messbaren materiell Schaden durch den Imageverlust gegenüber Kunden (vgl. [146]) und anderen Geschäftspartnern.
Schuster und Wilhelm [169] bezeichnen die Beherrschung und Anwendung der Informations- und Kommunikationsinfrastruktur als eine Kernkompetenz eines jeden produzierenden Unternehmens. Im Gegensatz dazu schreiben Böhmann und Krcmar [20], dass IT heutzutage immer mehr als Commodity (Gebrauchsgut, wie z. B. auch Strom) aufgefasst wird, welche auf globalen Dienstleistungsmärkten eingekauft wird. Stannat und Petri stellen in [180] die Ergebnisse einer Untersuchung unter den IT-Verantwortlichen in privatwirtschaftlichen und öffentlichen/kommunalen Unternehmen vor, in der sich beide Auffassungen wiederfinden. Ein Ergebnis der Untersuchung ist, dass sich die IT für einige Unternehmen zu einer Ressource entwickelt, welche am effizientesten als Dienstleistung von spezialisierten Unternehmen eingekauft wird. Als Ursache hierfür werden unter anderem sinkende IT-Budgets, die intensive Nutzung von Standardsoftware sowie die bereits in vielen Bereichen stattgefundene Auslagerung von IT-Infrastrukturleistungen genannt. Gleichzeitig zeigen die Ergebnisse der Befragung, dass einige Unternehmen IT als strategischen Erfolgsfaktor sehen. Stannat und Petri [180] stellen als Ergebnis ihrer Untersuchung für die zukünftige Entwicklung der IT heraus, dass sich die IT-Infrastruktur letztendlich bei allen Unternehmen zu einer Commodity-Ressource entwickeln wird. Ziel wird sein, diese Ressource unter Berücksichtigung der Entscheidungsfaktoren Zuverlässigkeit sowie Funktionsfähigkeit zu einem möglichst kostengünstigen Preis (auf dem Markt oder als interne Dienstleistung) zu beziehen [180].
Unabhängig von der Bereitstellung der IT als Dienstleistung oder als unternehmensinterne Ressource ist im Rahmen einer ganzheitlichen Betrachtung des Informationsmanagements die Entwicklung, die Umsetzung und die Überwachung von speziellen Lösungen des IT-Sicherheitsmanagements notwendig (vgl. hierzu auch [18, S. 85], [149, S. 254]).
Aufgabe des IT-Sicherheitsmanagements ist es, alle Aufgaben und Aktivitäten, die eine Beeinträchtigung der IV darstellen, zu vermeiden beziehungsweise deren Auswirkungen zu reduzieren [18, S. 85].
Ziele für die Erreichung der IT-Sicherheit sind nach ISO/IEC 17799:2000 [95]:

- die Vertraulichkeit,
- die Integrität und
- die Verfügbarkeit.

Häufig werden diese drei Ziele in der Literatur (siehe [149, S. 256],[152, S. 5], [184]) zusätzlich noch um das Ziel der Authentizität erweitert.

Mit dem Ziel *Vertraulichkeit* wird das Interesse der Kommunikationsteilnehmer nach dem Schutz der übertragenen Nachricht gegenüber unberechtigten Dritten gewahrt.
Ziel der *Integritätswahrung* ist es, Manipulationen der Nachricht während der Übertragung zu erkennen und dem Empfänger Methoden zur Prüfung der Unversehrtheit einer Nachricht zur Verfügung zu stellen.
Die *Verfügbarkeit* von Informationen zielt darauf ab, die benötigten Informationen zeitgerecht zur Verfügung zu stellen. Das Ziel Verfügbarkeit ist dabei eng mit dem im Rahmen der Arbeit untersuchten Planungsziel Zuverlässigkeit verknüpft. Die Verfügbarkeit drückt das Verhältnis zwischen der Zeit, in der das betrachtete System fehlerfrei arbeitet, zur betrachteten Gesamtzeit aus. Die Zuverlässigkeit gibt die Wahrscheinlichkeit dafür an, dass das betrachtete System innerhalb eines Zeitraums nicht ausfällt (vgl. Abschnitt 4.1.1 auf S. 65). Anhand der Beschreibungen wird deutlich, dass eine hohe Verfügbarkeit eine hohe Zuverlässigkeit (und umgekehrt) zur Folge hat.
Die Sicherstellung der Echtheit einer Nachrichtenquelle wird mit dem Ziel der *Authentizität* verfolgt. Hier werden Absendern und Empfängern von Nachrichten geeignete Mittel zum Anzeigen und Überprüfen von Nachrichtenquellen geboten. Humpert [91] weist darauf hin, dass durch die Nichterreichung von nur einem dieser Grundziele der IT-Sicherheit bei den meisten Anwendungen ein großer, existenzieller Schaden entstehen kann.
Biethahn u. a. [18, S. 88] ordnen die Aufgabe des IT-Sicherheitsmanagements als strategische Führungsaufgabe ein. Nach Auffassung der Autoren muss sich die Managementebene mit den Chancen und gleichzeitig mit den Risiken, die sich aus dem Einsatz von Informationstechnologie ergeben, beschäftigen. Thiel [184] stellt die Bedeutung eines ganzheitlichen IT-Sicherheitsmanagements heraus, „dessen Aufgabe es ist, die Unternehmensleitung dabei zu unterstützen, das Unternehmen in seiner Gesamtheit so zu gestalten, zu lenken und zu entwickeln, dass es hinsichtlich seiner IT-Risikolage jederzeit unter Kontrolle gehalten werden kann“. Beide Arbeiten werden der auf S. 18 geführten Diskussion zur IT als Commodity-Ressource gerecht. Egal auf welche Weise (ob nun als interne Ressource oder als externe Dienstleistung) die Bereitstellung der IT erfolgt, Zielsetzung des jeweiligen Anbieters der IT muss es sein, im Rahmen eines ganzheitlichen IT-Sicherheitsmanagements die Ziele der IT-Sicherheit zu verfolgen.
Das IT-Sicherheitsmanagement ist nach [18, S. 88] dabei als iterativer Prozess aufzufassen, der sich aus den in Abbildung 2.3 dargestellten Elementen zusammensetzt. Ausgehend von einer Riskoanalyse und den dabei identifizierten Gefahrenpotenzialen

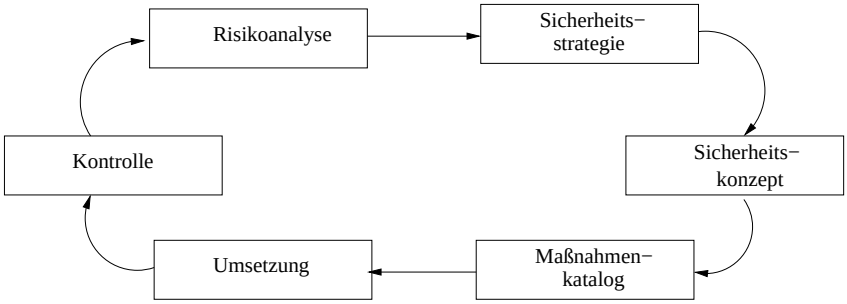


Abbildung 2.3: Managementzyklus des IT-Sicherheitsmanagements nach [18, S. 91]

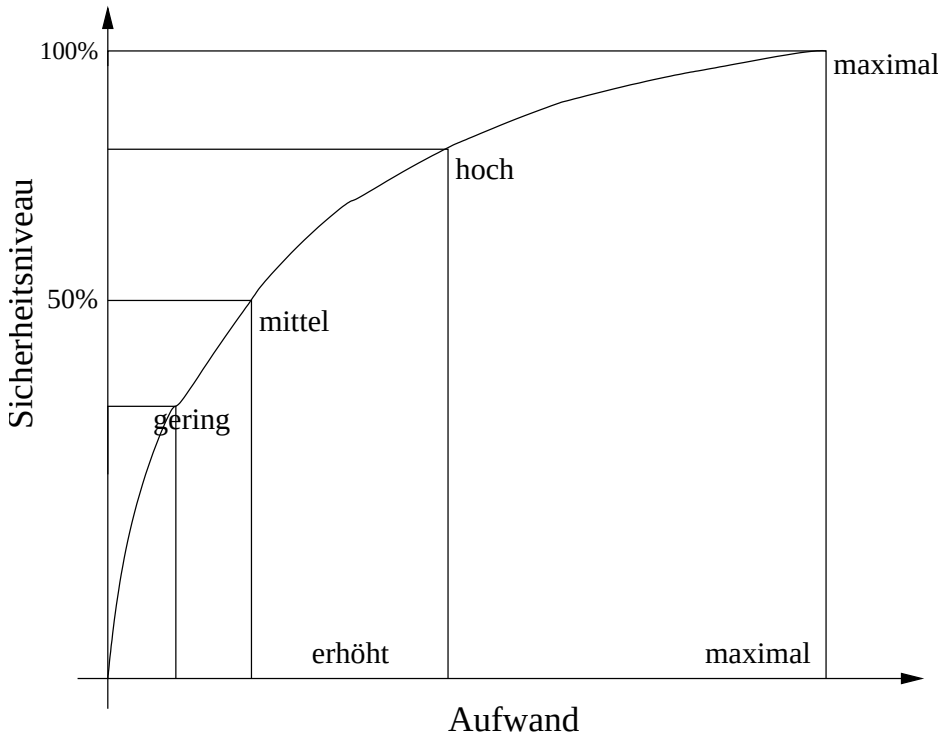


Abbildung 2.4: Zusammenhang Sicherheitsaufwand und Sicherheitsniveau nach [27]

.

muss eine auf die Unternehmensstrategie zugeschnittene Sicherheitsstrategie entworfen werden. Mit dieser werden organisatorische, technische und dokumentatorische Sicherheitsrichtlinien festgelegt. Basierend auf diesen Regeln wird mit einem Sicherheitskonzept festgelegt, wie sicherheitsrelevante IV-Systeme und IV-Infrastrukturen gemäß den IT-Sicherheitszielen zu schützen sind. Abgeleitet daraus stellt der Maßnahmenkatalog sowohl kurz- als auch langfristige Maßnahmen zur Vermeidung von Sicherheitsrisiken dar. Mit der Realisierung dieser Maßnahmen werden die strategischen IT-Sicherheitsziele umgesetzt. Die Kontrolle der IT-Sicherheit dient der Überwachung der Sicherheitsziele.
Im Rahmen dieser Arbeit werden Verfahren vorgeschlagen, die als Entscheidungsunterstützungswerkzeuge die Erstellung und die Umsetzung von IT-Sicherheitskonzepten unter Verwendung der Ziele Verfügbarkeit (Zuverlässigkeit) und Kosten berücksichtigen.

Aufwand-Nutzen-Relation des IT-Sicherheitsmanagements

Für die Umsetzung der Ziele der IT-Sicherheit im Unternehmen ist ein finanzieller Aufwand notwendig. Abbildung 2.4 stellt den Aufwand für das IT-Sicherheitsmanagement dem daraus resultierenden Sicherheitsnutzen gegenüber. In der Abbildung wird auf der Abzisse der erbrachte (finanzielle) Aufwand und auf der Ordinate das damit erzielbare Sicherheitsniveau dargestellt. Die Darstellung macht deutlich, dass bereits mit einem geringen Aufwand ein IT-Sicherheitsgrundschutz (geringes bis mittleres Sicherheitsniveau) realisierbar ist.

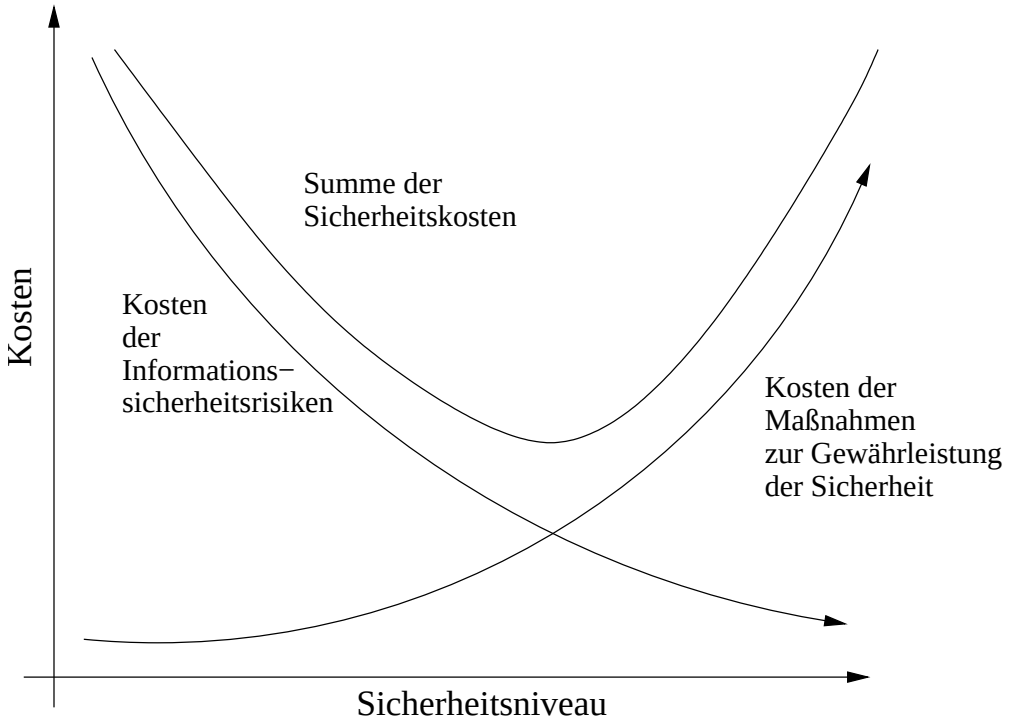


Abbildung 2.5: Zusammenhang Risikoakzeptanz und Risikoreduktion [146]

.

Die dargestellte Kurve zeigt deutlich, dass ein sehr hohes IT-Sicherheitsniveau mit einem sehr hohen Aufwand verbunden ist.
Sowohl Biethahn u. a. [18, S. 89] als auch Pietsch u. a. [149, S. 266] weisen darauf hin, dass einem angestrebten Sicherheitsniveaus ein aus betriebswirtschaftlicher Sicht vertretbarer Aufwand gegenüberstehen muss. Die Abbildung macht deutlich, dass die Erzielung eines Niveaus hundertprozentiger Sicherheit nur mit einem immensen Aufwand erreichbar und in der Regel in der Praxis nicht umsetzbar ist. Aufgabe des IT-Sicherheitsmanagements ist die Bestimmung einer Lösung, in der ein Gleichgewicht zwischen Risikoakzeptanz und Risikoreduktion durch den Einsatz von IT-Sicherheitsmaßnahmen erzielt wird [146].
Betrachtet man den Zusammenhang zwischen dem Sicherheitsaufwand und dem daraus resultierenden Sicherheitsniveau fokussiert auf die im Rahmen dieser Arbeit betrachteten Planungsziele Kosten und Zuverlässigkeit, so wird der mit Abbildung 2.5 dargestellte Zusammenhang ebenfalls deutlich. In der Praxis kann die Zuverlässigkeit einer IT-Infrastruktur durch den Einsatz von redundanten Komponenten (wie z. B. Switches, Servern oder Backup-Verbindungen) erhöht werden. Zwangsläufig resultieren aus dem Einsatz dieser Technologien höhere Kosten für die IT-Infrastruktur. Aufgrund der technischen Gegebenheiten (Zuverlässigkeiten der einzelnen Komponenten) kann dabei eine Zuverlässigkeit von 100 % niemals erreicht werden. Zielsetzung für den Planer sollte daher ein Streben nach einem akzeptablen „Gleichgewicht“ zwischen der Zuverlässigkeit und den Kosten sein.

Abbildung 2.5 veranschaulicht diesen Zusammenhang. Die Abbildung stellt die IT-Sicherheitskosten als Summe der Kosten für die Informationssicherheitsrisiken sowie den Maßnahmen für die IT-Sicherheit dar. Für das betrachtete Planungsproblem bilden die Kosten, welche für die Etablierung und Aufrechterhaltung einer zuverlässigen IT-Infrastruktur entstehen, die Kosten der Maßnahmen für die IT-Sicherheit. Die Kosten, welche durch den Ausfall (Nichtverfügbarkeit) der IT-Infrastruktur entstehen, werden durch die Kosten für die Informationssicherheitsrisiken abgebildet. Es ist erkennbar, dass eine optimale Lösung existiert, in der die Summe der IT-Sicherheitskosten minimal ist. Durch den geeigneten Einsatz von Entscheidungsunterstützungswerkzeugen ist es möglich, einem Entscheider Lösungsbausteine bereitzustellen, die ein optimales Verhältnis zwischen dem aufzubringenden Aufwand und dem damit erzielbaren IT-Sicherheitsniveau ermitteln. Im Rahmen dieser Arbeit werden hierfür Verfahren entwickelt, welche als Entscheidungsunterstützungswerkzeuge das Finden einer solchen Lösung unterstützen.

2.3.4 Kommunikationsnetzwerkplanung als Untersuchungsgegenstand des ganzheitlichen Informationsmanagements

Mit den vorangegangenen Abschnitten wurde das IT-Sicherheitsmanagement als ein Aufgabenbereich eines ganzheitlichen Informationsmanagements dargestellt. Das untersuchte Kommunikationsnetzwerksplanungsproblem lässt sich als Entscheidungsproblem des IT-Sicherheitsmanagements einordnen. Die in dieser Arbeit entwickelten Verfahren leisten einen Beitrag zum Entwurf zuverlässiger und ökonomischer Kommunikationsnetzwerke dar. Die hier eingeführten Verfahren berücksichtigen dabei insbesondere das IT-Sicherheitsziel der Verfügbarkeit. Die Arbeit leistet einen Beitrag dazu, Verfahren und Methoden für Entscheidungsunterstützungswerkzeuge zu entwerfen, mit deren Hilfe die Planung von Kommunikationsnetzen unter Berücksichtigung der gestellten Verfügbarkeitsanforderungen möglich wird.
Durch den Einsatz der vorgeschlagenen Methoden im Bereich der IV-Planung im Rahmen des Kommunikationsmanagements wird der Entwurf von Netzwerktopologien unter Berücksichtigung der Planungsziele Kosten und Zuverlässigkeit möglich. Zusätzlich stellt die vorliegende Arbeit mit Kapitel 4 Verfahren zur Untersuchung der Zuverlässigkeit von Kommunikationsinfrastrukturen vor und schlägt mit den Kapiteln 5–6 Verfahren zur Erhöhung der Ausfallsicherheit von Kommunikationsnetzwerken vor.

2.4 Zusammenfassung

Mit diesem Kapitel wurde ein Überblick über die betriebswirtschaftliche Entscheidungstheorie gegeben. Abschließend wurde die integrierte Informationsverarbeitung sowie das Konzept eines ganzheitlichen Informationsmanagements dargestellt.
Es wurde gezeigt, dass der Untersuchungsgegenstand der Arbeit als normatives Entscheidungsproblem einzuordnen ist. Mit dem Modell der integrierten Informationsverarbeitung wurde aufgezeigt, dass die Verarbeitung von Informationen in fast allen Bereichen einer Unternehmung anzutreffen ist. Anhand des Modells wird deutlich, dass eine Integration von Anwendungssystemen entlang der Wertschöpfungskette und innerhalb der unterschiedlichen betrieblichen Organisationsebenen stattfindet.

Unter Verwendung eines ganzheitlichen Informationsmanagements kann sowohl die Versorgung der innerbetrieblichen Informationssysteme als auch die Anbindung an externe Informationssysteme von Kunden, Kooperationspartnern und Zulieferern geplant, umgesetzt und gesteuert werden. Im Rahmen des ganzheitlichen Informationsmanagements ordnet sich die Arbeit in das Aufgabenfeld des Kommunikations- und IT-Sicherheitsmanagements ein. Mit der Arbeit werden Methoden für Entscheidungsunterstützungssysteme entworfen, die vollautomatisiert oder im Dialog beim Entwurf von ausfallsicheren und ökonomischen Kommunikationsnetzwerken einsetzbar sind und damit den Entscheider bei der Lösung des aufgeworfenen Entscheidungsproblems unterstützen.

3 Ansätze zur Planung von Kommunikationsnetzwerken

Nachdem im vorangegangenen Kapitel die Betrachtung des im Rahmen der Arbeit untersuchten Netzwerktopologieplanungsproblems aus betriebswirtschaftlicher und Wirtschaftsinformatik-Sicht erfolgte, wird in diesem Kapitel eine formale Beschreibung des Problems und eine Einführung in verschiedene Lösungsverfahren vorgenommen.
Mit Abschnitt 3.1 werden zunächst die Grundlagen kombinatorischer Optimierungsprobleme eingeführt. Aufbauend auf diesen Ausführungen wird das untersuchte Planungsproblem als kombinatorisches Optimierungsproblem dargestellt. Im Anschluss daran werden die Grundlagen zur Lösung kombinatorischer, mono- und multikriterieller Optimierungsprobleme eingeführt. Als Lösungsverfahren werden Metaheuristiken vorgeschlagen. Die Arbeit betrachtet dabei fokussiert evolutionäre Algorithmen zur mono- und multikriteriellen Optimierung und stellt diese im Detail vor. Den Abschluss des Kapitels bildet eine Literaturstudie, welche bisherige Planungsansätze bezüglich der im Rahmen der vorliegenden Arbeit untersuchten Ziele Kosten und Zuverlässigkeit betrachtet.

3.1 Grundlagen kombinatorischer Optimierungsprobleme

Bereits in Kapitel 2 wurde darauf hingewiesen, dass in der betrieblichen Praxis eine Reihe von Planungsproblemen existieren, welche sich aufgrund ihrer guten Strukturierbarkeit gut mit einem computergestützten Entscheidungsunterstützungsystem lösen lassen. Mit diesem Abschnitt werden die Grundlagen kombinatorischer Optimierungsprobleme vorgestellt, welche eine formale, abstrakte Beschreibung eines Problems ermöglichen. Die Ausführungen bilden die Grundlage für die Einordnung und Modellierung des untersuchten Kommunikationsnetzwerkplanungsproblems als kombinatorisches Optimierungsproblem, welche mit Abschnitt 3.1.1 erfolgt. Für die Lösung kombinatorischer Optimierungsprobleme werden mit Abschnitt 3.1.2 unterschiedliche Ansätze diskutiert. Die hier durchgeführte Modellierung des im Rahmen der Arbeit untersuchten Kommunikationsnetzwerkplanungsproblems als kombinatorisches Optimierungsproblem bildet dabei die Grundlage für die Erstellung der in Kapitel 5, 6 und 7 vorgeschlagenen (computergestützten) Lösungsverfahren.
Jedes berechenbare Problem $Prob$ hat eine Menge von Instanzen $prob$. Zu jeder Probleminstanz $prob \in Prob$ existiert eine Menge von Antworten $ans(prob)$. Ein Teil dieser Antworten $ans(prob)$ bildet die Menge der (akzeptablen) Lösungen $sol(prob)$ ($sol(prob) \in ans(prob)$) des Problems. Jede Lösung $sol(prob)$ repräsentiert eine Handlungsalternative, welche durch eine konkrete Realisierung sämtlicher Entscheidungsvariablen x_i für $prob$ beschrieben wird. Ein Algorithmus löst ein Problem, wenn er für eine Eingabe $prob \in Prob$ einen Output $y \in sol(prob)$ liefert. Gibt es keine zulässige

Lösung, ist die Outputmenge leer. Das Problem ist ein Suchproblem, wenn die Menge der Lösungen $sol(prob)$ leer ist, eine oder mehrere Lösungen hat.

Definition 3.1 (Kombinatorisches Optimierungsproblem)
Ein kombinatorisches Optimierungsproblem ist ein spezielles Suchproblem, bei dem jedes $prob \in Prob$ *eine endliche Anzahl von Lösungen* $y \in sol(prob)$ *besitzt und es eine Zielfunktion* f *gibt, die jedem* y *einen Zielfunktionswert zuordnet, den es zu minimieren oder maximieren gilt. Das Optimierungsproblem besteht im Finden der optimalen Lösung* $y^* \in sol(prob)$.

Jeder Instanz eines berechenbaren Problems $prob \in Prob$ ist ein Suchraum $S(prob)$ zugeordnet, der folgende Eigenschaften besitzt:

- Jedes Element $y \in S(prob)$ repräsentiert eine Antwort $ans(prob)$.
- Bei einem Optimierungsproblem wird die optimale Lösung $y^* \in sol(prob)$ durch ein Element von $S(prob)$ repräsentiert.

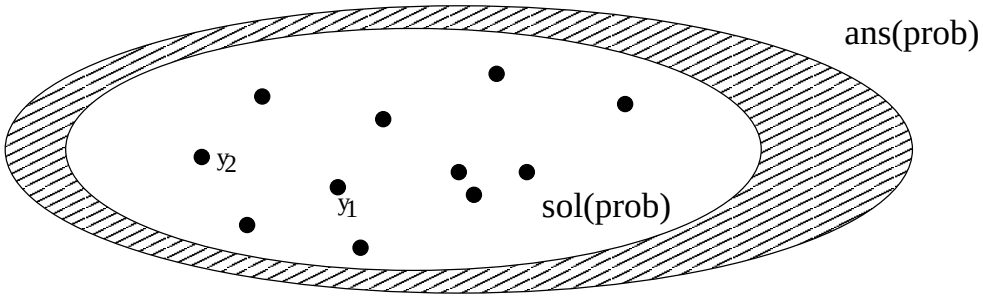


Abbildung 3.1: Grafische Repräsentation eines kombinatorischen Optimierungsproblems

Jedes Element $y \in S(prob)$ wird Konfiguration genannt. Die Menge der Nachbarn $Nach(y) \in S(prob)$ einer Konfiguration y wird definiert als die Menge der Konfigurationen, die durch Anwendung eines elementaren Operators z (Zug) aus der Menge der möglichen Züge $\mathcal{Z}(y, Nach(y))$ entsteht. Die möglichen Züge $\mathcal{Z}$ für die Nachbarschaft $Nach(y)$ sind dabei hinsichtlich ihrer Vorteilhaftigkeit zur Erreichung des Optimierungszieles zu bewerten.
Abbildung 3.1 zeigt eine grafische Repräsentation eines kombinatorischen Optimierungsproblems. Die schraffierte Fläche stellt die Menge aller Antworten ($ans(prob)$) dar. Innerhalb dieser Menge liegt die Menge der gültigen Lösungen ($sol(prob)$). Die gültigen Lösungen (Konfigurationen y) für ein kombinatorisches Optimierungsproblem werden durch die Punkte innerhalb der Lösungsmenge $sol(prob)$ repräsentiert.

Formal lässt sich jedes kombinatorische Optimierungsproblem beschreiben als (vgl. [19]):

- Eine Menge von Entscheidungsvariablen $X = \{x_1, x_2, \ldots, x_n\}$,
- die Variablen-Domains $DOM = \{Dom_1, \ldots, Dom_n\}$,
- die Nebenbedingungen $NB = \{nb_1, \ldots, nb_n\}$ sowie
- eine zu minimierende Zielfunktion[3] f, mit $f : Dom_1 \times \ldots Dom_n \rightarrow \mathbb{R}$.

Eine gültige Konfiguration $y \in S(prob)$ wird dabei durch eine konkrete Realisierung

$$X_y = \{x_1, \ldots, x_n\}, \text{mit } \forall x_i \in Dom_i \text{ und } \forall nb_i : \text{Bedingung ist erfüllt}$$

beschrieben. Unter dem Ziel der Minimierung des Zielfunktionswertes gilt für die optimale Lösung y^*: $\forall y \in sol(prob) : f(y^*) < f(y)$, respektive für ein Maximierungsproblem: $\forall y \in sol(prob) : f(y^*) > f(y)$.

3.1.1 Kommunikationsnetzwerkplanung als kombinatorisches Optimierungsproblem

Die vorliegende Arbeit untersucht die Planung von ökonomischen und zuverlässigen Netzwerken. Mit Abschnitt 2.1.4 auf S. 8 wurden die Planungsziele, die Handlungsalternativen sowie die Umweltzustände, welche bei der Kommunikationsnetzwerkplanung von Relevanz sind, vorgestellt. Es wurde herausgestellt, dass mit der vorliegenden Arbeit die Ziele Kosten und Zuverlässigkeit betrachtet werden. Abschnitt 2.1.4 nennt weiterhin die bei der Betrachtung (in Anlehnung an andere Arbeiten auf dem Gebiet) getroffenen Annahmen für den Untersuchungsgegenstand.
Die Abschnitte 2.1.4, 2.2.4 und 2.3.4 stellen das untersuchte Problem als Entscheidungsproblem aus Sicht der Betriebswirtschaftslehre und der Wirtschaftsinformatik dar. In diesem Abschnitt erfolgt eine Beschreibung des untersuchten Planungsproblems als kombinatorisches Optimierungsproblem.
In Anlehnung an [52, 49, 176, 14, 157, 159, 156, 158] lässt sich das untersuchte Problem der Planung von Kommunikationsnetzwerken unter Kosten- und Zuverlässigkeitsgesichtspunkten ausgehend von Definition 3.1 wie folgt beschreiben:

- Die Menge der Entscheidungsvariablen X sind die Verbindungen zwischen den verschiedenen zu verbindenden Standorten.
- Die Variablen-Domains DOM für diese Entscheidungsvariablen (des im Rahmen der Arbeit untersuchten Problems) sind die für eine Verbindung zur Verfügung stehenden Technologieoptionen, die sich hinsichtlich Zuverlässigkeit und Kosten unterscheiden. Im Rahmen der Arbeit werden Probleme untersucht, bei denen für eine Verbindung genau eine Technologieoption möglich ist, d.h. durch den Entscheider ist nur die Entscheidung zu treffen, ob in der entworfenen Netzwerktopologie eine direkte Verbindung zwischen zwei Knoten etabliert wird. Außerdem

[3] Für die Maximierung der Zielfunktion ist adäquat vorzugehen.

betrachtet die Arbeit Probleme, in denen unterschiedliche Technologieoptionen für eine Verbindung möglich sind. Durch die Technologieoptionen ist es z. B. möglich, Angebote verschiedener Dienstleister, die sich hinsichtlich Kosten und Zuverlässigkeit unterscheiden, abzubilden.

- Als Nebenbedingungen NB werden im Rahmen der Arbeit Zuverlässigkeitsanforderungen sowie Budgets für die entworfene Netzwerktopologie betrachtet.
- Als Zielfunktionen werden die Fixkosten, die mit der Installation einer Netzwerktopologie verbunden sind, sowie die All-Terminal-Zuverlässigkeit[4] der entworfenen Netzwerktopologie verwendet.

In dieser Arbeiten werden dabei die Begriffe Technologie und Technologieoption wie folgt definiert:

Definition 3.2 (Technologie)
Eine Technologie repräsentiert einen bestimmten Verbindungstyp. Im Rahmen der vorliegenden Arbeit werden Technologien verwendet, welche sich in den Eigenschaften Zuverlässigkeit und Kosten unterscheiden.

Unterschiedliche Technologien sind z. B. Glasfaser- und Kupferleitungen oder auch drahtgebundene und drahtlose (wie z. B. Richtfunk) Verbindungen.

Definition 3.3 (Technologieoption)
Eine Technologieoption (kurz als Option bezeichnet) repräsentiert eine für den Aufbau eines Kommunikationsnetzwerkes zur Verfügung stehende Technologie.

In Anlehnung an [39, S. 17ff][14, 50, 112, 63] werden für diese Arbeit die folgenden Annahmen getroffen:

- Die Positionen der zu verbindenden Standorte sind fix vorgegeben,
- sämtliche Knoten im Netzwerk haben eine Verfügbarkeit von 100 %,
- die Kosten und die Zuverlässigkeit der Verbindungen zwischen den Knoten sind bekannt,
- für jede Verbindung stehen $n \geq 1$ Technologieoptionen zur Verfügung, die sich hinsichtlich Installationskosten und Zuverlässigkeit unterscheiden,
- jede Verbindung kann den Zustand „operabel“ oder „ausgefallen“ annehmen,
- Verbindungen fallen unabhängig voneinander aus und
- Reparaturen der Verbindungen werden nicht berücksichtigt.

Das hier untersuchte Problem wird in [78, 176] als NP-schweres (englisch NP-hard) kombinatorisches Optimierungsproblem (vgl. [68]) dargestellt.

[4] Eine detaillierte Beschreibung dieses Zuverlässigkeitsmaßes gibt Abschnitt 4.3.

3.1.2 Lösungsverfahren für kombinatorische Optimierungsprobleme

Bisher wurden mit diesem Kapitel die Grundlagen kombinatorischer Optimierungsprobleme eingeführt und eine Modellierung der Kommunikationsnetzwerkplanung als kombinatorisches Optimierungsproblem vorgenommen. Mit diesem Abschnitt werden jetzt verschiedene Verfahren zur Lösung kombinatorischer Optimierungsprobleme vorgestellt. Ein Ansatz zur Lösung kombinatorischer Optimierungsprobleme ist die vollständige Enumeration, d.h. alle möglichen Lösungen werden auf ihre Optimalität hin überprüft. Für praktische Optimierungsprobleme muss und kann man sich jedoch im Sinne einer pragmatischen Entscheidungsunterstützung auf annäherungsweise gute (heuristische) Lösungen beschränken [61]. Fink und Voß [61] weisen für die Klasse der NP-schweren Probleme darauf hin, dass der aufzubringende Rechenaufwand zum Finden einer optimalen Lösung, wie es z. B. mit Hilfe einer vollständigen Enumeration möglich ist, unter praktischen Gesichtspunkten nicht gerechtfertigt ist.
Ein weitere Möglichkeit zur exakten Lösung solcher Probleme stellt der „Branch and Bound"-Ansatz [127] dar. Hier wird das Problem baumartig verzweigt. Jeder Knoten stellt ein zu lösendes Teilproblem dar. Durch die Erreichung einer unteren bzw. oberen Schranke und den Vergleich mit der bis dahin besten gefundenen Lösung kann frühzeitig entschieden werden, ob dieser Knoten weiter verzweigt werden muss. Dadurch sinkt der Rechenaufwand drastisch und es sind geringere Rechenzeiten als bei der vollständigen Enumeration erreichbar. Trotz der erreichten Rechenzeitverkürzung ist der „Branch and Bound"-Ansatz nicht geeignet, um große Probleme zu lösen, da das Verfahren für NP-schwere Probleme ein exponentielles Laufzeitverhalten zeigt und damit für große Probleminstanzen nicht praktikabel einsetzbar ist.
Heuristiken finden optimale oder fast optimale Lösungen in einem nach praktischen Gesichtspunkten vertretbaren Aufwand. Heuristische Lösungsverfahren beziehen dabei insbesondere problemspezifisches Wissen zur Findung einer guten (annähernd optimalen) Lösung in den Suchprozess mit ein. Im Vergleich zu exakten Verfahren garantieren sie jedoch keine gute oder gar optimale Lösung. Über die Jahre wurden für die Vielzahl von Problemen sehr viele Heuristiken entwickelt und erfolgreich eingesetzt (vgl. u. a. [144, 109, 39]). Sie haben insbesondere deswegen eine große praktische Bedeutung, da sie oft die einzigen Verfahren sind, die bei Problemen von praktischer Relevanz qualitativ gute Ergebnisse liefern [61].
Eine Erweiterung des Konzeptes der Heuristiken findet mit den Metaheuristiken statt, welche als nicht problemspezifische und damit generische Prinzipien zur Lösung von kombinatorischen Optimierungsproblemen häufig zum Einsatz kommen. Im folgenden Abschnitt werden die unterschiedlichen Lösungsverfahren, die in der Literatur unter dem Begriff Metaheuristiken zusammengefasst werden, vorgestellt.

3.2 Metaheuristiken als Lösungsansätze für kombinatorische Optimierungsprobleme

Der Begriff Metaheuristik wurde erstmals von Glover [72] eingeführt. Bereits die Zusammensetzung des Wortes aus den griechischen Begriffen Meta und Heuristik (abgeleitet

von dem Verb heuriskein (finden, entdecken)) macht das Wesen der verschiedenen unter diesem Begriff zusammengefassten Verfahren deutlich. Die Vorsilbe Meta ist Ausdruck dafür, dass die Verfahren generisch sind und somit die Heuristiken problemunabhängig arbeiten. In den letzten Jahrzehnten wurde eine Vielzahl von Metaheuristiken entwickelt. Im Allgemeinen lassen sich die Verfahren der Metaheuristiken in die in Abbildung 3.2 gezeigten Verfahrensklassen einteilen. Die Abbildung zeigt die Grobklassifizierung in

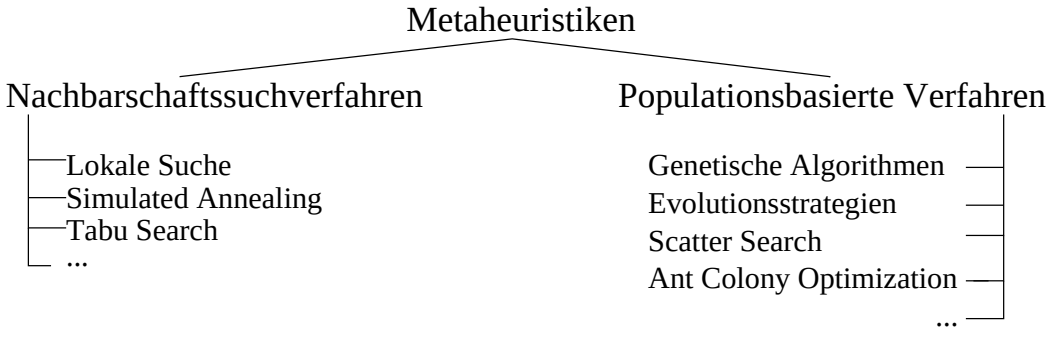


Abbildung 3.2: Klassifizierung von Metaheuristiken

Nachbarschaftssuchverfahren und populationsbasierte Ansätze und nennt jeweils einige typische Vertreter.
Die wesentlichen Eigenschaften einer Metaheuristik sind nach Blum und Roli [19]:

- Metaheuristiken steuern den Suchprozess,
- sie versuchen dabei den Lösungsraum effizient zu durchsuchen, um eine (annähernd) optimale Lösung zu finden,
- die in Metaheuristiken eingesetzten Verfahren reichen von lokalen Suchstrategien bis hin zu komplexen Lernprozessen,
- die Verfahren arbeiten dabei mit approximativen, häufig nicht deterministischen Ansätzen,
- Metaheuristiken stellen problemunabhängige Lösungsmechanismen dar,
- sie besitzen Mechanismen, die einer Stagnation der Suche in bestimmten Bereichen des Suchraums entgegenwirken,
- Metaheuristiken können problemspezifisches Wissen in Unterproblemlösern, welche durch die übergeordnete Metaheuristik gesteuert werden, nutzen und
- moderne Metaheuristiken nutzen Ergebnisse des Suchverlaufs, um die Suche gezielt zu steuern.

Im Folgenden werden die Grundzüge der in Abbildung 3.2 dargestellten Verfahrensklassen und deren typische Vertreter kurz erläutert.

3.2.1 Nachbarschaftssuchverfahren

Die Nachbarschaftssuchverfahren sind dadurch gekennzeichnet, dass die Verfahren jeweils mit einer einzigen Lösung arbeiten, welche in mehreren Iterationen hinsichtlich der Lösungsgüte (gemessen an der Verbesserung des Zielfunktionswertes) verbessert wird. Abbildung 3.3 stellt das Prinzip der einfachsten Nachbarschaftssuche, der lokalen Suche, dar. Ausgehend von einer Startlösung y werden in jeder Iteration die benachbarten Lösungen, die durch Anwendung eines Zuges $z \in \mathcal{Z}(y, Nach(y))$ generierbar sind, untersucht. Eine einfache lokale Suche ersetzt dabei die aktuelle Lösung y für das kombinatorische Optimierungsproblem durch die beste (im Sinne der Zielfunktionswertes) erreichbare Lösung y_{Nach}. Anschließend wird die lokale Suche mit y_{Nach} als Ausgangslösung neu gestartet. Das Verfahren stoppt mit der Lösung y^*, wenn in der Nachbarschaft $Nach(y)$ keine Lösung mit einem besseren Zielfunktionswert ($f(y_{Nach}) < f(y)$ für ein Minimierungsproblem) gefunden wird. Ein möglicher Verlauf einer lokalen Suche ist in Abbildung 3.3 gestrichelt dargestellt. Der Einsatz von Nachbarschaftssuchverfahren wie z. B. der lokalen Suche erfordert die Definition von Nachbarschaftsstrukturen, welche durch Anwendung der möglichen Züge $z \in \mathcal{Z}(y, Nach(y))$ generiert werden. Die Ge-

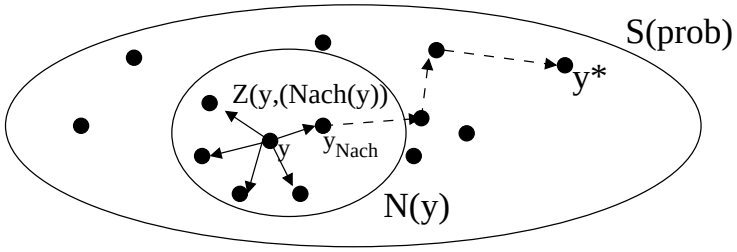


Abbildung 3.3: Ablauf eines Nachbarschaftssuchverfahrens

fahr beim Einsatz der hier dargestellten einfachen lokalen Suche besteht darin, dass der Suchprozess in einem lokalen Optimum konvergiert und ein globales Optimum aufgrund einer „unglücklich“ gewählten Startlösung niemals erreicht wird. Zur Überwindung dieses Problem gehören zur Klasse der Nachbarschaftssuchverfahren eine Reihe von Verfahren, die z. B. durch den Einsatz von modifizierten Akzeptanzkriterien für eine neue Startlösung, die Einbeziehung bisheriger Suchschritte bei der Generierung der Nachbarschaft sowie expliziten Mechanismen zur Vermeidung der Konvergenz in lokalen Optima das Finden von global optimalen Lösungen unterstützen. Als Vertreter dieser Klassen seien genannt: Iterated Local Search [132], Tabu Search [70], Simulated Annealing [104] und Variable Neighborhood Search [80].

3.2.2 Populationsbasierte Verfahren

Im Gegensatz zu den Nachbarschaftssuchverfahren, welche in jedem Iterationsschritt jeweils nur mit einer einzigen Lösung[5] arbeiten, verwenden populationsbasierte Ansätze in jeder Iteration eine Menge von Lösungen (in der Regel als Population bezeichnet).

[5] Eine Ausnahme bilden einige Nachbarschaftssuchverfahren für multikriterielle kombinatorische Optimierungsprobleme, auf die in Abschnitt 3.3.2 genauer eingegangen wird.

Die Lösungen in der Population werden dabei so über den Suchraum verteilt, dass sie die unterschiedlichen Bereiche gut abdecken und auf diese Weise in jeder Iteration unterschiedliche Bereiche des Suchraums parallel untersucht werden.
Mit Abbildung 3.4 wird das Grundprinzip der evolutionären Algorithmen (EA) grafisch dargestellt. Die Abbildung zeigt die Menge der Lösungen einer Population P_1 (hier $P_1 = \{y_1^1, y_2^1, \ldots, y_5^1\}$) sowie die Ausführung der beiden EA-Operatoren Rekombination und Mutation. Beide Operationen (Rekombination und Mutation) führen dazu, dass

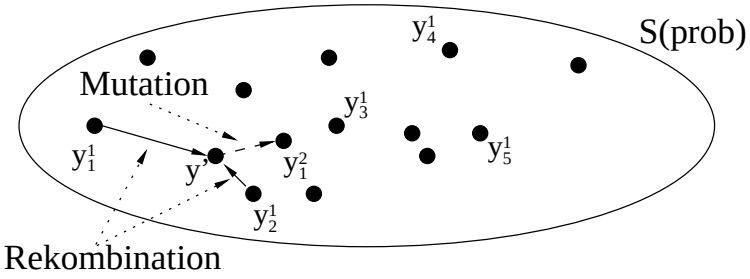


Abbildung 3.4: Ablauf einer populationsbasierten Suche

durch Veränderungen der Lösungen der aktuellen Population (Rekombination von zwei Lösungen zu einer neuen Lösungen y' oder kleine Veränderung (Mutation) an einer Lösung $y' \rightarrow y_1^2$) neue Lösungen für die nächste Iteration des Suchprozesses generiert werden. Eine detaillierte Beschreibung der evolutionären Verfahren und deren Operatoren gibt Abschnitt 3.5.
Neben den hier bereits erwähnten EA werden zu der Klasse der populationsbasierten Verfahren weiterhin die Metaheuristiken Ant Colony Optimization [54] und Swarm Intelligence [21] gezählt. Ebenso wie EA arbeiten diese Metaheuristiken mit einer Menge von parallel untersuchten Lösungen. Die Verfahren adaptieren dabei für die Suche verschiedene, aus der Natur bekannte Mechanismen (wie z. B. die Organisation eines Ameisenstaates) und übertragen diese Konzepte auf den Suchprozess.

3.3 Grundlagen für Entscheidungsprobleme mit mehreren Zielgrößen

Mit Abschnitt 3.1 wurde eine formale Beschreibung für ein kombinatorisches Optimierungsproblem gegeben. Dabei wurde herausgestellt, dass es Ziel eines Suchproblems ist, unter Beachtung einer gegebenen Zielfunktion f eine optimale Lösung y^* im Suchraum zu finden. Die vorgenommene Beschreibung der Zielfunktion lässt dabei jedoch lediglich die Abbildung einer Lösung auf einen Zielfunktionswert zu. Eine Vielzahl von Entscheidungsproblemen besitzt jedoch eine Menge von Zielen, die häufig zueinander konfliktär sind (vgl. [13, S. 49]). Im Nachfolgenden werden deshalb unterschiedliche Konzepte zur Lösung von Entscheidungsproblemem mit mehreren Zielkriterien vorgestellt.

3.3.1 Ansätze zur Optimierung mit mehreren Zielgrößen

Aufbauend auf der Beschreibung eines kombinatorischen Optimierungsproblems auf S. 26 kann ein Entscheidungsproblem mit mehreren Zielgrößen (im Folgenden als multikriterielles kombinatorisches Optimierungsproblem (MKOP) bezeichnet) beschrieben werden als (vgl. [107, 57]):

Definition 3.4 (Multikriterielles kombinatorisches Optimierungsproblem)
Ein multikriterielles kombinatorisches Optimierungsproblem ist ein spezielles Suchproblem, bei dem jedes prob ∈ Prob eine endliche Anzahl von Lösungen y ∈ sol(prob) besitzt und es eine Menge an Zielfunktionen f_i gibt, die jedem y einen Zielfunktionswert $f_i(y)$ zuordnen, den es zu minimieren oder maximieren gilt. Das Optimierungsproblem besteht im Finden einer Menge an pareto-optimalen Lösungen $\mathcal{Y}_{known} \in sol(prob)$.

Formal lässt sich ein multikriterielles kombinatorisches Optimierungsproblem (in Anlehnung an [99]) wie folgt darstellen :

$$\begin{aligned} & f(y) = \{f_1(y), f_2(y), \ldots, f_n(y)\} \\ & \text{mit} \\ & y = X = \{x_1, x_2, \ldots, x_n\} \text{ als Entscheidungsvariablen,} \\ & Dom_1, \ldots, Dom_n \text{ als Variablen-Domains und} \\ & NB = \{nb_1, \ldots, nb_n\} \text{ als Nebenbedingungen} \end{aligned} \tag{3.1}$$

Die Darstellung des MKOP zeigt die Zielfunktionen ($f(y) = \{f_1(y), f_2(y), \ldots, f_n(y)\}$) für das zu betrachtende Optimierungsproblem. Für jede dieser Zielfunktionen gilt, dass diese entweder zu maximieren oder zu minimieren ist. Weiterhin gelten für die Entscheidungsvariablen x_i die Domains Dom_i, die sich aus den an das Problem gestellten Nebenbedingungen NB ergeben. Für das in Abschnitt 3.1.1 beschriebene kombinatorische Optimierungsproblem gilt, dass sich dieses sowohl als monokriterielles als auch als multikriterielles kombinatorisches Optimierungsproblem darstellen lässt.
Wenn es dem Entscheider möglich ist, für jede der Zielfunktionen f_i eine individuelle Präferenz ϑ_i mit $\sum_{i=1}^{n} \vartheta_i = 1$ festzulegen, so kann die Zielfunktion[6] f formuliert werden als:

$$f(y) = \vartheta_1 \cdot f_1(y) + \vartheta_2 \cdot f_2(y) + \ldots + \vartheta_n \cdot f_n(y) \rightarrow \min$$

Auf diese Weise ist es möglich, die Zielfunktion des multikriteriellen Optimierungsproblems in eine aggregierte Zielfunktion eines monokriteriellen Optimierungsproblem umzuwandeln[7] (vgl. [189, S. 183]), welches eine optimale Lösung y^* besitzt.
Häufig ist die Vorgabe von Präferenzen jedoch ex ante nicht möglich. Daher wurden in der Vergangenheit eine Reihe von Verfahren zur Lösung von MKOP entwickelt, die eine simultane Optimierung mit mehreren Zielfunktionen ermöglichen. An Stelle einer einzigen optimalen Lösung generieren multikriterielle Lösungverfahren jedoch eine Menge

[6] Hier als Minimierungsproblem dargestellt. Die Betrachtung des Problems als Maximierungsproblem ohne Verlust der Allgemeingültigkeit der Aussagen ist ebenfalls möglich.
[7] Der Ansatz setzt eine Normierung der einzelnen Zielfunktionswerte bei der Abbildung in eine gemeinsamen Zielfunktion voraus.

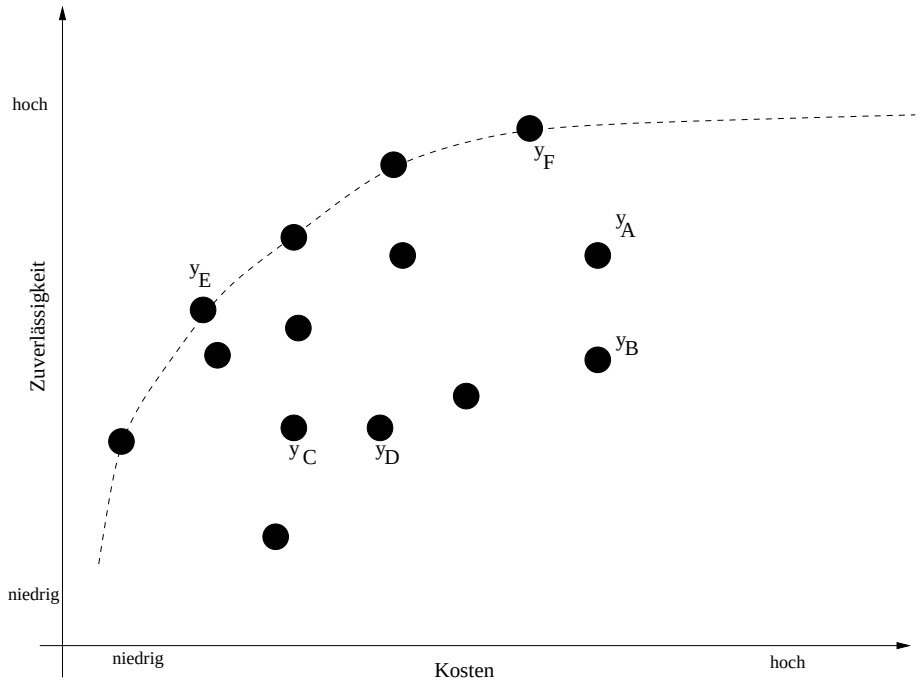


Abbildung 3.5: Lösungsmenge eines multikriteriellen Optimierungsproblems für die Ziele Kosten und Zuverlässigkeit

von Lösungen, aus denen der Entscheider ex post eine für ihn akzeptable Lösung auswählen muss.
An einem Beispiel wird die bei der multikriteriellen Optimierung entstehende Lösungsmenge mit ihren relevanten Eigenschaften eingeführt. Betrachtet werden im Folgenden die Kosten K und eine Zuverlässigkeit R. Ziel ist es, die Kosten zu minimieren und die Zuverlässigkeit zu maximieren. Der Entscheider trifft dabei keine Präferenzen bezüglich der beiden Ziele. Mögliche Konfigurationen als Lösung für das Problem, die sich hinsichtlich Kosten und Zuverlässigkeit unterscheiden, sind in Abbildung 3.5 dargestellt. Exemplarisch werden sechs dieser Lösungen herausgegriffen und paarweise einander gegenübergestellt. Für die Lösungen y_A und y_B gilt: $K(y_A) = K(y_B)$ und $R(y_A) > R(y_B)$. Bei gleichen Kosten kann somit mit Lösung y_A eine höhere Zuverlässigkeit erzielt werden. Bei der Auswahl zwischen diesen beiden Lösungen wird die Lösung y_A aufgrund der höheren Zuverlässigkeit bei gleichen Kosten ausgewählt. Für die Lösungen y_C und y_D gilt: $K(y_C) < K(y_D)$ und $R(y_C) = R(y_D)$. Da beide Lösungen die gleiche Zuverlässigkeit besitzen, ist bei einer Auswahl zwischen beiden der Lösung y_C aufgrund der geringeren Kosten der Vorzug zu geben. Da sowohl die Lösung y_A als auch die Lösung y_C für mindestens ein Ziel den gleichen Zielfunktionswert besitzen und für eines der Ziele besser als die jeweils gegenübergestellten Lösungen sind, werden die Lösungen y_B und y_D aus der Menge der betrachteten Lösungen gestrichen. Stelle man y_A der Lösung y_F gegenüber, so gilt: $K(y_A) > K(y_F)$ und $R(y_A) < R(y_F)$. Die Lösung y_F stellt unter Berücksichtigung beider Ziele somit aufgrund der geringeren Kosten und einer höheren Zuverlässigkeit die bessere Wahl dar. Für den Vergleich von y_C mit y_E gilt: $K(y_C) > K(y_E)$ und $R(y_C) < R(y_E)$. In diesem Vergleich ist die Lösung y_C der Lösung

y_E unterlegen und wird wie y_A aus der Menge der möglichen Lösungen gestrichen. Für die Lösungen y_E und y_F gilt: $R(y_E) < R(y_F)$ und $K(y_E) < K(y_F)$. Eine Entscheidung zwischen beiden Lösungen ohne die Angabe von Präferenzen ist nicht möglich.
Das Beispiel macht deutlich, dass für die Bewertung von Lösungen für ein MKOP die Betrachtung sämtlicher Ziele (und damit derer Zielfunktionswerte) notwendig ist. Hierfür wird das Konzept der Dominanz eingeführt.

Definition 3.5 (Dominanz)
Eine Lösung y_1 dominiert eine Lösung y_2 (notiert als $y_1 \succeq y_2$) wenn gilt:

$$\exists_{i=1}^{n} f_i(y_1) < f_i(y_2) \text{ und } \forall_{j=1,j\neq i}^{n} f_j(y_2) \geq f_j(y_1)$$

$$\text{mit } \forall_{j=1}^{n} f_j \rightarrow min$$

Definition 3.6 (Pareto-Optimalität)
Eine Lösung y ist pareto-optimal, wenn es keine andere Lösung y' gibt, die die Lösung y dominiert.

Definition 3.7 (Pareto-Front)
Die Menge aller pareto-optimalen Lösungen wird als Pareto-Front bezeichnet. Sie stellt die Lösungsmenge für ein MKOP dar.

Für die in Abbildung 3.5 dargestellte Lösungsmenge gilt:

- Die Lösungen y_A, y_B, y_C und y_D werden von den Lösungen y_E bzw. y_F dominiert.
- Die Lösungen y_E und y_F sind pareto-optimal.
- In der Abbildung 3.5 liegt die Menge der pareto-optimalen Lösungen (die Pareto-Front) auf der gestrichelt dargestellten Linie.

3.3.2 Metaheuristiken für multikriterielle Optimierungsprobleme

Für die Generierung der Lösungen der Pareto-Front wurden in den letzten Jahrzehnten eine Reihe unterschiedliche Verfahren entwickelt. Hierzu zählen z. B. verschiedene multikriterielle Simulated Annealing-Ansätze wie Serafinis Multiple-objective Simulated Annealing (SMOSA) [172], Multiple-objective Simulated Annealing (MOSA) [186], Pareto Simulated Annealing (PSA) [41] sowie eine Reihe von evolutionären Algorithmen (Strength Pareto evolutionary Algorithm 2 (SPEA-2) [195], Non-dominated Sorting Genetic Algorithm II (NSGA-II) [47]. Durch die Kombination von lokalen (nachbarschaftsbasierten) Suchverfahren mit populationsbasierten Verfahren – den sogenannten memetischen Algorithmen – sind eine Reihe weiterer multikriterieller Metaheuristiken wie z. B. Multi-objective Genetic Local Search (MOGLS) [98] und Memetic-Pareto Archived Evolution Strategy (M-PAES) [106] entstanden. Während nachbarschaftsbasierte Suchverfahren für monokriterielle Suchprobleme in jeder Iteration mit lediglich einer Lösung arbeiten, verwenden Nachbarschaftssuchverfahren wie z. B. PSA häufig ebenfalls eine Population an Lösungen, die parallel betrachtet werden. Durch die Verwendung der Population sind Nachbarschaftssuchverfahren in der Lage, unterschiedliche Bereiche

der Pareto-Front parallel zu untersuchen und damit in jeder Iteration eine Reihe von pareto-optimalen Lösungen zu finden. Zielsetzung eines jeden multikriteriellen Optimierungsverfahrens ist es, eine möglichst hohe Anzahl an pareto-optimalen Lösungen zu finden [47, S. 171]. Im Rahmen der Arbeit wird insbesondere der populationsbasierte Ansatz NSGA-II betrachtet. Dieses Verfahren wurde ausgewählt, da es als moderner multikriterieller evolutionärer genetischer Algorithmus sämtliche der in [194] aufgestellten Anforderungen an einen multikriteriellen evolutionären Algorithmus (vgl. hierzu Abschnitt 3.5.6) erfüllt.

3.4 No-Free Lunch Theorem

Wolpert u. a. [191] schreiben: „We show that all algorithms that search for an extremum of a cost function perform exactly the same, when averaged over all possible cost functions. In particular, if algorithm A outperforms algorithm B on some cost functions, then loosely speaking there must exist exactly as many other functions where B outperforms A.“ In dieser Aussage ist die Grundidee des No-Free Lunch (NFL) Theorems verankert. Dieses besagt, dass die Ergebnisse eines Lösungsverfahrens A gemittelt über alle Zielfunktionen gegenüber denen eines Lösungsverfahrens B nicht besser sein können. Fink und Voss [61] stellen als Schlussfolgerungen für das NFL-Theorem heraus: „Demgemäß ist festzustellen, dass es kein universelles Verfahren geben kann, welches andere Verfahren generell dominiert. In der Anwendung ist damit jeweils ein situativer Kompromiss zwischen Verfahren sowie dem Ausmaß der Verfahrensanpassung und dem entsprechenden Implementierungsaufwand einzugehen.“
Schumacher u. a. [168] zeigen, dass das NFL-Theorem nur dann gilt, wenn die Menge der betrachteten Probleme geschlossen unter Permutationen ist. Betrachtet man jedes beliebige Problem als eine Zuweisung von Fitnesswerten für jeden Punkt des Lösungsraums, so bedeutet die Annahme geschlossen unter Permutation, dass durch jede beliebige Permutation der Fitnesswerte für ein Problem ein anderes Problem erzeugt wird. Dies bedeutet, dass diese Probleme eine Struktur haben, die einem Algorithmus keinerlei Anhaltspunkte bieten, wo sich das Optimum befindet.
Corne und Knowles zeigen in [38], dass die Aussagen von Wolpert u. a. [191, 192] für monokriterielle Probleme auch für die Lösungsmengen multikriterieller Entscheidungsprobleme, welche geschlossen unter Permutation sind, zutreffen und somit das NFL-Theorem hier ebenfalls gilt.
Aufgrund der Annahme der Geschlossenheit unter Permutation ist jedoch davon auszugehen, dass das NFL-Theorem für nur wenige praktische Probleme gilt (vgl. hierzu auch [92]). Für multikriterielle Entscheidungsprobleme zeigen Knowles und Corne in [37], dass bei der Verwendung von absoluten Vergleichsmetriken einige multikriterielle Lösungsverfahren (aufgrund der verwendeten Archive) besser als andere Verfahren arbeiten und somit hier das NFL-Theorem nicht gilt.

3.5 Evolutionäre Verfahren

3.5.1 Grundlagen evolutionärer Algorithmen

Die Grundlagen der heutigen evolutionären Algorithmen gehen auf die von Darwin 1859 veröffentlichte Evolutionstheorie [43] „On the Origin of Species by Means of Natural Selection“ zurück. Darin stellte er die bis heute anerkannte Theorie auf, dass die Entwicklung der Lebewesen auf Basis von drei elementaren Prinzipien abläuft: Selektion, Rekombination und Mutation. Die Selektion (als natürliche Auswahl) sorgt dafür, dass einige Lebewesen (die am besten an ihre Umwelt angepasst sind (survival of the fittest)) besser überleben als andere und so ihr Erbgut weitergeben können. Neue Lebewesen werden durch geschlechtliche Fortpflanzung und der dabei durchgeführten Kombination der Erbanlagen gebildet. Dabei werden die verschiedenen (vorteilhaften) Eigenschaften der Eltern auf die Nachkommen übertragen. Mutation verändert das Erbgut zufällig. Die meisten Mutationen werden durch Selektion unterdrückt, da sie keine Verbesserung für das Individuum bringen. Einige Mutationen können mitunter jedoch vorteilhafte Eigenschaften hervorbringen und werden dann (da sie bei der Selektion vorteilhaft bewertet werden) auch an die Nachkommen weitergegeben.
Betrachtet man die Evolution als Werkzeug für die Lösung von kombinatorischen Optimierungsproblemen, ist sie ein spezielles Optimierungsverfahren, das in der Lage ist, komplexe Lebewesen hervorzubringen und an verschiedenste Lebensbedingungen anzupassen. In den 60er Jahren des letzten Jahrhunderts versuchten Wissenschaftler, die Prinzipien der Evolution erstmals auf Computer zu übertragen. Erste Arbeiten führte Rechenberg [154] mit dem Einsatz von Evolutionsstrategien zur Bauteiloptimierung durch. Parallel dazu entwickelte Holland [88] das Konzept der genetischen Algorithmen. Andere Ansätze wie das evolutionäre Programmieren, das von Fogel [64] entwickelt wurde, reichen ebenfalls bis in diese Zeit zurück. Heute werden diese Konzepte auf eine Vielzahl von Problemen angewendet (vgl. [75]). Oft sind die Grenzen zwischen ihnen verschwommen, im Allgemeinen findet sich die Grundidee von Darwin jedoch in allen Verfahren wieder. Das Konzept der evolutionären Algorithmen ist vielen Mechanismen der natürlichen Fortpflanzung nachempfunden. Die verwendete Begriffswelt ist dabei an die der Evolutionslehre der Biologie angelehnt. Einen guten Überblick über die biologischen Abläufe und Begriffe bieten Schöneburg u. a. [167, S.74ff]. Grundlegende Definitionen werden im Folgenden vorgestellt und kurz erläutert. Die Aussagen erfolgen dabei in Anlehnung an Goldberg [73, S.22], Michalewicz [142, S.5f], Pelikan [147], Nissen [145, S. 12] und Eiben [58, S.18ff].

Definition 3.8 (Genom)
Die Repräsentation des Problems für den evolutionären Algorithmus wird als Genom g (synonym auch als Chromosom oder String) bezeichnet.

Definition 3.9 (Gen)
Die einzelnen Teilbereiche eines Genoms werden als Gen bezeichnet. Jedes Gen hat eine Position, die als Locus bezeichnet wird.

Definition 3.10 (Allel/Allelset)
Die Ausprägung eines Gens g_i wird Allel genannt. Alle möglichen Ausprägungen eines Gens bilden das Allelset.

Durch die Verwendung der Allele wird es möglich, dass jedes Gen nicht nur eine reine binäre Information kodieren kann, sondern stattdessen für jedes Gen eine Zuordnung einer Wertedomain möglich wird. Auf diese Weise ist das Gen in der Lage, die Variablen-Domain D_i einer Entscheidungsvariablen x_i abzubilden.

Definition 3.11 (Genotyp)
Die Gesamtheit aller Gene eines Chromosoms bzw. Strings wird Genotyp genannt.

Definition 3.12 (Phänotyp)
In der Natur wird als Phänotyp das aus dem Genotyp in Interaktion mit seiner Umwelt entstandene Lebewesen bezeichnet. Bezogen auf den Bereich der genetischen Algorithmen ist also der Phänotyp das Problem in seiner normalen (problemspezifischen) Darstellung und der Genotyp die Repräsentation für den genetischen Algorithmus.

Die Wahl oder der Entwurf einer geeigneten Repräsentation des Problems (kodiert über den Phänotyp) in eine für den evolutionären Algorithmus verarbeitbare Repräsentation (als Genotyp) ist eine wichtige Voraussetzung für den Einsatz der Metaheuristik.

Definition 3.13 (Population)
Alle Individuen, die zu einem bestimmten Zeitpunkt t existieren, werden Population (notiert als P_t) genannt. Die Anzahl der Individuen einer Population wird mit $pop = |P_t|$ angegeben.

Definition 3.14 (Selektion)
Die Selektion (natürliche Auslese) ist ein von der Merkmalsausprägung (Phänotyp) der Individuen abhängiger Vorgang. Er entscheidet über die Zusammensetzung der nächsten Generation und die Chancen auf Fortpflanzung für ein Individuum.

Der Einsatz der Selektion zur Auswahl der Individuen setzt voraus, dass ein Maß existiert, das eine Entscheidung darüber zulässt, wie gut ein Individuum im Vergleich zu einem anderen Individuum ist.

Definition 3.15 (Zielfunktion)
Die Zielfunktion (auch als Bewertungsfunktion bezeichnet) steuert die Suche. Sie wird in Abhängigkeit von der Problemstellung minimiert oder maximiert und macht damit Chromosomen hinsichtlich ihrer Güte vergleichbar. Sie ist in sehr vielen Anwendungsbereichen der evolutionären Algorithmen als formales mathematisches Modell formuliert und kann durch einen Computer berechnet werden. Einige Anwendungen evolutionärer Algorithmen setzen die Eingabe des Zielfunktionswertes durch den Benutzer voraus (siehe z. B. [55]).

Definition 3.16 (Fitnessfunktion)
Die Fitnessfunktion gibt an, wie sich die Bewertung eines Chromosoms auf seine Chancen, sich in die nächste Generation fortzupflanzen, rechnerisch auswirkt. Sie ist meist eine einfache Transformation der Zielfunktion.

Mit Hilfe der Variationsoperatoren Rekombination und Mutation werden neue Individuen erzeugt.

Definition 3.17 (Rekombination)
Die Rekombination ist der Austausch von Genen oder Genabschnitten der Elternindividuen zur Erzeugung von Nachfahren.

Definition 3.18 (Mutation)
Mutationen sind spontane, mit geringer Wahrscheinlichkeit auftretende, strukturelle Veränderungen einzelner Gene.

Evolutionäre Verfahren führen die Optimierung der Lösungen hin zu einer guten Lösung im Sinne der Zielfunktion über mehrere Stufen (Iterationen) hinweg aus.

Definition 3.19 (Generation)
Die von einem evolutionären Algorithmus durchlaufenen Iterationen werden als Generationen bezeichnet.

Der prinzipielle Ablauf eines evolutionären Algorithmus ist in Algorithmus 3.1 dargestellt. Der evolutionäre Algorithmus startet mit der Erzeugung einer initialen Population

Algorithmus 3.1 Evolutionärer Algorithmus

```
t:=0
Erzeuge initiale Population P_0
Bewerte P_0
repeat
  P*_t = Selektiere Individuen aus P_t
  P'_t = Rekombiniere P*_t
  P''_t = Mutiere P'_t
  Bewerte P''_t
  P_{t+1} = Wähle gute Individuen aus P_t und P''_t
  t = t+1
until Abbruchkriterium
```

P_0. Diese kann aus zufällig generierten Individuen bestehen, sich aus mit Hilfe von Eröffnungsverfahren[8] generierten Individuen zusammensetzen oder aus einer Mischung aus zufälligen und konstruierten Lösungen bestehen. Durch die wiederholte Anwendung der Operatoren Selektion (Selektiere Individuen aus P_t), Rekombination (Rekombiniere P_t^*) und Mutation (Mutiere P_t') wird mit jeder Iteration eine neue Kindergeneration erstellt. Aus den Eltern- und Kinderindividuen werden nach einem Ersetzungsschema (Wähle gute Individuen aus P_t und P_t'') Individuen ausgewählt, die zur neuen Elternpopulation werden. Über die Generationen hinweg nähert sich der Zielfunktionswert des besten Individuums der Population immer weiter dem gesuchten Optimum an. Der Vorgang der

[8] Hierfür werden häufig Konstruktionsheuristiken wie z. B. die „Nächste Nachbar“ Heuristik für das Handelsreisendenproblem [100] oder das Verfahren von Prim [151] zur Erstellung eines minimalen Baumes eingesetzt.

Erzeugung von Kinderpopulationen wird so lange fortgesetzt, bis eine Abbruchbedingung erfüllt ist. Als Abbruchbedingungen eignen sich u. a. die Anzahl der berechneten Generationen, die Diversifikation der Fitnesswerte in der Population oder die verbrauchte CPU-Zeit.
Zur Klasse der evolutionären Algorithmen zählen:

- Evolutionsstrategien
- Evolutionäres Programmieren
- Genetische Algorithmen
- Genetisches Programmieren

Der nachfolgende Abschnitt stellt die unterschiedlichen Algorithmen vor.

3.5.2 Evolutionäre Verfahren zur Optimierung mit einer Zielgröße

3.5.2.1 Evolutionsstrategien

Erstmals vorgeschlagen wurden Evolutionsstrategien (ES) bereits vor 40 Jahren von Rechenberg [154] und Schwefel [170]. ES werden typischerweise für Probleme mit kontinuierlichen Parametern eingesetzt. Jedes Gen repräsentiert dabei eine Fließkommazahl. Mit Algorithmus 3.2 wird eine einfache (1+1)-Evolutionsstrategie gezeigt. Ausgehend

Algorithmus 3.2 (1+1)-Evolutionsstrategie

```
t = 0
Erzeuge initiale Lösung y = {x_1^t, x_2^t, ..., x_n^t} mit x_i ∈ ℝ
repeat
  for i = 1 ...n do
    Erzeuge mittels Normalverteilung (N(0,σ)) z_i
    x̂_i^t = x_i^t + z_i
  end for
  if f(y^t) ≤ f(ŷ^t) then
    y^{t+1} = y^t
  else
    y^{t+1} = ŷ^t
  end if
  t = t+1
until Abbruchkriterium
```

von der Lösung y^t wird in jeder Generation eine neue Lösung $\hat{y}^t$ generiert, indem zu jeder Entscheidungsvariablen x_i eine normalverteilte Zufallszahl z_i mit dem Erwartungswert Null und der Standardabweichung σ addiert wird. Die neue Lösung $\hat{y}^t$ ersetzt die Lösung y^t, wenn sie einen besseren Zielfunktionswert besitzt. Die Standardabweichung σ ist einer der entscheidenden Parameter der ES. Sie bestimmt, wie groß die Veränderungen sind, die durch den Mutationsoperator durchgeführt werden. Durch den Parameter σ

kann für jede Generation die Variationsbreite bei der Generierung der neuen Lösung gesteuert werden. Rechenberg [154] schlägt für die Anpassung des Parameters σ die noch heute verwendete 1/5-Erfolgsregel vor. Dabei werden die erfolgreichen Mutationen ins Verhältnis zu allen durchgeführten Mutationen gesetzt. Die Regel besagt, dass das Verhältnis (p_s) 1/5 betragen sollte. Ist das Verhältnis größer, so muss die Suchbreite vergrößert werden. Man geht dabei davon aus, dass sich die aktuelle Lösung weit entfernt vom Optimum befindet und eine breite Suche zur Annäherung an dieses notwendig ist. Ist das Verhältnis kleiner 1/5, so befindet sich der Suchprozess in der Nähe eines Optimums und die Suchbreite ist zu verringern. Die 1/5 Regel ist wie folgt definiert:

$$\sigma_{t+1} = \begin{cases} \sigma_t / c & : \text{ wenn } p_s > 1/5 \\ \sigma_t \cdot c & : \text{ wenn } p_s = 1/5 \\ \sigma_t & : \text{ sonst} \end{cases}$$

Schwefel [171] schlägt als Werte $0,89 \leq c \leq 1$ vor.
Heutige ES nutzen an Stelle der 1/5-Regel häufig das Konzept der Selbstanpassung (vgl. hierzu [17]). Hierfür wird für jede Entscheidungsvariable x_i eine Strategievariable σ_i eingeführt. Ein Genom setzt sich dadurch zusammen aus:

$$(x, \sigma) = (\{\underbrace{x_1, x_2, \ldots, x_n}_{X}\}, \{\underbrace{\sigma_1, \sigma_2, \ldots, \sigma_n}_{\overline{\sigma}}\})$$

Der Vektor $\overline{\sigma}$ bestimmt dabei für jede Entscheidungsvariable x_i mittels der Strategievariable σ_i deren Variationsbreite.
Bisher wurde nur die einfache (1+1)-ES mit einem Kind- und einem Eltern-Genom vorgestellt. Abweichend von dieser von Rechenberg [154] vorgeschlagenen Strategie, werden häufig sogenannte mehrgliedrige (μ_{ES}+λ_{ES})-ES eingesetzt. Bei einer solchen ES werden aus μ_{ES} Eltern und den λ_{ES} Nachkommen die besten (fittesten) μ_{ES} Nachkommen in die nächste Generation als neue Eltern übernommen. Im Gegensatz zur sogenannten Plusnotation (μ_{ES}+λ_{ES}) werden bei der Kommanotation (μ_{ES},λ_{ES}) für die Auswahl der Nachkommen μ_{ES} der nächsten Generation nur die Lösungen λ_{ES} betrachtet.
Wie andere evolutionäre Verfahren auch setzen Evolutionsstrategien die Rekombination zur Generierung neuer Nachkommen ein[9]. Im Gegensatz zu genetischen Algorithmen nimmt die Rekombination jedoch eine untergeordnete Rolle ein. Bei der Rekombination wird für ES unterschieden zwischen der lokalen und globalen Rekombination sowie zwischen einer diskreten und einer intermediären Rekombination. Formal lässt sich die (lokale) Rekombination zweier Individuen y_1 und y_2 zu einem Individuum y_3 beschreiben als:

$$\forall_{i=1}^{n} y_3(x_i) = \begin{cases} (y_1(x_i) + y_2(x_i))/2 & \text{intermediäre Rekombination} \\ y_1(x_i) \text{ oder } y_2(x_i) & \text{diskrete Rekombination} \end{cases}$$

[9] Zur Veranschaulichung des Grundprinzips der ES wurde in Algorithmus 3.2 auf die Darstellung der Rekombination verzichtet.

Während dabei lediglich zwei Individuen der Population betrachtet werden, wird dies für die globale Rekombination auf alle Individuen der Population erweitert. Bei der globalen Rekombination werden für jedes y_i jeweils zwei Individuen aus der Population ausgewählt und darauf die zuvor genannten Rekombinationsprinzipien angewendet.
In der Literatur existieren eine Reihe von ES, die unterschiedliche Mechanismen zur Anzahl und Strategie der Übernahme von Lösungen in die nächste Generation vorschlagen. Eine Übersicht hierzu findet man bei Beyer und Schwefel [17], Rechenberg [155] und Kost [113].

3.5.2.2 Evolutionäres Programmieren

Wie die Evolutionstrategien, so kann auch das evolutionäre Programmieren (EP) bereits auf eine lange Geschichte zurückschauen. Erste Arbeiten gehen auf Fogel u.a [64] in den 60er Jahren zurück. Damals stand jedoch nicht der Optimierungsgedanke im Vordergrund. Stattdessen versuchte man Systeme mit künstlicher Intelligenz zu generieren, die in der Lage sein sollten, selbständig Probleme zu lösen. Zielsetzung war die Entwicklung künstlicher Intelligenz, die in der Lage ist, Veränderungen in einer angeschlossenen Umgebung vorherzusagen. Die betrachtete Umgebung wird dabei als eine Folge von Zuständen beschrieben. Aufgabe des Algorithmus ist es, einen neuen Zustand des Systems (als Vorhersage) zu generieren. Die Güte des Algorithmus wird anhand der Genauigkeit der damit generierten Vorhersage gemessen. Die Repräsentation der Individuen für das EP erfolgt mittels endlicher Automaten. Anhand des in [140, S. 283f] eingeführten Beispiels wird das Prinzip des EP erläutert.
Abbildung 3.6 zeigt das Transitionsdiagramm für einen endlichen Automaten zur Paritätsprüfung für einen Binär-String. Für eine Eingabe $ein \in [0, 1]$ liefert der Automat die Ausgabe aus „wahr“ ($aus = 1$), wenn der Binär-String eine ungerade Anzahl von Einsen enthält. Anderenfalls ist die Ausgabe „falsch“ ($aus = 0$). In Form eines gerichteten Gra-

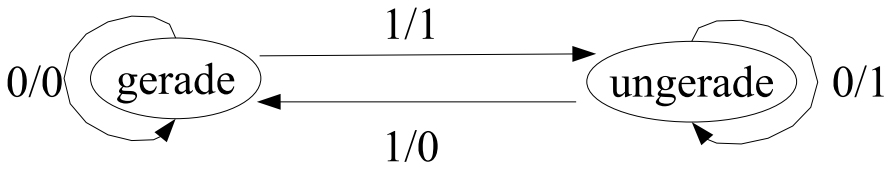


Abbildung 3.6: Endlicher Automat für eine Paritätsprüfung [140, S. 284]

phen zeigt das Transitionsdiagramm für jeden Zustand Z_i (modelliert als Knoten) die Transitionen $Tr = \{tr_1, \ldots, tr_n\}$ (modelliert als Kanten), die zu einem anderen Zustand Z_j führen. Eine Kante vom Knoten Z_i zu Z_j zeigt dabei mittels der Notation a/b den Output b an, der durch die Eingabe a entsteht, wenn sich das System im Zustand S_i befindet. Befindet sich das System im betrachteten Beispiel im Zustand „gerade“, so würde durch das Hinzufügen einer 0 zu dem Binär-String (Transition 0/0) die Ausgabe 0 erzeugt und der Automat im Zustand „gerade“ verweilen.

Beim EP besteht die Population aus einer Menge von endlichen Zustandsautomaten. Die Bewertung eines Individuums der Population erfolgt mittels der Vorhersagegenauigkeit des jeweiligen endlichen Automaten und für eine Menge bekannter Eingaben $(a_1, a_2, \ldots, a_n)$. Mit Hilfe des endlichen Automaten wird für jede Eingabe $a_1, \ldots, a_n$ eine Vorhersage für die Ausgabe a'_i bestimmt und diese mit a_{i+1} verglichen. Die Fitness eines Individuums spiegelt dabei die Vorhersagegenauigkeit für alle betrachteten Zustände wider [140, S. 284].
Der Hauptoperator des EP ist die Mutation. Zur Generierung der Individuen der nächsten Generation wird zunächst aus jedem Individuum der Elterngeneration ein Nachkomme erstellt. Hierfür kennt das EP fünf Mutationsoperatoren: Wechseln eines Eingabesymbols an einer Transition, Ändern einer Transition (Umlenken zu einem anderen Zustand), Hinzufügen eines weiteren Zustandes, Löschen eines Zustandes sowie Ändern des Initialzustandes des Automaten. Für die Mutation wird einer dieser Operatoren zufällig ausgewählt. Aus den erzeugten Nachkommen und den Eltern werden die besten 50 % als neue Elterngeneration in die nächste Generation übernommen. Die Rekombination wird innerhalb des EP nicht eingesetzt [58, S. 94].
In den letzten 15 Jahren wurden verstärkt Variationen des EP mit reellwertigen Parametern vorgestellt, die ähnlich den ES arbeiten. Eine detaillierte Betrachtung zu dieser Entwicklung und vorgeschlagenen Verfahren bietet Eiben [58, S. 91ff].

3.5.2.3 Genetische Algorithmen

Das Konzept der genetischen Algorithmen (GA) geht auf die Arbeiten von Holland [88] zurück. Parallel und unabhängig entwickelten Holland (GA) und Rechenberg (ES) ihre evolutionären Algorithmen, die bis heute als leistungsfähige Metaheuristiken eingesetzt werden.
Goldberg [73, S. 1] definiert einen GA als:

Definition 3.20 (Genetischer Algorithmus)
„Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics. They combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search. In every generation, a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure.“

Die Definition macht den Bezug der Metaheuristik zur Evolutionslehre deutlich. Abgeleitet von der Definition lassen sich für die GA die folgenden grundlegenden Eigenschaften feststellen:

- Sie arbeiten nicht mit den Problemparametern selbst, sondern benutzen eine Kodierung („artificial creatures (strings)“),
- sie sind ein populationsbasierter Ansatz („In every generation, a set of artificial creatures“), d.h. die Suche wird parallel mit einer Vielzahl von Lösungen durchgeführt,

- Genetische Algorithmen sind stochastische Suchverfahren („randomized information exchange“) und
- sie sind Metaheuristiken, die zur Lösung verschiedenster Probleme eingesetzt werden können.

Den Ablauf eines einfachen GAs[10] stellt Algorithmus 3.3 dar. Zu Beginn wird die Po-

Algorithmus 3.3 Einfacher genetischer Algorithmus (in Anlehnung an [60, S. 80])

```
t = 0
Belege Initialpopulation P_0 mit zufällig generierten Lösungen
Bewerte(P_0)
y_best = Beste Lösung in P_0
repeat
  FSUM = Σ_{i=1}^{pop} f(y_i)
  P_{t+1} = ∅
  for i = 1 ... pop do
    Wähle zufällig 2 Individuen y_1, y_2 ∈ P_t gemäß der relativen Fitness f(y)/FSUM
    Erzeuge Nachkommen y_{N_1}, y_{N_2} mittels One-Point-Crossover und Flip-Mutation
    Bewerte Nachkommen y_{N_1}, y_{N_2}
    P_{t+1} = P_{t+1} ∪ {y_{N_1}, y_{N_2}}
    if f(y_{N_1} besser f(y_best)) then
      y_best = y_{N_1}
    end if
    if f(y_{N_2} besser f(y_best)) then
      y_best = y_{N_2}
    end if
  end for
  t = t + 1
until Abbruchbedingung
```

pulation P_0 mit zufällig generierten Lösungen gefüllt. Jede Lösung der Population P_0 wird anschließend bewertet. Die Lösung mit dem besten Fitnesswert wird in y_{best} gespeichert. In jeder Generation werden *pop* (Anzahl der Individuen in der Population) neue Lösungen aus der Elterngeneration P_t für die Nachkommengeneration P_{t+1} erzeugt. Das Verhältnis der Fitness eines Individuums im Verhältnis zur Summe sämtlicher Fitnesswerte ($FSUM$) bestimmt die Wahrscheinlichkeit, mit der ein Individuum für die Fortpflanzung ausgewählt wird. Durch die Verwendung der relativen Fitness $f(y)/FSUM$ haben Lösungen mit einer hohen Fitness eine höhere Wahrscheinlichkeit, als Elter gewählt zu werden. Die ausgewählten Eltern durchlaufen einen One-Point-Crossover (vgl. Abbildung 3.8 auf S. 48) und generieren dabei die Nachkommen y_{N_1}, y_{N_2}. Diese werden anschließend mittels einer Flip-Mutation (vgl. Abbildung 3.9 auf S. 48) mutiert. Beide Kinder werden der neuen Generation P_{t+1} hinzugefügt. Abschließend wird für beide Lösungen (y_{N_1}, y_{N_2}) geprüft, ob diese besser als die bisher beste gefundene Lösung sind

[10] In der Literatur ([60, S. 80],[58, S. 38]) auch als kanonischer genetischer Algorithmus bezeichnet.

und gegebenenfalls die beste Lösung aktualisiert. Der Prozess der Auswahl zweier Eltern und der Erzeugung von zwei Kindern wird solange wiederholt, bis die Population P_{t+1} aufgefüllt ist. Der GA stoppt, wenn ein Abbruchkriterium (vgl. Abschnitt 3.5.1) erreicht ist. Ausgehend von Algorithmus 3.3 werden im Folgenden wichtige Elemente des GAs sowie mögliche Formen der Problemrepräsentation detailliert betrachtet.
Eine grundlegende Entscheidung, die beim Einsatz eines GAs zu treffen ist, ist die Wahl der richtigen Repräsentation. Eiben schreibt [58, S. 40]: „Getting the representation right is one of the most difficult parts of designing a good evolutionary algorithm." Die Wahl der Repräsentation einer Lösung in dem GA bestimmt, wie die Zuordnung zwischen dem Phänotyp und dem Genotyp erfolgt. Eine der einfachsten Repräsentationen ist der Binär-String. Dabei nimmt jedes Allel den Wert Null oder Eins an. Viele Entscheidungsvariablen lassen sich jedoch nicht mit Hilfe eines einfachen Wahrheitswertes, wie er bei einem Binär-String verwendet wird, ausdrücken. Für die Repräsentation ordinaler Eigenschaften bietet sich die Verwendung von Integer-Strings an. Hier kann jedes Gen Integer-Werte innerhalb eines definierten Bereiches (Allelset) annehmen. Für Probleme mit Entscheidungsvariablen mit einem kontinuierlichen Wertebereich kommt, ähnlich wie bei den Evolutionsstrategien, eine Fließkomma-Repräsentation zum Einsatz. In der Literatur existieren eine Reihe weiterer Repräsentationen (vgl. u. a. [163, 33]), die hier nicht im Detail betrachtet werden. Eine gute Übersicht (insbesondere auch im Anwendungskontext der Kommunikationsnetzwerkplanung) bietet Rothlauf [162].
Für die Bestimmung der Individuen, die für die Fortpflanzung herangezogen werden, führt der GA eine Bewertung mittels einer Fitnessfunktion durch. Die Fitnessfunktion ist eine Transformation der Bewertungsfunktion und bestimmt die Wahrscheinlichkeit dafür, dass ein Individuum für die Fortpflanzung ausgewählt wird. Individuen mit einer guten Bewertung[11] müssen somit einen hohen Fitnesswert aufweisen. Dadurch wird erreicht, dass Individuen, die dem Zielkriterium gut entsprechen, sich mit großer Wahrscheinlichkeit fortpflanzen bzw. in die nächste Generation übernommen werden. Eine Aufgabe der Fitnessfunktion ist es also, diese Relation, z. B. durch die Verwendung des reziproken oder negativen Zielfunktionswertes bei Minimierungsproblemen, widerzuspiegeln. Die andere Aufgabe der Fitnessfunktion ist die Regelung des Wettbewerbsdrucks, um eine optimale Performance des GAs zu erreichen „...to regulate the level of competition among members of the population to achieve the interim and ultimate algorithm performance we desire." [73, S. 76]. Diese sogenannte Fitness-Skalierung kann auf verschiedene Weise erfolgen. Eine häufig verwendete Fitnessfunktion [167, S.196] ist die proportionale Fitness. Hier wird die Bewertung einer Lösung y mit Hilfe eines Skalierungsfaktors $a \in \mathbb{R}$ in das Verhältnis zur Summe der Fitness aller Populationsmitglieder gesetzt:

$$f_{prop}(y) = a \cdot \frac{f(y)}{\sum_{i=1}^{n} f(y_i)}$$

Ein anderes Verfahren zur Bestimmung der Auswahlwahrscheinlichkeit eines Individuums stellt die rangbasierte Selektion [10] dar. Innerhalb der Population wird dabei jedem Individuum in Abhängigkeit von seiner Fitness ein Rang zugewiesen. Das beste Indi-

[11] Für Minimierungsprobleme sind dies Lösungen mit einem kleineren Zielfunktionswert, für Maximierungsprobleme Lösungen mit einem größeren Zielfunktionswert.

viduum erhält den höchsten Rang, das schlechteste den niedrigsten. Die Berechnung zwischen einem Rang und der daraus resultierenden Auswahlwahrscheinlichkeit für die Fortpflanzung erfolgt mit Hilfe einer beliebigen Funktion. Gebräuchlich ist ein lineares Rangschema, das die Auswahlwahrscheinlichkeit nach (vgl. [58, S. 60]):

$$p_{Rang}(i) = \frac{2 - sel}{pop} + \frac{2 \cdot i \cdot (sel - 1)}{pop \cdot (pop - 1)}$$

für das Individuum mit dem Rang i angibt. Das schlechteste Individuum hat den Rang 1 und das beste den Rang pop. Für den Parameter sel, der den Selektionsdruck steuert, gilt: $1,0 \leq sel \leq 2,0$. Je größer sel wird, um so mehr nimmt der Selektionsdruck zu. Ein höherer Selektionsdruck bedeutet, dass für Individuen mit einem hohen Fitnesswert die Auswahlwahrscheinlichkeit zunimmt.
Mit dem Selektionsoperator (in der deutschen Literatur oftmals auch als Heirats-Schema bezeichnet) wird bestimmt, welche Lösungen der Elternpopulation zur Erzeugung neuer Individuen der Kindergeneration herangezogen werden. Jede Lösung, die für die Fortpflanzung ausgewählt wird, wird zunächst in einen Pool mit den für die Rekombination vorgesehenen Lösungen eingestellt. Für die Auswahl sind verschiedene Mechanismen möglich, denen aber allen eine Grundidee gemeinsam ist: „Make more copies of solutions that perform better at the expense of solutions that perform worse.“ [147, S. 10]. Einer der einfachsten und häufig eingesetzten Selektionsoperatoren ist die sogenannte „Roulette-Rad Selektion“ (Roulette Wheel). Dabei wird auf einem manipulierten Roulette-Rad (siehe Abbildung 3.7) jedem Individuum ein Bereich zugewiesen, dessen Größe proportional zu dessen Auswahlwahrscheinlichkeit ist. So erhalten Lösungen mit einer hohen Fitness größere Chancen sich fortzupflanzen. Lösungen mit geringerer Fitness haben schlechtere Chancen, sind aber nicht ganz ausgeschlossen.
Tabelle 3.1 zeigt exemplarisch die Population $P_{Bsp} = \{y_1, y_2, y_3, y_4\}$. Jedem Individuum wurde ein Fitnesswert $f(y)$ zugeordnet[12]. Die letzte Spalte zeigt die fitnessproportionale Auswahlwahrscheinlichkeit für jedes Individuum in P_{Bsp} an. In Abbildung 3.7 wurden der prozentuale Anteil der Fitness über alle Populationsmitglieder an dem „manipulierten“ Roulette-Rad abgetragen. Die Auswahl der Lösungen für die Rekombination

Tabelle 3.1: Beispiel für die Fitnessbewertung und anschließende Selektion eines genetischen Algorithmus

P_{Bsp}	Genom	$f(y_i)$	prozentualer Anteil an $FSUM$
y_1	101100	233	55 %
y_2	010001	47	11 %
y_3	011001	124	29 %
y_4	101010	23	5 %
		$FSUM = 427$	$\sum 100\,\%$

erfolgt durch das Ziehen einer Zufallszahl $r \in [0, 1]$. Eine andere einfache Form der Selektion lässt sich mit Hilfe der „Wettkampf“-Selektion (auch als Tournament-Selektion

[12] Die Bestimmung des Fitnesswertes soll hier nicht betrachtet werden.

bezeichnet) [74] erreichen. Statt jedem Element in P einen Rang und/oder eine Auswahlwahrscheinlichkeit zuzuordnen, werden bei der Selektion zufällig n Individuen[13] aus P ausgewählt und miteinander hinsichtlich ihrer Fitness verglichen. Die jeweils beste Lösung wird als ein Elter für die Fortpflanzung ausgewählt. Die „Wettkämpfe“ werden solange wiederholt, bis ausreichend Eltern für die Rekombination zur Verfügung stehen. Eine weiterführende Betrachtung zu Selektionsprinzipien bietet Nissen [145, S. 64ff].
An die Auswahl der Individuen für die Fortpflanzung schließt sich der Rekombinationsprozess an. Die Rekombination (auch als Crossover bezeichnet) ist der wichtigste Opera-

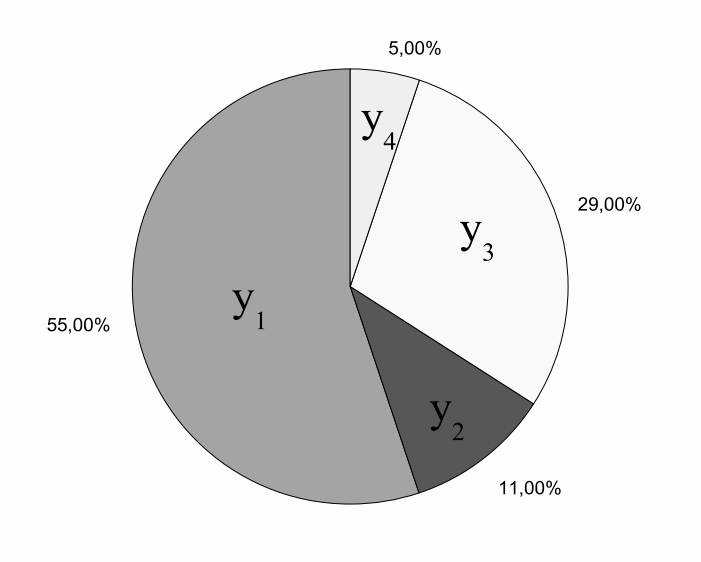
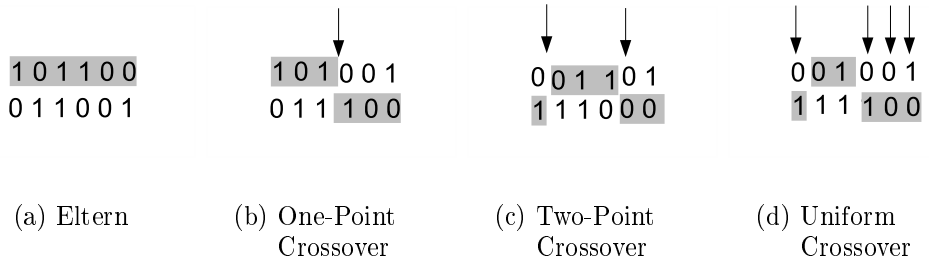


Abbildung 3.7: Beispiel Roulette-Wheel Selektion

tor des genetischen Algorithmus [145, S. 38]. Er stellt sicher, dass der Suchraum effizient durchschritten wird und dabei Lösungen mit hoher durchschnittlicher Fitness schnell erreicht werden. Schöneburg u. a. stellten heraus: „Der Hauptgrund für die Konzentration der GA-Theoretiker auf die Crossover-Mechanismen ist, dass in ihnen problemspezifisches prozedurales Wissen über den Suchraum abgelegt werden kann. Der Grundalgorithmus eines GAs bleibt unverändert, während die Crossover-Mechanismen individuell auf die speziellen Optimierungsprobleme zugeschnitten und angepasst werden können “ [167, S.198].
Abbildung 3.8 stellt vier der am häufigsten verwendeten Crossover-Operatoren vor. Für die Rekombination wurden exemplarisch die Individuen y_1 und y_3 aus Tabelle 3.1 ausgewählt. Beim One-Point Crossover wird zufällig ein Punkt innerhalb des Genoms ausgewählt, und alle Gene, die dahinter liegen werden wechselseitig ausgetauscht. Diese Idee kann vom Two-Point Crossover bis zum Multi-Point Crossover erweitert werden. Bei diesen Varianten werden alle Gene, die zwischen zwei zufällig ausgewählten Punkten liegen, ausgetauscht. Im Gegensatz dazu findet beim Uniform-Crossover ein zufälliger Austausch an jedem Locus statt. Die Crossover-Wahrscheinlichkeit p_{cross} bestimmt, ob zwei für die Rekombination ausgewählte Individuen miteinander rekombiniert werden.

[13] Für $n = 2$ spricht man von einer binären „Wettkampf“-Selektion (binary tournament).



(a) Eltern (b) One-Point Crossover (c) Two-Point Crossover (d) Uniform Crossover

Abbildung 3.8: Beispiel Crossover-Operatoren

Abbildung 3.9: Beispiel Flip-Mutation

Findet keine Rekombination statt, werden die Eltern-Genome ohne Veränderungen an die Kinder übertragen (vgl. [73, S. 64f]). Eine weiterführende Betrachtung zu Crossover-Operatoren bieten Eiben [58, S. 46ff] und Nissen [145, S. 51ff].
In einem GA werden einige Individuen nach der Rekombination einer Mutation unterzogen. Mit Hilfe des Mutationsoperators werden kleine zufällige Veränderungen am Genotyp durchgeführt. Durch den Einsatz der Mutation erfolgt eine lokale Suche. Die Mutation verhindert gleichzeitig eine frühzeitige Konvergenz der Population mit einer Vielzahl gleichartiger Lösungen. Die Mutationswahrscheinlichkeit p_{mut} bestimmt die Wahrscheinlichkeit dafür, dass eine Lösung einer Mutation unterworfen wird. Da die Mutation für den GA eine untergeordnete Rolle einnimmt, sollte diese Wahrscheinlichkeit gering gewählt werden [162, S. 18]. Abbildung 3.9 zeigt die Mutation eines Bit-Strings an Loci 1 und 3. Mittels einer Flip-Mutation werden die Werte der Allele zufällig auf einen zulässigen Wert aus dem Alleleset verändert.
An die Erzeugung der Nachkommen mittels Rekombination und der Veränderung einiger dieser Nachkommen mittels Mutation schließt sich die Erstellung der neuen Population an. Holland [88] und Goldberg [73] schlagen für einen einfachen (kanonischen) GA die komplette Ersetzung der Elterngeneration durch die Kindergeneration vor. De Jong [44] zeigt, dass durch die Übernahme guter Lösungen der Elterngeneration in die nächste (Nachkommen-)Generation die Performance des GAs gesteigert werden kann. Mit den sogenannten Steady-State-GAs wird dieses Konzept umgesetzt. In einem Steady-State-GA wird die Population aus den besten Individuen der Nachkommenpopulation und den besten Individuen der Elterngeneration aufgebaut. Auf diese Weise wird sichergestellt, dass gute Lösungen über mehrere Generationen hinweg in der Population erhalten bleiben. Mit Abbildung 3.10 wird die Erstellung der Population für die nächste Generation bei einem Steady-State-GA grafisch dargestellt. Ausgehend von der Elterngeneration P_t wird mittels der GA Operatoren die Kindergeneration P_t'' erstellt. Anschließend werden die besten Lösungen aus P_t und P_t'' ausgewählt und daraus die Elterngeneration P_{t+1} für die nächste Generation erstellt.

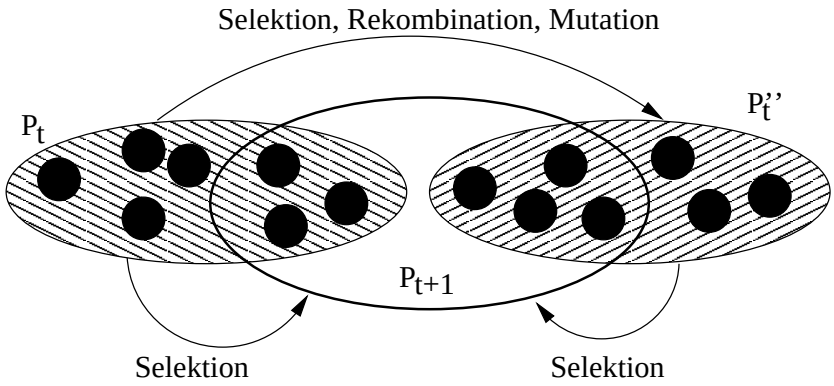
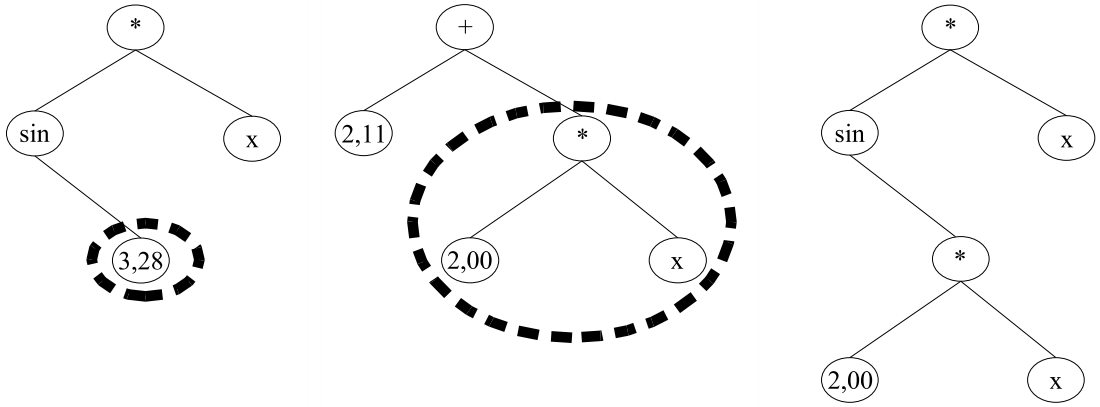


Abbildung 3.10: Erstellung einer neuen Population für einen Steady-State-GA



(a) Elter y_1 (b) Elter y_2 (c) Kind y_3

Abbildung 3.11: Beispiel Repräsentation und Rekombination für genetisches Programmieren [140, S. 286]

3.5.2.4 Genetisches Programmieren

Im Gegensatz zu den bisher vorgestellten evolutionären Verfahren liegt der Ursprung des letzten Vertreters – dem genetischen Programmieren (GP) – in der jüngeren Vergangenheit. Erstmals vorgestellt wurde das Konzept des GP 1990 von Koza [114]. Ähnlich wie das evolutionäre Programmieren löst das GP nicht das Problem an sich, sondern beschäftigt sich mit der Fragestellung nach einem guten (optimalen) Programm zur Lösung des Problems. Michalewicz [140, S. 286f] beschreibt das GP als „... a population of executable computer programs is created, individual programs compete against each other, weak programs die, and strong ones reproduce...“. Die Bewertung eines Genoms erfolgt mittels einer Menge an vordefinierten Testfällen. Die Bewertungsfunktion berechnet die Summe der Abstände aus den korrekten und den durch das kodierte Programm

erzielten Ergebnissen [140, S. 286]. Die Wahrscheinlichkeit, dass eine Lösung für die Rekombination ausgewählt wird, ist proportional zu ihrer Güte. GP arbeitet mitunter mit sehr großen Populationsgrößen. Für Populationen mit $|P| > 1000$ wird häufig die sogenannte Over-Selection-Methode eingesetzt. Hierfür wird die Population nach dem Fitnessranking in zwei Gruppen unterteilt und Lösungen für die Rekombination nach einer zuvor festgelegten Gewichtung aus beiden Gruppen gewählt. Details zur Over-Selection-Methode findet man bei Eiben [58, S. 109].
Eine Problemlösung wird beim GP typischerweise mittels eines hierarchisch strukturierten Computerprogramms, welches sich während der Evolution dynamisch ändert, repräsentiert [115, S.80]. In Abbildung 3.11 ist die Repräsentation dreier Lösungen dargestellt. Die Lösungen y_1, y_2, y_3 (Abbildung 3.11(a)-Abbildung 3.11(c)) stehen für die Ausdrücke $f(x) = x \cdot \sin(3, 28)$, $f(x) = 2 \cdot x + 2, 11$ und $f(x) = x \cdot \sin(2 \cdot x)$. Die Abbildung zeigt ebenfalls den Rekombinationsmechanismus des GP. Hier wird der beim GP häufig verwendete Austausch von Teilbäumen dargestellt. Ausgehend von den Eltern y_1 und y_2 werden bei der Rekombination Teilbäume (hier gestrichelt dargestellt) beider Lösungen ausgetauscht. In Abbildung 3.11(c) wird die dabei entstehende Lösung y_3 dargestellt. Für die Mutation setzt das GP ebenfalls auf zufällige Veränderungen einer Lösung. Ein typisches Vorgehen ist dabei ausgehend von einem zufällig gewählten Knoten die zufällige Ersetzung des Teilbaums unterhalb des gewählten Knotens [58, S. 107]. Eine detaillierte Beschreibung zum GP geben [115, 116, 117].

3.5.3 Genetische Algorithmen als Metaheuristik für die Kommunikationsnetzwerkplanung

Die Zielsetzung der Arbeit ist es, evolutionsbasierte Verfahren zur Entscheidungsunterstützung bei der Kommunikationsnetzwerkplanung mit den Zielen Zuverlässigkeit und Kosten zu entwerfen. Mit den vorangegangenen Abschnitten wurden die unterschiedlichen evolutionären Verfahren ES, EP, GA und GP vorgestellt.
In Abschnitt 3.1.1 wurde das im Rahmen der Arbeit untersuchte Problem als kombinatorisches Optimierungsproblem vorgestellt. Das Problem besitzt diskrete Entscheidungsvariablen (die auszuwählenden Verbindungen und deren Technologieoptionen). Evolutionäre Verfahren wie ES und EP sind aufgrund ihres primären Einsatzes für Probleme mit kontinuierlichen Entscheidungsvariablen als eher ungeeignet anzusehen. Ebenso ist das GP, welches die Zielsetzung hat, einen Algorithmus zu finden, der ein Problem gut löst, für die betrachtete Problemstellung nicht geeignet.
Der Einsatz von GAs für das untersuchte Planungsproblem scheint sinnvoll: Für GAs wurde in der Literatur eine Vielzahl von unterschiedlichen Repräsentationen vorgeschlagen, die es ermöglichen, unterschiedlichste Phänotypen als Genotypen für einen GA zu kodieren. Für die im Rahmen der Arbeit untersuchte Problemstellung kann beispielsweise mittels eines Integer-Strings die Verbindungsmatrix für ein Kommunikationsnetzwerk kodiert werden. Der erfolgreiche Einsatz von GAs für die Kommunikationsnetzwerkplanung in der Vergangenheit [52, 51, 49, 63, 14, 159, 158, 122] zeigte, dass diese eine geeignete Metaheuristik darstellen.

3.5.4 Ansätze zur Berücksichtigung von Restriktionen im Design genetischer Algorithmen

In dem vorgestellten Grundkonzept (Algorithmus 3.3 auf S. 44f) für GAs fehlt es an der Möglichkeit, Restriktionen für die Ziele (wie z. B. ein maximal zur Verfügung stehendes Budget) im Design des GAs zu berücksichtigen. Für Problemstellungen mit Nebenbedingungen[14] hinsichtlich einer oder mehrerer Zielgrößen der Art:

$$f_1(y) \rightarrow \min \text{ und } \forall_{i=2}^{n} f_i(y) \leq nb_i$$

sind daher Anpassungen des GAs notwendig.
Die Behandlung von Nebenbedingungen in GAs wurde in der Vergangenheit intensiv untersucht. Im Allgemeinen lassen sich zwei grundlegende Konzepte (direkte und indirekte Behandlung) für die Berücksichtigung von Nebenbedingungen unterscheiden [140, S. 121ff] [40]. Bei der indirekten Behandlung einer Nebenbedingung wird diese direkt in die Fitnessfunktion mit einbezogen. Für den Fall, dass eine Lösung eine Nebenbedingung verletzt, wird deren Fitnesswert in Abhängigkeit von der Abweichung von der Bedingung verschlechtert. Die Ausführungen in Abschnitt 5.1.1 auf S. 88 zeigen, dass das Finden eines geeigneten Strafterms oftmals sehr kompliziert ist und der Einsatz von Straftermen die Suche mitunter in eine falsche Richtung lenken kann. Eine weitergehende Untersuchung zu den Herausforderungen beim Einsatz von Straftermen findet man bei Gottlieb [76]. Im Gegensatz dazu wird bei der direkten Berücksichtigung von Nebenbedingungen der GA um zusätzliche Mechanismen erweitert. Typische Vorgehen hierfür sind (vgl. [189, S. 172ff]):

- Ungültige Lösungen werden aus der Population entfernt.
- Durch den Einsatz spezieller Repräsentationen wird sichergestellt, dass Lösungen stets gültig sind.
- Die genetischen Operatoren (Initialisierung, Rekombination und Mutation) stellen sicher, dass nur gültige Lösungen erzeugt werden.
- Ungültige Lösungen werden durch eine Reparaturfunktion in eine gültige Lösung überführt.

Die erste Möglichkeit (das Entfernen der Lösung aus der Population) stellt eine sehr ineffektive Methode dar, da hierdurch möglicherweise gutes genetisches Material, welches durch Rekombination oder Mutation eine gute neue Lösung hervorbringen könnte, frühzeitig im Evolutionsprozess verloren geht [40, 102]. Das Finden einer Repräsentation, welche sicherstellt, dass nur gültige Lösungen innerhalb der Population existieren, ist häufig sehr schwer. Exemplarisch sei für den Einsatz von problemspezifischen Repräsentationen und Operatoren auf Cheng [33] verwiesen. Dort wird eine problemspezifische Repräsentation sowie ein modifizierter Crossover-Operator für die Planung

[14] Eine Verallgemeinerung der hier dargestellen Minimierung einer Zielgröße und einer oberen Schranke für alle anderen Zielgrößen auf Maximierungsprobleme sowie Mindestanforderungen für weitere Zielkriterien ist möglich.

von kostenminimalen Kommunikationsnetzwerken unter Berücksichtigung des 2-fach-Zusammenhangs als Nebenbedingung für die entworfene Netzwerktopologie vorgestellt. Arbeiten, in denen eine solche Repräsentation oder spezielle Operatoren für die Verwendung der All-Terminal-Zuverlässigkeit, welche als Zuverlässigkeitsmaß im Rahmen dieser Arbeit verwendet wird, vorgeschlagen werden, wurden bisher nicht veröffentlicht. Mit dem Einsatz einer Reparaturfunktion wird sichergestellt, dass eine Population nur gültige Lösungen enthält. Der Einsatz eines solchen Verfahrens verlangsamt jedoch die Ausführung des GAs, da stets jede ungültige Lösung in eine gültige Lösung zu überführen ist. Für eine Vielzahl von kombinatorischen Optimierungsproblemen wurde in den letzten Jahren der Einsatz von Reparaturheuristiken vorgeschlagen [141, S. 239]. Anwendungen von Reparaturheuristiken für das hier betrachtete Planungsproblem findet man u. a. in [52, 49, 159, 158] sowie in den Kapiteln 5–7 dieser Arbeit.

3.5.5 Evolutionäre Verfahren für multikriterielle Optimierungsprobleme

Bisher wurden evolutionäre Verfahren zur Lösung von Problemstellungen mit einer Zielgröße betrachtet. Mit diesem Abschnitt werden evolutionäre Verfahren vorgestellt, welche die Lösung von Problemstellungen mit mehreren Zielfunktionen (welche mit Abschnitt 3.3 auf S. 32 eingeführt wurden) vorgestellt.
Das Potenzial evolutionärer Verfahren zur Lösung von MKOP wurde bereits Ende der 60er Jahren von Rosenberg [161] erkannt. Die erste Implementierung eines multikriteriellen evolutionären Verfahrens, des Vector Evaluated Genetic Algorithm (VEGA), wurde 1984 von Schaffer [166] vorgestellt. VEGA wurde von Schaffer für das maschinelle Lernen entwickelt. Heutzutage existieren eine Reihe von unterschiedlichen multikriteriellen evolutionären Algorithmen wie z. B. NSGA-II [47] und SPEA-2 [195], die in einer Vielzahl von Anwendungsdomänen eingesetzt werden. Eine gute Übersicht zu den verschiedenen Verfahren bietet Deb [45].
Für den Aufbau eines multikriteriellen evolutionären Algorithmus stellen Zitzler u. a. [194] drei wichtige Ziele heraus:

- die Zuweisung eines skalaren Fitnesswertes,
- die Sicherstellung einer Diversifikation in der Population und
- die Speicherung pareto-optimaler Lösungen während der Suche

3.5.5.1 Fitnesszuweisung für multikriterielle Optimierung

Für die Fitnesszuweisung in einem multikriteriellen EA wird zwischen der aggregierten Fitnesszuweisung, der kriterienbasierten Fitnesszuweisung und der pareto-basierten Fitnesszuweisung unterschieden. Bei der aggregierten Fitnesszuweisung wird der Fitnesswert jedes Individuums mittels eines Gewichtungsvektors für die einzelnen Zielkriterien auf einen skalaren Fitnesswert (als gewichtete Summe) umgerechnet. Die Gewichte der einzelnen Fitnessfunktionen werden dabei während der Optimierung verändert, um sicherzustellen, dass das Verfahren nicht nur eine einzige Lösung generiert. Anwendungen

dieser Art der Fitnessberechnung findet man beispielsweise beim Multi Objective Genetic Local Search [94]. Die erste kriterienbasierte Fitnesszuweisung für die multikriterielle evolutionäre Optimierung wurde von Schaffer [166] mit VEGA vorgestellt. Für die Fitnessbewertung wird bei VEGA die Population in i (wobei i gleich der Anzahl der Fitnessfunktionen ist) Teilpopulationen zerteilt. Jede dieser Teilpopulationen wird hinsichtlich eines anderen Fitnesskriteriums bewertet. Dabei gilt: Für Lösungen in der ersten Teilpopulation entspricht der Fitnesswert dem Wert der ersten Zielfunktion; Lösungen der zweiten Teilpopulation wird eine Fitness entsprechend dem zweiten Zielfunktionswert zugewiesen usw. Die Auswahl der Lösungen für die Rekombination erfolgt in jeder Teilpopulation separat auf Basis der zugewiesenen Fitnesswerte. Bei der Analyse der Arbeiten von Schaffer erkannte Goldberg [73], dass auf Basis der Dominanz einer Lösung über die Menge von anderen Lösungen die Zuweisung eines skalaren Fitnesswertes für jede Lösung möglich ist. Goldberg schlägt in [73, S. 200f] eine Non-Dominated Sorting Prozedur zur Bestimmung der Fitness einer Lösung vor. Einige Verfahren wie z. B. NSGA-II [47] zerlegen die Population hierfür in einzelne Sub-Populationen (bei NSGA-II als Fronten[15] bezeichnet)und führen die Fitnesszuweisung anhand der Front, zu der eine Lösung gehört, durch. Andere Ansätze wie z. B. SPEA-2 [195] beziehen in die Berechnung der Fitness die Anzahl der von der Lösung dominierten Lösungen sowie den Rang der Lösung mit ein. Ein entscheidender Vorteil der pareto-basierten Fitnesszuweisung ist, dass bei der Fitnessbewertung einer Lösung stets sämtliche Zielfunktionswerte gleichzeitig betrachtet werden und ein Vergleich der Lösung zu allen anderen Lösungen der Population durchgeführt wird.

3.5.5.2 Sicherstellung der Diversifikation während der Suche

Die meisten multikriteriellen EA versuchen durch die Einbeziehung von Informationen über die Dichte von Lösungen an einem bestimmten Punkt den Selektionsprozess gezielt zu steuern und dadurch eine Diversifikation in der Menge der aktuellen pareto-optimalen Lösungen zu sichern [194]. Lösungen, in deren Umfeld sich eine Vielzahl anderer Lösungen befinden, erhalten einen geringeren Fitnesswert als Lösungen, die einen sehr großen Abstand zur nächsten Lösung in ihrer Nachbarschaft besitzen. Für die Bestimmung der Dichte von Lösungen werden die folgenden Verfahren eingesetzt:

- Kernel-Methode [174, S. 100ff],
- Nächster Nachbar [174, S. 96ff] und
- Histogramme [174, S. 7ff].

Viele Verfahren (z. B. MOGA [66] und NPGA [89]) setzen zur Erhaltung der Diversifikation in der Lösungsmenge das Fitness-Sharing-Konzept ein. Mittels einer Kernel-Methode wird hier die Distanz einer Lösung zu allen anderen Lösungen bestimmt. Hierfür wird für jedes Lösungspaar der Abstand zwischen den Zielfunktionswerten aufsummiert.

[15] Details zur Generierung der Fronten findet man in Abschnitt 3.5.6 auf S. 55.

Ein anderes Konzept der Distanzberechnung wird mit dem „nächster Nachbar“ (nearest neighbor)-Konzept verwendet. Hier werden lediglich k Lösungen in der Nachbarschaft einer Lösung betrachtet und daraus auf die Dichte der Lösungen geschlossen. Dieses Konzept der Dichtebestimmung wird beispielsweise bei SPEA-2 [195] eingesetzt.
Als dritte Möglichkeit bietet sich der Einsatz von Histogrammen an. Dabei wird der Lösungsraum in Hypergrids eingeteilt. Als Maß für die Dichte wird die Anzahl der Lösungen verwendet, die sich im gleichen Hypergrid befinden. Die Bestimmung der Hypergrids kann fest sein oder aber über die Generationen hinweg variieren. Einsatz findet die Methode u. a. bei PAES [108].

3.5.5.3 Elitismus zur Propagierung guter Lösungen

Das Problem, gute Lösungen während der Suche zu erhalten und über die Generationen hinweg zu propagieren, wurde bereits in Abschnitt 3.5.2.3 auf S. 43ff dargestellt. Dort wurde der Steady-State-GA vorgestellt, bei dem sich die Lösungen der nächsten Generation aus den besten Lösungen der Elterngeneration und der Kindergeneration zusammensetzen. Wie bei der monokriteriellen Suche auch, ist man bei multikriteriellen EAs bemüht, gute (pareto-optimale) Lösungen während des Suchprozesses zu erhalten. Ein Lösungsansatz ist auch hier die Möglichkeit des Aufbaus der nächsten Generation aus den besten Lösungen der Eltern- und Kinderlösungen. Eine solche Technik wird z. B. von NSGA-II [47] genutzt. Eine Alternative hierzu bildet der Einsatz eines Archives, in dem sämtliche während der Suche gefundenen pareto-optimalen Lösungen gespeichert sind. Hier wird für jedes Individuum einer Kindergeneration geprüft, ob dies eine neue pareto-optimale Lösung ist. Dabei wird die Lösung mit den Lösungen im Archiv verglichen. Ist die Lösung pareto-optimal zu Lösungen im Archiv, so wird sie zu diesem hinzugefügt. Werden dabei durch die hinzugefügte Lösung andere Lösungen im Archiv dominiert, so werden diese Lösungen aus dem Archiv entfernt. Da für die Speicherung sämtlicher pareto-optimaler Lösungen im Archiv unter Umständen nicht ausreichend Ressourcen zur Verfügung stehen, setzen multikriterielle Verfahren mit Archiv häufig zusätzliche Archivverwaltungstechniken ein, um die Menge der im Archiv gespeicherten Lösungen zu steuern. Ziel ist es dabei häufig, eine gute Streuung der Lösungen über die Pareto-Front zu erreichen. Hierfür werden Lösungen in dicht besetzten Bereichen mittels Clustering zu einer Lösung zusammengezogen und damit der Speicherbedarf des Archives verringert. Zum Einsatz kommt die Archivtechnik z. B. bei SPEA-2 [195].

3.5.5.4 Grundform eines multikriteriellen evolutionären Algorithmus

Der prinzipielle Ablauf eines multikriteriellen evolutionären Algorithmus (ohne Archiv) wird mit Algorithmus 3.4 dargestellt. Als Erweiterung zu einem einfachen (monokriteriellen) EA wird für den multikriteriellen EA zusätzlich die Vektor/Fitness-Transformation eingeführt. Da bei der multikriteriellen Suche eine Lösung hinsichtlich einer Menge unterschiedlicher Zielfunktionen f_i zu bewerten ist, wird zunächst für jede der Zielfunktionen f_i der Zielfunktionswert $f_i(y)$ einer Lösung y berechnet. Mittels der Vektor/Fitness-Transformation ist es möglich, den Fitnessvektor mit den Zielfunktionswer-

Algorithmus 3.4 Multikriterieller evolutionärer Algorithmus (in Anlehnung an [35, S. 52])

$t = 0$
Erzeuge initiale Population P_t
Bewerte P_t mittels skalarer Fitnessfunktion {vgl. Abschnitt 3.5.5.1}
repeat
  P_t^* = Selektiere Individuen aus P_t
  P_t' = Rekombiniere P_t^*
  P_t'' = Mutiere P_t'
  Bewerte P_t'' mittels skalarer Fitnessfunktion {vgl. Abschnitt 3.5.5.1}
  P_{t+1} = Wähle gute Individuen aus P_t und P_t''
  t = t+1
until Abbruchkriterium

ten auf einen Wert zu skalieren und auf diese Weise die Lösungen in der Population miteinander zu vergleichen.

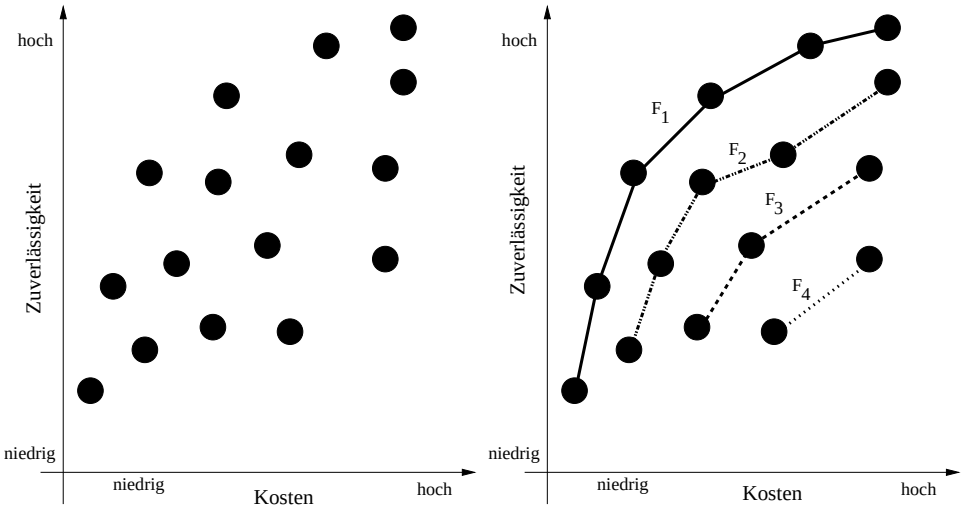
3.5.6 NSGA-II – Eine populationsbasierte Metaheuristik für multikriterielle kombinatorische Optimierungsprobleme

Mit Abschnitt 3.5.5 wurden die Anforderungen an einen multikriteriellen EA beschrieben. Als ein Vertreter, der all diesen Anforderungen genügt, wird im Folgenden der von Deb u. a. [47] vorgeschlagene Non-Dominated Sorting Algorithm (kurz als NSGA-II bezeichnet) im Detail vorgestellt. Srinivas und Deb schlugen die Non-Dominated Sorting Technik erstmals 1994 mit dem NSGA [179] vor. Aufbauend hierauf und unter Berücksichtigung verschiedener Kritikpunkte (siehe hierzu [47]) entstand das NSGA-II-Verfahren.
Bevor die Hauptschleife des NSGA-II-Algorithmus vorgestellt wird, werden zunächst zwei elementare Funktionen eingeführt. Mit Algorithmus 3.5 wird die Prozedur zum sogenannten Fast-Nondominated Sort gezeigt. Ziel der Methode ist es, die Lösungen in einer Population P hinsichtlich ihrer Dominanz über andere Lösungen zu sortieren. Durch die Methode werden die Lösungen in verschiedene Teillösungsmengen $(\mathcal{F}_1, \mathcal{F}_2 \ldots \mathcal{F}_n)$ aufgeteilt. Eine solche Teilmenge an Lösungen wird als Front bezeichnet. Das Verfahren geht dabei wie folgt vor: Zu Beginn werden alle Lösungen der Population miteinander verglichen. Dabei werden mit der Variablen n_{y_i} die Anzahl der Lösungen, welche die Lösung y_i dominieren gezählt und in der Menge S_{y_i} die Lösungen, welche durch die Lösung y_i dominiert werden, abgespeichert. Sämtliche Lösungen mit $n_{y_i} = 0$ gehören zur Pareto-Front und werden deshalb in der Front $\mathcal{F}_1$ gespeichert. Nach der Bildung der Front $\mathcal{F}_1$ werden deren Lösungen temporär aus der Lösungsmenge entfernt. Hierfür wird für jede verbleibende Lösung der Wert für die Anzahl der sie dominierenden Lösungen (n_{y_i}) angepasst. Alle Lösungen mit $n_{y_i} = 0$ bilden die nächste Front. Anschließend werden wiederum die Lösungen aus der Lösungsmenge entfernt und mit der Aktualisierung der verbleibenden Lösungen fortgefahren. Der Algorithmus stoppt, wenn sämtliche Lösungen einer Front zugeordnet wurden. Nach Anwendung von Algorithmus 3.5 befinden sich alle pareto-

Algorithmus 3.5 Fast-Nondominated Sort für NSGA-II (nach [47])

Eingabe: Population P
  for all $y_p \in P$ **do**
    for all $y_q \in P$ **do**
      if $y_p \succeq y_q$ **then**
        $\mathcal{S}_{y_p} = \mathcal{S}_{y_p} \cup y_q$
      else if $y_q \succeq y_p$ **then**
        $n_{y_p} = n_{y_p} + 1$
      end if
    end for
    if $n_{y_p} = 0$ **then**
      $\mathcal{F}_1 = \mathcal{F}_1 \cup y_p$
    end if
  end for
  $i = 1$
  while $\mathcal{F}_i \neq \emptyset$ **do**
    $\mathcal{H} = \emptyset$
    for all $y_p \in \mathcal{F}_i$ **do**
      for all $y_q \in \mathcal{S}_{y_p}$ **do**
        $n_{y_q} = n_{y_q} - 1$
        if $n_{y_q} = 0$ **then**
          $\mathcal{H} = \mathcal{H} \cup y_q$
        end if
      end for
    end for
    $i = i + 1$
    $\mathcal{F}_i = \mathcal{H}$
  end while

optimalen Lösungen in der Front $\mathcal{F}_1$. Für alle anderen Fronten gilt: Die Lösungen der Front $\mathcal{F}_i$ (mit $i > 1$) werden von den Lösungen der Front $\mathcal{F}_{i-1}$ dominiert und dominieren die Lösungen der Front $\mathcal{F}_{i+1}$. Mit Abbildung 3.12 wird dieser Zusammenhang für das bereits in Abschnitt 3.3 verwendete Beispiel grafisch dargestellt. Abbildung 3.12(a) zeigt eine Population P, welche als Eingabe für das Fast-Nondominated Sort dient. Durch die Anwendung von Algorithmus 3.5 wird die Lösungsmenge in vier in Abbildung 3.12(b) dargestellte Fronten aufgeteilt.
Zur Bestimmung der Lösungsdichte führen Deb u. a. [47] die Crowding-Distanz ein. Zur Berechnung der Crowding-Distanz für eine Lösung wird der mittlere Abstand der Lösung zu seinen jeweils benachbarten Lösungen betrachtet. Die Berechnung der Crowding-Distanz stellt Algorithmus 3.6 dar. Als Eingabe wird an den Algorithmus eine Menge von Lösungen $\mathcal{I}$ (z. B. die Lösungen einer zuvor identifizierten Front $\mathcal{F}_i$) übergeben, für welche die Crowding-Distanz $\mathcal{I}[y_i]_{Distanz}$ mit $y_i \in \mathcal{I}$ berechnet wird. Durch die Methode wird zu Beginn die Distanz für jede Lösung $\mathcal{I}[y_i]_{Distanz}$ gleich Null gesetzt. Anschließend wird für jede Lösung y_i und für jede Zielfunktion f_m die Distanz zu den beiden



(a) Lösungsmenge

(b) Lösungsmenge nach NSGA-II-Frontenbildung

Abbildung 3.12: Beispiel Frontenbildung NSGA-II

Algorithmus 3.6 Crowding-Distanz Berechnung für NSGA-II (nach [47])

Eingabe: $\mathcal{I}$
  $l = |\mathcal{I}|$
  for all $y_i \in \mathcal{I}$ **do**
    $\mathcal{I}[y_i]_{Distanz} = 0$
  end for
  for all Zielfunktion m **do**
    $\mathcal{I} =$ sort$(\mathcal{I}, m)$
    $\mathcal{I}[y_1]_{Distanz} = \mathcal{I}[y_l]_{Distanz} = \infty$
    for $i = 2$ to $(l-1)$ **do**
      $\mathcal{I}[y_i]_{Distanz} = \mathcal{I}[y_i]_{Distanz} + (f_m(\mathcal{I}[y_{i+1}]) - f_m(\mathcal{I}[y_{i-1}]))$
    end for
  end for

benachbarten Lösungen berechnet. Hierfür wird die Lösungmenge nach dem jeweils betrachteten Zielkriterium sortiert. Für die Lösungen mit dem kleinsten bzw. größten Zielfunktionswert gilt $\mathcal{I}[y_1]_{Distanz} = \mathcal{I}[y_l]_{Distanz} = \infty$. Für alle anderen Lösungen wird die Distanz jeweils um den Abstand zwischen den Zielfunktionswerten der beiden benachbarten Lösungen (y_{i-1} und y_{i+1}) erhöht.
Abbildung 3.13 veranschaulicht die Bestimmung der Distanz für die Lösung y_i bei $\mathcal{I} = \mathcal{F}_1$. Als Distanzmaß für y_i ergibt sich: $\mathcal{I}[y_i]_{Distanz} = f_{Kosten}(\mathcal{I}[y_{i+1}]) - f_{Kosten}(\mathcal{I}[y_{i-1}]) + f_{Zuver.}(\mathcal{I}[y_{i+1}]) - f_{Zuver.}(\mathcal{I}[y_{i-1}])$.
Aufbauend auf der Crowding-Distanz setzt NSGA-II den Crowding-Comparator ein, welcher die Aufgabe hat, die Diversifikation während der Suche sicherzustellen. Beim

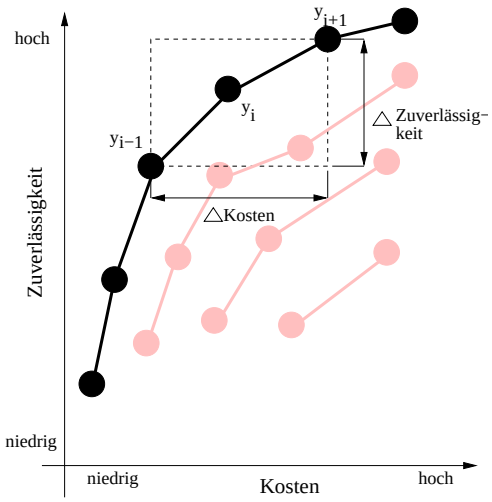


Abbildung 3.13: Beispiel Crowding Distanz-Berechnung mit NSGA-II

Vergleich zweier Lösungen werden deshalb die Kriterien Rang einer Lösung sowie Crowding-Distanz einer Lösung herangezogen. Für den Crowding-Comparator $\geq_n$ gilt:

$$\begin{aligned} y_i \geq_n y_j \text{ wenn } & (rang_{y_i} < rang_{y_j}) \\ & \text{oder } ((rang_{y_i} = rang_{y_j}) \text{ und } (\mathcal{I}[y_i]_{Distanz} > \mathcal{I}[y_j]_{Distanz})). \end{aligned} \tag{3.2}$$

Der Rang ($rang_{y_n}$) einer Lösung y_n entspricht dabei der Nummer der Front $\mathcal{F}$, in der sich die Lösung befindet. Eine Lösung y_i wird einer Lösung y_j gegenüber präferiert, wenn die Lösung y_i einen höheren Rang als y_j besitzt. Besitzen beide Lösungen den gleichen Rang, so präferiert der Crowding-Comparator die Lösung, die eine geringere Dichte aufweist, d.h. in deren Umgebung sich weniger andere Lösungen befinden.
Die bisher vorgestellten Methoden dienen dem NSGA-II als Hilfsfunktionen für den eigentlichen Hauptalgorithmus. Mit Algorithmus 3.7 wird der vollständige Ablauf von NSGA-II dargestellt. Mittels der Fast-Nondominated Sort-Methode wird jeder Lösung in der Startpopulation P_0 ein Fitnesswert entsprechend der ermittelten Front zugewiesen. Anschließend wird mittels „Wettkampf"-Selektion, Rekombination und Mutation die neue Generation P'_0 gebildet. In der Hauptschleife des Verfahrens wird die Kindergeneration P'_t mit der Elterngeneration P_t zu P''_t vereint. Die Größe von P''_t entspricht dabei der doppelten Populationsgröße. Den Lösungen der Population P''_t wird mittels der Fast-Nondominated Sort Methode ein Fitnesswert entsprechend ihrem Dominanzrang (der Front zu der sie gehören) zugewiesen. Die nächste Elternpopulation P_{t+1} wird aus den besten Lösungen (diejenigen mit dem höchsten Rang) aufgebaut. Hierfür werden solange die Fronten $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n$ zu P_{t+1} hinzugefügt, wie die Anzahl der Lösungen in P_{t+1} kleiner der vorgegebenen Populationsgröße *pop* ist. Für die Front $\mathcal{F}_i$, durch welche die Anzahl der Lösungen in P_{t+1} die vorgegebene Populationsgröße übersteigt, werden die Lösungen mit Hilfe des Crowding-Comparators sortiert und nur die besten Lösungen (bis $|P_{t+1}| = pop$) zu P_{t+1} hinzugefügt.

Algorithmus 3.7 Non-Dominated Sorting Genetic Algorithm (NSGA-II) (in Anlehnung an [47])

$t = 0$
Erzeuge initiale Population P_t
Fast-Nondominated-Sort P_t
Erzeuge P'_t aus P_t
repeat
  $P''_t = P'_t \cup P_t$
  Fast-Nondominated-Sort P''_t
  i := 1
  while $(|P_{t+1}| + |\mathcal{F}_i|) < pop$ **do**
    $P_{t+1} \cup \mathcal{F}_i$
    i:= i +1
  end while
  Sort$(\mathcal{F}_i, \geq_n)$
  $P_{t+1} = \mathcal{F}_i[0 : pop - |P_{t+1}|]$
  Erzeuge P'_{t+1} aus P_{t+1}
  $t = t + 1$
until Abbruchkriterium

Wie bereits für die initiale Population P_0 wird mittels binärer „Wettkampf"-Selektion[16], Rekombination und Mutation die neue Kindergeneration P' gebildet. Als Selektionskriterium wird dabei der Crowding-Comparator verwendet. Die Hauptschleife wird solange wiederholt, bis ein Abbruchkriterium[17] erfüllt ist.
Mit der Vorstellung der einzelnen Methoden des NSGA-II wird deutlich, dass das Verfahren alle die in Abschnitt 3.5.5 aufgestellten Anforderungen an einen multikriteriellen evolutionären Algorithmus erfüllt. Durch die Auswahl der besten Lösungen aus Eltern- und Kinderpopulation wird der Anforderung an die Propagierung guter Lösungen Rechnung getragen. Die Verwendung des Crowding-Comparators erlaubt die Einbeziehung der Dichte von Lösungen bei der Selektion. Der Einsatz der Fast-Nondominated Sort Methode ermöglicht eine Fitnesszuweisung, die sämtliche Zielfunktionswerte einbezieht. Die im Rahmen der Arbeit entwickelten Anwendungen des NSGA-II-Verfahrens für die multikriterielle Netzwerktopologieplanung werden in Kapitel 7 vorgestellt.

3.6 Bisherige Ansätze zur Planung von Kommunikationsnetzwerken

Die Planung von Kommunikationsnetzwerken wurde in der Vergangenheit bereits intensiv untersucht. Mit den nachfolgenden Abschnitten findet eine Analyse bisheriger Planungsansätze und deren Defizite statt. Es wird dabei untersucht, inwieweit bisherige

[16] Die „Wettkampf"-Selektion verwendet für den Vergleich zweier Lösungen den Crowding-Comparator $\geq_n$.
[17] Als Abbruchkriterium eignet sich z. B. die maximale Anzahl an Generationen.

Ansätze zur Kommunikationsnetzwerkplanung die im Rahmen dieser Arbeit untersuchten Planungsziele Zuverlässigkeit (unter Verwendung der All-Terminal-Zuverlässigkeit als Planungsziel) und Kosten (Installationskosten für eine Netzwerktopologie) berücksichtigen.
Verwandte Anwendungsfelder der im Rahmen der Arbeit untersuchten Problemstellung sind z.B. die Planung von Gas-/Öl-Pipelines sowie die Planung von Netzwerken für die Wasser-/Energieversorgerung. Arbeiten zu diesen Anwendungsgebieten findet man unter anderem bei Goldberg [73, S. 125ff], Backes u.a. [9], Prasad und Park [150] sowie Cantoni u.a. [133].

3.6.1 Bisherige Ansätze zur Planung von Kommunikationsnetzwerken mit einer Zielgröße

Mit diesem Abschnitt werden bisherige Arbeiten zur Planung von zuverlässigen und ökonomischen Kommunikationsnetzwerken mit einer Zielgröße (Kosten oder Zuverlässigkeit) betrachtet. Im Folgenden werden sowohl Arbeiten für Probleme mit Technologieoptionen als auch Arbeiten für Probleme ohne Technologieoptionen vorgestellt.
Als exaktes Lösungsverfahren wird in [96] ein „Branch and Bound"-Algorithmus für die Planung von kostenminimalen Netzwerktopologien unter Berücksichtigung einer Zuverlässigkeitsnebenbedingung (welche die All-Terminal-Zuverlässigkeit der Netzwerktopologie berücksichtigt) vorgestellt. Das Verfahren wird lediglich für Netzwerke mit maximal 20 Knoten und 30 Verbindungen sowie identischen Zuverlässigkeiten (ohne Technologieoptionen) für sämtliche Verbindungen des Netzwerkes untersucht. Koide u.a. [110] erweitern den „Branch and Bound"-Algorithmus aus [96] für die Planung mit unterschiedlichen Technologieoptionen je Verbindung. Der in [110] vorgestellte „Branch and Bound"-Algorithmus wird lediglich für kleine Netzwerke mit sieben Knoten und 21 Kanten untersucht.
Wie in Abschnitt 3.1.1 auf S. 27 ausgeführt, wird das im Rahmen der Arbeit untersuchte Problem in die Klasse der NP-schweren Probleme eingeordnet. Für diese Probleme gilt, dass die Laufzeit der Algorithmen zur Lösung des Problems mit der Größe der Eingabedaten exponentiell ansteigt. Für diese Problemklasse existieren keine exakten Verfahren, welche die Probleme dieser Problemklasse effizient lösen. Als Approximationsmethoden stellen die Metaheuristiken geeignete Methoden bereit, um NP-schwere Probleme in akzeptabler Zeit mit einer akzeptablen (nicht optimalen) Lösungen zu lösen (vgl. hierzu die Ausführungen in Abschnitt 3.3.2). Im Folgenden werden deshalb unterschiedliche in der Literatur veröffentlichte Verfahren (welche Metaheuristiken verwenden) betrachtet. Eine Übersicht zu verschiedenen Metaheuristiken für die Planung von ausfallsicheren und ökonomischen Netzwerken findet man bei Altiparmak u.a. [6].
Für die Planung von Ringtopologien, unter Berücksichtigung von Routing und Kapazitäten schlagen White u.a. in [1] eine spezielle Repräsentation für einen GA vor, die alle dort behandelten Designaspekte gleichzeitig abbilden kann. Durch die Verwendung einer Ringtopologie wird sichergestellt, dass jede der durch das vorgeschlagene Verfahren generierte Lösung 2-fach-kantenzusammenhängend[18] ist. Das weitergehende Zuverlässigkeitsmaß All-Terminal-Zuverlässigkeit wird jedoch nicht berücksichtigt.

[18] Eine Beschreibung dieses Zuverlässigkeitsmaßes bietet Abschnitt 4.3.1 auf S. 70.

Eine Anwendung der Tabu Search Metaheuristik für die Netzwerkplanung zeigen Glover u. a. [71]. Die Autoren verwenden als Planungsziel die Kosten, die die Installation einer neuen Netzwerktopologie verursacht. Die Robustheit des Netzwerkes gegenüber Störungen und Ausfällen der Netzwerkkomponenten, wie sie mit Hilfe der im Rahmen dieser Arbeit verwendeten Zuverlässigkeitsmaße stattfindet, wird in [71] nicht betrachtet.
Aggarwal u. a. [3] veröffentlichten eine Heuristik zur Maximierung der All-Terminal-Zuverlässigkeit von Kommunikationsnetzwerken unter Berücksichtigung einer Budgetvorgabe. Das Verfahren setzt für alle Verbindungen des Netzwerkes identische Zuverlässigkeiten voraus.
Einen Simulated Annealing basierten Ansatz für die Maximierung der Zuverlässigkeit (All-Terminal-Zuverlässigkeit) einer Netzwerktopologie mit unterschiedlichen Verbindungzuverlässigkeiten (Technologieoptionen) unter Beachtung einer Budgetvorgabe schlagen Atiqullah und Rao in [8] vor. Das vorgeschlagene Verfahren wird jedoch lediglich für sehr kleine Netzwerke mit vier Knoten und sechs Verbindungen untersucht.
In [69] schlagen Ghosh u. a. einen GA für die Planung von ausfallsicheren Kommunikationsnetzwerken unter Verwendung von Budgetvorgaben vor. Als Maß für die Ausfallsicherheit wird der 2-fach-Zusammenhang der entworfenen Netzwerktopologie verwendet. Ausfallwahrscheinlichkeiten von Verbindungen fließen in die Betrachtung nicht ein. Lösungen, die mittels der GA-Operatoren generiert werden und die Budgetvorgabe nicht erfüllen, werden durch den GA verworfen.
Ein Simulated Annealing-Ansatz zur Erweiterung einer bestehenden Netzwerktopologie mit dem Ziel der Zuverlässigkeitsmaximierung unter Verwendung einer Kostenrestriktion wird in [31] vorgestellt. In der Arbeit werden nur Problemstellungen mit identischen Zuverlässigkeiten (d.h. ohne Technologieoptionen) für sämtliche Verbindungen des Netzwerkes untersucht. Außerdem wird für die Bewertung der Zuverlässigkeit der entworfenen Netzwerktopologien lediglich die k-Terminal-Zuverlässigkeit (vgl. hierzu Abschnitt 4.3.2 auf S. 71) verwendet, welche im Vergleich zu der im Rahmen dieser Arbeit verwendeten All-Terminal-Zuverlässigkeit ein einfaches Zuverlässigkeitsmaß darstellt (vgl. hierzu Abschnitt 4.3.2).
Einen GA für die Planung von Netzwerktopologien mit minimalen Kosten unter Berücksichtigung der k-Terminal-Zuverlässigkeit findet man bei Liu und Iwanura [130]. Durch die Autoren werden dabei unterschiedliche k-Terminal-Zuverlässigkeiten zwischen verschiedenen Knoten im Netzwerk parallel betrachtet. Eine Untersuchung der All-Terminal-Zuverlässigkeit findet nicht statt.
Sayoud und Takahashi veröffentlichen in [165] einen GA für die Planung von kostenminimalen Netzwerktopologien. Als weitere Planungsgrößen werden die Kapazitäten der Verbindungen und die gegebenen Datentransferanforderungen beim Entwurf betrachtet. Die Ausfallsicherheit der entworfenen Topologie findet jedoch keine Berücksichtigung.
Hewitt u. a. schlagen in [86] den GA CHARLEY zur Erstellung einer Netzwerktopologie mit minimalen Kosten vor. Neben den Kosten werden beim Design zusätzlich Kapazitäten, Verfügbarkeit und Verzögerung betrachtet. Als Maß für die Zuverlässigkeit wird dabei lediglich der n-fach-Zusammenhang verwendet, welcher im Vergleich der im Rahmen dieser Arbeit verwendeten All-Terminal-Zuverlässigkeit ein deutlich einfacheres Zuverlässigkeitsmaß darstellt.

Arabas und Kozdrowski [7] stellen einen GA für die Planung kostenminimaler Netzwerktopologien unter Beachtung von Kommunikationsanforderungen und Aufrechterhaltung der Kommunikationsfähigkeit beim Ausfall einzelner Verbindungen vor. Statt der im Rahmen dieser Arbeit verwendeten All-Terminal-Zuverlässigkeit als Zuverlässigkeitsmaß wird in [7] lediglich der 2-fach-Zusammenhang als Zuverlässigkeitsanforderung an die zu entwerfenden Netzwerke verwendet.
Kumar u. a. [121] schlagen GAs für den Aufbau und die Erweiterung bestehender Netzwerke zur Verbesserung der Zuverlässigkeit[19] vor. Das im Rahmen der vorliegenden Arbeit untersuchte Zuverlässigkeitsmaß All-Terminal-Zuverlässigkeit wird mit diesen Arbeiten jedoch nicht berücksichtigt.
In [120] wird ein GA zur Maximierung der All-Terminal-Zuverlässigkeit unter Berücksichtigung einer Nebenbedingung für den maximalen Abstand zwischen zwei Knoten des Netzwerkes vorgeschlagen. Die Kosten der entworfenen Netzwerktopologie werden durch die Zielfunktion und die aufgestellte Nebenbedingung nicht berücksichtigt.
Clouqueur und Grover untersuchen in [34] die Planung von ausfallsicheren Netzwerken. Als Maß für die Ausfallsicherheit wird im Gegensatz zur vorliegenden Arbeit lediglich der 2-fach-Zusammenhang genutzt.
Einen Simulated Annealing-Ansatz für die Kommunikationsnetzwerkplanung wird in [177] vorgestellt. Die Autoren schlagen dabei ein mehrstufiges Entwurfsverfahren für die kombinierte kostenminimale Planung von Zugangs- und Backbone-Netzwerken vor. Zur Aufrechterhaltung der Kommunikationsfähigkeit des Netzwerkes bei Verbindungsfehlern werden alternative Kommunikationspfade beim Design berücksichtigt. Weiterreichende Zuverlässigkeitsmaße wie die All-Terminal-Zuverlässigkeit für die Knoten des Backbone-Netzwerkes werden in [177] jedoch nicht verwendet.
Einen auf dem Einsatz eines künstlichen neuronalen Netzes basierenden Ansatz für die Planung kostenminimaler Kommunikationsnetzwerke unter Verwendung einer Zuverlässigkeitsnebenbedingung findet man in [2]. Die Autoren verwenden als Zuverlässigkeitsmaß die All-Terminal-Zuverlässigkeit. Das Verfahren ist jedoch nur für Planungsprobleme mit identischen Zuverlässigkeiten (ohne Technologieoptionen) anwendbar.
In den letzten Jahren wurde das im Rahmen der vorliegenden Arbeit untersuchte Planungsproblem der Kommunikationsnetzwerkplanung unter Zuverlässigkeits- und Kostenaspekten von Smith und Dengiz intensiv erforscht: In [52] stellen Dengiz u. a. einen GA für die Planung kostenminimaler Netzwerktopologien unter Zuverlässigkeitsrestriktionen (einer minimal geforderten All-Terminal-Zuverlässigkeit der Netzwerktopologie) bei identischen Verbindungszuverlässigkeiten (ohne Technologieoptionen) vor. Baran und Laufer [15] erweitern den Ansatz aus [52] durch den Einsatz eines verteilten, parallelen GA. Einen Simulated Annealing-Ansatz für das gleiche Planungsproblem findet man bei Dengiz und Alabap [50]. In [159] wird gezeigt, dass der in [52, 15] verwendete Straftermansatz zur Berücksichtigung der All-Terminal-Zuverlässigkeitsnebenbedingung im GA-Design unzulässige Lösungen generiert. Für die Planung mit Technologieoptionen schlagen Deeter und Smith in [49, 48] einen GA für die Planung von kostenminimalen Netzwerktopologien unter Beachtung einer All-Terminal-Zuverlässigkeitsneben-

[19] Hier gemessen an den Zielen: durchschnittliche Distanz zwischen zwei Kommunikationsendpunkten, der Verzögerung bei der Übertragung und dem Grad der Knoten.

bedingung bei unterschiedlichen Verbindungszuverlässigkeiten vor. In [158] (vgl. hierzu auch Abschnitt 5.2.3 auf S. 109) konnte gezeigt werden, dass ein evolutionärer Planungsansatz mit einer Reparaturfunktion den in [49, 48] vorgeschlagenen straftermbasierten GA in der Qualität der Lösungen überlegen ist.
Der Literaturüberblick zeigt, dass bereits eine Reihe von Verfahren für die Planung von Kommunikationsnetzwerken existieren. Bei den in [120, 165] vorgestellten Verfahren wird das im Rahmen der Arbeit betrachtete Planungsziel der Zuverlässigkeit nicht betrachtet. Eine Betrachtung der Ausfallsicherheit/Zuverlässigkeit findet in [7, 31, 34, 69, 86, 121, 130, 177] statt. In diesen Arbeiten wird jedoch stets nur ein einfaches Zuverlässigkeitsmaß wie z.B. der n-fach-Zusammenhang (vgl. hierzu Abschnitt 4.3) verwendet. Die Arbeiten [3, 8, 15, 49, 48, 52, 50, 96, 110, 120] verwenden die All-Terminal-Zuverlässigkeit zur Bestimmung der Ausfallsicherheit der Kommunikationsinfrastruktur. Voraussetzung für den Einsatz der in [2, 15, 52, 96] vorgeschlagenen Methoden ist, dass diese nur bei identischen Ausfallwahrscheinlichkeiten der Verbindungen einsetzbar sind. Lediglich die in [49, 48, 110, 158] vorgestellten Verfahren arbeiten mit unterschiedlichen Verbindungszuverlässigkeiten. Bisher wurden mit den dort vorgeschlagenen Verfahren jedoch lediglich Probleme mit bis zu maximal 19 Knoten und 171 Verbindungen untersucht. Die Ergebnisse in [14, 112] zeigen, dass durch Verwendung von multikriteriellen EAs bessere als die in [49] gezeigten Lösungen gefunden werden. Diese Ergebnisse machen deutlich, dass in bisher veröffentlichten Verfahren bezüglich der Qualität der Lösungen Defizite bestehen, da diese Verfahren noch keine optimale Lösung[20] finden.

3.6.2 Bisherige Ansätze zur Planung von Kommunikationsnetzwerken mit mehreren Zielgrößen

Wie in Abschnitt 3.6.1 dargestellt wurde, existieren für die Planung von Kommunikationsnetzwerken unter Zuverlässigkeits- und Kostengesichtspunkten eine Vielzahl von Verfahren, welche bereits viele Jahre zurückreichen. Im Gegensatz dazu wird die Problemstellung als multikriterielles Optimierungsproblem erst seit wenigen Jahren untersucht.
Kumar u.a. stellen in [123, 122] ein multikriterielles evolutionäres Verfahren vor, bei dem die Ziele Kosten und durchschnittliche Verzögerung simultan minimiert werden. Die Zuverlässigkeit der dabei entworfenen Netzwerktopologien wird jedoch nicht betrachtet. In [63, 14] stellen Flores und Cegla multikriterielle GAs für die Planung von zuverlässigen und ökonomischen Netzwerktopologien vor. Die vorgestellten Verfahren werden dabei jeweils nur an einem einzigen aus [49] entnommenen Netzwerk untersucht. In [157] konnte in einer experimentellen Studie gezeigt werden, dass mit dem Einsatz von multikriteriellen Simulated Annealing-Verfahren bessere als die von Flores und Cegla [63, 14] veröffentlichten Ergebnisse erreichbar sind. Konak und Smith [112] stellten ebenfalls einen multikriteriellen GA für die Planung von zuverlässigen und ökonomischen Netzwerktopologien vor. Die Autoren untersuchen dabei eine Reihe von in [49, 52] vorgestellten Testproblemen. Nachteilig an dem Ansatz von Konak und Smith ist, dass der dabei vorgeschlagene multikriterielle GA keine der in Abschnitt 3.5.5 auf-

[20] Hier zu verstehen als globales Optimum bzw. als eine Lösung, die näher an diesem liegt als bisher veröffentlichte Ergebnisse.

gestellten Anforderungen an einen multikriteriellen EA berücksichtigt. In [136, 137] werden Verfahren zur multikriteriellen Planung zuverlässiger Netzwerke vorgeschlagen. Als Ziele verwenden Marsequerra u. a. ([136, 137]) die Maximierung der All-Terminal-Zuverlässigkeit und die Minimierung der Varianz für die Schätzung der All-Terminal-Zuverlässigkeit. Die Kosten der entworfenen Netzwerktopologie werden in [136, 137] nicht betrachtet.

3.7 Zusammenfassung

Dieses Kapitel hat eine Einführung in die Grundlagen der kombinatorischen Optimierung gegeben und das im Rahmen der Arbeit untersuchte Problem als kombinatorisches Optimierungsproblem dargestellt. Als Lösungsverfahren wurde die Verfahrensklasse der Metaheuristiken vorgestellt. Es wurden sowohl die Grundlagen für die Lösung von kombinatorischen Optimierungsproblemen mit einer Zielgröße als auch die Grundlagen für kombinatorische Optimierungsprobleme mit mehreren Zielgrößen eingeführt. Als Lösungsverfahren für das im Rahmen der Arbeit untersuchte Problem wurden die evolutionären Algorithmen ausgewählt. Aufbauend auf einer Übersicht zu den unterschiedlichen Verfahren, die unter dem Dach der evolutionären Algorithmen zusammengefasst sind, wurden die genetischen Algorithmen als geeignete Metaheuristik ausgewählt.
Basierend auf den Empfehlungen aus der Literatur wurden Anforderungen für Verfahren zur Lösung von multikriteriellen Entscheidungsproblemen aufgestellt. Als ein Vertreter dieser Verfahren, der all diese Anforderungen erfüllt, wurde das NSGA-II-Verfahren im Detail identifiziert und erläutert.
Ein abschließender Literaturüberblick hat gezeigt, dass bereits eine Reihe von Verfahren für die monokriterielle Planung von Kommunikationsnetzwerken unter Kosten- und Zuverlässigkeitsaspekten existieren. In der Vergangenheit wurden jedoch zumeist nur kleine Probleme sowie Probleme mit unzureichenden Zuverlässigkeitsmaßen untersucht oder vereinfachte Modellannahmen vorgenommen. Zusätzlich motiviert wird die Arbeit für die monokriterielle Planung durch Ergebnisse, welche mittels multikriterieller Ansätze erzielt werden konnten. Mittels dieser Methoden wurde in der jüngeren Vergangenheit gezeigt, dass für ausgewählte Probleme bessere als die bisher (mittels monokriteriellen Planngsverfahren) gefundenen Ergebnisse erreichbar sind.
Für existierende multikriterielle Ansätze konnte in einer experimentellen Studie mit einem multikriteriellen Simulated Annealing-Ansatz eine deutlich höhere Lösungsqualität als die bisher veröffentlichten Ergebnisse erreicht werden. Somit wird für diese Arbeit das Ziel abgeleitet, die im Rahmen der Arbeit untersuchte Problemstellung ebenfalls als multikriterielles Optimierungsproblem zu untersuchen.

4 Einfluss von Ausfällen und Störungen auf die Kommunikationsnetzwerkzuverlässigkeit

Für die Bestimmung der Zuverlässigkeit eines Kommunikationsnetzwerkes können verschiedene Bewertungsmaße verwendet werden. Dieses Kapitel gibt einen Überblick über Ansätze zur Bewertung der Zuverlässigkeit eines Kommunikationsnetzwerkes. Wichtig für die Zuverlässigkeit von Kommunikationsnetzen ist die Fähigkeit des Systems, trotz des Ausfalls einzelner Verbindungen die Kommunikationsfähigkeit des gesamten Systems sicherzustellen. Ein Netzwerk ist zuverlässig, wenn nach Ausfällen von einzelnen Knoten oder Verbindungen immer noch Daten zwischen allen noch vorhandenen Knoten übertragen werden können.
Das Kapitel ist wie folgt gegliedert: Zunächst erfolgt eine Betrachtung von Qualitätsmaßen für Kommunikationsnetzwerke sowie eine Klassifizierung unzuverlässiger Netzwerkkomponenten. Anschließend werden die graphentheoretischen Grundlagen für die Modellierung eines Kommunikationsnetzwerkes und die Bestimmung von Zuverlässigkeitsmaßen eingeführt. Mit Abschnitt 4.3 werden unterschiedliche Maße für die Bestimmung der Zuverlässigkeit von Kommunikationsnetzwerken vorgestellt und ein geeignetes Maß für die Analyse von modernen Kommunikationsinfrastrukturen ausgewählt. Abschnitt 4.3.3 stellt exakte sowie approximative Verfahren zur Bestimmung der All-Terminal-Zuverlässigkeit vor. Auf Grundlage einer empirischen Untersuchung von ausgewählten Verfahren zur Bestimmung der All-Terminal-Zuverlässigkeit werden Empfehlungen für den Einsatz verschiedener Methoden gegeben.

4.1 Grundlagen zur Analyse der Zuverlässigkeit von Kommunikationsnetzwerkkomponenten

4.1.1 Qualitätsmaße für Kommunikationsnetzwerke

Die Informations- und Kommunikationstechnik stellt heutzutage das Rückgrat einer Vielzahl von Geschäftsprozessen in der Industrie, im Handel und der Verwaltung dar. Ohne leistungsfähige und hoch verfügbare Kommunikationsnetzwerke ist eine unternehmensübergreifende Vernetzung von Informationssystemen entlang der Wertschöpfungskette, wie sie beispielsweise für die Umsetzung moderner Supply-Chain-Konzepte notwendig sind, kaum denkbar. Zwei Qualitätsmaße für Hard- und Softwarekomponenten sind beim Entwurf von Netzwerken von besonderer Bedeutung: Verfügbarkeit und Zuverlässigkeit [178, S. 533]. Die Verfügbarkeit drückt das Verhältnis zwischen der Zeit, in der das betrachtete System fehlerfrei arbeitet, zur betrachteten Gesamtzeit aus. Gemäß ITU-T X.137 ist die Verfügbarkeit einer Kommunikationskomponente definiert als:

$$\text{Verfügbarkeit} = \frac{MTBF}{MTBF - MTTR} \tag{4.1}$$

Dabei entspricht MTBF (Mean Time Between Failures) dem mittleren Ausfallabstand und MTTR (Mean Time To Repair) der mittleren Reparaturdauer.
Die Zuverlässigkeit gibt die Wahrscheinlichkeit dafür an, dass das betrachtete System innerhalb eines Zeitraums $\triangle\, t$ nicht ausfällt [178, S. 535]. Sie ist definiert als:

$$\text{Zuverlässigkeit} = e^{-\triangle t/MTBF} \tag{4.2}$$

Betrachtet man beispielsweise eine System, dessen MTBF 20.000 Stunden beträgt für ein Zeitintervall $\triangle\, t$ von einem Jahr (8760 Stunden), so ergibt sich nach Gleichung 4.2 eine Zuverlässigkeit von $e^{-8760/20000} = 64,53\,\%$. Die Unzuverlässigkeit (Ausfallwahrscheinlichkeit) des Systems beträgt somit $100\,\% - 64,53\,\% = 35,47\,\%$. Beide Qualitätsmaße ermöglichen es, Anforderungen an die Stabilität und die Qualität eines Kommunikationsnetzwerkes zu definieren. Bereits die Betrachtung einer der beiden Teilaspekte in Topologieplanungsverfahren erfordert spezialisierte Methoden und Algorithmen. Im Mittelpunkt dieses Abschnittes steht die Analyse der Zuverlässigkeit von Kommunikationsnetzwerken.

4.1.2 Unzuverlässige Netzwerkkomponenten

Betrachtet man die verschiedenen Komponenten, die in einem Netzwerk für die Kommunikation benötigt werden, so lassen sich zwei Klassen einteilen:

- Kommunikationsverbindungen und
- Kommunikationsknoten.

Die Klasse der Kommunikationsverbindungen fasst sämtliche für die Vernetzung unterschiedlicher Standorte, Konzentratoren und Netzübergangspunkte notwendigen Verbindungstypen zusammen. In der vorliegenden Arbeit soll dabei keine weitere Klassifizierung nach Netzwerktopologien (Bus, Ring, vermascht), verwendeten Übertragungstechniken (ATM, Ethernet, FDDI, etc.) sowie den verwendeten Übertragungsmedien (Glasfaser, Kupfer, etc.) erfolgen.
In der Klasse der Kommunikationsknoten werden sämtliche Punkte subsumiert, die Endpunkt einer Kommunikationsverbindung sind. Darunter sind beispielsweise die Übergangspunkte zwischen dem Unternehmensnetzwerk und dem Netzwerk des Internet-Providers, Übergabepunkte zwischen verschiedenen Providern sowie Konzentratoren im internen Netz des Providers zu verstehen. Sämtliche dieser Kommunikationsknoten werden durch Hardwaresysteme realisiert, die aufgrund von software- sowie systemtechnischer Störungen ausfallen können.
Für diese Arbeit wird in Anlehnung an andere Arbeiten auf dem Gebiet (vgl. hierzu Abschnitt 3.1.1 auf S. 27 für die getroffenen Modellannahmen) eine Abstraktion in der Art vorgenommen, dass die Annahme getroffen wird, dass lediglich die Kommunikationsverbindungen in einem Kommunikationsnetzwerk unzuverlässig arbeiten und eine Verbindung zwischen zwei Knoten aufgrund von Verschleiß, externen Störungen (z. B. Zerstörung einer Leitung durch Bauarbeiten) oder durch Wartungsarbeiten nicht immer verfügbar ist. Da die Untersuchung zu Verfügbarkeiten und Ausfallwahrscheinlichkeiten,

-verteilungen sowie präventiven Maßnahmen zur Erhöhung der Zuverlässigkeit der Kommunikationsverbindungen nicht den Schwerpunkt der vorliegenden Arbeit bilden, sei an dieser Stelle auf weiterführende Literatur [131, 178] verwiesen.

4.2 Graphentheoretische Grundlagen für die Zuverlässigkeitsuntersuchung von Kommunikationsnetzwerken

Zunächst werden einige grundlegende Begriffe der Graphentheorie eingeführt. In Anlehnung an [22, 53, 185] werden in dieser Arbeit die nachfolgenden Definitionen verwendet. Anhand von Abbildung 4.1 werden die Definitionen am Beispiel veranschaulicht.

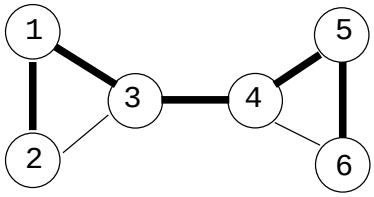


Abbildung 4.1: Beispielgraph $G_{Beispiel}$ zur Erläuterung der graphentheoretischen Grundbegriffe

Definition 4.1 (Graph)
Ein Graph $G = (V, E)$ ist ein Tupel der Mengen V (engl. vertices, Knoten) und E (engl. edges, Kanten). $|V|$ bezeichnet die Mächtigkeit der Menge der Knoten in G. $|E|$ gibt die Mächtigkeit der Menge der Kanten in G an.

Eine Kante zwischen zwei Knoten v_i, v_j wird im Folgenden durch e_{v_i,v_j} notiert. Die Notation e_i bezeichnet eine beliebige Kante des Graphen. Die Notation v_i bezeichnet einen beliebigen Knoten des Graphen.

Definition 4.2 (Ungerichteter Graph)
Für einen ungerichteten Graphen gilt für die Kantenmenge E, dass jede Kante ein ungeordnetes Knotenpaar ($\{v_1, v_2\}$ mit $v_1, v_2 \in V$) verbindet. Dies bedeutet, dass die Kante e_{v_i,v_j} identisch mit der Kante e_{v_j,v_i} ist.

Für den in Abbildung 4.1 dargestellten Graph $G_{Beispiel}$ gilt:
Er ist ungerichtet. Der Graph besteht aus der Knotenmenge $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ sowie der Kantenmenge $E = \{e_{v_1,v_2}, e_{v_1,v_3}, e_{v_2,v_3}, e_{v_3,v_4}, e_{v_4,v_5}, e_{v_4,v_6}, e_{v_5,v_6}\}$.

Definition 4.3 (Subgraph)
Jeder Graph $G_{sub}(V_{sub}, E_{sub})$ für den gilt $\forall v_i \in V_{sub} : v_i \in V$ und $\forall e_i \in E_{sub} : e_i \in E$, wird als Subgraph von G bezeichnet.

Ein möglicher Subgraph von $G_{Beispiel}$ wäre ein Graph G_{sub} mit den Knoten $V_{sub} = \{v_1, v_2, v_3\}$ und den Kanten $E = \{e_{v_1,v_2}, e_{v_1,v_3}, e_{v_2,v_3}\}$.

Definition 4.4 (Inzident)
Ein Knoten $v \in V$ heißt mit einer Kante $e_{v_i,v_j} \in E$ inzident, wenn gilt $v \in \{v_i, v_j\}$. Die beiden inzidenten Knoten einer Kante e_{v_i,v_j} mit $v_i, v_j \in V$ sind die Terminal-Knoten einer Kante, die durch e_{v_i,v_j} verbunden werden.

Für die Kante e_{v_3,v_4} in $G_{Beispiel}$ sind die Terminal-Knoten die inzidenten Knoten v_3 und v_4.

Definition 4.5 (Grad eines Knotens)
Die Menge der mit einem Knoten v_i inzidenten Kanten wird als Grad (engl. degree) des Knotens ($deg(v_i)$) bezeichnet.

In dem Graphen $G_{Beispiel}$ haben die Knoten v_3 und v_4 den höchsten Grad: ($deg(v_3) = deg(v_4) = 3$).

Definition 4.6 (Schlichter Graph)
Eine Kante e_{v_i,v_j} mit $v_i, v_j \in V$ heißt Schlinge, wenn gilt $i = j$. Zwei Kanten $e_1, e_2 \in E$ heißen parallele Kanten, wenn ihre Terminal-Knoten identisch sind. Ein Graph heißt schlicht, wenn er keine Schlingen und parallelen Kanten enthält.

Gemäß der Definition 4.6 handelt es sich bei $G_{Beispiel}$ um einen schlichten Graphen.

Definition 4.7 (Benachbarte Knoten)
Zwei Knoten $v_i, v_j \in V$ sind benachbarte (adjazent), wenn es eine Kante $e_{v_i,v_j} \in E$ gibt.

Definition 4.8 (Vollständiger Graph)
Ein Graph heißt vollständiger Graph, wenn je zwei Knoten in G benachbart sind. Anderenfalls wird der Graph als nicht vollständig bezeichnet.
Für einen vollständigen ungerichteten Graphen gilt: $|E| = \frac{|V| \cdot |V-1|}{2}$.

Definition 4.9 (Kantenzug)
Ein Kantenzug ist eine Folge von Kanten $e_1, e_2, \ldots, e_i$ für die gilt, dass es eine Folge von Knoten $v_0, v_1, \ldots v_i$ gibt, so dass für jedes j (mit $1 \leq j \leq i$) die Kante e_j mit e_{v_{j-1},v_j} in dem Kantenzug vorhanden ist.

Definition 4.10 (Pfad)
Ein Kantenzug wird als Pfad bezeichnet, wenn alle Kanten verschieden sind. Ein Pfad heißt geschlossen, wenn gilt $v_0 = v_i$. Pfade zwischen zwei Knoten v_i und v_j sind zueinander kantendisjunkt, wenn sie keine gemeinsamen Kanten besitzen.

Zu einem geschlossenen Pfad in $G_{Beispiel}$ gehören z. B. die Kanten $\{e_{v_1,v_2}, e_{v_2,v_3}, e_{v_1,v_3}\}$. Ein nicht geschlossener Pfad, der die Knoten v_2 und v_6 verbindet ist in Abbildung 4.1 fett dargestellt.

Definition 4.11 (Zusammenhang)
Ein Graph ist zusammenhängend, wenn für jedes Knotenpaar $\{v_i, v_j\} \in V$ ein Pfad existiert.

Der in Abbildung 4.1 gezeigte Graph ist zusammenhängend, da für jedes Knotenpaar in $G_{Beispiel}$ ein Pfad existiert.

Definition 4.12 (Zusammenhangskomponenten)
Die Knoten V_{Sub} mit den inzidenten Kanten E_{Sub} eines zusammenhängenden Subgraphen von G werden als Zusammenhangskomponente bezeichnet, wenn gilt: $\forall\{v_i, v_j\} \in V_{Sub} \exists$ Pfad $\{e_i, \ldots, e_j\} : e \in E_{Sub}$.

Definition 4.13 (Wald)
Ein ungerichteter Graph, der einen geschlossenen Pfad enthält, wird als Wald bezeichnet. Die Zusammenhangskomponenten des Waldes heißen Bäume.

Definition 4.14 (Spannbaum)
Ein zusammenhängender schlichter Subgraph G_{Sub} von G mit $V_{Sub} = V$, $E_{Sub} \subseteq E$ und $|E| = |V| - 1$ wird als Spannbaum bezeichnet.

Für den Graphen $G_{Beispiel}$ gehört die fett dargestellte Kantenmenge zu einem Spannbaum in $G_{Beispiel}$.
Für viele praktische Anwendungen ist es notwendig, den Kanten und Knoten eines Graphen mit Hilfe einer Gewichtsfunktion Werte zuzuweisen. Auf diese Weise lässt sich beispielsweise die Länge einer Verbindung zwischen zwei Punkten abbilden oder eine maximale Kapazität eines Knotens festlegen. Eine Funktion $g : V, E \rightarrow \mathbb{R}$ wird als Gewichtsfunktion bezeichnet. Sie ordnet dabei jedem Knoten und jeder Kante des Graphen ein Gewicht $g(v)$ bzw. $g(e)$ mit $v \in V, e \in E$ zu.

Definition 4.15 (Stochastischer Graph)
Ein stochastischer Graph ist ein ungerichteter, schlichter, zusammenhängender Graph $G = (V, E)$, dessen Kanten und Knoten unabhängig voneinander mit gegebenen Wahrscheinlichkeiten (p_{fail}) ausfallen. Eine Gewichtsfunktion r ordnet jedem Element des stochastischen Graphen G eine Zuverlässigkeit ($1 - p_{fail}$) zu.

Modellierung des Netzwerkdesignproblems mit Hilfe der Graphentheorie

Mit Hilfe der hier vorgestellten graphentheoretischen Grundlagen ist die Modellierung des im Rahmen der Arbeit untersuchten kombinatorischen Optimierungsproblems (siehe Abschnitt 3.1.1) möglich.
Ein Kommunikationssystem wird im Folgenden als stochastischer Graph $G = (V, E)$ modelliert. Die Knoten des Kommunikationssystems (Standorte) bilden die Knotenmenge V des Graphen. Die Verbindungen zwischen den einzelnen Kommunikationsknoten werden in der Kantenmenge E zusammengefasst. Über die Gewichtsfunktion r wird jeder Kante $e_{v_i,v_j} \in E$ ein Gewicht $r(e_{v_i,v_j})$ zugeordnet, welches die Zuverlässigkeit der jeweiligen Verbindung angibt. Auf diese Weise werden in der Praxis auftretende Wartungs- und Instandhaltungszeiten sowie nicht planbare externe Störungen abgebildet. Jede Kante e_i kann sich im Zustand $state_{e_i}$ „operabel“ ($state_{e_i} = 1$) oder „ausgefallen“ ($state_{e_i} = 0$) befinden. Gemäß der Modellannahmen aus Abschnitt 4.1.2 arbeiten die Knoten des Kommunikationsnetzwerkes zuverlässig ($p_{fail} = 0$), so dass $r(v_i) = 1 : \forall v_i \in V$.
Ein Lösungsalgorithmus erhält als Eingabedaten eine Menge von Knoten $K_{Eingabe}$ und eine Menge von möglichen Verbindungen $L_{Eingabe}$ (Links) zwischen diesen Knoten. Aus diesen beiden Mengen wird der ungerichtete Eingabegraph G mit $V = K_{Eingabe}$ und $E = L_{Eingabe}$ erstellt. Wenn nachfolgend die Notation G verwendet wird, so entspricht

dies wenn nicht anders angegeben, dem Eingabegraphen. Eine Lösung des Netzwerkdesignproblems wird durch eine Netzwerktopologie N repräsentiert. Diese Topologie enthält die Menge der Knoten $V_N = K_{Eingabe}$, die mit der Eingabe zur Verfügung gestellt wurden und eine Teilmenge der Verbindungen $E_N \subseteq L_{Eingabe}$. Eine solche Netzwerktopologie N wird durch einen Subgraphen von G, der als G_N (mit $V_N = V$ und $E_N \subseteq E$) bezeichnet wird, modelliert. Die Kostenfunktion $C(G_N)$ gibt die Kosten[21] an, die mit der Installation der Netzwerktopologie N entstehen. Die All-Terminal-Zuverlässigkeit[22] für eine Netzwerktopologie N wird mittels $R_{All}(G_N)$ angegeben.
Für jede Kante $e_i \in E$ des Graphen G werden die Kosten $c(e_i)$, die durch die Verwendung der Kante in G_N entstehen, zugeordnet. Zusätzlich wird jeder Kante in G eine Zuverlässigkeit $r(e_i)$ zugeordnet.
Wenn als Eingabe für eine Verbindung mehrere Optionen zur Verfügung stehen, so werden diese als Technologieoptionen (vgl. S. 28) l_k ($L_{Options} = l_1, l_2, \ldots, l_{k_{max}}$) modelliert. Für eine Verbindung mit mehreren Technologieoptionen entspricht $c(l_k(e_{v_i,v_j}))$ den Kosten der Kante e_{v_i,v_j} für die Technologieoption l_k und $r(l_k(e_{v_i,v_j}))$ der Zuverlässigkeit der Kante für Technologieoption l_k. Für die Technologieoptionen gilt: $c(l_1(e_{v_i,v_j})) < c(l_2(e_{v_i,v_j})) < \ldots < c(l_{k_{max}}(e_{v_i,v_j}))$ und $r(l_1(e_{v_i,v_j})) < r(l_2(e_{v_i,v_j})) < \ldots < r(l_{k_{max}}(e_{v_i,v_j}))$.

4.3 Zuverlässigkeitsmaße für Kommunikationsnetzwerke

4.3.1 Komponentenzusammenhang in Kommunikationsnetzwerken

Mit Hilfe des Zusammenhangs (vgl. Definition 4.11) eines Kommunikationsnetzes können bereits einfache Aussagen über seine Zuverlässigkeit getroffen werden. Durch den Ausfall einer Verbindung in einem stochastischen Graph steht die Verbindung zwischen zwei Knoten nicht mehr zur Verfügung. Jeder einfach-zusammenhängende Graph kann durch den Ausfall von nur einer Kante in zwei Zusammenhangskomponenten zerfallen. Der in Abbildung 4.1 auf S. 67 gezeigte Graph repräsentiert ein Kommunikationsnetzwerk, welches als Beispiel für die Zusammenhangsmaße verwendet wird. Der Graph ist einfach-zusammenhängend. Bereits durch den Ausfall von nur einer Kante (Kante zwischen den Knoten v_3 und v_4) zerfällt der Graph in zwei Subgraphen, die aus den Zusammenhangskomponenten mit den Knoten $\{v_1, v_2, v_3\}$ und $\{v_4, v_5, v_6\}$ sowie den inzidenten Kanten bestehen. Bei Verwendung einer zusätzlichen Kante zwischen beiden Zusammenhangskomponenten (z. B. eine Kante von Knoten v_1 nach Knoten v_6) wäre der Graph zweifach-zusammenhängend und dessen Zuverlässigkeit erhöht. In einem zweifach-zusammenhängenden Graphen müssen mindestens zwei Kanten gelöscht werden, damit nicht mehr alle Knoten miteinander verbunden sind und das damit modellierte Kommunikationsnetz unzuverlässig wird. In der Literatur wird ein Netzwerk, welches durch einen zweifach-zusammenhängenden Graph repräsentiert wird, häufig auch als 1-FT-Netzwerk (one fault-tolerant network) bezeichnet [69]. Im allgemeinen Fall enthält ein n-fach-zusammenhängender Graph mindestens n kantendisjunkte Pfade für jedes Knotenpaar. Solange weniger als n Kanten aus dem Graph gelöscht werden, ist ein

[21] Die Kosten der Verbindungen der im Rahmen dieser Arbeit untersuchten Planungsprobleme werden in Geldeinheiten gemessen.
[22] Eine detaillierte Beschreibung für dieses Zuverlässigkeitsmaß bietet Abschnitt 4.3.3.

n-fach-zusammenhängender Graph verbunden und alle Knoten zusammenhängend. Der Begriff des Zusammenhangs wird in der Literatur mitunter falsch verwendet. Dengiz u.a. [52] schlagen z. B. eine Reparaturheuristik vor, die einen 2-fach-Zusammenhang für einen Graphen herstellt. Für einen 2-fach-Zusammenhangscheck prüft die Heuristik, ob der Grad jedes Knotens $v_i \in E : deg(v_i) \geq 2$ ist. Diese einfache Form der Überprüfung kann jedoch nur eingesetzt werden, wenn es sich bei dem getesteten Graphen um einen Ring handelt. Mit dem in Abbildung 4.1 dargestellten Graphen wird deutlich, dass die Aussage für andere Graphenstrukturen nicht gilt. Obwohl der Grad sämtlicher Knoten größer eins ist, zerfällt der Graph beim Ausfall der Verbindung (e_{v_3,v_4}) in zwei nicht mehr verbundene Zusammenhangskomponenten.
Die Betrachtung von Komponentenzusammenhängen zur Bewertung der Zuverlässigkeit ist unter praktischen Gesichtspunkten nicht ausreichend, da dadurch nur prinzipielle Aussagen über die Überlebensfähigkeit von Kommunikationsnetzen getroffen werden können. Aussagen über Wahrscheinlichkeiten, mit der ein Netzwerk beim Ausfall einzelner Verbindungen nicht mehr zusammenhängend ist, sind durch dieses Maß nicht möglich.

4.3.2 Source-Terminal- und k-Terminal-Zuverlässigkeit

Die Unzulänglichkeiten von Komponentenzusammenhängen zur Bewertung der Zuverlässigkeit von Netzwerken können durch die Source-Terminal (s-t)-Zuverlässigkeit R_2 behoben werden.

Definition 4.16 (s-t-Zuverlässigkeit)
Die s-t-Zuverlässigkeit zwischen zwei Knoten (Source und Terminal) ist definiert als die Wahrscheinlichkeit, dass in einem stochastischen Graph mindestens ein Pfad existiert, über den die beiden Knoten miteinander kommunizieren können [36, S. 3].

Durch die s-t-Zuverlässigkeit sind Aussagen über die Wahrscheinlichkeit der Kommunikation zwischen zwei Knoten bei Ausfällen einzelner Verbindungen möglich. Die ausschließliche Betrachtung der Kommunikationsverbindungen zwischen zwei Knoten reicht in der Realität allerdings zur Beurteilung der Zuverlässigkeit des gesamten Netzwerkes nicht aus. Daher wurde das Konzept der s-t-Zuverlässigkeit auf k Knoten erweitert und die Zuverlässigkeit für alle Verbindungen zwischen k Knoten im Netzwerk durch die k-Terminal-Zuverlässigkeit R_k beschrieben [173, S. 286].

Definition 4.17 (k-Terminal-Zuverlässigkeit)
Die k-Terminal-Zuverlässigkeit R_k gibt die Wahrscheinlichkeit dafür an, dass zwischen beliebigen k Knoten in einem stochastischen Graph ein Pfad existiert.

Verfahren zur Berechnung der k-Terminal-Zuverlässigkeit für stochastische Graphen findet man in [129, 90].

4.3.3 Das All-Terminal-Zuverlässigkeitsmaß

Für die Zuverlässigkeitsbeurteilung moderner Kommunikationsinfrastrukturen ist die Betrachtung von jeweils k Knoten mit Hilfe der k-Terminal-Zuverlässigkeit allerdings

noch immer nicht ausreichend, sondern es müssen die Verbindungen zwischen sämtlichen Knoten des Netzwerkes betrachtet werden. So muss z. B. in Backbone-Netzwerken von Unternehmen sichergestellt werden, dass alle Standorte des Unternehmens mit allen anderen Standorten kommunizieren können. Die Bewertung der Zuverlässigkeit der Kommunikation zwischen allen vorhandenen Knoten eines Netzwerkes wird durch die All-Terminal-Zuverlässigkeit R_{All} möglich. Diese stellt eine Erweiterung der k-Terminal-Zuverlässigkeit dar.

Definition 4.18 (All-Terminal-Zuverlässigkeit)
Die All-Terminal-Zuverlässigkeit ist definiert als die Wahrscheinlichkeit, dass zwischen jedem Knotenpaar eines stochastischen Graphen ein Pfad existiert [32]. R_{All} kann auch als die Wahrscheinlichkeit interpretiert werden, dass in dem Graph mindestens ein funktionierender Spannbaum existiert, welcher sämtliche Knoten miteinander verbindet [36, S. 3].

Die All-Terminal-Zuverlässigkeit stellt für die Planung und Erweiterung von Kommunikationsnetzwerken ein geeignetes Gütemaß zur Beurteilung der Robustheit der entworfenen Topologien beim Ausfall einzelner Verbindungen des Netzwerkes dar [52, 49, 63, 112, 159].

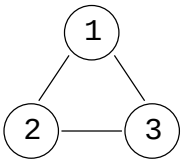
4.4 Verfahren zur Berechnung der All-Terminal-Zuverlässigkeit

Die exakte Berechnung von R_{All} zählt zu den NP-schweren Problemen [193]. Daher sind exakte Ansätze zur Berechnung der All-Terminal-Zuverlässigkeit (z. B. das Verfahren von Buzacott [143, 65]) in der Regel aufgrund des hohen Ressourcenbedarfs und langen Laufzeiten lediglich für Netzwerktopologien mit sehr wenigen Knoten effizient einsetzbar. In den vergangenen Jahrzehnten wurden eine Reihe unterschiedlicher Methoden zur Berechnung der All-Terminal-Zuverlässigkeit entwickelt. In [36] wird ein ausführlicher Überblick über verschiedene exakte sowie approximative Verfahren gegeben. Eine Analyse zur Komplexität der Zuverlässigkeitsberechnung unter Verwendung identischer Zuverlässigkeiten für sämtliche Verbindungen im Netzwerk bietet [12]. Exakte Verfahren zur Berechnung von R_{All} werden in [32, 143, 65, 4, 93, 30] vorgestellt. Die Verwendung unterschiedlicher Monte-Carlo-Simulationstechniken zur Bestimmung von R_{All} findet man in [56, 101, 134, 90]. Der anschließende Abschnitt gibt einen Überblick über ausgewählte exakte und approximative Verfahren zur Berechnung von R_{All}.

4.4.1 Exakte Berechnungsverfahren

4.4.1.1 Zuverlässigkeitsberechnung mittels vollständiger Zustandsenumeration

Eine der einfachsten Methoden zur Berechnung der All-Terminal-Zuverlässigkeit bei vorgegebenen Zuverlässigkeiten für die einzelnen Verbindungen besteht in der vollständigen Enumeration sämtlicher Zustände S des Netzwerkes. Jede Kante $e_i \in E$ bekommt dabei entweder den Zustand $state_{e_i} = 1$ für „operabel“ oder $state_{e_i} = 0$ für „ausgefallen“ zugeordnet. Ein Zustand $s_i \in S$ repräsentiert als Vektor eine konkrete Realisierung



$r(e_{1,2}) = 0{,}9$
$r(e_{1,3}) = 0{,}99$
$r(e_{2,3}) = 0{,}95$

Zustand $s_i(e_{v1,v2}, e_{v1,v3}, e_{v2,v3})$	$P(s_i)$	$\Phi(s_i)$
000	(1-0,9)·(1-0,99)·(1-0,95)	0
001	(1-0,9)·(1-0,99)·0,95	0
010	(1-0,9)·0,99·(1-0,95)	0
011	(1-0,9)·0,99·0,95	1
100	0,9·(1-0,99)·(1-0,95)	0
101	0,9·(1-0,99)·0,95	1
110	0,9·0,99·(1-0,95)	1
111	0,9·0,99·0,95	1

Abbildung 4.2: Beispiel Berechnung R_{All} mittels vollständiger Zustandsenumeration

sämtlicher Kantenzustände des Netzwerkes. $\Phi(s_i)$ sei eine Funktion, die den Zusammenhang des Graphen für einen Zustand s_i beschreibt. $\Phi(s_i)$ ist 1, wenn der Graph für den Zustand s_i zusammenhängend ist und anderenfalls 0. Die Eintrittswahrscheinlichkeit eines Zustands s_i sei $P(s_i)$. Für die All-Terminal-Zuverlässigkeit ergibt sich somit:

$$R_{All} = \sum_{s_i \in S} P(s_i) \cdot \Phi(s_i) \tag{4.3}$$

Die Anzahl der zu untersuchenden Zustände berechnet sich als $2^{|E_N|}$. Aufgrund des exponentiellen Wachstums der Zustände bei zunehmender Kantenanzahl ($|E_N|$) ist das Verfahren der vollständigen Zustandsenumeration nur für kleine Graphen sinnvoll einsetzbar, da z. B. für einen Graph mit 25 Kanten schon $3,3 \cdot 10^7$ verschiedene Zustände zu untersuchen sind.
Abbildung 4.2 demonstriert die Berechnung von R_{All} mittels vollständiger Zustandsenumeration an einem vollständig verbundenen Graphen mit drei Knoten. Für jede Kante ist die Zuverlässigkeit $r(e_{v_i,v_j})$ angegeben. Für jeden einzelnen Zustand s_i des Graphen, der durch den Ausfall einer Kante entsteht, zeigt die Tabelle die Eintrittswahrscheinlichkeit $P(s_i)$ und den Wert der Funktion $\Phi(s_i)$. Für die Berechnung von R_{All} sind nur die Zustände von Interesse mit $\Phi(s_i) = 1$. Die All-Terminal-Zuverlässigkeit ergibt sich aus Formel 4.3 somit als $R_{All} = (1 - 0,9) \cdot 0,99 \cdot 0,95 \cdot 1 + 0,9 \cdot (1 - 0,99) \cdot 0,95 \cdot 1 + 0,9 \cdot 0,99 \cdot (1 - 0,95) \cdot 1 + 0,9 \cdot 0,99 \cdot 0,95 \cdot 1 = 0,9936$.

4.4.1.2 Ein Dekompositionsansatz zur Zuverlässigkeitsberechnung von Kommunikationsnetzwerken

Durch die Reduktion der zu untersuchenden Zustände S kann die Berechnung von R_{All} beschleunigt werden [32, 65]. Im Folgenden wird ein mehrstufiger Dekompositionsansatz aus [32] vorgestellt, welcher sich durch eine effiziente Berechnung von R_{All} sowie eine leichte Parallelisierbarkeit auszeichnet. Eine detaillierte Beschreibung des Verfahrens wird in [32] gegeben.

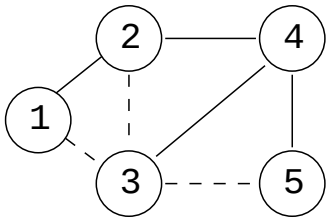


Abbildung 4.3: Beispielnetzwerk für den Dekompositionsansatz

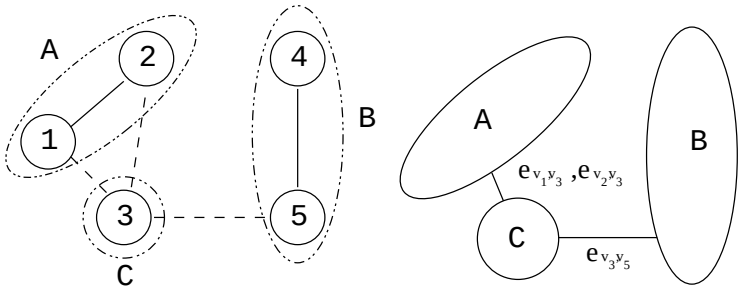
Bei diesem exakten Verfahren wird zu Beginn genau ein zufällig ausgewählter Spannbaum T mit den Kanten $e_1, e_2, \ldots, e_{n-1}$ betrachtet. Ausgehend von T werden die disjunkten Komplementärereignisse

$$\underline{T} = (\underline{e_1}, e_2, \ldots, e_{n-1}) \cup (e_1, \underline{e_2}, \ldots, e_{n-1}) \cup \ldots \cup (\underline{e_1}, \underline{e_2}, \ldots, e_{n-1}) \cup \ldots \cup (\underline{e_1}, \underline{e_2}, \ldots, \underline{e_{n-1}})$$

bestimmt. Jedes einzelne Ereignis $s_i \in \underline{T}$ repräsentiert dabei einen Zustand, in dem bis zu maximal n-1 Kanten ausgefallen sind (markiert durch $\underline{e_i}$). Die Berechnung von R_{All} erfolgt gemäß:

$$R_{All} = \prod_{e_i \in T} r(e_i) + \sum_{s_i \in \underline{T}} P(s_i) \cdot R_{All}(G_{s_i}) \tag{4.4}$$

Der erste Term berechnet die Wahrscheinlichkeit, dass keine Kante des Baums T ausfällt. $P(s_i)$ ist die Eintrittswahrscheinlichkeit für den Zustand $s_i \in \underline{T}$. Für die Berechnung von $R_{All}(G_{s_i})$ muss für jeden Zustand $s_i \in \underline{T}$ der daraus resultierende Subgraph G_{s_i} aus G gebildet werden. Hierfür werden im Graphen G zunächst sämtliche ausgefallenen Kanten des Baums T gelöscht. Die verbleibenden Zusammenhangskomponenten werden dann jeweils zu einem neuen Knoten zusammengefasst, so dass G_{s_i} entsteht. Anschließend werden die nicht in T verwendeten Kanten berücksichtigt. Hierbei werden alle parallelen Kanten zwischen zwei Knoten in G_{s_i} zu einer Kante zusammengefasst. Die Zuverlässigkeit der neuen Kanten ergibt sich aus den Einzelzuverlässigkeiten der jeweiligen parallelen Kanten. Für den in Abbildung 4.3 gezeigten Graphen G wurden die Kanten $\{e_{v_1,v_2}, e_{v_2,v_4}, e_{v_4,v_5}, e_{v_3,v_4}\}$ als Spannbaum ausgewählt. Die verbleibenden nicht verwendeten Kanten (hier gestrichelt dargestellt) werden bei der Erstellung des neuen Subgraphen G_{s_i} für die Zustände in T verwendet. Abbildung 4.4 illustriert die Vorgehensweise und zeigt für den Baum $T = \{e_{v_1,v_2}, e_{v_2,v_4}, e_{v_4,v_5}, e_{v_3,v_4}\}$ und das Ereignis $s = \{e_{v_1,v_2}, \underline{e_{v_2,v_4}}, e_{v_4,v_5}, \underline{e_{v_3,v_4}}\}$ die Erstellung des entsprechenden Subgraphen G_{s_i}. Durch den Ausfall der Kanten e_{v_2,v_4} und e_{v_3,v_4} im Baum entstehen die Zusammenhangskomponenten mit den Knoten $\{v_1, v_2\}$, $\{v_4, v_5\}$ und $\{v_3\}$ (vgl. Abbildung 4.4(a)). Die Knoten v_1 und v_2 sowie v_4 und v_5 werden jeweils zu einem neuen Knoten v_A bzw. v_B zusammengezogen. Anschließend werden die nicht für den Spannbaum verwendeten Kanten $\{e_{v_1,v_3}, e_{v_3,v_5}, e_{v_2,v_3}\}$ hinzugefügt. Die parallelen Kanten e_{v_1,v_3} und e_{v_2,v_3} zwischen den Knoten v_A und v_C werden hierbei zu einer Kante zusammengefasst und man erhält den in Abbildung 4.4(b) dargestellten Subgraphen G_{s_i}. Gemäß der Formel 4.4 wird für alle Zustände $s \in \underline{T}$ die Eintrittswahrscheinlichkeit $P(s_i)$ des Subgraphen G_{s_i} mit dessen All-Terminal-Zuverlässigkeit $R_{All}(G_{s_i})$ multipliziert und über alle Zustände aufsummiert.



(a) Finden der Zusammenhangskomponenten in G nach dem Ausfall der Kanten e_{v_2,v_4} und e_{v_3,v_4} in T

(b) Konstruktion von G_{s_i} mit Hilfe der Kanten $\{e_{v_1,v_3}, e_{v_2,v_3}, e_{v_3,v_5}\}$

Abbildung 4.4: Ablauf des Dekompositionsansatzes zur Erstellung eines neuen Graphen G_{s_i} beim Ausfall von zwei Kanten in T

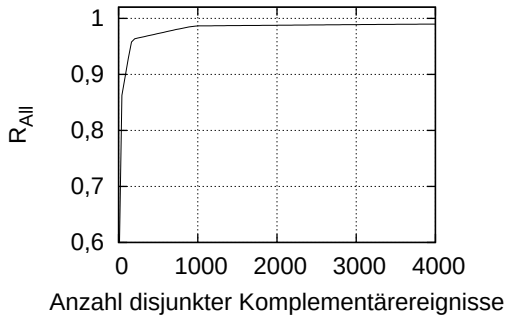


Abbildung 4.5: Entwicklung von R_{All} in Abhängigkeit von der Anzahl der betrachteten Komplementärereignisse $s \in \underline{T}$

Für die Berechnung von R_{All} wird diese Summe zur Zuverlässigkeit des Spannbaums T ($\prod_{e_i \in T} r(e_i)$) addiert.

Abbildung 4.5 zeigt die Entwicklung der All-Terminal-Zuverlässigkeit in Abhängigkeit von der Anzahl der betrachteten disjunkten Komplementärereignisse $s \in \underline{T}$ für einen Graphen mit 19 Knoten und 40 Kanten. Bereits nach der Bewertung der ersten Ereignisse ist ein großer Zuwachs für R_{All} zu erkennen. Ereignisse, welche zuletzt betrachtet werden, wie z. B. $(\underline{e_1}, \underline{e_2}, \ldots, \underline{e_{n-1}})$, haben dagegen nur einen sehr geringen Einfluss auf R_{All}. Aufgrund dieser Tatsache ist das Verfahren gut für Anwendungen geeignet, in denen R_{All} eine vorgegebene Zuverlässigkeitsschranke R_0 erfüllen muss. Sobald R_{All} größer als die geforderte Schranke R_0 ist, müssen die restlichen Ereignisse nicht mehr berechnet werden.

Ein weiterer Vorteil ist dabei die leichte Parallelisierbarkeit. Während zentral lediglich der Baum T zu finden ist, kann die Berechnung des zweiten Summanden aus Formel 4.4 verteilt erfolgen. Hierfür ist eine zentrale Koordinierungsinstanz notwendig, welche die einzelnen Ereignisse $s \in \underline{T}$ an die Clients verteilt und die Berechnung von R_{All} nach Formel 4.4 vornimmt.

4.4.2 Stochastische und approximative Berechnungsmethoden

Trotz möglicher Parallelisierung und der Einführung von Abbruchkriterien ist die exakte Berechnung von R_{All} mit Hilfe der in Abschnitt 4.4.1 vorgestellten Verfahren für große Netzwerke mit einem hohen Berechnungsaufwand und langen Laufzeiten verbunden. Da insbesondere bei der rechnergestützten, automatisierten Planung und Optimierung von Kommunikationsnetzwerken die All-Terminal-Zuverlässigkeit für eine große Menge von unterschiedlichen Netzwerktopologien bestimmt werden muss, sind langwierige Berechnungen hierfür nur begrenzt einsetzbar. Mit Hilfe von Näherungs- und Schätzverfahren lassen sich Abschätzungen für die tatsächliche All-Terminal-Zuverlässigkeit eines stochastischen Graphen ermitteln. Zwischen exakten Verfahren und approximativen Verfahren existiert dabei ein „Tradeoff“ bezüglich Genauigkeit und Laufzeit. Exakte Verfahren benötigen viel Rechenaufwand. Mithilfe approximativer Verfahren kann der Aufwand zur Berechnung von R_{All} auf Kosten einer geringeren Genauigkeit reduziert werden.

4.4.2.1 Approximationsverfahren zur Bestimmung von Zuverlässigkeitsschranken

Die Berechnung von Zuverlässigkeitsschranken ist für Anwendungen von Interesse, in denen Vorgaben für die maximale und minimale All-Terminal-Zuverlässigkeit existieren. Ausgehend von den Ergebnissen kann bei Bedarf eine nachfolgende exakte Berechnung von R_{All} durchgeführt werden. Ein repräsentatives Beispiel für Zuverlässigkeitsschranken stellt das Verfahren von [111] dar, welches die Berechnung einer oberen Schranke für R_{All} ermöglicht.
Nach [111] berechnet sich eine obere Schranke für R_{All} nach

$$R_{All} \leq 1 - \left[\sum_{i=1}^{|V|} \left(\prod_{e_{v_k,v_i} \in E_i} (1 - r(e_{v_k,v_i})) \prod_{j=1}^{i-1} \left(1 - \frac{\prod_{e_{v_k,v_j} \in E_j} (1 - r(e_{v_k,v_j}))}{1 - r(e_{v_i,v_j})} \right) \right) \right] \tag{4.5}$$

E_i ist die Menge der mit dem Knoten v_i inzidenten Kanten. Die Zuverlässigkeit einer Kante zwischen den Knoten v_i und v_j gibt $r(e_{v_i,v_j})$ an. Basierend auf Formel 4.5 lässt sich schnell bestimmen, ob ein Graph eine geforderte Zuverlässigkeit R_{All} erreichen kann.
Auf eine detaillierte Betrachtung zur Berechnung einer unteren Schranke für die All-Terminal-Zuverlässigkeit wird an dieser Stelle verzichtet, da diese für die in den Kapiteln 5–7 verwendeten Verfahren nicht benötigt wird. Eine ausführliche Betrachtung zu unteren Schranken findet der interessierte Leser in [36, S.83ff].

4.4.2.2 Einsatz der Monte-Carlo-Simulation für die Netzwerkzuverlässigkeitsberechnung

Neben der Abschätzung von R_{All} mit Hilfe von Schranken bedient man sich für die Bestimmung der Netzwerkzuverlässigkeit häufig der Monte-Carlo-Simulation [52, 62, 134, 63]. Das Grundprinzip sämtlicher Monte-Carlo-Techniken ist identisch. In mehreren unabhängig voneinander durchgeführten Stichproben generiert das Verfahren jeweils einen Zustand $s \in S$ des Graphen und untersucht dessen Zusammenhang. Aus einer Vielzahl von Stichproben wird anschließend eine Schätzung für die tatsächliche All-Terminal-Zuverlässigkeit vorgenommen. Das einfachste Monte-Carlo-Verfahren wird in Algorithmus 4.1 dargestellt.

Algorithmus 4.1 Einfache Monte-Carlo-Simulation nach [62, S. 61]

Eingabe: M, G
Ausgabe: $\hat{R}_{All}$

```
m = 0
R̂_All = 0
while m < M do
  for all e_{v_i,v_j} ∈ E do
    Generiere Zufallszahl z = [0, 1]
    if z ≤ (1 − r(e_{v_i,v_j})) then
      s_{e_{v_i,v_j}} = 0
    else
      s_{e_{v_i,v_j}} = 1
    end if
  end for
  if G ist zusammenhängend für s_i then
    Φ(s_i) = 1
  else
    Φ(s_i) = 0
  end if
  R̂_All += Φ(s_i)
  m = m + 1
end while
R̂_All = (1/M) · R̂_All
```

Einen genaueren Schätzwert für R_{All} erhält man mittels der in [134] vorgeschlagenen Strategie zur Generierung der Stichproben, welche einen „Fishman Sampling“-Plan verwendet. Die Anwendung dieser Strategie setzt allerdings einen vorgelagerten, möglichst effizienten Algorithmus zum Finden aller disjunkten Spannbäume sowie aller disjunkten minimalen Schnitte im Netzwerk voraus. In einem Graphen ist ein minimaler Schnitt die kleinste Menge an Kanten, durch deren gleichzeitigen Ausfall der Graph in mehrere Zusammenhangskomponenten zerfällt. Im Rahmen der Stichprobengenerierung werden drei disjunkte Mengen S_1, S_2, S_3 gebildet. Die Menge S_1 enthält die Ereignisse mit $\Phi(s_i) = 1$. Dabei gilt für alle $s_i \in S_1$, dass in mindestens einem der disjunkten Spannbäu-

me sämtliche Verbindungen im Status „operabel“ sind und damit das Netzwerk zusammenhängend ist. In der Menge S_2 werden alle die Zustände zusammengefasst, bei denen in jedem disjunkten, minimalen Schnitt mindestens eine Verbindung im Status „operabel“ ist und in jedem Spannbaum mindestens eine Verbindung den Status „ausgefallen“ besitzt. In der Menge S_3 sind schließlich die Ereignisse mit $\Phi(s_i) = 0$ enthalten. Damit gilt für alle $s_i \in S_3$, dass in mindestens einem der disjunkten Schnitte alle Kanten den Status „ausgefallen“ besitzen und damit das Netzwerk nicht mehr zusammenhängend ist. Anschließend kann die All-Terminal-Zuverlässigkeit als

$$\begin{aligned} \hat{R}_{All} &= P(s_i \in S_1) + \sum_{s_i \in S_2} \Phi(S_i) \cdot P(S_i) \\ &= P(s_i \in S_1) + P(s_i \in S_2) \cdot \sum [\Phi | S_2] \\ &= \pi_1 + \pi_2 \cdot \sum [\Phi | S_2] \\ &\text{mit: } \pi_k = P(s_i \in S_k) \end{aligned} \tag{4.6}$$

geschätzt werden. Die von [134] vorgeschlagene Monte-Carlo-Simulation liefert neben einer Schätzung für R_{All} zusätzlich eine obere (UB) und untere Grenze (LB) für R_{All}. Dabei gilt: $\pi_1 \leq R_{All} \leq \pi_1 + \pi_2$.

4.5 Eine empirische Untersuchung verschiedener Verfahren zur Bestimmung der All-Terminal-Zuverlässigkeit

Der in Abschnitt 4.4.1.2 vorgestellte Dekompositionsansatz, dessen Parallelisierung, die obere Grenze aus Abschnitt 4.4.2.1 sowie die in Abschnitt 4.4.2.2 vorgestellten Monte-Carlo-Simulationstechniken zur Berechnung der All-Terminal-Zuverlässigkeit werden im Folgenden für Netzwerke aus der Literatur sowie für Netzwerke, welche mit Hilfe eines Topologieentwurfsverfahrens erstellt wurden, untersucht. Die dabei gewonnenen experimentellen Ergebnisse erlauben einen Vergleich der praktischen Einsetzbarkeit der unterschiedlichen Verfahren. Der Fokus liegt hierbei auf der Genauigkeit und der Laufzeit der Verfahren. Beide Eigenschaften sind wichtig für die Auswahl geeigneter Methoden zur Bestimmung der All-Terminal-Zuverlässigkeit. Diese beiden Größen sind auch entscheidend für die Leistungsfähigkeit von Planungsverfahren, welche einen automatisierten Entwurf von Netzwerktopologien unter Berücksichtigung von Zuverlässigkeitsmaßen ermöglichen [52, 49, 50, 63, 159].
Als Motivation für die nachfolgende empirische Untersuchung dient die mit Tabelle 4.1 gezeigte Auswertung der prozentualen Anteile der verschiedenen Methodenaufrufe an der Gesamtlaufzeit eines Netzwerktopologieplanungsverfahrens bei der Untersuchung eines Netzwerkes mit zehn Knoten und 45 Kanten. Mit Hilfe eines Profilers (gprof) wurde eine Implementierung des in [49] vorgestellten Verfahrens bei einer Populationsgröße von 200 und einer Bestimmung von R_{All} mittels einer einfachen Monte-Carlo-Simulation untersucht. Die Anteile der Methodenaufrufe an der Gesamtlaufzeit stellt Tabelle 4.1 dar. Die Auswertung des Profilers macht deutlich, dass die Berechnung der All-Terminal-

Tabelle 4.1: Analyse der Anteile der Methodenaufrufe für die Zuverlässigkeitsberechnung an der Gesamtlaufzeit eines Netzwerktopologieplanungsverfahrens

Anteil an der Gesamtlaufzeit	Methodenname	Beschreibung
44,9 %	Simple_MC	Berechnung von R_{All} mittels einfacher Monte-Carlo-Simulation
⋮		
5,5 %	GetNetworkCosts	Ermittlung der Kosten einer Netzwerkkonfiguration
⋮		
0,1 %	Rel_UpperBound	Ermittlung einer oberen Grenze für R_{All}

Zuverlässigkeit (Aufruf der Methode Simple_MC) mit zirka 45 % den größten Anteil an der Gesamtlaufzeit hat. Daraus folgt, dass für einen computergestützten Netzwerktopologieentwurf Verfahren zur Verfügung stehen müssen, die eine effiziente und hinreichend genaue Bewertung der All-Terminal-Zuverlässigkeit mit akzeptabler Laufzeit ermöglichen.

4.5.1 Probleminstanzen und Testumgebung

Für die experimentelle Untersuchung der Verfahren werden sechs Netzwerke (manzi18, dengiz8, türkei1, türkei2, türkei3 und belgien) aus verschiedenen Literaturstellen [52, 49, 134] sowie zwei Netzwerke (deutsch15 und deutsch20) basierend auf Kommunikationsnetzwerkstrukturen in Deutschland verwendet (siehe Anhang A.3).

Testprobleme – Das Testproblem manzi18 wurde in [134] vorgestellt. Das Netzwerk enthält 18 Knoten und 20 Verbindungen. Die Verbindungen haben unterschiedliche Zuverlässigkeiten zwischen 0,90 und 0,99. Die exakte All-Terminal-Zuverlässigkeit wird in [134] mit 0,8773 angegeben. Das Testproblem dengiz8 wurde in [52] untersucht und verbindet acht Knoten, die zufällig auf einem zweidimensionalen Raster der Größe 100x100 platziert wurden. Sämtliche Kanten haben die Zuverlässigkeit 0,9. Die All-Terminal-Zuverlässigkeit des Netzwerkes beträgt 0,8992.

Reale Netzwerktopologien – Die verbleibenden sechs Netzwerke repräsentieren unterschiedliche Netzwerktopologien in Europa. Die Netzwerke türkei1, türkei2 und türkei3 sind mögliche Lösungen für das in [49] vorgestellte Optimierungsproblem, bei dem die 19 größten universitären Forschungseinrichtungen der Türkei durch ein Netzwerk mit $R_{All} = 0,999$ zu verbinden sind. Eine detaillierte Beschreibung des Netzwerkes bietet Abschnitt 5.2.3 auf S. 109. Aus [134] wurde das größte Netzwerk (belgien) mit 52 Knoten und 73 Verbindungen entnommen. Für die Experimente wurde jeder Kante eine Zuverlässigkeit von 0,95 zugewiesen. In dieser Arbeit werden zusätzlich die Netzwerke deutsch15 und deutsch20 untersucht. Die Knoten der beiden Netze repräsentieren die 15 bzw. 20 größten Städte Deutschlands. Die Verbindungen zwischen den Knoten haben eine Zuverlässigkeit zwischen $0,7 - 0,9$. Die vorgestellten Netzwerke haben jeweils 29 bzw. 39 Kanten und wurden durch ein Topologieentwurfsverfahren (siehe [159]) generiert.

Testumgebung – Die Implementierung der in Abschnitt 4.4 vorgestellten Verfahren erfolgte in C++ unter Linux. Für die Parallelisierung des Dekompositionsansatzes wurde die freie MPI-Bibliothek LAM verwendet. Die lokalen Experimente erfolgten auf einem P4-2GHz-PC. Für das parallele Dekompositionsverfahren kam ein Cluster mit sieben P4-2GHz-PCs zum Einsatz. Das parallelisierte Dekompositionsverfahren teilt zunächst jedem Client einen Teilbereich von $\underline{T}$ zu. Jeder Client generiert dann die für seinen Bereich relevanten Komplementärereignisse und beginnt mit der Berechnung. Ist die Berechnung für eines dieser Ereignisse abgeschlossen, wird das Ergebnis an den Master-PC gesendet. Dieser prüft, ob die geforderte Schranke R_0 durch das empfangene Ergebnis überschritten wurde. Tritt dieser Fall ein, erhält der Client die Nachricht, die Berechnung abzubrechen. Anderenfalls wird dem Client signalisiert, mit der Berechnung fortzufahren. Da der Master-PC selbst keine Möglichkeit hat, die Clients über einen Abbruch der Berechnung zu informieren, wird die verteilte Berechnung erst gestoppt, nachdem alle Clients das nächste Ergebnis zum Master-PC gesendet haben.

4.5.2 Experimentelle Ergebnisse

Die Ergebnisse der nach [111] berechneten oberen Schranke zeigt Tabelle 4.2. Hierbei bezeichnet $|V|$ die Anzahl der Knoten und $|E|$ die Anzahl der vorhandenen Kanten im Netzwerk. Auf die Angabe der Laufzeit des Verfahrens zur Bestimmung der oberen

Tabelle 4.2: Ergebnisse für die Berechnung der oberen Schranke von R_{All}

Testproblem	$\|V\|$	$\|E\|$	Obere Schranke für R_{All}
türkei1	19	30	0,9992
türkei2	19	33	0,9991
türkei3	19	54	0,9995
deutsch15	15	29	0,9557
deutsch20	20	39	0,9547
dengiz8	8	9	0,9427
manzi18	18	20	0,9645
belgien	52	73	0,9408

Grenze wird verzichtet, da diese für alle Testinstanzen kleiner als eine Millisekunde ist. Die mittels des exakten Dekompositionsansatzes aus Abschnitt 4.4.1.2 erzielten Ergebnisse zeigt Tabelle 4.3. Für die Experimente kamen eine lokale Implementierung (dabei wurden sämtliche Ereignisse auf einem einzigen PC berechnet) sowie eine parallele Implementierung (hier wurde die Berechnung auf alle PCs des Clusters verteilt) des Verfahrens zum Einsatz. Der Dekompositionsansatz wurde hierbei entweder nach dem Erreichen einer vorgegebenen unteren Zuverlässigkeitsschranke R_0 abgebrochen oder komplett bis zum Ende durchgeführt.
Die Ergebnisse, welche durch eine vollständige Durchführung des Dekompositionsansatzes ermittelt wurden, stellen die exakten Werte für die All-Terminal-Zuverlässigkeit dar. Eine vollständige und exakte Berechnung der All-Terminal-Zuverlässigkeit war für die Netzwerke belgien und türkei3 nicht möglich. Für das Netzwerk belgien überstieg die Anzahl der Zustände, die für T zu untersuchen waren, die zur Verfügung stehende

Speicherkapazität. Die Berechnung für türkei3 wurde nach 14 Stunden abgebrochen und die bis zu diesem Zeitpunkt ermittelte Zuverlässigkeit angegeben.
Die mit Hilfe der beiden in Abschnitt 4.4.2.2 vorgestellten Monte-Carlo-Simulationstechniken ermittelten Ergebnisse sind in Tabelle 4.4 zusammengefasst. R_{All} bezeichnet die jeweils ermittelte All-Terminal-Zuverlässigkeit und t gibt die dafür benötigte Laufzeit in Sekunden an. Die für die Monte-Carlo-Simulation angegebenen Werte sind die Mittelwerte aus zehn unabhängig voneinander durchgeführten Läufen.

Tabelle 4.3: Ergebnisse des Dekompositionsansatzes bei lokaler und verteilter Berechnung

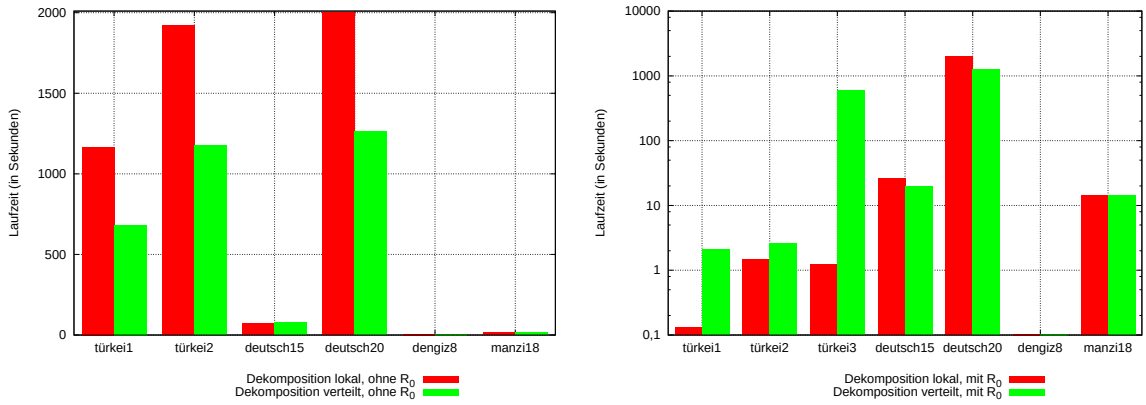
Test -problem	Dekomposition lokal auf einem Rechner					Dekomposition verteilt auf Cluster				
	Abbruch bei $R_{All} \geq R_0$			vollständig		Abbruch bei $R_{All} \geq R_0$			vollständig	
	R_0	R_{All}	$t(ins)$	R_{All}	t	R_0	R_{All}	t	R_{All}	t
türkei1	0,99	0,9990	0,13	0,9991	1167	0,99	0,9990009	2,09	0,9991	683
türkei2	0,99	0,9990	1,44	0,9991	1921	0,99	0,9990018	2,55	0,9991	1178
türkei3	0,99	0,9990	1,23	0,9990	>14h	0,99	0,9990005	592,62	0,9990	>14h
deutsch15	0,95	0,9500	25,70	0,9502	73,1	0,95	0,9500000	19,45	0,9502	76,51
deutsch20	0,95	0,9500	2000,00	0,9500	2000	0,95	0,9499999	1261,3	0,9500	1261
dengiz8	0,9	0,8992	0,00	0,8992	0	0,9	0,8992000	0,00	0,8992	0,00
manzi18	0,9	0,8773	14,20	0,8773	14,2	0,9	0,8773200	14,42	0,8773	14,42

Tabelle 4.4: Ergebnisse der Monte-Carlo-Simulationstechniken

Testproblem	Einfache Monte-Carlo (M=10000)		Fishman Monte-Carlo (M=10000)		Einfache Monte-Carlo (M=20000)		Fishman Monte-Carlo (M=20000)		Einfache Monte-Carlo (M=30000)		Fishman Monte-Carlo (M=30000)	
	R_{All}	t	R_{All}	t	R_{All}	t	R_{All}	t	R_{All}	t	R_{All}	t
türkei1	0,9990	0,13	0,9991	0,44	0,999	0,26	0,9991	0,85	0,9992	0,40	0,9991	1,27
türkei2	0,9992	0,14	0,9992	0,44	0,999	0,29	0,9991	0,88	0,9991	0,45	0,9991	1,39
türkei3	0,9994	0,23	0,9995	0,77	1,000	0,45	0,9995	1,53	0,9995	0,68	0,9995	2,31
deutsch15	0,9503	0,15	0,9500	0,44	0,951	0,31	0,9499	0,88	0,9498	0,46	0,9503	1,32
deutsch20	0,9505	0,20	0,9507	0,50	0,950	0,41	0,9508	1,29	0,9507	0,62	0,9507	1,81
dengiz8	0,9001	0,05	0,8990	0,13	0,899	0,10	0,8993	0,26	0,8984	0,15	0,8994	0,40
manzi18	0,8787	0,09	0,8769	0,33	0,878	0,20	0,8764	0,67	0,8773	0,31	0,8776	1,08
belgien	0,9077	0,33	0,9046	1,64	0,905	0,67	0,9050	3,28	0,9037	1,01	0,9049	5,39

4.5.3 Auswertung

Im Folgenden werden die Ergebnisse bezüglich der Laufzeit und der Lösungsqualität analysiert und interpretiert.
Laufzeit – Die Tabellen 4.3 und 4.4 zeigen, dass für Netze mit wenigen Knoten und Verbindungen wie z. B. dengiz8 der Dekompositionsansatz eine sehr schnelle Berechnung von R_{All} ermöglicht. Ein großer Nachteil des Verfahrens (ohne Abbruch bei $R_{All} \geq R_0$) liegt allerdings in der sehr langen Laufzeit für Netzwerke mit mehr als 29 Verbindungen für die lokale, nicht verteilte Dekomposition. Durch die Parallelisierung des Verfahrens



(a) lokal/verteilt exakt

(b) lokal/verteilt mit Schranke R_0

Abbildung 4.6: Vergleich der Laufzeiten des Dekompositionsansatzes für die lokale und die verteilte Berechnung

wird die Berechnung teilweise beschleunigt. Die Laufzeiten für die untersuchten Probleme mit $|E| > 29$ sind jedoch noch immer sehr hoch.
Abbildung 4.6(a) stellt die Laufzeiten des lokalen und verteilten/parallelen Dekompositionsverfahrens bei der Berechnung der exakten Zuverlässigkeit gegenüber. Auf die Angabe der Werte für die Netzwerke türkei3 und belgien wurde verzichtet, da für beide Netzwerke kein exakter Wert ermittelbar ist. Es ist erkennbar, dass die Parallelisierung des Verfahrens zu einer deutlichen Verkürzung der Laufzeit führt. Weiterhin stellt Abbildung 4.6(b) die Laufzeiten der lokalen und verteilten Berechnung mit Hilfe des Dekompositionsansatzes bei Verwendung einer Zuverlässigkeitsschranke R_0 gegenüber[23]. Für die Netzwerke deutsch15 und deutsch20 kann mit dem Einsatz der verteilten Berechnung ein Geschwindigkeitsvorteil gegenüber dem lokalen Verfahren erzielt werden. Für die Netzwerke türkei1, türkei2 und türkei3 sowie manzi18 ist jedoch die zentrale, nichtverteilte Berechnung der All-Terminal-Zuverlässigkeit schneller als die verteilte Berechnung mit Hilfe eines Clusters. Ursache hierfür ist das bei der Implementierung verwendete Verteilungs- und Kommunikationskonzept (siehe Abschnitt 4.5.1).
In Abbildung 4.7 werden die Laufzeiten für die Monte-Carlo-Simulationstechniken gegenübergestellt. Hierbei wird deutlich, dass für eine vorgegebene Anzahl an Kanten die Laufzeit der einfachen Monte-Carlo-Simulation linear mit zunehmendem Stichprobenumfang M zunimmt. Für das „Fishman Sampling“ ist festzuhalten, dass die Laufzeit des Verfahrens ebenfalls mit der Kantenanzahl ($|E|$) steigt. Aufgrund der Vorverarbeitung des Findens der Schnitte und Bäume, die für die Anwendung des „Fishman Sampling“-Plans notwendig sind, ist der Anstieg der Berechnungszeit im Vergleich zur einfachen Monte-Carlo-Simulation allerdings deutlich größer.

[23] Aufgrund der großen Unterschiede in den Laufzeiten für die lokale und die verteilte Berechnung wurde für Abbildung 4.6(b) eine logarithmische Einteilung der Ordinate gewählt.

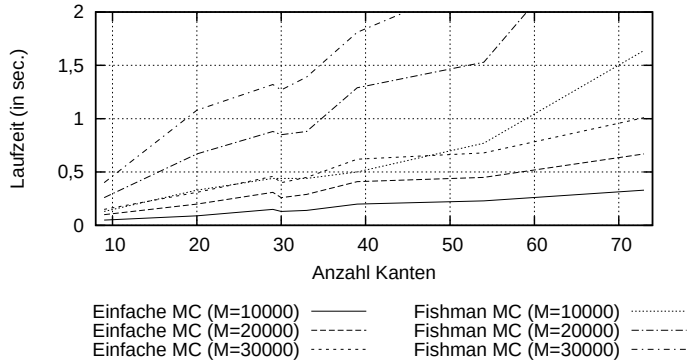


Abbildung 4.7: Vergleich der Laufzeiten bei Monte-Carlo-Simulation

Für alle untersuchten Netzwerke ist die einfache Monte-Carlo-Simulation aufgrund der simplen Schritte zur Generierung einer Stichprobe der „Fishman Sampling"-Methode bei identischer Stichprobenanzahl beim Vergleich der Laufzeiten überlegen.
Zusammenfassend kann bezüglich der Laufzeit der Verfahren festgestellt werden, dass beim Vorhandensein einer vorgegebenen minimalen Zuverlässigkeitsanforderung R_0 das Dekompositionsverfahren dann sehr schnell ist, wenn $R_{All} > R_0$. Die Laufzeit des Dekompositionsverfahrens liegt bei kleinen Netzwerken wie türkei1 und dengiz8 noch unter der einer einfachen Monte-Carlo-Simulation. Kritisch ist das Verfahren, wenn $R_{All} < R_0$. Wie man am Netzwerk deutsch20 erkennt, muss das Verfahren hier sämtliche Zustände untersuchen, was zu einer sehr langen Laufzeit führt. Darüber hinaus zeigen die Experimente, dass eine Parallelisierung des Dekompositionsansatzes nur für die Untersuchung von komplexeren Netzwerkstrukturen wie z. B. deutsch20 sinnvoll ist. Für kleinere Netze ist der Koordinationsaufwand zu hoch und eine lokale Implementierung vorzuziehen. Die Grenzen des Dekompositionsverfahrens zeigen sich bei der Bestimmung von R_{All} für Netzwerke wie türkei3 und belgien, bei denen die Berechnung aufgrund zu langer Laufzeiten und unzureichender Speicherressourcen abgebrochen wird. Für eine schnelle Ermittlung einer Schätzung für die All-Terminal-Zuverlässigkeit bietet sich bei größeren und komplexeren Netzen der Einsatz einer der vorgestellten Monte-Carlo-Simulationstechniken an.
Genauigkeit – Während die Dekompositionsmethode dann die exakte All-Terminal-Zuverlässigkeit liefert, wenn sie nicht vorzeitig abgebrochen wird, geben die anderen Verfahren nur einen Schätzwert bzw. eine obere Schranke für R_{All} an. In Tabelle 4.5 wird die prozentuale Abweichung zwischen der exakten Zuverlässigkeit R_{All} und der durch die Schätzungs-/Näherungsverfahren bestimmten Werte angegeben. Eine Angabe der Abweichungen für die Netzwerke türkei3 sowie belgien ist nicht möglich, da hierfür keine exakten Zuverlässigkeiten bekannt sind. Die obere Zuverlässigkeitsschranke weist für die Netzwerke dengiz8 und manzi18 eine sehr starke Abweichung vom tatsächlichen Wert auf. Betrachtet man die Monte-Carlo-Simulationstechniken, so ist festzustellen, dass im Allgemeinen nur geringe Abweichungen existieren. Häufig ist der Schätzwert für R_{All} jedoch geringer als die tatsächliche All-Terminal-Zuverlässigkeit. Diese Tatsache kann problematisch sein, da z. B. die Schätzer für das Netzwerk deutsch20 und dengiz8 (nur bei Simple MC mit M=10000) eine Zuverlässigkeit $R_{All} > R_0$ liefern, obwohl die

Tabelle 4.5: Prozentuale Abweichung der ermittelten Näherungswerte (nach Formel 4.5) bzw. Schätzwerte (ermittelt mittels einfacher Monte-Carlo-Simulation) von der exakten All-Terminal-Zuverlässigkeit R_{All}

	türkei1	türkei2	deutsch15	deutsch20	dengiz8	manzi18
Obere Schranke	0,01 %	0,00 %	0,58 %	0,50 %	4,84 %	9,94 %
Einfache MC (M=10000)	0,00 %	0,01 %	0,01 %	0,05 %	0,10 %	0,16 %
Einfache MC (M=20000)	0,00 %	0,02 %	0,07 %	0,03 %	−0,04 %	0,08 %
Einfache MC (M=30000)	0,01 %	0,00 %	−0,04 %	0,08 %	−0,09 %	0,00 %
Fishman MC (M=10000)	0,00 %	0,01 %	−0,02 %	0,07 %	−0,02 %	−0,05 %
Fishman MC (M=20000)	0,00 %	0,00 %	−0,03 %	0,08 %	0,01 %	−0,10 %
Fishman MC (M=30000)	0,00 %	0,00 %	0,01 %	0,07 %	0,02 %	0,04 %

tatsächliche Zuverlässigkeit knapp unter der geforderten Zuverlässigkeit R_0 liegt. Derartige Probleme können aber durch eine Erhöhung der Anzahl der Iterationen M einfach umgangen werden.

Für den Einsatz einer Methode zur Bestimmung der All-Terminal-Zuverlässigkeit sind isolierte Betrachtungen zur Laufzeit und Genauigkeit der Ergebnisse nur bedingt geeignet. In Planungsverfahren, in denen eine Vielzahl von Netzwerken zu untersuchen ist, müssen Methoden bereitstehen, die in akzeptabler Zeit eine Bestimmung der Zuverlässigkeit eines Netzwerkes ermöglichen. Betrachtet man beispielsweise die in [52, 49, 63, 159] vorgestellten Planungsverfahren, so müssen jeweils eine Vielzahl von Netzwerken bezüglich ihrer Zuverlässigkeit bewertet werden. Die Evaluierung der Zuverlässigkeit eines einzelnen Netzwerkes sollte somit schnell sein und in nur wenigen Sekunden erfolgen.

Für den Einsatz der Methoden ergeben sich anhand der vorgestellten Ergebnisse deshalb die nachfolgenden Vorgehensempfehlungen: Unabhängig von der Größe des zu untersuchenden Netzwerkes sollte zunächst eine obere Zuverlässigkeitsschranke ermittelt werden. Falls die Schranke die vorgegebene Zuverlässigkeit überschreitet, kann anschließend in Abhängigkeit von der Komplexität des Netzwerkes entschieden werden, ob mit Hilfe einer nachgelagerten exakten oder stochastischen Methode eine genauere Bestimmung der Zuverlässigkeit erfolgen soll.

Für Netzwerke mit bis zu zehn Knoten und maximal 30 Verbindungen bietet sich das lokale Dekompositionsverfahren an. Für größere Netzwerke ist bei der Zuverlässigkeitsanalyse ohne Angabe einer unteren Schranke den Monte-Carlo-Simulationstechniken der Vorzug zu geben. Wie die Ergebnisse für die Monte-Carlo-Simulationen zeigen, ist allerdings der Zuwachs an Sicherheit, der sich durch den Einsatz des modifizierten Stichprobenplans beim „Fishman Sampling“ ergibt, gering. Setzt man diesen ins Verhältnis zu der zusätzlich benötigten Rechenzeit, erweist sich die einfache Monte-Carlo-Simulation als die effizientere Methode. Wie in den Ergebnissen für die Netzwerke dengiz8 und deutsch20 zu sehen ist, reicht eine Schätzung mittels Monte-Carlo-Simulation nicht immer aus, so dass bei der Planung von Netzwerken eine exakte Untersuchung einzelner weniger Lösungen mit Hilfe der Dekompositionsmethode vertretbar ist. Die rechentechnischen Möglichkeiten für den Einsatz der Dekompositionsmethode sind allerdings auf Netzwerke mit ca. 20 Knoten und 40 Kanten begrenzt. Für größere Netzwerke ist damit nur der Einsatz der Monte-Carlo-Simulation möglich.

4.6 Zusammenfassung

Mit diesem Kapitel wurden die grundlegenden graphentheoretischen Definitionen eingeführt und eine Beschreibung des untersuchten Netzwerkdesignproblems mit Hilfe eines stochastischen Graphen vorgenommen. Es wurden unterschiedliche Verfahren zur Bewertung der Zuverlässigkeit von Kommunikationsnetzwerken vorgestellt und entsprechende Methoden zur Berechnung der Zuverlässigkeit von Netzwerken untersucht. Als geeignetes Maß für die Bewertung der Zuverlässigkeit von Kommunikationsnetzwerken wurde die All-Terminal-Zuverlässigkeit herausgearbeitet. Es wurden verschiedene Methoden für die exakte und approximative Berechnung der All-Terminal-Zuverlässigkeit betrachtet. Als exakte Verfahren wurden die vollständige Enumeration möglicher Zustände sowie ein mehrstufiger, parallelisierbarer Dekompositionsansatz vorgestellt. Als approximative Verfahren wurden eine einfache Monte-Carlo-Simulation und ein darauf aufbauendes Verfahren, welches einen „Fishman Sampling“-Plan verwendet, betrachtet. Weiterhin wurde ein Verfahren zur Ermittlung einer oberen Schranke für die All-Terminal-Zuverlässigkeit vorgestellt. In einer empirischen Untersuchung wurden die verschiedenen Verfahren anhand der Laufzeiten und der Genauigkeit der ermittelten Zuverlässigkeitswerte für eine Reihe von Testproblemen analysiert. Basierend auf den Ergebnissen wurde eine Empfehlung für den Einsatz der Methoden gegeben.
Die Experimente zeigen, dass das exakte Dekompositionsverfahren nur für relativ kleine Netzwerke mit weniger als 30-40 Kanten geeignet ist. Durch eine mögliche Verteilung der Berechnung mit einem Computerclusters können zwar Geschwindigkeitssteigerungen erreicht werden, jedoch stoßen auch verteilte Ansätze schnell an die Grenzen und sind nur für relativ kleine Netzwerke einsetzbar. Die Zuverlässigkeit von Netzwerken kann allerdings recht genau und mit vertretbarem Aufwand mit Hilfe von Monte-Carlo-Simulationen bestimmt werden. Insbesondere bei der Bewertung der Zuverlässigkeit von größeren Netzwerken sind derartige Verfahren exakten Verfahren vorzuziehen. Ein direkter Vergleich der einfachen Monte-Carlo-Simulation mit der erweiterten Methode mit „Fishman Sampling“ zeigt, dass der zusätzliche Aufwand für das „Fishman Sampling“ nicht gerechtfertigt ist und die einfachere Monte-Carlo-Simulation genauere Schätzungen bei gleichem Zeitaufwand liefert. Es lässt sich feststellen, dass in der Praxis durch den kombinierten Einsatz verschiedener Methoden die besten Ergebnisse bei der Berechnung der All-Terminal-Zuverlässigkeit von Netzwerken erzielbar sind. In einem ersten Schritt sollte eine obere Schranke für die Zuverlässigkeit eines Netzwerkes bestimmt werden. Eine derartige Schätzung kann sehr schnell durchgeführt werden und eignet sich gut für Problemstellungen, bei denen ein aufzubauendes Netzwerk eine vorgegebene Mindestzuverlässigkeit erfüllen muss. Falls die obere Grenze größer als die Mindestzuverlässigkeit ist, sollte im Anschluss daran entweder das Dekompositionsverfahren (bei kleinen Problemen mit weniger als 30 Kanten) oder eine Monte-Carlo-Simulation (bei größeren Problemen ($|V| \geq 30$)) eingesetzt werden.

5 Kommunikationsnetzwerkplanung unter Zuverlässigkeitsrestriktionen

Mit Kapitel 3 wurde die Planung von Kommunikationsnetzwerken unter Kosten- und Zuverlässigkeitsgesichtspunkten als kombinatorisches Optimierungsproblem dargestellt und genetische Algorithmen als geeignete Metaheuristik zur Lösung dieses Problems ausgewählt. Mit diesem Kapitel werden Planungsprobleme von Kommunikationsnetzwerken unter Zuverlässigkeitsrestriktionen untersucht. Bei diesen Planungsproblemen steht der Netzwerkplaner vor der Aufgabe, eine Netzwerktopologie mit minimalen Kosten zu entwerfen, deren All-Terminal-Zuverlässigkeit eine vorgegebene (minimal geforderte) untere Schranke R_0 nicht unterschreitet. Als Zielfunktion, welche es zu minimieren gilt, werden in diesem Kapitel die Kosten eines Netzwerkes betrachtet. Die All-Terminal-Zuverlässigkeit eines Netzwerkes bestimmt als Nebenbedingung ($R_{All} \geq R_0$), ob eine Lösung durch ein Planungsverfahren als gültig angesehen wird.
Im Rahmen der Arbeit wird im Folgenden zwischen Netzwerken mit identischen Verbindungszuverlässigkeiten ohne Technologieoptionen und Netzwerken mit Technologieoptionen unterschieden. Dabei gilt für erstere, dass sämtliche Verbindungen des Netzwerkes die gleiche Zuverlässigkeit $r(e_i)$ besitzen und sich die Kosten für eine Verbindung aus der Distanz zwischen den Terminal-Knoten der Verbindung berechnen. Für Netzwerke mit Technologieoptionen besteht für jede Verbindung die Möglichkeit zur Auswahl zwischen unterschiedlichen Zuverlässigkeiten, welche als Technologieoptionen l_k bezeichnet werden (siehe S. 28). Jeder Technologieoption ist ein Kostenfaktor zugeordnet. Die Kosten einer Verbindung ergeben sich aus der Distanz zwischen den Terminal-Knoten der Verbindung multipliziert mit dem Kostenfaktor der gewählten Technologieoption. Für Netzwerke mit Technologieoptionen kann jede Verbindung des Netzwerkes eine unterschiedliche Zuverlässigkeit besitzen.
Dieses Kapitel führt genetische Algorithmen ein, die eine Planung unter Verwendung identischer Zuverlässigkeiten sämtlicher Verbindungen ermöglichen. Weiterhin werden genetische Algorithmen vorgestellt, die die Planung bei unterschiedlichen Verbindungszuverlässigkeiten und der Möglichkeit zur Auswahl zwischen verschiedenen Technologieoptionen je Verbindung gestatten.
Das Kapitel ist wie folgt gegliedert: Zu Beginn erfolgt die Betrachtung des Kommunikationsnetzwerkplanungsproblems bei identischen Verbindungszuverlässigkeiten ohne Technologieoptionen. Hierfür wird eine neue Reparaturheuristik vorgestellt, welche auf Basis der Spannbäume in einem Graphen arbeitet. In einer experimentellen Studie mit verschiedenen aus der Literatur entnommenen sowie neu eingeführten Testproblemen wird ein genetischer Algorithmus, der die vorgestellte Reparaturheuristik verwendet, untersucht und einem straftermbasierten Ansatz gegenübergestellt. In Abschnitt 5.2.1 wird die Reparaturheuristik CURE eingeführt, die die Kommunikationsnetzwerkplanung mit verschiedenen Technologieoptionen je Verbindung, die sich in der Zuverlässigkeit und

den Kosten unterscheiden, ermöglicht. Abschnitt 5.2.2 stellt zwei evolutionäre Verfahren unter Verwendung der CURE-Reparaturheuristik vor. In einer experimentellen Untersuchung werden anschließend beide Verfahren gegenübergestellt. Das Kapitel schließt mit einer Zusammenfassung.

5.1 Kommunikationsnetzwerkplanung unter Zuverlässigkeitsrestriktionen bei identischen Verbindungszuverlässigkeiten ohne Technologieoptionen

Mit diesem Abschnitt werden Planungsprobleme untersucht, bei denen jede Verbindung des Netzwerkes die gleiche Zuverlässigkeit besitzt und der Netzwerkplaner nicht zwischen unterschiedlichen Technologieoptionen wählen kann.
Mit Abschnitt 3.5.4 wurden die verschiedenen Verfahren zur Behandlung der Nebenbedingung (hier die minimal geforderte All-Terminal-Zuverlässigkeit) vorgestellt. Exemplarisch für die bereits in Abschnitt 3.5.4 dargestellten Probleme, die mit dem Einsatz einer indirekten Behandlung einer Zuverlässigkeitsnebenbedingung mit Hilfe eines Straftermansatzes entstehen können, wird mit Abschnitt 5.1.1 die indirekte Berücksichtigung der All-Terminal-Zuverlässigkeitsnebenbedingung mittels Straftermansatz für die Planung von Kommunikationsnetzwerken bei identischen Verbindungszuverlässigkeiten ohne Technologieoptionen untersucht. Es wird gezeigt, dass der in [52] vorgeschlagene Straftermansatz zur Lösung des Problems ungeeignet ist. Daher wird mit dieser Arbeit die direkte Behandlung der Nebenbedingung mit Hilfe einer Reparaturheuristik vorgeschlagen und untersucht.

5.1.1 Analyse eines Planungsansatzes

Für die Kommunikationsnetzwerkplanung unter Zuverlässigkeitsrestriktionen bei identischen Verbindungszuverlässigkeiten ohne Technologieoptionen wird in [52] ein GA vorgestellt, welcher die All-Terminal-Zuverlässigkeitsnebenbedingung mit Hilfe eines Straftermansatzes in den GA integriert. Im Folgenden wird dieser Straftermansatz vorgestellt und anhand einiger ausgewählter Netzwerke untersucht. Es wird gezeigt, dass der gewählte Strafterm Lösungen generiert, welche die aufgestellte All-Terminal-Zuverlässigkeitsnebenbedingung nicht erfüllen. Das Problem wird in [52] wie folgt definiert:

$$\begin{aligned} C(G_N) &= \sum_{e_{v_i,v_j} \in E_N} c(e_{v_i,v_j}) \rightarrow \min \\ &\text{mit: } R_{All}(G_N) \geq R_0 \text{ (Zuverlässigkeitsnebenbedingung)} \end{aligned} \tag{5.1}$$

$C(G_N)$ entspricht den Kosten einer Lösung, die durch den Graphen G_N beschrieben wird und $c(e_{v_i,v_j})$ den Kosten der Verbindung zwischen den Knoten v_i und v_j. Als Zu-

Tabelle 5.1: Ergebnisse einer experimentellen Untersuchung des Strafterms nach [52] für die Kommunikationsnetzwerkplanung bei identischen Verbindungszuverlässigkeiten ohne Technologieoptionen

Testproblem	$r(e_i)$	R_0	C_{opt}	C_{Beste}	$R_{All}^{C_{Beste}}$
8-Knoten - Netzwerk 1	0,9	0,90	208	194,222	0,8991
8-Knoten - Netzwerk 1	0,9	0,95	247	223,073	0,9490
8-Knoten - Netzwerk 3	0,9	0,9	211	211	0,9022
8-Knoten - Netzwerk 3	0,9	0,95	245	233,067	0,9499

verlässigkeitsmaß wird die All-Terminal-Zuverlässigkeit $R_{All}(G_N)$ verwendet. Durch die Verwendung des Strafterms ergibt sich nach [52] als Zielfunktion:

$$\begin{aligned} C(G_N) &= \sum_{e_{v_i,v_j} \in E_N} c(e_{v_i,v_j}) + \omega \cdot (c_{max}(R_{All}(G_N) - R_0))^2 \rightarrow \min \\ \omega &= \begin{cases} 0, \text{wenn } R_{All}(G_N) \geq R_0 \\ 1, \text{wenn } R_{All}(G_N) < R_0 \end{cases} \\ c_{max} &= max\{\forall e_{v_i,v_j} \in E_N : c(e_{v_i,v_j})\} \end{aligned} \tag{5.2}$$

Durch den Strafterm gehen für jede ungültige Lösung die maximalen Verbindungskosten gewichtet mit der Abweichung von der geforderten Zuverlässigkeit quadratisch als Strafe in die Berechnung der Fitness einer Lösung mit ein.
Tabelle 5.1 zeigt die experimentellen Ergebnisse, die unter Verwendung des vorgeschlagenen Strafterms für einige der in [52] vorgestellten 8-Knoten-Testprobleme erzielt wurden. Für jedes der vier untersuchten Testprobleme zeigt die Tabelle die Kosten der optimalen Lösungen (C_{opt}), die Kosten der besten unter Verwendung des Strafterms gefundenen Lösung (C_{Beste}) und die für diese Lösung ermittelte All-Terminal-Zuverlässigkeit $R_{All}^{C_{Beste}}$. Für die Testprobleme gilt: Sämtliche Verbindungen haben die (identische) Zuverlässigkeit $r = 0,9$. Die Probleme wurden für $R_0 = 0,9$ und $R_0 = 0,95$ untersucht. An Stelle der in [52] verwendeten Monte-Carlo-Simulation wurde in der Untersuchung die in Abschnitt 4.4.1.2 vorgestellte Dekompositionsmethode als exaktes Berechnungsverfahren[24] zur Bestimmung der All-Terminal-Zuverlässigkeit eingesetzt.
Eine Analyse der Ergebnisse in Tabelle 5.1 zeigt, dass der GA unter Verwendung des Strafterms Lösungen als beste Lösung akzeptiert, deren exakte All-Terminal-Zuverlässigkeit $R_{All}^{C_{Beste}}$ unterhalb des geforderten R_0 liegt. Die dargestellten Beispiele machen deutlich, dass das in [52] vorgeschlagene Verfahren nicht in der Lage ist, die gestellte Zuverlässigkeitsnebenbedingung im GA-Design ausreichend zu berücksichtigen und als Planungsansatz somit ungeeignet ist.

[24] Die exakte Berechnung wurde bei Erreichung der geforderten All-Terminal-Zuverlässigkeit gestoppt.

5.1.2 STC – Eine Spannbaum-Reparaturheuristik für die Kommunikationsnetzwerkplanung

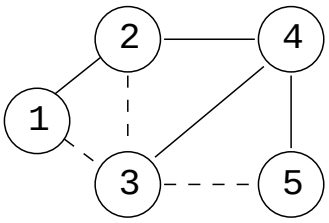
Mit Abschnitt 5.1.1 wurden die Defizite des in [52] vorgestellten Planungsansatzes für die Kommunikationsnetzwerkplanung bei identischen Verbindungszuverlässigkeiten ohne Technologieoptionen aufgezeigt. An Stelle eines straftermbasierten Ansatzes wird in diesem Abschnitt deshalb eine Reparaturheuristik vorgestellt, die eine Lösung, die eine geforderte Zuverlässigkeitsanforderungen nicht erfüllt, so repariert, dass diese gültig im Sinne der gestellten Nebenbedingungen ist.
Die Analyse der prozentualen Anteile der einzelnen Funktionsaufrufe an der Gesamtlaufzeit eines Planungsverfahrens in Abschnitt 4.5 zeigte, dass die Berechnung der All-Terminal-Zuverlässigkeit im Vergleich zu anderen Funktionen einen sehr hohen Berechnungsaufwand verursacht. Das Ziel des Entwurfes eines Verfahrens, welches ungültige (unzuverlässige) Lösungen in gültige Lösungen überführt, geht deshalb einher mit der Anforderung nach dem Einsatz einer Reparaturmethode, welche nur einen geringen Berechnungsaufwand verursacht. Aufgabe einer solchen Reparaturmethode ist es, die Verbindungen zu dem bestehenden Netzwerk hinzuzufügen, welche einen hohen Zuwachs an Zuverlässigkeit und einen geringen Netzwerkkostenzuwachs verursachen.
In [59] wird eine Methode zur schnellen Berechnung der Anzahl der Spannbäume bei Veränderung der Kantenanzahl in einem Graphen vorgestellt, die als Basis für die zu entwerfende Reparaturheuristik dient. Gemäß Definition 4.18 drückt die All-Terminal-Zuverlässigkeit die Wahrscheinlichkeit dafür aus, dass mindestens ein funktionierender Spannbaum in einem stochastischen Graphen existiert. Daraus lässt sich ableiten, dass je höher die Anzahl der Spannbäume in einem Graphen ist, desto höher auch dessen All-Terminal-Zuverlässigkeit (vgl. [59]) ist. Macht man sich diese Aussage für den Entwurf einer Reparaturheuristik zunutze, so muss ein solches Verfahren jene Verbindungen zu einem Netzwerk hinzufügen, die einen hohen Zuwachs der Anzahl der Spannbäume zur Folge haben.
Die Berechnung der Anzahl der Spannbäume in einem Graphen wird in [103, 26] wie folgt beschrieben: Für den Graphen G wird eine Matrix $|V| \times |V|$, die als Gradmatrix D des Graphen bezeichnet wird, erstellt. Für jedes Element d_{v_i,v_j} (mit $v_i, v_j \in V$) der Matrix D gilt:

$$\begin{aligned} d_{v_i,v_i} &= deg(v_i) \\ d_{v_i,v_j} &= -1, \text{wenn eine Kante } e_{v_i,v_j} \in E \text{ existiert} \\ d_{v_i,v_j} &= 0, \text{wenn keine Kante } e_{v_i,v_j} \in E \text{ existiert} \end{aligned} \tag{5.3}$$

Aus D wird die reduzierte Gradmatrix D' gebildet, welche entsteht, wenn man die letzte Spalte und die letzte Zeile der Matrix D eines Graphen löscht. Berechnet man die Determinante[25] der Matrix D', so erhält man die Anzahl der Spannbäume des Graphen.
Im Folgenden wird in einem Beispiel das Verfahren vorgestellt. Als Beispiel soll der in Abbildung 5.1(a) dargestellte Graph dienen[26]. Die Gradmatrix für den Beispielgraphen

[25] Die Determinante einer Matrix berechnet sich nach dem Laplace'schen Entwicklungssatz (vgl. [25, S. 269ff] für die Entwicklung nach der i-ten Zeile als $det(D) = \sum_{k=1}^{n}(-1)^{(i+k)} \cdot d_{i,k} \cdot det(D_{i,k})$.

[26] Die Kanten des Graphen sind als durchgezogene Linie dargestellt. Kanten, die nachträglich zum Graphen hinzugefügt werden können, sind gestrichelt dargestellt.



(a) Beispielgraph für Berechnung der Anzahl der Spannbäume

	v_1	v_2	v_3	v_4	v_5
v_1	1	-1	0	0	0
v_2	-1	2	0	-1	0
v_3	0	0	1	-1	0
v_4	0	-1	-1	3	-1
v_5	0	0	0	-1	1

(b) Gradmatrix für Beispielgraph

Abbildung 5.1: Beispielgraph mit zugehöriger Gradmatrix für die Berechnung der Anzahl der Spannbäume

zeigt Abbildung 5.1(b). Für den in Abbildung 5.1(a) dargestellten Graphen ergibt sich die Anzahl der Spannbäume (durch Streichen der letzten Spalte und Zeile der Gradmatrix und anschließender Berechnung der Determinante) somit als:

$$det(D') = det \begin{vmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & -1 & -1 & 3 \end{vmatrix} = 1 \tag{5.4}$$

Wie bereits in Abbildung 5.1(a) erkennbar ist, enthält der Graph genau einen Spannbaum. Fügt man die Kante e_{v_2,v_3} zwischen den Knoten v_2 und v_3 zum Graphen hinzu, so wird die Anzahl der Spannbäume (und somit dessen Ausfallsicherheit) erhöht. Wenn eine neue Kante zum Graphen hinzugefügt wurde, so genügt eine einfache Aktualisierungsoperation für die Matrix D, um die Anzahl der Spannbäume nach der Veränderung des Graphen zu berechnen. Nach der Aktualisierung der Gradmatrix berechnet sich die Anzahl der Spannbäume nach Gleichung 5.5 als:

$$det(D') = det \begin{vmatrix} 1 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 \\ 0 & -1 & 2 & -1 \\ 0 & -1 & -1 & 3 \end{vmatrix} = 3 \tag{5.5}$$

Für die STC (Spanning Tree Counting)-Reparaturheuristik wird die Anzahl der Spannbäume in einem Graphen verwendet, um invalide Lösungen, welche die geforderte Mindestzuverlässigkeit R_0 nicht aufweisen, durch das Hinzufügen von bisher nicht verwendeten Kanten zu verbessern.
Den Ablauf der Reparaturheuristik stellt Algorithmus 5.1 dar. Als Eingabeparameter erhält die Prozedur den Graphen G_N, der eine unzulässige Lösung für das Planungsproblem repräsentiert, den Graphen G, der sämtliche zur Verfügung stehenden Kanten enthält, und die geforderte All-Terminal-Zuverlässigkeit R_0. Im ersten Schritt wird eine Liste sämtlicher derzeit nicht in G_N verwendeter Kanten aus G erstellt. Die Elemente der Liste werden aufsteigend nach den Kosten der Kanten sortiert. Anschließend wird

Algorithmus 5.1 STC-Reparaturheuristik

Eingabe: G_N, G , R_0
Ausgabe: G_N

while $(R_{All}(G_N) < R_0)$ **do**
  $L = \{e_1, .., e_m\}$ mit Kantenmenge $e_i \in G$ und $e_i \notin G_N$ sortiert nach Kosten $c(e_i)$
  $trees = \#\text{SpanningTrees}(G_N)$
  for $i = 0; i < |L|; i{+}{+}$ **do**
    $E_N = E_N \cup \{e_i\}$
    $numTrees = \#\text{SpanningTrees}(G_N)$
    $tree_\triangle = numTrees - trees$
    $improveRatio = c(e_i)/tree_\triangle$
    $E_N = E_N \setminus \{e_i\}$
  end for
  Füge e_i mit dem besten $improveRatio$ zu G_N hinzu
  Berechne $R_{All}(G_N)$
end while

die Anzahl der Spannbäume in G_N bestimmt (#SpanningTrees(G_N)). In einer Schleife wird jede Kante der Liste temporär in G_N eingefügt und für diese Kante der Zuwachs ($tree_\triangle = numTrees - trees$) für die Anzahl der Spannbäume berechnet. Dieser Zuwachs wird ins Verhältnis zu den Kosten der jeweiligen Kante gesetzt. Nachdem sämtliche Kanten getestet wurden, wird die Kante mit dem besten Verhältnis zwischen Zuwachs für die Anzahl der Spannbäume und den Kosten zu G_N hinzugefügt und die All-Terminal-Zuverlässigkeit für G_N neu berechnet. Konnte durch die Reparatur noch nicht die geforderte Mindestzuverlässigkeit erreicht werden, wird die Prozedur wiederholt. Der Einsatz der STC-Reparaturheuristik in einem genetischen Algorithmus zur Planung zuverlässigerer Netzwerktopologien wird im nachfolgenden Abschnitt beschrieben.

5.1.3 STC-GA – Ein evolutionäres Planungsverfahren für die Kommunikationsnetzwerkplanung unter Verwendung der STC-Reparaturheuristik

Mit Abschnitt 3.5.4 wurden unterschiedliche Ansätze (direkte Berücksichtigung und indirekte Berücksichtigung) zur Behandlung einer Nebenbedingung in einem GA vorgestellt. Mit Abschnitt 5.1.1 konnte gezeigt werden, dass die Verwendung einer indirekten Berücksichtung einer Nebenbedingung in Form eines Strafterms für das hier untersuchte Problem kritisch ist und im Ergebnis zu ungültigen Lösungen (im Sinne der Nebenbedingung) führen kann. Für das nachfolgende Planungsverfahren wird daher die Verwendung einer direkten Behandlung der Zuverlässigkeitsnebenbedingung in einem GA mit Hilfe der Reparaturheuristik aus Abschnitt 5.1.2 vorgeschlagen. Dieser GA wird für aus der Literatur entnommene und mit dieser Arbeit neu eingeführte Testprobleme für die Kommunikationsnetzwerkplanung bei identischen Verbindungszuverlässigkeiten ohne Technologieoptionen untersucht und die Ergebnisse einem straftermbasierten Ansatz gegenübergestellt.

Zielsetzung des im Folgenden vorgestellten evolutionären Verfahrens ist es, für eine gegebene Menge von Verbindungen, die eine feste Anzahl an vorgegebenen Netzwerkknoten verbindet, unter Beachtung einer minimal geforderten All-Terminal-Zuverlässigkeit R_0 eine kostenminimale Kantenmenge auszuwählen. Wie in Abschnitt 4.2 beschrieben, wird das Netzwerk hierfür als stochastischer ungerichteter Graph modelliert. Es gelten die Modellannahmen aus Abschnitt 3.1.1. Zusätzlich wird das Modell dahingehend erweitert, dass alle Kanten des Graphen die gleiche Zuverlässigkeit $r(e_i)$ besitzen. Die Zielfunktion für das in diesem Abschnitt betrachtete Optimierungsproblem lautet:

$$f(y) = C(G_N) = \sum_{e_{v_i,v_j} \in E_N} c(e_{v_i,v_j}) \rightarrow \min$$
$$\text{mit: } R_{All}(G_N) \geq R_0 \text{ (Zuverlässigkeitsnebenbedingung)} \tag{5.6}$$

Wie in Abschnitt 4.2 eingeführt, entspricht $C(G_N)$ den Kosten einer Lösung, die durch den Graphen G_N beschrieben wird und $c(e_{v_i,v_j})$ den Kosten der Verbindung zwischen den Knoten v_i und v_j.

5.1.3.1 Experimentelles Design

Für die experimentelle Untersuchung der entworfenen Reparaturheuristik wurde ein Steady-State GA verwendet. Der Steady-State GA wurde ausgewählt, da dieser im Gegensatz zu einem einfachen (kanonischen) GA in jeder Generation zur Erzeugung der nächsten Generation die Eltern- und Kinderpopulation betrachtet und aus beiden Populationen die jeweils besten Lösungen in die nächste Generation übernimmt (vgl. Abschnitt 3.5.2.3, S. 48). Auf diese Weise wird der Elitismus[27] sichergestellt und ein höherer Selektionsdruck erzeugt. Der höhere Selektionsdruck führt dazu, dass der Steady-State GA schneller mit Lösungen hoher Güte konvergiert.
Als Zielfunktionswert für den GA werden die Kosten einer Lösung verwendet. Für den GA wird eine Lösung binär durch ein Genom der Länge $|E|$ kodiert. Abbildung 5.2 zeigt die Kodierung eines Graphen[28] mit vier Knoten und sechs möglichen Kanten[29] als Genom. Der für die Experimente[30] verwendete GA nutzt einen Uniform-Crossover Ope-

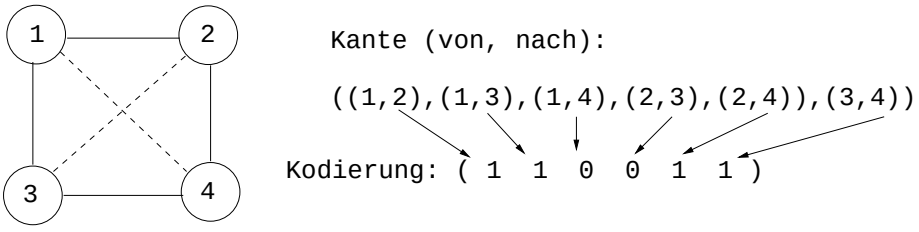


Abbildung 5.2: Kodierung eines Netzwerkes als Genom

[27] Dies bezeichnet die Fähigkeit, die jeweils besten Lösungen einer Generation in die nächste Generation zu übernehmen.
[28] Mögliche, aber derzeit nicht im Graphen verwendete Kanten sind gestrichelt dargestellt.
[29] Aus Gründen der Vereinfachung wurde für die Darstellung kein vollständig verbundener ungerichteter Graph gewählt.

rator (siehe S. 47) und setzt eine Flip-Mutation (siehe S. 48) ein. Die initiale Population wurde aus zufällig generierten Lösungen erstellt. Die Wahrscheinlichkeit (p_{init}), dass eine Verbindung zwischen zwei Knoten durch das Initialisierungsverfahren erstellt wird, beträgt 40 %. Das Initialisierungsverfahren stellt sicher, dass jede initiale Lösung mindestens 2-fach-kantenzusammenhängend ist. Zur Beschleunigung der All-Terminal-Zuverlässigkeitsberechnung wurde der Test mit der oberen Schranke aus Abschnitt 4.4.2.1 eingesetzt. Nur für Lösungen mit $R_{obereSchranke} \geq R_0$ wird eines der Verfahren aus Abschnitt 4.4.1 bzw. 4.4.2.2 zur Berechnung der exakten All-Terminal-Zuverlässigkeit oder eine Monte-Carlo basierte Schätzung verwendet. Für Netzwerke mit weniger als 16 Knoten wurde für die Berechnung von R_{All} das exakte Berechnungsverfahren nach [32] genutzt. Dabei wurde die Berechnung gestoppt, wenn der Wert für $R_{All} \geq R_0$ war. Für Netzwerke mit mehr als 15 Knoten wurde eine einfache Monte-Carlo-Simulation mit jeweils $M = 30.000$ Stichproben durchgeführt[31].
Wenn der GA bei der Initialisierung, durch die Rekombination oder durch die Mutation eine Lösung generiert, welche die geforderte All-Terminal-Zuverlässigkeitsnebenbedingung $R_{All} \geq R_0$ nicht erfüllt, so wird diese Lösung zur Reparatur an die STC-Reparaturheuristik übergeben. Diese fügt zu der Lösung solange Kanten hinzu, bis die geforderte Zuverlässigkeit erreicht wird[32].
Für Vergleichszwecke wurde ein weiterer GA, der mit den gleichen GA-Operatoren arbeitet, jedoch an Stelle der STC-Reparatur invalide Lösungen mit zusätzlichen Kosten in der Fitnessbewertung abstraft, entworfen. Den Algorithmus zur Bildung der Strafkosten stellt Algorithmus 5.2 dar.

Algorithmus 5.2 Strafterm-Funktion für die Planung bei identischen Verbindungszuverlässigkeiten ohne Technologieoptionen

Eingabe: C_{min}, G_N, R_0
Ausgabe: $Strafe$

```
1: Strafe = 0
2: if C_Min = 0 then
3:    Strafe = ∞
4: else
5:    while (C(G_N) + Strafe) < C_min do
6:       Strafe+ = (R_0 − R_All(G_N))^2 · C_min
7:    end while
8: end if
```

[30] Die Auswahl der hier vorgestellten GA-Operatoren und -Parameter wurden in einer experimentellen Studie als die beste Parameterkombination ermittelt.
[31] Wurde eine Lösung gefunden, deren Kosten kleiner als die bisher kleinsten Kosten waren, so wurde zusätzlich eine Monte-Carlo-Zuverlässigkeitsberechnung mit $M = 50.000$ Stichproben durchgeführt.
[32] Für die Testprobleme wurde sichergestellt, dass die geforderte All-Terminal-Zuverlässigkeit durch die Verwendung sämtlicher Verbindungen des Eingabegraphen eingehalten wird.

Tabelle 5.2: Parameter für STC-GA und Strafterm-GA

GA	Steady-State-GA
Crossover-Operator	Uniform-Crossover, $p_{cross} = 0,9$
Mutations-Operator	Flip-Mutation, $p_{mut} = 0,01$
Initialisierung von P_0	zufallsbasiert, $p_{init} = 0,4$
Bestimmung von R_{All}	Für $\|V\| > 15$ Berechnung obere Schranke sowie einfache Monte-Carlo-Simulation sonst lokaler Dekompositionsansatz
p_{mut}	0,01
p_{cross}	0,9
pop	100
Anzahl GA-Läufe	10

Als Ergebnis liefert der Algorithmus die Strafkosten $Strafe$. Solange die Kosten C_{min} gleich Null sind[33], wird die $Strafe$ auf ∞ gesetzt. Ist $C_{min} > 0$, so errechnen sich die Strafkosten ($Strafe$) aus C_{min} gewichtet mit der quadratischen Abweichung von $(R_0 - R_{All}(G_N))$. Zusätzlich wird mit der while-Schleife (Zeile 5) sichergestellt, dass die Kosten der besten, gültigen Lösung stets kleiner sind als die Kosten der besten mit dem Strafterm abgestraften Lösung. Anders als für den in Abschnitt 5.1.1 untersuchten Strafterm ist damit für den hier vorgestellten Strafterm sichergestellt, dass die Kosten für keine der mit einem Strafterm bewerteten Lösungen besser als die minimalen Kosten einer gültigen Lösung sind.
Für jede Lösung mit $R_{All}(G_N) < R_0$ wird mittels des Algorithmus 5.2 eine Strafe (zusätzliche Kosten) berechnet. Die Fitnessfunktion des GA lautet:

$$f(y) = C(G_N) + \omega \cdot \text{Strafe mit: } \omega = \begin{cases} 0, \text{ wenn } R_{All}(G_N) \geq R_0 \\ 1, \text{ wenn } R_{All}(G_N) < R_0 \end{cases} \tag{5.7}$$

Ein direkter Vergleich des hier vorgeschlagenen GAs (mit der STC-Reparaturheuristik) mit dem in [52] vorgeschlagenen Strafterm-basierten GA ist nicht möglich, da dieser wie in Abschnitt 5.1.1 gezeigt, ungültige Lösungen generiert.
Für die Experimente wurden die folgenden GA Parameter verwendet: Ein Steady-State-GA mit sich zu 50 % überlappenden Populationen[34]. Die Crossover-Wahrscheinlichkeit beträgt $p_{cross} = 0,9$, die Mutationswahrscheinlichkeit beträgt $p_{mut} = 0,01$ und die Populationsgröße wurde auf $pop = 100$ festgelegt. Ein GA-Durchlauf wurde nach 200 Generationen oder wenn in den letzten 20 Generationen keine Verbesserung gefunden wurde, angehalten. Für jedes Testproblem wurden zehn voneinander unabhängige Läufe durchgeführt. Beide GAs wurden mit der GALib [188] unter Linux implementiert und getestet. Tabelle 5.2 fasst die für den STC-GA und Strafterm-GA festgelegten Parameter zusammen. Die Experimente wurden auf einem P4-2-GHz System durchgeführt.

[33] In diesem Fall wurde noch keine gültige Lösung durch den GA gefunden.

[34] Der prozentuale Anteil für die Überlappung der Populationen bestimmt dabei den Anteil der Lösungen in der Elternpopulation, die durch bessere Lösungen aus der Kindergeneration ersetzt werden.

Tabelle 5.3: Ergebnisse für die Netzwerke aus [52] mit STC-GA und Strafterm-GA ($r(e_i) = 0,9$ und $R_0 = 0,9$)

				STC-GA					Strafterm-GA				
	$\|V\|$	$\|E\|$	C_{opt}	C_{Beste} μ	C_{Beste} σ	C_{Beste}	#Evals	t_{conv}	C_{Beste} μ	C_{Beste} σ	C_{Beste}	#Evals	t_{conv}
1	6	15	231	231,0	0,0	231	288	8,3	238,5	7,3	231	158	1,1
2	6	15	239	239,0	0,0	239	184	7,1	249,5	9,3	239	280	1,1
3	6	15	227	227,0	0,0	227	171	7,5	235,7	8,0	227	412	1,3
4	6	15	212	212,0	0,0	212	278	9,5	219,9	5,7	212	197	1,1
5	6	15	184	184,0	0,0	184	193	7,7	198,6	9,0	184	190	1,0
1	7	21	189	189,0	0,0	189	598	19,7	218,9	11,9	207	260	1,9
2	7	21	184	184,0	0,0	184	403	17,5	215,5	17,1	193	299	2,2
3	7	21	243	248,3	4,6	243	418	18,4	305,0	14,6	283	302	1,9
4	7	21	129	130,1	1,4	129	657	19,9	152,9	7,0	140	606	2,5
5	7	21	124	124,0	0,0	124	319	16,2	184,9	28,4	124	291	1,7
1	8	28	208	208,0	0,0	208	401	33,9	292,2	21,6	261	458	3,9
2	8	28	203	203,0	0,0	203	507	32,6	282,6	38,1	203	708	5,5
3	8	28	211	211,0	0,0	211	685	34,3	297,8	29,5	218	527	4,5
4	8	28	291	291,3	0,9	291	710	33,7	361,8	19,8	321	412	3,8
5	8	28	178	179,5	1,5	178	887	39,1	248,0	22,1	201	357	3,5
1	9	36	239	239,0	0,0	239	790	73,0	387,2	51,7	314	833	8,4
2	9	36	191	194,0	3,3	191	979	56,9	307,5	75,1	218	1.588	13,7
3	9	36	257	262,6	2,3	257	1.051	86,3	368,3	33,3	318	530	7,1
4	9	36	171	171,0	0,0	171	714	50,7	280,3	46,1	202	1.234	10,7
5	9	36	198	198,0	0,0	198	809	63,3	312,3	44,5	245	854	8,5
1	10	45	131	131,7	0,5	131	1.282	142,2	191,6	43,4	146	2.488	30,1
2	10	45	154	154,0	0,0	154	940	142,7	279,3	64,0	192	1.669	24,6
3	10	45	267	267,6	1,3	267	1.207	176,6	463,2	71,0	304	665	16,1
4	10	45	263	263,0	0,0	263	791	100,1	466,8	78,4	298	1.187	24,6
5	10	45	293	301,4	5,5	293	1.208	158,5	469,4	78,9	320	1.782	24,0
1	11	55	246	246,0	0,0	246	1.200	365,6	386,2	49,7	332	2.173	57,2

5.1.3.2 Experimentelle Ergebnisse und Auswertung

Die Tabellen 5.3, 5.4, 5.5 und 5.6 zeigen die mit Hilfe der genetischen Algorithmen ermittelten Ergebnisse. Die Testprobleme für die Tabellen 5.3, 5.4 und 5.5 wurden aus [52] entnommen. Testprobleme mit einer gleichen Anzahl an Knoten und Kanten unterscheiden sich in der Position der Knoten auf einer 100x100 großen Fläche[35]. Die Zuverlässigkeit für alle Verbindungen des Netzwerkes ist identisch ($r(e_i) = 0,9$ bzw. $r(e_i) = 0,95$). Die Kosten einer Verbindung entsprechen der euklidischen Distanz zwischen zwei Knoten.

[35] Die Maßeinheit für die Entfernungen wird in [52] nicht angegeben.

Die Netzwerke aus Tabelle 5.6 auf S. 100 werden in dieser Arbeit neu eingeführt. Die Knoten der Netzwerke repräsentieren die jeweils größten Städte[36] in Deutschland[37]. Die Verbindungen im Netzwerk haben die identische Zuverlässigkeit $r(e_i) = 0,95$. Jede Tabelle zeigt für ein Testproblem die Anzahl der Knoten $|V|$, die Anzahl der Verbindungen $|E|$, die Kosten der besten Lösung (C_{opt}), den Mittelwert (μ) sowie die Standardabweichung (σ) der Kosten der durchschnittlich besten Lösung C_{Beste} (gemittelt über alle zehn Läufe), die Kosten der besten in allen zehn Läufen gefundenen Lösung C_{Beste}, die durchschnittlich in einem Lauf durchgeführte Anzahl der Fitnessbewertungen $\#Eval$ und die Laufzeit[38] t_{conv} in Sekunden (aufsummiert für alle zehn Läufe).
Für die Probleme deutsch15, deutsch20, deutsch25 und deutsch30 ist keine optimale Lösung bekannt. In Tabelle 5.6 wird deshalb mit $C_{BestEver}$ der Zielfunktionswert für die beste bisher gefundene Lösung angegeben.
Die Ergebnisse in den Tabellen 5.3, 5.4 und 5.5[39] zeigen, dass der hier vorgeschlagene genetische Algorithmus mit der STC-Reparaturfunktion mit einer Ausnahme (das Netzwerk 1 mit $|V| = 10, |E| = 45, r(e_i) = 0,9, R_0 = 0,95$) für sämtliche Testprobleme die optimale Lösung in mindestens einem von zehn Testläufen findet. Betrachtet man zusätzlich die Ergebnisse für die durchschnittlich beste Lösung über alle zehn Läufe, so ist festzuhalten, dass der GA stets mit sehr guten (nahe am Optimum gelegenen) Lösungen stoppt.
Betrachtet man die mit Hilfe des Strafterm-GAs erzielten Ergebnisse, so lässt sich feststellen, dass dieser GA nicht für alle aus [52] entnommenen Testprobleme die optimale Lösung findet. Außerdem zeigen die dargestellten Ergebnisse, dass die mittels des Strafterm-GAs gefundenen besten Lösungen eine breite Streuung aufweisen (gemessen an μ und σ für C_{Beste}). Es kann dabei festgestellt werden, dass die Streuung der Ergebnisse (gemessen an σ) mit der Problemgröße wächst. Für kleine Probleme mit sechs und acht Knoten (und 15 bzw. 21 Kanten) findet der Strafterm-GA in vielen der durchgeführten Experimente (GA-Läufe) ähnlich gute Lösungen. Für Testinstanzen mit $|V| \geq 8$ (und $|E| \geq 28$) zeigen die dargestellten Ergebnisse ein Ansteigen der Standardabweichung für die Kosten der besten gefundenen Lösung (C_{Beste}) aus allen durchgeführten GA-Läufen. Eine Gegenüberstellung beider Ansätze macht deutlich, dass die Lösungsqualität (gemessen an den durchschnittlichen besten Kosten der gefundenen Lösungen) des GAs unter Verwendung einer Reparaturheuristik deutlich höher ist. Vergleicht man den Aufwand für das Erreichen der Lösungen, so zeigen die Ergebnisse, dass der Strafterm-GA weniger Fitnessevaluationen benötigt und schneller ist. Hierfür sind zwei Ursachen zu nennen. Zum einen, wird jede Lösung bei Verwendung eines straftermbasierten Ansatzes stets nur einmal durch die Fitnessfunktion bewertet. Im Gegensatz dazu muss eine Lösung beim Einsatz einer Reparaturfunktion unter Umständen mehrmals wäh-

[36] Die Knoten des Netzwerkes deutsch15 repräsentieren beispielsweise die 15 größten Städte Deutschlands.
[37] Eine Liste der verwendeten Städte sowie der Entfernungsmatrix für die Probleme wird in Anhang A.3 gegeben.
[38] Aufgrund der geringen Laufzeiten (< 1 Sekunde) für Testprobleme mit sechs und acht Knoten wird die Laufzeit aller GA-Läufe dargestellt.
[39] Die optimale Lösung für das Netzwerk 1 mit 11 Knoten in Tabelle 5.4 und Tabelle 5.5 wurde in [52] nicht angegeben. Der Wert entspricht der besten gefundenen Lösung.

Tabelle 5.4: Ergebnisse für die Netzwerke aus [52] mit STC-GA und Strafterm-GA ($r(e_i) = 0,9$ und $R_0 = 0,95$)

				STC-GA					Strafterm-GA				
				C_{Beste}					C_{Beste}				
	$\|V\|$	$\|E\|$	C_{opt}	μ	σ	C_{Beste}	#Evals	t_{conv}	μ	σ	C_{Beste}	#Evals	t_{conv}
1	6	15	254	254,0	0,0	254	207	9,1	278,4	5,0	273	214	2,7
2	6	15	286	286,0	0,0	286	152	9,0	304,2	12,1	286	357	2,1
3	6	15	275	275,0	0,0	275	173	9,2	284,4	3,8	275	211	2,2
4	6	15	255	255,0	0,0	255	128	8,9	266,3	5,6	255	114	1,7
5	6	15	198	198,0	0,0	198	270	9,9	225,6	8,3	209	323	2,1
1	7	21	205	205,0	0,0	205	391	20,8	255,1	3,9	249	263	4,0
2	7	21	209	209,0	0,0	209	276	17,0	268,2	8,8	258	215	3,3
3	7	21	268	268,0	0,0	268	329	22,8	285,7	13,4	268	426	4,2
4	7	21	143	143,0	0,0	143	266	19,2	175,8	7,7	163	377	3,8
5	7	21	153	153,0	0,0	153	301	20,8	198,1	31,1	153	511	4,8
1	8	28	247	247,0	0,0	247	445	46,3	296,6	29,0	249	573	13,1
2	8	28	247	247,0	0,0	247	600	59,5	307,9	24,4	273	210	6,4
3	8	28	245	245,6	0,9	245	631	55,6	315,1	12,2	298	479	9,9
4	8	28	334	334,4	0,8	334	800	63,4	386,1	18,3	365	363	14,1
5	8	28	202	202,0	0,0	202	746	61,0	247,7	21,5	202	513	9,1
1	9	36	286	288,5	7,5	286	819	137,0	442,7	34,6	379	422	20,5
2	9	36	220	220,0	0,0	220	630	131,8	402,8	74,0	257	902	38,5
3	9	36	305	306,0	0,0	305	902	159,0	430,2	40,9	341	487	23,3
4	9	36	219	221,8	3,6	219	825	134,6	328,5	52,9	219	1.072	37,9
5	9	36	237	237,0	0,0	237	600	132,8	376,3	51,0	289	632	24,8
1	10	45	153	156,4	0,9	154	1.199	526,7	247,4	61,0	158	1.983	87,1
2	10	45	197	197,8	1,0	197	1.067	1.067,3	281,0	84,9	210	2.586	227,8
3	10	45	311	321,1	6,7	311	962	350,9	492,6	89,4	357	1.042	110,5
4	10	45	291	291,0	0,0	291	845	369,1	487,0	83,6	319	1.340	111,7
5	10	45	358	358,0	0,0	358	834	418,7	536,2	94,1	418	1.123	100,8
1	11	55	277	277,0	0,0	277	1.049	1.306,5	489,3	46,6	397	1.484	292,7

rend der Reparatur bewertet werden. Zum anderen wird beim straftermbasierten Ansatz für Lösungen mit $R_{All} < R_0$ keine exakte All-Terminal-Zuverlässigkeitsberechnung durchgeführt. Während der STC-GA invalide Lösungen während der Suche stets durch valide Lösungen ersetzt, bleiben diese Lösungen beim straftermbasierten Ansatz unverändert. Für den STC-GA bedeutet dies, dass für jede Lösung mindestens eine exakte All-Terminal-Zuverlässigkeitsberechnung durchgeführt wird. Im Gegensatz dazu werden invalide Lösungen im straftermbasierten Ansatz lediglich mit der deutlich schneller arbeitenden Methode zur Bestimmung einer maximalen All-Terminal-Zuverlässigkeit aus Abschnitt 4.4.2.1 getestet. Experimentelle Untersuchungen haben gezeigt, dass die Lösungsgüte des Strafterm-GAs auch mit einer höheren Populationsgröße sowie einer größeren Anzahl der zu durchlaufenden Generationen (vgl. hierzu auch Abbildung 5.4 auf S. 101) keinen Zuwachs an Lösungsqualität (gemessen an den Kosten der besten gefundenen Lösung) zur Folge hat.

Tabelle 5.5: Ergebnisse für die Netzwerke aus [52] mit STC-GA und Strafterm-GA ($r(e_i) = 0,95$ und $R_0 = 0,95$)

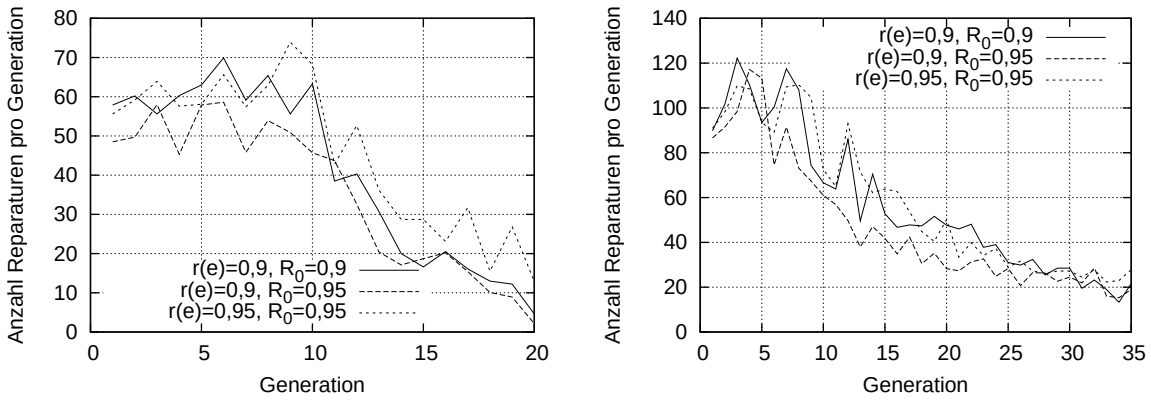
				STC-GA					Strafterm-GA				
	$\|V\|$	$\|E\|$	C_{opt}	C_{Beste} μ	σ	C_{Beste}	#Evals	t_{conv}	C_{Beste} μ	σ	C_{Beste}	#Evals	t_{conv}
1	6	15	227	229,5	5,5	227	758	8,4	250,5	4,6	246	189	1,1
2	6	15	213	213,0	0,0	213	284	6,4	224,2	6,3	213	331	1,1
3	6	15	190	190,0	0,0	190	201	6,0	218,9	11,6	207	187	0,9
4	6	15	200	201,0	3,0	200	436	7,1	217,6	9,5	200	180	1,0
5	6	15	179	179,0	0,0	179	762	8,1	185,8	2,5	184	212	1,0
1	7	21	185	185,0	0,0	185	977	11,0	217,6	16,6	185	406	1,6
2	7	21	182	182,0	0,0	182	782	9,7	237,0	8,8	224	398	1,4
3	7	21	230	232,4	2,0	230	416	9,0	262,3	22,2	230	428	1,4
4	7	21	122	122,7	2,1	122	846	10,3	152,2	7,1	141	422	1,5
5	7	21	124	124,0	0,0	124	291	8,2	200,5	15,7	172	389	1,8
1	8	28	179	179,5	1,5	179	522	16,1	262,6	33,6	179	471	2,0
2	8	28	194	194,6	0,9	194	836	19,2	251,9	36,7	194	855	2,7
3	8	28	197	197,9	1,6	197	1.070	19,7	277,0	31,0	221	413	2,1
4	8	28	276	282,0	5,3	276	805	18,2	331,9	25,0	286	301	1,7
5	8	28	173	175,8	1,5	173	1.133	18,9	226,5	22,7	189	625	2,2
1	9	36	209	209,0	0,0	209	683	29,6	373,8	28,6	326	383	3,4
2	9	36	171	172,2	1,5	171	1.261	41,4	310,2	78,6	209	1.340	6,3
3	9	36	233	234,6	4,8	233	1.103	36,7	358,8	54,3	277	725	4,7
4	9	36	151	153,7	4,1	151	757	33,9	257,0	46,0	170	963	5,7
5	9	36	185	185,3	0,9	185	1.062	34,7	259,1	38,2	213	1.061	6,2
1	10	45	121	124,2	1,3	121	1.614	74,5	183,6	43,6	137	2.210	16,5
2	10	45	136	136,0	0,0	136	891	53,3	241,8	76,7	145	2.002	19,3
3	10	45	236	241,4	4,4	236	1.096	64,6	395,7	53,8	291	1.648	12,2
4	10	45	245	245,3	0,5	245	1.037	66,2	423,8	80,6	269	1.434	11,4
5	10	45	268	270,4	3,7	268	1.352	80,9	440,7	65,0	334	1.303	10,6
1	11	55	210	212,8	3,6	210	1.543	152,9	368,2	58,2	279	1.504	20,0

Ähnliche Ergebnisse wie für die aus [52] entnommenen Testprobleme sind für die Testprobleme deutsch15, deutsch20, deutsch25 und deutsch30 festzustellen (vgl. Tabelle 5.6). Für sämtliche Probleme werden mit dem STC-GA die Lösungen mit den geringsten Kosten und somit den besten Fitnesswerten gefunden. Die Kosten der besten durch den STC-GA gefundenen Lösungen für die Testprobleme deutsch15, deutsch20, deutsch25 und deutsch30 sind zirka 40 % geringer als die besten durch den Strafterm-GA gefundenen Lösungen. Betrachtet man die über alle zehn durchgeführten Läufe gemittelten Kosten der jeweils besten Lösung für die untersuchten Testinstanzen deutsch15, deutsch20 und deutsch25, so zeigen die Ergebnisse, dass die mittlere Lösungsqualität (Mittelwert μ für C_{Beste}) für den STC-GA zirka 50 % geringer ist als die des Strafterm-GAs. Für das Testproblem deutsch30, das größte hier untersuchte Problem, liegen die durchschnittlich besten Kosten für den STC-GA 30% unter denen des Strafterm-GAs. Betrachtet man den Aufwand, der zur Erreichung der Ergebnisse verursacht wird, so zeigt Tabel-

Tabelle 5.6: Ergebnisse für die Netzwerke deutsch15, deutsch20, deutsch25 und deutsch30 mit STC-GA und Strafterm-GA ($r(e_i) = 0,95$ und $R_0 = 0,95$)

		STC-GA					Strafterm-GA				
$\|V\|$	$C_{BestEver}$	C_{Beste} μ	σ	C_{Beste}	#Evals	t_{conv}	C_{Beste} μ	σ	C_{Beste}	#Evals	t_{conv}
15	2.367	2.452,7	60,3	2.367	10.435	2.033	4.336,4	375,2	3.681	2.817	218
20	2.861	2.978,8	118,2	2.861	16.739	1.587	5.577,0	638,4	4.738	4.135	202
25	3.424	3.789,4	486,9	3.424	20.380	37.380	7.281,3	840,6	5.539	47.201	10.320
30	6.063	7.551,3	1.385,2	6.063	17.558	49.020	10.538,2	981,2	9.508	5.712	16.800

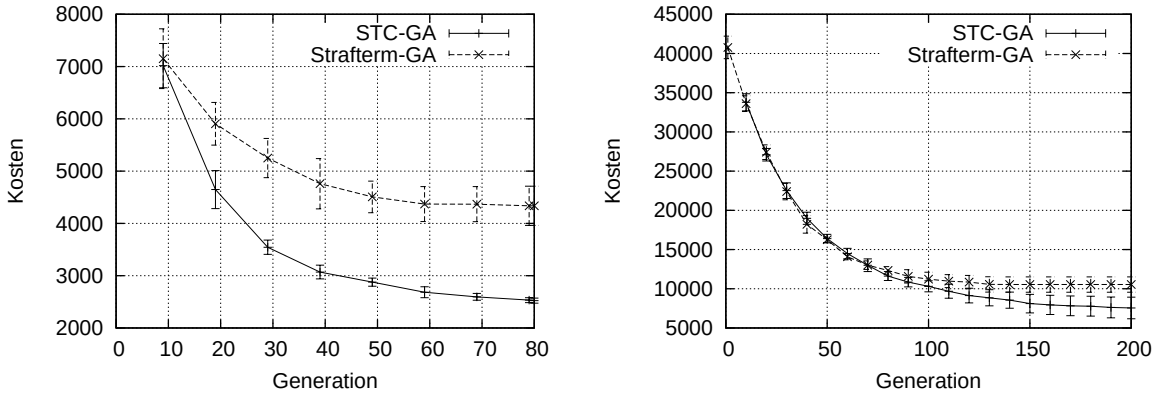
le 5.6, dass der Strafterm-GA weniger Fitnessbewertungen (#Evals) durchführte und eine deutlich geringere Laufzeit (t_{conv}) aufweist. Wie bereits zuvor für die in den Tabellen 5.3, 5.4 und 5.5 gezeigten Ergebnisse dargestellt wurde, wird die längere Laufzeit und die höhere Anzahl an Fitnessbewertungen durch den Einsatz der STC-Reparaturheuristik verursacht.
Zusammenfassend lässt sich für die besten Kosten und die durchschnittlich besten Kosten der untersuchten Testprobleme feststellen, dass der STC-GA unabhängig von der Problemgröße stets die beste Lösung findet und damit einen Strafterm-GA in der Qualität der Lösungen überlegen ist. Da für das in diesem Abschnitt betrachtete Planungsproblem keine Beschränkungen bezüglich der zum Finden einer guten Lösung zur Verfügung stehenden Zeit bestehen, ist der höhere Berechnungsaufwand und damit verbunden die höhere Laufzeit des STC-GA in Anbetracht der Lösungsqualität als akzeptabel anzusehen.
Neben der Betrachtung des durch den STC-GA verursachten Berechnungsaufwandes anhand der Fitnesswerte soll zusätzlich auch eine Betrachtung des durch den Algorithmus verursachten Reparaturaufwands stattfinden. Abbildung 5.3 stellt den Reparaturaufwand exemplarisch für ein 6-Knoten- und ein 10-Knoten-Testproblem bei unterschiedlichen Kantenzuverlässigkeiten ($r(e_i) = 0,9$ und $r(e_i) = 0,95$) und verschiedenen All-Terminal-Zuverlässigkeitsanforderungen ($R_0 = 0,9$ und $R_0 = 0,95$) gegenüber. Für jede Parameterkombination $(r(e_i), R_0)$ wird jeweils die Anzahl der Reparaturen je Generation gemittelt über zehn Läufe gezeigt. Wie aus den dargestellten Verläufen deutlich wird, nimmt die Anzahl der durchgeführten Reparaturen mit steigender Generationszahl ab. Dies bedeutet, dass mit steigender Generationszahl die Anzahl der ungültigen Lösungen in der Population abnimmt. Vergleicht man den Verlauf der Reparaturschritte für unterschiedliche Parameterkombinationen, so lässt sich feststellen, dass für beide in Abbildung 5.3 betrachteten Testprobleme die Parameter $r(e_i)$ und R_0 keinen maßgeblichen Einfluss auf die durchgeführten Reparaturen haben. Daraus folgt für die STC-Reparaturheuristik, dass diese unabhängig von der geforderten Mindestzuverlässigkeit und den gegebenen Verbindungszuverlässigkeiten in der Lage ist, invalide Lösungen in der Population so zu reparieren, dass die Anzahl der unzulässigen Lösungen in der Population abnimmt.
Abschließend vergleicht Abbildung 5.4 den Fitnessverlauf der besten Lösungen je Generation für den STC-GA und den Strafterm-GA anhand der Testprobleme deutsch15 und deutsch30 für $r(e_i) = 0,95$ und $R_0 = 0,95$. Die Werte entsprechen dabei dem besten



(a) 6-Knoten-Testproblem

(b) 10-Knoten-Testproblem

Abbildung 5.3: Vergleich der mittleren Anzahl an Reparaturen für STC-GA bei unterschiedlichen $r(e_i)$ und R_0



(a) deutsch15

(b) deutsch30

Abbildung 5.4: Performancevergleich von STC-GA und Strafterm-GA für die Testprobleme deutsch15 und deutsch30

Fitnesswert je Generation gemittelt über alle zehn Läufe. Der dargestellte Fitnessverlauf zeigt, dass der Strafterm-GA schneller als der STC-GA konvergiert. Die Fitnesswerte für den STC-GA sinken langsamer als die des Strafterm-GAs ab und konvergieren am Ende in einer besseren Lösung.
Zusammenfassend lässt sich für die hier untersuchten Testprobleme feststellen, dass die vorgestellte STC-Reparaturheuristik in der Lage ist, beim Einsatz in einem genetischen Algorithmus unabhängig von der geforderten Zuverlässigkeit und den gegebenen Verbindungszuverlässigkeiten ungültige Lösungen so zu reparieren, dass ein GA unter Verwendung dieser Reparaturheuristik in der Lage ist, optimale Lösungen zu finden.

5.2 Kommunikationsnetzwerkplanung mit Zuverlässigkeitsrestriktionen unter Verwendung von Technologieoptionen

Mit Abschnitt 5.1 wurden Planungsprobleme untersucht, in denen sämtliche Verbindungen des Kommunikationsnetzwerkes die gleiche Verbindungszuverlässigkeit besitzen und der Planer keine Möglichkeit zur Auswahl zwischen unterschiedlichen Technologieoptionen besitzt. Im Folgenden werden Kommunikationsnetzwerkplanungsprobleme unter Zuverlässigkeitsrestriktionen untersucht, bei denen die Verbindungen des Netzwerkes unterschiedliche Zuverlässigkeiten besitzen können und der Netzwerkplaner zwischen unterschiedlichen Technologieoptionen wählen kann.
Mit Abschnitt 5.1.1 wurde gezeigt, dass der Einsatz eines Strafterms zur Berücksichtigung einer Zuverlässigkeitsnebenbedingung im GA-Design zu unzulässigen Lösungen führen kann. Die Ergebnisse in Abschnitt 5.1.3.2 zeigen, dass ein Strafterm-GA einem reparaturbasierten Ansatz in der Qualität der generierten Lösungen unterlegen ist. Zur Berücksichtigung der Zuverlässigkeitsnebenbedingung bei der Planung unter Verwendung von Technologieoptionen im Design eines GAs wird deshalb mit diesem Abschnitt ein direkter Ansatz gewählt, der ungültige Lösungen mit Hilfe einer Reparaturheuristik in gültige Lösungen überführt.
Abschnitt 5.2.1 führt die Reparaturheuristik CURE (CUt based REpair heuristic) ein, welche bei der Kommunikationsnetzwerkplanung mit Zuverlässigkeitsrestriktionen unter Verwendung unterschiedlicher Technologieoptionen Lösungen, welche eine aufgestellte Zuverlässigkeitsnebenbedingung R_0 nicht erfüllen, in gültige Lösungen überführt. Für die Integration der Reparaturheuristik CURE in einen GA werden mit Abschnitt 5.2.2 zwei unterschiedliche Konzepte untersucht und die GAs LaBORNet und BaBORNet entworfen. In einer experimentellen Untersuchung werden beide GAs für aus der Literatur entnommene und mit der Arbeit neu eingeführte Testprobleme untersucht und die Ergebnisse denen eines aus der Literatur entnommenen Strafterm-GAs gegenübergestellt.

5.2.1 CURE – Eine Min-Cut Reparaturheuristik für die Kommunikationsnetzwerkplanung

Die in Abschnitt 5.1.2 vorgestellte Reparaturheuristik berücksichtigt während des Reparaturprozesses lediglich die Anzahl der Spannbäume in einem Graphen. Werden durch den Eingabegraphen Kanten mit unterschiedlichen Technologieoptionen zur Verfügung gestellt, so kann dies mit der STC-Heuristik nicht berücksichtigt werden. Mit der Reparaturheuristik CURE wird ein Verfahren vorgestellt, welches in der Lage ist, beim Entwurf ökonomischer und zuverlässiger Netzwerktopologien unter Verwendung von unterschiedlichen Technologieoptionen Kosten und Zuverlässigkeit gleichermaßen zu berücksichtigen. Die Reparaturheuristik CURE verwendet Netzwerkstrukturen, welche durch ein heuristisches Optimierungsverfahren erzeugt wurden und verbessert diese solange, bis die All-Terminal-Zuverlässigkeit $R_{All}(G_N)$ eines Graphen größer als eine minimal geforderte Schranke R_0 ist. CURE geht dabei prinzipiell so vor, dass in einem ersten Schritt versucht wird, die Zuverlässigkeit von einzelnen Kanten zu erhöhen. Die Zuverlässigkeit der einzelnen Kanten wird dadurch erhöht, indem für eine Kante e_{v_i,v_j} eine Technologieoption l_k mit einer höheren Zuverlässigkeit $r(l_k(e_{v_i,v_j}))$ gewählt wird. Falls für jede mögliche Kante in G_N die maximal mögliche Zuverlässigkeit $r(l_k(e_{v_i,v_j}))$ gewählt ist und das geforderte R_0 noch nicht erreicht ist, werden in einem zweiten Schritt zusätzliche Verbindungen mit Hilfe der STC-Reparaturheuristik hinzugefügt.
Durch diese Vorgehensweise stellt CURE sicher, dass stets eine Netzwerktopologie erzeugt wird, welche die Zuverlässigkeitsnebenbedingung $R_{All}(G_N) \geq R_0$ erfüllt. Der Einsatz einer solchen Prozedur in einem heuristischen Optimierungsverfahren wie z. B. einem GA ermöglicht es, ungültige Lösungen, welche im Laufe des Optimierungsprozesses entstehen, hinsichtlich der gestellten Zuverlässigkeitsnebenbedingung zu reparieren. Für die Auswahl der Kanten, deren Zuverlässigkeit im ersten Schritt von CURE vergrößert wird, wird die Theorie der minimalen Schnitte in Graphen verwendet.

Definition 5.1 (Minimaler Schnitt)
Ein Schnitt $V_C \subset V$ in einem beliebigen Graphen G ist eine nichtleere Teilmenge der Knoten V. Jeder Menge V_C an Knoten wird die Menge E_C an Kanten e_{v_i,v_j} zugeordnet, für die gilt $\forall e_{v_i,v_j} \in E_C : v_i \in V_C$ und $v_j \notin V_C$. Löscht man nun alle Kanten E_C aus G, so zerfällt G in zwei Subgraphen, welche jeweils aus den Knotenmengen V_C und $V \setminus V_C$ bestehen. Das Gewicht eines Schnittes V_C ist die Summe der Gewichte der Kanten E_C. Ein minimaler Schnitt ist der Schnitt mit dem geringsten Gewicht.

Beim Entwurf zuverlässiger Netzwerktopologien kann das Konzept der minimalen Schnitte zum Finden der Verbindungen in einem Netzwerk genutzt werden, deren Ausfall das Netzwerk trennen würde. Durch die CURE-Heuristik wird jede Kante des Graphen G_N mit den Kosten $c(l_{k+1}(e_{v_i,v_j}))$ der nächstzuverlässigeren Technologieoption $l_{k+1}(e_{v_i,v_j})$ bewertet und für G_N ein minimaler Schnitt bestimmt.
Durch die Wahl von Kanten, die zu einem Schnitt gehören, wird sichergestellt, dass die Zuverlässigkeit der Kanten des Graphen verbessert wird, durch deren gleichzeitigen Ausfall das Netzwerk nicht mehr verbunden wäre. Durch die Verwendung des minimalen Schnittes wird genau die Kantenmenge gefunden, bei deren Verbesserung der geringste

Kostenzuwachs für die Netzwerktopologie entsteht. Den Ablauf von CURE stellt Algorithmus 5.3 dar.

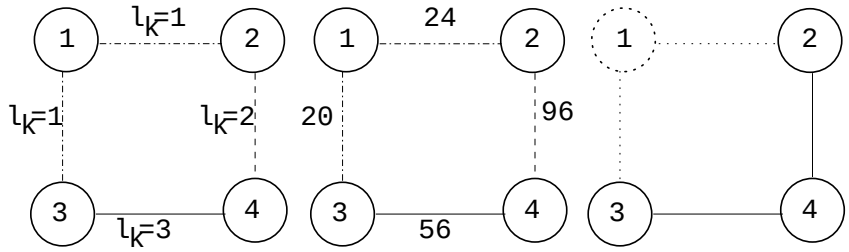
Algorithmus 5.3 CURE-Reparaturheuristik

Eingabe: G_N, G, R_0
Ausgabe: G_N

Queue $Q = \emptyset$, Q.append(G_N)
while (!Q.empty) & ($R_{All}(G_N) < R_0$)) **do**
G_{work} = Q.first()
Setzte das Gewicht (Kosten) für $e_{v_i,v_j} \in G_{work}$:
$c(l_k(e_{v_i,v_j})) = \begin{cases} c(l_{k+1}(e_{v_i,v_j})) & \text{if} \quad k < k_{max} \\ c(l_k(e_{v_i,v_j})) & \text{if} \quad k = k_{max} \end{cases}$
V_C = MinCut(G_{work}) (unter Verwendung der Gewichte $c(l_k(e_{v_i,v_j}))$
Erhöhe Zuverlässigkeit $\forall e_{v_i,v_j} \in E_C : l_k(e_{v_i,v_j}) = \begin{cases} l_{k+1}(e_{v_i,v_j}) & \text{if} \quad k < k_{max} \\ l_k(e_{v_i,v_j}) & \text{if} \quad k = k_{max} \end{cases}$
Berechne $R_{All}(G_N)$
$G_{N_1} = G_{work} \setminus \{V_C\}, G_{N_2} = V_C$
if Anzahl der Knoten in $(G_{N_1}) > 1$ **then**
Q.append(G_{N_1})
end if
if Anzahl der Knoten in $(G_{N_2}) > 1$ **then**
Q.append(G_{N_2})
end if
Q.remove(G_{work})
if (Q.empty) & ($\exists e_{v_i,v_j} \in E_N$ mit $l_k(e_{v_i,v_j} k < k_{max}$) **then**
Q.append(G_N)
end if
end while
if $R_{All}(G_N) < R_0$ **then**
Füge $e_{v_i,v_j} \in E \setminus E_N$ zu G_N hinzu, $\forall e_{v_i,v_j} \in G_N : l_k(e_{v_i,v_j}) = l_1(e_{v_i,v_j})$
Rufe CURE erneut auf
end if

Im ersten Schritt werden sämtliche Kanten aus G_N mit den Kosten $c(l_{k+1}(e_{v_i,v_j}))$ der nächstzuverlässigeren Technologieoption bewertet. k gibt hierbei die Nummer der aktuellen Technologieoption an. Ist für eine der Kanten bereits die Technologieoption mit der höchsten Zuverlässigkeit gewählt ($k = k_{max}$), so wird die Kante mit den Kosten der aktuellen Technologieoption bewertet.
Der zu verbessernde Graph wird anschließend in einer Queue[40] gespeichert. Während des Verbesserungsprozesses wählt die Prozedur jeweils den ersten Graphen G_{work} der Queue und ermittelt für diesen die Kanten des minimalen Schnittes (MinCut). Zum Finden des minimalen Schnittes V_C wird das Verfahren aus [182] genutzt. Anschließend

[40] Der Datencontainer Queue arbeitet nach dem First-In- First-Out-Prinzip. Das Element, das als erstes in der Queue gespeichert wird, wird auch als erstes Element zurückgegeben.



(a) G_N mit den Technologieoptionen l_k (b) G_N mit Gewichten $c(l_k(e_{v_i,v_j}))$ (c) Subgraph G_{N_1}

Abbildung 5.5: Ablauf der CURE-Reparaturheuristik

Tabelle 5.7: CURE Beispiel: Zuverlässigkeit $r(l_k(e_{v_i,v_j}))$ und Kosten $c(l_k(e_{v_i,v_j}))$ für die Technologieoptionen l_k

	$l_1(e_{v_i,v_j})$		$l_2(e_{v_i,v_j})$		$l_3(e_{v_i,v_j})$	
	r	c	r	c	r	c
$e_{1,2}$	0,8	10	0,9	20	0,95	40
$e_{2,4}$	0,8	14	0,9	28	0,95	56
$e_{3,4}$	0,8	24	0,9	48	0,95	96
$e_{1,3}$	0,8	12	0,9	24	0,95	48

wird für alle Kanten des minimalen Schnittes die Zuverlässigkeit auf die nächste Technologieoption erhöht und die Kanten E_C aus G_{work} entfernt, so dass zwei Subgraphen G_{N_1} und G_{N_2} entstehen. Jeder Subgraph mit $|V| > 1$ wird am Ende der Queue angefügt. Wird durch die erste Verbesserung die geforderte Zuverlässigkeit R_0 nicht erreicht, so werden auf diese Weise rekursiv die neu entstandenen Subgraphen G_{N_1}, G_{N_2} ebenfalls mittels CURE verbessert. Wenn die Queue leer ist und noch nicht für sämtliche Kanten die höchste Technologieoption ($k = k_{max}$) gewählt wurde, so wird der komplette Graph erneut in die Queue eingestellt und CURE gestartet. Reichen die in G_N enthaltenen Kanten mit ihrer maximal möglichen Zuverlässigkeit nicht aus, um die geforderte Zuverlässigkeit R_0 zu erfüllen, so wird eine neue Kante mittels STC-Reparaturheuristik aus G in G_N eingefügt und CURE erneut gestartet.
Der Ablauf von CURE wird am Beispiel des Graphen G_N aus Abbildung 5.5(a) erläutert. Abbildung 5.5(b) zeigt die Bewertung des Graphen aus Abbildung 5.5(a) entsprechend der gewählten Technologieoptionen. Die Kosten und Zuverlässigkeiten der einzelnen Kanten zeigt Tabelle 5.7.
In Abbildung 5.5(c) werden der Schnitt V_C sowie die Kanten E_C gepunktet dargestellt. Als minimaler Schnitt wird $V_C = \{v_1\}$ mit $E_C = \{e_{v_1,v_2}, e_{v_1,v_3}\}$ mit einem Gewicht von 44 ermittelt. Für die Kanten e_{v_1,v_2} und e_{v_1,v_3} werden die nächstbesseren Technologieoptionen ($l_2(e_{v_1,v_2})$ und $l_2(e_{v_1,v_3})$) ausgewählt und die Kanten anschließend aus G_N gelöscht. Nach dem Löschen entstehen die Subgraphen G_{N_1} mit den Knoten $\{v_2, v_3, v_4\}$ und G_{N_2} mit dem Knoten $\{v_1\}$. Da G_{N_2} nur einen Knoten besitzt, wird der Subgraph

durch CURE nicht weiter betrachtet. Konnte durch die Verbesserung der Kanten e_{v_1,v_2} und e_{v_1,v_3} die geforderte Zuverlässigkeit R_0 nicht erreicht werden, so wird das Verfahren rekursiv auf G_{N_1} angewendet.

5.2.2 Zwei evolutionäre Planungsverfahren für die Kommunikationsnetzwerkplanung unter Verwendung der CURE-Reparaturheuristik

In Abschnitt 5.2.1 wurde eine Reparaturheuristik vorgestellt, die es ermöglicht, invalide Lösungen unter Berücksichtigung einer minimalen Zuverlässigkeitsanforderung zu reparieren. Im Gegensatz zu STC ist CURE in der Lage, unterschiedliche Technologieoptionen, die sich hinsichtlich Kosten und Zuverlässigkeit unterscheiden, während der Reparatur zu berücksichtigen. Stehen einem Netzwerkplaner für den Entwurf einer Netzwerktopologie verschiedene Technologieoptionen, die sich hinsichtlich der zuvor genannten Kriterien unterscheiden, zur Verfügung, so muss das in Abschnitt 5.1.3 eingeführte Problem und damit die Zielfunktion für das Optimierungsproblem erweitert werden. Unter Berücksichtigung einer Zuverlässigkeitsnebenbedingung und der Möglichkeit, bei der Installation einer Verbindung zwischen zwei Knoten zwischen verschiedenen Technologieoptionen zu wählen, lautet die Zielfunktion:

$$f(y) = C(G_N) = \sum_{e_{v_i,v_j} \in E_N} c(l_k(e_{v_i,v_j})) \rightarrow \min, \quad \text{mit: } R(G_N) \geq R_0 \text{ (Zuverlässigkeitsnebenbedingung)} \tag{5.8}$$

Dabei gilt:[41] $C(G_N)$ sind die Kosten einer Lösung, die durch den Graphen G_N beschrieben wird und $c(l_k(e_{v_i,v_j}))$ entspricht den Kosten der Verbindung zwischen den Knoten v_i und v_j unter Verwendung der Technologieoption l_k für diese Verbindung.
Mit diesem Abschnitt werden zwei evolutionäre Planungsverfahren eingeführt, die unter Verwendung der Reparaturheuristik aus Abschnitt 5.2.1 die Planung von zuverlässigen und ökonomischen Netzwerken ermöglichen. Mit Abschnitt 3.5.4 wurden unterschiedliche Methoden zur Integration einer Nebenbedingung in ein evolutionäres Verfahren sowie der Umgang mit invaliden Lösungen während der Optimierung vorgestellt.
In Abschnitt 5.1.3.2 wurde gezeigt, dass die Verwendung einer Reparaturheuristik die Gültigkeit sämtlicher Lösungen sicherstellt und ein genetischer Algorithmus mit einer solchen Reparaturheuristik in der Lage ist, optimale Lösungen zu finden. In Anlehnung an die aus der Biologie bekannten Evolutionstheorien werden im Folgenden zwei Verfahren entworfen, die eine Planung von Netzwerktopologien unter Berücksichtigung einer Zuverlässigkeitsnebenbedingung ermöglichen. Die für die biologische Evolution propagierten Evolutionstheorien lassen sich einfach auf bestehende evolutionäre Planungsverfahren wie z. B. genetische Algorithmen übertragen. Im Folgenden werden die Evolutionstheorien von Lamarck und Baldwin genutzt, um zwei Planungsverfahren für das in dieser Arbeit betrachtete Planungsproblem zu entwerfen. Beide Evolutionstheorien unterscheiden sich in der Art und Weise, wie sich Änderungen und Beeinflussungen des

[41] Vgl. Abschnitt 4.2.

Individuums, welche es während seines Lebens erfährt, auf seine Fortpflanzungswahrscheinlichkeit auswirken und wie das durch das Individuum erworbene Wissen an seine Nachkommen weitergegeben wird.
Lamarck formuliert in seinem zweiten Gesetz [124, 125]: „Alles, was die Individuen durch den Einfluss der Verhältnisse, denen ihre Rasse lange Zeit hindurch ausgesetzt ist und folglich durch den Einfluss des vorherrschenden Gebrauchs oder konstanten Nichtgebrauchs eines Organs erwerben oder verlieren, wird durch die Fortpflanzung auf die Nachkommen vererbt, vorausgesetzt, dass die erworbenen Veränderungen beiden Geschlechtern oder den Erzeugern dieser Individuen gemein sind."
Gemäß der Lamarck'schen Evolutionstheorie vererbt somit jedes Individuum sein erworbenes Wissen an seine Nachkommen (vgl. [190]). Überträgt man diese Theorie auf einen genetischen Algorithmus mit einer Reparaturfunktion, so wird das (durch Reparatur) erlernte Wissen und somit die Änderungen am Genom bei Rekombination an die Nachkommen weitergegeben. Nach der Lamarck'schen Evolutionstheorie verändert ein Reparaturverfahren (was hier als Lernen verstanden werden soll) die Kodierung des Individuums und passt bei der Reparatur dessen Lösungsgüte (Fitness) an.
Im Gegensatz zur Theorie von Lamarck basiert der von Baldwin propagierte Baldwin-Effekt [11] auf der Annahme, dass Individuen ihr erworbenes Wissen nicht an ihre Nachkommen weitergeben, sondern dieses lediglich Einfluss auf deren Fortpflanzungswahrscheinlichkeit hat. Überträgt man dieses Prinzip auf eine Reparaturheuristik eines genetischen Algorithmus, so bedeutet dies, dass die Reparatur lediglich den Fitnesswert eines Individuums verändert. Im Gegensatz zum Lamarck'schen Ansatz bleibt das Genom (und damit die Kodierung des Problems) vom Reparaturprozess unberührt. Beide Evolutionstheorien werden im Folgenden für das hier untersuchte Netzwerktopologieplanungsproblem adaptiert.

5.2.2.1 LaBORNet - Ein evolutionärer Planungsansatz unter Verwendung unterschiedlicher Technologieoptionen

LaBORNet (Lamarckian Based Optimizer for Reliable Network Design Problems) setzt die Lamarck'sche Evolutionstheorie in einem genetischen Algorithmus um. Generiert der GA ungültige Lösungen, so werden diese an eine Reparaturheuristik übergeben (hier CURE), welche das Genom und dessen Fitness so verändert, dass die Lösung den gestellten Nebenbedingungen genügt. Für LaBORNet wird jede Lösung als Vektor g mit der Länge $|E|$ kodiert. Jedes Element von g repräsentiert dabei eine Kante mit der für die Kante gewählten Technologieoption l_k ($1 \leq k \leq k_{max}$) des Eingabegraphen G. Wenn eine Kante aus G in einer Lösung nicht verwendet wird, so ist $g_i = 0$. Abbildung 5.6 zeigt die Kodierung einer Lösung für LaBORNet. Verbindungen von G, die für die abgebildete Lösung nicht verwendet werden, sind gestrichelt dargestellt.
Wenn durch die Operatoren des genetischen Algorithmus (Initialisierung, Mutation, Rekombination) eine ungültige Lösung generiert wird, so wird der Vektor g durch die Reparaturheuristik verändert. LaBORNet kann dabei als eine Kombination eines genetischen Algorithmus mit einem lokalen Suchverfahren verstanden werden. Für jede ungültige Lösung wird mittels CURE eine lokale (benachbarte) gültige Lösung gesucht. Aufgabe von LaBORNet ist es, für die Eingabedaten sowohl eine geeignete Topologie

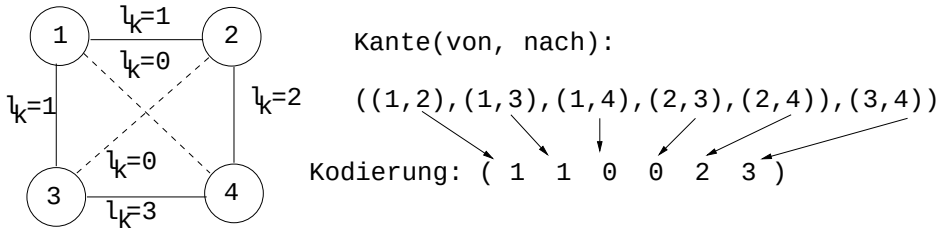


Abbildung 5.6: Kodierung einer Lösung für LaBORNet

zu finden, als auch für die dadurch gewählten Verbindungen die richtige Auswahl für die Technologieoptionen zu treffen, so dass das entworfene Netzwerk die gestellte Zuverlässigkeitsanforderung erfüllt. Somit ergibt sich für LaBORNet ein Suchraum der Größe $(k_{max}+1)^{|E|}$. Für ein Netzwerk mit 30 möglichen Verbindungen und der Auswahl zwischen drei Technologieoptionen pro Verbindung ergeben sich somit $4^{30} \approx 1,15 \cdot 10^{18}$ Lösungen. Ein Teil dieser Lösungen erfüllt die aufgestellte Zuverlässigkeitsnebenbedingung ($R_{All} \geq R_0$) nicht und muss mittels der Reparaturheuristik in eine gültige Lösung überführt werden.

5.2.2.2 BaBORNet - Ein hybrider evolutionärer Planungsansatz unter Verwendung unterschiedlicher Technologieoptionen

Inspiriert durch den mit dem Baldwin-Effekt vorgeschlagenen Evolutionsprozess wurde BaBORNet (Baldwin Based Optimizer for Reliable Network Design Problems) entworfen. Das Verfahren kombiniert ebenfalls einen GA mit einem lokalen Suchverfahren. Im Gegensatz zu LaBORNet wird für BaBORNet jedoch eine Dekomposition des Netzwerkdesignproblems in zwei Unterprobleme vorgenommen. Eine Lösung wird in BaBORNet lediglich über ihre Netzwerktopologie repräsentiert. Der genetische Algorithmus besitzt keine Kenntnisse darüber, welche Technologieoption für eine Verbindung der Topologie zu wählen ist. Jede Lösung wird als Binärvektor g der Länge $|E|$ kodiert. Jedes Allel des Genoms entspricht dabei einer Kante des Eingabegraphen G.
Die Kodierung der Lösung aus Abbildung 5.6 als Genom für BaBORNet zeigt Abbildung 5.7. Für jede Kante aus G, die in der kodierten Lösung verwendet wird, gilt $g_i = 1$.

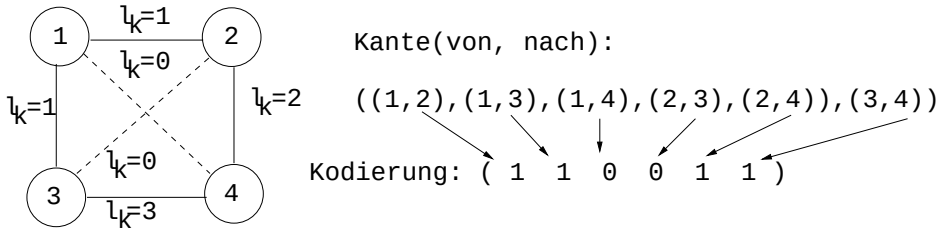


Abbildung 5.7: Kodierung einer Lösung für BaBORNet

Für alle anderen Kanten ist $g_i = 0$. Für die Fitnessbewertung einer BaBORNet-Lösung werden den Verbindung die Technologieoptionen mit den geringsten Kosten zugewiesen. Genügt die daraus resultierende All-Terminal-Zuverlässigkeit nicht der an das Netzwerk gestellten Ausfallsicherheit, so wird die Lösung mittels CURE so verändert, dass diese

gültig wird. Die Reparaturheuristik wählt dabei für die einzelnen Kanten zuverlässigere Technologieoptionen aus und fügt gegebenfalls weitere Verbindungen ein. Nachdem die reparierte Lösung von CURE zurückgeliefert wurde, wird deren Fitness (für die gewählten Technologieoptionen) den invaliden Lösungen zugewiesen. Wurden mittels CURE weitere Verbindungen in das Netzwerk eingefügt, so werden diese Verbindungen ebenfalls in den Lösungsvektor eingefügt. Die Information, welche Technologieoptionen für die einzelnen Verbindungen gewählt wurde, wird jedoch nicht im Lösungsvektor gespeichert.
Im Gegensatz zu LaBORNet wird durch BaBORNet der Suchraum für den GA erheblich reduziert. Aufgabe des genetischen Algorithmus ist es, eine Topologie zu finden, auf der aufbauend sich eine (unter Berücksichtigung der Zuverlässigkeitsanforderung) geeignete Auswahl von Technologieoptionen für die Verbindungen finden lässt. Letztere Aufgabe fällt dabei einer lokalen Suche zu. Für BaBORNet wird in dieser Arbeit die CURE-Reparaturheuristik eingesetzt. Die Anwendung eines Verfahrens wie BaBORNet setzt dabei voraus, dass eine durch den GA generierte Lösung durch die lokale Suche stets identisch verändert wird. Durch die deterministische Arbeitsweise von CURE wird diese Bedingung erfüllt. Das eigentliche Problem wurde dabei in die Teilprobleme der Topologiesuche und der Auswahl der Technologieoptionen zerlegt. Besteht der Suchraum für LaBORNet für ein Netzwerk mit 30 möglichen Verbindungen und der Auswahl zwischen drei Technologieoptionen pro Verbindung aus $4^{30} \approx 1,15 \cdot 10^{18}$ Lösungen, so sind es bei BaBORNet nur noch $2^{30} \approx 1,07 \cdot 10^{9}$ Lösungen. Trotz des kleineren Suchraums für BaBORNet ist beim Einsatz des Verfahrens ein schlechteres Laufzeitverhalten als bei LaBORNet zu erwarten, da viele der durch den GA generierten Lösungen die Zuverlässigkeitsnebenbedingung nicht erfüllen und bei der Fitnessbewertung stets der lokalen Suche (Reparatur) mittels CURE unterzogen werden.

5.2.3 Eine empirische Untersuchung der Planungsansätze LaBORNet und BaBORNet

Die in den letzten beiden Abschnitten vorgestellten evolutionären Verfahren LaBORNet und BaBORNet wurden für sechs unterschiedliche Testprobleme, die teilweise künstliche Netzwerke und teilweise reale Planungsprobleme widerspiegeln, untersucht:
deeter10 (10 Knoten, 45 mögliche Verbindungen): Das Testproblem wurde in [49] eingeführt. Die Knoten wurden zufällig auf einer 100x100 großen Fläche[42] verteilt. Für die Verbindungen des Netzwerkes besteht die Auswahl zwischen den drei Technologieoptionen ($r(l_1) = 0,7$, $r(l_2) = 0,8$ und $r(l_3) = 0,9$ sowie $c(l_1) = 8$, $c(l_2) = 10$ und $c(l_3) = 14$). Die Kosten für eine Verbindung ergeben sich aus dem Produkt der euklidischen Distanz zwischen zwei Knoten multipliziert mit den Kosten $c(l_i)$ der gewählten Technologieoption l_i. Als beste gefundene Lösung für $R_0 = 0,95$ werden in [49] Kosten von 5.661 angegeben.
türkei19 (19 Knoten, 171 mögliche Verbindungen): Dieses Problem stellt eine vereinfachte Version eines realen Planungsproblems in der Türkei dar. Das Problem wird in [49] vorgestellt. Zielsetzung ist es, 19 Universitäten und Forschungseinrichtungen in

[42] Die Entfernungsmatrix für das Problem ist in Anhang A.2 zu finden.

neun Städten in der Türkei miteinander zu vernetzen. Die entworfene Netzwerktopologie muss dabei eine All-Terminal-Zuverlässigkeit von $R_0 = 0,99$ besitzen. Für die Verbindungen des Netzwerkes besteht die Auswahl zwischen den drei Technologieoptionen ($r(l_1) = 0,960$, $r(l_2) = 0,975$ und $r(l_3) = 0,990$ sowie $c(l_1) = 333$, $c(l_2) = 433$ und $c(l_3) = 583$). Die Kosten für eine Verbindung ergeben sich aus dem Produkt der Entfernung[43] zwischen zwei Knoten multipliziert mit den Kosten $c(l_i)$ der gewählten Technologieoption l_i. Als Kosten für die beste gefundene Lösung wird in [49] 7.694.708 angegeben. Unter Verwendung eines multikriteriellen genetischen Algorithmus konnte dieses Ergebnis in [14] auf 1.755.474 verbessert werden.
deutsch15, deutsch20, deutsch25, deutsch30 (15, 20, 25, 30 Knoten ; 105, 190, 300, 435 mögliche Verbindungen): Jedes Testproblem repräsentiert mit seinen Knoten die größten Städte Deutschlands[44]. Für die mit Abschnitt 5.1.3.2 auf S. 97 eingeführten Probleme wurden die Verbindungen des Netzwerkes um Technologieoptionen erweitert. Für eine Verbindung besteht die Auswahl zwischen den drei Technologieoptionen (mit $r(l_1) = 0,7$, $r(l_2) = 0,8$, $r(l_3) = 0,9$ und $c(l_1) = 8$, $c(l_2) = 10$, $c(l_3) = 14$). Wie für die Testprobleme deeter10 und türkei19 ergeben sich die Verbindungskosten aus der Entfernung zwischen zwei Städten multipliziert mit den Kosten für die gewählte Technologieoption. In dieser Arbeit werden die Probleme für $R_0 = 0,95$ untersucht.

5.2.3.1 Experimentelles Design

Für die beschriebenen Testprobleme wurden die Performance der Verfahren LaBORNet und BaBORNet sowie die eines Strafterm-GAs aus [49] untersucht. Die Experimente wurden mit einem Steady-State-GA mit 50 % überlappenden Populationen[45] unter Verwendung eines Uniform-Crossover (siehe S. 47) und einer Flip-Mutation (siehe S. 48) durchgeführt. Die Initialisierung der ersten Population erfolgt mit zufällig generierten Lösungen. Dabei betrug die Wahrscheinlichkeit dafür, dass eine Verbindung zwischen zwei Knoten erstellt wird, $p_{init} = 0,4$. Für LaBORNet und für den Strafterm-GA wurde die initiale Technologieoption für eine erstellte Verbindung gleichverteilt mit $p_{init}^{opt} = 1/k_{max}$ gewählt. Die Crossover-Wahrscheinlichkeit beträgt $p_{cross} = 0,9$, die Mutationswahrscheinlichkeit beträgt $p_{mut} = 0,01$. Für LaBORNet und BaBORNet wurden Experimente mit den Populationsgrößen $pop = 50, 100, 200, 400$ durchgeführt. Für den Strafterm-GA betrug die Populationsgröße $pop = 200$. In einem GA-Lauf wurden maximal 200 Generationen durchlaufen. Wenn über 20 Generationen hinweg keine Verbesserung der besten Lösungen erfolgte, so wurde der Lauf vorzeitig abgebrochen. Pro Populationsgröße und Testproblem wurden jeweils zehn unabhängige Testläufe durchgeführt.
Für die All-Terminal-Zuverlässigkeitsberechnung kam für das Problem deeter10 das exakte Berechnungsverfahren aus [32] zum Einsatz. Für alle anderen Testprobleme wur-

[43] Die Entfernungsmatrix für das Problem türkei19 ist in Anhang A.1 angegeben.
[44] Eine Liste der verwendeten Städte sowie der Entfernungsmatrix für die Probleme wird in Anhang A.3 gegeben.
[45] Eine Begründung zur Wahl des Steady-State-GA sowie die Bedeutung der Überlappung gibt Abschnitt 5.1.3.1.

Tabelle 5.8: Parameter für LaBORNet und BaBORNet

GA	Steady-State-GA
Crossover-Operator	Uniform-Crossover, $p_{cross} = 0,9$
Mutations-Operator	Flip-Mutation, $p_{mut} = 0,01$
Initialisierung von P_0	zufallsbasiert, $p_{init} = 0,4$ $p_{init}^{opt} = 1/lk_{max}$
Bestimmung von R_{All}	Für $\|V\| > 10$ Berechnung der oberen Schranke sowie einfache Monte-Carlo-Simulation sonst lokaler Dekompositionsansatz
p_{mut}	0,01
p_{cross}	0,9
pop	50, 100, 200, 400
gen	200
Anzahl GA-Läufe	10

den Monte-Carlo-Simulationstechniken [62, 134] verwendet. Zu Beginn eines GA-Laufes wurden durch die Monte-Carlo-Simulation jeweils $M = 30.000$ Stichproben generiert. Während des GA-Laufes wurde die Anzahl der Stichproben alle fünf Generationen um 5000 erhöht. Auf diese Weise wurde die Genauigkeit der Zuverlässigkeitsschätzung mit steigender Generationszahl erreicht. Zur Beschleunigung der Zuverlässigkeitsberechnung erfolgt vorgelagert eine Schätzung von R_{All} mit Hilfe der oberen Schranke (vgl. Abschnitt 4.4.2.1). Nur Netzwerke mit $R_{obereSchranke} \geq R_0$ werden an die Monte-Carlo-Simulationsroutine übergeben. Alle drei Verfahren wurden in der GALib [188] unter Linux implementiert und getestet. Die Experimente wurden auf einem P4-2-GHz System durchgeführt. Tabelle 5.8 fasst die Parameter[46] für die durchgeführten Experimente zusammen.

5.2.3.2 Experimentelle Ergebnisse und Auswertung

Die Tabellen 5.9 (auf S. 112) und 5.10 (auf S. 113) fassen die Ergebnisse für die Testprobleme zusammen. Für jedes untersuchte Testproblem geben die Tabellen die verwendete Populationsgröße *pop*, den eingesetzten GA (Verfahren), die Kosten der besten Lösung C_{Beste}, den Mittelwert (μ) der Kosten der besten Lösung (C_{Beste}) für alle zehn Läufe sowie die zugehörige Standardabweichung (σ), die über alle zehn Läufe gemittelte Laufzeit (in Sekunden) für einen GA-Durchlauf (t_{conv}) sowie die mittlere Anzahl (μ) der pro Lauf durchgeführten Fitnessbewertungen ($\#Eval$) und die zugehörige Standardabweichung (σ) an. Da der in [49] vorgeschlagene Strafterm-GA lediglich als Vergleichsmethode zu den in dieser Arbeit entworfenen evolutionären Verfahren dient, wurden die Experimente lediglich mit einer Populationsgröße $pop = 200$ durchgeführt. Für jedes Testproblem sind die Kosten der besten gefundenen Lösung jeweils fett hervorgehoben. Vergleicht man die für die Testprobleme deeter10 und türkei19 erzielten Ergebnisse mit den bisher besten gefundenen Lösungen, so ist für beide Probleme eine deutliche Verbesserung zu

[46] Die Auswahl der hier vorgestellten GA-Operatoren und -Parameter wurden in einer experimentellen Studie als die beste Parameterkombination ermittelt.

Tabelle 5.9: Ergebnisse für LaBORNet und BaBORNet für die Testprobleme deutsch15, deutsch20, deutsch25 und deutsch30 bei unterschiedlichen Populationsgrößen

Testproblem	*pop*	Verfahren	C_{Beste}		C_{Beste}	t_{conv}	#Eval	
			μ	σ			μ	σ
deutsch15	50	LaBORNet	46.327	3.151	44.048	3.048	3.525	952
		BaBORNet	44.856	913	43.866	3.186	2.568	1.164
	100	LaBORNet	49.284	2.132	46.326	3.390	5.480	2.161
		BaBORNet	45.832	463	45.004	23.408	7.245	2.036
	200	LaBORNet	47.419	1.823	44.648	10.800	11.130	3.639
		BaBORNet	45.101	695	43.996	45.400	12.510	4.627
	400	LaBORNet	43.248	867	**42.440**	21.838	23.480	970
		BaBORNet	43.730	336	43.150	47.896	21.580	780
	200	Strafterm-GA	55.402	1.898	52.644	6.300	13.240	2.045
deutsch20	50	LaBORNet	64.126	6.789	52.614	2.298	4.785	278
		BaBORNet	52.840	2.435	47.626	7.410	4.555	697
	100	LaBORNet	58.732	4.652	51.064	10.950	8.985	1.487
		BaBORNet	52.553	818	51.064	23.580	7.945	1.722
	200	LaBORNet	57.981	3.059	53.988	26.100	13.370	2.857
		BaBORNet	51.277	660	50.214	73.620	17.100	2.862
	400	LaBORNet	48.868	2.245	**46.014**	42.900	29.180	1525
		BaBORNet	47.683	872	46.754	76.517	30.780	882
	200	Strafterm-GA	74.910	9.983	64.822	8.700	18.270	2.747
deutsch25	50	LaBORNet	104.269	4.909	96.878	8.340	4.755	560
		BaBORNet	66.720	3.066	62.020	9.195	4.800	219
	100	LaBORNet	91.064	5.350	84.620	14.520	10.000	147
		BaBORNet	59.077	1.805	56.046	24.780	8.580	1.271
	200	LaBORNet	76.750	4.767	70.396	45.370	19.530	1.017
		BaBORNet	56.718	685	55.300	78.391	17.480	2.616
	400	LaBORNet	64.275	3.184	59.638	62.208	39.240	209
		BaBORNet	54.569	715	**53.102**	81.947	36.560	352
	200	Strafterm-GA	155.315	50.730	122.980	24.240	19.930	395
deutsch30	50	LaBORNet	204.288	10.467	184.206	12.402	4.778	428
		BaBORNet	108.064	3.383	101.960	15.375	4.880	208
	100	LaBORNet	171.649	14.541	158.664	21.750	9.030	1.412
		BaBORNet	91.177	3602	84.548	24.960	9.585	492
	200	LaBORNet	147.674	6972	138.646	44.040	18.830	2.453
		BaBORNet	83.968	2123	79.662	82.753	19.540	563
	400	LaBORNet	123.678	3811	116.470	78.768	38.820	327
		BaBORNet	89.315	18097	**77.962**	94.757	35.080	1.016
	200	Strafterm-GA	268.033	25019	246.106	38.340	19.730	762

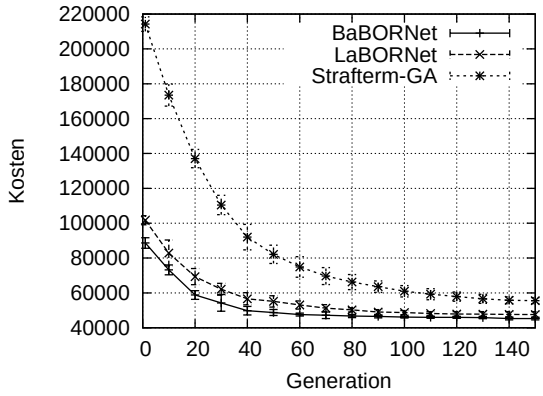
Tabelle 5.10: Ergebnisse für LaBORNet und BaBORNet für die Testprobleme deeter10 und türkei19 bei unterschiedlichen Populationsgrößen

Testproblem	*pop*	Verfahren	C_{Beste}		C_{Beste}	t_{conv}	#Eval	
			μ	σ			μ	σ
deeter10	50	LaBORNet	4.540	93	4.454	402	1.428	548
		BaBORNet	4.648	70	4.559	852	933	365
	100	LaBORNet	4.501	67	4.433	710	2.640	512
		BaBORNet	4.651	35	4.598	1.091	1.155	394
	200	LaBORNet	4.458	17	**4.386**	1.897	4.450	1185
		BaBORNet	4.621	35	4.598	1.508	2.290	525
	400	LaBORNet	4.450	13	4.433	3.153	9.220	256
		BaBORNet	4.561	6	4.559	6.966	7.120	376
	200	Strafterm-GA	5.240	210	4.949	773	6.280	1.248
türkei19	50	LaBORNet	2.477.005	216.457	2.006.650	3.533	4.703	551
		BaBORNet	1.707.455	61.270	1.647.040	9.504	4.225	683
	100	LaBORNet	2.348.899	398.185	1.886.350	6.300	8.250	1.305
		BaBORNet	1.670.062	34.704	1.620.210	23.210	8.315	1.573
	200	LaBORNet	2.160.075	170.579	1.802.870	25.200	15.460	3.095
		BaBORNet	1.650.683	18.254	1.624.960	50.090	13.980	3.473
	400	LaBORNet	2.023.240	179.716	1.689.800	26.033	25.140	1.366
		BaBORNet	1.610.041	13.875	**1.591.110**	51.024	29.040	728
	200	Strafterm-GA	2.898.091	263.881	2.499.080	8.100	18.980	1.603

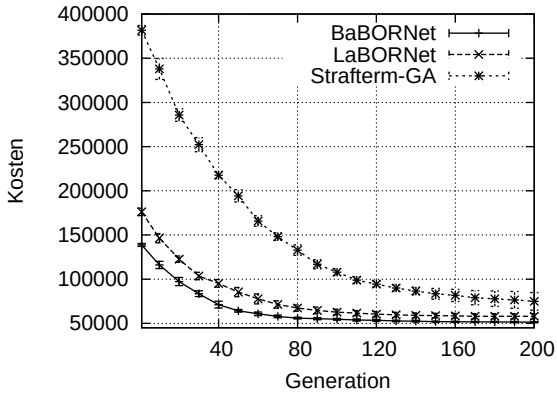
erkennen. Für das Problem deeter10 wurden in [49] die Kosten der besten Lösung mit 5.661 angegeben. Durch den Einsatz von LaBORNet konnte eine Lösung mit den Kosten von 4.386 gefunden werden. Dies entspricht einer Verbesserung um zirka 21%. Für das Problem türkei19 geben Deeter und Smith in [49] als minimale Kosten 7.694.708 an. Die beste hier gefundene Lösung besitzt die Kosten von 1.591.110 und wurde mittels BaBORNet gefunden. Dieses Ergebnis stellt gegenüber [49] eine Verbesserung um zirka 80 % dar. Im Vergleich zu den in [14] veröffentlichten minimalen Kosten von 1.755.474 beträgt die hier erzielte Verbesserung noch zirka 9 %.
Auffällig ist ebenfalls, dass mit der hier gewählten Implementierung des Strafterm-GAs aus [49] für die Testprobleme deeter10 und türkei19 deutlich bessere Ergebnisse als die im Originalaufsatz angegebenen Werte erzielt werden konnten. Ursache hierfür sind zwei Modifikationen, die für das Verfahren vorgenommen wurden. Zum einen betrug bei der Initialisierung die Wahrscheinlichkeit für die Erstellung einer Verbindung zwischen zwei Knoten im Originalaufsatz $p_{init} = 0,75$ und wurde in der vorliegenden Arbeit auf $p_{init} = 0,4$ geändert. Zum anderen wurde für die Experimente ein Steady-State-GA mit überlappenden Populationen eingesetzt. Im Gegensatz zu dem im Originalaufsatz verwendeten einfachen GA führt dies zu einem höheren Selektionsdruck und steigert die Performance (vgl. hierzu Abschnitt 5.1.3.2 S. 97) des GAs.
Für sämtliche untersuchten Testprobleme zeigen die Ergebnisse, dass sowohl LaBORNet als auch BaBORNet einem straftermbasierten GA überlegen sind. Bereits mit einer sehr kleinen Populationsgröße von 50 Individuen finden beide Verfahren Lösungen mit einer höheren Güte als der Strafterm-GA. Abbildung 5.8 auf S. 115 stellt die Performance

der Verfahren LaBORNet, BaBORNet und Strafterm-GA anhand der Kosten der besten Lösung je Generation (gemittelt über alle zehn Läufe) bei einer Populationsgröße von $pop = 200$ gegenüber. Die Abbildung zeigt jeweils den Verlauf der Kosten der besten Lösung je Generation gemittelt über alle zehn GA-Läufe sowie die zugehörige Standardabweichung.
Für sämtliche hier untersuchten Verfahren nehmen (wie zu erwarten) die Kosten der besten Lösung über die Generationen hinweg ab, da immer bessere Lösungen mit geringeren Kosten gefunden werden. Auffällig ist, dass die Kosten der besten Lösung für LaBORNet und BaBORNet bereits in den ersten Generationen deutlich unter denen des Strafterm-GAs liegen. Ursache hierfür ist der Einsatz der CURE-Reparaturheuristik in beiden Verfahren. Wird bei LaBORNet oder BaBORNet eine ungültige Lösung generiert, so wird diese solange verbessert, bis die Zuverlässigkeitsnebenbedingung erfüllt ist. Ungültige Lösungen, die nur knapp die geforderte Zuverlässigkeit verfehlen, müssen lediglich mit geringem Aufwand repariert werden. Solche Lösungen weisen bereits vor der Reparatur sehr geringe Kosten auf. Wenn mittels CURE nur geringfügige Veränderungen an dem Netzwerk vorgenommen werden, so besitzt das Ergebnis der Reparatur letztendlich einen sehr guten Fitnesswert (geringe Kosten).
Abbildung 5.8 zeigt ebenfalls, dass mit steigender Problemgröße (Anzahl möglicher Verbindungen) ebenfalls die Generation, in der die Verfahren mit einer Lösung konvergieren, zunimmt. Ein Vergleich des Konvergenzverhaltens verdeutlicht, dass der Strafterm-GA später als LaBORNet und BaBORNet zu Lösungen mit geringeren Kosten konvergiert. Die Streuung der Kosten der besten Lösung für eine Generation (gemessen anhand der Standardabweichung) zeigt, dass die durch den Strafterm-GA generierten Lösungen für alle Testprobleme deutlich breiter gestreut sind, als die Ergebnisse von LaBORNet und BaBORNet. Es wird dabei deutlich, dass die Standardabweichung mit zunehmender Problemgröße ebenfalls ansteigt. Die Kostenverläufe für LaBORNet und BaBORNet zeigen, dass lediglich für das Problem deeter10 bei LaBORNet die Kosten der besten Lösung unter denen von BaBORNet liegen. Für alle anderen Testprobleme liegen die Kosten der besten Lösung von BaBORNet stets unter denen von LaBORNet.
Mit Abbildung 5.9 erfolgt eine Gegenüberstellung der Ergebnisse für unterschiedliche Populationsgrößen. Für jedes untersuchte Problem zeigt die jeweilige Abbildung für LaBORNet und BaBORNet den Wert der mittleren Kosten[47] C_{AVG} am Ende eines GA-Laufes (gemittelt über alle zehn Läufe) sowie die Kosten der besten Lösung C_{Beste}, die in allen zehn Läufen erzielt wurde. Anhand der in Tabelle 5.9 und Tabelle 5.10 präsentierten Ergebnisse und der hier dargestellten Verläufe wird sichtbar, dass (mit einer Ausnahme) die Lösungen mit den geringsten Kosten bei einer Populationsgröße von $pop = 400$ gefunden wurden. Einzige Ausnahme stellt das kleinste hier betrachtete Problem (deeter10) dar. Hier wird die beste Lösung bei Populationgröße von $pop = 200$ erreicht.
Für den Einfluss der Populationsgröße kann erwartungsgemäß festgestellt werden, dass die Lösungsgüte (gemessen an C_{AVG}) mit zunehmender Populationsgröße steigt. Das bedeutet, je mehr Individuen in der Population enthalten sind, umso besser kann der Problemraum durchsucht werden. Für LaBORNet gilt dabei, dass für größere Proble-

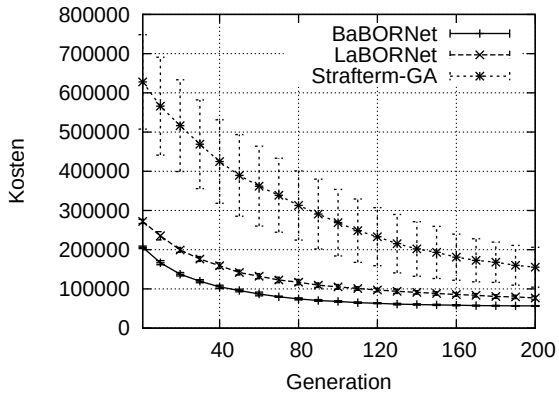
[47] Berechnet als Mittelwert über die Kosten aller Lösungen einer Population.



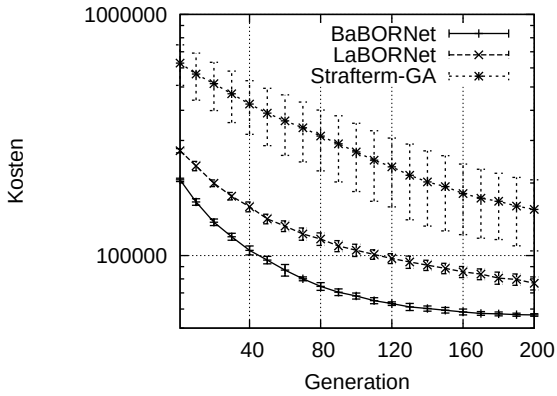
(a) deutsch15



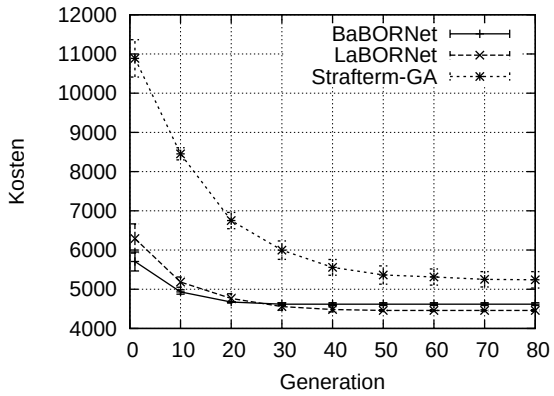
(b) deutsch20



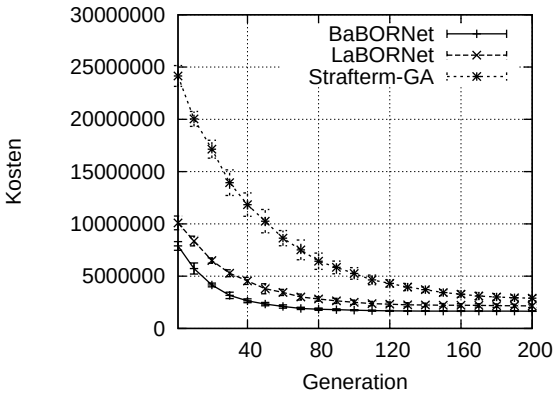
(c) deutsch25



(d) deutsch30



(e) deeter10

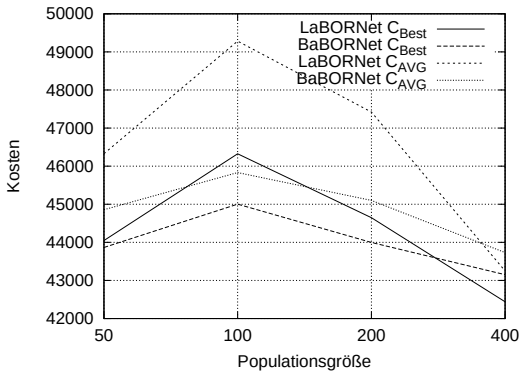


(f) türkei19

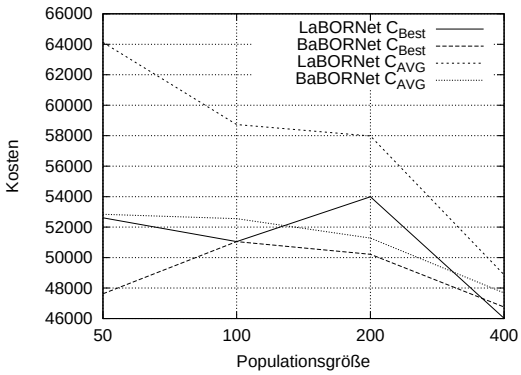
Abbildung 5.8: Performancevergleich anhand der Kosten der besten Lösungen je Generation (gemittelt über alle zehn Läufe) für LaBORNet, BaBORNet und Strafterm-GA bei $pop = 200$

me (deutsch25 und deutsch30) durch eine Verdopplung der Populationsgröße stets eine signifikante Verbesserung in der Lösungsgüte (betrachtet für C_{AVG}) messbar ist (vgl. auch [81]). Im Gegensatz dazu führt die Verdopplung für die gleichen Testprobleme bei BaBORNet lediglich zu einer sehr geringen Verbesserung für C_{AVG}.
Ein eher atypisches Verhalten zeigt der Verlauf der Kurven für das Problem deutsch15. In den durchgeführten Experimenten steigen die Kosten der besten und der durchschnittlichen besten Lösung bei der Verdopplung der Populationsgröße $pop = 50$ auf $pop = 100$ an. Erst ab $pop = 100$ ist der Effekt einer Verbesserung der Lösungsgüte mit zunehmender Populationsgröße erkennbar. Ursache hierfür mag in der „stochastischen Natur" des genetischen Algorithmus liegen. Wenn das Verfahren schon bei der Initialisierung sehr gute Lösungen erzeugt, so kann es bereits bei einer geringen Populationsgröße mit sehr guten Lösungen enden.
Die Abbildungen 5.10 und 5.11 zeigen einen Performancevergleich (anhand der Kosten der besten gefundenen Lösung) von LaBORNet und BaBORNet bei unterschiedlichen Populationsgrößen. Für die Probleme deeter10 und deutsch30 zeigen die Grafiken jeweils den Verlauf der Kosten der besten Lösung für beide Verfahren. Hierfür wurde in jeder Generation jeweils der beste Kostenwert aus allen zehn GA-Läufen für die jeweilige Generation abgetragen.
Die Abbildungen 5.10 und 5.11 veranschaulichen, dass beide Verfahren für kleine Probleme wie deeter10 eine ähnliche Performance zeigen. Bereits in den ersten 20 Generationen können die Kosten der besten Lösung sehr stark abgesenkt werden. Für sämtliche Populationsgrößen konvergieren beide Verfahren nach zirka 50 Generationen. Damit wird deutlich, dass das Problem im Vergleich zu den anderen hier untersuchten Problemen sehr einfach ist und durch einen genetischen Algorithmus mit vergleichweise geringem Aufwand gelöst werden kann.
Ein anderes Ergebnis zeigt die Untersuchung des größten hier betrachteten Problems (deutsch30). Wie aus Abbildung 5.11 deutlich wird, liegen die Kosten für BaBORNet über sämtliche Generationen hinweg unter denen von LaBORNet. Deutlich sichtbar ist ebenfalls der Einfluss der Populationsgröße. Für LaBORNet zeigt die Abbildung 5.11, dass eine größere Anzahl an Individuen in der Population stets dazu beiträgt, dass bessere Lösungen gefunden werden. Für BaBORNet ist dieser Effekt nicht so ausgeprägt. Wie der Verlauf der Kosten über die Generationen zeigt, liegen die Lösungen für $pop = 100$, $pop = 200$ und $pop = 400$ sehr nah beieinander.
Für eine Gegenüberstellung der Verfahren wird abschließend noch der Aufwand, der mit dem Einsatz der Verfahren verbunden ist, betrachtet. Zur Bestimmung der Laufzeit eines GA-Laufes ermittelt das Verfahren jeweils die Uhrzeit[48] zu Beginn und am Ende und zieht beide voneinander ab. Aufgrund der extrem langen Laufzeiten von mitunter mehr als 24 Stunden (86.400 Sekunden) können externe Störungen auf einen GA-Durchlauf, wie sie z. B. durch nächtliche Backups oder die Anmeldung anderer User am System entstehen, die gemessenen Ergebnisse mitunter leicht verfälschen.

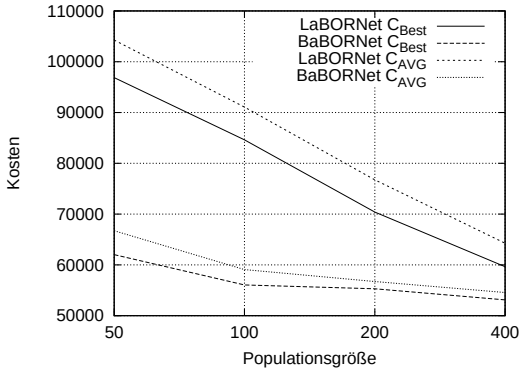
[48] Eine exakte Berechnung der verbrauchten CPU-Zeit mittels der C-Funktion clock() war nicht möglich, da der Wertebereich zur Abspeicherung der Zeit nicht ausreichte.



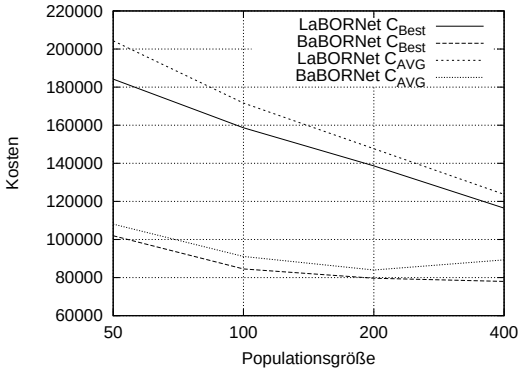
(a) deutsch15



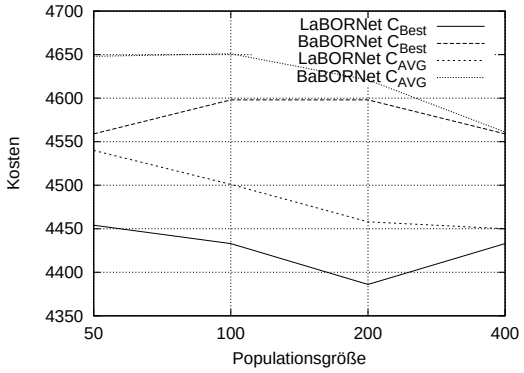
(b) deutsch20



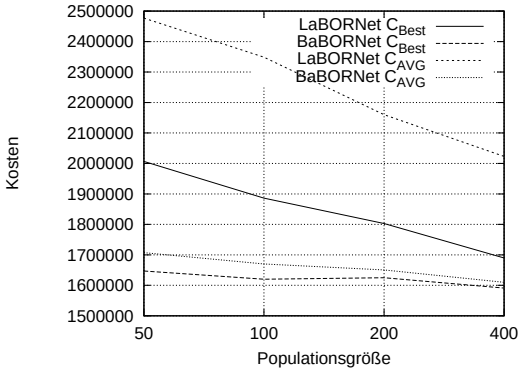
(c) deutsch25



(d) deutsch30

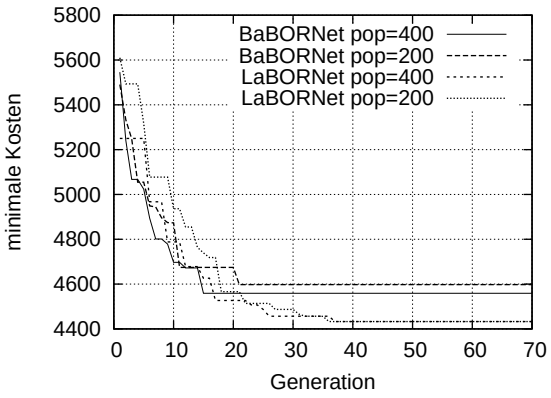


(e) deeter10

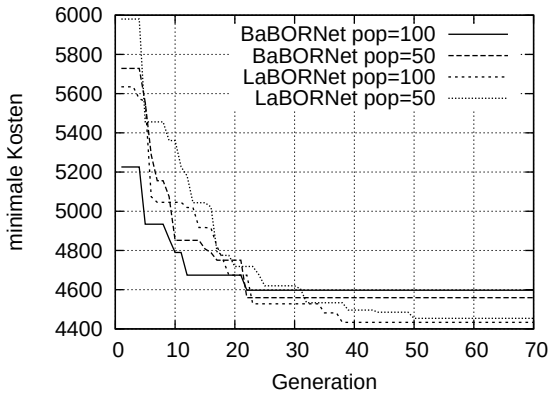


(f) türkei19

Abbildung 5.9: Untersuchung des Einflusses der Populationsgröße auf die Lösungsqualität für LaBORNet und BaBORNet

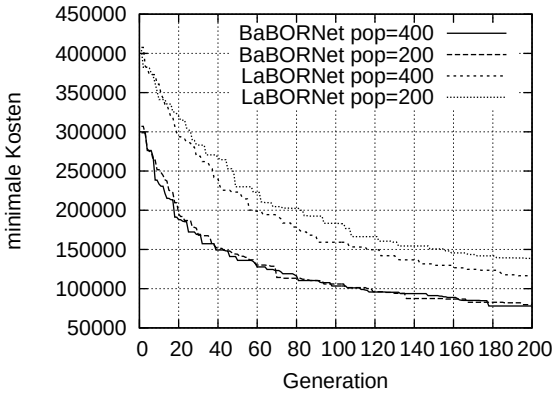


(a) Populationsgröße *pop*=200 und *pop*=400

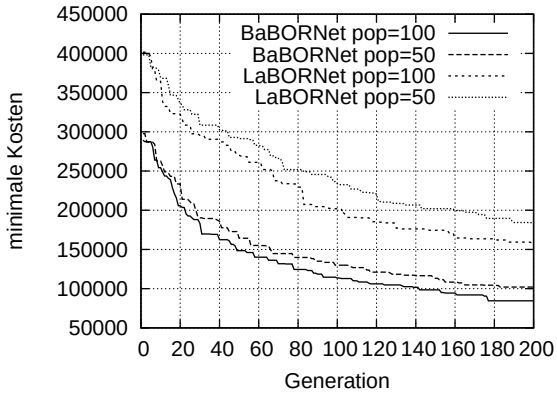


(b) Populationsgröße *pop*=50 und *pop*=100

Abbildung 5.10: Performancevergleich für LaBORNet und BaBORNet bei unterschiedlichen Populationsgrößen für das Testproblem deeter10



(a) Populationsgröße *pop*=200 und *pop*=400



(b) Populationsgröße *pop*=50 und *pop*=100

Abbildung 5.11: Performancevergleich für LaBORNet und BaBORNet bei unterschiedlichen Populationsgrößen für das Testproblem deutsch30

Bei einer Populationsgröße von $pop = 200$ zeigen die Ergebnisse in Tabelle 5.9 und Tabelle 5.10, dass durch LaBORNet und BaBORNet weniger Fitnessbewertungen als durch den Strafterm-GA benötigt werden.
Eine Vergleich der Laufzeiten t_{conv} macht deutlich, dass der Strafterm-GA dabei mehr Lösungen in geringerer Zeit bewerten kann. Dies liegt ursächlich darin begründet, dass der Strafterm-GA für jede Lösung lediglich einmal eine Berechnung der All-Terminal-Zuverlässigkeit durchführen muss. Wird die Zuverlässigkeitsnebenbedingung dabei nicht eingehalten, werden die Lösungen durch den Strafterm abgestraft. Dies ist mit einem sehr geringen Rechenaufwand verbunden. Beim Einsatz von LaBORNet und BaBORNet hingegen werden Lösungen, die mittels CURE repariert werden, mehrmals einer Zuverlässigkeitsüberprüfung unterzogen. Da jede Zuverlässigkeitsberechnung einen hohen Berechnungsaufwand (vgl. Abschnitt 4.5 auf S. 78) verursacht, sind beide Verfahren langsamer.
Wie bereits bei der Vorstellung von BaBORNet vermutet, weist das Verfahren bei gleichen Parametern im Vergleich zu LaBORNet eine deutlich längere Laufzeit auf. Da BaBORNet in jeder Generation eine Vielzahl von Lösungen mittels CURE reparieren muss, entsteht hier ein höherer Berechnungsaufwand, der das Verfahren verlangsamt. So benötigt LaBORNet für das Problem deeter10 bei einer annähernd gleichen Anzahl an Fitnessbewertungen lediglich die halbe Zeit.
Ein Vergleich des Strafterm-GAs mit LaBORNet und BaBORNet für eine vorgegebene Laufzeit zeigt, dass LaBORNet und BaBORNet in der gleichen Zeit mit einer geringeren Populationsgröße Lösungen mit einer höheren Güte (besseren Fitness) generieren. So findet der Strafterm-GA für das Testproblem deutsch25 in 6:44 Stunden (24.240 Sekunden) (gemittelt über alle zehn GA-Durchläufe) die Lösung mit den minimalen Kosten von 122.980. Durch BaBORNet konnte in annähernd gleicher Zeit (24.780 Sekunden) eine Lösung mit den Kosten von 56.046 gefunden werden.
Zusammenfassend lässt sich anhand der empirisch gewonnenen Ergebnisse festhalten, dass die hier entworfenen Verfahren LaBORNet und BaBORNet besser als bekannte Verfahren für aus der Literatur entnommene Testprobleme arbeiten und Lösungen mit geringeren Kosten finden. Die Dekomposition des untersuchten Netzwerkdesignproblems in zwei Teilprobleme, wie sie mit BaBORNet vorgenommen wurde, erlaubt es, Probleminstanzen unterschiedlicher Größe effizient zu lösen. Die Ergebnisse zeigen, dass mittels BaBORNet die besten Ergebnisse erzielt werden, das Verfahren im Vergleich zu anderen Ansätzen jedoch einen deutlich höheren Berechnungsaufwand verursacht. Da es sich bei dem hier betrachteten Problem um keine echtzeitkritische Anwendung handelt, ist der zusätzliche Berechnungsaufwand aufgrund der sehr guten Lösungsqualität vertretbar.

5.3 Zusammenfassung

Das Kapitel zeigte eingangs, dass der Einsatz eines indirekten Verfahrens zur Behandlung einer Zuverlässigkeitsnebenbedingung in Form eines Strafterms mit Problemen verbunden ist. Es konnte nachgewiesen werden, dass ein in der Literatur vorgeschlagener Strafterm-Ansatz für aus der Literatur entnommene Testinstanzen ungültige Lösungen findet. Daher wurde hier die direkte Behandlung der All-Terminal-Zuverlässigkeitsne-

benbedingung mittels einer Reparaturheuristik gewählt. Mit diesem Kapitel wurden die Reparaturheuristiken STC und CURE eingeführt. Beide Heuristiken erlauben es, eine durch ein Optimierungsverfahren generierte Lösung, die einer gestellten Zuverlässigkeitsnebenbedingung nicht genügt, in eine gültige Lösung zu überführen. Während STC lediglich für Probleme mit identischen Zuverlässigkeiten für sämtliche Verbindungen in einem Netzwerk einsetzbar ist, arbeitet CURE mit unterschiedlichen Zuverlässigkeiten und unterstützt die Auswahl von unterschiedlichen Technologieoptionen für eine Netzwerkverbindung. Beide Reparaturheuristiken wurden für den Einsatz in evolutionären Planungsverfahren getestet.
Die empirisch gewonnenen Ergebnisse zeigen, dass ein genetischer Algorithmus mit einer auf STC basierenden Reparaturfunktion für aus der Literatur entnommene Testprobleme optimale Lösungen findet. Für die mit dieser Arbeit neu eingeführten Testprobleme kann der GA einen straftermbasierten genetischen Algorithmus für eine Reihe von Testproblemen schlagen.
Unter Verwendung der Reparaturheuristik CURE wurden die evolutionären Planungsverfahren LaBORNet und BaBORNet entworfen, die sich an den Evolutionstheorien von Lamarck und Baldwin orientieren. Während LaBORNet parallel die Topologie und die Auswahl einer guten Technologie verfolgt, wurden beide Schritte in BaBORNet voneinander getrennt. Untersuchungen für zwei aus der Literatur entnommene Probleme zeigen, dass LaBORNet und BaBORNet bessere Lösungen als bisher bekannte Ansätze liefern. In der direkten Gegenüberstellung beider Verfahren zeigt sich, dass der mit BaBORNet verfolgte Ansatz für große Testprobleme in der Lösungsgüte LaBORNet überlegen ist, jedoch einen deutlich höheren Berechnungsaufwand verursacht.

6 Kommunikationsnetzwerkplanung unter Kostenrestriktionen

Mit Kapitel 5 wurden verschiedene Verfahren zur Planung von Kommunikationsnetzwerken unter Berücksichtigung von Zuverlässigkeitsnebenbedingungen vorgestellt. Mit diesem Kapitel werden Verfahren für die Kommunikationsnetzwerkplanung vorgeschlagen, mit deren Hilfe es möglich ist, zuverlässige Kommunikationsnetzwerktopologien zu entwerfen und dabei eine an den Entscheider gestellte Budgetvorgabe für die maximalen Kosten der Netzwerktopologie zu beachten. In der Literatur wurde dieses Entscheidungsproblem in der Vergangenheit nur unzureichend untersucht (vgl. Abschnitt 3.6.1).
Im Nachfolgenden wird zunächst die Reparaturheuristik STC2 vorgestellt, die (integriert in einen GA) die Planung unter den zuvor genannten Kriterien bei identischen Verbindungszuverlässigkeiten ohne Technologieoptionen ermöglicht. Da die Kommunikationsnetzwerkplanung mit Budgetvorgaben bei identischen Verbindungszuverlässigkeiten ohne Technologieoptionen in der Literatur bisher nicht untersucht wurde, wird zusätzlich eine einfache Greedy-Reparaturheuristik eingeführt und in einen GA integriert. In einer empirischen Untersuchung werden beide GAs für Netzwerke mit unterschiedlicher Komplexität (Anzahl Knoten und Verbindungen) untersucht.
Für die Planung mit unterschiedlichen Technologieoptionen wird eine Reparaturheuristik STC3 vorgeschlagen, welche für einen GA mit einer adaptiven Budgetschranke getestet wird. Für die Planung unter Verwendung unterschiedlicher Technologieoptionen (welche sich hinsichtlich Kosten und Zuverlässigkeit unterscheiden) wird dieser GA anhand unterschiedlicher Testinstanzen empirisch untersucht.

6.1 Kommunikationsnetzwerkplanung mit Budgetvorgaben bei identischen Verbindungszuverlässigkeiten ohne Technologieoptionen

Aufbauend auf der formalen Problembeschreibung in Abschnitt 4.2 auf S. 69 lässt sich das in diesem Abschnitt untersuchte Planungsproblem des Entwurfs von Netzwerktopologien wie folgt beschreiben:

$$\begin{aligned} R_{All}(G_N) &\rightarrow \max \\ C(G_N) &\leq C_B \\ \text{mit } C(G_N) &= \sum_{e_{v_i,v_j} \in E_N} c(e_{v_i,v_j}) \end{aligned} \tag{6.1}$$

Mit Hilfe des Planungsverfahrens soll die Netzwerktopologie (G_N) mit der maximalen All-Terminal-Zuverlässigkeit ($R_{All}(G_N) \rightarrow \max$) und den Kosten ($C_N$) kleiner gleich ei-

ner Kostenschranke/Budget (C_B) gefunden werden. Es wird dabei davon ausgegangen, dass für jede Verbindung lediglich eine Technologieoption l zur Verfügung steht und für alle $e_i \in E$ eine identische Verbindungszuverlässigkeit $r(e_i)$ gegeben ist.
Mit Abschnitt 5.1 konnten die Defizite eines Straftermansatzes dargestellt werden. Die experimentellen Ergebnisse in Abschnitt 5.1.3.2 und Abschnitt 5.2.3.2 zeigen, dass ein Straftermansatz Lösungen schlechterer Güte findet als Verfahren, welche eine Nebenbedingung direkt in Form einer Reparaturheuristik berücksichtigen. Für die Berücksichtigung einer Budgetvorgabe im GA-Design wird deshalb die direkte Berücksichtigung der Budgetnebenbedingung mittels einer Reparaturheuristik gewählt. Mit Abschnitt 6.1.1 wird die Reparaturheuristik STC2 (Spanning Tree Counting 2) vorgestellt, die dies ermöglicht.

6.1.1 STC2 – Eine Spannbaum-Reparaturheuristik für die Kommunikationsnetzwerkplanung bei identischen Verbindungszuverlässigkeiten ohne Technologieoptionen

Für die Planung von Kommunikationsnetzwerken mit dem Ziel der Zuverlässigkeitsmaximierung unter Berücksichtigung eines Budgets muss ein Planungsverfahren wie z. B. ein GA über Mechanismen verfügen, um ungültige Lösungen, deren Kosten das Budget übersteigen, in eine zulässige Lösung zu überführen. In Abschnitt 5.1.2 wurde die Methodik zur Ermittlung der Anzahl der Spannbäume in einem Graphen vorgestellt. Das Verfahren setzt die Erstellung der Gradmatrix voraus und kann mit deren Hilfe die Anzahl der Spannbäume ermitteln. Basierend auf dieser Technik wurde in Abschnitt 5.1.2 die STC-Reparaturheuristik vorgeschlagen. Mit STC2 wird jetzt eine Reparaturheuristik vorgeschlagen, die ebenfalls die Anzahl der Spannbäume im Graphen während des Reparaturprozesses nutzt. Ziel des Verfahrens ist es, die teuersten Kanten, welche den geringsten Einfluss auf die Zuverlässigkeit haben, im Graphen zu identifizieren und diese bis zur Erfüllung der aufgestellen Kostennebenbedingung zu entfernen.
Mit Algorithmus 6.1 wird der komplette Ablauf der STC2-Reparaturheuristik dargestellt. An die Reparaturheuristik wird eine im Sinne der Budgetnebenbedingung ungültige Lösung G_N, welche mit Hilfe eines heuristischen Verfahrens wie z. B. einem GA erzeugt wurde, sowie das Budget C_B übergeben. Der Algorithmus gibt die reparierte Lösung G_N zurück. Für die Eingabe G_N wird zunächst geprüft, ob diese zusammenhängend ist ($trees = 0$). Wird als Eingabe kein zusammenhängender Graph an die Reparaturheuristik übergeben, so werden durch die STC2-Reparaturheuristik solange die teuersten Kanten aus G_N entfernt, bis die Kosten unterhalb der aufgestellten Budgetschranke C_B liegen.
Ist der Eingabegraph G_N verbunden, so versucht die STC2-Reparaturheuristik zur Erreichung der vorgegebenen Budgetkosten von den teuersten Kanten in G_N jeweils die Kanten mit den höchsten Kosten und dem geringsten Einfluss auf die Zuverlässigkeit (hier gemessen an der Anzahl der Spannbäume im Graphen) zu bestimmen. Hierfür werden die Kanten $e_{v_i,v_j} \in E_N$ hinsichtlich ihrer Kosten $c(e_{v_i,v_j})$ sortiert. In der for-Schleife (Zeile 11) werden nacheinander[49] die Kanten in L temporär aus G_N entfernt und die

[49] Es wird jeweils nur eine Kante aus G_N temporär entfernt.

Algorithmus 6.1 STC2-Reparaturheuristik

Eingabe: G_N, C_B
Ausgabe: G_N

1: $trees$ = #SpanningTrees(G_N) {Bestimmt mittels Gradmatrix}
2: **if** $trees = 0$ **then**
3: **while** $C(G_N) > C_B$ **do**
4: $e_{work} = max\{c(e_{v_i,v_j})|e_{v_i,v_j} \in E_N\}$
5: $E_N = E_N \setminus \{e_{work}\}$
6: **end while**
7: return G_N
8: **else**
9: $L = \{e_1, .., e_m\}$ mit $e_{v_i,v_j} \in G_N$ sortiert nach Kosten $c(e_{v_i,v_j})$
10: $minRatio = \infty, e_{minRatioEdge} = \emptyset$
11: **for** $n = 0$; $n < |L|$; $n{+}{+}$ **do**
12: Lösche $e_n \in L$ temporär in G_N und erstelle Gradmatrix DM_{temp}
13: $numTrees$ = #SpanningTrees(G_N) {Bestimmt mittels $det(DM_{temp})$}
14: $ratio = numTrees/c(e_n)$
15: **if** $ratio < minRatio$ **then**
16: $minRatio = ratio$
17: $e_{minRatioEdge} = e_n$
18: **end if**
19: **end for**
20: $E_N = E_N \setminus \{e_{minRatioEdge}\}$
21: **if** $C(G_N) > C_B$ **then**
22: rufe STC2 auf
23: **end if**
24: **end if**

Veränderung in der Anzahl der Spannbäume (#SpanningTrees) gemessen. Für jede der Kanten werden ihre Kosten in das Verhältnis zur Anzahl der Spannbäume, die aus der temporären Löschung der Kante resultieren, gesetzt. Die Kante mit dem geringsten Verhältnis wird anschließend aus der Kantenmenge E_N des Graphen G_N gelöscht. Wenn die Kosten $C(G_N)$ des Graphen nach der Löschung größer dem Budget C_B sind, so wird die STC2-Reparaturheuristik erneut aufgerufen.

6.1.2 CostliestGreedy – Eine Greedy-Heuristik für die Kommunikationsnetzwerkplanung

Bisher wurden in der Literatur noch keine Verfahren veröffentlicht, die die Kommunikationsnetzwerkplanung unter Kostenrestriktionen bei identischen Verbindungszuverlässigkeiten ermöglichen. Für die experimentelle Untersuchung der STC2-Reparaturheuristik wird daher als Vergleichsverfahren zusätzlich eine einfache Greedy-Heuristik eingeführt. Anders als die STC2-Reparaturheuristik bezieht diese Greedy-Heuristik die Zuverlässigkeit der Netzwerktopologie nur sehr begrenzt mit in den Reparaturprozess ein.

Den vollständigen Ablauf der Greedy-Reparaturheuristik zeigt Algorithmus 6.2. Die Reparaturheuristik trägt den Namen CostliestGreedy, was bereits den Ansatz des Verfahrens deutlich macht. Mittels der while-Schleife (Zeile 4-12) wird solange die teuerste

Algorithmus 6.2 CostliestGreedy-Reparaturheuristik

Eingabe: G_N, C_B
Ausgabe: G_N

```
1:  Queue disconEdges = ∅
2:  L = {e_1, .., e_m} mit e_{v_i,v_j} ∈ G_N sortiert nach Kosten c(e_{v_i,v_j})
3:  counter = 0
4:  while C(G_N) > C_B und counter < |L| do
5:     e_work = e_counter mit e_counter ∈ L
6:     E_N = E_N \ {e_work}
7:     if #SpanningTrees(G_N) = 0 then
8:        E_N = E_N ∪ {e_work}
9:        disconEdges ∪ {e_work}
10:    end if
11:    counter++
12: end while
13: counter = 0
14: while C(G_N) > C_B do
15:    e_work = e_counter mit e_counter ∈ disconEdges
16:    E_N = E_N \ {e_work}
17: end while
```

Kante aus dem Graph G_N gelöscht, bis dessen Kosten $C(G_N) \leq C_B$ sind. Damit die „reparierten“ Lösungen durch die Reparatur nicht eine All-Terminal-Zuverlässigkeit von null[50] besitzen, löscht die Reparaturheuristik Kanten, durch deren Löschung der Graph nicht mehr zusammenhängend ist, nur dann, wenn deren Löschung für die Einhaltung des Budgets zwingend erforderlich ist. Hierfür werden in der while-Schleife (Zeile 4-12) sämtliche Verbindungen, deren Löschung den Zusammenhang[51] des Graphen zerstörten, zunächst in der Queue *disconEdges* gespeichert und nicht gelöscht (Zeile 7-10).
Wenn die Kosten des Graphen nach der Löschung sämtlicher Kanten, die den Zusammenhang im Graphen nicht zerstören, noch immer größer als das geforderte Budget sind, so werden sukzessive die Kanten aus *disconEdges* entfernt, bis $C(G_N) \leq C_B$.

[50] Eine All-Terminal-Zuverlässigkeit von null tritt dann ein, wenn der Graph nicht zusammenhängend ist.
[51] Für einen nicht zusammenhängenden Graphen G_N gilt: #SpanningTrees(G_N) = 0.

6.1.3 Ein evolutionäres Planungsverfahren für die Kommunikationsnetzwerkplanung unter Budgetvorgaben unter Verwendung der STC2-Reparaturheuristik

Mit diesem Abschnitt wird die Integration der in Abschnitt 6.1.1 und Abschnitt 6.1.2 beschriebenen Reparaturheuristiken in einen GA für die Kommunikationsnetzwerkplanung mit Budgetvorgaben bei identischen Verbindungszuverlässigkeiten ohne Technologieoptionen betrachtet. In einer experimentellen Studie werden beide Verfahren an unterschiedlichen Testinstanzen untersucht und die gewonnenen Ergebnisse gegenübergestellt.
Zielsetzung des im Folgenden vorgestellten evolutionären Verfahren ist es, für eine gegebene Menge von Verbindungen, die eine feste Anzahl an vorgegebenen Netzwerkknoten verbinden, eine Kantenmenge auszuwählen, welche die All-Terminal-Zuverlässigkeit R_{All} maximiert und dabei eine Budgetvorgabe C_B für die Kosten der Netzwerktopologie nicht überschreitet. Ein Netzwerk wird hierfür wie in Abschnitt 4.2 beschrieben als stochastischer ungerichteter Graph modelliert. Es gelten die Modellannahmen aus Abschnitt 3.1.1. Es ist gegeben, dass sämtliche Kanten des Graphen die gleiche Zuverlässigkeit $r(e_i)$ besitzen. Die Zielfunktion für das in diesem Abschnitt betrachtete Optimierungsproblem lautet:

$$\begin{aligned} f(y) = R_{All}(G_N) &\rightarrow \max \\ C(G_N) &\leq C_B \end{aligned} \tag{6.2}$$

Wie in Abschnitt 4.2 eingeführt, entspricht $C(G_N)$ den Kosten einer Lösung, die durch den Graphen G_N beschrieben wird und $c(e_{v_i,v_j})$ den Kosten der Verbindung zwischen den Knoten v_i und v_j.

6.1.3.1 Experimentelles Design

Für die experimentelle Untersuchung der entworfenen Reparaturheuristiken wurde ein Steady-State-GA verwendet. Als Zielfunktionswert wird die All-Terminal-Zuverlässigkeit einer Lösung verwendet. Für den GA wird eine Lösung binär durch ein Genom der Länge $|E|$ kodiert. Abbildung 6.1 zeigt die Kodierung eines Graphen mit vier Knoten und sechs möglichen Kanten als Genom[52].

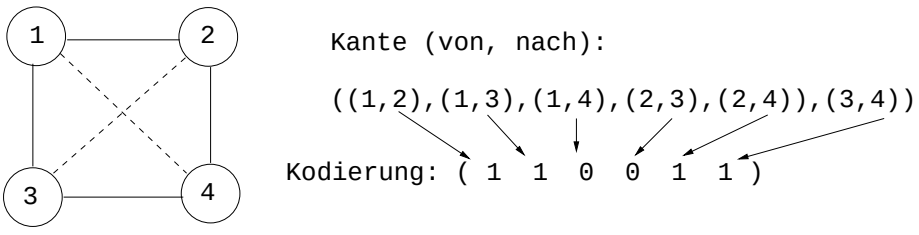


Abbildung 6.1: Kodierung eines Netzwerkes als Genom für STC2-GA und CostliestGreedy-GA

[52] Mögliche, aber derzeit nicht im Graphen verwendete Kanten sind gestrichelt dargestellt.

Tabelle 6.1: Parameter für STC-GA und Strafterm-GA

GA	Steady-State-GA
Crossover-Operator	Uniform-Crossover, $p_{cross} = 0,9$
Mutations-Operator	Flip-Mutation, $p_{mut} = 0,01$
Initialisierung von P_0	zufallsbasiert, $p_{init} = 0,6$
Bestimmung von R_{All}	Für $\|V\| > 15$ einfache Monte-Carlo-Simulation sonst lokaler Dekompositionsansatz
pop	100
Anzahl GA-Läufe	10

Der für die Experimente verwendete GA nutzt einen Uniform-Crossover-Operator (siehe S. 47) und setzt eine Flip-Mutation (siehe S. 48) ein. Die initiale Population wurde aus zufällig generierten Lösungen erstellt. Die Wahrscheinlichkeit (p_{init}), dass eine Verbindung zwischen zwei Knoten durch das Initialisierungsverfahren erstellt wird, beträgt 60 %. Das Initialisierungsverfahren stellt dabei sicher, dass jede initiale Lösung zusammenhängend ist. Zur Berechnung der All-Terminal-Zuverlässigkeit wurde der in Abschnitt 4.4.1.2 beschriebene Dekompositionansatz sowie eine Monte-Carlo basierte Schätzung eingesetzt. Für Netzwerke mit weniger als 16 Knoten wurde für die Berechnung von R_{All} das exakte Berechnungsverfahren mittels des Dekompositionansatzes durchgeführt. Für Netzwerke mit mehr als 15 Knoten wurde eine einfache Monte-Carlo-Simulation mit jeweils $M = 50.000$ Stichproben genutzt.
Wurde durch einen der GA-Operatoren (Initialisierung, Rekombination, Mutation) eine Lösung erzeugt, deren Kosten größer als das geforderte Budget waren, so wurde diese mittels einer der beiden Reparaturheuristiken (STC2 oder CostliestGreedy) in eine gültige Lösung überführt. Für die Testprobleme wurde sichergestellt, dass für das verwendete Budget (C_B) eine Lösung mit $R_{All} > 0$ existiert.
Für die Experimente wurden die folgenden GA-Parameter verwendet: Ein Steady-State-GA[53] mit sich zu 50 % überlappenden Populationen[54]. Die Crossover-Wahrscheinlichkeit beträgt $p_{cross} = 0,9$, die Mutationswahrscheinlichkeit beträgt $p_{mut} = 0,01$ und die Populationsgröße wurde auf $pop = 100$ festgelegt. Ein GA-Durchlauf wurde nach 200 Generationen oder wenn in den letzten 20 Generationen keine Verbesserung der besten Lösung stattgefunden hat, angehalten. Für jedes Testproblem und jeden GA mit der jeweiligen Reparaturheuristik wurden zehn voneinander unabhängige Läufe durchgeführt. Tabelle 6.1 fasst die für die Experimente verwendeten Parameter zusammen. Die Auswahl der hier vorgestellten GA-Operatoren und -Parameter wurde in einer experimentellen Studie als die beste Parameterkombination ermittelt. Beide GAs wurden in der GALib [188] unter Linux implementiert und getestet. Für die Experimente kam auf ein P4-2-GHz-System zum Einsatz.

[53] Die Vorteile des Steady-State-GAs gegenüber einem kanonischen GA werden in Abschnitt 5.1.3.1 auf S. 93 beschrieben.

[54] Der prozentuale Anteil für die Überlappung der Populationen bestimmt dabei den Anteil der Lösungen in der Elternpopulation, die durch bessere Lösungen aus der Kindergeneration ersetzt werden.

6.1.3.2 Experimentelle Ergebnisse und Auswertung

Für die empirische Untersuchung der GAs unter Verwendung der CostliestGreedy- und STC2-Reparaturheuristik werden die in [52] eingeführten Testinstanzen sowie die mit Abschnitt 5.1.3.2 auf S. 96 eingeführten Netzwerke deutsch15, deutsch20, deutsch25 und deutsch30 verwendet. Die Budgets C_B für die aus [52] entnommenen Testprobleme orientieren sich an den Kosten der optimalen Lösung. Für die Netzwerke deutsch15, deutsch20, deutsch25 und deutsch30 wurde das Budget in Anlehnung an die in Abschnitt 5.2.3 ermittelten minimalen Kosten gewählt.
Mit Tabelle 6.2 und Tabelle 6.3 werden die Ergebnisse für einen GA unter Verwendung der STC2- und der CostliestGreedy[55]-Reparaturheuristik für die Testprobleme aus [52] gezeigt. Tabelle 6.4 zeigt die Ergebnisse für die Netzwerke deutsch15, deutsch20, deutsch25 und deutsch30.
Die Tabellen 6.2, 6.3 und 6.4 zeigen jeweils die Anzahl der Knoten[56] $|V|$ sowie die Zuverlässigkeit der Verbindungen $r(e_i)$. Netzwerke mit identischer Knoten- und Verbindungsanzahl in Tabelle 6.2 und Tabelle 6.3 unterscheiden sich jeweils in der Position der Knoten und daraus resultierend in den Verbindungskosten. Für den GA mit der STC2-Reparaturheuristik und den GA mit der CostliestGreedy-Reparaturheuristik wird der Mittelwert[57] der besten All-Terminal-Zuverlässigkeit μ von R_{All}^{max} der zehn Läufe, die maximale All-Terminal-Zuverlässigkeit R_{All}^{Beste} aus allen zehn Läufen, der Mittelwert[58] der Kosten (μ von $C_{R_{All}^{max}}$) der jeweils besten Lösungen aus allen zehn Läufen sowie die Kosten C_{Beste} der Lösung mit der höchsten All-Terminal-Zuverlässigkeit dargestellt. Die Spalte R_{All}^{opt} zeigt für jedes untersuchte Testproblem die jeweils beste All-Terminal-Zuverlässigkeit aus allen GA-Durchläufen. Die letzte Spalte $\#Eval$ gibt für jedes Verfahren die über alle zehn Läufe gemittelte Anzahl der Fitnessbewertungen pro GA-Durchlauf an. Betrachtet man die maximal erreichte All-Terminal-Zuverlässigkeit R_{All}^{opt}, die unter den gegebenen Budgets für die aus [52] entnommenen Testprobleme erreicht wurde, so ist festzustellen, dass für fast alle untersuchten Probleme diese größer bzw. gleich der für die Netzwerke untersuchten Zuverlässigkeitsschranke R_0 in Abschnitt 5.1.3.2 ist. Die Ergebnisse zeigen die Fähigkeit der untersuchten GAs (mit den hier propagierten Reparaturheuristiken), für die untersuchten Testprobleme ebenfalls die in Abschnitt 5.1.3.2 gezeigten Ergebnisse zu erzielen. Die beste hier ermittelte Lösung (erreicht mit dem CostliestGreedy-GA) für das Problem deutsch15 liegt mit den Kosten $C_{Beste} = 2366$ (bei Einhaltung der Zuverlässigkeitsnebenbedingung $R_0 \geq 0,95$) sogar noch eine Geldeinheit unter der besten Lösung aus Abschnitt 5.1.3.2.
Eine Gegenüberstellung der Ergebnisse des STC2-GAs und des CostliestGreedy-GAs verdeutlicht, dass beide Verfahren für fast alle der hier untersuchten Netzwerke identische beste Lösungen erzeugen. Somit ist anhand der Qualität der besten Lösung keine klare Präferenz für eine der beiden vorgeschlagenen Reparaturheuristiken zu sehen. Eine Betrachtung des Mittelwertes (μ) für die All-Terminal-Zuverlässigkeit (R_{All}^{max}) der bes-

[55] In den Tabellen kurz als Greedy-GA bezeichnet.
[56] Da sämtliche Eingabegraphen vollständig verbunden sind, ergibt sich die Anzahl der möglichen Verbindungen als $|V| \cdot |V-1|/2$
[57] Aus Platzgründen wurde auf die Angabe der Standardabweichung verzichtet.
[58] Aus Platzgründen wurde auf die Angabe der Standardabweichung verzichtet.

Tabelle 6.2: Ergebnisse für die Netzwerke mit sechs bis acht Knoten aus [52] mit STC2-GA und Greedy-GA

					STC2-GA					Greedy-GA				
	$\|V\|$	$r(e_i)$	C_B	R_{All}^{opt}	R_{All}^{max} μ	R_{All}^{Beste}	$C_{R_{All}^{max}}$ μ	C_{Beste}	#Eval	R_{All}^{max} μ	R_{All}^{Beste}	$C_{R_{All}^{max}}$ μ	C_{Beste}	#Eval
1	6	0,9	240	0,9343	0,8938	0,9343	226,2	231	1.055	0,8837	0,8837	224,0	224	1.057
2	6	0,9	240	0,9454	0,9454	0,9454	239,0	239	1.055	0,9454	0,9454	239,0	239	1.101
3	6	0,9	230	0,9408	0,9408	0,9408	227,0	227	1.057	0,9408	0,9408	227,0	227	1.070
4	6	0,9	220	0,9240	0,9240	0,9240	212,0	212	1.057	0,9143	0,9240	212,3	212	1.214
5	6	0,9	190	0,9389	0,9027	0,9389	178,60	184	1.073	0,8853	0,9389	176	184	1.090
1	6	0,95	230	0,9666	0,9463	0,9463	224,0	224	1.056	0,9495	0,9666	224,4	228	1.067
2	6	0,95	220	0,9686	0,9532	0,9686	203,4	213	1.084	0,9337	0,9686	207,5	213	1.080
3	6	0,95	200	0,9669	0,9669	0,9669	190,0	190	1.055	0,9669	0,9669	190,0	190	1.066
4	6	0,95	200	0,9682	0,9507	0,9682	199,4	200	1.478	0,9420	0,9682	199,1	200	1.086
5	6	0,95	180	0,9475	0,9462	0,9462	173,0	173	1.058	0,9475	0,9475	178,0	178	1.054
1	7	0,9	190	0,9145	0,8791	0,9145	184,1	189	1.275	0,8808	0,9145	186,7	189	1.206
2	7	0,9	190	0,9153	0,9008	0,9149	181,8	184	1.080	0,9007	0,9153	181,0	184	1.203
3	7	0,9	250	0,9052	0,8988	0,9052	238,5	243	1.200	0,8923	0,9052	235,7	243	1.290
4	7	0,9	130	0,9149	0,8765	0,9149	129,8	129	1.062	0,8697	0,9149	129,8	129	1.101
5	7	0,9	130	0,9261	0,9261	0,9261	124,0	124	1.061	0,9166	0,9261	123,7	124	1.203
1	7	0,95	190	0,9456	0,9449	0,9456	183,2	180	1.065	0,9439	0,9444	180,0	180	1.108
2	7	0,95	190	0,9783	0,9741	0,9783	183,3	184	1.116	0,9783	0,9783	184,0	184	1.127
3	7	0,95	230	0,9562	0,9371	0,9387	224,4	224	1.061	0,9391	0,9562	224,6	230	1.068
4	7	0,95	130	0,9781	0,9580	0,9781	129,6	129	1.100	0,9524	0,9781	129,4	129	1.116
5	7	0,95	130	0,9823	0,9823	0,9823	124,0	124	1.062	0,9823	0,9823	124,0	124	1.212
1	8	0,9	210	0,9336	0,9336	0,9336	208,0	208	1.168	0,9336	0,9336	208,0	208	1.304
2	8	0,9	210	0,9227	0,9203	0,9227	207,5	208	1.061	0,9227	0,9227	208,0	208	1.303
3	8	0,9	220	0,9281	0,9124	0,9281	212,8	213	1.537	0,8943	0,9281	216,3	213	1.577
4	8	0,9	300	0,9201	0,9134	0,9197	292,7	295	1.651	0,8929	0,9201	289,1	291	1.608
5	8	0,9	180	0,9082	0,8749	0,9082	174,7	178	1.166	0,8680	0,9082	173,5	178	1.342
1	8	0,95	180	0,9646	0,9646	0,9646	179,0	179	1.062	0,9646	0,9646	179,0	179	1.256
2	8	0,95	200	0,9692	0,9629	0,9692	195,5	197	1.081	0,9642	0,9692	196,2	197	1.239
3	8	0,95	200	0,9698	0,9414	0,9698	194,9	199	1.157	0,9478	0,9698	196,9	199	1.403
4	8	0,95	280	0,9642	0,9469	0,9642	276,4	279	1.533	0,9441	0,9617	274,4	280	1.515
5	8	0,95	180	0,9766	0,9507	0,9766	175,0	178	1.117	0,9470	0,9766	173,9	178	1.258

Tabelle 6.3: Ergebnisse für die Netzwerke mit neun bis elf Knoten aus [52] mit STC2-GA und Greedy-GA

					STC2-GA					Greedy-GA				
	$\|V\|$	$r(e_i)$	C_B	R_{All}^{opt}	R_{All}^{max} μ	R_{All}^{Beste}	$C_{R_{All}^{max}}$ μ	C_{Beste}	#Eval	R_{All}^{max} μ	R_{All}^{Beste}	$C_{R_{All}^{max}}$ μ	C_{Beste}	#Eval
1	9	0,9	240	0,9074	0,8767	0,9074	236,8	239	1.311	0,8774	0,9074	235,7	239	1.755
2	9	0,9	200	0,9240	0,9240	0,9240	200,0	200	1.180	0,9122	0,9240	198,8	200	1.515
3	9	0,9	260	0,9030	0,8636	0,9030	257,9	257	1.734	0,8741	0,8988	250,7	252	2.129
4	9	0,9	180	0,9151	0,9133	0,9151	171,7	171	1.098	0,9130	0,9151	172,6	171	1.485
5	9	0,9	200	0,9149	0,9149	0,9149	198,0	198	1.324	0,9041	0,9149	198,1	198	1.399
1	9	0,95	210	0,9669	0,9667	0,9669	209,0	209	1.696	0,9537	0,9669	206,5	209	1.478
2	9	0,95	200	0,9807	0,9807	0,9807	200,0	200	1.134	0,9760	0,9807	199,6	200	1.255
3	9	0,95	240	0,9677	0,9278	0,9677	231,3	233	1.481	0,9322	0,9677	234,5	233	1.809
4	9	0,95	160	0,9600	0,9438	0,9588	159,4	160	1.076	0,9509	0,9600	157,0	151	1.290
5	9	0,95	190	0,9688	0,9399	0,9615	185,8	185	1.127	0,9425	0,9688	185,9	188	1.344
1	10	0,9	140	0,9366	0,9366	0,9366	140,0	140	1.870	0,9208	0,9366	139,8	140	1.847
2	10	0,9	160	0,9045	0,9039	0,9045	154,2	154	1.248	0,9008	0,9045	154,8	154	1.548
3	10	0,9	270	0,9039	0,8745	0,9039	267,6	267	1.610	0,8694	0,9039	268,0	267	1.682
4	10	0,9	270	0,9354	0,8761	0,9354	265,7	263	1.127	0,8925	0,9354	264,8	263	1.464
5	10	0,9	300	0,9066	0,8625	0,9066	294,7	293	2.017	0,8576	0,9066	293,5	293	1.738
1	10	0,95	130	0,9741	0,9525	0,9741	128,8	128	1.690	0,9578	0,9720	129,0	130	1.831
2	10	0,95	140	0,9618	0,9618	0,9618	136,0	136	1.458	0,9526	0,9618	135,7	136	1.942
3	10	0,95	240	0,9324	0,9324	0,9324	240,0	240	1.077	0,9322	0,9324	239,5	240	1.216
4	10	0,95	250	0,9741	0,9502	0,9741	248,1	246	1.148	0,9510	0,9741	248,1	246	1.244
5	10	0,95	270	0,9310	0,9268	0,9310	268,7	270	1.189	0,9267	0,9310	269,2	270	1.508
	11	0,9	250	0,9177	0,9055	0,9177	246,6	246	1.846	0,9112	0,9177	246,9	246	1.896
	11	0,95	210	0,9531	0,9203	0,9531	206,2	210	1.609	0,9159	0,9531	205,0	210	1.353

ten Lösungen der einzelnen GA-Läufe für beide getestete GAs lässt ebenfalls keine klare Dominanz eines Ansatzes über den anderen Ansatz erkennen. Beide GAs erzeugen im Mittel in allen Läufen sehr gute Lösungen, die nahe an der besten gefundenen Lösung liegen. Für einige Probleme, wie z. B. für die Testinstanz 6-Knoten, Netzwerk 3, $r(e_i) = 0,9$ wurde in allen zehn Läufen von beiden Verfahren jeweils die gleiche beste Lösung gefunden. Die experimentellen Ergebnisse zeigen, dass der Abstand zwischen $\mu_{R_{All}^{max}}$ und R_{All}^{Beste} mit der Problemgröße und damit verbunden mit der Größe des Lösungraums[59] wächst.

Abbildung 6.2 stellt die Entwicklung der Qualität der Lösungen (hier gemessen an der über alle zehn Läufe je Generation gemittelten All-Terminal-Zuverlässigkeit der besten und schlechtesten Lösung) grafisch dar. Exemplarisch werden die Ergebnisse für das Netzwerk 1 mit zehn Knoten und $r(e_i) = 0,9$ (siehe Abbildung 6.2(a)) sowie die Ergebnisse für das Netzwerk deutsch20 (siehe Abbildung 6.2(b)) dargestellt. Die Verläufe der Kurven in Abbildung 6.2(a) zeigen, dass die All-Terminal-Zuverlässigkeit der besten und

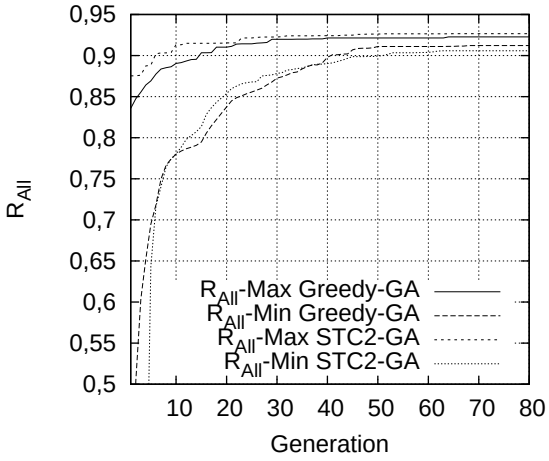
[59] Während für ein Problem mit sechs Knoten und 15 Kanten lediglich $2^{15} = 32.768$ Lösungen möglich sind, besteht der Lösungsraum für ein 10-Knoten-Netzwerk mit 45 Kanten bereits aus $2^{45} = 3,52 \cdot 10^{13}$ Lösungen.

Tabelle 6.4: Ergebnisse mit STC2-GA und Greedy-GA für die Netzwerke deutsch15, deutsch20, deutsch 25 und deutsch30

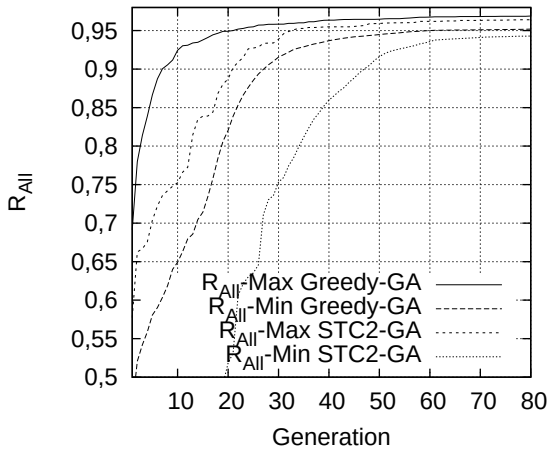
				STC2-GA					Greedy-GA				
$\|V\|$	$r(e_i)$	C_B	R_{All}^{opt}	R_{All}^{max} μ	R_{All}^{Beste}	$C_{R_{All}^{max}}$ μ	C_{Beste}	#Eval	R_{All}^{max} μ	R_{All}^{Beste}	$C_{R_{All}^{max}}$ μ	C_{Beste}	#Eval
15	0,95	2400	0,9545	0,9512	0,9538	2.371	2368	2944	0,9505	0,9545	2679	2366	2.679
20	0,95	2900	0,9716	0,9639	0,9695	2.870	2870	2872	0,9675	0,9716	2863	2878	2.898
25	0,95	3500	0,9891	0,6494	0,9840	34.687	3435	2722	0,9860	0,9891	3475	3486	2.783
30	0,95	6100	0,9991	0,9980	0,9991	6.072	6074	3641	0,9975	0,9987	5990	6081	2.206

der schlechtesten Lösung über die Generationen hinweg zunimmt. Daraus ableiten lässt sich, dass der GA unter Verwendung der hier vorgeschlagenen Reparaturheuristiken in der Lage ist, die Suche auf Regionen mit guten Lösungen im Suchraum zu konzentrieren. Beide in Abbildung 6.2(a) dargestellten GAs stoppen nach 80 Generationen aufgrund der vorgegebenen Abbruchbedingung für die Konvergenz bei gleicher bester Lösung in den letzten 20 Generationen. In beiden Grafiken ist eine nahezu identische All-Terminal-Zuverlässigkeit der besten und schlechtesten Lösungen zu erkennen.
Während für das 10-Knoten-Problem die Kurvenverläufe und damit das Konvergenzverhalten für den STC2-GA und den CostliestGreedy-GA nahezu identisch sind, wird in Abbildung 6.2(b) ein anderes Verhalten sichtbar. Wie Abbildung 6.2(b) zeigt, liegt die All-Terminal-Zuverlässigkeit der besten und der schlechtesten Lösung für den CostliestGreedy-GA bis Generation 30 deutlich über der des STC2-GA. Ab Generation 40 nähern sich die Verläufe der Kurven für die beste Lösung für beide Verfahren an, so dass beide GAs in Generation 80 mit nahezu identischen Lösungen konvergieren. Im Gegensatz zur Darstellung in Abbildung 6.2(a) liegen die Werte für die schlechteste und beste Lösung in Abbildung 6.2(b) bis Generation 50 jedoch noch deutlich auseinander. Die Anzahl der je GA-Durchlauf durchgeführten Fitnessbewertungen[60] ermöglicht eine Aussage über den Rechenaufwand, der mit dem Einsatz des Verfahrens verbunden ist. Wie eine Laufzeitanalyse eines GAs zur Planung von Netzwerktopologien in Abschnitt 4.5 zeigte, hat die Berechnung der All-Terminal-Zuverlässigkeit einen sehr großen Einfluss auf die Gesamtlaufzeit des Verfahrens. Für die hier vorgestellten Verfahren trifft diese Aussage ebenfalls zu. Der Berechnungsaufwand, der durch die Reparaturheuristiken entsteht, ist im Vergleich zum Berechnungsaufwand für die Fitnessbewertung (und hier insbesondere der All-Terminal-Zuverlässigkeitsbestimmung) zu vernachlässigen. Daher erfolgt jetzt die Analyse der Laufzeit anhand der jeweils durchgeführten Fitnessbewertungen. Abbildung 6.3 stellt die über jeweils zehn Läufe gemittelte Anzahl der Fitnessbewertungen ($\#Eval$) je GA-Durchlauf für die unterschiedlichen hier untersuchten Problemgrößen dar. Die Laufzeiten für Testinstanzen mit der gleichen Anzahl an Knoten wurden hierfür zusammengefasst. Abbildung 6.3(a) zeigt dabei die Anzahl der Fitnessbewertung für Netzwerke mit $r(e_i) = 0,9$ und Abbildung 6.3(b) die Anzahl der Fitnessbewertungen bei $r(e_i) = 0,95$.

[60] Pro Fitnessbewertung erfolgt die Ermittlung der Kosten und der All-Terminal-Zuverlässigkeit einer Lösung.



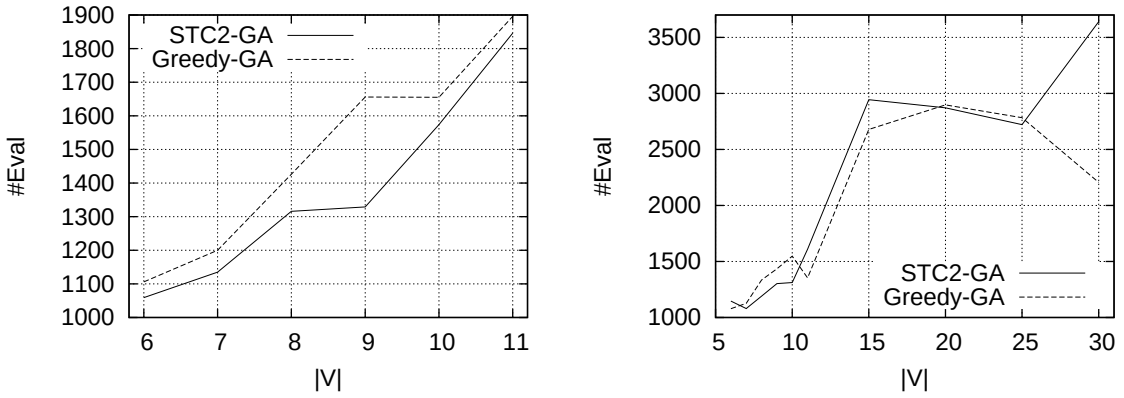
(a) 10 Knoten, Netzwerk 1, $r(e_i) = 0,9$, $C_B = 140$



(b) deutsch20, $r(e_i) = 0,95$, $C_B = 2900$

Abbildung 6.2: Performancevergleich für STC2-GA und CostliestGreedy-GA anhand der Entwicklung der gemittelten maximalen und minimalen All-Terminal-Zuverlässigkeit

Prinzipiell lässt sich feststellen, dass die Anzahl der durchschnittlich durchgeführten Fitnessbewertungen mit der Größe des Problemraums zunimmt. Für die untersuchten Netzwerke mit $|V| \leq 11$ und $r(e_i) = 0,9$ bzw. $r(e_i) = 0,95$ liegt die Anzahl der Fitnessbewertungen, die durch den STC2-GA durchgeführt wurden, stets unter denen des CostliestGreedy-GA. Im Durchschnitt benötigt der STC2-GA 50–100 Fitnessbewertungen weniger. Eine Ausnahme bildet die deutliche Abweichung in Abbildung 6.3(a) für die 9-KnotenProbleme. Aufgrund der hohen Anzahl an Fitnessbewertungen für das Netzwerk 3 ($\#Eval = 2.129$) durch den CostliestGreedy-GA ergibt sich hier ein signifikanter Unterschied zu den STC2-GA Fitnessbewertungen.
Betrachtet man für dieses Netzwerk zusätzlich die Qualität der besten Lösung, so ist der zusätzliche Rechenaufwand gerechtfertigt. Für den CostliestGreedy-GA sind die gemittelte All-Terminal-Zuverlässigkeit und die gemittelten Kosten der besten Lösungen über alle zehn durchgeführten GA-Läufe besser als die STC2-GA Ergebnisse und die All-Terminal-Zuverlässigkeit der besten Lösung nur geringfügig schlechter als die der besten STC2-GA Lösung (bei fünf Geldeinheiten geringeren Kosten).
Die in Abbildung 6.3(b) dargestellten durchschnittlichen Fitnessbewertungen in Abhängigkeit von der Problemgröße zeigen, dass für den STC2-GA und den CostliestGreedy-GA bis zu einer Problemgröße von $|V| \leq 25$ keine signifikanten Unterschiede existieren. Für das Netzwerk deutsch30 liegt der Berechnungsaufwand des STC2-GAs jedoch deutlich über dem des CostliestGreedy-GA. Da für die Testinstanz deutsch30 im Gegensatz zu den Netzwerken aus [52] jedoch nur zehn GA-Läufe für die Ermittlung des Ergebnis-

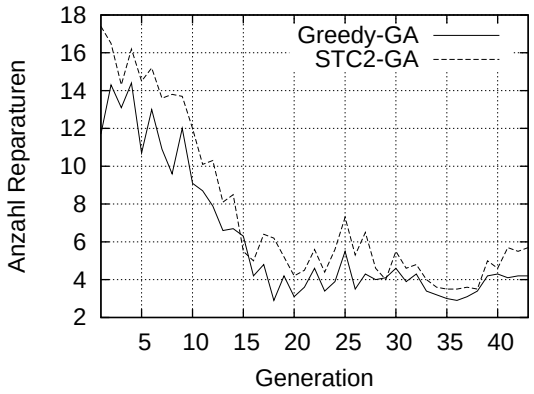


(a) Verbindungszuverlässigkeit $r(e_i) = 0,9$ (b) Verbindungszuverlässigkeit $r(e_i) = 0,95$

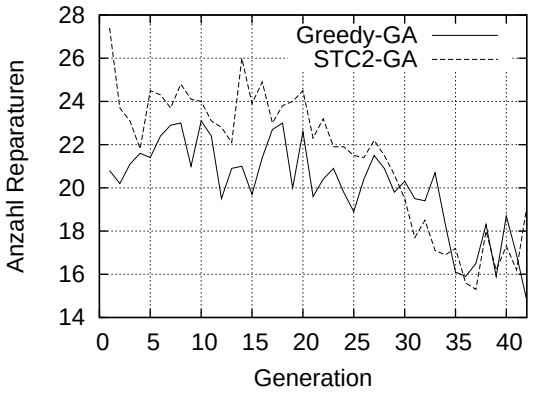
Abbildung 6.3: Gegenüberstellung der Laufzeiten STC2-GA und CostliestGreedy-GA bei unterschiedlichen Verbindungszuverlässigkeiten und unterschiedlichen Problemgrößen

ses zur Verfügung stehen[61], wird auf eine Verallgemeinerung der Ergebnisse für $|V| \geq 15$ verzichtet.
Mit Abbildung 6.4 wird die gemittelte Anzahl der Aufrufe der Reparaturheuristik für ein 6-Knoten- und ein 10-Knoten-Netzwerk im Vergleich zur Entwicklung der gemittelten All-Terminal-Zuverlässigkeit sämtlicher Lösungen einer Population über die Generationen hinweg dargestellt. Der Verlauf der Kurven zeigt, dass die Anzahl der Reparaturen über die Generationen hinweg abnimmt. Für die 6-Knoten-Netzwerke ist festzustellen, dass bereits nach zirka 15 Generationen nur noch wenige (im Durchschnitt fünf) Lösungen in jeder Generation durch die jeweilige Reparaturheuristik in eine gültige Lösung überführt werden müssen. Betrachtet man den Verlauf der mittleren All-Terminal-Zuverlässigkeit (gemessen an R_{AVG}) ab diesem Zeitpunkt, so ist festzustellen, dass der GA beginnt zu konvergieren. Dabei enthält die Population eine Vielzahl gleichartiger, gültiger Lösungen. Veränderungen in der Population werden zu diesem Zeitpunkt primär durch den Mutationsoperator verursacht. Dieser generiert nur wenige ungültige Lösungen, welche an die Reparaturheuristik übergeben werden, so dass die Anzahl der Reparaturaufrufe abnimmt.
Für das in Abbildung 6.4(b) und Abbildung 6.4(d) dargestellte 10-Knoten-Netzwerk ist ein ähnlicher Verlauf zu erkennen. Über die Generationen hinweg nimmt die Anzahl der Reparaturen ab. Gleichzeitig zeigt der dargestellte Fitnessverlauf, dass eine Konvergenz auf einem All-Terminal-Zuverlässigkeitsniveau stattfindet.
Stellt man die Reparaturen des CostliestGreedy- und STC2-GAs gegenüber, so zeigen die dargestellten Verläufe, dass bei Verwendung der vorgeschlagenen CostliestGreedy-Reparaturheuristik häufig weniger Reparaturen je Generation notwendig sind, als beim Einsatz des STC2-Verfahrens.

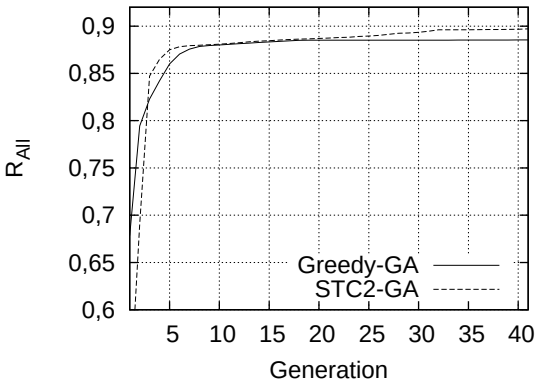
[61] Für die Netzwerke mit sechs bis acht Knoten wurden jeweils fünf Netzwerkinstanzen mit jeweils zehn GA-Läufen gemittelt.



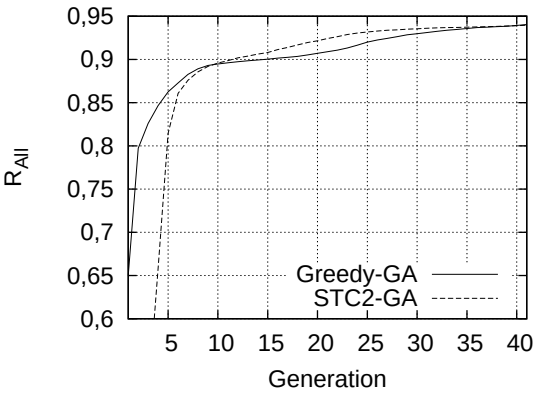
(a) Reparaturaufwand - 6-Knoten, Netzwerk 1, $r(e_i) = 0,9$, $C_B = 240$



(b) Reparaturaufwand - 10-Knoten, Netzwerk 2, $r(e_i) = 0,95$, $C_B = 160$



(c) Fitnessentwicklung - 6-Knoten, Netzwerk 1, $r(e_i) = 0,9$, $C_B = 240$



(d) Fitnessentwicklung - 10-Knoten, Netzwerk 2, $r(e_i) = 0,95$, $C_B = 160$

Abbildung 6.4: Gegenüberstellung des Reparaturaufwandes und des Fitnessverlaufs für STC2-GA und CostliestGreedy-GA

Betrachtet man die Qualität der Lösungen anhand der erreichten Zielfunktionswerte für die All-Terminal-Zuverlässigkeit und den dafür benötigten Aufwand gemessen an der Anzahl der Fitnessbewertungen, so kann zusammenfassend festgestellt werden, dass beide Verfahren sehr gute Lösungen mit einem vertretbaren Aufwand generieren. Beide der hier vorgeschlagenen Reparaturheuristiken eignen sich als Methoden zur Einhaltung einer Kostennebenbedingung in einem GA zur Planung von zuverlässigen Netzwerktopologien bei identischen Verbindungszuverlässigkeiten. Aufgrund der in der experimentellen Untersuchung erzielten Ergebnisse kann keine klare Empfehlung für eine der beide Reparaturheuristiken gegeben werden. Betrachtet man die Komplexität der Reparaturheuristiken, so ist mit der CostliestGreedy-Reparaturheuristik aufgrund des einfachen Vorgehens der Vorzug zu geben.

Eine allgemeingültige Aussage zur Leistungsfähigkeit der untersuchten GAs mit den hier vorgeschlagenen Reparaturheuristiken ist derzeit nicht möglich, da vergleichbare Ansätze in der Literatur bisher nicht veröffentlicht wurden.

6.2 Kommunikationsnetzwerkplanung mit Budgetvorgaben unter Verwendung unterschiedlicher Technologieoptionen

Während in Abschnitt 6.1 die Kommunikationsnetzwerkplanung unter Budgetvorgaben lediglich für identische Verbindungszuverlässigkeiten ohne Technologieoptionen untersucht wurde, wird im Folgenden eine Reparaturheuristik für die Planung mit verschiedenen Technologieoptionen vorgestellt. Diese Reparaturheuristik überführt ungültige Lösungen, deren Kosten die Budgetvorgabe C_B nicht erfüllen, in gültige Lösungen (mit $C(G_N) \leq C_B$).
Aufbauend auf der formalen Problembeschreibung in Abschnitt 4.2 lässt sich das untersuchte Planungsproblem wie folgt beschreiben:

$$\begin{aligned} f(y) = R_{All}(G_N) &\rightarrow \max \\ C(G_N) &\leq C_B \\ \text{mit } C(G_N) &= \sum_{e_{v_i,v_j} \in E_N} c(l_k(e_{v_i,v_j})) \end{aligned} \tag{6.3}$$

Mit Hilfe des Planungsverfahrens soll die Netzwerktopologie (G_N) gefunden werden, deren All-Terminal-Zuverlässigkeit maximal ist und deren Kosten (C_N) eine Kostenschranke/Budget (C_B) nicht übersteigen. Pro Verbindung stehen dabei unterschiedliche Technologieoptionen $l_1, \ldots, l_{k_{max}}$ zur Verfügung, die sich hinsichtlich ihrer Installationskosten $c(l_k(e_{v_i,v_j}))$ und der Verbindungszuverlässigkeit $r(l_k(e_{v_i,v_j}))$ unterscheiden.

6.2.1 STC3 – Eine Reparaturheuristik für die Kommunikationsnetzwerkplanung mit Budgetvorgaben unter Verwendung unterschiedlicher Technologieoptionen

In den Abschnitten 5.1.2, 5.2.1 und 6.1.1 wurden bereits verschiedene Reparaturheuristiken für die Kommunikationsnetzwerkplanung vorgestellt. Aufgabe des jeweiligen Reparaturverfahrens ist es, eine ungültige Lösung, welche eine für das Entscheidungsproblem aufgestellte Nebenbedingung verletzt, in eine gültige (im Sinne der Nebenbedingung) Lösung zu überführen. Mit diesem Abschnitt wird die Reparaturheuristik STC3 (Spanning Tree Counting 3) vorgeschlagen, die es ermöglicht, bei der Kommunikationsnetzwerkplanung mit Budgetvorgaben und unter Verwendung beliebiger Verbindungszuverlässigkeiten, Lösungen, welche eine aufgestellte Budgetgrenze überschreiten, durch Reparatur in eine Lösung mit Kosten $C(G_N) \leq C_B$ zu überführen.
Wie bereits in Abschnitt 6.1.1 verwendet die im Folgenden vorgeschlagene STC3-Reparaturheuristik die Anzahl der Spannbäume eines Graphen als Basis für den Reparaturprozess. Der Ablauf der STC3-Reparaturheuristik wird in Algorithmus 6.3 dargestellt.

Als Eingabe erhält die STC3-Reparaturheuristik den Graphen G_N, welcher eine unzulässige Lösung mit $C(G_N) > C_B$ darstellt, und das Budget C_B. Als Ausgabe wird die reparierte Lösung zurückgegeben.
Mittels #SpanningTrees(G_N) wird die Anzahl der Spannbäume in G_N bestimmt. Anschließend wird für jede Kante $e_{v_i,v_j} \in E_N$ deren Einfluss auf die Zuverlässigkeit des Graphen ermittelt. Hierfür wird jede Kante temporär aus G_N entfernt und die Differenz $|STC|_\triangle$ in der Anzahl der Spannbäume, die daraus resultiert, berechnet. Diese Differenz wird ins Verhältnis zu den Kosten der aktuell gewählten Technologieoption der Kante gesetzt und abgespeichert (Zeile 3-7). Nach der Bewertung der Kanten werden die Technologieoptionen für die Kanten mit dem höchsten *ratio* sukzessive verringert oder die Kanten komplett aus G_N entfernt. Die STC3-Heuristik versucht dabei, zunächst die Technologieoptionen $l_k(e_{v_i,v_j})$ einer Kante anzupassen, indem für eine Kante e_{v_i,v_j} die nächstgünstigere Technologieoption mit einer geringeren Zuverlässigkeit ausgewählt wird. Ist es nicht möglich, durch die Veränderung der Technologieoption einer Kante das gewünschte Budget C_B zu erreichen, so wird die Kante komplett aus G_N gelöscht. Die Reparaturheuristik stoppt, sobald $C(G_N) > C_B$ oder sämtliche Kanten verarbeitet wurden.

Algorithmus 6.3 STC3-Reparaturheuristik

Eingabe: G_N, C_B
Ausgabe: G_N
1: $L = \emptyset$
2: $|STC| = \#\text{SpanningTrees}(G_N)$
3: **for all** $e_{v_i,v_j} \in E_N$ **do**
4: $\quad |STC|_{e_{v_i,v_j}} = \#\text{SpanningTrees}(G_N \setminus \{e_{v_i,v_j}\})$
5: $\quad |STC|_\triangle = |STC| - |STC|_{e_{v_i,v_j}}$
6: $\quad ratio_{e_{v_i,v_j}} = \frac{c(l_k(e_{v_i,v_j}))}{|STC|_\triangle}$
7: $\quad L \cup \{e_{v,v_ij}\}$
8: **end for**
9: Sortiere L absteigend nach $ratio_{e_{v_i,v_j}}$
10: $Counter = 0$
11: **while** $C(G_N) > C_B$ und $Counter < |L|$ **do**
12: $\quad e_{work} = e_{Counter}$ mit $e_{Counter} \in L$
13: $\quad$ **while** $l_k(e_{work}) > 1$ und $C(G_N) > C_B$ **do**
14: $\quad\quad l_{k-1}(e_{work})$
15: $\quad$ **end while**
16: $\quad$ **if** $C(G_N) > C_B$ und $l_k(e_{work}) = 1$ **then**
17: $\quad\quad E_N \setminus e_{work}$
18: $\quad$ **end if**
19: $\quad L \setminus \{e_{work}\}$
20: **end while**

6.2.2 SAGA - Ein evolutionärer Planungsansatz mit einer adaptiven Budgetschranke

Mit STC3 wurde im vorangegangen Abschnitt eine Reparaturheuristik eingeführt, die es ermöglicht, ungültige Lösungen unter Berücksichtigung der gestellten Budgetbedingung zu reparieren. Mit dem Self-Adapting-GA (SAGA) wird jetzt ein GA für die Planung von Kommunikationsnetzwerken unter Budgetvorgaben bei Verwendung unterschiedlicher Technologieoptionen vorgestellt. Wie zuvor ausgeführt, stehen dabei für jede Verbindung verschiedene Technologieoptionen zur Auswahl, die sich hinsichtlich der Zuverlässigkeit und den Kosten unterscheiden. Werden durch den GA ungültige Lösungen generiert, so werden diese an die STC3-Reparaturheuristik übergeben, welche die Lösung so verändert, dass diese der Budget-Nebenbedingung genügt.
Für SAGA wird jede Lösung als Vektor g mit der Länge $|E|$ kodiert. Jedes Element von g repräsentiert dabei eine Kante mit einer gewählten Technologieoption l_k $(1 \leq k \leq k_{max})$ des Eingabegraphen G. Wenn eine Kante aus G in einer Lösung nicht verwendet wird, so ist $g_i = 0$. Abbildung 6.5 zeigt die Kodierung einer Lösung[62] für SAGA. Wenn

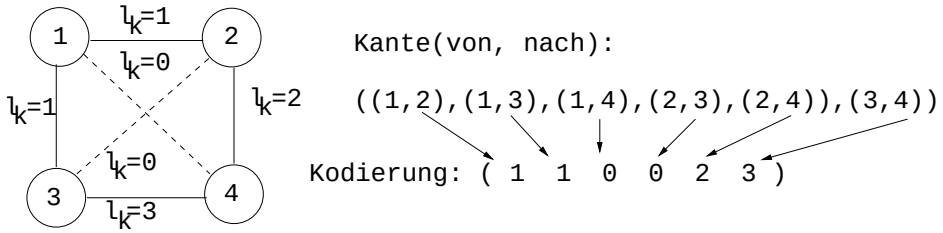


Abbildung 6.5: Kodierung einer Lösung für SAGA

durch die Operatoren des genetischen Algorithmus (Initialisierung, Mutation, Rekombination) eine ungültige Lösung generiert wird, so wird der Vektor g durch die STC3-Reparaturheuristik verändert.
Beim Entwurf von SAGA wurde in experimentellen Untersuchungen festgestellt, dass der Einsatz eines einfachen GAs unter Verwendung der STC3-Reparaturheuristik dazu führt, dass der GA frühzeitig mit Lösungen geringer Güte (d.h. sehr geringen All-Terminal-Zuverlässigkeiten) konvergiert. Ursache hierfür ist die initiale Population[63], die häufig eine hohe Anzahl an Lösungen enthält, welche die Zuverlässigkeitsnebenbedingung verletzen. Diese wurden durch die STC3-Reparaturheuristik in zulässige Lösungen überführt. Dabei wurde jedoch häufig der Zusammenhang der Netzwerke, welche die Lösungen repräsentieren, zerstört, so dass die All-Terminal-Zuverlässigkeit $R_{All} = 0$ betrug. Um dieser Entwicklung entgegenzuwirken, wurde der GA um einen sich während eines Optimierungslaufs selbstständig anpassenden Budget-Parameter erweitert.
Die Grundidee für den Einsatz des Budget-Parameters ist, dass ähnlich der Simulated Annealing-Metaheuristik zu Beginn der Optimierung ungültige Lösungen in der Population verbleiben können und mit Fortschreiten der Zeit diese Relaxation der Nebenbedingung eingeengt wird und damit über die Zeit hinweg immer weniger ungültige

[62] Verbindungen in G, die für die abgebildete Lösung nicht verwendet werden, sind gestrichelt dargestellt.
[63] Es wurden Experimente mit unterschiedlichen Initialisierungsverfahren durchgeführt.

Lösungen in einer Population akzeptiert werden. Dieses Konzept wurde mit dem Self-Adapting-GA (SAGA) umgesetzt. Den Ablauf von SAGA stellt Algorithmus 6.4 dar. In einem ersten Schritt erstellt SAGA die initiale Population P_0. Alle Individuen in

Algorithmus 6.4 Self-Adapting-GA (SAGA)

Eingabe: C_B, G

1: P_0 = initPop
2: bewerte(P_0)
3: **if** $0,75 \cdot C_{AVG}(P_0) > C_B$ **then**
4: $\quad C_B^{Relax} = 0,75 \cdot C_{AVG}(P_0)$
5: **else**
6: $\quad C_B^{Relax} = C_B$
7: **end if**
8: bewerte(P_0, C_B^{Relax})
9: $t = 1$
10: erstelle P_t aus P_0
11: **while** Abbruchkriterium nicht erfüllt **do**
12: $\quad$ **if** $C_B^{Relax} > C_B$ und $R_{All}^{min}(P_t) > 0$ und $R_{All}^{AVG}(P_t) > 0,5$ **then**
13: $\quad\quad$ **if** $C_B^{Relax} \cdot 0,9 > C_B$ **then**
14: $\quad\quad\quad C_B^{Relax} = C_B^{Relax} \cdot 0,9$
15: $\quad\quad$ **else**
16: $\quad\quad\quad C_B^{Relax} = C_B$
17: $\quad\quad$ **end if**
18: $\quad$ **end if**
19: $\quad$ bewerte(P_t,C_B)
20: $\quad$ erstelle P_{t+1} aus P_t
21: $\quad t = t + 1$
22: **end while**

P_0 werden hinsichtlich ihrer All-Terminal-Zuverlässigkeit und ihrer Kosten bewertet. Lösungen mit $C(G_N) > C_B$ werden dabei nicht an die Reparaturheuristik übergeben, sondern verbleiben unverändert in der Population. Für die Steuerung der Akzeptanz ungültiger Lösungen (im Sinne der Budget-Nebenbedingung) wird der Parameter C_B^{Relax} eingeführt. Liegen die durchschnittlichen Kosten $C_{AVG}(P_0)$ der Lösungen der initialen Population P_0 über dem Budget C_B, so wird für die initiale Population der Parameter C_B^{Relax} auf den Wert $0,75 \cdot C_{AVG}(P_0)$ festgelegt. Liegt der Mittelwert der Kosten ($C_{AVG}(P_0)$) unterhalb des Budgets C_B, so gilt $C_B^{Relax} = C_B$.
Nach der Bestimmung von C_B^{Relax} wird die initiale Population P_0 unter Verwendung des Budget-Parameters C_B^{Relax} bewertet (Zeile 8). Dabei werden Lösungen, mit $C(G_N) > C_B^{Relax}$ mit Hilfe der STC3-Reparaturheuristik in Lösungen mit $C(G_N) \leq C_B^{Relax}$ überführt.
In der Schleife (Zeile 11-22) wird in jeder Generation geprüft, ob eine Anpassung von C_B^{Relax} erfolgen muss. Hierfür wird von SAGA die minimale ($R_{All}^{min}(P_t)$) und mittlere All-Terminal-Zuverlässigkeit ($R_{All}^{AVG}(P_t)$) der Lösungen in der Population P_t betrachtet. Nur wenn die minimale Fitness $R_{All}^{min}(P_t) > 0$ (d.h. alle Lösungen der Population sind

zusammenhängend) und die mittlere All-Terminal-Zuverlässigkeit $R_{All}^{AVG}(P_t) > 0,5$ sind, wird C_B^{Relax} für die nächste Generation an C_B angenähert (Zeile 12-18). Bei jeder Anpassung wird sichergestellt, dass $C_B^{Relax} > C_B$ ist. Auf diese Weise nähert sich C_B^{Relax} über die Generationen hinweg immer mehr an das tatsächliche Budget C_B an und akzeptiert immer weniger Lösungen, die die aufgestellte Kostennebenbedingung nicht erfüllen.
Mit Algorithmus 6.4 wird eine vereinfachte Form des im Rahmen dieser Arbeit getesteten GAs gezeigt. Zusätzlich zur hier dargestellten Form führt der Algorithmus mit jeder Anpassung von C_B^{Relax} eine Anpassung bei der All-Terminal-Zuverlässigkeitsbewertung durch. Zur Berechnung wird eine einfache Monte-Carlo-Simulation eingesetzt. Während zu Beginn lediglich eine geringe Anzahl von Stichproben für die Monte-Carlo-Simulation verwendet wird, nimmt der Stichprobenumfang mit jeder Anpassung von C_B^{Relax} zu. Auf diese Weise wird die Genauigkeit der Schätzung mit einer Annäherung von C_B^{Relax} an C_B erhöht. Ebenfalls vereinfacht dargestellt wurde in Algorithmus 6.4 der Prozess der Erstellung der neuen Generation (erstelle P_t aus P_0). Details hierzu erläutert der nächste Abschnitt.

6.2.2.1 Experimentelles Design

In einer empirischen Untersuchung wurde SAGA für die Netzwerke deutsch15, deutsch20, deutsch25, deutsch30 und türkei19 analysiert. Eine Beschreibung der Testinstanzen gibt Abschnitt 5.2.3 auf S. 109. Die Budgets für die Netzwerke wurden in Anlehnung an die Kosten der besten in Abschnitt 5.2.3.2 vorgestellten Ergebnisse gewählt.
Der für die Experimente verwendete SAGA nutzt einen Uniform-Crossover-Operator (siehe S. 47) und setzt eine Flip-Mutation (siehe S. 48) ein. Die initiale Population wurde aus zufällig generierten Lösungen erstellt. Das Initialisierungsverfahren erzeugt hierfür für jede Lösung einen zufälligen Spannbaum in G. Für jede ausgewählte Verbindung wurde die Technologieoption gleichverteilt aus der Menge der zur Verfügung stehenden Technologieoptionen bestimmt. Das Initialisierungsverfahren stellt dabei sicher, dass jede initiale Lösung zusammenhängend ist.
Für die Experimente wurden die folgenden GA-Parameter verwendet: Ein Steady-State-GA mit sich zu 50 % überlappenden Populationen. Die Crossover-Wahrscheinlichkeit beträgt $p_{cross} = 0,9$, die Mutationswahrscheinlichkeit beträgt $p_{mut} = 0,01$ und die Populationsgröße wurde auf $pop = 100$ festgelegt. Ein GA-Durchlauf wurde nach 2000 Generationen oder wenn in den letzten 50 Generationen bei $C_B^{Relax} = C_B$ keine Verbesserung der besten Fitness stattfand, angehalten.
Da aufgrund der Vielzahl von All-Terminal-Zuverlässigkeitsbewertungen der Einsatz eines exakten Verfahrens zur Berechnung wegen der hohen Laufzeiten nicht praktikabel ist, kommt für den SAGA eine einfache Monte-Carlo-Simulation zum Einsatz. Da diese als stochastisches Verfahren lediglich einen Schätzwert für die All-Terminal-Zuverlässigkeit ermittelt, wird beim Vergleich zweier Lösungen hinsichtlicher ihrer Fitness ein angenäherter 95 %-Vertrauensbereich verwendet (vgl. Anhang B). Zwei Lösungen werden dabei nicht als identisch betrachtet, wenn die All-Terminal-Zuverlässigkeiten beider Lösungen auf dem 5 %-Niveau einen statistisch signifikanten Unterschied[64] aufweisen.

[64] Details zum Vergleich zweier mittels einfacher Monte-Carlo-Simulation ermittelter All-Terminal-Zuverlässigkeiten bietet Anhang B.

Tabelle 6.5: Parameter für STC-GA und Strafterm-GA

GA	Steady-State-GA
Crossover-Operator	Uniform-Crossover, $p_{cross} = 0,9$
Mutations-Operator	Flip-Mutation, $p_{mut} = 0,01$
Initialisierung von P_0	zufallsbasiert mit einem Spannbaum in G $p_{init}^{opt} = 1/l_{k_{max}}$
Bestimmung von R_{All}	Einfache Monte-Carlo-Simulation
pop	100
Anzahl GA-Läufe	10

Tabelle 6.5 fasst die für die Experimente verwendeten Parameter zusammen. Für jedes Netzwerk wurden zehn voneinander unabhängige Läufe durchgeführt. Der SAGA wurde in der GALib [188] unter Linux implementiert und getestet. Die Experimente wurden auf einem P4-2-GHz System durchgeführt.

6.2.2.2 Experimentelle Ergebnisse und Auswertung

Mit Tabelle 6.6 werden die mittels SAGA erzielten Ergebnisse dargestellt. Die Tabelle zeigt für jedes untersuchte Netzwerk das verwendete Budget C_B, die maximale in allen SAGA-Durchläufen erreichte All-Terminal-Zuverlässigkeit R_{All}^{opt} sowie den Mittelwert (μ) und die Standardabweichung (σ) für R_{All}^{max} gemittelt über sämtliche GA-Durchläufe. Weiterhin werden mit der Tabelle die Kosten $C_{R_{All}^{max}}$ der Lösung mit der höchsten All-Terminal-Zuverlässigkeit (R_{All}^{opt}) sowie der Mittelwert (μ) der Kosten der Lösungen mit den jeweils besten All-Terminal-Zuverlässigkeiten in jedem SAGA-Durchlauf und die zugehörige Standardabweichung (σ) gezeigt. Für eine Analyse des Laufzeitverhaltens wird für jedes Netzwerk die gemittelte Anzahl an Fitnessbewertungen $\#Eval$ pro GA-Durchlauf sowie die mittlere Laufzeit t_{conv} (in Sekunden) je SAGA-Durchlauf angegeben.
Neben der All-Terminal-Zuverlässigkeit, welche es zu maximieren gilt, zeigt die Tabelle die Kosten der jeweils besten Lösungen. Auch wenn diese bei der Kommunikationsnetzwerkplanung mit Budgetvorgaben unter Verwendung unterschiedlicher Technologieoptionen nur als Nebenbedingung betrachtet werden, so ist davon auszugehen, dass ein Netzwerkplaner eine Lösung y_A einer Lösung y_B vorzieht, wenn die Lösung y_A die gleiche All-Terminal-Zuverlässigkeit wie Lösung y_B besitzt und die Kosten von y_A kleiner als die Kosten von y_B sind.
Wie die dargestellten Ergebnisse zeigen, konnte für alle hier untersuchten Netzwerke eine Lösung mit einer hohen All-Terminal-Zuverlässigkeit unter Einhaltung der Budgetschranke gefunden werden. Betrachtet man den Mittelwert und die Standardabweichung für R_{All}^{max} (der All-Terminal-Zuverlässigkeit der jeweils besten Lösungen eines SAGA-Durchlaufs), so ist festzuhalten, dass die Lösungen nur eine sehr geringe Streuung aufweisen und damit in allen zehn Durchläufen sehr gute und sehr ähnliche Lösungen gefunden wurden. Diese Beobachtung wird mit einer Analyse der Kosten der besten Lösungen ($C_{R_{All}^{max}}$) bestätigt. Auch hier ist eine sehr geringe Abweichung vom Mittelwert über alle besten Lösungen festzustellen.

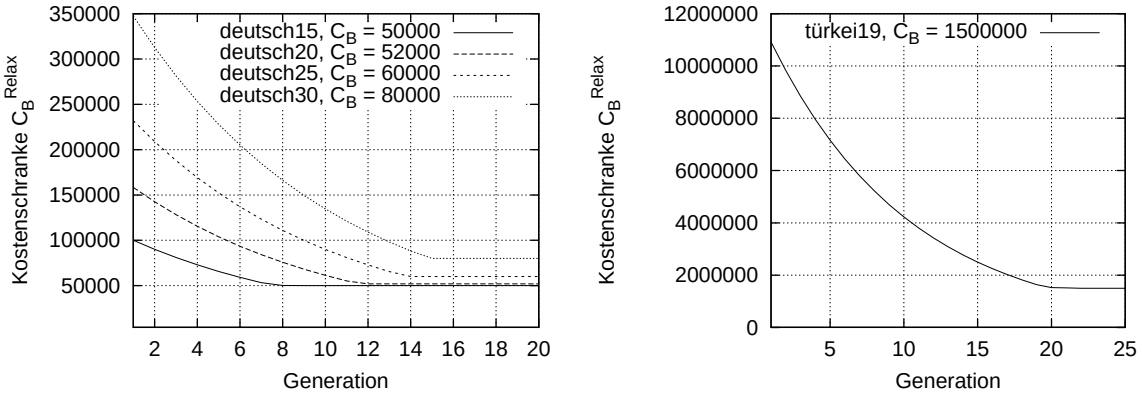
Tabelle 6.6: Ergebnisse für die Netzwerke deutsch15, deutsch20, deutsch25, deutsch30 sowie türkei19 bei Verwendung des SAGA

Netzwerk	C_B	R_{All}^{opt}	R_{All}^{max}		$C_{R_{All}^{opt}}$	$C_{R_{All}^{max}}$		#Eval	t_{conv}
			μ	σ		μ	σ		
deutsch15	50.000	0,984	0,984	0,00096	49.986	49.478	205,8	11.453	3.396
deutsch20	52.000	0,980	0,979	0,00160	51.874	51.580,6	109,3	14.043	5.160
deutsch25	60.000	0,989	0,987	0,00182	59.792	58.652,2	496,9	13.774	7.518
deutsch30	80.000	0,994	0,993	0,00140	79.862	77.576	336,0	13.667	9.900
türkei19	1.500.000	0,9882	0,9871	0,00085	1.496.520	1.471.730	10.311,5	9.898	2.160

Eine Analyse der durchgeführten Fitnessbewertungen zeigt, dass diese für die Netzwerke deutsch20, deutsch25 und deutsch30 sehr nah beieinander liegen. Gleichzeitig nimmt hier mit der Größe des untersuchten Netzwerkes die Laufzeit[65] eines SAGA-Durchlaufs zu. Dies ist darauf zurückzuführen, dass mit zunehmender Größe der Netzwerke häufiger Netzwerke mit $C(G_N) > C_B$ erzeugt werden und dadurch die STC3-Reparaturheuristik öfter aufgerufen wird.
Das Netzwerk türkei19, welches sich hinsichtlich der Verbindungszuverlässigkeiten und den Verbindungskosten von den „deutsch“-Netzwerken unterscheidet, braucht für das hier getestete Budget weniger Fitnessbewertungen und eine geringere Laufzeit als die ähnlich komplexen Netzwerke deutsch20 und deutsch30. Ursächlich lässt sich dies damit begründen, dass der SAGA hier früher Lösungen mit einer sehr hohen All-Terminal-Zuverlässigkeit findet und früher konvergiert.
In der Literatur existieren derzeit noch keine Arbeiten, die Verfahren für die Kommunikationsnetzwerkplanung mit Budgetvorgaben unter Verwendung unterschiedlicher Technologieoptionen vorstellen. Daher werden die hier erzielten Ergebnisse in Relation zu den in Abschnitt 5.2.3 gezeigten Ergebnissen gesetzt.
Ein Vergleich der besten hier gefundenen Lösungen zu den in Abschnitt 5.2.3 ermittelten Ergebnissen zeigt, dass die besten hier erzielten Lösungen für die Netzwerke deutsch15, deutsch20, deutsch25 und deutsch30 höhere Kosten aufweisen, jedoch auch eine höhere All-Terminal-Zuverlässigkeit besitzen. Ursache hierfür ist das gewählte Budget, dass in Anlehnung an Abschnitt 5.2.3 festgelegt wurde. SAGA wurde so entworfen, dass eine möglichst hohe Ausschöpfung des Budgets zur Maximierung der All-Terminal-Zuverlässigkeit erfolgt. Die hier verwendeten Budgets liegen häufig mehrere 1000 Geldeinheiten von den Kosten der besten Lösung entfernt, so dass die billigsten in Abschnitt 5.2.3 gefundenen Netzwerke aufgrund des Strebens nach maximaler Budgetausnutzung nicht als beste Lösung erzeugt werden.
Für das Netzwerk türkei19 wurde das Budget mit $C_B = 1.500.000$ so gewählt, dass dieses kleiner als die Kosten der besten in Abschnitt 5.2.3 für dieses Problem gefundenen Lösung ist. Die Ergebnisse zeigen, dass SAGA unter Verwendung dieses Budgets keine Lösung findet, deren All-Terminal-Zuverlässigkeit besser als die in Abschnitt 5.2.3 verwendete Zuverlässigkeitsnebenbedingung $R_0 = 0,95$ ist, jedoch sehr nahe an der besten in Abschnitt 5.2.3 gefundenen Lösung liegt.

[65] Der Aufwand zur Berechnung der All-Terminal-Zuverlässigkeit einer Lösung ist identisch, da für alle Netzwerke der gleiche Stichprobenumfang verwendet wurde.

Anhand von Abbildung 6.6 erfolgt eine Analyse des in SAGA eingeführten adaptiven Budget-Parameters C_B^{Relax}. In der Abbildung wird jeweils der Wert für C_B^{Relax} je Generation gemittelt über alle zehn SAGA-Durchläufe dargestellt. Die Kurven zeigen für die hier untersuchten Probleme jeweils die Anpassung von C_B^{Relax} hin zu C_B über die Generationen hinweg.
Man sieht, dass sich C_B^{Relax} im Laufe der Optimierung immer mehr an das tatsächliche Budget C_B annähert und dieses nach spätestens 20 Generationen erreicht. Für die

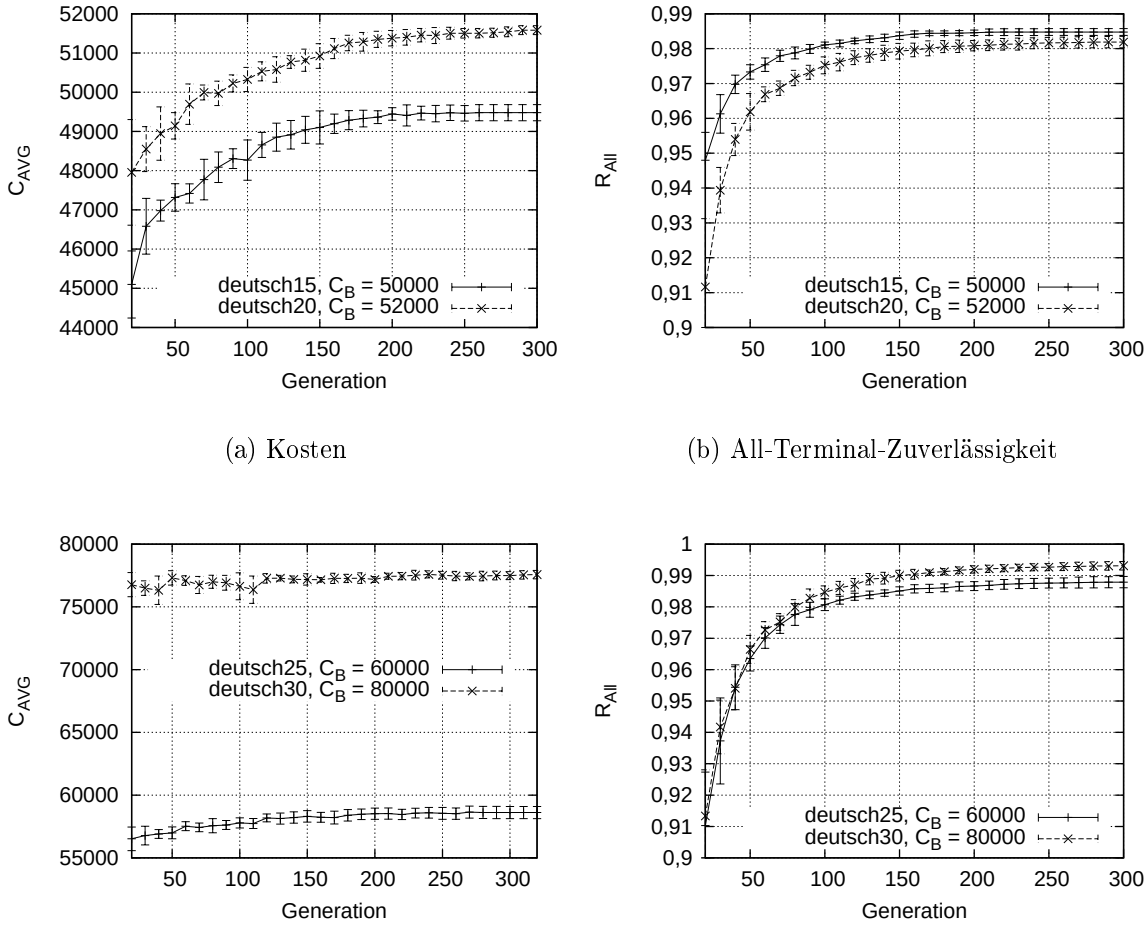


(a) Netzwerke deutsch15, deutsch20, deutsch25 und deutsch30

(b) Netzwerk türkei19

Abbildung 6.6: Anpassung der Budgetschranke C_B^{Relax} an C_B über die Generationen hinweg für die Netzwerke deutsch15, deutsch20, deutsch25, deutsch30 und türkei19

Netzwerke deutsch15, deutsch20, deutsch25 und deutsch30 ist zu erkennen, dass die Adaption $C_B^{Relax} = C_B$ mit zunehmender Problemgröße im zeitlichen Verlauf später erfolgt.
Abbildung 6.7 zeigt den Performancevergleich mittels durchschnittlicher All-Terminal-Zuverlässigkeit (Abbildung 6.7(b) und Abbildung 6.7(d)) und den durchschnittlichen Kosten (Abbildung 6.7(a) und Abbildung 6.7(c)) in der Population für jede Generation. Da in den Generationen mit $C_B^{Relax} > C_B$ zunächst noch eine All-Terminal-Zuverlässigkeitsbewertung mit weniger Stichproben erfolgt und hier das während der Bewertung verwendete Budget größer dem geforderten Budget ist, zeigen die Abbildung 6.7 und die Abbildung 6.8 die Verläufe erst ab dem Zeitpunkt $C_B^{Relax} = C_B$. Die abgetragenen Werte je Generation entsprechen dabei jeweils den über alle zehn SAGA-Durchläufe gemittelten Ergebnissen.
Die dargestellten Verläufe zeigen, dass sich die durchschnittlichen Kosten in der Population über die Zeit hinweg dem Budget C_B annähern. Gleichzeitig nimmt die durchschnittliche All-Terminal-Zuverlässigkeit über die Generationen hinweg zu. SAGA konvergiert nach zirka 300 Generationen für alle vier dargestellten Netzwerke, da keine Verbesserung mehr erreichbar ist. Die für den jeweiligen Verlauf abgetragene Standardabweichung zeigt, dass die Qualität der Lösungen über sämtliche SAGA-Läufe sehr homogen ist und eine ähnlich gute All-Terminal-Zuverlässigkeit besitzt.



(a) Kosten

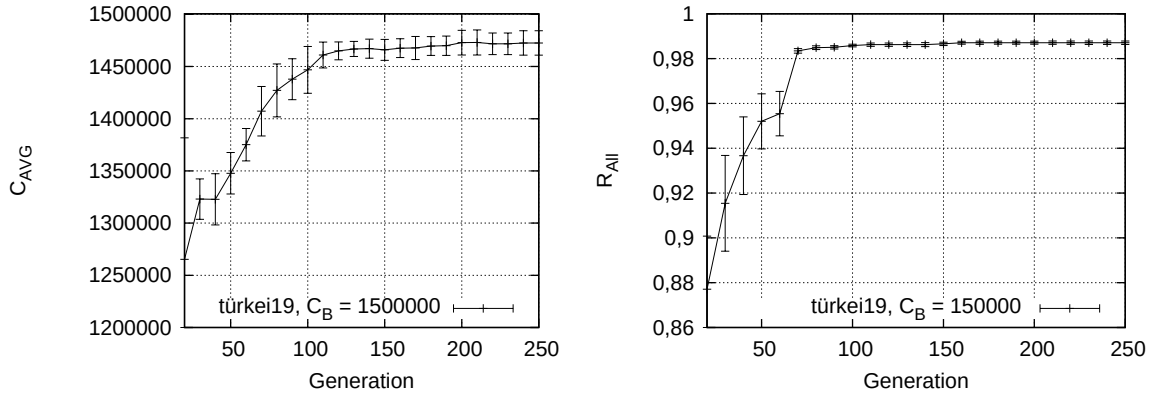
(b) All-Terminal-Zuverlässigkeit

(c) Kosten

(d) All-Terminal-Zuverlässigkeit

Abbildung 6.7: Performancevergleich anhand der durchschnittlichen All-Terminal-Zuverlässigkeit und der mittleren Kosten je Generation für die Netzwerke deutsch15, deutsch20, deutsch25 und deutsch30

Abbildung 6.8 zeigt die Performance des SAGAs für das Netzwerk türkei19. Die Abbildung zeigt die durchschnittliche All-Terminal-Zuverlässigkeit (Abbildung 6.8(b)) und die durchschnittlichen Kosten (Abbildung 6.8(a)) aller Lösungen einer Population über die Generationen hinweg. Die Werte je Generation entsprechen dabei jeweils den über alle zehn SAGA-Durchläufe gemittelten Ergebnissen.



(a) Kosten

(b) All-Terminal-Zuverlässigkeit

Abbildung 6.8: Performancevergleich anhand der All-Terminal-Zuverlässigkeit und der mittleren Kosten für das Netzwerk türkei19

Wie bereits bei der Auswertung der Ergebnisse aus Tabelle 6.6 dargelegt, ist zu erkennen, dass SAGA für das Netzwerk türkei19 deutlich früher als für die Netzwerke deutsch15, deutsch20, deutsch25 und deutsch30 mit einer guten Lösung konvergiert und dadurch weniger Fitnessbewertungen benötigt und eine geringere Laufzeit aufweist. Der dargestellte Verlauf der durchschnittlichen Kosten in Abbildung 6.8(a) zeigt, dass nach 150 Generationen eine Annäherung an das Budget erfolgt. Für die Netzwerke deutsch15, deutsch20, deutsch25 und deutsch30 ist diese Annäherung erst ab Generation 200 festzustellen (vgl. Abbildung 6.7(a) und Abbildung 6.7(c)).
Zusammenfassend lässt sich für den hier vorgeschlagenen SAGA anhand der durchgeführten empirischen Untersuchung feststellen, dass das Verfahren unter Einsatz der STC3-Reparaturheuristik in der Lage ist, zuverlässige Kommunikationsnetzwerke unter Verwendung eines Budgets zu planen. Für den eingeführten adaptiven Budgetparameter C_B^{Relax} wurde gezeigt, dass dieser durch eine Lockerung des Budgets zu Beginn eines SAGA-Durchlaufs die Suche gezielt in Regionen mit einer hohen Fitness lenkt und der GA letztendlich mit Lösungen hoher Güte konvergiert.
Aufgrund des Fehlens von vergleichbaren Verfahren in der Literatur ist abschließend keine allgemeingültige Aussage über die Qualität der mittels SAGA gefundenen Lösungen möglich.

6.3 Zusammenfassung

Mit diesem Kapitel wurden drei Reparaturheuristiken für die Planung von zuverlässigen Kommunikationsnetzwerken unter Budgetvorgaben vorgeschlagen und deren Einsatz zur Reparatur invalider Lösungen in einem GA empirisch untersucht. Während die vorgeschlagene STC2- und CostliestGreedy-Reparaturheuristik nur für die Planung mit identischen Verbindungszuverlässigkeiten ohne Technologieoptionen einsetzbar sind, eignet sich die vorgeschlagene STC3-Reparaturheuristik, welche auf dem Konzept von STC2 aufbaut, für die Planung mit unterschiedlichen Technologieoptionen.
In einer empirischen Untersuchung wurden die Reparaturheuristiken CostliestGreedy und STC2 mit einem Steady-State-GA für Netzwerke mit unterschiedlicher Komplexität untersucht. Die dabei gewonnenen Ergebnisse zeigen, dass beide Reparaturheuristiken zuverlässige Kommunikationsnetze unter Einhaltung des gegebenen Budgets finden. Es ist jedoch keine klare Überlegenheit einer der beiden Reparaturheuristiken zu erkennen.
Mit SAGA wurde ein GA mit einer adaptiven Budgetschranke für die Kommunikationsnetzwerkplanung mit Budgetvorgaben unter Verwendung unterschiedlicher Technologieoptionen vorgeschlagen. Für den hier vorgestellten SAGA konnte gezeigt werden, dass ein GA unter Verwendung der STC3-Reparaturheuristik und dem Einsatz eines adaptiven Budget-Parameters in der Lage ist, zuverlässige Lösungen innerhalb der getesteten Budgets zu finden.
Da in der Literatur für die Planung von Kommunikationsnetzwerken mit Budgetvorgaben derzeit keine Verfahren vorgeschlagen werden, die die hier untersuchten Problemstellungen lösen, ist ein Vergleich der vorgeschlagenen Verfahren zu anderen Arbeiten nicht möglich.

7 Kommunikationsnetzwerkplanung als multikriterielles Planungsproblem mit den Zielen Kosten und Zuverlässigkeit

Erfolgte bisher eine Betrachtung des im Rahmen der Arbeit untersuchten Entscheidungsproblems als Planungsproblem mit einer Zielgröße und Nebenbedingungen, so wird mit diesem Abschnitt eine Betrachtung als multikriterielles Entscheidungsproblem mit den Zielen Zuverlässigkeit und Kosten vorgenommen. Zu Beginn des Kapitels erfolgt eine formale Darstellung des Zielsystems für die zu untersuchenden Verfahren. Im Anschluss werden die notwendigen Kennzahlen eingeführt, die einen Vergleich zwischen unterschiedlichen Pareto-Fronten und damit auch einen Vergleich der Lösungsgüte unterschiedlicher Planungsverfahren ermöglichen. Anschließend werden die mit Hilfe einer NSGA-II-Implementierung erzielten Lösungsfronten für das betrachtete Problem präsentiert. Diese Pareto-Fronten werden im weiteren Verlauf als Vergleichslösungen für die im Rahmen der Arbeit vorgeschlagenen multikriteriellen Planungsansätze verwendet. Weiterhin werden mit diesem Kapitel zwei modifizierte NSGA-II-Planungsverfahren, welche um eine lokale Suche erweitert wurden, für die Planung von zuverlässigen und ökonomischen Kommunikationsnetzwerken vorgeschlagen. Beide Verfahren werden in einer experimentellen Studie an ausgewählten Netzwerken untersucht und die Ergebnisse untereinander und mit denen des unmodifizierten NSGA-II verglichen.

7.1 Ein Zielsystem für die Kommunikationsnetzwerkplanung als multikriterielles Planungsproblem mit den Zielen Kosten und Zuverlässigkeit

Ausgehend von den in Abschnitt 3.3 dargestellten Grundlagen der MKOP erfolgt zunächst eine Beschreibung des Zielsystems für die multikriterielle Kommunikationsnetzwerkplanung mit den Zielen Kosten und Zuverlässigkeit.
Betrachtet man die Planungsziele Kosten und Zuverlässigkeit beim Netzwerktopologieentwurf parallel, so ergibt sich als Zielsystem für ein multikriterielles Planungsverfahren:

$$\begin{aligned} f(y) = f(G_N) &= \{f_1(G_N), f_2(G_N)\} \\ &\text{mit} \\ f_1(G_N) &= R_{All}(G_N) \rightarrow \max \\ f_2(G_N) &= C(G_N) \rightarrow \min \end{aligned} \tag{7.1}$$

Die Zielfunktion $f(y)$ setzt sich aus den Zielen Maximierung der All-Terminal-Zuverlässigkeit $R_{All}(G_N)$ und Minimierung der Netzwerkkosten $C(G_N)$ zusammen. Sind keine Präferenzen des Entscheiders für die beiden Zielkriterien bekannt, so generiert ein

Lösungsverfahren für dieses MKOP eine Menge von zueinander pareto-optimalen Lösungen. Als zusätzliche Nebenbedingung wird festgelegt, dass jede Lösung für das Problem 2-fach-kantenzusammenhängend[66] sein muss. Diese Festlegung erfolgt zum einen, um eine Vergleichbarkeit zu den in [63, 14] dargestellten Ergebnissen zu ermöglichen und zum anderen, um sicherzustellen, dass die erzeugten Lösungen ein Mindestmaß an Ausfallsicherheit besitzen.

7.2 Methodische Ansätze zur Bewertung und zum Vergleich von Lösungen multikriterieller kombinatorischer Optimierungsprobleme

Wie in Abschnitt 3.3 dargestellt, generieren multikriterielle Optimierungsverfahren eine Menge zueinander pareto-optimaler Lösungen, welche als Pareto-Front bezeichnet werden. Während es in Kapitel 5 und Kapitel 6 beim Vergleich zweier unterschiedlicher Lösungen ausreichte, deren Zielfunktionswerte zu vergleichen, so erfordert die Analyse der Lösungen von multikriteriellen Optimierungsverfahren weitergehende Kennzahlen. Dieser Abschnitt gibt zunächst eine Einführung in unterschiedliche Kennzahlen für den Vergleich von Pareto-Fronten. Mit Hilfe der Kennzahlen wird dabei ein Vergleich zwischen einer generierten Pareto-Front $\mathcal{Y}_{known}$ und der wahren/optimalen Pareto-Front[67] $\mathcal{Y}_{true}$ ermöglicht. Zielsetzung eines jeden multikriteriellen Verfahrens ist es, die wahre Pareto-Front $\mathcal{Y}_{true}$ mit der durch das Verfahren generierten Pareto-Front $\mathcal{Y}_{known}$ möglichst gut abzudecken.

7.2.1 Quantitative Kennzahlen zur Bewertung einer Pareto-Front

Dieser Abschnitt führt Kennzahlen ein, mit denen ein quantitativer Vergleich zwischen einer wahren Pareto-Front $\mathcal{Y}_{true}$ und einer generierten Pareto-Front $\mathcal{Y}_{known}$ möglich ist. Als quantitative Kennzahlen werden in Anlehnung an [187, S. 6-13ff] die nachfolgenden Kennzahlen verwendet:

- Overall Non-dominated Vector Generation ($ONVG$) - Gibt die Anzahl der Lösungen in der Pareto-Front $\mathcal{Y}_{known}$ an.

$$ONVG = |\mathcal{Y}_{known}| \tag{7.2}$$

  Dabei entspricht $|\mathcal{Y}|$ der Mächtigkeit der Menge $\mathcal{Y}$.

- Overall True Non-dominated Vector Generation ($OTNVG$) - Gibt die Anzahl der Lösungen in der Pareto-Front $\mathcal{Y}_{known}$ an, welche in der Pareto-Front $\mathcal{Y}_{true}$ enthalten sind.

$$OTNVG = |\{y | y \in \mathcal{Y}_{known} \wedge y \in \mathcal{Y}_{true}\}| \tag{7.3}$$

[66] Die Eigenschaften eines 2-fach-kantenzusammenhängenden Graphen erläutert Abschnitt 4.3.1 auf S. 70.

[67] Ist die wahre Pareto-Front $\mathcal{Y}_{true}$ für ein Problem nicht bekannt, so wird diese im Folgenden aus der Menge aller bekannten, pareto-optimalen Lösungen gebildet.

- Overall Non-dominated Vector Generation Ratio ($ONVGR$) - Gibt das Verhältnis zwischen der Anzahl der Lösungen in $\mathcal{Y}_{known}$ zur Anzahl der Lösungen in der Pareto-Front $\mathcal{Y}_{true}$ an.

$$ONVGR = \frac{ONVG}{|\mathcal{Y}_{true}|} \tag{7.4}$$

- Fehlerrate

$$\begin{aligned} Error &= \frac{\sum_{i=1}^{|\mathcal{Y}_{known}|} error(y_i)}{ONVG} \\ &\text{mit} \\ error(y_i) &= 0 \text{ wenn } y_i \in \mathcal{Y}_{known} \wedge y_i \in \mathcal{Y}_{true} \\ error(y_i) &= 1 \text{ sonst} \end{aligned} \tag{7.5}$$

  Das Fehlerrate spiegelt das Verhältnis zwischen Lösungen in der Menge $\mathcal{Y}_{known}$ wider, welche keine Mitglieder von $\mathcal{Y}_{true}$ sind. Ein $Error$ nahe eins bedeutet einen schwachen Zusammenhang zwischen beiden Pareto-Fronten. Ein $Error$ nahe null zeigt einen hohen Zusammenhang an.

Czyzak und Jaszkiewicz[41] weisen darauf hin, dass das Verhältnis $\frac{OTNVG}{|Y_{true}|}$ als Verhältnis der Anzahl der gefundenen Lösungen, welche ebenfalls in der Referenzfront enthalten sind, häufig als das natürlichste Qualitätsmaß für den Vergleich von $\mathcal{Y}_{known}$ und $\mathcal{Y}_{true}$ empfunden wird. Mit dessen Einsatz sind jedoch aus praktischen Gesichtspunkten Nachteile verbunden. Betrachtet man reale multikriterielle Optimierungsprobleme, so ist es in der Praxis häufig nicht möglich und notwendig, in einer angemessenen Laufzeit sämtliche Lösungen einer wahren Pareto-Front $\mathcal{Y}_{true}$ und damit einen hohen Wert für das Verhältnis von $OTNVG$ zu $|Y_{true}|$ zu erreichen [41]. Vielmehr ist aus praktischen Gesichtspunkte eine sehr gute Annäherung von $\mathcal{Y}_{known}$ an $\mathcal{Y}_{true}$ von Interesse.
Mit Abbildung 7.1 wird dieser Gedanke mittels eines Beispiels für die Annäherung dreier unterschiedlicher pareto-optimaler Lösungsmengen $\mathcal{Y}_{known}$ an eine wahre Pareto-Front $\mathcal{Y}_{true}$ mit zwei Zielfunktionen f_1 und f_2 dargestellt. Vergleicht man die einzelnen Pareto-Fronten $\mathcal{Y}_{known}$ in den drei unterschiedlichen Abbildungen, so wird häufig die in Abbildung 7.1(a) gezeigte Pareto-Front $\mathcal{Y}_{known}$ als beste Lösungsmenge empfunden [41]. Berechnet man für die in Abbildung 7.1(a) und Abbildung 7.1(b) dargestellten Pareto-Fronten $\mathcal{Y}_{known}$ die zuvor genannten Kennzahlen, so gilt u. a. für beide Pareto-Fronten $OTNVG = 0$, da keine ihrer Lösungen in $\mathcal{Y}_{true}$ enthalten ist. Für beide Pareto-Fronten würden nach den von Veldhuizen [187] vorgeschlagenen Kennzahlen sehr schlechte Ergebnisse ermittelt. Im Gegensatz dazu sind aus der mit Abbildung 7.1(c) dargestellten Pareto-Front fast alle Lösungen ebenfalls in $\mathcal{Y}_{true}$ enthalten, so dass die Kennzahlen $OTNVG$ und $Error$ diese Pareto-Front als sehr gute Pareto-Front klassifizieren. Das Ziel, eine möglichst gute Annäherung von $\mathcal{Y}_{known}$ an $\mathcal{Y}_{true}$ zu bewerten, wird daher durch die Kennzahlen nur teilweise berücksichtigt.
Anhand der Beispiele wird deutlich, dass für die Beurteilung einer Menge von pareto-optimalen Lösungen im Vergleich zu einer wahren Pareto-Front $\mathcal{Y}_{true}$ neben den rein

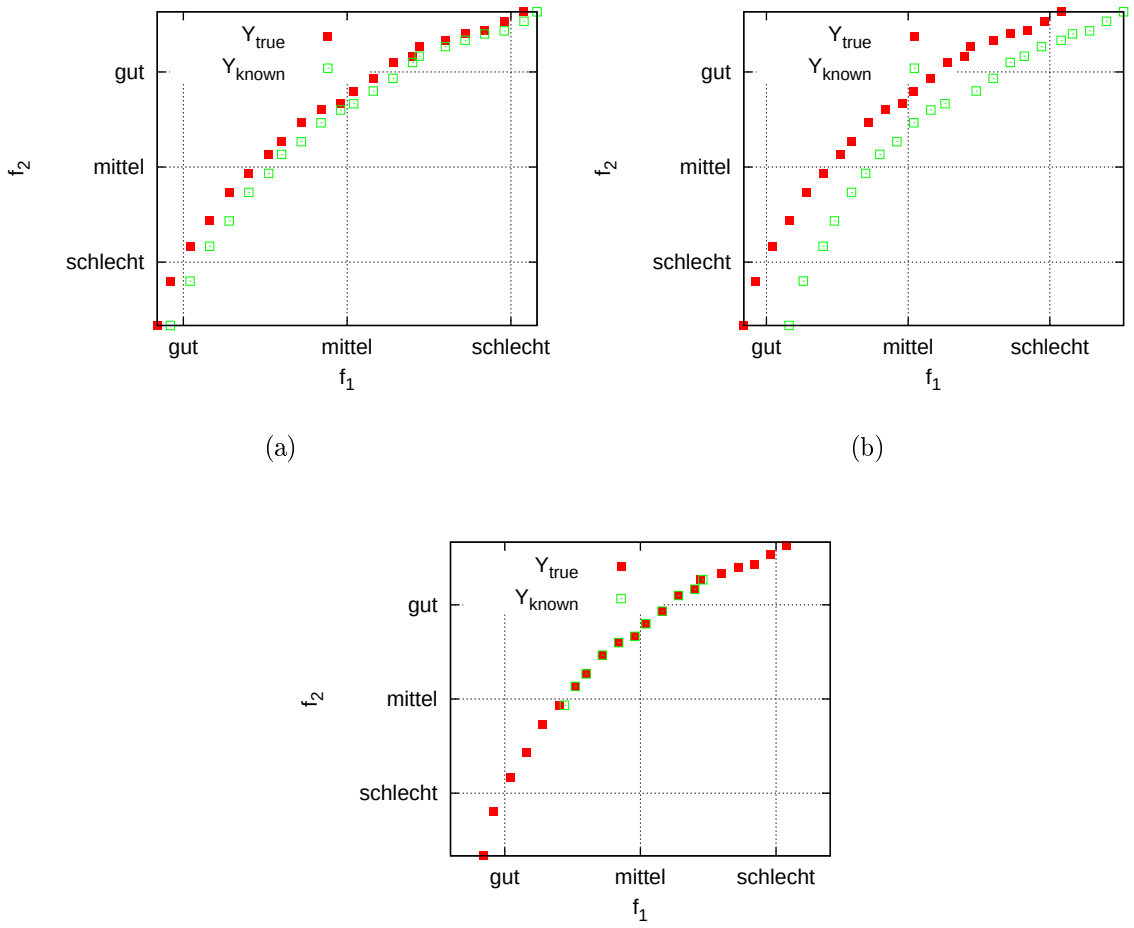


Abbildung 7.1: Veranschaulichung der unterschiedlichen Annäherung der Lösungsmenge $\mathcal{Y}_{known}$ an eine wahre Pareto-Front $\mathcal{Y}_{true}$ (in Anlehnung an [41])

quantitativen Kennzahlen zusätzlich eine qualitative Beurteilung, welche die Verteilung der Lösungen und deren Abstand zu $\mathcal{Y}_{true}$ berücksichtigt, notwendig ist.

7.2.2 Qualitative Kennzahlen zur Bewertung einer Pareto-Front

Während Abschnitt 7.2.1 Kennzahlen vorstellte, welche lediglich eine quantitative Beurteilung einer Pareto-Front ermöglichen, werden nachfolgend Kennzahlen eingeführt, mit denen eine qualitative Beurteilung einer Lösungsmenge $\mathcal{Y}_{known}$ im Vergleich zu einer Pareto-Front $\mathcal{Y}_{true}$ anhand unterschiedlicher Distanzmaße möglich ist.

Veldhuizen [187, S. 6-15] schlägt für die Messung des Abstandes zwischen einer Pareto-Front $\mathcal{Y}_{known}$ und einer Pareto-Front $\mathcal{Y}_{true}$ die „Generational Distance“ ($dist$) vor. Ihre Berechnung erfolgt als:

$$dist = \frac{\sqrt{\sum_{y \in \mathcal{Y}_{known}} \left(\min_{y' \in \mathcal{Y}_{known}} d(y, y') \right)^2}}{ONVG} \tag{7.6}$$

Die Berechnung der Distanz $d(y, y')$ zwischen zwei Lösungen $y \in \mathcal{Y}_{known}$ und $y' \in \mathcal{Y}_{known}$ erfolgt in dieser Arbeit in Anlehnung an [186] als:

$$d(y, y') = \sqrt{\sum_{j=1}^{J} (w_j (f_j(y_k) - f_j(y_l)))^2} \tag{7.7}$$

mit $w_j, j = 1, ..., J$ als Gewichte zur Normalisierung der Zielfunktionen f_j

Dabei gilt:

$$\begin{aligned} w_j &= \frac{1}{\Delta_j} \\ \text{mit } \Delta_j &= \max_{y \in \mathcal{Y}_{true}} f_j(y) - \min_{y \in \mathcal{Y}_{true}} f_j(y) \end{aligned} \tag{7.8}$$

Für $\max_{y \in \mathcal{Y}_{true}} f_j(y) = \min_{y \in \mathcal{Y}_{true}} f_j(y)$ ist $\Delta_j = 1$.
Die Generational Distance ($dist$) misst die Distanz zwischen den Pareto-Fronten $\mathcal{Y}_{known}$ und $\mathcal{Y}_{true}$. Dabei entspricht $d(y, y')$ dem Abstand zwischen den Fitnesswerten von jedem Element in $\mathcal{Y}_{known}$ und seinem nächsten zugehörigen Nachbarn in der Pareto-Front $\mathcal{Y}_{true}$. Eine hohes $dist$ bedeutet, dass $\mathcal{Y}_{known}$ weit entfernt von $\mathcal{Y}_{true}$ ist. Für $dist = 0$ sind beide Pareto-Fronten identisch.
Czyzak und Jaszkiewicz [41] schlagen ein Kennzahlensystem vor, welches es ermöglicht, den Abstand einer Pareto-Front $\mathcal{Y}_{true}$ zu einer Pareto-Front $\mathcal{Y}_{known}$ zu messen. Grundgedanke der im Folgenden vorgestellten Kennzahlen ist, dass eine gute Annäherung der Lösungsmenge $\mathcal{Y}_{true}$ an $\mathcal{Y}_{known}$ dann erreicht ist, wenn für jede Lösung $y \in \mathcal{Y}_{true}$ eine (annähernd) gleiche Lösung $y' \in \mathcal{Y}_{known}$ existiert. Dabei bewerten Czyzak und Jaszkiewicz im Gegensatz zur zuvor vorgestellten Kennzahl $dist$ nicht den Abstand von $\mathcal{Y}_{known}$ zu $\mathcal{Y}_{true}$, sondern den Abstand von $\mathcal{Y}_{true}$ zu $\mathcal{Y}_{known}$. Als Maß für den Abstand zweier Pareto-Fronten werden die Kennzahlen

$$\begin{aligned} dist_1 &= \frac{1}{|\mathcal{Y}_{true}|} \sum_{y \in \mathcal{Y}_{true}} \left(\min_{y' \in \mathcal{Y}_{known}} (d(y', y)) \right) \\ dist_2 &= \max_{y \in \mathcal{Y}_{true}} \left(\min_{y' \in \mathcal{Y}_{kwnown}} (d(y', y)) \right) \end{aligned} \tag{7.9}$$

genutzt. Dabei entspricht $d(y, y')$ der Distanz zwischen zwei Lösungen, welche nach Gleichung 7.7 bestimmt wird.
Die Kennzahl $dist_1$ drückt die durchschnittliche Distanz von der Lösung $y \in \mathcal{Y}_{true}$ zur nächsten Lösung $y' \in \mathcal{Y}_{known}$ aus. Mit $dist_2$ wird der maximale Abstand zwischen

einer Lösung $y \in \mathcal{Y}_{true}$ zur nächsten Lösung $y' \in \mathcal{Y}_{known}$ betrachtet. Als Maß für die Gleichförmigkeit geben Czyzak und Jaszkiewicz das Verhältnis

$$dist_3 = \frac{dist_2}{dist_1} \tag{7.10}$$

an. Für die Kennzahlen gilt: Je kleiner die Werte $dist_1$ und $dist_2$ sind, um so näher ist die Pareto-Front $\mathcal{Y}_{true}$ der Pareto-Front $\mathcal{Y}_{known}$. Je kleiner der Wert für $dist_3$ ist, um so gleichmäßiger sind die Lösungen in $\mathcal{Y}_{known}$ zu den Lösungen in $\mathcal{Y}_{true}$ verteilt. Für $dist_1 = 0$ gilt $dist_3 = 1$.

7.3 Multikriterielle Kommunikationsnetzwerkplanung mit Hilfe des NSGA-II

Als erstes Lösungsverfahren für das Problem der multikriteriellen Kommunikationsnetzwerkplanung mit den Zielen Kosten und Zuverlässigkeit wird mit diesem Abschnitt NSGA-II verwendet. Das Konzept und der Ablauf von NSGA-II sind in Abschnitt 3.5.6 auf S. 55 dargestellt. Für die nachfolgend durchgeführten experimentellen Untersuchungen wurden die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19 untersucht. Die Netzwerke wurden bereits für die Experimente in den Abschnitten 5.2.3 und 6.2.2 genutzt. Für jedes Netzwerk gilt: Für die Verbindungen im Netzwerk stehen jeweils drei unterschiedliche Technologieoptionen zur Verfügung, welche sich hinsichtlich Zuverlässigkeit und Kosten unterscheiden. Die charakteristischen Eigenschaften der Netzwerke sind in Abschnitt 5.2.3 auf S. 109 zu finden.
Eine Lösung G_N wird für den NSGA-II als Genom mit der Länge $|E|$ dargestellt. Jedes Element des Genoms g repräsentiert dabei eine Kante mit der gewählten Technologieoption l_k ($1 \leq k \leq k_{max}$) des Eingabegraphen G. Wie in Abbildung 7.2 gezeigt, ist $g_i = 0$ für jede Kante aus G, welche in einer Lösung nicht verwendet wird und entspricht anderenfalls der für die Kante gewählten Technologieoption l_k.

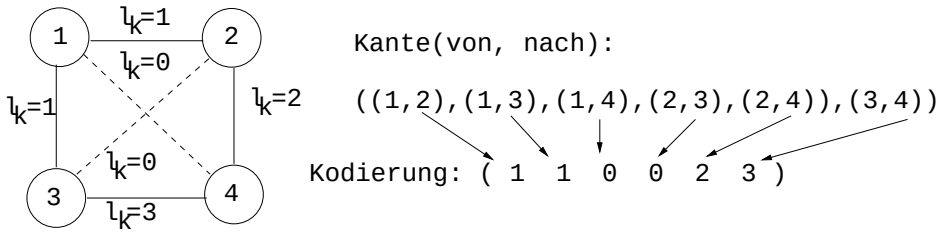


Abbildung 7.2: Kodierung einer Lösung für NSGA-II

7.3.1 Experimentelles Design für die Untersuchung des NSGA-II

Für die beschriebenen Testprobleme wurde die Fähigkeit des NSGA-II zum Finden einer Pareto-Front $\mathcal{Y}_{known}$, welche eine gute Annäherung an eine wahre Pareto-Front $\mathcal{Y}_{true}$ ist, untersucht. Für das Netzwerk türkei19 werden die hier erzielten Ergebnisse einer aus der Literatur bekannten Pareto-Front $\mathcal{Y}_{true}$ gegenübergestellt.
Die Experimente wurden mit einem NSGA-II unter Verwendung eines Uniform-Crossover (siehe S. 47) und einer Flip-Mutation (siehe S. 48) durchgeführt. Die Populationsgröße beträgt $pop = 100$. Die Mutationswahrscheinlichkeit beträgt $p_{mut} = 0,02$. In einem NSGA-II-Lauf wurden 300 Generationen durchlaufen. Ein vorzeitiger Abbruch eines Durchlaufs erfolgt nicht. Zur zufälligen Initialisierung der ersten Population wurde aus dem vollständig verbundenen Graphen eine zufällige Anzahl von Kanten $1 \leq n < |E|$ gelöscht. Die initiale Auswahl der Technologieoption für eine erstellte Verbindung erfolgte gleichverteilt mit $p_{init}^{opt} = 1/k_{max}$. Für jede durch Initialisierung, Mutation oder Crossover erzeugte Lösung wurde der 2-fach-Kantenzusammenhang geprüft und gegebenenfalls wiederhergestellt.
Pro Testproblem wurden jeweils zehn unabhängige Testläufe durchgeführt. Zur Bestimmung der All-Terminal-Zuverlässigkeit wurde für alle Testprobleme eine einfache Monte-Carlo-Simulationstechnik mit einem Stichprobenumfang $M = 100.000$ verwendet. Für die experimentelle Untersuchung wurde der in der Multiple Objective MetaHeuristics Library in C++ (MOMHLib++ [97]) Bibliothek implementierte NSGA-II angepasst. Die Experimente wurden auf einem P4-2-GHz-System unter Linux durchgeführt. Tabelle 7.1 fasst die für die Experimente relevanten Parameter zusammen.

Tabelle 7.1: Parameter für die experimentelle Untersuchung mit NSGA-II

GA	NSGA-II
Crossover-Operator	Uniform-Crossover, $p_{cross} = 1,0$
Mutations-Operator	Flip-Mutation, $p_{mut} = 0,02$
Initialisierung von P_0	Zufallsbasiert $p_{init}^{opt} = 1/k_{max}$
Bestimmung von R_{All}	Einfache Monte-Carlo-Simulation
pop	100
gen	300
Anzahl GA-Läufe	10

7.3.2 Experimentelle Ergebnisse und Auswertung

In diesem Abschnitt werden die mittels NSGA-II gewonnenen Ergebnisse ausgewertet. Eingangs erfolgt exemplarisch für das Netzwerk deutsch20 eine Gegenüberstellung der in den einzelnen Experimenten generierten Pareto-Fronten $\mathcal{Y}_i$ zu der Pareto-Front Y_{known}, welche die pareto-optimalen Lösungen sämtlicher Läufe enthält. Im Anschluss daran werden jeweils nur die aggregierten Pareto-Fronten betrachtet. Für das Netzwerk türkei19 wird die hier erzeugte Lösungsfront anderen, aus der Literatur entnommenen Pareto-Fronten, gegenübergestellt. Da für die Netzwerke deutsch15, deutsch20 und

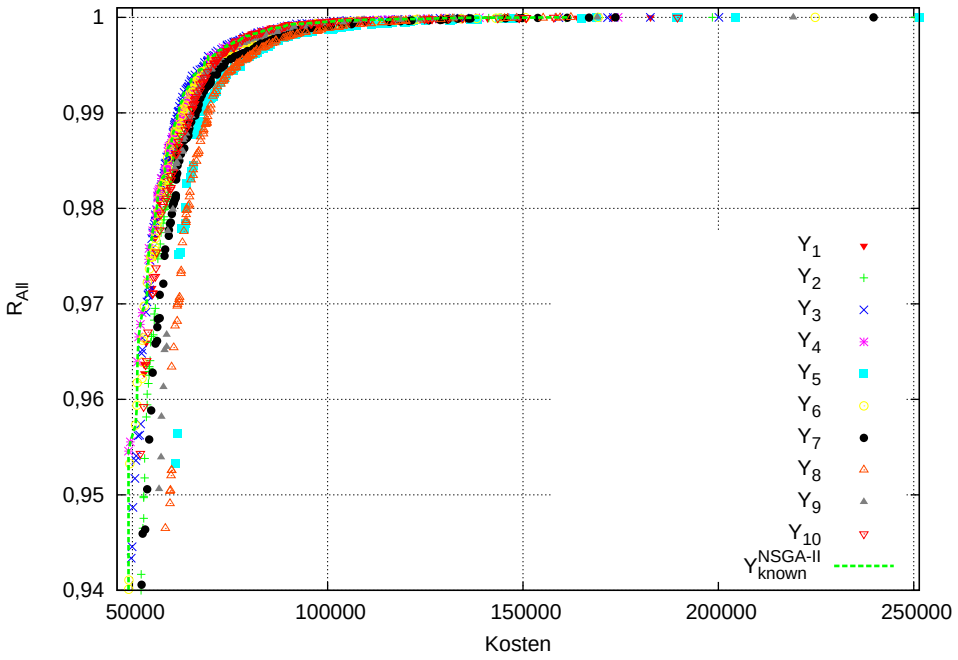


Abbildung 7.3: Darstellung der Lösungsmengen $\mathcal{Y}_{known}^{NSGA-II}$ und $\mathcal{Y}_1 \ldots \mathcal{Y}_{10}$ des unmodifizierten NSGA-II für das Netzwerk deutsch20

deutsch30 bisher keine Pareto-Fronten veröffentlicht wurden, werden die erzielten Ergebnisse lediglich in Relation zu den in Abschnitt 5.2.3.2 und Abschnitt 6.2.2.2 gezeigten Lösungen für die monokriteriellen Planungsprobleme gesetzt.
Mit Abbildung 7.3 (auf S. 152) und Tabelle 7.2 (auf S. 153) erfolgt eine Auswertung der in den einzelnen NSGA-II-Läufen generierten Pareto-Fronten $\mathcal{Y}_1, \ldots, \mathcal{Y}_{10}$ am Beispiel des Netzwerkes deutsch20. Es wird dabei der Einfluss der Ergebnisse der einzelnen Durchläufe auf die aus allen Läufen zusammengefasste Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ und die Aussagekraft der einzelnen Kennzahlen untersucht. Zur Bildung von $\mathcal{Y}_{known}^{NSGA-II}$ werden die Lösungsmengen $\mathcal{Y}_1, \ldots, \mathcal{Y}_{10}$ zu einer Menge zusammengefasst und anschließend sämtliche dominierte Lösungen aus dieser Lösungsmenge gelöscht.
Abbildung 7.3 stellt einen Ausschnitt[68] der einzelnen Pareto-Fronten grafisch dar. Dabei sind auf der Abszisse jeweils die Kosten einer Lösung und auf der Ordinate deren All-Terminal-Zuverlässigkeit abgetragen. Wie der Abbildung zu entnehmen ist, liegen die Lösungen der einzelnen Pareto-Fronten sehr nah zusammen. Nur die Lösungen der Pareto-Fronten $\mathcal{Y}_5$ und $\mathcal{Y}_8$ werden durch die Lösungen der anderen Pareto-Fronten für $C(G_N) < 100.000$ deutlich dominiert. Man sieht ebenfalls, dass eine Vielzahl an Lösungen direkt auf oder nah an der (grün gestrichelt dargestellten) aggregierten Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ liegen.
Eine detailliertere Analyse der Lösungsqualität der einzelnen Pareto-Fronten im Vergleich zur Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ wie sie die grafische Analyse bietet, ist mit Hilfe der in Tabelle 7.2 (auf S. 153) präsentierten Kennzahlen möglich. Die Tabelle zeigt für jede

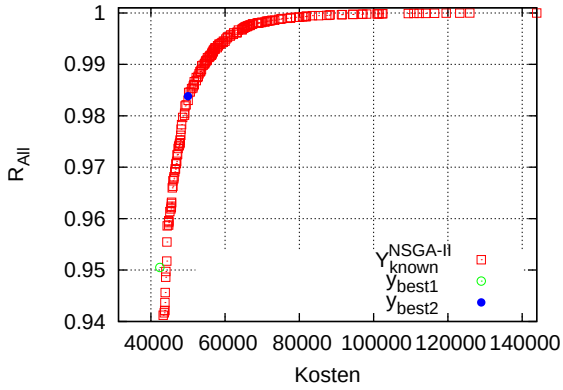
[68] Auf die komplette Darstellung über den Wertebereich beider Zielfunktionen wurde zur besseren Veranschaulichung der Ergebnisse verzichtet.

Tabelle 7.2: Gegenüberstellung der einzelnen Pareto-Fronten $\mathcal{Y}_1, \ldots, \mathcal{Y}_{10}$ zur aggregierten Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ für das Netzwerk deutsch20

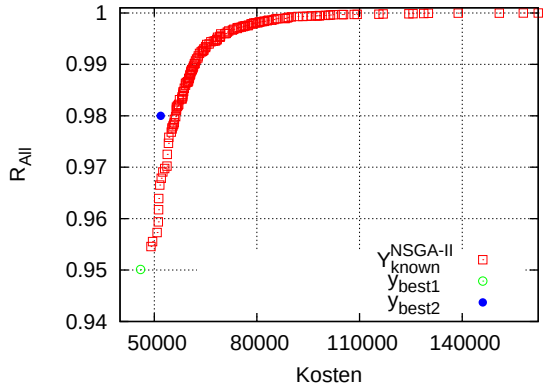
Pareto-Front	$ONVG$	$OTNVG$	$ONVGR$	$Error$	$dist$	$dist_1$	$dist_2$	$dist_3$
$\mathcal{Y}_1$	136	2	0,84	0,99	0,0015	0,026	0,663	25,5
$\mathcal{Y}_2$	179	0	1,1	1,00	0,0044	0,008	0,047	5,8
$\mathcal{Y}_3$	154	84	0,95	0,46	0,0020	0,010	0,486	46,7
$\mathcal{Y}_4$	142	31	0,88	0,78	0,0031	0,018	0,587	33,5
$\mathcal{Y}_5$	103	0	0,64	1,00	0,0058	0,029	0,197	6,8
$\mathcal{Y}_6$	160	14	0,99	0,91	0,0023	0,005	0,041	8,3
$\mathcal{Y}_7$	158	0	0,98	1,00	0,0031	0,016	0,074	4,8
$\mathcal{Y}_8$	160	1	0,99	0,99	0,0035	0,040	0,096	2,4
$\mathcal{Y}_9$	102	1	0,63	0,99	0,0036	0,020	0,33	16,4
$\mathcal{Y}_{10}$	115	31	0,71	0,73	0,0019	0,023	0,586	26

Pareto-Front $\mathcal{Y}_i$ jeweils die Anzahl der Lösungen in der Pareto-Front ($ONVG$), die Anzahl der Lösungen aus $\mathcal{Y}_i$ welche in $\mathcal{Y}_{known}^{NSGA-II}$ enthalten sind ($OTNVG$), das Verhältnis zwischen der Anzahl der Lösungen in $\mathcal{Y}_i$ und $\mathcal{Y}_{known}^{NSGA-II}$, die Fehlerrate $Error$ sowie die Distanzkennzahlen $dist$, $dist_1$, $dist_2$ und $dist_3$.
Wie man anhand des Wertes für $ONVG$ sieht, schwankt die Anzahl der pareto-optimalen Lösungen, welche in einem Durchlauf erzeugt werden zwischen 102 und 179. Die aggregierte Pareto-Front $\mathcal{Y}_{known}$ enthält 162 Lösungen. Der Wert für $OTNVG$ zeigt deutlich, dass einige Pareto-Fronten wie $\mathcal{Y}_3$, $\mathcal{Y}_4$ und $\mathcal{Y}_6$ eine sehr hohe Anzahl von Lösungen zu $\mathcal{Y}_{known}^{NSGA-II}$ beigetragen haben, während z. B. aus der Pareto-Front $\mathcal{Y}_7$ keine einzige Lösung ($OTNVG = 0$) in $\mathcal{Y}_{known}^{NSGA-II}$ enthalten ist.
Die Ergebnisse machen aber auch deutlich, dass eine alleinige Bewertung der Pareto-Fronten anhand der quantitativen Kennzahlen nicht ausreicht. So ist nur eine Lösung aus der Pareto-Front $\mathcal{Y}_2$ in $\mathcal{Y}_{known}^{NSGA-II}$ enthalten. Der Wert $dist_1 = 0,0139$ zeigt an, dass für jede Lösung aus $\mathcal{Y}_{known}^{NSGA-II}$ eine sehr nahe Lösung in $\mathcal{Y}_2$ existiert und der maximale Abstand zwischen den Lösungen aus $\mathcal{Y}_{known}^{NSGA-II}$ und $\mathcal{Y}_2$ gemessen an $dist_2 = 0,0831$ ebenfalls sehr gering ist. Der Wert für $dist_3$ macht ebenfalls deutlich, dass die Lösungen aus $\mathcal{Y}_2$ im Vergleich zu $\mathcal{Y}_{known}^{NSGA-II}$ ähnlich gut über den Lösungsraum verteilt sind. Anhand der Kennzahlen $dist_1$ und $dist_2$ wird deutlich, dass für alle Lösungen aus $\mathcal{Y}_{known}^{NSGA-II}$ eine annähernd gleiche Lösung in $\mathcal{Y}_6$ existiert. Diese Pareto-Front ist über alle Lösungen aus $\mathcal{Y}_{known}^{NSGA-II}$ gemittelt diejenige, welche die beste Näherung für $\mathcal{Y}_{known}^{NSGA-II}$ darstellt. Dabei gilt es zu beachten, dass nur zirka 9 % der Lösungen aus $\mathcal{Y}_{known}^{NSGA-II}$ in $\mathcal{Y}_6$ enthalten sind.
Wie bereits in der grafischen Darstellung deutlich wird, besitzen die Pareto-Fronten $\mathcal{Y}_5$ und $\mathcal{Y}_8$ den größten Abstand zu $\mathcal{Y}_{known}^{NSGA-II}$. Dies spiegelt sich ebenfalls in den großen Werten für $dist_1$ wider. Die geringen Werte für $dist_3$ zeigen jedoch an, dass beide Pareto-Fronten ähnlich wie $\mathcal{Y}_{known}^{NSGA-II}$ über den Lösungsraum verteilt sind.
Zusammenfassend lässt sich feststellen: Mit der durchgeführten grafischen und analytischen Betrachtung zum Beitrag der einzelnen Pareto-Fronten zur aggregierten Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ konnte gezeigt werden, dass eine isolierte Betrachtung anhand der eingeführten quantitativen Kennzahlen nicht sinnvoll ist. Insbesondere mit dem Ziel, mit

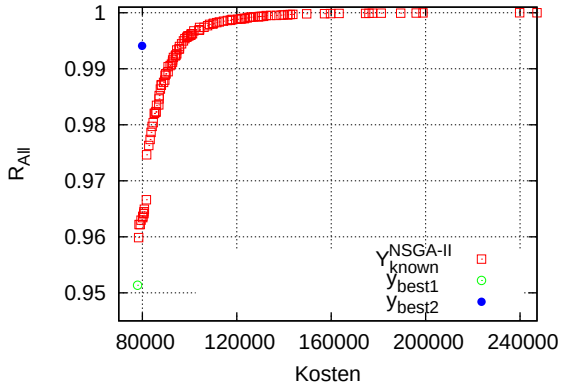
einem multikriteriellen Verfahren eine Pareto-Front zu generieren, welche eine optimale Pareto-Front möglichst gut abdeckt, müssen die vorgestellten qualitativen Kennzahlen ebenfalls herangezogen werden. In diesem Kapitel werden im Folgenden jeweils nur die aggregierten Pareto-Fronten $\mathcal{Y}_{known}$ betrachtet und auf eine Analyse der einzelnen Pareto-Fronten $\mathcal{Y}_i$ verzichtet.
Für die mittels NSGA-II untersuchten Netzwerke deutsch15, deutsch20 und deutsch30 wurden bisher in der Literatur keine pareto-optimalen Lösungsmengen veröffentlicht. Ein Vergleich zu anderen Arbeiten ist daher an dieser Stelle nicht möglich. Mit Abbildung 7.4 werden deshalb nur die Pareto-Fronten $\mathcal{Y}_{known}^{NSGA-II}$ für das jeweilige Problem mit den besten in Abschnitt 5.2.3.2 und Abschnitt 6.2.2.2 ermittelten Ergebnissen verglichen.
Die Abbildungen 7.4(a), 7.4(b) und 7.4(c) zeigen jeweils die aus allen zehn durchgeführten NSGA-II-Läufen aggregierte Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$, die beste in Abschnitt 5.2.3.2 gefundene Lösung y_{best1} sowie die beste in Abschnitt 6.2.2.2 gefundene Lösung y_{best2}.
Wie man in Abbildung 7.4(a) sieht, liegt y_{best2} direkt auf der Pareto-Front und y_{best1} weist nur einen geringen Abstand zu dieser auf. Dabei werden die Lösungen aus $\mathcal{Y}_{known}$ mit $R_{All}(G_N) \lesssim 0,95$ durch y_{best2} dominiert. Für das Testproblem deutsch20 dominiert die Lösung y_{best2} all jene Lösungen aus $\mathcal{Y}_{known}^{NSGA-II}$ mit $C(N) > 51.874$ und $R_{All}(G_N) < 0,98$. Die Lösung y_{best1} hingegen ist pareto-optimal zu den Lösungen in $\mathcal{Y}_{known}$, da durch den NSGA-II in diesem Bereich des Lösungsraums keine Lösungen generiert wurden.
Ein deutlicher Unterschied zeigt sich für das untersuchte Netzwerk deutsch30. Hier werden eine Vielzahl von Lösungen in $\mathcal{Y}_{known}$ durch die Lösung y_{best2} (all jene mit $R_{All}(G_N) < 0,995$ und $C(G_N) > 79.862$) in $\mathcal{Y}_{known}$ mit $R_{All}(G_N) \leq 0,9899$ dominiert. Die Lösung y_{best1} ist hingegen pareto-optimal. Zusammenfassend ist anhand von Abbildung 7.4 für die durchgeführten Experimente mit den Netzwerken deutsch15, deutsch20 und deutsch30 festzustellen, dass mit zunehmender Problemgröße der Abstand der besten mittels eines GAs unter Verwendung von STC3 gefundenen Lösungen zu den Lösungen in der Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ wächst.
Mit Abbildung 7.5 erfolgt die Auswertung der mittels NSGA-II ermittelten Ergebnisse für das Netzwerk türkei19. Dieses Netzwerk wurde bereits in der Literatur [63, 157] als multikriterielles Optimierungsproblem untersucht, so dass eine Gegenüberstellung der hier erzielten Ergebnisse zu bisher veröffentlichten Ergebnissen möglich ist.
Abbildung 7.5 zeigt die durch NSGA-II generierte Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$, die in [157] gezeigte Pareto-Front $\mathcal{Y}_{known}^{SA}$, welche mittels eines multikriteriellen Simulated Annealing (SA) Verfahrens generiert wurde, sowie die aus [63] entnommene Pareto-Front $\mathcal{Y}_{known}^{SPEA}$, welche die Autoren mit Hilfe einer SPEA-Implementierung erzeugten.
Weiterhin wurden in der Abbildung die besten in Abschnitt 5.2.3.2 (y_{best1}) und Abschnitt 6.2.2.2 (y_{best2}) gefundenen Lösungen abgetragen. Beide liegen direkt auf der Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ und dominieren eine Reihe von Lösungen in den Pareto-Fronten $\mathcal{Y}_{known}^{SA}$ und $\mathcal{Y}_{known}^{SPEA}$.
Bereits die grafische Gegenüberstellung der Pareto-Fronten verdeutlicht, dass die mit Hilfe des NSGA-II-Verfahrens generierten Lösungen pareto-optimal zu einer Vielzahl der Lösungen der anderen beiden gezeigten Pareto-Fronten sind. Für eine analytische Betrachtung wird die Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ den Pareto-Fronten $\mathcal{Y}_{known}^{SA}$ und $\mathcal{Y}_{known}^{SPEA}$



(a) deutsch15



(b) deutsch20



(c) deutsch30

Abbildung 7.4: Gegenüberstellung der mittels des unmodifizierten NSGA-II ermittelten Pareto-Fronten $\mathcal{Y}_{known}^{NSGA-II}$ und der besten Lösungen y_{best1} und y_{best2} für die Netzwerke deutsch15, deutsch20 und deutsch30

der Pareto-Front[69] $\mathcal{Y}_{true}$ gegenübergestellt. Tabelle 7.3 zeigt einen Vergleich anhand der in Abschnitt 7.2 eingeführten Kennzahlen. Der Wert für $ONVG$ zeigt die Anzahl der Lösungen in der jeweiligen Pareto-Front. Im Vergleich zu den Pareto-Fronten $\mathcal{Y}_{known}^{SPEA}$ und $\mathcal{Y}_{known}^{SA}$ besitzt die Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ eine deutlich größere Anzahl ($ONVG = 134$) an Lösungen. Der Wert für $OTNVG$ zeigt, dass keine der Lösungen aus $\mathcal{Y}_{known}^{SPEA}$ in $\mathcal{Y}_{true}$ enthalten ist und somit sämtliche Lösungen in $\mathcal{Y}_{known}^{SPEA}$ von den Lösungen aus $\mathcal{Y}_{known}^{SA}$ und $\mathcal{Y}_{known}^{NSGA-II}$ dominiert werden. Aus der Pareto-Front $\mathcal{Y}_{known}^{SA}$ sind nur 13 Lösungen in $\mathcal{Y}_{true}$ enthalten. Hingegen beinhaltet die Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ sämtliche Lösungen aus $\mathcal{Y}_{true}$.

[69] Die Pareto-Front $\mathcal{Y}_{true}$ wurde aus allen pareto-optimalen Lösungen aus $\mathcal{Y}_{known}^{NSGA-II}$, $\mathcal{Y}_{known}^{SA}$ und $\mathcal{Y}_{known}^{SPEA}$ gebildet.

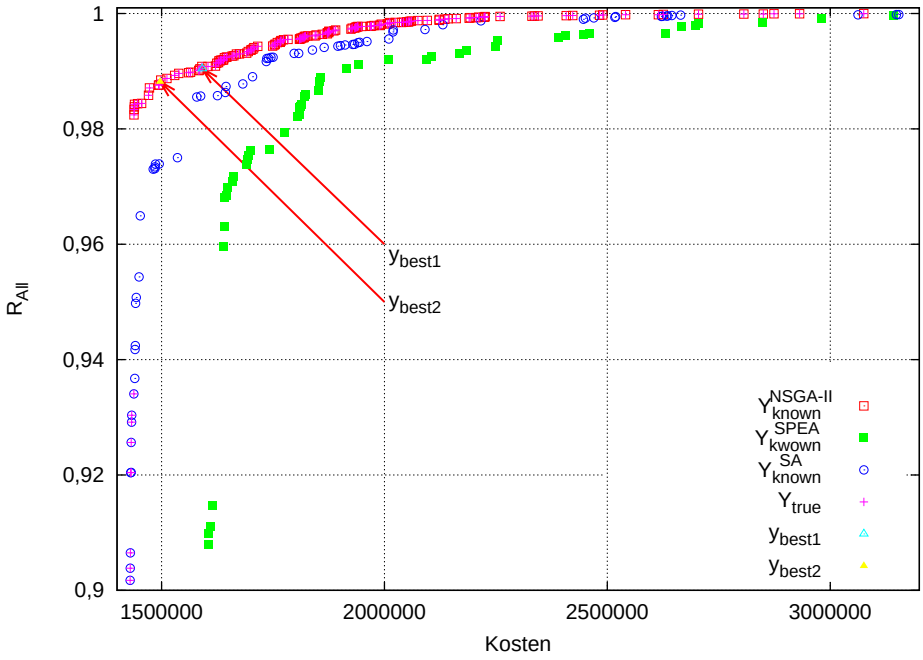


Abbildung 7.5: Gegenüberstellung der Pareto-Front $\mathcal{Y}_{true}$ im Vergleich zur Pareto-Front $\mathcal{Y}_{known}^{SA}$ aus [157], der Pareto-Front $\mathcal{Y}_{known}^{SPEA}$ aus [63], der mittels unmodifizierten NSGA-II ermittelten Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ und der besten Lösungen y_{best1} und y_{best2}

Tabelle 7.3: Analytische Auswertung der Ergebnisse für den unmodifizierten NSGA-II sowie der Ergebnisse aus [63] und [157] für das Problem türkei19

Pareto-Front	$ONVG$	$OTNVG$	$ONVGR$	$Error$	$dist$	$dist_1$	$dist_2$	$dist_3$
$\mathcal{Y}_{known}^{SPEA}$	57	0	0,388	1	0,0152	0,0424	0,131	3,1
$\mathcal{Y}_{known}^{SA}$	80	13	0,544	0,838	0,0091	0,0178	0,050	2,8
$\mathcal{Y}_{known}^{NSGA-II}$	134	134	0,912	0	0	0,0515	0,861	16,7

Über das Maß $dist$ wird der mittlere Abstand der Lösungen einer Pareto-Front $\mathcal{Y}_{known}$ zur nächsten Lösung in $\mathcal{Y}_{true}$ gemessen. Da alle Lösungen von $\mathcal{Y}_{known}^{NSGA-II}$ in $\mathcal{Y}_{true}$ enthalten sind, ergibt sich für das Distanzmaß $dist$ der Wert null. Wie bereits in der Gegenüberstellung in Abbildung 7.5 deutlich wird, besitzt die Pareto-Front $\mathcal{Y}_{known}^{SPEA}$ den größten Abstand gemessen an $dist$.
Mit den Werten für die Distanzmaße $dist_1$, $dist_2$ und $dist_3$, welche ein Ausdruck für die Annäherung einer Pareto-Front $\mathcal{Y}_{known}$ an $\mathcal{Y}_{true}$ sind, wird der in Abbildung 7.5 erkennbare Nachteil der Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ widergespiegelt. Da die Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ keine Lösungen mit $R_{All}(G_N) < 0,98$ enthält, ist der Abstand der Lösungen aus $\mathcal{Y}_{true}$ mit $R_{All}(G_N) < 0,98$ zu der nächsten Lösung in $\mathcal{Y}_{known}^{NSGA-II}$ sehr groß, so dass die Distanzkennzahlen im Vergleich zu den Pareto-Fronten $\mathcal{Y}_{known}^{SA}$ und $\mathcal{Y}_{known}^{SPEA}$ einen sehr hohen Wert ausweisen. Wie bereits in [157] gezeigt werden konnte, bestätigen die

Kennzahlen $dist_1$, $dist_2$ und $dist_3$, dass die Pareto-Front $\mathcal{Y}_{known}^{SA}$ eine bessere Annäherung an $\mathcal{Y}_{true}$ als $\mathcal{Y}_{known}^{SPEA}$ darstellt.
Zusammenfassend lässt sich aus der grafischen und analytischen Gegenüberstellung der Pareto-Fronten $\mathcal{Y}_{known}^{NSGA-II}$, $\mathcal{Y}_{known}^{SA}$ und $\mathcal{Y}_{known}^{SPEA}$ zu $\mathcal{Y}_{true}$ festhalten, dass die Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ für $R_{All}(G_N) > 0,98$ die besten bisher bekannten pareto-optimalen Lösungen enthält. Nachteilig ist, dass der NSGA-II keine pareto-optimalen Lösungen mit $R_{All}(G_N) < 0,98$ gefunden hat.
Die Tatsache, dass beide der mit Hilfe eines monokriteriellen Planungsverfahrens ermittelten Lösungen y_{best1} und y_{best2} in der Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ enthalten sind, zeigt die Fähigkeit des NSGA-II, ohne Kenntnis der in Abschnitt 5.2.2 und Abschnitt 6.2 unterstellten Präferenzen für die Ziele Kosten und Zuverlässigkeit, die besten Lösungen als Ergebnis einer multikriteriellen Planung zu generieren. Als Methode in einem Entscheidungsunterstützungswerkzeug bietet NSGA-II gegenüber den monokriteriellen Verfahren den Vorteil, dass es dem Entscheider nicht nur eine unter den angenommenen Präferenzen optimale Lösung präsentiert, sondern verschiedene Entscheidungsalternativen (welche die unter den angenommenen Präferenzen optimalen Lösungen enthalten) anbietet. Der Entscheider kann sich aus diesen Alternativen ex post diejenige Lösung auswählen, welche seine individuellen Präferenzen am besten befriedigt.

7.4 Zwei multikriterielle evolutionäre Planungsansätze mit einer integrierten lokalen Suche für die multikriterielle Kommunikationsnetzwerkplanung

Abschnitt 7.3 zeigte die experimentellen Ergebnisse unter Verwendung des NSGA-II für die multikriterielle Kommunikationsnetzwerkplanung für die Netzwerke deutsch15, deutsch20, deutsch30, und türkei19. Mit diesem Abschnitt werden zwei auf NSGA-II aufbauende Planungsverfahren eingeführt, welche durch die Integration einer lokalen Suche in das Planungsverfahren eine noch bessere Annäherung an die wahre Pareto-Front $\mathcal{Y}_{true}$ ermöglichen sollen.
Das in Abschnitt 7.4.1 vorgeschlagene Verfahren führt hierfür nach dem (unmodifizierten) NSGA-II-Durchlauf eine lokale Suche für sämtliche Elemente der Lösungsmenge durch. Mit Abschnitt 7.4.2 wird ein Verfahren vorgeschlagen, welches die lokale Suche direkt in NSGA-II integriert und dabei die Lösungen jeder Generation mit Hilfe einer lokalen Suche verbessert. Die mit beiden Verfahren experimentell erzielten Ergebnisse werden jeweils einzeln den Ergebnissen aus Abschnitt 7.3 gegenübergestellt. Mit Abschnitt 7.5 werden die Ergebnisse beider Verfahren direkt miteinander verglichen.

7.4.1 COMNETEA – Ein evolutionärer Planungsansatz mit einer nachgelagerten lokalen Suche

Mit dem Planungsverfahren COMNETEA (COMmunication NEtwork Topology dEsign Algorithm) wird in diesem Abschnitt ein multikriterielles evolutionäres Kommunikationsnetzwerkplanungsverfahren mit einer nachgelagerten lokalen Suche vorgeschlagen und experimentell untersucht.

Zielsetzung bei dem Entwurf von COMNETEA ist es, eine durch eine Metaheuristik generierte Menge von pareto-optimalen Lösungen $\mathcal{Y}_{true}$ durch die Verwendung einer lokalen Suche für jede dieser Lösungen in Richtung einer wahren/optimalen Lösungsmenge $\mathcal{Y}_{known}$ zu verbessern. In [46] schlagen Deb und Goel einen hybriden multikriteriellen evolutionären Algorithmus vor, welcher die Lösungen einer mit Hilfe von NSGA-II generierten Pareto-Front durch eine nachgelagerte lokale Suche verbessert. Die Autoren zeigen, dass durch die Anwendung der lokalen Suche eine zielgerichtete Verbesserung der Lösungen der Pareto-Front hin zu $\mathcal{Y}_{true}$ möglich ist. Mit COMNETEA wird das in [46] vorgestellte Konzept auf die multikriterielle Kommunikationsnetzwerkplanung mit den Zielen Zuverlässigkeit und Kosten übertragen.
In einer ersten Stufe wird in COMNETEA ein unmodifizierter NSGA-II eingesetzt, welcher die Lösungfront $\mathcal{Y}_{known}^{NSGA-II}$ erzeugt. Im Anschluss wird eine einfache lokale Suche durchgeführt. Den vollständige Ablauf der lokalen Suche in COMNETEA zeigt Algorithmus 7.1.
Für die lokale Suche werden an den Algorithmus die Lösungsmenge $\mathcal{Y}_{known}^{NSGA-II}$, welche die Lösungen des NSGA-II-Laufs enthält, sowie der Parameter LS, welcher die Anzahl der lokalen Suchschritte je Lösung in $\mathcal{Y}_{known}^{NSGA-II}$ bestimmt, übergeben. Nach Beendigung

Algorithmus 7.1 COMNETEA - lokale Suche

Eingabe: $\mathcal{Y}_{known}^{NSGA-II}$, LS
Ausgabe: $\mathcal{Y}_{known}^{COMNETEA}$

1: **for all** $y_{work} \in \mathcal{Y}_{known}^{NSGA-II}$ **do**
2: $\quad\mathcal{Y}_{known}^{COMNETEA} = \mathcal{Y}_{known}^{COMNETEA} \cup_{nondom} y_{work}$
3: $\quad$Berechne $w_{R_{All}}$ und w_C
4: $\quad$Berechne $f(y_{work})$
5: $\quad counter = 0$
6: $\quad$**while** $counter < LS$ **do**
7: $\quad\quad y_{Nach} = z(y_{work})$
8: $\quad\quad$Berechne $f(y_{Nach})$
9: $\quad\quad$**if** $f(y_{Nach}) < f(y_{work})$ **then**
10: $\quad\quad\quad y_{work} = y_{Nach}$
11: $\quad\quad$**end if**
12: $\quad\quad counter{+}{+}$
13: $\quad$**end while**
14: $\quad\mathcal{Y}_{known}^{COMNETEA} = \mathcal{Y}_{known}^{COMNETEA} \cup_{nondom} y_{work}$
15: **end for**

der lokalen Suche für sämtliche Lösungen wird die Lösungsmenge $\mathcal{Y}_{known}^{COMNETEA}$ zurückgegeben.
Mit der äußeren Schleife (Zeile 1 – 15) wird für jede der Lösungen in $\mathcal{Y}_{known}^{NSGA-II}$ die lokale Suche ausgeführt. Die aktuell verarbeitete Lösung wird in y_{work} gespeichert. Jede Lösung aus $\mathcal{Y}_{known}^{NSGA-II}$ wird zunächst mittels $\cup_{nondom}$[70] in die Lösungsmenge $\mathcal{Y}_{known}^{COMNETEA}$

[70] Der Operator $\cup_{nondom}$ fügt eine neue Lösung y einer pareto-optimalen Lösungsmenge $\mathcal{Y}$ hinzu, wenn y pareto-optimal zu Lösungen in $\mathcal{Y}$ ist. Lösungen in $\mathcal{Y}$, welche y dominiert, werden durch den Operator $\cup_{nondom}$ aus $\mathcal{Y}$ gelöscht.

übernommen. Im Gegensatz zu NSGA-II führt die lokale Suche keine multikriterielle Suche durch, sondern berechnet für jede einzelne Lösung einen skalaren Zielfunktionswert $f(y)$, welcher sich aus den gewichteten Zielfunktionswerten der All-Terminal-Zuverlässigkeit und den Kosten der Lösung zusammensetzt. Die Zielfunktion für die lokale Suche von COMNETEA lautet:

$$f(y) = w_C \cdot \frac{C(G_N)}{C^{max}_{\mathcal{Y}^{NSGA-II}_{known}} - C^{min}_{\mathcal{Y}^{NSGA-II}_{known}}} + w_{R_{All}} \cdot (1 - \frac{R_{All}(G_N)}{R_{All}{}^{max}_{\mathcal{Y}^{NSGA-II}_{known}} - R_{All}{}^{min}_{\mathcal{Y}^{NSGA-II}_{known}}}) \rightarrow \min \tag{7.11}$$

Für die Berechnung der Zielfunktion ist die Normierung der einzelnen Zielfunktionswerte für die Kosten und die All-Terminal-Zuverlässigkeit notwendig. Hierfür wird der jeweilige Zielfunktionswert durch die Spannweite[71] des jeweiligen Kriteriums dividiert. Zusätzlich berücksichtigt die Zielfunktion, dass die Kosten des Netzwerkes zu minimieren sind und die All-Terminal-Zuverlässigkeit zu maximieren ist. Mittels des Terms $(1 - \frac{R_{All}(G_N)}{R_{All}{}^{max}_{\mathcal{Y}^{NSGA-II}_{known}} - R_{All}{}^{min}_{\mathcal{Y}^{NSGA-II}_{known}}})$ wird die All-Terminal-Zuverlässigkeitsmaximierung in ein Minimierungsproblem überführt, so dass die Bildung der zusammengesetzen Zielfunktion möglich ist.

Ausschlaggebend für die Bestimmung der Gewichte[72] ist die Lage der Lösung y_{work} auf der Pareto-Front $\mathcal{Y}^{NSGA-II}_{known}$. Die Gewichte $w_{R_{All}}$ und w_C einer Lösung y_{work} werden wie folgt berechnet:

$$\begin{aligned} w_C &= \frac{C^{max}_{\mathcal{Y}^{NSGA-II}_{known}} - C(G_N)}{C^{max}_{\mathcal{Y}^{NSGA-II}_{known}} - C^{min}_{\mathcal{Y}^{NSGA-II}_{known}}} \cdot \frac{1}{w_C + w_{R_{All}}} \\ w_{R_{All}} &= \frac{R_{All}(G_N) - R_{All}{}^{min}_{\mathcal{Y}^{NSGA-II}_{known}}}{R_{All}{}^{max}_{\mathcal{Y}^{NSGA-II}_{known}} - R_{All}{}^{min}_{\mathcal{Y}^{NSGA-II}_{known}}} \cdot \frac{1}{w_C + w_{R_{All}}} \end{aligned} \tag{7.12}$$

Die Werte $C^{max}_{\mathcal{Y}^{NSGA-II}_{known}}$ bzw. $R_{All}{}^{max}_{\mathcal{Y}^{NSGA-II}_{known}}$ entsprechen dem maximalen Wert für die Kosten bzw. der All-Terminal-Zuverlässigkeit. Die Werte $C^{min}_{\mathcal{Y}^{NSGA-II}_{known}}$ bzw. $R_{All}{}^{min}_{\mathcal{Y}^{NSGA-II}_{known}}$ sind die jeweils kleinsten Werte der Zielfunktionen. Wenn der maximale und minimale Zielfunktionswert identisch sind, so wird das Gewicht auf eins gesetzt. Über den Term $\frac{1}{w_C + w_{R_{All}}}$ werden die Gewichte normiert, so dass gilt $w_C + w_{R_{All}} = 1$.

Mit Hilfe von Abbildung 7.6 wird die Wahl der Gewichte veranschaulicht. Für jede der beiden Zielfunktionen ist die Spannweite der Zielfunktionswerte ($C^{max}_{\mathcal{Y}^{NSGA-II}_{known}} - C^{min}_{\mathcal{Y}^{NSGA-II}_{known}}$ bzw. $R_{All}{}^{max}_{\mathcal{Y}^{NSGA-II}_{known}} - R_{All}{}^{min}_{\mathcal{Y}^{NSGA-II}_{known}}$) dargestellt. Exemplarisch werden drei der Lösungen aus $\mathcal{Y}^{NSGA-II}_{known}$ betrachtet.

[71] Die Spannweite eines Zielfunktionswertes berechnet sich aus der Differenz zwischen dem maximalen und dem minimalen Zielfunktionswert der Kosten bzw. der All-Terminal-Zuverlässigkeit der Lösungen in $\mathcal{Y}^{NSGA-II}_{known}$.

[72] Für $C^{max}_{\mathcal{Y}^{NSGA-II}_{known}} = C^{min}_{\mathcal{Y}^{NSGA-II}_{known}}$ bzw. $R_{All}{}^{max}_{\mathcal{Y}^{NSGA-II}_{known}} = R_{All}{}^{min}_{\mathcal{Y}^{NSGA-II}_{known}}$ werden die Gewichte der jeweiligen Zielfunktion als Zufallszahl zwischen null und eins bestimmt.

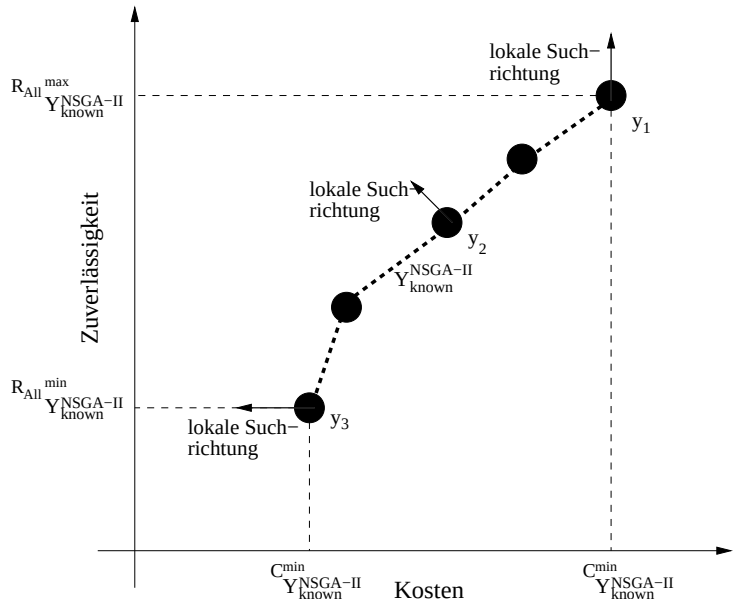


Abbildung 7.6: Schematische Darstellung einer pareto-optimalen Lösungsmenge $\mathcal{Y}_{known}^{NSGA-II}$ zur Bestimmung der Gewichte für die lokale Suche

Die Lösung y_1 besitzt die höchste All-Terminal-Zuverlässigkeit und weist die höchsten Kosten auf. Die Lösung y_3 besitzt die geringsten Kosten und die geringste All-Terminal-Zuverlässigkeit. Für die Lösung y_2, welche sich in der Mitte der Pareto-Front befindet, gilt: $R_{All}(y_3) < R_{All}(y_2) < R_{All}(y_1)$ und $C(y_3) < C(y_2) < C(y_1)$. Alle drei Lösungen sind zueinander pareto-optimal.
Gemäß Gleichung 7.12 ergeben sich für y_1 die Gewichte $w^{y_1}_{C(G_N)} = 0$ und $w^{y_1}_{R_{All}(G_N)} = 1$. Für y_3 werden die Gewichte $w^{y_3}_{C(G_N)} = 1$ und $w^{y_3}_{R_{All}(G_N)} = 0$ berechnet. Da sowohl $R_{All}(y_2)$ als auch $C(y_2)$ den gleichen Abstand vom jeweiligen maximalen und minimalen Zielfunktionswert haben, ergibt sich für die Gewichte dieser Lösung $w^{y_2}_{C(G_N)} = 0,5$ und $w^{y_2}_{R_{All}(G_N)} = 0,5$.
Mittels der while-Schleife (Zeile 6–13) in Algorithmus 7.1 wird die lokale Suche unter Verwendung der mit Hilfe der Gewichte gebildeten Zielfunktion durchgeführt. Für jede Lösung in $\mathcal{Y}_{known}^{NSGA-II}$ werden durch das lokale Suchverfahren LS Suchschritte durchgeführt. Bei der lokalen Suche wird in jeder Iteration für die Lösung y_{work} eine Lösung y_{Nach} erzeugt. Der Zug $z(y_{work})$ erzeugt eine Lösung y_{Nach} in der Nachbarschaft von y_{work}, indem er die Technologieoptionen für zwei zufällig gewählte Verbindungen zufallsbasiert verändert[73]. Für die Lösung y_{Nach} wird der skalare Zielfunktionswert $f(y_{Nach})$ mit Hilfe der zuvor bestimmten Gewichte für die beiden Zielfunktionen berechnet. Ist der Zielfunktionswert $f(y_{Nach}) < f(y_{work})$, wird y_{work} durch y_{Nach} ersetzt und die lokale Suche mit der neuen Lösung fortgesetzt.

[73] Der Zug stellt dabei sicher, dass durch die Veränderung der 2-fach-Zusammenhang des Graphen nicht zerstört wird.

Nach Beendigung der lokalen Suche wird die aktuelle Lösung y_{work} mittels $\cup_{nondom}$ der Lösungsmenge $\mathcal{Y}_{known}^{COMNETEA}$ hinzugefügt.

7.4.1.1 Experimentelles Design für die Untersuchung von COMNETEA

Nachfolgend wird COMNETEA für die multikriterielle Kommunikationsnetzwerkplanung mit den Zielen Zuverlässigkeit und Kosten für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19 untersucht.
Wie bereits zuvor für die Verfahren LaBORNet (in Abschnitt 5.2.2.1 auf S. 107) und SAGA (in Abschnitt 6.2.2 auf S. 136) dargestellt, wird auch für COMNETEA eine Lösung als Vektor g mit der Länge $|E|$ kodiert. Jedes Element von g repräsentiert dabei eine Kante mit der für die Kante gewählten Technologieoption l_k $(1 \leq k \leq k_{max})$ des Eingabegraphen G. Wenn eine Kante aus G in einer Lösung nicht verwendet wird, so ist $g_i = 0$. Abbildung 7.7 zeigt die Kodierung einer Lösung für COMNETEA[74].

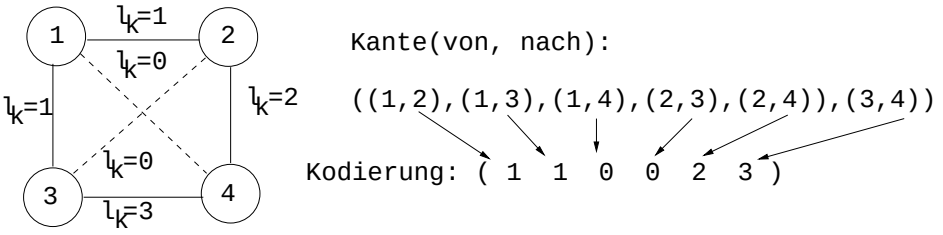


Abbildung 7.7: Kodierung einer Lösung für COMNETEA

Die Lösungen der initialen Population P_0 wurden zufallsbasiert erzeugt. Hierfür wurde für jede Lösung aus einem vollständig verbundenen Graphen eine zufällig bestimmte Anzahl an Verbindungen gelöscht. Die initiale Technologieoption für jede im Graphen verbliebene Verbindung wurde gleichverteilt mit $p_{init}^{opt} = 1/k_{max}$ gewählt. Als Mutationsoperator wird für die experimentelle Untersuchung eine Flip-Mutation (siehe S. 48) mit einer Mutationswahrscheinlichkeit von $p_{mut} = 0,02$ genutzt. Für die Rekombination wird ein Uniform-Crossover-Operator (siehe S. 47) mit einer Crossoverwahrscheinlichkeit von $p_{cross} = 1$ verwendet. In der experimentellen Untersuchung wird der Einfluss der lokalen Suchschritte LS (100, 200, 500), der Einfluss der Anzahl der durchlaufenen Generationen *gen* (50, 100, 150, 300) sowie der Einfluss der Populationsgröße *pop* (50, 100) untersucht. Ein Durchlauf wurde stets nach der maximalen Anzahl an Generationen *gen* beendet. Tabelle 7.4 fasst die für die empirische Untersuchung verwendeten Parameter zusammen.

7.4.1.2 Experimentelle Ergebnisse und Auswertung

Mit diesem Abschnitt werden die experimentell erzielten Ergebnisse dargestellt und ausgewertet. Es erfolgt zunächst eine Analyse des Einflusses der Anzahl der lokalen

[74] Verbindungen von G, die für die abgebildete Lösung nicht verwendet werden, sind gestrichelt dargestellt.

Tabelle 7.4: Parameter für die experimentelle Untersuchung mit COMNETEA

GA	NSGA-II
Crossover-Operator	Uniform-Crossover, $p_{cross} = 1$
Mutations-Operator	Flip-Mutation, $p_{mut} = 0,02$
Initialisierung von P_0	Zufallsbasiertes Löschen von Kanten in einem vollständigen Graphen; $p_{init}^{opt} = 1/k_{max}$
Bestimmung von R_{All}	Einfache Monte-Carlo-Simulation
pop	50,100
gen	50, 100, 150, 300
Anzahl GA-Läufe	5

Suchschritte (LS). Daran schließt sich eine Untersuchung des Einflusses der Anzahl der Generationen sowie der Populationsgröße an. Abschließend werden die Ergebnisse, welche mit der besten Parameterkombination (bestehend aus LS, *gen* und *pop*) ermittelt wurden, einer aus allen Ergebnissen aggregierten Pareto-Front gegenübergestellt und ein Vergleich zu den Ergebnissen eines unmodifizierten NSGA-II vorgenommen.

Untersuchung des Einflusses der Suchtiefe

Mit Hilfe des Parameters LS wird die Anzahl der lokalen Suchschritte, welche für eine Lösung in $\mathcal{Y}_{known}^{NSGA-II}$ durchgeführt werden, für COMNETEA festgelegt. Für jede Lösung $y \in \mathcal{Y}_{known}^{NSGA-II}$ stoppt die lokale Suche, nachdem LS mal eine Lösung y_{Nach} in der Nachbarschaft von y_{work} erzeugt und beide hinsichtlich der mit Gleichung 7.11 ermittelten Zielfunktionswerte verglichen wurden. Für die Untersuchung des Einflusses des Parameters LS wurden das Netzwerk türkei19 und die Parameter $pop = 50$, $gen = 100$ und $LS = 100, 200, 500$ gewählt. Die dargestellten Pareto-Fronten wurden jeweils aus allen pareto-optimalen Lösungen aus jeweils fünf COMNETEA-Durchläufen je Parameterkombination gebildet.
Abbildung 7.8 auf S. 163 zeigt drei mit Hilfe dieser Parameter erzeugte Pareto-Fronten[75]. Wie Abbildung 7.8 verdeutlicht, liegen die Ergebnisse sehr dicht beieinander. Die grafische Darstellung macht jedoch deutlich, dass erwartungsgemäß mit einer Erhöhung der Anzahl der lokalen Suchschritte eine Verbesserung der Ergebnisse möglich ist. So zeigen die in Abbildung 7.8 dargestellten Ergebnisse, dass viele der Lösungen für $LS = 100$ durch Lösungen der Pareto-Fronten für $LS = 200$ und $LS = 500$ dominiert werden. Für die mit $LS = 200$ und $LS = 500$ ermittelten Pareto-Fronten ist anhand der grafischen Auswertung keine eindeutige Aussage bezüglich der Dominanz der einen Pareto-Front über die andere möglich, so dass eine weitergehende Analyse mit Hilfe der in Abschnitt 7.2 eingeführten Kennzahlen durchgeführt wird.
Tabelle 7.5 zeigt die analytische Auswertung der mit $LS = 100, 200, 500$ ermittelten Pareto-Fronten im Vergleich zu einer aus allen drei Pareto-Fronten aggregierten Pareto-Front $\mathcal{Y}_{known}^{*}$. Für jede Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$ wird die Anzahl der durchgeführten

[75] Für eine bessere Veranschaulichung der Ergebnisse zeigt Abbildung 7.8 nur den Bereich $0,98 \leq R_{All} \leq 1$.

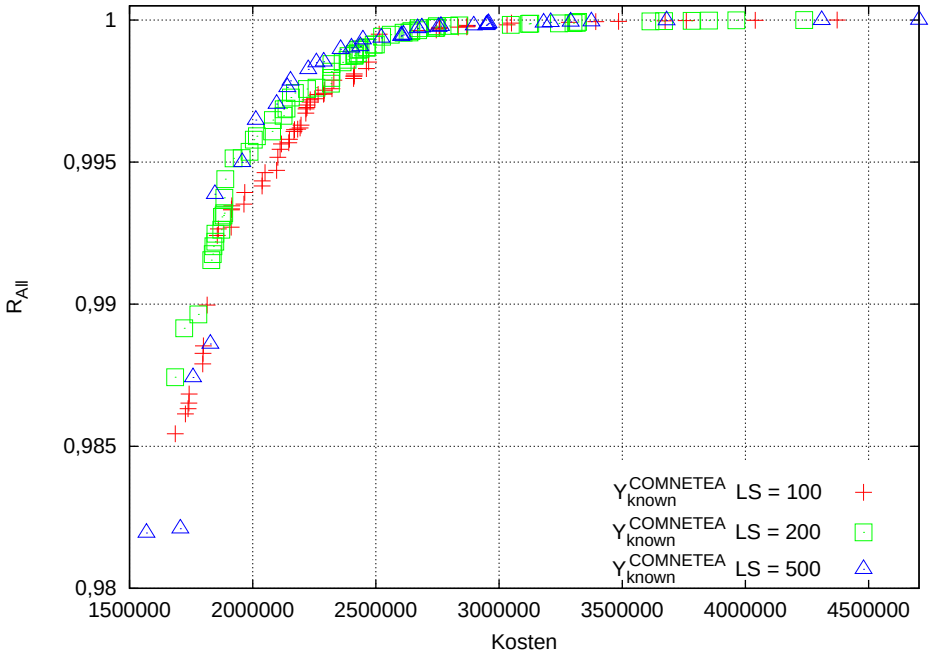


Abbildung 7.8: Auswertung des Einflusses der lokalen Suchtiefe auf die Ergebnisqualität für das Netzwerk türkei19 mit $pop = 50$ und $gen = 150$

Tabelle 7.5: Analytische Auswertung des Einflusses der lokalen Suchtiefe auf die Ergebnisqualität für das Netzwerk türkei19 mit $pop = 50$ und $gen = 150$

LS	$ONVG$	$OTNVG$	$ONVGR$	$Error$	$dist$	$dist_1$	$dist_2$	$dist_3$
100	68	6	1,31	0,912	0,0035	0,0401	0,6070	15,1
200	71	21	1,37	0,704	0,0022	0,0222	0,4100	18,4
500	39	25	0,75	0,359	0,0040	0,0116	0,0871	7,5

lokalen Suchschritte LS, die Anzahl der Lösungen in der Pareto-Front ($ONVG$), die Anzahl der Lösungen ($OTNVG$) aus $\mathcal{Y}_{known}^{COMNETEA}$, welche in $\mathcal{Y}_{known}^{*}$ enthalten sind, das Verhältnis ($ONVGR$) zwischen der Anzahl der Lösungen in $\mathcal{Y}_{known}^{COMNETEA}$ und der Anzahl der Lösung in der Pareto-Front $\mathcal{Y}_{known}^{*}$, die Fehlerrate ($Error$), der mittlere Abstand der Lösungen aus $\mathcal{Y}_{known}^{COMNETEA}$ zur nächsten Lösung in $\mathcal{Y}_{known}^{*}$ sowie die Abstandsmaße $dist_1$, $dist_2$ und $dist_3$, welche den Abstand von $\mathcal{Y}_{known}^{*}$ zu $\mathcal{Y}_{known}^{COMNETEA}$ bewerten, dargestellt.

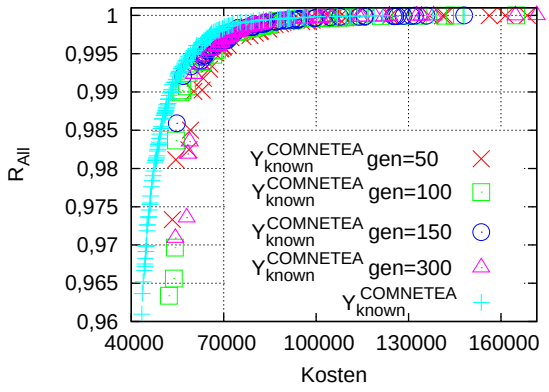
Die Ergebnisse zeigen, dass die Anzahl der Lösungen in der Pareto-Front ($ONVG$) mit einem Zuwachs von LS abnimmt. Gleichzeitig steigt mit einer Zunahme von LS die Anzahl der Lösungen aus $\mathcal{Y}_{known}^{COMNETEA}$, die ebenfalls in $\mathcal{Y}_{known}^{*}$ enthalten sind. Betrachtet man das Abstandsmaß $dist$, so fällt auf, dass die Pareto-Fronten für $LS = 100$ und $LS = 200$ sehr nah an $\mathcal{Y}_{known}^{*}$ liegen. Deren Lösungen haben im Mittel einen geringeren Abstand zur nächsten Lösung in $\mathcal{Y}_{known}^{*}$, als die Lösungen der Pareto-Front für $LS = 500$. Daraus folgt, dass beide Pareto-Fronten einen Teilbereich von $\mathcal{Y}_{known}^{*}$ sehr gut repräsentieren. Die Distanzkennzahl $dist_1$ macht deutlich, dass die Lösungen aus $\mathcal{Y}_{known}^{*}$ im Mittel den geringsten Abstand zu den Lösungen der Pareto-Front für $LS = 500$

besitzen. Auch das Maximum des minimalen Abstandes einer Lösung aus $\mathcal{Y}^*_{known}$ zur nächsten Lösung in $\mathcal{Y}^{COMNETEA}_{known}$ (gemessen mit $dist_2$) ist für $LS = 500$ am geringsten. Aufgrund dieser Kennzahlen wird deutlich, dass die Pareto-Front für $LS = 500$ insgesamt die beste Annäherung für sämtliche Lösungen aus $\mathcal{Y}^*_{known}$ bietet.
Dem Ziel, eine sehr gute Annäherung einer Pareto-Front $\mathcal{Y}^{COMNETEA}_{known}$ an eine optimale Lösungsmenge $\mathcal{Y}_{true}$ (hier durch $\mathcal{Y}^*_{known}$ ersetzt) zu erreichen, wird somit die Pareto-Front für $LS = 500$ am besten gerecht. Für die weiteren Experimente wurde daher die Anzahl der Suchschritte mit $LS = 500$ festgelegt.

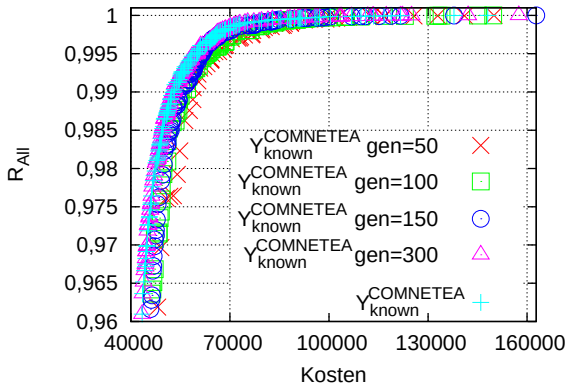
Untersuchung des Einflusses der Populationsgröße und der Anzahl der Generationen

Nach der Festlegung der Anzahl der lokalen Suchschritte erfolgt jetzt eine Analyse des Einflusses der Anzahl der durchlaufenen Generationen (*gen*) und der Populationsgröße (*pop*) auf die Ergebnisqualität. Von Interesse ist dabei, inwiefern durch die Erweiterung des NSGA-II-Verfahrens durch die nachgelagerte lokale Suche (wie es mit COMNETEA umgesetzt wurde) eine bessere Annäherung einer Lösungsfront $\mathcal{Y}_{known}$ an $\mathcal{Y}_{true}$ möglich ist.
Mit Abbildung 7.9 und Abbildung 7.10 werden die Pareto-Fronten für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19 für unterschiedliche Populationsgrößen und Anzahlen der zu durchlaufenden Generationen dargestellt[76]. Für jede Parameterkombination wurden jeweils fünf Durchläufe ausgeführt und die dabei erzielten Ergebnisse in einer Pareto-Front zusammengefasst. Zusätzlich zeigen Abbildung 7.9 und Abbildung 7.10 für jedes Netzwerk die aus allen Ergebnissen aggregierte Pareto-Front $\mathcal{Y}^{COMNETEA}_{known}$.
Für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19 wird anhand der in Abbildung 7.9 auf S. 165 und Abbildung 7.10 auf S. 166 dargestellten Pareto-Fronten deutlich, dass die besten Lösungen mit $pop = 100$ erzielt wurden. Es ist zu erkennen, dass die Durchläufe mit einer höheren Anzahl an Generationen näher an $\mathcal{Y}^{COMNETEA}_{known}$ liegen. Die Abbildungen 7.9 und Abbildung 7.10 zeigen ebenfalls den Einfluss der Populationsgröße auf die Lösungsfronten. Durch eine Verdopplung der Populationsgröße von 50 auf 100 konnte bei der gleichen Anzahl an Generationen eine deutliche Verbesserung der Pareto-Fronten erreicht werden. Auffällig ist, dass für die Netzwerke deutsch15, deutsch20 und deutsch30 für $pop = 50$ nur wenige Lösungen mit $R_{All}(G_N) < 0,985$ gefunden werden. Für alle drei Netzwerke wird der Bereich $0,96 \leq R_{All} \leq 0,985$ der aggregierten Pareto-Front $\mathcal{Y}^{COMNETEA}_{known}$ vorwiegend aus Lösungen, welche in Experimenten mit $pop = 100$ erzeugt wurden, gebildet. Eine mögliche Ursache hierfür kann die größere Suchbreite für $pop = 100$ sein. Während in einem Durchlauf mit $pop = 50$ je Generation lediglich 50 Lösungen parallel betrachtet werden, wird für $pop = 100$ die doppelte Menge untersucht.

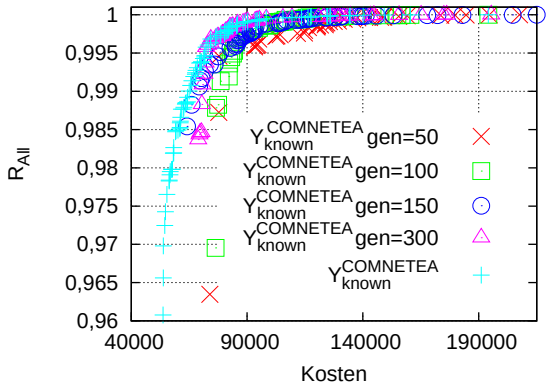
[76] Zur Veranschaulichung der Ergebnisse wurde jeweils nur ein Teilbereich der Pareto-Fronten dargestellt.



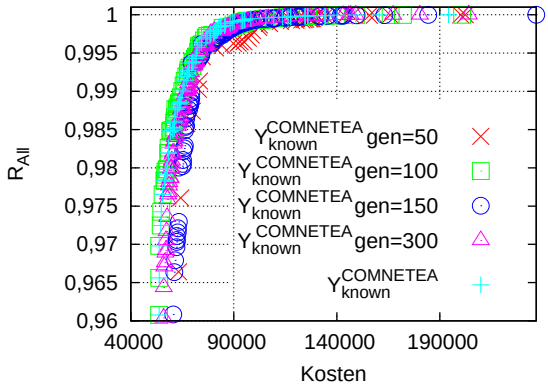
(a) deutsch15 $pop = 50$



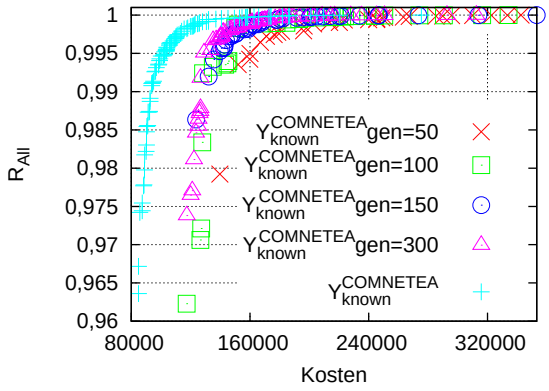
(b) deutsch15 $pop = 100$



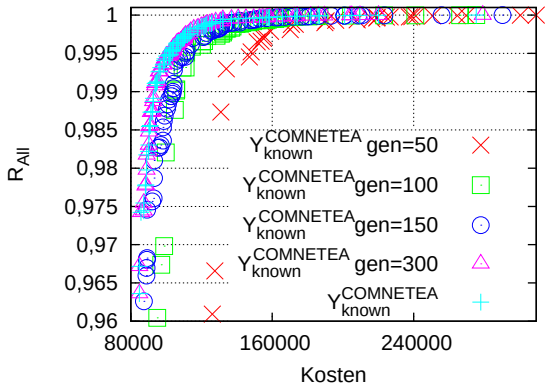
(c) deutsch20 $pop = 50$



(d) deutsch20 $pop = 100$

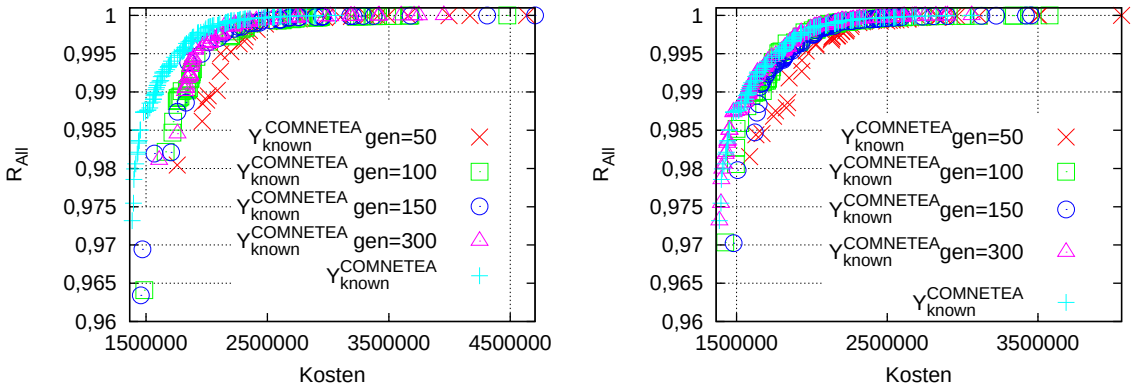


(e) deutsch30 $pop = 50$



(f) deutsch30 $pop = 100$

Abbildung 7.9: Gegenüberstellung der aggregierten Pareto-Fronten $\mathcal{Y}_{known}^{COMNETEA}$ im Vergleich zu den einzelnen Pareto-Fronten für unterschiedliche Populationsgrößen und Anzahlen der Generationen für die Netzwerke deutsch15, deutsch20 und deutsch30



(a) türkei19 $pop = 50$

(b) türkei19 $pop = 100$

Abbildung 7.10: Gegenüberstellung der aggregierten Pareto-Fronten $\mathcal{Y}_{known}^{COMNETEA}$ im Vergleich zu den einzelnen Pareto-Fronten für unterschiedliche Populationsgrößen und Anzahl der Generationen für das Netzwerk türkei19

Gleichzeitig wird deutlich, dass die Anzahl der Lösungen in der Ergebnisfront für größere Populationen (vgl. hierzu Tabelle 7.6) zunimmt. Hierdurch stehen für die lokale Suche mehr Lösungen zur Verfügung und es kann eine breitere Suche mit mehreren (unterschiedlichen) Suchrichtungen (bestimmt durch die Gewichte der skalierten Fitnessfunktion) erfolgen.
Neben einer rein grafischen Analyse der Ergebnisse erfolgt zusätzlich eine Analyse der erzeugten Pareto-Fronten mit Hilfe der in Abschnitt 7.2 vorgestellten Kennzahlen.
In Tabelle 7.6 auf S. 168 und Tabelle 7.7 auf S. 169 wird die Bewertung der mit den einzelnen Parameterkombinationen (*pop* und *gen*) erzeugten Pareto-Fronten anhand der Kennzahlen aus Abschnitt 7.2 gezeigt. Für jedes Netzwerk wird die in einer Parameterkombination ermittelte Pareto-Front einer aus allen Pareto-Fronten aggregierten Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$ gegenübergestellt.
Tabelle 7.6 zeigt für jedes untersuchte Netzwerk die verwendete Populationsgröße (*pop*), die Anzahl der durchlaufenen Generationen (*gen*), die Anzahl der Lösungen ($ONVG$) in der ermittelten Pareto-Front, die Anzahl der Lösungen, welche in der aggregierten Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$ enthalten sind, das Verhältnis ($OTNVG$) der Anzahl der Lösungen in $\mathcal{Y}_{known}^{COMNETEA}$ zu $ONVG$ und die Fehlerrate ($Error$). Mit Tabelle 7.7 wird die Analyse der einzelnen Pareto-Fronten im Vergleich zur aggregierten Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$ anhand der Distanzkennzahlen aus Abschnitt 7.2.2 vorgenommen. Für jedes Netzwerk wird die Populationsgröße (*pop*), die Anzahl der Generationen (*gen*), der mittlere Abstand ($dist$) der Lösungen der jeweils betrachteten Pareto-Front zur nächsten Lösung in der aggregierten Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$, der mittlere Abstand ($dist_1$) der Lösungen der aggregierten Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$ zur nächsten Lösung der jeweils betrachteten Pareto-Front, der maximale Abstand ($dist_2$) einer Lösung der aggregierten Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$ zur nächsten Lösung der jeweils betrachteten Pareto-Front sowie das Verhältnis ($dist_3$)von $dist_2$ zu $dist_1$ angegeben.

Zusätzlich wird mit Tabelle 7.7 die durchschnittliche Anzahl (gemittelt über alle fünf Durchläufe) der Fitnessbewertungen $\#Eval$ gezeigt.
Anhand der in Tabelle 7.6 auf S. 168 angegebenen Ergebnisse für $ONVG$ wird deutlich, dass die Anzahl der Lösungen in der durch COMNETEA generierten Pareto-Front mit zunehmender Populationsgröße und Anzahl der Generationen wächst. Die Kennzahl $OTNVG$ zeigt, dass (mit Ausnahme des Netzwerkes deutsch20, hier haben die Ergebnisse für $pop = 100, gen = 100$ einen sehr hohen Anteil an der aggregierten Pareto-Front) eine Vielzahl der Lösungen für $pop = 100, gen = 300$ in der aggregierten Pareto-Front enthalten sind.
Der Wert für $OTNVG$ bei $pop = 50$ und unterschiedlichem gen ist für sämtliche Netzwerke sehr klein. Dies bestätigt die bereits bei der grafischen Auswertung deutlich gewordene Dominanz der Lösungen für $pop = 100$ über die Lösungen $pop = 50$ bei der Bildung der aggregierten Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$.
Betrachtet man den mittleren Abstand ($dist$) der Lösungen für die jeweiligen Experimente zur nächsten Lösung in der aggregierten Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$, so zeigen die dargestellten Ergebnisse, dass dieser für $pop = 100$ und $gen = 300$ am geringsten ist. Die weiteren in Tabelle 7.7 dargestellten qualitativen Kennzahlen zeigen ebenfalls, dass der Abstand ($dist_1$) der aggregierten Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$ zu der Lösungfront für $pop = 100$ und $gen = 300$ für die Netzwerke deutsch15, deutsch30 und türkei19 am geringsten ist.
Für das Netzwerk deutsch20 besitzt die Pareto-Front für die Parameter $pop = 100$ und $gen = 100$ den geringsten Wert für $dist_1$. Die Pareto-Front, welche mit den Parametern $pop = 100$ und $gen = 300$ erzeugt wurde, besitzt einen nur geringfügig größeren Wert für $dist_1$. Gleichzeitig ist das Distanzmaß $dist_2$ dieser Pareto-Front geringer. Daher ergibt sich für die Pareto-Front bei $pop = 100, gen = 100$ ein schlechterer Wert für $dist_3$. Zieht man alle drei Ergebnisse zusammen, so ist festzuhalten, dass für das Netzwerk deutsch20 in der Parameterkombination $pop = 100, gen = 100$ ebenfalls das beste Ergebnis erzielt wurde, da die Lösungen der dort gefundenen Pareto-Front sämtliche Lösungen in der aggregierten $\mathcal{Y}_{known}^{COMNETEA}$ sehr gut repräsentieren (gemessen an $dist_3$) und gleichzeitig der durchschnittliche minimale Abstand ($dist_1$) einer Lösung in der aggregierten Pareto-Front zu einer Lösung in der Ergebnis-Front sehr gering ist.
Auffällig sind die sehr hohen Werte für $dist_3$ mit $pop = 100$ und $gen = 300$ für die Netzwerke deutsch15, deutsch30 und türkei19. Für alle drei Netzwerke ist $dist_3$ für diese Parameterkombination am höchsten. Ursache hierfür ist, dass all diese Pareto-Fronten zwar einen sehr geringen Wert für $dist_1$ besitzen, aber das Maximum des minimalen Abstandes ($dist_2$) einer Lösung der aggregierten Pareto-Front zur jeweiligen Lösungsfront im Vergleich dazu sehr hoch ist. Ursache hierfür sind pareto-optimale Lösungen in der aggregierten Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$ mit $R_{All}(G_N) < 0,8$, welche durch eine andere Parameterkombination gefunden wurden. Da die Pareto-Fronten keine Lösung in diesem Bereich des Lösungsraums besitzen, ergibt sich ein sehr hoher Abstand zu diesen Lösungen, der sich in $dist_2$ widerspiegelt. Betrachtet man den Berechnungsaufwand, der zur Erzeugung der jeweiligen Pareto-Fronten notwendig ist, so zeigen die mit Tabelle 7.7 dargestellten Werte für die Anzahl der Fitnessbewertungen, dass diese mit zunehmender Populationsgröße und steigender Anzahl der Generationen erwartungsgemäß wächst.

Tabelle 7.6: Gegenüberstellung der aggregierten Pareto-Fronten $\mathcal{Y}_{known}^{COMNETEA}$ im Vergleich zu den einzelnen Lösungsmengen für unterschiedliche Populationsgrößen und Generationsanzahl für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19 anhand der quantitativen Kennzahlen aus Abschnitt 7.2.1

Netzwerk	*pop*	*gen*	*ONVG*	*OTNVG*	*ONVGR*	*Error*
deutsch15	50	50	49	0	0,24	1
		100	63	0	0,30	1
		150	71	1	0,34	0,99
		300	75	0	0,36	1
	100	50	72	1	0,35	0,99
		100	133	3	0,64	0,98
		150	156	12	0,75	0,92
		300	202	191	0,97	0,06
deutsch20	50	50	43	0	0,42	1
		100	55	3	0,54	0,95
		150	72	0	0,71	1
		300	69	27	0,68	0,61
	100	50	44	0	0,43	1
		100	107	61	1,05	0,43
		150	103	4	1,01	0,96
		300	134	7	1,31	0,95
deutsch30	50	50	32	0	0,31	1
		100	32	0	0,31	1
		150	48	0	0,46	1
		300	71	0	0,68	1
	100	50	33	0	0,31	1
		100	57	3	0,54	0,95
		150	82	2	0,78	0,98
		300	103	100	0,98	0,03
türkei19	50	50	31	0	0,23	1
		100	57	0	0,42	1
		150	39	0	0,29	1
		300	64	0	0,47	1
	100	50	50	0	0,37	1
		100	79	6	0,58	0,92
		150	87	3	0,64	0,97
		300	135	128	0,99	0,05

Tabelle 7.7: Gegenüberstellung der aggregierten Pareto-Fronten $\mathcal{Y}_{known}^{COMNETEA}$ im Vergleich zu den einzelnen Lösungsmengen für unterschiedliche Populationsgrößen und Generationsanzahl für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19 anhand der qualitativen Kennzahlen aus Abschnitt 7.2.2

Netzwerk	*pop*	*gen*	*dist*	$dist_1$	$dist_2$	$dist_3$	$\#Eval$
deutsch15	50	50	0,0043	0,0282	0,221	7,8	26.112
		100	0,0031	0,0247	0,115	4,7	36.712
		150	0,0030	0,0365	0,143	3,9	47.259
		300	0,0035	0,0337	0,231	6,9	59.918
	100	50	0,0037	0,0205	0,124	6,1	37.304
		100	0,001	0,0093	0,092	9,8	77.925
		150	0,0011	0,0089	0,137	15,4	89.529
		300	0,0004	0,0019	0,103	55,1	128.349
deutsch20	50	50	0,0063	0,0419	0,232	5,6	24.128
		100	0,0044	0,0517	0,246	4,8	33.698
		150	0,0032	0,0179	0,094	5,2	56.765
		300	0,0037	0,0291	0,241	8,3	56.892
	100	50	0,0044	0,0274	0,296	10,8	28.814
		100	0,0018	0,0073	0,244	33,2	61.912
		150	0,0024	0,0143	0,187	13,1	88.039
		300	0,0016	0,0079	0,093	11,8	104.353
deutsch30	50	50	0,0176	0,1120	0,307	2,8	20.631
		100	0,0112	0,0691	0,579	8,4	26.214
		150	0,0075	0,0599	0,496	8,3	57.790
		300	0,0039	0,0714	0,828	11,6	58.388
	100	50	0,0111	0,0916	0,335	3,7	23.334
		100	0,0043	0,0169	0,095	5,6	38.414
		150	0,0024	0,0137	0,432	31,6	60.517
		300	0,0002	0,0050	0,461	92,1	82.872
türkei19	50	50	0,0264	0,0823	0,211	2,6	24.126
		100	0,0123	0,0475	0,310	6,5	83.701
		150	0,0182	0,0417	0,296	7,1	37.278
		300	0,0102	0,0503	0,642	12,8	52.895
	100	50	0,011	0,0439	0,256	5,8	32.307
		100	0,0047	0,0164	0,429	26,1	54.399
		150	0,0042	0,0225	0,175	7,8	64.009
		300	0,0001	0,0003	0,013	38,7	97.861

Subtrahiert man den Berechnungsaufwand, der durch NSGA-II verursacht wird (zirka $gen \cdot pop + gen \cdot pop \cdot p_{mut}$), ist für die „bereinigte“ Anzahl der Fitnessbewertungen noch immer ein Anstieg mit zunehmender Populationsgröße und Generationsanzahl festzustellen. Verursacht wird dieser durch die Anzahl der Lösungen, welche in der durch NSGA-II erzeugten Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ enthalten sind. Die Anzahl der Lösungen wächst für steigende *pop* und *gen*, so dass durch die lokale Suche mehr Fitnessbewertungen[77] erfolgen.

Gegenüberstellung der aggregierten Pareto-Fronten $\mathcal{Y}_{known}^{COMNETEA}$ und $\mathcal{Y}_{known}^{NSGA-II}$ zur Pareto-Front $\mathcal{Y}_{true}$

Abschließend erfolgt für COMNETEA eine Gegenüberstellung der experimentell erzielten Ergebnisse zu den in Abschnitt 7.3 präsentierten Ergebnissen. Für den Vergleich wurde für die Netzwerke deutsch15, deutsch20 und deutsch30 die Pareto-Front $\mathcal{Y}_{true}$ aus sämtlichen mittels COMNETEA ermittelten pareto-optimalen Lösungen und den NSGA-II-Lösungen aus Abschnitt 7.3 für das jeweilige Netzwerk gebildet[78]. Für das Netzwerk türkei19 setzt sich $\mathcal{Y}_{true}$ aus der in Abschnitt 7.3 verwendeten Pareto-Front $\mathcal{Y}_{true}$ erweitert um die mittels COMNETEA gefundenen Lösungen zusammen.
Neben einem Vergleich der mittels NSGA-II und COMNETEA gebildeten Pareto-Fronten erfolgt hier zusätzlich eine Analyse bezüglich des durch das Verfahren benötigten Berechnungsaufwandes. Von Interesse ist dabei die Möglichkeit, eine durch NSGA-II erzeugte Pareto-Front durch den Einsatz einer lokalen Suche (hier COMNETEA) so zu verbessern, dass eine sehr gute Annäherung an die Pareto-Front $\mathcal{Y}_{true}$ erfolgt.
Mit Tabelle 7.8 und Tabelle 7.9 werden die Pareto-Fronten $\mathcal{Y}_{known}^{COMNETEA}$ für die Parameterkombinationen $\{(pop = 50, gen = 100), (pop = 100, gen = 50), (pop = 100, gen = 300)\}$ und $\mathcal{Y}_{known}^{NSGA-II}$ (für $pop = 100$ und $gen = 300$) der Pareto-Front $\mathcal{Y}_{true}$ gegenübergestellt. Für den Vergleich wurden die Pareto-Fronten $\mathcal{Y}_{known}^{COMNETEA}$ für die Parameterkombinationen $\{ (pop = 50, gen = 100), (pop = 100, gen = 50)\}$ ausgewählt, da diese einen annähernd gleichen Berechnungsaufwand wie NSGA-II für $pop = 100, gen = 300$ verursachen. Zusätzlich wurde die Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$ für $pop = 100, gen = 300$ ausgewählt, da diese für die hier untersuchten Netzwerke die besten Ergebnisse geliefert hat. Jede der ermittelten Pareto-Fronten wird der aggregierten Pareto-Front $\mathcal{Y}_{true}$ des jeweiligen Netzwerkes gegenübergestellt.
Tabelle 7.8 zeigt für jede betrachtete Pareto-Front das untersuchte Netzwerk, die verwendete Populationsgröße (*pop*), die Anzahl der durchlaufenen Generationen (*gen*), die Anzahl der Lösungen ($ONVG$) in der jeweiligen Pareto-Front, die Anzahl der Lösungen der untersuchten Pareto-Front, welche in der Pareto-Front $\mathcal{Y}_{true}$ enthalten sind, das Verhältnis ($OTNVG$) der Anzahl der Lösungen in $\mathcal{Y}_{true}$ zu $ONVG$ und die Fehlerrate ($Error$). Mit Tabelle 7.7 wird die Analyse der einzelnen Pareto-Fronten im Vergleich zur aggregierten Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$ anhand der Kennzahlen aus Abschnitt 7.2.2 vorgenommen. Für jedes Netzwerk wird die Populationsgröße (*pop*), die Anzahl der

[77] Die Anzahl der Fitnessbewertungen ergibt sich aus $\mathcal{Y}_{known}^{NSGA-II} \cdot LS$.
[78] Die aggregierte Pareto-Front enthält dabei nur zueinander pareto-optimale Lösungen. Sämtliche Lösungen, welche durch andere Lösungen dominiert wurden, werden aus der Lösungsmenge gelöscht.

Tabelle 7.8: Gegenüberstellung der Pareto-Fronten $\mathcal{Y}_{known}^{COMNETEA}$ bei unterschiedlichen Populationsgrößen und Generationsanzahl sowie der Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ zur Pareto-Front $\mathcal{Y}_{true}$ für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19 anhand der quantitativen Kennzahlen aus Abschnitt 7.2.1

Netzwerk	Pareto-Front	*pop*	*gen*	*ONVG*	*OTNVG*	*ONVGR*	*Error*
deutsch15	$\mathcal{Y}_{known}^{COMNETEA}$	50	100	63	0	0,30	1
		100	50	72	1	0,34	0,99
		100	300	202	189	0,95	0,06
	$\mathcal{Y}_{known}^{NSGA-II}$	100	300	109	10	0,51	0,91
deutsch20	$\mathcal{Y}_{known}^{COMNETEA}$	50	100	55	0	0,32	1
		100	50	44	0	0,26	1
		100	300	134	3	0,78	0,98
	$\mathcal{Y}_{known}^{NSGA-II}$	100	300	162	159	0,94	0,02
deutsch30	$\mathcal{Y}_{known}^{COMNETEA}$	50	100	32	0	0,49	1
		100	50	33	0	0,5	1
		100	300	103	6	1,56	0,94
	$\mathcal{Y}_{known}^{NSGA-II}$	100	300	58	57	0,88	0,02
türkei19	$\mathcal{Y}_{known}^{COMNETEA}$	50	100	57	0	0,39	1
		100	50	50	0	0,34	1
		100	300	135	75	0,91	0,44
	$\mathcal{Y}_{known}^{NSGA-II}$	100	300	134	69	0,91	0,49

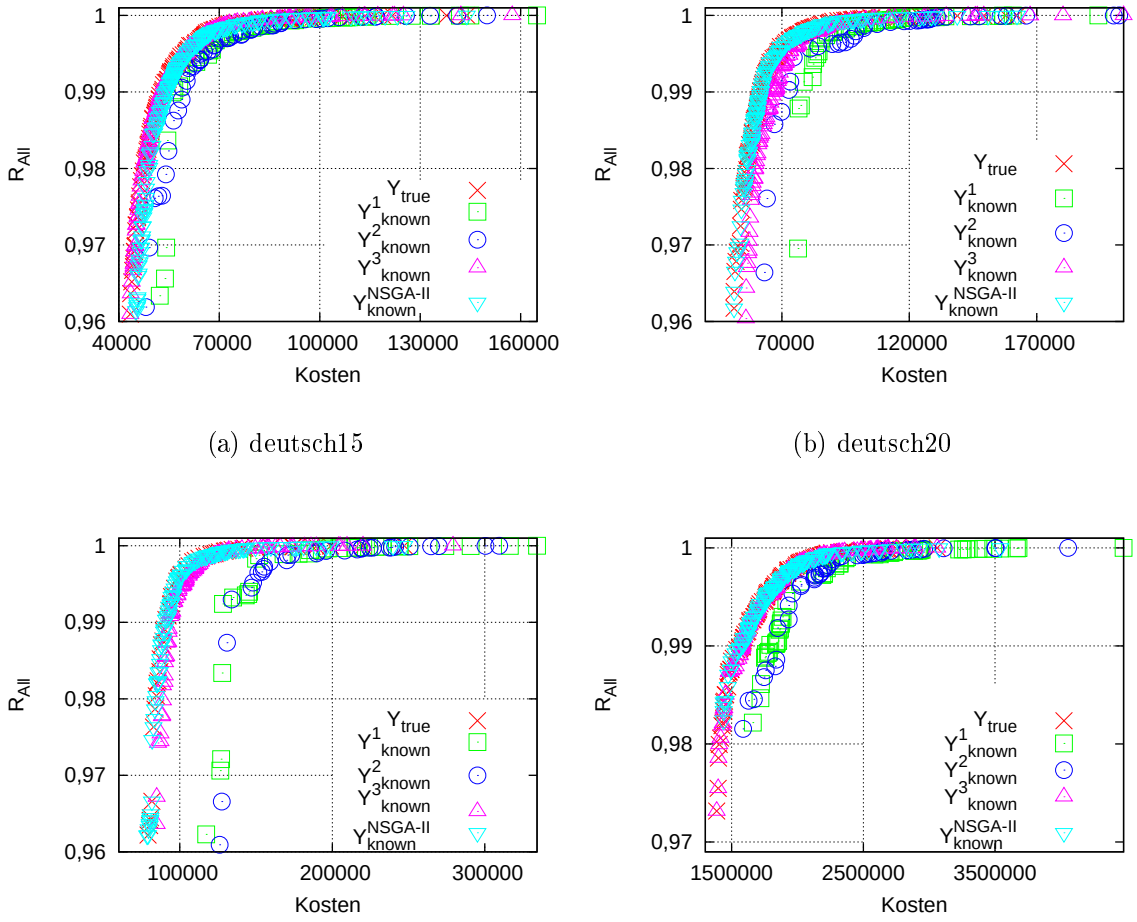
Generationen (*gen*), der mittlere Abstand (*dist*) der Lösungen der jeweils betrachteten Pareto-Front zur nächsten Lösung in der Pareto-Front $\mathcal{Y}_{true}$, der mittlere Abstand ($dist_1$) der Lösungen der aggregierten Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$ zur nächsten Lösung der jeweils betrachteten Pareto-Front, der maximale Abstand ($dist_2$) einer Lösung der aggregierten Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$ zur nächsten Lösung der jeweils betrachteten Pareto-Front sowie das Verhältnis ($dist_3$) von $dist_2$ zu $dist_1$ angegeben. Zusätzlich wird mit Tabelle 7.7 die durchschnittliche Anzahl (gemittelt über alle fünf Durchläufe) der Fitnessbewertungen $\#Eval$ gezeigt.
Betrachtet man die Anzahl der Lösungen (gemessen an $ONVG$) der einzelnen Pareto-Fronten in Tabelle 7.8, so wird deutlich, dass für $pop = 100$ und $gen = 300$ die Pareto-Fronten mit der höchsten Anzahl an Lösungen erzeugt wurden. Anhand der Kennzahl $OTNVG$ ist ersichtlich, dass für das Netzwerk deutsch15 mittels COMNETEA für $pop = 100$ und $gen = 300$ ein großer Anteil der Lösungen in $\mathcal{Y}_{true}$ gefunden wurde. Im Gegensatz dazu hat für das Netzwerk deutsch20 die mittels NSGA-II erstellte Pareto-Front den größten Anteil an $\mathcal{Y}_{true}$. Für das Netzwerk deutsch30 sind für COMNETEA bei $pop = 100$ und $gen = 300$ nur sechs der 103 gefundenen pareto-optimalen Lösungen in $\mathcal{Y}_{true}$ enthalten.

Tabelle 7.9: Gegenüberstellung der Pareto-Fronten $\mathcal{Y}_{known}^{COMNETEA}$ bei unterschiedlichen Populationsgrößen und Generationsanzahl sowie der Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ zur Pareto-Front $\mathcal{Y}_{true}$ für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19 anhand der qualitativen Kennzahlen aus Abschnitt 7.2.2

Netzwerk	Pareto-Front	*pop*	*gen*	*dist*	$dist_1$	$dist_2$	$dist_3$	*#Eval*
deutsch15	$\mathcal{Y}_{known}^{COMNETEA}$	50	100	0,003	0,024	0,115	4,7	36.712
		100	50	0,004	0,021	0,124	6,0	37.034
		100	300	0,001	0,002	0,103	54,1	128.349
	$\mathcal{Y}_{known}^{NSGA-II}$	100	300	0,001	0,016	0,433	26,8	30.914
deutsch20	$\mathcal{Y}_{known}^{COMNETEA}$	50	100	0,006	0,061	0,246	4,0	33.698
		100	50	0,009	0,031	0,296	9,5	28.814
		100	300	0,002	0,010	0,093	9,3	104.353
	$\mathcal{Y}_{known}^{NSGA-II}$	100	300	0,001	0,014	0,84	59,2	30.734
deutsch30	$\mathcal{Y}_{known}^{COMNETEA}$	50	100	0,015	0,082	0,579	7,1	26.214
		100	50	0,014	0,105	0,335	3,2	23.334
		100	300	0,002	0,016	0,461	29,8	82.872
	$\mathcal{Y}_{known}^{NSGA-II}$	100	300	0,001	0,031	0,883	28,4	30.832
türkei19	$\mathcal{Y}_{known}^{COMNETEA}$	50	100	0,012	0,050	0,31	6,2	83.701
		100	50	0,011	0,047	0,256	5,4	54.399
		100	300	0,001	0,003	0,018	7,3	97.861
	$\mathcal{Y}_{known}^{NSGA-II}$	100	300	0,002	0,010	0,662	66,6	30.878

Für das Netzwerk türkei19 sind sowohl für COMNETEA mit $pop = 100$ und $gen = 300$ als auch für NSGA-II eine annähernd gleiche Anzahl an Lösungen aus den Pareto-Fronten $\mathcal{Y}_{known}$ in $\mathcal{Y}_{true}$ enthalten.
Für die Pareto-Fronten, welche mit $pop = 50$ und $gen = 100$ sowie $pop = 100$ und $gen = 50$ erstellt wurden, ist anhand der in Tabelle 7.8 dargestellten Ergebnisse ersichtlich, dass diese fast keine Lösung aus $\mathcal{Y}_{true}$ enthalten. Lediglich für das Netzwerk deutsch15 ist für $pop = 100$ und $gen = 50$ eine Lösung in $\mathcal{Y}_{true}$ enthalten. Für alle anderen Pareto-Fronten werden die Lösungen in $\mathcal{Y}_{known}^{COMNETEA}$ für die Parameterkombinationen durch Lösungen in $\mathcal{Y}_{true}$ dominiert. Anhand der Kennzahl $ONVGR$ wird weiterhin deutlich, dass die Pareto-Fronten $\mathcal{Y}_{known}^{COMNETEA}$ für $pop = 50$ und $gen = 100$ sowie $pop = 100$ und $gen = 50$ einen Wert $ONVGR \leq 0,5$ besitzen. Daraus folgt, dass die Anzahl der Lösungen in $\mathcal{Y}_{true}^{COMNETEA}$ nur halb so groß ist wie die Anzahl der Lösungen in $\mathcal{Y}_{true}$.
Die qualitativen Kennzahlen in Tabelle 7.9 zeigen, dass der Abstand der Lösungen für COMNETEA mit den Parametern $pop = 50$ und $gen = 100$ sowie $pop = 100$ und $gen = 50$ für alle untersuchten Netzwerke annähernd gleich groß ist. Stellt man diesen Ergebnissen die Werte der Distanzkennzahlen für COMNETEA und des NSGA-II Ver-

fahrens (bei $pop = 100$ und $gen = 300$) gegenüber, so ist zu erkennen, dass die Pareto-Fronten für $pop = 50$ und $gen = 100$ sowie $pop = 100$ und $gen = 50$ einen deutlich größeren Abstand zu $\mathcal{Y}_{true}$ besitzen.
Eine Analyse der Distanzkennzahlen für COMNETEA und NSGA-II bei $pop = 100$ und $gen = 300$ zeigt, dass die Lösungen der Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ sehr nah an Lösungen in $\mathcal{Y}_{true}$ liegen. Gleichzeitig wird aber auch deutlich, dass die Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$ sämtliche Lösungen in $\mathcal{Y}_{true}$ besser repräsentiert. Für das Netzwerk deutsch15 zeigt der Wert für $dist$ an, dass sowohl für die Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$ als auch für die Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ eine annähernd gleiche Lösung in $\mathcal{Y}_{true}$ existiert. Die geringeren Werte für $dist_1$ und $dist_2$ zeigen jedoch an, dass sämtliche Lösungen aus $\mathcal{Y}_{true}$ durch die Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$ besser repräsentiert werden. Für das Netzwerk deutsch20 besitzt die Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ den geringsten Distanzwert für $dist$. Das heißt, dass für sämtliche Lösungen in $\mathcal{Y}_{known}^{NSGA-II}$ eine annähernd gleiche Lösung in $\mathcal{Y}_{true}$ existiert. Anhand von $dist_1$ und $dist_2$ wird aber ersichtlich, dass $\mathcal{Y}_{known}^{NSGA-II}$ nicht sämtliche Lösungen aus $\mathcal{Y}_{true}$ abdeckt. Die Distanzkennzahlen $dist_1$ und $dist_2$ zeigen, dass die Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$ dem Ziel, eine gute Annäherung an $\mathcal{Y}_{true}$ zu erreichen und dabei sämtliche Lösungen in $\mathcal{Y}_{true}$ durch Lösungen in $\mathcal{Y}_{known}$ zu repräsentieren, besser gerecht wird. Für das Netzwerk deutsch30 weist die Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ ebenfalls den geringsten Wert für die Distanzkennzahl $dist$ auf. Wie bereits zuvor bei dem Netzwerk deutsch20 herausgestellt, muss aber hier darauf hingewiesen werden, dass die Distanzkennzahlen $dist_1$ und $dist_2$ zeigen, dass $\mathcal{Y}_{known}^{NSGA-II}$ nur einen Teilbereich der Lösungen aus $\mathcal{Y}_{true}$ abdeckt. Die Lösungen auf $\mathcal{Y}_{known}^{COMNETEA}$ sind zwar im Durchschnitt weiter von einer annähernd gleichen Lösung in $\mathcal{Y}_{true}$ entfernt (gemessen an $dist$). Wie die Distanzkennzahlen $dist_1$ und $dist_2$ aber zeigen, werden sämtliche Lösungen aus $\mathcal{Y}_{true}$ durch $\mathcal{Y}_{known}^{COMNETEA}$ besser repräsentiert. Für das Netzwerk türkei19 ist die Pareto-Front $\mathcal{Y}_{known}^{COMNETEA}$ der Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ klar überlegen. Für alle betrachteten Distanzkennzahlen weist $\mathcal{Y}_{known}^{COMNETEA}$ bessere Werte auf.
Für alle hier untersuchten Netzwerke kann zusammenfassend festgestellt werden, dass durch den Einsatz einer lokalen Suche zur Verbesserung der Lösungen des NSGA-II, wie es mit COMNETEA umgesetzt wurde, eine qualitativ hochwertigere Pareto-Front erstellt werden kann. Gleichzeitig muss dabei aber auch der für die Erreichung der Ergebnisse notwendige Aufwand betrachtet werden. Tabelle 7.9 zeigt anhand der Fitnessbewertungen ($\#Eval$), dass COMNETEA einen bis zu vierfach höheren Berechnungsaufwand als der reine NSGA-II verursacht.
Stellt man die Ergebnisse für die Pareto-Fronten $\mathcal{Y}_{known}^{COMNETEA}$ für $pop = 50$ und $gen = 100$ sowie $pop = 100$ und $gen = 50$, welche einen annähernd gleichen Berechnungsaufwand wie ein „unmodifizierter“ NSGA-II mit $pop = 100$ und $gen = 300$ verursachen, gegenüber, so ist festhalten, dass es für die hier untersuchten Netzwerke nicht möglich war, durch den Einsatz der nachgelagerten lokalen Suche die Lösungen einer Pareto-Front $\mathcal{Y}_{known}$ zu verbessern und dabei den Berechnungsaufwand für das NSGA-II-Verfahren zu reduzieren.



(a) deutsch15

(b) deutsch20

(c) deutsch30

(d) türkei19

Abbildung 7.11: Gegenüberstellung der Pareto-Fronten $\mathcal{Y}_{known}^{COMNETEA}$ mit $pop = 50$ und $gen = 100$ ($\mathcal{Y}_{known}^{1}$), $pop = 100$ und $gen = 50$ ($\mathcal{Y}_{known}^{2}$) und $pop = 100$ und $gen = 200$ ($\mathcal{Y}_{known}^{3}$) sowie der Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ zur Pareto-Front $\mathcal{Y}_{true}$ für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19

Abbildung 7.11 stellt abschließend die Pareto-Fronten $\mathcal{Y}_{true}$, $\mathcal{Y}_{known}^{NSGA-II}$ und $\mathcal{Y}_{known}^{COMNETEA}$ gegenüber[79]. Die Pareto-Front $\mathcal{Y}_{known}^{1}$ stellt die Ergebnisse für COMNETEA mit $pop = 50$ und $gen = 100$ dar. Die Pareto-Front $\mathcal{Y}_{known}^{2}$ zeigt die Ergebnisse für COMNETEA mit $pop = 100$ und $gen = 50$. Mit der Pareto-Front $\mathcal{Y}_{known}^{3}$ werden die Ergebnisse für COMNETEA mit $pop = 100$ und $gen = 300$ gezeigt.

[79] Für eine übersichtlichere Darstellung werden die gezeigten Pareto-Fronten kurz als $\mathcal{Y}_{known}^{1}$, $\mathcal{Y}_{known}^{2}$ und $\mathcal{Y}_{known}^{3}$ bezeichnet.

Die in Abbildung 7.11 dargestellten Pareto-Fronten bestätigen die bereits bei der Analyse mit Hilfe der quantitativen und qualitativen Kennzahlen herausgestellten Ergebnisse. Die Lösungen der Pareto-Fronten $\mathcal{Y}_{known}^{COMNETEA}$ für $pop = 50, 100$ und $gen = 50, 100$ werden für alle vier der hier untersuchten Netzwerke durch Lösungen der anderen Pareto-Fronten dominiert.
Betrachtet man die Abbildungen 7.11(a), 7.11(b) und 7.11(c), so wird deutlich, dass mit zunehmender Problemgröße der Abstand zwischen $\mathcal{Y}_{true}$ und $\mathcal{Y}_{known}^{COMNETEA}$ für $pop = 50, 100$ und $gen = 50, 100$ wächst. Die Abbildungen zeigen, dass die dargestellten Pareto-Fronten $\mathcal{Y}_{known}^{NSGA-II}$ und $\mathcal{Y}_{known}^{3}$ für alle untersuchten Netzwerke sehr nah beieinander liegen.
Anhand von Abbildung 7.11(d) werden die analytisch erzielten Ergebnisse für das Netzwerk türkei19 bestätigt. Es ist sehr gut ersichtlich, dass $\mathcal{Y}_{known}^{NSGA-II}$ für $0,98 \leq R_{All} \leq 1$ die Pareto-Front $\mathcal{Y}_{true}$ sehr gut abdeckt. Insgesamt stellt die Pareto-Front $\mathcal{Y}_{known}^{3}$, welche die Ergebnisse für COMNETEA bei $pop = 100$ und $gen = 300$ darstellt, aber eine bessere Annäherung an die Pareto-Front $\mathcal{Y}_{true}$ dar.

7.4.2 TPNDA – Ein multikriterieller Dekompositionsansatz für die Planung von Kommunikationsnetz-werken mit den Zielen Zuverlässigkeit und Kosten

Mit Abschnitt 7.4.1 wurde ein Verfahren vorgestellt, dass die Lösungen, welche durch einen multikriteriellen evolutionären Algorithmus erzeugt wurden, mit Hilfe einer nachgelagerten lokalen Suche verbessert und dabei eine bessere Annäherung an eine „wahre“ Pareto-Front $\mathcal{Y}_{true}$ erreicht. Mit diesem Abschnitt wird das multikriterielle Planungsverfahren Two Phase Network Design Algorithm (TPNDA) vorgeschlagen, das die lokale Suche direkt in den evolutionären Algorithmus integriert.
Für TPNDA wird die bereits mit Abschnitt 5.2.2.2 umgesetzte Baldwin'sche Evolutionstheorie in einem multikriteriellen Dekompositionsansatz für die Kommunikationsnetzwerkplanung mit den Zielen Zuverlässigkeit und Kosten umgesetzt. Als multikriterieller evolutionärer Algorithmus wurde NSGA-II gewählt. Wie bereits für das Verfahren BaBORNet (vgl. Abschnitt 5.2.2.2, S. 108) wird der Planungsprozess in zwei Teilprobleme, die Topologieplanung und die Auswahl geeigneter Technologieoptionen für die gewählte Topologie, zerlegt. Die Aufgabe der Topologieplanung wird für TPNDA durch NSGA-II gelöst. Die Aufgabe der Auswahl der Technologie für die generierten Topologien erfolgt durch eine lokale Suche.
Aufgrund der Trennung von Topologiedesign und der Wahl der Technologieoptionen besitzt der evolutionäre Algorithmus keine Information darüber, welche Technologieoption für eine Verbindung gewählt wird. Eine Lösung wird daher für TPNDA als Binärvektor der Länge $|E|$ kodiert. Abbildung 7.12 zeigt die Kodierung der Lösung aus Abbildung 7.7 auf S. 161 als Genom für TPNDA. Für jede Kante aus G, die in der kodierten Lösung verwendet wird, gilt $g_i = 1$. Für alle anderen Kanten ist $g_i = 0$.
Für TPNDA wurde der auf S. 59 beschriebene NSGA-II (Algorithmus 3.7) um ein lokales Suchverfahren erweitert. Den kompletten Ablauf von TPNDA zeigt Algorithmus 7.2. Der Standardablauf des NSGA-II wurde in Zeile 7 um eine lokale Suche erweitert, welche für jede Lösung aus P_t'' aufgerufen wird. Durch die lokale Suche werden für die Lösung

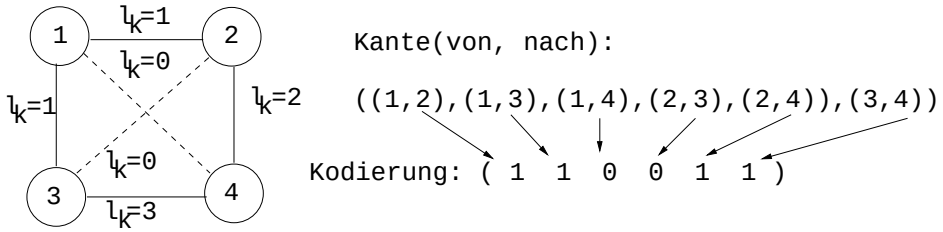


Abbildung 7.12: Kodierung einer Lösung für TPNDA

Algorithmus 7.2 TPNDA

```
1:  t=0
2:  erzeuge initiale Population P_t
3:  Fast-Nondominated-Sort P_t
4:  Erzeuge P'_t aus P_t
5:  repeat
6:     P''_t = P'_t ∪ P_t
7:     for all y_work ∈ P''_t do
8:        y' = lokaleSuche(y_work)
9:     end for
10:    Fast-Nondominated-Sort P''_t
11:    i = 1
12:    while (|P_{t+1}| + |F_i|) < pop do
13:       P_{t+1} ∪ F_i
14:       i= i +1
15:    end while
16:    sort(F_i, ≥_n)
17:    P_{t+1} = F_i[0 : pop − |P_{t+1}|]
18:    Erzeuge P'_{t+1} aus P_{t+1}
19:    t = t + 1
20: until Abbruchkriterium
```

y_{work} die Technologieoptionen für die mit der Lösung kodierten Topologie bestimmt. Die dabei ermittelten Zielfunktionswerte für $R_{All}(G_N)$ und $C(G_N)$ der Lösung y' werden der Lösung y_{work} zugewiesen.
Algorithmus 7.3 stellt den Ablauf der lokalen Suche für TPNDA dar. Im Gegensatz zu COMNETEA werden bei der lokalen Suche von TPNDA die Gewichte der beiden Ziele nicht berechnet, sondern zufällig bestimmt[80]. Dieser Ansatz wurde gewählt, da im Gegensatz zu COMNETEA bei TPNDA zu Beginn der lokalen Suche noch keine Informationen über die All-Terminal-Zuverlässigkeit und die Kosten der durch NSGA-II erzeugten Lösung vorliegen.

[80] Die Gewichte werden zusätzlich normiert, so dass gilt: $w_{R_{All}} + w_C = 1$.

Algorithmus 7.3 TPNDA - lokale Suche

Eingabe: y_{work}, $steps$
Ausgabe: y_{work}

```
1: wähle w_{R_All} und w_C zufallsbasiert
2: berechne f(y_work)
3: counter = steps
4: while counter > 0 do
5:    y_Nach = z(y_work)
6:    berechne f(y_Nach)
7:    if f(y_Nach) < f(y_work) then
8:       y_work = y_Nach
9:       counter = steps
10:   else
11:      counter = counter − 1
12:   end if
13: end while
```

Wie bereits in Abschnitt 7.4.1 für COMNETEA dargestellt, erfolgt die lokale Suche unter Verwendung einer skalaren Zielfunktion, in der beide Ziele mittels Gewichten zu einem Zielfunktionswert zusammengefasst werden. Die Zielfunktion für die lokale Suche von TPNDA lautet:

$$f(y) = w_C \cdot \frac{C(G_N)}{C^{max}_{\mathcal{Y}^{NSGA-II}_{known}} - C^{min}_{\mathcal{Y}^{NSGA-II}_{known}}} + w_{R_{All}} \cdot (1 - \frac{R_{All}(G_N)}{R_{All}{}^{max}_{\mathcal{Y}^{NSGA-II}_{known}} - R_{All}{}^{min}_{\mathcal{Y}^{NSGA-II}_{known}}}) \rightarrow \min \tag{7.13}$$

Mit Hilfe der Gewichte wird der skalare Zielfunktionswert $f(y_{work})$ berechnet[81].
Für jede Lösung y_{work} wird durch die iterative Anwendung eines Zuges $z(y_{work})$ eine neue Lösung y_{Nach} in der Nachbarschaft von y_{work} erzeugt. Der Zug $z(y_{work})$ verändert hierfür zufällig zwei Technologieoptionen der Lösung y_{work}. Für y_{Nach} wird ebenfalls der skalare Zielfunktionswert berechnet und mit dem Zielfunktionswert für die Lösung y_{work} verglichen. Wenn $f(y_{Nach}) < f(y_{work})$ ist, wird y_{work} durch y_{Nach} ersetzt und die lokale Suche fortgesetzt. Anders als COMNETEA verwendet TPNDA keine globale Schranke, welche die maximale Anzahl der zu untersuchenden Nachbarschaften begrenzt. Für TPNDA bestimmt der Parameter *steps* die Anzahl der für eine Lösung zu untersuchenden Nachbarschaftslösungen. Wird in den *steps* Iterationen durch den Zug $z(y_{work})$ in der Nachbarschaft der aktuell betrachteten Lösung eine Lösung mit einem besseren Zielfunktionswert gefunden, so werden für die neue beste Lösung maximal *steps* benachbarte Lösungen, welche durch den Zug $z(y_{work})$ erzeugt werden, untersucht. Konnte keine bessere Lösung in der Nachbarschaft von y_{work} gefunden werden, so bricht die lokale Suche ab und gibt die aktuell beste Lösung y_{work} mit den durch die lokale Suche gewählten Technologieoptionen zurück.
Das NSGA-II-Verfahren kann anhand des Ergebnisses aus der lokalen Suche die Zielfunktionswerte C und R_{All} für y' ermitteln (vgl. Algorithmus 7.2 Zeile 8) und dem Genom

[81] Die Zielfunktionswerte wurden, wie in Abschnitt 7.4.1 für COMNETEA dargestellt, normiert.

zuweisen. Während des NSGA-II-Ablaufs wird jede Topologie, welche in dem Genom kodiert ist und durch das evolutionäre Verfahren bestimmt wurde, anhand einer auf dieser Topologie aufbauenden Auswahl an Technologieoptionen und den daraus resultierenden Kosten und der jeweiligen All-Terminal-Zuverlässigkeit bewertet. Die durch die gewählten Technologieoptionen ermittelten Zielfunktionswerte bestimmen die Position einer Lösung bei der NSGA-II-Frontenbildung und beeinflussen somit die Auswahlwahrscheinlichkeit für die Übernahme einer Lösung in die nächste Generation. Da NSGA-II keine Informationen darüber besitzt, welche Technologieoptionen für die im Genom kodierte Topologie gewählt wurde, arbeiten der Rekombinations- und Mutationsoperator lediglich auf der Topologieebene und verändern dabei die Anzahl der Verbindungen in einer Lösung.
Zur Generierung einer Lösungsmenge wird ein globales Archiv pareto-optimaler Lösungen verwendet, in dem jede während der lokalen Suche generierte Lösung (mit den gewählten Technologieoptionen) gespeichert wird. Für jede dem Archiv neu hinzugefügte Lösung wird geprüft, ob diese durch andere Lösungen, welche bereits in dem Archiv enthalten sind, dominiert wird. Ist dies der Fall, wird die Lösung dem Archiv nicht hinzugefügt. Dominiert die Lösung hingegen Lösungen, welche bereits im Archiv enthalten sind oder ist sie zu diesen pareto-optimal, so wird sie dem Archiv hinzugefügt und gegebenenfalls dominierte Lösungen aus dem Archiv entfernt.

7.4.2.1 Experimentelles Design für die Untersuchung von TPNDA

Eine empirische Studie untersucht TPNDA für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19.
Dabei wird zwischen zwei Versionen unterschieden: einer Variante mit Initialisierung der Technologieoptionen (kurz als TPNDA m.I. bezeichnet) und einer Variante ohne Initialisierung der Technologieoptionen (kurz als TPNDA o.I. bezeichnet). Für TPNDA m.I. wird für jede Kante einer Lösung y_{work} vor dem Beginn der lokalen Suche zufällig eine der zur Verfügung stehenden Technologieoptionen ausgewählt. In der Variante TPNDA o.I. wird für sämtliche Verbindungen initial die Technologieoption mit den geringsten Kosten gewählt (d. h. $\forall e \in E_N : l(e) = 1$). Die Lösungen der initialen Population P_0 wurden zufallsbasiert erzeugt. Hierfür wurden für jede Lösung aus einem vollständig verbundenen Graphen eine zufällig bestimmte Anzahl an Verbindungen gelöscht. Als Mutationsoperator wird für die experimentelle Untersuchung eine Flip-Mutation (siehe S. 48) mit einer Mutationswahrscheinlichkeit von $p_{mut} = 0,02$ verwendet. Für die Rekombination wird ein Uniform-Crossover-Operator (siehe S. 47) mit einer Crossoverwahrscheinlichkeit von $p_{cross} = 1$ verwendet. In der experimentellen Untersuchung wurde der Parameter *steps* für die lokale Suche auf zehn festgelegt. Aufgrund der Vielzahl von Fitnessbewertungen, welche durch TPNDA mit der lokalen Suche durchzuführen sind, wurden für die Experimente eine kleine Populationsgröße sowie eine geringe Anzahl an zu durchlaufenden Generationen gewählt. Für die Netzwerke deutsch15, deutsch20 und deutsch30 wurde mit einer Populationsgröße $pop = 50$ und einer Generationsanzahl von $gen = 300$ gearbeitet. Das Netzwerk türkei19 wurde aufgrund der höheren Zuverlässigkeiten mit einer Populationgröße $pop = 100$ und $gen = 100$ untersucht. Ein Durchlauf wurde stets

Tabelle 7.10: Parameter für die experimentelle Untersuchung mit TPNDA

GA	NSGA-II
Crossover-Operator	Uniform-Crossover, $p_{cross} = 1$
Mutations-Operator	Flip-Mutation, $p_{mut} = 0,02$
Initialisierung von P_0	Zufallsbasiertes Löschen von Kanten in einem vollständigen Graphen
Bestimmung von R_{All}	Einfache Monte-Carlo-Simulation
pop	50 (deutsch15, deutsch20 und deutsch30) 100 (türkei19)
gen	100
Anzahl GA-Läufe	5

nach der maximalen Anzahl an Generationen *gen* beendet. Tabelle 7.10 fasst die für die empirische Untersuchung verwendeten Parameter zusammen.

7.4.2.2 Experimentelle Ergebnisse und Auswertung

Mit diesem Abschnitt werden die Ergebnisse der empirischen Untersuchung des TPNDA vorgestellt. Es werden die Lösungen gezeigt, welche mit TPNDA o.I und TPNDA m.I. generiert wurden, und einander gegenübergestellt. Außerdem findet ein Vergleich der durch TPNDA gebildeten Pareto-Front im Vergleich zu den in Abschnitt 7.3 vorgestellten Pareto-Fronten, welche ein unmodifizierter NSGA-II erzeugte, statt. Zum Abschluss des Abschnitts wird der Berechnungsaufwand bewertet, der mit den Einsatz von TPNDA entsteht.

Gegenüberstellung der Verfahren TPNDA o.I. und TPNDA m.I.

Zunächst wird eine Gegenüberstellung von TPNDA o.I. und TPNDA m.I. anhand der Ergebnisse vorgenommen, welche für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19 erzielt wurden. Mit Tabelle 7.11 auf S. 180 und Tabelle 7.12 auf S. 181 werden die Kennzahlen der experimentell ermittelten Ergebnisse dargestellt. Für die Auswertung wurden die Ergebnisse aus den einzelnen Durchläufen jeweils zu einer gemeinsamen Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ bzw. $\mathcal{Y}_{known}^{TPNDA\ o.I}$ aggregiert. Aus den Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ o.I}$ wurden die jeweiligen Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA}$ gebildet, welche für den analytischen und grafischen Vergleich zwischen beiden Verfahrensvarianten herangezogen werden.
Tabelle 7.11 zeigt das untersuchte Netzwerk, die jeweils betrachtete Pareto-Front, die Anzahl der Lösungen in der betrachteten Pareto-Front ($ONVG$), die Anzahl der Lösungen der betrachteten Pareto-Front, welche in $\mathcal{Y}_{known}^{TPNDA}$ enthalten sind, das Verhältnis zwischen der Anzahl der Lösungen der betrachteten Pareto-Front zur Anzahl der Lösungen in $\mathcal{Y}_{known}^{TPNDA}$ ($ONVGR$) sowie den Anteil ($Error$) der Lösungen der betrachteten Pareto-Front, welche nicht in $\mathcal{Y}_{known}^{TPNDA}$ enthalten sind.

Tabelle 7.11: Gegenüberstellung der Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ zur aggregierten Pareto-Front $\mathcal{Y}_{known}^{TPNDA}$ für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19 anhand der quantitativen Kennzahlen aus Abschnitt 7.2.1

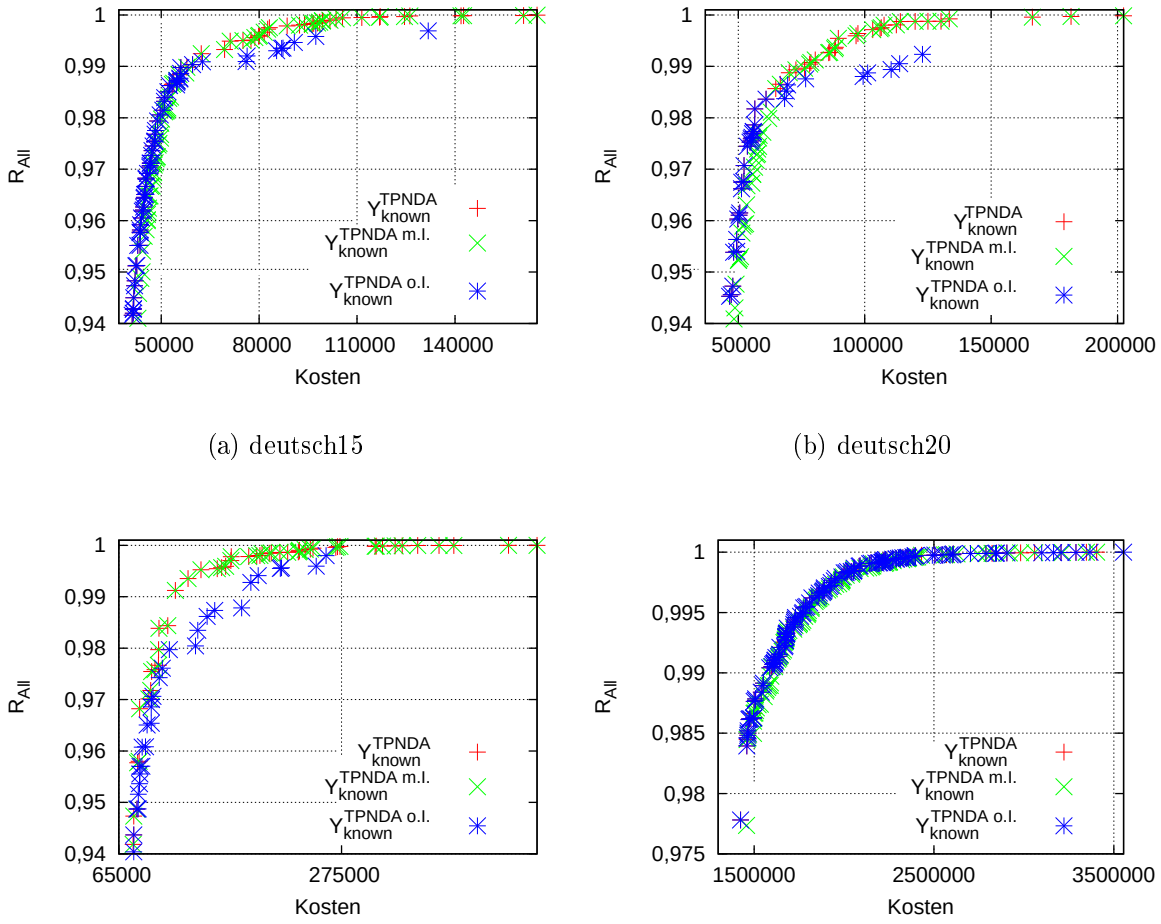
Netzwerk	Pareto-Front	$ONVG$	$OTNVG$	$ONVGR$	$Error$
deutsch15	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	150	37	0,655	0,753
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	207	193	0,904	0,068
deutsch20	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	135	36	0,918	0,733
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	123	111	0,837	0,098
deutsch30	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	99	62	0,861	0,374
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	91	53	0,791	0,418
türkei19	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	114	21	0,912	0,816
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	121	108	0,968	0,107

Anhand der Kennzahl $ONVG$ wird deutlich, dass für die Netzwerke türkei19, deutsch20 und deutsch30 mit beiden Varianten von TPNDA eine annähernd gleiche Anzahl an Lösungen in der jeweils generierten Pareto-Front enthalten ist. Lediglich für das kleinste Netzwerk deutsch15 werden mit TPNDA o.I. deutlich mehr (30 %) pareto-optimale Lösungen als mit TPNDA m.I. generiert.
Die Kennzahl $OTNVG$ zeigt an, dass für die Netzwerke deutsch15, deutsch20 und türkei19 die Pareto-Front $\mathcal{Y}_{known}^{TPNDA}$ aus einer Vielzahl von Lösungen besteht, welche mit TPNDA o.I. erzeugt wurden. Durch die Variante TPNDA m.I. konnten nur für das Netzwerk deutsch30 mehr Lösungen zu $\mathcal{Y}_{known}^{TPNDA}$ beigesteuert werden. Gleichzeitig ist die Anzahl der Lösungen in $\mathcal{Y}_{known}^{TPNDA}$, welche durch die Variante TPNDA o.I. gefunden wurden, fast gleich hoch.
Das Verhältnis $ONVGR$ zwischen der Anzahl der Lösungen in der jeweiligen TPNDA-Pareto-Front im Vergleich zur Anzahl der Lösungen in $\mathcal{Y}_{known}^{TPNDA}$ ist für beide Varianten von TPNDA bei den Netzwerken deutsch20, deutsch30 und türkei19 annähernd gleich groß. Nur für das Netzwerk deutsch15 enthält die Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ deutlich weniger Lösungen als $\mathcal{Y}_{known}^{TNPDA}$.
Zusätzlich zu den in Tabelle 7.11 dargestellten quantitativen Kennzahlen erfolgt mit Tabelle 7.12 eine Betrachtung der erzeugten Pareto-Fronten mit Hilfe der in Abschnitt 7.2.2 vorgestellten qualitativen Kennzahlen.
Mit Tabelle erfolgt für jede Pareto-Front eine Betrachtung des Abstandes ($dist$) der Lösungen aus $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ bzw. $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ zur nächsten Lösung in $\mathcal{Y}_{known}^{TPNDA}$ sowie eine Analyse mittels der Distanzmaße $dist_1$, $dist_2$ und $dist_3$, welche den Abstand der Lösungen in $\mathcal{Y}_{known}^{TPNDA}$ zur nächsten Lösung in $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ bzw. $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ messen.
Anhand der Distanzkennzahl $dist$ ist ersichtlich, dass die Lösungen der von TPNDA in beiden Varianten erzeugten Pareto-Fronten stets eine sehr nahe Lösung in $\mathcal{Y}_{known}^{TPNDA}$ besitzen. Für das Netzwerk deutsch15, für das bereits bei der Betrachtung der quantitativen Kennzahlen deutlich wurde, dass von TPNDA m.I. mehr Lösungen in $\mathcal{Y}_{known}^{TPNDA}$ enthalten sind, ist in Tabelle 7.12 auf S. 181 anhand von $dist$ ebenfalls ein größerer Ab-

Tabelle 7.12: Gegenüberstellung der Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ zur aggregierten Pareto-Front $\mathcal{Y}_{known}^{TPNDA}$ für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19 anhand der qualitativen Kennzahlen aus Abschnitt 7.2.2

Netzwerk	Pareto-Front	$dist$	$dist_1$	$dist_2$	$dist_3$
deutsch15	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	0,00123	0,0190	0,153	8,06
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	0,00024	0,0064	0,231	36,2
deutsch20	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	0,00059	0,0064	0,045	6,99
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	0,00031	0,0142	0,455	31,9
deutsch30	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	0,00059	0,0032	0,056	17,4
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	0,00058	0,0263	0,481	18,3
türkei19	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	0,00102	0,0081	0,029	3,61
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	0,00065	0,0013	0,036	27,2

stand zu $\mathcal{Y}_{known}^{TPNDA}$ als bei TPNDA o.I. zu erkennen. Für alle vier untersuchten Netzwerke weisen die Pareto-Fronten, die mit TPNDA o.I. erzeugt wurden, stets einen geringeren Wert als die jeweiligen Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA\ m.I}$ für $dist$ auf. Die Lösungen der Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA\ o.I}$ liegen somit näher an $\mathcal{Y}_{known}^{TPNDA}$.
Betrachtet man die Distanzkennzahlen $dist_1$ und $dist_2$, welche den mittleren und den maximalen Abstand einer Lösung aus $\mathcal{Y}_{known}^{TPNDA}$ zur nächsten Lösung in $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ bzw. $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ anzeigen, so ist kein klarer Vorteil für eine der beiden TPNDA-Varianten erkennbar.
Wie die Kennzahl $dist_1$ zeigt, repräsentiert die Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ die aggregierte Pareto-Front $\mathcal{Y}_{known}^{TPNDA}$ für das Netzwerk deutsch15 im Durchschnitt besser. Der maximale Abstand ($dist_2$) einer Lösung aus $\mathcal{Y}_{known}^{TPNDA}$ zu einer Lösung in der TPNDA o.I. Pareto-Front ist jedoch höher als beim TPNDA m.I. Anhand von $dist_3$ kann auf eine gleichmäßigere Verteilung von $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ geschlossen werden.
Für das Netzwerk deutsch20 wird anhand von $dist_1$ und $dist_2$ deutlich, dass die Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ sämtliche Lösungen in $\mathcal{Y}_{known}^{TPNDA}$ besser repräsentiert als die Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ o.I.}$.
Ähnliche Resultate sind für das Netzwerk deutsch30 erkennbar. Auch hier sind alle Lösungen in $\mathcal{Y}_{known}^{TPNDA}$ durch die Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ besser repräsentiert. Für das Netzwerk türkei19 ist keine klare Aussage über die Vorteile eines der beiden Verfahren zu treffen. Während der mittlere Abstand ($dist_1$) der Lösungen aus $\mathcal{Y}_{known}^{TPNDA}$ zu den Lösungen der betrachteten Pareto-Front für $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ am geringsten ist, weist die Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ für den maximalen Abstand einer Lösung aus $\mathcal{Y}_{known}^{TPNDA}$ zur nächsten Lösung in der betrachteten Pareto-Front den geringeren Wert auf.



(a) deutsch15

(b) deutsch20

(c) deutsch30

(d) türkei19

Abbildung 7.13: Gegenüberstellung der Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ zur aggregierten Pareto-Front $\mathcal{Y}_{known}^{TPNDA}$ für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19

Eine zusätzliche Möglichkeit für den Vergleich beider Varianten von TPNDA bietet die grafische Gegenüberstellung der generierten Pareto-Fronten, welche mit Abbildung 7.13 erfolgt[82]. Die Abbildung zeigt für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19 jeweils einen Ausschnitt der Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA\ m.I.}$, $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und $\mathcal{Y}_{known}^{TPNDA}$.
Wie in Abbildung 7.13(a) für das Netzwerk deutsch15 deutlich wird, dominieren die Lösungen der Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ mit $R_{All}(G_N) > 0,99$ die Lösungen in $\mathcal{Y}_{known}^{TPNDA\ m.I.}$. Gleichzeitig ist die Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ der Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ für Lösungen mit $R_{All}(G_N) < 0,98$ überlegen.

[82] Zur besseren Veranschaulichung der Ergebnisse wurden die Pareto-Fronten nur ausschnittsweise dargestellt.

Für die Netzwerke deutsch20 und deutsch30 ist in Abbildung 7.13 eine ähnliche Konstellation zu erkennen. Lösungen mit $R_{All} > 0,99$ (für deutsch20) bzw. $R_{All} > 0,98$ (für deutsch30) in $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ werden komplett durch Lösungen in der Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ dominiert.
Wie Abbildung 7.13(d) zeigt, liegen die Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ sehr nah beieinander, so dass anhand der Darstellung kein Vorteil für eine der hier untersuchten TPNDA-Varianten erkennbar ist.
Summa summarum lässt sich für den analytischen und grafischen Vergleich der beiden Varianten von TPNDA festhalten, dass kein genereller Vorteil für eine der beiden hier untersuchten Varianten von TPNDA zu erkennen ist. Wie die grafische Darstellung zeigt, werden allerdings mit TPNDA m.I. bessere Lösungen im Bereich hoher Zuverlässigkeiten ($R_{All} > 0,99$) erzeugt. Aus praktischen Gesichtspunkten wäre somit der TPNDA-Variante mit Initialisierung der Vorzug zu geben.

Gegenüberstellung der Verfahren TPNDA m.I., TPNDA o.I. und NSGA-II

Während in diesem Abschnitt bisher die beiden Varianten von TPNDA in einem isolierten Vergleich lediglich einander gegenübergestellt wurden, erfolgt jetzt ein Vergleich der mit TPNDA generierten Pareto-Fronten mit den in Abschnitt 7.3 vorgestellten NSGA-II-Ergebnissen. Für den Vergleich wurde für die Netzwerke deutsch15, deutsch20 und deutsch30 die Pareto-Front $\mathcal{Y}_{true}$ mittels $\mathcal{Y}_{true} = \mathcal{Y}_{known}^{NSGA-II} \cup_{nondom} \mathcal{Y}_{known}^{TPNDA\ o.I.} \cup_{nondom} \mathcal{Y}_{known}^{TPNDA\ m.I.}$ gebildet. Für das Netzwerk türkei19 setzt sich $\mathcal{Y}_{true}$ aus der in Abschnitt 7.3 verwendeten Pareto-Front $\mathcal{Y}_{true}$ und den Lösungen in $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ zusammen[83].
Für die Auswertung werden die Pareto-Fronten $\mathcal{Y}_{known}$ der Verfahren TPNDA m.I., TPNDA o.I. und NSGA-II mit der „wahren“ Pareto-Front $\mathcal{Y}_{true}$ verglichen. Einen analytischen Vergleich zeigen die Tabellen 7.13 (auf S. 184) und 7.14 (auf S. 185). Mit Tabelle 7.13 wird das untersuchte Netzwerk, die betrachtete Pareto-Front, die Anzahl der Lösungen ($ONVG$) in dieser Pareto-Front, die Anzahl der Lösungen ($OTNVG$) der Pareto-Front, welche in $\mathcal{Y}_{known}$ enthalten sind, das Verhältnis ($ONVGR$) der Mächtigkeit der betrachteten Pareto-Front im Vergleich zur Mächtigkeit von $\mathcal{Y}_{true}$ sowie die Fehlerrate ($Error$) gezeigt. Anhand der Kennzahl $ONVG$ ist ersichtlich, dass die Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ für alle vier Netzwerke die höchste Anzahl an Lösungen enthält. Für die untersuchten Netzwerke zeigt die Kennzahl $OTNVG$ an, dass eine Vielzahl der Lösungen aus $\mathcal{Y}_{known}^{NSGA-II}$ und $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ in $\mathcal{Y}_{true}$ enthalten sind, während hingegen aus $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ nur wenige Lösungen in $\mathcal{Y}_{true}$ zu finden sind.
Die Kennzahl $ONVGR$ zeigt, dass für jedes Netzwerk das Verhältnis zwischen der Anzahl der Lösungen in der jeweils betrachteten Pareto-Front zur Anzahl der Lösungen in $\mathcal{Y}_{true}$ für alle drei Verfahren annähernd gleich groß ist.
Mit Tabelle 7.14 wird der Vergleich der Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA\ o.I.}$, $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ und $\mathcal{Y}_{known}^{NSGA-II}$ zur „wahren“ Pareto-Front $\mathcal{Y}_{true}$ mit Hilfe der qualitativen Kennzahlen aus Abschnitt 7.2.2 präsentiert. Die Tabelle zeigt das untersuchte Netzwerk, die betrachtete Pareto-Front, die Distanz $dist$, welche den Abstand zwischen den Lösungen in $\mathcal{Y}_{known}$

[83] Für die Bildung wurde ebenfalls der $\cup_{nondom}$ Operator verwendet.

Tabelle 7.13: Vergleich der Pareto-Fronten $\mathcal{Y}_{known}^{NSGA-II}$, $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ zur Pareto-Front $\mathcal{Y}_{true}$ für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19 anhand der quantitativen Kennzahlen aus Abschnitt 7.2.1

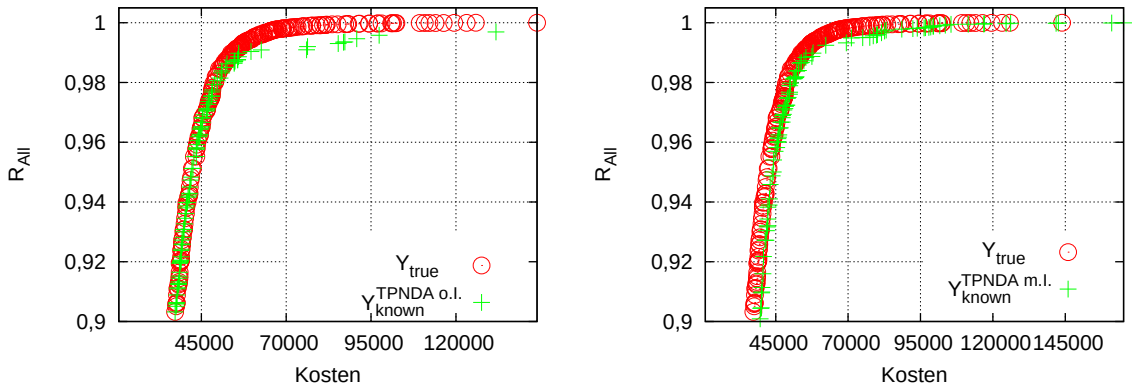
Netzwerk	Pareto-Front	$ONVG$	$OTNVG$	$ONVGR$	$Error$
deutsch15	$\mathcal{Y}_{known}^{NSGA-II}$	217	152	0,65	0,3
	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	150	0	0,449	1
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	207	182	0,62	0,121
deutsch20	$\mathcal{Y}_{known}^{NSGA-II}$	162	144	0,643	0,111
	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	135	8	0,536	0,941
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	123	100	0,488	0,187
deutsch30	$\mathcal{Y}_{known}^{NSGA-II}$	178	113	1,53	0,365
	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	178	14	1,8	0,921
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	178	51	1,96	0,713
türkei19	$\mathcal{Y}_{known}^{NSGA-II}$	134	18	0,905	0,866
	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	114	8	0,77	0,93
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	121	41	0,818	0,661

zur nächsten Lösung in $\mathcal{Y}_{true}$ betrachtet sowie die Distanzkennzahlen $dist_1$, $dist_2$ und $dist_3$, welche den Abstand zwischen sämtlichen Lösungen in $\mathcal{Y}_{true}$ zur jeweils nächsten Lösung in $\mathcal{Y}_{known}$ bewerten.
Für das Netzwerk deutsch15 zeigt $dist$, dass die Lösungen in $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ den geringsten Abstand zu Lösungen in $\mathcal{Y}_{true}$ besitzen. Die Werte für $dist_1$ und $dist_2$ zeigen, dass auch die Lösungen in $\mathcal{Y}_{true}$ den geringsten durchschnittlichen und maximalen Abstand zu den Lösungen in $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ haben.
Die Ergebnisse für das Netzwerk deutsch20 zeigen, dass die Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ einen ähnlich kleinen Abstand wie $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ zu $\mathcal{Y}_{true}$ besitzt. Die Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ besitzt hier, wie auch bei dem Netzwerk deutsch15, einen deutlich größeren Abstand (gemessen an $dist$). Anhand von $dist_1$ wird deutlich, dass die Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ sämtliche Lösungen in $\mathcal{Y}_{true}$ besser als $\mathcal{Y}_{known}^{NSGA-II}$ repräsentiert. Gleichzeitig besitzen beide Pareto-Fronten im Vergleich zu $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ einen sehr hohen Wert für $dist_2$. Ursache hierfür sind Lösungen in $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ mit $R_{All} < 0,6$, welche in $\mathcal{Y}_{true}$ enthalten sind und für die in $\mathcal{Y}_{known}^{NSGA-II}$ und $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ keine Lösungen existieren.
Für das Netzwerk deutsch30 zeigt $dist$, dass sämtliche Lösungen in $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ sehr nah an $\mathcal{Y}_{true}$ liegen. Der Wert für $dist_2$ lässt jedoch zu erkennen, dass $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ nur einen Teilbereich von $\mathcal{Y}_{true}$ abdeckt und die Lösungen in $\mathcal{Y}_{true}$ den durchschnittlich höchsten Abstand zu einer Lösung in $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ besitzen. Die beste Annäherung an $\mathcal{Y}_{true}$ erreicht die Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$. Für diese Pareto-Front sind die Distanzkennzahlen $dist_1$ und $dist_2$ am geringsten. Dies bedeutet, dass $\mathcal{Y}_{known}^{NSGA-II}$ für jede Lösung in $\mathcal{Y}_{true}$ eine annähernd gleiche Lösung enthält.
Ähnliche Ergebnisse sind für das Netzwerk türkei19 festzuhalten. Die Pareto-Front für NSGA-II weist die geringsten Werte für $dist$ und $dist_1$ auf. Gleichzeitig ist aber ein

Tabelle 7.14: Vergleich der Pareto-Fronten $\mathcal{Y}_{known}^{NSGA-II}$, $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ zur Pareto-Front $\mathcal{Y}_{true}$ für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19 anhand der qualitativen Kennzahlen aus Abschnitt 7.2.2

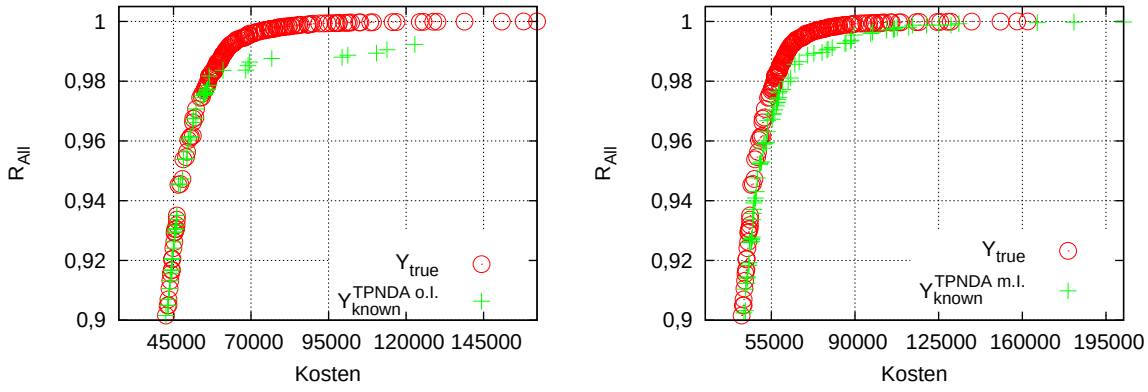
Netzwerk	Pareto-Front	$dist$	$dist_1$	$dist_2$	$dist_3$
deutsch15	$\mathcal{Y}_{known}^{NSGA-II}$	0,0007	0,0308	0,411	13,3
	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	0,0018	0,0164	0,153	9,4
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	0,0003	0,009	0,135	15,0
deutsch20	$\mathcal{Y}_{known}^{NSGA-II}$	0,0003	0,106	0,872	8,2
	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	0,0020	0,0099	0,0882	8,9
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	0,0003	0,0162	0,292	18,0
deutsch30	$\mathcal{Y}_{known}^{NSGA-II}$	0,0166	0,0003	0,013	46,0
	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	0,0009	0,0357	0,513	14,4
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	0,0017	0,0087	0,0909	10,5
türkei19	$\mathcal{Y}_{known}^{NSGA-II}$	0,0005	0,0037	0,208	55,8
	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	0,0015	0,0121	0,0908	7,5
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	0,0011	0,0057	0,0688	12,0

sehr hoher Wert für $dist_2$ zu erkennen, welcher wiederum durch Lösungen in $\mathcal{Y}_{true}$ mit $R_{All} < 0,8$ (welche aus $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ kommen) verursacht wird, für welche es keine entsprechenden Lösungen in $\mathcal{Y}_{known}^{NSGA-II}$ gibt.
Für eine zusätzliche grafische Analyse der Ergebnisse werden mit den Abbildungen 7.14, 7.15, 7.16 und 7.17 die Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ der „wahren“ Pareto-Front gegenübergestellt. Auf die Abbildung von $\mathcal{Y}_{known}^{NSGA-II}$ wurde aus Gründen einer besseren Veranschaulichung der Ergebnisse verzichtet.
Für das Netzwerk deutsch15 zeigt Abbildung 7.14(a) die Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und Abbildung 7.14(b) die Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ sowie $\mathcal{Y}_{true}$. Wie in Abbildung 7.14(a) ersichtlich, liegen die Lösungen aus $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ für $R_{All} < 0,98$ sehr nah bei $\mathcal{Y}_{true}$. Für $R_{All} > 0,98$ ist hingegen eine größere Distanz erkennbar. Für die Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ in Abbildung 7.14(b), ist für $R_{All} > 0,98$ eine bessere Annäherung an $\mathcal{Y}_{true}$ erkennbar. Es wird aber auch deutlich, dass nicht alle Lösungen in $\mathcal{Y}_{true}$ in diesem Bereich durch Lösungen in $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ repräsentiert werden. Dies zeigt, dass insbesondere in dem Bereich $R_{All} > 0,98$ eine Vielzahl von Lösungen in $\mathcal{Y}_{true}$ Lösungen in $\mathcal{Y}_{known}^{NSGA-II}$ entsprechen.
Für das Netzwerk deutsch20, für das die Abbildung 7.15 die Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ zeigt, sind ähnliche Ergebnisse wie für das Netzwerk deutsch15 erkennbar. Für $R_{All} > 0,98$ sind nur noch wenige Lösungen aus $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ in $\mathcal{Y}_{true}$ enthalten. Im Vergleich zu Abbildung 7.14(b) wird für Abbildung 7.15(b) deutlich, dass der Abstand zwischen $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ und $\mathcal{Y}_{true}$ zunimmt. Erst für $C(G_N) > 125.000$ sind aus $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ Lösungen in $\mathcal{Y}_{true}$ enthalten.
Die Zunahme der Distanz der Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ zu $\mathcal{Y}_{true}$ mit zunehmender Problemgröße wird anhand von Abbildung 7.16(a), welche die Pareto-Fronten für das



(a) TPNDA o.I. (b) TPNDA m.I.

Abbildung 7.14: Gegenüberstellung der Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ zur Pareto-Front $\mathcal{Y}_{true}$ für das Netzwerk deutsch15

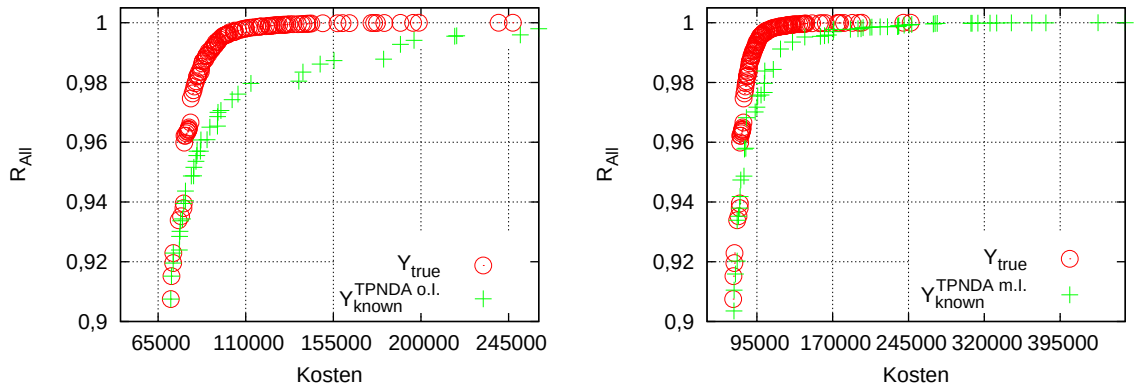


(a) TPNDA o.I. (b) TPNDA m.I.

Abbildung 7.15: Gegenüberstellung der Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ zur Pareto-Front $\mathcal{Y}_{true}$ für das Netzwerk deutsch20

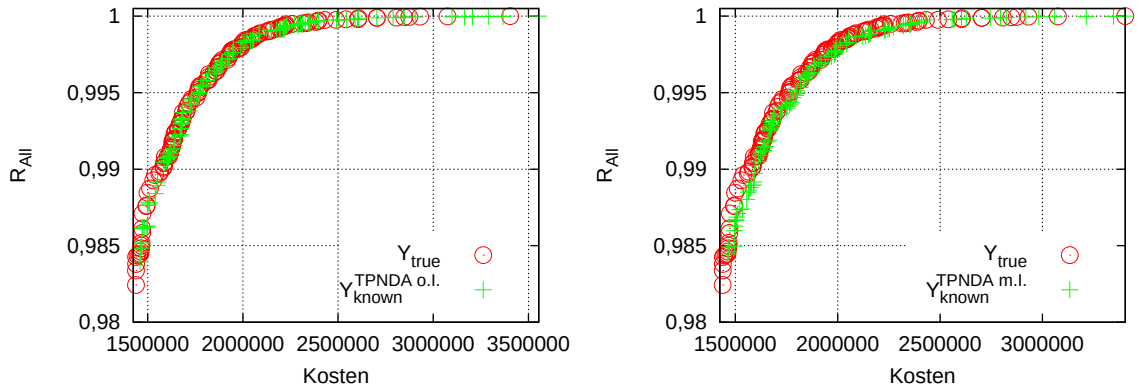
Netzwerk deutsch30 zeigt, bestätigt. Wie in Abbildung 7.16(b) ersichtlich ist, nimmt der Abstand zwischen der Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ und $\mathcal{Y}_{true}$ für größere R_{All} (hier $R_{All} > 0,96$) zu. Erst für $C(G_N) > 170.000$ und $R_{All} \approx 1$ nimmt der Abstand von $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ wieder ab. Daraus lässt sich die Schlussfolgerung ableiten, dass insbesondere in dem Bereich $95.000 \leq C(G_N) \leq 170.000$ die Lösungen in $\mathcal{Y}_{true}$ aus $\mathcal{Y}_{known}^{NSGA-II}$ entnommen wurden.

Für das Netzwerk türkei19, welches sich gegenüber den Netzwerken deutsch15, deutsch20 und deutsch30 hinsichtlich der Kosten- und Zuverlässigkeiten der Technologieoptionen unterscheidet, zeigen die in Abbildung 7.17 dargestellten Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ o.I.}$, dass beide Pareto-Fronten nur einen sehr geringen Abstand zu Lö-



(a) TPNDA o.I. (b) TPNDA m.I.

Abbildung 7.16: Darstellung der Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ zur Pareto-Front $\mathcal{Y}_{true}$ für das Netzwerk deutsch30



(a) TPNDA o.I. (b) TPNDA m.I.

Abbildung 7.17: Darstellung der Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ zur Pareto-Front $\mathcal{Y}_{true}$ für das Netzwerk türkei19

sungen in $\mathcal{Y}_{true}$ besitzen. Eine größere Distanz zwischen den Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ zu $\mathcal{Y}_{true}$ für hohe All-Terminal-Zuverlässigkeiten wie sie bei den anderen hier untersuchten Netzwerken deutlich wird, ist für das Netzwerk türkei19 nicht festzustellen.

Zusammenfassend verdeutlichen die Ergebnisse, dass mit zunehmender Problemgröße (vgl. Ergebnisse für die Netzwerke deutsch15, deutsch20 und deutsch30) der Abstand einer durch TPNDA erzeugten Pareto-Front zu einer „wahren“ Pareto-Front für hohe All-Terminal-Zuverlässigkeiten zunimmt. Der Abstand zwischen $\mathcal{Y}_{true}$ und $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ ist dabei größer als der Abstand zu $\mathcal{Y}_{known}^{TPNDA\ m.I.}$. Für Netzwerke mit sehr hohen Zuverlässigkeiten der Technologieoptionen, wie sie für das Netzwerk türkei19 untersucht wurden,

Tabelle 7.15: Durchschnittliche Anzahl der lokalen Suchschritte für TPNDA o.I. und TPNDA m.I. für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19

Netzwerk	*#Steps* TPNDA m.I.	*#Steps* TPNDA o.I.
deutsch15	23,9	43,9
deutsch20	27,4	50,6
deutsch30	31,6	67,7
türkei19	23,1	19,6

ist auf Basis einer grafischen Auswertung kein Unterschied zwischen den NSGA-II- und den TPNDA-Lösungen erkennbar. Eine Analyse mit den Kennzahlen für den Vergleich einer Pareto-Front $\mathcal{Y}_{known}$ zu einer „wahren“ Pareto-Front $\mathcal{Y}_{true}$ hat gezeigt, dass mit TPNDA in den untersuchten Varianten für die Netzwerke deutsch15 und deutsch20 eine pareto-optimale Lösungsmenge generiert wurde, welche eine bessere Annäherung an die „wahre“ Pareto-Front $\mathcal{Y}_{true}$ bietet, als die mit NSGA-II generierte pareto-optimale Lösungsmenge. Für das Netzwerk deutsch30 ist TPNDA aufgrund der zunehmenden Distanz zwischen $\mathcal{Y}_{known}^{TPNDA}$ und $\mathcal{Y}_{true}$ NSGA-II unterlegen, da hier eine schlechtere pareto-optimale Lösungsmenge gefunden wurde. Für das Netzwerk türkei19 liegen die Ergebnisse aus allen drei untersuchten Verfahren sehr nah beieinander, so dass keine eindeutige Aussage über den Vorzug eines der untersuchten Verfahren möglich ist. Herauszustellen ist der Vorteil beider TPNDA-Varianten, einen größeren Bereich des Lösungsraums mit $\mathcal{Y}_{known}^{TPNDA}$ abzudecken. Sowohl TPNDA m.I. als auch TPNDA o.I. enthalten in ihren pareto-optimalen Lösungsmengen jeweils Lösungen mit $R_{All} < 0,98$, welche in $\mathcal{Y}_{known}^{NSGA-II}$ nicht enthalten sind. Dies ist dahingehend als vorteilhaft zu bewerten, da somit dem Entscheider eine breitere Auswahl an Lösungen zur Verfügung steht.

Analyse des durch TPNDA verursachten Berechnungsaufwandes

Zum Abschluss dieses Abschnittes findet eine Betrachtung zu dem durch TPNDA m.I. und TPNDA o.I. verursachten Berechnungsaufwand statt. Da durch TPNDA jede Lösung der Population in jeder Generation einer lokalen Suche unterworfen wird, ist davon auszugehen, dass durch TPNDA eine sehr hohe Anzahl an Fitnessbewertungen (Bestimmung der All-Terminal-Zuverlässigkeit und der Kosten) erfolgt.
Mit Tabelle 7.15 wird die mittlere Anzahl (*#Steps*) an Suchschritten pro Individuum während eines TPNDA-Laufs für TPNDA m.I. und TPNDA o.I. für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19 gezeigt. Die Ergebnisse zeigen, dass die Anzahl der durchgeführten lokalen Suchschritte mit der Populationsgröße und dem damit verbundenen Wachstum des Problemraums ebenfalls zunimmt. Anhand des Netzwerkes türkei19 wird deutlich, dass bei annähernd gleicher Problemgröße (türkei19 und deutsch20) für das Netzwerk mit den zuverlässigeren Technologieoptionen weniger lokale Suchschritte durchgeführt werden.

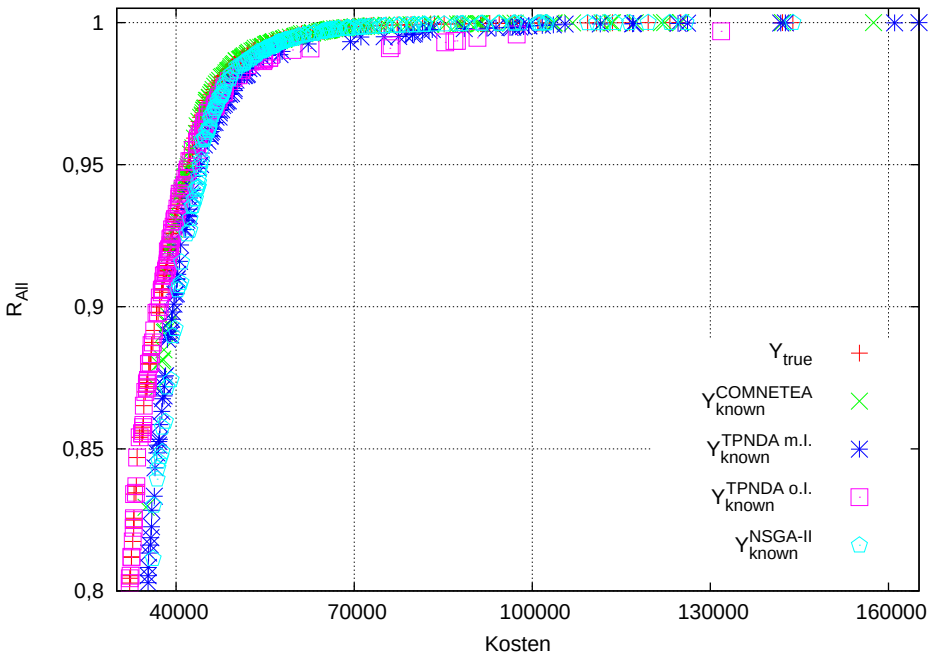
Die Gegenüberstellung der Suchschritte für TPNDA o.I. und TPNDA m.I. zeigt, dass in der Variante mit Initialisierung (mit Ausnahme für türkei19) die lokale Suche früher als in der Variante ohne Initialisierung stoppt. Für das Netzwerk türkei19 führt TPNDA in beiden Varianten annähernd die gleiche Anzahl an lokalen Suchschritten aus.
Die Anzahl der durchgeführten lokalen Suchschritte bestimmt die Anzahl der durch das jeweilige TPNDA Verfahren durchgeführten Fitnessbewertungen. Die durchschnittliche Anzahl der Fitnessbewertungen ($\#Eval$) ergibt sich als $\#Eval = \#Steps \cdot 2 \cdot pop \cdot gen$. Durch NSGA-II wird in jeder Generation die Elternpopulation mit der Kindergeneration vereint und die jeweils besten Lösungen für die nächste Elterngeneration ausgewählt. Aufgrund dieses Vorgehens sind in jeder Generation durch die lokale Suche $2 \cdot pop$ Lösungen zu bewerten. Für das Netzwerk deutsch20 wurden z. B. von TPNDA o.I. $27,4 \cdot 2 \cdot 50 \cdot 100 = 274.000$ Fitnessbewertungen durchgeführt. Für TPNDA m.I. beträgt die Anzahl der Fitnessbewertungen $50,6 \cdot 2 \cdot 50 \cdot 100 = 506.000$.

7.5 Vergleichende Betrachtung der multikriteriellen Planungsansätze COMNETEA, TPNDA und NSGA-II

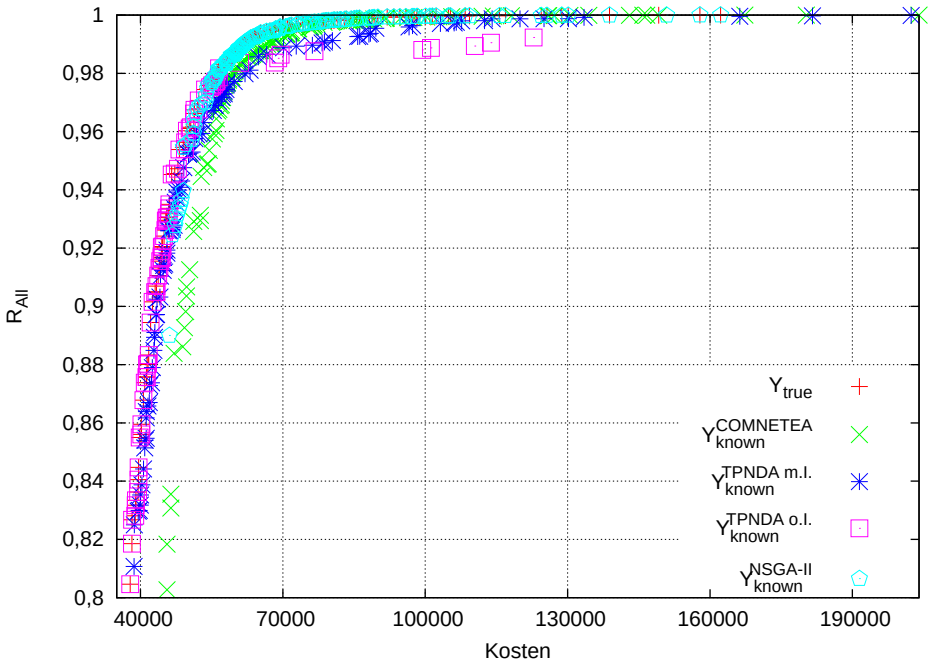
Mit den Abschnitten 7.3, 7.4.1 und 7.4.2 wurden die in den experimentellen Studien erzielten Ergebnisse jeweils isoliert betrachtet. Die Referenzfront $\mathcal{Y}_{true}$ wurde aus den pareto-optimalen Lösungen, welche das jeweilige Verfahren generierte, sowie den Lösungen aus $\mathcal{Y}_{known}^{NSGA-II}$ gebildet[84]. Mit diesem Abschnitt erfolgt eine abschließende Gegenüberstellung der pareto-optimalen Lösungsmengen für COMNETEA, TPNDA o.I., TPNDA m.I. und NSGA-II gegen eine „wahre“ Pareto-Front $\mathcal{Y}_{true}$, welche die Menge sämtlicher für ein Problem bekannten pareto-optimalen Lösungen enthält. Für das hier vorgeschlagene Verfahren COMNETEA wurde für die Auswertung das beste Ergebnis aus Abschnitt 7.4.1.2, welches mit $pop = 100$ und $gen = 300$ erzielt wurde, ausgewählt. De Pareto-Fronten $\mathcal{Y}_{known}^{COMNETEA}$, $\mathcal{Y}_{known}^{TPNDA\ o.I.}$, $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ und $\mathcal{Y}_{known}^{NSGA-II}$ werden mit Abbildung 7.18 und Abbildung 7.19 der „wahren“ Pareto-Front $\mathcal{Y}_{true}$ grafisch gegenübergestellt. Zur besseren Veranschaulichung der Ergebnisse werden mit Abbildung 7.20 für die Netzwerke deutsch15, deutsch20 und deutsch30 die Pareto-Fronten ausschnittsweise für hohe All-Terminal-Zuverlässigkeiten ($R_{All} > 0,98$ für die Netzwerke deutsch15 und deutsch20 bzw. $R_{All} > 0,99$ für deutsch30) gezeigt.
Für das Netzwerk deutsch15 ist in Abbildung 7.20(a) erkennbar, dass die Lösungen in $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ für $R_{All} > 0,99$ durch Lösungen in $\mathcal{Y}_{known}^{NSGA-II}$ und $\mathcal{Y}_{known}^{COMNETEA}$ dominiert werden. Mittels Abbildung 7.18(a) wird deutlich, dass die Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ für $R_{All} < 0,95$ eine Vielzahl der Lösungen aus $\mathcal{Y}_{true}$ enthält und diese die Lösungen der anderen Pareto-Fronten dominieren.
Mit den Abbildungen 7.18(b) und 7.20(b) werden die Pareto-Fronten für das Netzwerk deutsch20 gezeigt. In Abbildung 7.18(b) ist zu erkennen, dass für $R_{All} < 0,92$ eine Vielzahl der Lösungen aus $\mathcal{Y}_{true}$ in $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ enthalten sind, während die Pareto-Fronten $\mathcal{Y}_{known}^{COMNETEA}$ und $\mathcal{Y}_{known}^{NSGA-II}$ in diesem Bereich des Lösungsraums keine (NSGA-II) bzw. durch die Lösungen der anderen Verfahren dominierte Lösungen

[84] Für das Netzwerk türkei19 wurden zusätzlich die Ergebnisse aus [63, 157] zu $\mathcal{Y}_{true}$ hinzugefügt.

(COMNETEA) erzeugen. Wie Abbildung 7.20(b) zeigt, werden auch für das Netzwerk deutsch20 die Lösungen in $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ für $R_{All} > 0,985$ durch Lösungen in $\mathcal{Y}_{known}^{COMNETEA}$ und $\mathcal{Y}_{known}^{NSGA-II}$ dominiert.
Für das Netzwerk deutsch30 ziegen die Abbildungen 7.19(a) und 7.20(c) die Pareto-Fronten. Es ist zu erkennen, dass die Pareto-Fronten $\mathcal{Y}_{known}^{NSGA-II}$ und $\mathcal{Y}_{known}^{COMNETEA}$ nur sehr wenige Lösungen mit $R_{All} < 0,9$ besitzen und dadurch diesen Bereich der Pareto-Front $\mathcal{Y}_{true}$ nur sehr schlecht abdecken. In Abbildung 7.20(c) wird ersichtlich, dass die Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ wiederum für $R_{All} > 0,99$ einen sehr großen Abstand zur Pareto-Front $\mathcal{Y}_{known}$, welche sich hier vorwiegend aus Lösungen der Lösungsmengen $\mathcal{Y}_{known}^{NSGA-II}$ und $\mathcal{Y}_{known}^{COMNETEA}$ zusammensetzt, aufweisen. Wie bereits in Abschnitt 7.4.2.2 herausgestellt wurde, zeigen die Abbildungen 7.20(a), 7.20(b) und 7.20(c), dass der Abstand zwischen der Pareto-Front $\mathcal{Y}_{true}$ und den Pareto-Fronten $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ sowie $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ mit der Problemgröße wächst. Abbildung 7.20(c) macht ebenfalls deutlich, dass die Pareto-Fronten $\mathcal{Y}_{known}^{NSGA-II}$ und $\mathcal{Y}_{known}^{COMNETEA}$ beide einen sehr geringen Abstand zu $\mathcal{Y}_{true}$ besitzen.
Für das Netzwerk türkei19 wurde bereits in der Auswertung für die experimentelle Untersuchung von COMNETEA (in Abschnitt 7.4.1.2) sowie in der Analyse der Ergebnisse in Abschnitt 7.4.2.2 deutlich, dass sämtliche hier generierten Pareto-Fronten zueinander einen sehr geringen Abstand besitzen. Mit Abbildung 7.19(b) wird dies ebenfalls bestätigt. Für $R_{All} > 0,98$ weisen alle abgebildeten Pareto-Fronten einen sehr geringen Abstand zueinander und zu $\mathcal{Y}_{true}$ auf. Eine Präferenz für eine der Pareto-Fronten ist anhand der Abbildung nicht feststellbar. Für eine detaillierte Analyse ist die Bewertung der Pareto-Fronten $\mathcal{Y}_{known}^{COMNETEA}$, $\mathcal{Y}_{known}^{TPNDA\ o.I.}$, $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ und $\mathcal{Y}_{known}^{NSGA-II}$ im Vergleich zu $\mathcal{Y}_{true}$ notwendig.
Mit Tabelle 7.16 und Tabelle 7.17 wird eine analytische Bewertung der Lösungsmengen der Verfahren NSGA-II, COMNETEA und TPNDA o.I. und TPNDA m.I. vorgenommen.
Tabelle 7.16 auf S. 194 zeigt das untersuchte Netzwerk, die betrachtete Pareto-Front, die in der Pareto-Front enthaltenen Lösungen ($ONVG$), die Anzahl ($OTNVG$) der Lösungen, welche ebenfalls in $\mathcal{Y}_{true}$ enthalten sind, das Verhältnis zwischen $ONVG$ und $|\mathcal{Y}_{true}|$ sowie die Fehlerrate $Error$.
Die Kennzahl $ONVG$ zeigt, dass für alle vier untersuchten Netzwerke mit NSGA-II und COMNETEA die Menge an pareto-optimalen Lösungen mit der höchsten Kardinalität erzeugt wurde.
Anhand von $OTNVG$ wird deutlich, dass einige Lösungsmengen nur sehr wenige oder keine Lösungen aus $\mathcal{Y}_{true}$ enthalten. Insbesondere aus $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ sind für alle vier der hier untersuchten Netzwerke nur sehr wenige Lösungen in $\mathcal{Y}_{true}$ enthalten.
Das Verhältnis $ONVGR$ zwischen der Anzahl der Lösungen in der jeweiligen Lösungsmenge $\mathcal{Y}_{known}$ zu $\mathcal{Y}_{true}$ liegt für alle Netzwerke zwischen $0,455 \leq ONVGR \leq 0,906$, woraus sich ableiten lässt, dass sämtliche Pareto-Fronten weniger Lösungen als die jeweilige „wahre“ Pareto-Front besitzen.
Die Fehlerrate $Error$ unterstreicht noch einmal den direkten Zusammenhang zwischen einer pareto-optimalen Lösungsmenge $\mathcal{Y}_{known}$ und einer „wahren“ Pareto-Front $\mathcal{Y}_{true}$. Aufgrund der Definition der Fehlerrate $Error$ (vgl. Abschnitt 7.2.1 auf S. 146) weisen Pareto-Fronten wie z. B. $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ (Netzwerk deutsch15), welche keine Lösung aus

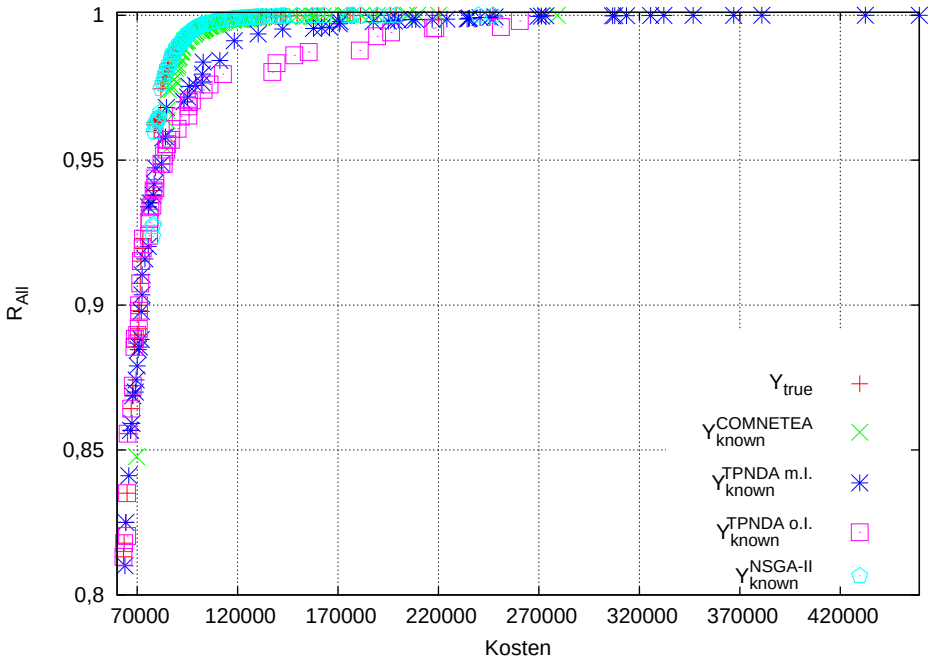


(a) deutsch15

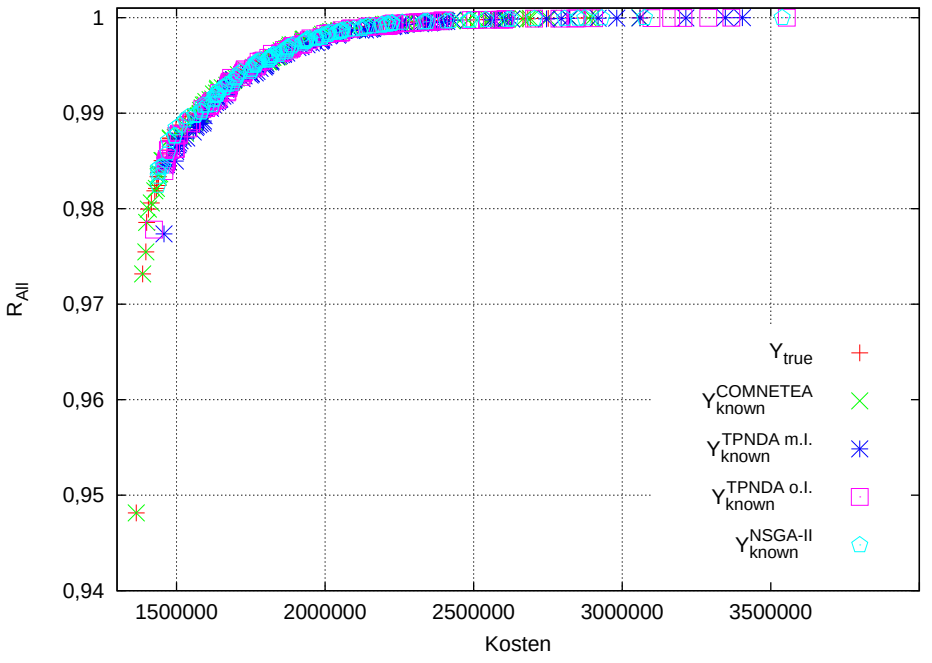


(b) deutsch20

Abbildung 7.18: Gegenüberstellung der Pareto-Fronten $\mathcal{Y}_{true}$, $\mathcal{Y}_{known}^{COMNETEA}$, $\mathcal{Y}_{known}^{TPNDA\ o.I.}$, $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ und $\mathcal{Y}_{known}^{NSGA-II}$ für die Netzwerke deutsch15 und deutsch20

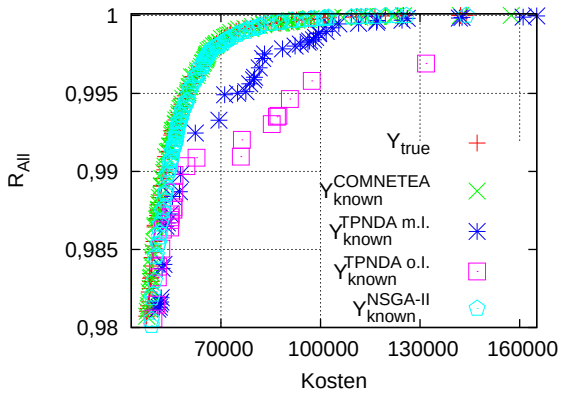


(a) deutsch30

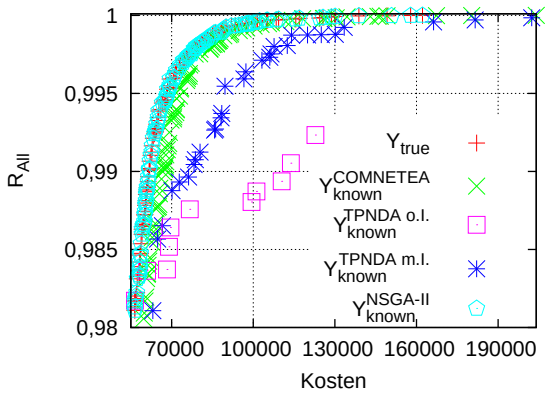


(b) türkei19

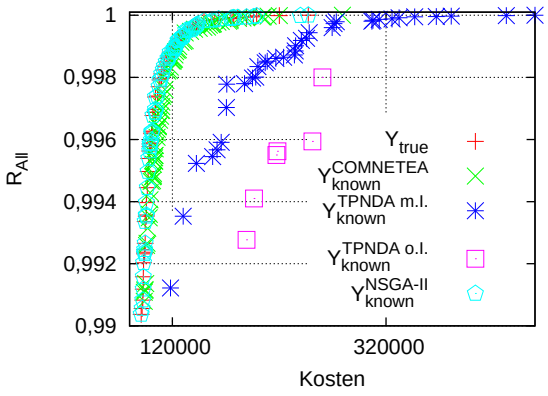
Abbildung 7.19: Gegenüberstellung der Pareto-Fronten $\mathcal{Y}_{true}$, $\mathcal{Y}_{known}^{COMNETEA}$, $\mathcal{Y}_{known}^{TPNDA\ o.I.}$, $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ und $\mathcal{Y}_{known}^{NSGA-II}$ für die Netzwerke deutsch30 und türkei19



(a) deutsch15



(b) deutsch20



(c) deutsch30

Abbildung 7.20: Ausschnittsweise Darstellung der Pareto-Fronten $\mathcal{Y}_{true}$, $\mathcal{Y}_{known}^{NSGA-II}$, $\mathcal{Y}_{known}^{TPNDA\ o.I.}$, $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ und $\mathcal{Y}_{known}^{COMNETEA}$ der Netzwerke deutsch15, deutsch20 und deutsch30 für hohe All-Terminal-Zuverlässigkeiten

$\mathcal{Y}_{true}$ enthalten, eine Fehlerrate von 1 auf. Für Pareto-Fronten wie z. B. $\mathcal{Y}_{known}^{COMNETEA}$ (Netzwerk deutsch15), welche eine große Anzahl an Lösungen in $\mathcal{Y}_{true}$ enthalten, ist *Error* hingegen sehr klein.

Qualitative Aussagen über den Abstand der unterschiedlichen Lösungsmengen der hier untersuchten Verfahren zu einer „wahren“ Pareto-Front sind mit den in Tabelle 7.17 gezeigten Kennzahlen möglich.

Mit Tabelle 7.17 wird das untersuchte Netzwerk, die betrachtete Pareto-Front $\mathcal{Y}_{known}$, der Abstand *dist* der Lösungen aus $\mathcal{Y}_{known}$ zur nächsten Lösung in $\mathcal{Y}_{true}$ sowie die Abstände $dist_1$, $dist_2$ und $dist_3$ der Lösungen in $\mathcal{Y}_{true}$ zur nächsten Lösung in $\mathcal{Y}_{known}$ gezeigt. Weiterhin stellt die Tabelle für den zur Erzeugung der Pareto-Front notwendigen Berechnungsaufwand auf Basis der mittleren Anzahl der Fitnessbewertungen ($\#Eval$) dar.

Tabelle 7.16: Gegenüberstellung der Pareto-Fronten $\mathcal{Y}_{known}^{NSGA-II}$, $\mathcal{Y}_{known}^{TPNDA\ m.I.}$,$\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und $\mathcal{Y}_{known}^{COMNETEA}$ zur Pareto-Front $\mathcal{Y}_{true}$ für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19 anhand der quantitativen Kennzahlen aus Abschnitt 7.2.1

Netzwerk	Pareto-Front	*ONVG*	*OTNVG*	*ONVGR*	*Error*
deutsch15	$\mathcal{Y}_{known}^{NSGA-II}$	217	29	0,660	0,866
	$\mathcal{Y}_{known}^{COMNETEA}$	202	152	0,614	0,248
	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	150	0	0,456	1
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	207	148	0,629	0,285
deutsch20	$\mathcal{Y}_{known}^{NSGA-II}$	162	143	0,643	0,117
	$\mathcal{Y}_{known}^{COMNETEA}$	134	1	0,532	0,993
	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	135	8	0,536	0,941
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	123	100	0,488	0,187
deutsch30	$\mathcal{Y}_{known}^{NSGA-II}$	116	112	0,648	0,035
	$\mathcal{Y}_{known}^{COMNETEA}$	103	3	0,575	0,971
	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	99	13	0,553	0,869
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	91	51	0,508	0,44
türkei19	$\mathcal{Y}_{known}^{NSGA-II}$	134	14	0,899	0,896
	$\mathcal{Y}_{known}^{COMNETEA}$	135	65	0,906	0,519
	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	114	1	0,765	0,991
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	121	24	0,812	0,802

Für das Netzwerk deutsch15 weisen die Kennzahlen in Tabelle 7.17 die Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ als die Lösungsmenge mit dem geringsten Abstand zu $\mathcal{Y}_{true}$ aus. Anhand der Distanzkennzahl *dist* ist zu erkennen, dass die Lösungen der Pareto-Fronten $\mathcal{Y}_{known}^{NSGA-II}$, $\mathcal{Y}_{known}^{COMNETEA}$ und $\mathcal{Y}_{known}^{TNPDAo.I.}$ einen sehr geringen Abstand zur jeweils nächsten Lösung in $\mathcal{Y}_{true}$ besitzen. Mit Hilfe der Kennzahl $dist_1$ wird deutlich, dass die Lösungen in $\mathcal{Y}_{known}^{NSGA-II}$ und $\mathcal{Y}_{known}^{COMNETEA}$ eine Teilmenge der Lösungen aus $\mathcal{Y}_{true}$ gut repräsentieren, während $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ für alle Lösungen in $\mathcal{Y}_{true}$ im Mittel eine sehr nahe Lösung enthält. Ursächlich für die gute Bewertungen von $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ ist, dass diese Lösungen mit $R_{All} < 0,95$ enthält, welche in $\mathcal{Y}_{true}$ einfließen. Wie Abbildung 7.18(a) auf S. 191 zeigt, dominieren die Lösungen aus $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ sämtliche Lösungen der anderen Pareto-Fronten für $R_{All} \leq 0,95$. Wie in Abbildung 7.18(a) ebenfalls sichtbar wird, sind die Lösungen in $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ über den kompletten Lösungsraum verteilt und können damit eine Vielzahl der Lösungen in $\mathcal{Y}_{true}$ sehr gut repräsentieren.
Für das Netzwerk deutsch20 wird die Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ durch *dist* als die Pareto-Front identifiziert, deren Lösungen den geringsten Abstand zu einer Lösung in $\mathcal{Y}_{true}$ besitzen. Betrachtet man zusätzlich das Abstandsmaß $dist_1$, so ist zu erkennen, dass die Lösungen wiederum nur einen Teilbereich von $\mathcal{Y}_{true}$ sehr gut abdecken. Der hohe Wert $dist_1$ bestätigt das bereits in der grafischen Analyse deutlich gewordene Defizit

Tabelle 7.17: Gegenüberstellung der Pareto-Fronten $\mathcal{Y}_{known}^{NSGA-II}$, $\mathcal{Y}_{known}^{TPNDA\ m.I.}$, $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und $\mathcal{Y}_{known}^{COMNETEA}$ zur Pareto-Front $\mathcal{Y}_{true}$ für die Netzwerke deutsch15, deutsch20, deutsch30 und türkei19 anhand der qualitativen Kennzahlen aus Abschnitt 7.2.2

Netzwerk	Pareto-Front	$dist$	$dist_1$	$dist_2$	$dist_3$	$\#Eval$
deutsch15	$\mathcal{Y}_{known}^{NSGA-II}$	0,0007	0,0328	0,411	12,5	30.914
	$\mathcal{Y}_{known}^{COMNETEA}$	0,0006	0,0137	0,172	12,5	128.349
	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	0,0018	0,0176	0,153	8,7	239.000
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	0,0003	0,0095	0,131	13,8	439.000
deutsch20	$\mathcal{Y}_{known}^{NSGA-II}$	0,0003	0,106	0,872	8,2	30.734
	$\mathcal{Y}_{known}^{COMNETEA}$	0,0024	0,0292	0,242	8,3	104.353
	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	0,0020	0,0099	0,090	9,1	274.000
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	0,0004	0,0162	0,292	18,0	506.000
deutsch30	$\mathcal{Y}_{known}^{NSGA-II}$	0,0004	0,101	0,935	9,2	30.832
	$\mathcal{Y}_{known}^{COMNETEA}$	0,0016	0,053	0,669	12,6	82.872
	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	0,0095	0,0103	0,056	5,4	316.000
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	0,0013	0,0174	0,059	3,4	677.000
türkei19	$\mathcal{Y}_{known}^{NSGA-II}$	0,0017	0,0104	0,662	63,9	30.878
	$\mathcal{Y}_{known}^{COMNETEA}$	0,0004	0,0031	0,023	7,6	97.861
	$\mathcal{Y}_{known}^{TPNDA\ m.I.}$	0,0023	0,0151	0,565	37,4	462.000
	$\mathcal{Y}_{known}^{TPNDA\ o.I.}$	0,0025	0,0107	0,573	53,7	392.000

der Pareto-Front, keine Lösungen mit $R_{All} < 0,88$ zu besitzen, so dass sämtliche Lösungen in $\mathcal{Y}_{true}$ mit $R_{All} < 0,88$ einen sehr großen Abstand zur nächsten Lösung in $\mathcal{Y}_{known}^{NSGA-II}$ aufweisen. Die Lösungen aus $\mathcal{Y}_{true}$ besitzen im Mittel einen sehr geringen Abstand zu den nächsten Lösungen in $\mathcal{Y}_{known}^{COMNETEA}$. Wie in Abbildung 7.18(b) auf S. 191 deutlich wird, nimmt der Abstand zwischen $\mathcal{Y}_{true}$ und $\mathcal{Y}_{known}^{COMNETEA}$ für $R_{All} > 0,8$ im Vergleich zu $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ aber zu. Als die Pareto-Front, welche die „wahre“ Pareto-Front $\mathcal{Y}_{true}$ am besten abbildet, wird anhand der Kennzahlen $dist_1$ und $dist_2$ in Tabelle 7.17 die Pareto-Front $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ klassifiziert. Die Werte für $dist_1$ und $dist_2$ zeigen, dass die Pareto-Front für jede Lösung in $\mathcal{Y}_{true}$ eine annähernd gleiche Lösung besitzt.

Für das Netzwerk deutsch30 zeigt der Wert für $dist$, dass die Lösungen in $\mathcal{Y}_{known}^{NSGA-II}$ wiederum den geringsten Abstand zur jeweils nächsten Lösung in $\mathcal{Y}_{true}$ besitzen. Die Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ deckt aber wieder nur einen Teilbereich von $\mathcal{Y}_{true}$ ab und enthält keine Lösungen mit $R_{All} < 0,9$. Daher fallen die Abstandswerte für $dist_1$ und $dist_2$ sehr hoch aus. Die Werte für $dist_1$ und $dist_2$ zeigen, dass von den vier hier betrachteten Verfahren mit TPNDA m.I. die Lösungsmenge $\mathcal{Y}_{known}$ erzeugt wurde, welche sämtliche Lösungen in $\mathcal{Y}_{true}$ am besten repräsentiert.

Während mit der grafischen Analyse für das Netzwerk türkei19 in Abbildung 7.19(b) keine klare Aussage über die Qualität der verschiedenen Pareto-Fronten möglich ist,

sind anhand Tabelle 7.17 Unterschiede im Abstand der einzelnen Pareto-Fronten zur „wahren“ Pareto-Front $\mathcal{Y}_{true}$ erkennbar. Der sehr kleine Wert für *dist* für die Lösungen in $\mathcal{Y}_{known}^{COMNETEA}$ zeigt an, dass diese Lösungen sehr nahe an Lösungen in $\mathcal{Y}_{true}$ liegen oder in $\mathcal{Y}_{true}$ enthalten sind. Die Kennzahlen $dist_1$ und $dist_2$ zeigen ebenfalls, dass der Mittelwert des Abstandes und der maximale Abstand von Lösungen in $\mathcal{Y}_{true}$ zur nächsten Lösung in $\mathcal{Y}_{known}$ für $\mathcal{Y}_{known}^{COMNETEA}$ am geringsten ist. Die Pareto-Fronten $\mathcal{Y}_{known}^{NSGA-II}$, $\mathcal{Y}_{known}^{TPNDA\ o.I.}$ und $\mathcal{Y}_{known}^{TPNDA\ m.I.}$ besitzen annähernd den gleichen Abstand zu $\mathcal{Y}_{true}$, können die guten Distanzwerte für $\mathcal{Y}_{known}^{COMNETEA}$ jedoch nicht erreichen.
Betrachtet man den Berechnungsaufwand (gemessen an $\#Eval$), der zur Generierung der jeweiligen Pareto-Fronten notwendig ist, so zeigt Tabelle 7.17, dass mit NSGA-II der geringste und mit TPNDA o.I. der höchste Berechnungsaufwand verursacht wird. Die experimentell erzielten Ergebnisse zeigen, dass der zusätzliche Berechnungsaufwand, welcher durch die lokale Suche entsteht, für alle vier der hier untersuchten Netzwerke eine bessere Annäherung der Lösungsmenge $\mathcal{Y}_{known}$ an $\mathcal{Y}_{true}$ ermöglicht hat. Somit ist insbesondere unter Berücksichtigung der Tatsache, dass es sich bei dem hier untersuchten Planungsproblem um kein echtzeitkritisches Problem handelt, der zusätzliche Berechnungsaufwand in Anbetracht der Steigerung der Ergebnisqualität vertretbar.
Die grafische Analyse der Ergebnisse hat gezeigt, dass für die Netzwerke deutsch20, deutsch25 und deutsch30 $\mathcal{Y}_{true}$ für $R_{All} > 0,98$ primär Lösungen aus den Pareto-Fronten $\mathcal{Y}_{known}^{NSGA-II}$ und $\mathcal{Y}_{known}^{COMNETEA}$ beinhaltet. Die Lösungen beider Verfahren fokussieren dabei sehr stark auf diesen Lösungsbereich. Im Gegensatz dazu werden durch TPNDA o.I. und TPNDA m.I. mehr Lösungen mit $R_{All} < 0,98$ gefunden. Wie die Abbildungen 7.18(a), 7.18(b) und 7.19(a) zeigen, sind die Lösungen für beide TPNDA-Varianten breiter über den Lösungsraum verteilt.

7.6 Zusammenfassung

Mit diesem Kapitel wurde die Kommunikationsnetzwerkplanung als multikriterielles Planungsproblem mit den Zielen Kosten und Zuverlässigkeit als multikriterielles kombinatorisches Optimierungsproblem untersucht. Hierfür wurden die Ziele Kosten und Zuverlässigkeit während der Optimierung parallel betrachtet. Für die Beurteilung der durch ein multikriterielles Planungsverfahren generierten Lösungsmenge wurden quantitative und qualitative Kennzahlen eingeführt, die eine Gegenüberstellung einer durch ein Verfahren generierten Pareto-Front zu einer „wahren“/optimalen Pareto-Front ermöglichen.
Mit Abschnitt 7.3 wurden die in einer experimentellen Studie erzielten Ergebnisse für NSGA-II, welches für die Kommunikationsnetzwerkplanung als multikriterielles Planungsproblem mit den Zielen Kosten und Zuverlässigkeit angepasst wurde, dargestellt. Für die Netzwerke deutsch15, deutsch20 und deutsch30 dienten die dabei erzielten Ergebnisse als Referenzlösungsmenge, welche den im Rahmen der Arbeit entworfenen Verfahren COMNETEA, TPNDA o.I. und TPNDA m.I. gegenübergestellt wurden. Für das Netzwerk türkei19 konnte die mittels NSGA-II erzielte Lösungsmenge mit mehreren aus der Literatur entnommenen Lösungsmengen $\mathcal{Y}_{true}$ verglichen werden. Es wurde gezeigt, dass das hier verwendete NSGA-II-Verfahren bessere, als die bisher bekannten Lösungen findet.

Mit COMNETEA und TPNDA wurden zwei unterschiedliche Konzepte für die Erweiterung eines einfachen NSGA-II-Ablaufs um eine lokale Suche vorgeschlagen und empirisch untersucht. Mit COMNETEA wird nach Abschluss des NSGA-II für die Lösungen in $\mathcal{Y}_{known}^{NSGA-II}$ eine lokale Suche durchgeführt, welche die Lösung näher an die Pareto-Front $\mathcal{Y}_{true}$ bringen soll. Für COMNETEA wurde gezeigt, dass durch den Einsatz der nachgelagerten lokalen Suche eine Verschiebung der Lösungen aus $\mathcal{Y}_{known}^{NSGA-II}$ in Richtung $\mathcal{Y}_{true}$ möglich ist.
Mit TPNDA wurde die lokale Suche direkt in den NSGA-II integriert. Hierfür wurde der Planungsprozess in die Planung der Topologie, welche durch das NSGA-II-Verfahren erfolgt, und die Wahl der Technologieoptionen für die entworfene Topologie, welche mittels lokaler Suche erfolgt, unterteilt. TPNDA führt hierfür in jeder Generation für jede Lösung der aktuellen Population eine lokale Suche durch, mit der die Technologieoptionen bestimmt werden. Anhand der Ergebnisse einer experimentellen Untersuchung konnte gezeigt werden, dass die durch TPNDA erzeugten Pareto-Fronten einen größeren Bereich des Lösungsraums als die Pareto-Front $\mathcal{Y}_{known}^{NSGA-II}$ abdecken.
Eine Gegenüberstellung der durch die Verfahren NSGA-II, COMNETEA und TPNDA erzeugten Pareto-Fronten gegen die „wahre“ Pareto-Front $\mathcal{Y}_{true}$, welche sämtliche pareto-optimalen Lösungen aus sämtlichen Experimenten enthält, hat gezeigt, dass für die Netzwerke deutsch15, deutsch20 und deutsch30 mittels TPNDA die Pareto-Front erzeugt wurde, welche sämtliche Lösungen in $\mathcal{Y}_{true}$ am besten repräsentiert. Nachteilig an den Ergebnissen für TPNDA ist, dass hier die Lösungen mit $R_{All} > 0,98$ durch Lösungen aus $\mathcal{Y}_{known}^{COMNETEA}$ und $\mathcal{Y}_{known}^{NSGA-II}$ dominiert werden. Dem gegenüber weisen die Pareto-Fronten $\mathcal{Y}_{known}^{COMNETEA}$ und $\mathcal{Y}_{known}^{NSGA-II}$ das Defizit auf, dass diese jeweils nur einen Teilbereich von $\mathcal{Y}_{true}$ sehr gut abbilden. Für das Netzwerk türkei19 konnten die besten Ergebnisse mit COMNETEA erzielt werden. Insbesondere für dieses Netzwerk konnte gezeigt werden, dass die nachgelagerte lokale Suche, wie sie mit COMNETEA realisiert wurde, eine Verschiebung der Ergebnisse in Richtung $\mathcal{Y}_{true}$ ermöglicht.

8 Zusammenfassung

Mit der vorliegenden Arbeit wurde die Planung von Kommunikationsnetzwerken unter Kosten- und Zuverlässigkeitsgesichtspunkten untersucht. Eingangs wurde das Planungsproblem als Teilaufgabe eines ganzheitlichen Informationsmanagements dargestellt und in den Aufgabenbereich des IT-Sicherheitsmanagements eingeordnet. Zielsetzung der Arbeit war es, Verfahren vorzuschlagen, welche integriert in ein Entscheidungsunterstützungswerkzeug, den Planungsprozess von ausfallsicheren und ökonomischen Kommunikationsnetzwerken im Rahmen des IT-Sicherheitsmanagements unterstützen. Dabei wurden sowohl Problemstellungen untersucht, in denen sämtliche Verbindungen des Netzwerkes die gleiche Zuverlässigkeit besitzen als auch Problemstellungen, bei denen für jede Verbindung des Netzwerkes unterschiedliche Technologien zur Verfügung stehen, welche sich hinsichtlich der Kosten und Ausfallsicherheit unterscheiden.
Das betrachtete Problem wurde als kombinatorisches Optimierungsproblem modelliert. Zur Lösung des Problems wurden Metaheuristiken vorgeschlagen. Als Lösungsverfahren wurden aus der Klasse der populationsbasierten Metaheuristiken die genetischen Algorithmen ausgewählt.
Für die Bewertung der Kosten, welche mit der Etablierung einer entworfenen Netzwerktopologie entstehen, wurden die Fixkosten verwendet, welche durch Installation der ausgewählten Verbindungen entstehen. Für die Bewertung der Zuverlässigkeit wurden in Kapitel 4 unterschiedliche Zuverlässigkeitsmaße zur Beurteilung der Ausfallsicherheit von Kommunikationsnetzwerken betrachtet. In dieser Arbeit wurde die All-Terminal-Zuverlässigkeit als Zuverlässigkeitmaß, welches die Kommunikationsfähigkeit zwischen allen Knoten eines Netzwerkes bewertet, verwendet. Zur Beschleunigung der exakten Berechnung der All-Terminal-Zuverlässigkeit wurde ein verteiltes Berechnungsverfahren vorgeschlagen. In einer empirischen Untersuchung wurden dieses Verfahren und eine Reihe von aus der Literatur entnommenen Verfahren zur Bestimmung der All-Terminal-Zuverlässigkeit untersucht und aus den Ergebnissen Vorschläge für den Einsatz der unterschiedlichen Methoden bei unterschiedlichen Problemgrößen abgeleitet.
Bei der Modellierung des hier betrachteten Planungsproblems als kombinatorisches Optimierungsproblem wurde herausgestellt, dass die Planung von zuverlässigen und ökonomischen Kommunikationsnetzwerken als mono- und multikriterielles Planungsproblem erfolgen kann. Im Rahmen dieser Arbeit wurden die Verfahren STC-GA, LaBORNet und BaBORNet für die monokriterielle Planung unter Verwendung der Kosten als Zielfunktion und der Berücksichtigung der All-Terminal-Zuverlässigkeit als Nebenbedingung entworfen. In einer experimentellen Untersuchung wurde gezeigt, dass die hier vorgeschlagenen Verfahren STC-GA, LaBORNet und BaBORNet bisherigen in der Literatur vorgeschlagenen Lösungsansätzen überlegen sind.
Als Planungsverfahren mit dem Ziel der Maximierung der All-Terminal-Zuverlässigkeit unter Berücksichtigung eines maximalen Budgets als Kostenschranke für die zu entwerfende Netzwerktopologie wurden die Verfahren STC2-GA und SAGA vorgeschlagen. In

einer experimentellen Studie wurden beide Verfahren für Netzwerke mit unterschiedlicher Größe und unterschiedlichen Verbindungszuverlässigkeiten untersucht. Die Ergebnisse zeigen, dass sowohl der STC2-GA als auch SAGA in der Lage sind, Lösungen mit einer sehr hohen Zuverlässigkeit innerhalb des vorgegebenen Budgets zu finden.
Mit Kapitel 7 wurde die Planung von ökonomischen und zuverlässigen Kommunikationsnetzwerken als multikriterielles kombinatorisches Optimierungsproblem untersucht. Es wurden unterschiedliche Metriken vorgestellt, welche einen Vergleich der Lösungsmengen verschiedener multikriterieller Lösungsverfahren ermöglichen. Für die simultane Planung von Kommunikationsnetzwerken mit den Zielen Kosten und Zuverlässigkeiten wurden die Verfahren COMNETEA und TPNDA vorgeschlagen. Eine experimentelle Untersuchung stellte beide Verfahren der NSGA-II-Metaheuristik gegenüber. Für COMNETEA und TPNDA wurde gezeigt, dass beide Verfahren bessere Lösungen als das NSGA-II-Verfahren finden. Weiterhin zeigte die Untersuchung, dass COMNETEA und TPNDA bisher veröffentlichten Verfahren in der Lösungsqualität überlegen sind.
Zusammenfassend gilt: Die vorliegende Arbeit schlägt Verfahren für die mono- und multikriterielle Planung von Kommunikationsnetzwerken unter Kosten- und Zuverlässigkeitsaspekten vor, für die gezeigt werden konnte, dass diese in der Qualität der Lösungen bisherigen Arbeiten auf diesem Gebiet überlegen sind. Als Werkzeuge zur Entscheidungsunterstützung können die Verfahren STC-GA, LaBORNet, BaBORNet, STC2-GA, SAGA, COMNETEA und TPNDA im Rahmen des IT-Sicherheitsmanagements den Entscheider beim Finden einer annähernd optimalen Netzwerktopologie unterstützen.

9 Ausblick

Der Einsatz der IT in nahezu allen Bereichen einer Unternehmung hat zu einer hochgradigen Abhängigkeit von den IT-Systemen geführt. Die Verfügbarkeit der IT nimmt eine Schlüsselrolle in der Durchführung von Geschäftsprozessen ein. In Anlehnung an andere Arbeiten auf dem Gebiet der Kommunikationsnetzwerkplanung mit den Zielen Kosten und Zuverlässigkeit wurden für diese Arbeit die auf S. 9 dargestellten Annahmen getroffen. Durch die "Aufweichung" der im Rahmen dieser Arbeit getroffenen Annahmen wäre eine noch praxisnähere Modellierung des Problems möglich. Ansatzpunkte für weitere Forschungsarbeiten ergeben sich z. B. durch Erweiterung des Modells um Kommunikationsanforderungen zwischen den einzelnen Kommunikationsknoten und der Berücksichtigung von Übertragungskapazitäten der Verbindungen. Ein anderer Ansatzpunkt für weitere Forschungsarbeiten liegt in der Verwendung des Performability-Maßes [82] für die simultane Bewertung der Performance und Zuverlässigkeit von Kommunikationsnetzwerken.
Für die Berechnung der exakten All-Terminal-Zuverlässigkeit wurde im Rahmen der vorliegenden Arbeit ein verteilter Dekompositionsansatz vorgeschlagen. Es wurde gezeigt, dass dieser Ansatz für einige der untersuchten Probleme eine sehr schnelle Berechnung ermöglicht. In der experimentellen Untersuchung wurde jedoch auch aufgezeigt, dass das verwendete Verteilungskonzept noch Defizite aufweist. Zur Verbesserung des Verfahrens könnte beispielsweise auf Basis der zu einem Entscheidungszeitpunkt während der Berechnung bereits vorliegenden Teilergebnisse mit Hilfe der nichtlinearen Ausgleichsrechnung eine Abschätzung für die maximal erreichbare All-Terminal-Zuverlässigkeit erfolgen. Auf diese Weise kann während der verteilten Berechnung eine Abschätzung der maximalen All-Terminal-Zuverlässigkeit erfolgen und die Berechnung z. B. bei Vorgabe einer maximal geforderten, aber nicht erreichbaren, All-Terminal-Zuverlässigkeit frühzeitig gestoppt werden. Ein anderer Ansatzpunkt im Bereich der Zuverlässigkeitsbewertung von Kommunikationsinfrastrukuren ist die Einbeziehung der Störung bzw. des Ausfalls von Kommunikationsknoten bei der Zuverlässigkeitsberechnung.
Für die multikriterielle Planung wurde mit den beiden hybriden genetischen Algorithmen COMNETEA und TPNDA gezeigt, wie durch die Integration der lokalen Suche die Qualität der durch das Verfahren generierten Lösungen verbessert wird. Die dabei verwendeten Integrationskonzepte für die Kopplung einer populationsbasierten Suche mit einem Nachbarschaftssuchverfahren könnte ebenfalls in anderen Anwendungsbereichen Einsatz finden und dort zu einer Erhöhung der Lösungsqualität beitragen. Ein weiterer Ansatzpunkt für zukünftige Forschungsarbeiten liegt weiterhin in der Einbeziehung weiterer Planungsziele (wie z. B. Ausnutzung von Übertragungskapazitäten, Erfüllung von Kommunikationsanforderungen) bei der multikriteriellen Planung.

Anhang A

Testprobleme

A.1 Testproblem türkei19

Das Testproblem türkei19 wurde in [49] eingeführt. Tabelle A.1 zeigt die Entfernungsmatrix für das Problem. Mit Tabelle A.2 werden die zur Verfügung stehenden Technologieoptionen gezeigt.

Tabelle A.1: Entfernungsmatrix für das Netzwerk türkei19 (aus [49])

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	111	126	120	122	115	116	132	346	968	343	344	106	107	105	454	613	828	1261
2		15	15	17	5	6	243	458	1079	454	456	10	11	5	565	724	939	1342
3			15	17	13	14	258	473	1094	469	471	25	26	23	580	740	954	1357
4				2	3	6	248	460	1082	456	457	12	13	15	570	730	943	1353
5					8	9	251	463	1085	459	460	15	16	18	573	733	946	1355
6						1	246	457	1080	454	455	10	9	12	568	728	940	1350
7							245	456	1079	453	454	9	8	11	567	727	939	1351
8								384	383	380	381	235	236	240	322	542	831	1301
9									766	3	4	450	451	453	580	542	487	920
10										763	764	1074	1075	1077	1345	1307	972	624
11											1	450	451	453	582	544	489	921
12												449	459	452	583	545	490	922
13													1	4	560	720	932	1337
14														3	561	721	933	1338
15															563	723	934	1340
16																469	898	1424
17																	553	1079
18																		526

Tabelle A.2: Kosten und Zuverlässigkeiten der Technologieoptionen für das Netzwerk türkei19 (aus [49])

Technologieoption (l_k)	Zuverlässigkeit	Kosten pro Entfernungseinheit
0 (nicht verbunden)	0	0
1	0,960	333
2	0,975	433
3	0,990	583

A.2 Testproblem deeter10

Das Testproblem deeter10 wurde in [49] eingeführt. Tabelle A.3 zeigt die Entfernungsmatrix für das Problem. Mit Tabelle A.4 werden die zur Verfügung stehenden Technologieoptionen gezeigt.

Tabelle A.3: Entfernungsmatrix für das Netzwerk deeter10 (aus [49])

	2	3	4	5	6	7	8	9	10
1	47.3995	41.7519	24.999	48.876	54.7605	33.7527	34.6182	43.7072	27.3028
2		20.3195	28.9359	39.9203	31.7823	65.4767	68.7027	32.1302	62.2745
3			34.7425	19.8154	15.4418	49.4191	53.0689	11.8351	64.1167
4				50.814	50.1819	54.7977	56.7332	43.0894	33.3708
5					13.6208	41.8359	45.8073	8.2105	75.0041
6						54.1884	58.1259	12.0843	78.7647
7							3.97178	42.6274	58.0398
8								46.5197	57.6986
9									68.8805

Tabelle A.4: Kosten und Zuverlässigkeiten der Technologieoptionen für das Netzwerk deeter10 (aus [49])

Technologieoption (l_k)	Zuverlässigkeit	Kosten pro Entfernungseinheit
0 (nicht verbunden)	0	0
1	0,7	8
2	0,8	10
3	0,9	14

A.3 Testprobleme Deutschland

Mit dieser Arbeit werden die Netzwerke deutsch15, deutsch20, deutsch25 und deutsch30 neu eingeführt. Die Knoten der Testprobleme repräsentieren jeweils die 15, 20 ,25 bzw. 30 größten Städte Deutschlands (nach [181, S. 53ff]). Mit Tabelle A.5 werden die 30 größten Städte Deutschlands mit ihren Koordinaten gezeigt. Tabelle A.6 und Tabelle A.7 zeigen für die 30 größten Städte die Entfernungsmatrix. Tabelle A.8 zeigt die zur Verfügung stehenden Technologieoptionen.

Tabelle A.5: Die 30 größten deutschen Städte mit geographischen Koordinaten

Nr.	Stadt	östl. Länge	nördl. Breite
1.	Berlin	13°20'	52°52'
2.	Hamburg	10°03'	53°56'
3.	München	11°56'	48°13'
4.	Köln	6°94'	50°94'
5.	Bremen	8°81'	53°08'
6.	Essen/Ruhr	7°02'	51°46'
7.	Frankfurt am Main	8°68'	50°12'
8.	Stuttgart	9°18'	48°78'
9.	Dortmund	7°46'	51°51'
10.	Düsseldorf	6°78'	51°23'
11.	Duisburg	6°76'	51°43'
12.	Dresden	13°73'	51°05'
13.	Hannover	9°73'	52°38'
14.	Nürnberg	11°08'	49°45'
15.	Leipzig	12°38'	51°33'
16.	Wuppertal	7°20'	51°26'
17.	Bochum	7°21'	51°48'
18.	Mannheim	8°46'	49°48'
19.	Chemnitz	12°91'	50°83'
20.	Bielefeld	8°53'	52°02'
21.	Bonn	7°10'	50°73'
22.	Magdeburg	11°61'	52°12'
23.	Gelsenkrichen	7°10'	51°54'
24.	Münster	7°63'	51°96'
25.	Wiesbaden	8°23'	50°10'
26.	Karlsruhe	8°40'	49°02'
27.	Aachen	6°08'	50°78'
28.	Mönchengladbach	6°43'	51°20'
29.	Rostock	12°13'	54°08'
30.	Braunschweig	10°53'	52°26'

Tabelle A.6: Entfernungsmatrix für die Netzwerke deutsch2-15

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	-	232	467	536	355	430	414	528	429	496	496	112	273	362	131
2		-	561	432	133	303	362	560	302	352	352	304	130	451	262
3			-	427	595	442	312	149	442	491	492	363	468	112	342
4				-	360	130	143	312	131	110	110	507	305	377	438
5					-	232	334	559	232	261	261	408	130	491	352
6						-	133	363	1	69	69	421	178	362	350
7							-	233	133	180	180	371	234	243	304
8								-	363	396	397	441	444	184	397
9									-	70	70	420	178	362	350
10										-	1	490	236	420	419
11											-	490	235	420	419
12												-	299	264	70
13													-	362	235
14														-	234
15															-
16															
17															
18															
19															
20															
21															
22															
24															
25															
26															
27															
28															
29															
30															

Tabelle A.7: Entfernungsmatrix für die Netzwerke deutsch16-30

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	430	430	486	232	342	475	136	430	429	415	486	537	497	232	205
2	303	303	466	361	175	393	131	303	302	362	467	433	353	172	112
3	442	442	248	235	494	368	445	442	442	312	248	428	492	671	451
4	130	131	182	429	261	70	413	131	132	142	183	1	110	603	356
5	233	232	444	432	111	340	231	232	231	334	445	360	261	286	175
6	1	1	234	371	131	111	299	1	1	132	235	131	70	475	236
7	132	133	111	286	222	72	306	133	133	1	111	144	180	522	263
8	363	363	133	313	449	266	466	363	364	233	132	312	397	698	450
9	1	1	234	371	130	111	298	1	1	132	234	132	70	474	235
10	69	69	264	438	177	131	363	69	70	180	264	110	1	525	298
11	69	69	264	439	177	131	363	69	70	180	265	111	1	524	298
12	420	420	421	130	364	439	178	421	420	371	421	508	490	41	236
13	178	178	341	305	69	263	137	178	177	234	342	306	236	299	68
14	362	362	218	134	394	310	333	362	362	243	218	378	420	560	341
15	350	350	362	110	298	371	130	350	350	304	362	439	420	333	177
16	-	1	233	371	131	110	299	1	1	132	234	131	70	475	236
17		-	234	371	130	111	298	1	1	132	235	131	70	474	236
18			-	310	333	133	395	234	234	111	1	183	264	620	362
19				-	357	358	232	371	371	287	311	430	439	443	262
20					-	232	205	130	129	222	334	262	177	348	137
21						-	357	111	112	72	134	71	131	560	306
22							-	299	298	306	395	414	364	232	69
23								-	1	133	235	131	70	474	236
24									-	133	235	132	70	473	235
25										-	111	143	180	522	263
26											-	183	265	621	363
27												-	110	604	357
28													-	525	298
29														-	259
30															-

Tabelle A.8: Kosten und Zuverlässigkeiten der Technologieoptionen für die Netzwerke deutsch2-30

Technologieoption (l_k)	Zuverlässigkeit	Kosten pro Entfernungseinheit
0 (nicht verbunden)	0	0
1	0,7	8
2	0,8	10
3	0,9	14

Anhang B

Vergleich auf statistisch signifikanten Unterschied zweier mittels einfacher Monte-Carlo-Simulation ermittelter All-Terminal-Zuverlässigkeiten

Durch den Einsatz der einfachen Monte-Carlo-Simulation als Verfahren zur Bestimmung der All-Terminal-Zuverlässigkeit ergibt sich für einige der im Rahmen der Arbeit vorgeschlagenen Planungsmethoden die Notwendigkeit des Vergleichs auf einen statistisch signifikanten Unterschied der Ergebnisse für die Zuverlässigkeitsschätzung. Ein solcher Vergleich zweier Prozentsätze $\hat{p}_1$ und $\hat{p}_2$ (mit $\hat{p}_1 > \hat{p}_2$), wie sie die All-Terminal-Zuverlässigkeiten darstellen, ist mittels des angenäherten Vertrauensbereiches (hier 95%) zweier unabhängiger Zufallsstichproben π_1 und π_2 (bei großen Stichprobenumfängen $n_1 \gtrapprox 50$ und $n_2 \gtrapprox 50$) möglich (vgl. hierzu [164, S. 442ff]). Nach [164, S. 443] gilt für den Vergleich:

$$\hat{p}_1 - \hat{p}_2 - 1,96 \cdot \sqrt{z} \lessapprox \pi_1 - \pi_2 \lessapprox \hat{p}_1 - \hat{p}_2 + 1,96 \cdot \sqrt{z}$$
$$\text{mit } z = \frac{\hat{p}_1 \cdot (1 - \hat{p}_1)}{n_1} + \frac{\hat{p}_2 \cdot (1 - \hat{p}_2)}{n_2} \tag{B.1}$$

Nach [164, S. 443] lässt sich Gleichung B.1 verbessern, indem man für $\hat{p}_1 - \hat{p}_2 < 0$ zu der Differenz den Wert $0,5 \cdot (\frac{1}{n_1} + \frac{1}{n_2})$ addiert, beziehungsweise diesen für $\hat{p}_1 - \hat{p}_2 > 0$ subtrahiert.
Am Beispiel für den Vergleich zweier mittels einfacher Monte-Carlo-Simulation ermittelter Schätzungen $R^1_{All} = 0,9963$ bei einem Stichprobenumfang von $M_1 = 30000$ und $R^2_{All} = 0,9945$ bei einem Stichprobenumfang von $M_2 = 30000$ ergibt sich als angenäherter 95%-Vertrauensbereich:

$$z = \frac{0,9963 \cdot (1 - 0,9963)}{30000} + \frac{0,9945 \cdot (1 - 0,9945)}{30000} = 0,0005525$$

$$1,96 \cdot 0,0005525 = 0,0010828; \qquad 0,5 \cdot (\frac{1}{30000} + \frac{1}{30000}) = 0,0000333$$

$$0,9963 - 0,9945 - 0,0000333 = 0,0017667; \qquad 0,0017667 \pm 0,0010828$$

$$0,0006836 \lessapprox \pi_1 - \pi_2 \lessapprox 0,0028494$$

Da die Null ausgeschlossen ist, besteht zwischen den geschätzen All-Terminal-Zuverlässigkeiten R^1_{All} und R^2_{All} auf dem Niveau 5 % ein statistisch signifikanter Unterschied.

Literaturverzeichnis

[1] A. White, G. S.: Genetic algorithms and network ring design. In: *Annals of Operations Research* 86 (1999), Nr. 0, S. 347 – 371

[2] Aboelfotoh, H.M.F. ; Al-Sumait, L.S.: A neural approach to topological optimization of communication networks, with reliability constraints. In: *IEEE Transactions on Reliability* 50 (2001), Nr. 4, S. 397–405

[3] Aggarwal, K. K. ; Chopra, Y. C. ; Bajwa, J. S.: Topological layout of links for optimising the overall reliability in a computer communication system. In: *Microelectronics and Reliability* 22 (1982), S. 347–351

[4] Ahmad, S. H.: A Simple Technique for Computing Network Reliability. In: *IEEE Transactions on Reliability* R-31 (1982), Nr. 1, S. 41–44

[5] Alt, R. ; Lenger, C. ; Österle, H.: Virtuelle Organisationen - Konzept, Realität und Umsetzung. In: *HMD – Praxis der Wirtschaftsinformatik* 242 (2005), April, S. 7–19

[6] Altiparmak, F. ; Dengiz, B. ; Smith, A.: Optimal Design of Reliable Computer Networks: A Comparison of Metaheuristics. In: *Journal of Heuristics* 9 (2003), Nr. 6, S. 471–487

[7] Arabas, J. ; Kozdrowski, S.: Applying an evolutionary algorithm to telecommunication network design. In: *IEEE Transactions on Evolutionary Computing* 5 (2001), Nr. 4, S. 309–322

[8] Atiqullah, M. ; Rao, S.: Reliability Optimization of communication networks using simulated annealing. In: *Microelectronics and Reliability* 33 (1993), Nr. 9, S. 1303–1319

[9] Backes, J. ; Koglin, H.-J. ; Klein, L.: Network Planning Under Economic Aspects with Special Regard to Reliability. In: *CIRED 1997: Proceedings of 4th International Conference and Exhibition on Electricity Distribution* Bd. 6, IEE Conference Publishing No. 438, 1997, S. 6.2.1–6.2.5

[10] Baker, J. E.: Reducing bias and inefficiency in the selection algorithm. In: *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application.* Mahwah, NJ, USA : Lawrence Erlbaum Associates, Inc., 1987, S. 14–21

[11] Baldwin, J.M.: A new factor in evolution. In: *American Naturalist* 30 (1896), S. 441–451

[12] Ball, M. O.: Computational Complexity of Network Reliability Analysis: An Overview. In: *IEEE Transactions on Reliability* R-35 (1986), Nr. 3, S. 230–239

[13] Bamberg, G. ; Coenenberg, G.: *Betriebswirtschaftliche Entscheidungstheorie.* 10. München : Vahlen-Verlag, 2000

[14] Baran, B. ; Duarte, S. ; Benítez, D.: Telecommunication Network Design with Parallel Multi-objective Evolutionary Algorithms. In: *IFIP/ACM Latin America Networking Conference.* New York : ACM Press, 2003

[15] Baran, B. ; Laufer, F.: Topological Optimization of Reliable Networks using A-Teams. In: *Proceedings of World Multiconference on Systemics, Cybernetics and Informatics - SCI '99 and ISAS '99* Bd. 5. Orlando, Florida, USA, 1999

[16] Bause, F. ; Kaczmarek, M.: Modellierung und Analyse von Supply Chains. In: *Wirtschaftsinformatik* 43 (2001), Nr. 6, S. 569–578

[17] Beyer, H.-G. ; Schwefel, H.-P.: Evolution strategies – A comprehensive introduction. In: *Natural Computing: an international journal* 1 (2002), Nr. 1, S. 3–52. – ISSN 1567–7818

[18] Biethahn, J. ; Mucksch, H. ; Ruf, W.: *Ganzheitliches Informationsmanagement.* 6. München : Oldenburg-Verlag, 2004

[19] Blum, C. ; Roli, A.: Metaheuristics in Combinatorial Optimization; Overview and Conceptual Comparison. In: *ACM Computer Surveys* 35 (2003), Nr. 3, S. 268–308

[20] Böhmann, T. ; Krcmar, H.: Grundlagen und Entwicklungstrends im IT-Servicemanagement. In: *HMD – Praxis der Wirtschaftsinformatik* 237 (2004), S. 7–21

[21] Bonabeau, E. ; Dorigo, M. ; Theraulaz, G.: *Swarm intelligence: from natural to artificial systems.* New York, NY, USA : Oxford University Press, Inc., 1999

[22] Brandstädt, A.: *Graphen und Algorithmen.* Stuttgart : Teubner-Verlag, 1994

[23] Brenner, W.: *Konzepte des Informationsmanagements.* 1. Heidelberg : Physica-Verlag, 1994

[24] Bretzke, W.-R.: *Der Problembezug von Entscheidungsmodellen.* Tübingen : Mohr-Verlag, 1980

[25] Bronstein, I. N. ; Semendjajew, K. A. ; Musiol, G. ; Mühlig, H.: *Taschenbuch der Mathematik.* 5. Verlag Harri Deutsch AG Thun, 2001

[26] Brooks, R. L. ; Smith, C. A. B. ; Stone, A. H. ; Tutte, W. T.: The dissection of rectangles into squares. In: *Duke Mathematical Journal* 7 (1940), Nr. 1, S. 312–340

[27] Bundesamt für Sicherheit in der Informationstechnik: *IT-Grundschutzhandbuch.* Mai 2005. – URL: `http://www.bsi.bund.de/gshb/deutsch` abgerufen am 22.7.2005

[28] Busch, A. (Hrsg.) ; Dangelmaier, W. (Hrsg.): *Integriertes Supply Chain Management.* 1. Wiesbaden : Gabler-Verlag, 2002

[29] Buxmann, P.: Elektronische Marktplätze und Supply Chain Management. In: *Wirtschaftsinformatik* 43 (2001), Nr. 6, S. 541

[30] Cancela, H. ; Rubion, G. ; Urquhart, M.: An algorithm to compute all-terminal reliability measure. In: *OPSearch* 38 (2001), Nr. 6, S. 567–579

[31] Cancela, H. ; Urquhart, M.: Simulated annealing for communication network reliability improvement. In: *Proceedings of the XXI Latin American Conference On Informatics (PANEL'95).* New York : ACM Press, 1995, S. 1413–1424

[32] Chen, Y. ; Li, J. ; Chen, J.: A New Algorithm For Network Probabilistic Connectivity. In: *Proceedings of Military Communications Conference, MILCOM' 99* Bd. 2. Atlantic City, 1999, S. 920–923

[33] Cheng, S.-T.: Topological Optimization of a Reliable Communication Network. In: *IEEE Transactions on Reliability* 47 (1998), Nr. 3, S. 225–233

[34] Clouqueur, M. ; Grover, W. D.: Computational and design studies on the unavailability of meshrestorable networks. In: *Proceedings of Second International Workshop on the Design of Reliable Communication Networks, DRCN 2000.* München, 2000, S. 181–186

[35] Coello Coello, C. A. ; Van Veldhuizen, D. A. ; Lamont, Gary B.: *Evolutionary Algorithms for Solving Multi-Objective Problems.* New York : Kluwer Academic Publishers, 2002

[36] Colbourn, C. J.: *The Combinatorics of Network Reliability.* Oxford University Press, 1987

[37] Corne, D. ; Knowles, J.: Some Multiobjective Optimizers are Better than Others. In: *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)* Bd. 4. Canberra, Australia : IEEE Press, Dezember 2003, S. 2506–2512

[38] Corne, D. ; Knowles, J. D.: No Free Lunch and Free Leftovers Theorems for Multiobjective Optimisation Problems. In: C. M. Fonseca (Hrsg.) u. a.: *Evolutionary Multi-Criterion Optimization, Second International Conference, EMO 2003, Faro, Portugal, April 8-11, 2003, Proceedings*, Springer-Verlag, 2003 (LNCS 2632), S. 327–341

[39] Corne, D. (Hrsg.) ; M. J. Oates (Hrsg.) ; Smith, G. D. (Hrsg.): *Telecommunications optimization: heuristic and adaptive techniques.* John Wiley & Sons, 2000

[40] Craenen, B.G.W. ; Eiben, A.E. ; E.Marchiori: How to Handle Constraints with Evolutionary Algorithms. In: Chambers, L. (Hrsg.): *Practical Handbook Of Genetic Algorithms: Applications.* Chapman & Hall/CRC, 2000, S. 341–361

[41] Czyzak, P. ; Jaszkiewicz, A.: Pareto Simulated Annealing - A Metaheuristic Technique for Multiple-Objective Combinatorial Optimization. In: *Journal of Multi-Criteria Decision Analysis* 7 (1998), S. 34–47

[42] Cäsar, M. A. ; Alt, R. ; Grau, J. U.: Elektronische Marktplätze im Handel und Konsumgüterbereich. In: *HMD – Praxis der Wirtschaftsinformatik* 223 (2002), S. 20–30

[43] Darwin, C.: *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life.* 1. London : John Murray, 1859

[44] De Jong, K. A.: *An analysis of the behavior of a class of genetic adaptive systems*, University of Michigan, Diss., 1975

[45] Deb, K.: *Multi-objective optimization using evolutionary algorithms.* 1. Chichester : John Wisly & Son Ltd., 2001

[46] Deb, K. ; Goel, T.: A Hybrid Multi-Objective Evolutionary Approach to Engineering Shape Design. In: Zitzler, E. (Hrsg.) u. a.: *First International Conference on Evolutionary Multi-Criterion Optimization.* Springer-Verlag, 2001, S. 385–399

[47] Deb, K. ; Pratap, A. ; Agarwal, S. ; Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. In: *IEEE Transactions on Evolutionary Computing* 6 (2002), Nr. 2, S. 182–197

[48] Deeter, D. ; Smith, A.: Heuristic optimization of network design considering all-terminal reliability. In: *Proceedings of the Annual Reliability and Maintainability Symposium*, IEEE Press, 1997, S. 194–199

[49] Deeter, D. ; Smith, A. E.: Economic design of reliable networks. In: *IIE Transactions* 30 (1998), S. 1161–1174

[50] Dengiz, B. ; Alabap, C.: A Simulated Annealing Algorithm for Design of Computer Communication Networks. In: Callaos, Nagib (Hrsg.) u. a.: *Proceedings of World Multiconference on Systemics, Cybernetics and Informatics, SCI 2001* Bd. 5. Orlando : IIIS, 2001, S. 188–193

[51] Dengiz, B. ; Altiparmak, F. ; Smith, A. E.: A genetic algorithm approach to optimal topological design of all terminal networks. In: *Intelligent Engineering Systems Through Artificial Neural Networks, Volume 5*, ASME Press, 1995, S. 405–410

[52] Dengiz, B. ; Altiparmak, F. ; Smith, A. E.: Local Search Genetic Algorithm for Optimal Design of Reliable Networks. In: *IEEE Transactions on Evolutionary Computation* 1 (1997), Nr. 3, S. 179–188

[53] Diestel, R.: *Graphentheorie.* 2. Berlin u.a. : Springer-Verlag, 2000

[54] *Kapitel* „The ant colony optimization meta-heuristic“. In: Dorigo, M. ; Caro, G. di: *New ideas in optimization.* Maidenhead, UK, England : McGraw-Hill Ltd., UK, 1999, S. 11–32

[55] Draves, S.: The Electric Sheep Screen-Saver: A Case Study in Aesthetic Evolution. In: Rothlauf, F. (Hrsg.) u. a.: *Applications of Evolutionary Computing, EvoWorkshops 2005: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, and EvoSTOC, Lausanne, Switzerland, March 30 - April 1, 2005, Proceedings* Bd. 3449, Springer-Verlag, 2005, S. 458–467

[56] Easton, M. ; Wong, C. K.: Sequential Destruction Method for Monte Carlo Evaluation of System Reliability. In: *IEEE Transactions on Reliability* 29 (1980), Nr. 1, S. 27–32

[57] Ehrgott, M. ; Gandibleux, X.: A Survey and Annotated Bibliography of Multiobjective Combinatorial Optimization. In: *OR Spektrum* 22 (2000), S. 425–460

[58] Eiben, A. ; Smith, J.: *Introduction to evolutionary computing.* Berlin, Heidelberg : Springer-Verlag, 2003

[59] Fard, N. ; Lee, T.-H.: Spanning tree approach in all-terminal network reliability expansion. In: *Computer Communications* 24 (2001), S. 1348–1353

[60] Feldmann, M.: *Naturanaloge Verfahren: Metaheuristiken zur Reihenfolgeplanung.* Wiesbaden : Gabler-Verlag und Deutscher Universitäts-Verlag, 1999

[61] Fink, A. ; Voss, S.: Anwendungen von Metaheuristiken zur Lösung betrieblicher Planungsprobleme. In: *Wirtschaftsinformatik* 45 (2003), Nr. 4, S. 395–407

[62] Fishman, G.S.: *Monte Carlo Simulation.* New York : Springer-Verlag, 1995

[63] Flores, S. D. ; Cegla, B. B.: Multiobjective Network Design Optimization Using Parallel Evolutionary Algorithms. In: Montilva, C. (Hrsg.) ; Besembel, I. (Hrsg.): *Proceedings of XXVII Conferencia Latinoamericana de Informática CLEI'2001.* Merida, Venezuela, 2001, S. 112–122

[64] Fogel, L. ; Owens, A.J. ; Walsh, M.J.: *Artificial intelligence through simulated evolution.* 1. John Wiley, New York, 1966

[65] Fong, C.C. ; Buzacott, J.A.: An Algorithm for Symbolic Reliability Computation with Path-sets or Cut-sets. In: *IEEE Transactions on Reliability* 36 (1987), Nr. 1, S. 34–37

[66] Fonseca, C. M. ; Fleming, P. J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In: Forrest, S. (Hrsg.): *Proceedings of the Fifth International Conference on Genetic Algorithms.* San Mateo, California : Morgan Kauffman Publishers, 1993, S. 416–423

[67] Fröschle, H.-P.: CRM - Unterstützungspotential. In: *HMD – Praxis der Wirtschaftsinformatik* 221 (2001), S. 5–12

[68] Garey, M. R. ; Johnson, D. S.: *Computers and Intractibility: A Guide to the Theory of NP-Completeness.* San Fransisco : W. H. Freeman and Company, 1979

[69] Ghosh, L. ; Mukherjee, A. ; Saha, D.: Design of 1-FT Communication Network under Budget Constraint. In: *IWDC '02: Proceedings of the 4th International Workshop on Distributed Computing, Mobile and Wireless Computing.* London, UK : Springer-Verlag, 2002, S. 300–311

[70] Glover, F. ; Laguna, M.: *Tabu Search.* Kluwer Acadamic Publisher, 1997

[71] Glover, F. ; Lee, F. ; Ryan., J.: Least-cost network topology design for a new service. In: *Annals of Operations Research* 33 (1991), S. 351–362

[72] Glover, F. ; McMillan, C.: The general employee scheduling problem: an integration of MS and AI. In: *Computers and operations research* 13 (1986), Nr. 5, S. 563–573

[73] Goldberg, D. E.: *Genetic algorithms in search, optimization, and machine learning.* 26. Printing. Reading, MA : Addison-Wesley, 2004

[74] Goldberg, D. E. ; Korb, B. ; Deb, K.: Messy genetic algorithms: motivation, analysis, and first results. In: *Complex Systems* 3 (1989), Nr. 5, S. 493–530

[75] Goldberg, D. E. ; Sastry, Kumara: Genetic and Evolutionary Algorithms in the Real World / IlliGAL. 1999 (1999013). – Forschungsbericht. Url: `ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/99013.ps.Z` abgerufen am: 1.7.05

[76] Gottlieb, J.: *Evolutionary Algorithms for Constrained Optimization Problems.* Clausthal, Technische Universität Clausthal, Institut für Informatik, Diss., 1999

[77] Grochla, E.: *Handwörterbuch der Organisation.* Stuttgart : Pöschel-Verlag, 1969

[78] Grötschel, M. ; Monma, C.L. ; Stoer, M.: Design of Survivable Networks. In: Ball, M. O. (Hrsg.) u. a.: *Network Models* Bd. 7. North-Holland, 1995, S. 617–672

[79] Hansen, H. R. ; Neumann, G.: *Wirtschaftsinformatik I – Grundlagen der betrieblichen Informationsverarbeitung.* 8. Stuttgart : Lucius und Lucius-Verlag, 2001

[80] Hansen, P. ; Mladenovic, N.: Variable neighborhood search: Principles and applications. In: *European Journal of Operational Research* 130 (2001), Nr. 3, S. 449–467

[81] Harik, G. ; Cantú-Paz, E. ; Goldberg, D. E. ; Miller, B. L.: The Gambler's Ruin Problem, Genetic Algorithms, and the Sizing of Populations. In: *Evolutionary Computation* 7 (1999), Nr. 3, S. 231–253

[82] Haverkorta, B. R. (Hrsg.) ; Marie, R. (Hrsg.) ; Rubino, G. (Hrsg.) ; Trivedi, K. S. (Hrsg.): *Performability Modelling : Techniques and Tools.* Chichester u.a. : John Wiley & Sons, 2001

[83] Heilmann, H.: Integration: Ein zentraler Begriff der Wirtschaftsinformatik im Wandel der Zeit. In: *HMD – Theorie und Praxis der Wirtschaftsinformatik* 150 (1998), S. 46–58

[84] Heinrich, L. ; Heinzl, A. ; Roithmayr, F.: *Wirtschaftsinformatik-Lexikon.* 7. München Wien : Oldenburg-Verlag, 2004

[85] Heinrich, L. J.: *Informationsmanagement: Planung, Überwachung, Steuerung der Informationsinfrastruktur.* 5. München : Oldenburg-Verlag, 1996

[86] Hewitt, J. ; Soper, A. ; McKenzie, S.: Charley: A Genetic Algorithm for the design of mesh networks. In: *Proceedings of 1st International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, IEE, 1995, S. 118–122

[87] Hildebrandt, K.: *Informationsmanagement.* 2. München, Wien : Oldenburg-Verlag, 2001

[88] Holland, J. H.: *Adaption in natural and artificial systems.* 2. Druck, 1. MIT Press, 1993. – 1. Auflage 1975: The University of Michigan Press

[89] Horn, J. ; Nafpliotis, N. ; Goldberg, D. E.: A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In: *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence* Bd. 1. Piscataway, New Jersey : IEEE Service Center, June 1994, S. 82–87

[90] Hui, K.-P. ; Bean, N. ; Kraetzl, M. ; Krosse, D.: The Tree Cut and Merge Algorithm for estimation of network reliability. In: *Probability in the Engineering and Informational Sciences* 17 (2003), S. 23–45

[91] Humpert, F.: IT-Sicherheit. In: *HMD – Praxis der Wirtschaftsinformatik* 236 (2004), S. 7–18

[92] Igel, C. ; Toussaint, M.: On classes of functions for which No Free Lunch results hold. In: *Information Processing Letters* 86 (2003), Nr. 6, S. 317–321

[93] Imai, H. ; Sekine, K. ; Imai, K.: Computational Investigation of All-Terminal Network Reliability via BDDs. In: *IEICE Transactions Fundamentals* E82-A (1999), Nr. 5, S. 714–721

[94] Ishibuchi, H. ; Murata, T.: Multi-Objective Genetic Local Search Algorithm. In: *International Conference on Evolutionary Computation*, 1996, S. 119–124

[95] ISO/IEC: *International Standard ISO/IEC 17799, Reference Number ISO/IEC 17799:2000(E), Information Technology – Code of practice for information security management.* 2000

[96] Jan, R.-H. ; Hwang, F.-J. ; Chen, S.-T.: Topological optimization of a communication network subject to a reliability constraint. In: *IEEE Transactions on Reliability* 42 (93), Nr. 1, S. 63–70

[97] Jaszkiewicz, A.: *MOMHLib++: Multiple Objective MetaHeuristics Library in C++.* URL: `http://www-idss.cs.put.poznan.pl/~jaszkiewicz/MOMHLib` abgerufen am 22.7.2005,

[98] Jaszkiewicz, A.: Genetic local search for multiple objective combinatorial optimization / Institute of Computing Science, Poznan University of Technology. 1998 (RA-014/98). – Arbeitsbericht

[99] Jaszkiewicz, A.: Evaluation of Multiple Objective Metaheuristics. In: Gandibleux, Xavier (Hrsg.) u. a.: *Metaheuristics for Multiobjective Optimisation.* Berlin : Springer-Verlag, 2004, S. 65–89

[100] Johnson, D. S. ; McGeoch, L. A.: The Traveling Salesman Problem: A Case Study. In: Aarts, E. H. L. (Hrsg.) ; Lenstra, J. K. (Hrsg.): *Local Search in Combinatorial Optimization.* Wiley and Sons, New York, 1997, S. 215–310

[101] Karger, D. R.: A Randomized Fully Polynomial Time Approximation Scheme for the All-Terminal Network Reliability Problem. In: *SIAM Review* 43 (2001), Nr. 3, S. 499–522

[102] Kimbrough, S. O. ; Lu, M. ; Wood, D. H. ; Wu, D.J.: Exploring a Two-Population Genetic Algorithm. In: Cantu-Paz, E. (Hrsg.) u. a.: *Proceedings of the Genetic and Evolutionary Computation Conference 2003.* Berlin : Springer-Verlag, 2003, S. 1148–1159

[103] Kirchhoff, G.: Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Verteilung galvanischer Ströme geführt wird. In: *Annalen der Physik und Chemie* 72 (1847), S. 497–508

[104] Kirkpatrick, C. Gelatt und M. V.: Optimization by simulated annealing. In: *Science* 220 (1983), S. 671–680

[105] Klein, S. ; Gogolin, M. ; Dziuk, M.: Elektronische Marktplätze im Überblick. In: *HMD – Praxis der Wirtschaftsinformatik* 223 (2002), S. 7–19

[106] Knowles, J. ; Corne, D.: M-PAES: A Memetic Algorithm for Multiobjective Optimization. In: *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*. La Jolla Marriott Hotel La Jolla, California, USA : IEEE Press, 6-9 2000, S. 325–332

[107] Knowles, J. ; Corne, D.: Bounded Pareto Archiving: Theory and Practice. In: Gandibleux, X. (Hrsg.) u. a.: *Metaheuristics for Multiobjective Optimisation*. Berlin : Springer-Verlag, 2004, S. 39–64

[108] Knowles, J. D. ; Corne, D. W.: The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimisation. In: *1999 Congress on Evolutionary Computation*. Washington, D.C. : IEEE Service Center, July 1999, S. 98–105

[109] Koch, O. ; Weigl, H.: Traffic and road planning simulation: modeling ambulance service of the austrian Red Cross. In: *WSC '03: Proceedings of the 35th conference on Winter simulation*, Winter Simulation Conference, 2003, S. 1701–1706

[110] Koide, T. ; Shinmori, S. ; Ishii, H.: Topological optimization with a network reliability constraint. In: *Discrete Applied Mathematics* 115 (2001), Nr. 1-3, S. 135–149

[111] Konak, A. ; Smith, A.: *An improved general upperbound for all-terminal network reliability*. 1998. – URL `http://citeseer.nj.nec.com/495815.html` abgerufen am 22.7.2005

[112] Konak, A. ; Smith, A. E.: Multiobjective Optimization of Survivable Networks Considering Reliability. In: *Proceedings of The 10th International Conference on Telecommunication Systems, Modeling and Analysis (ICTSM10)*, 2002

[113] Kost, B.: *Optimierung mit Evolutionsstrategien*. Harri Deutsch-Verlag, 2003

[114] Koza, J.: Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems / Dept. of Computer Science, Stanford University. 1990 (STAN-CS-90-1314). – Arbeitsbericht

[115] Koza, J.: *Genetic Programming*. Cambridge, MA : MIT Press, 1992

[116] Koza, J.: *Genetic Programming II*. Cambridge, MA : MIT Press, 1994

[117] Koza, J.: *Genetic Programming III*. San Francisco : Morgan Kaufmann Publisher, 1999

[118] Krcmar, H.: *Informationsmanagement*. 3. Berlin u.a. : Springer-Verlag, 2003

[119] Krcmar, H. ; Son, S.: IV-Controlling. In: *Wirtschaftsinformatik* 46 (2004), Nr. 3, S. 165–166

[120] Kumar, A. ; Pathak, R. M. ; Gupta, M. C.: Genetic algorithm based approach for designing computer network topology. In: *CSC '93: Proceedings of the 1993 ACM conference on Computer science*. New York : ACM Press, 1993. – ISBN 0–89791–558–5, S. 358–365

[121] Kumar, A. ; Pathak, R. M. ; Gupta, Y. P.: Genetic Algorithm Based Reliability Optimization for Computer Network Expansion. In: *IEEE Transactions on Reliability* 44 (1995), Nr. 1, S. 63–72

[122] Kumar, R. ; Banerjee, N.: Multicriteria Network Design Using Evolutionary Algorithm. In: Cantú-Paz, E. (Hrsg.) u. a.: *Genetic and Evolutionary Computation - GECCO 2003, Genetic and Evolutionary Computation Conference, Chicago, IL, USA, July 12-16, 2003. Proceedings, Part II* Bd. 2724, Springer-Verlag, 2003, S. 2179–2190

[123] Kumar, R. ; Parida, P. P. ; Gupta, M.: Topological Design of Communication Networks using Multiobjective Genetic Optimization. In: *Congress on Evolutionary Computation (CEC'2002)* Bd. 1. Piscataway, New Jersey : IEEE Service Center, May 2002, S. 425–430

[124] Lamarck, J.-B. de: *Philosophie Zoologique*. Paris : Chez Dentu L'Auteu, 1809

[125] Lamarck, J.-B. de: *Philosophie Zoologique*. University of Chicago, 1984. – Englische Übersetzung

[126] Laux, H.: *Entscheidungstheorie*. 5. Berlin u.a. : Springer-Verlag, 2003

[127] Lawler, E. L. ; Wood, D. E.: Branch-and-bound methods: A survey. In: *Operations Research* 14 (1966), Nr. 4, S. 699–719

[128] Lehner, F. ; Maier, R. ; Hildebrand, K.: *Wirtschaftsinformatik - Theoretische Grundlagen*. München : Carl Hanser-Verlag, 1995

[129] Lin, Min-Sheng ; Chen, Deng-Jyi ; Horng, Maw-Sheng: The reliability analysis of distributed computing systems with imperfect nodes. In: *The Computer Journal* 42 (1998), Nr. 2, S. 129–141

[130] Liu, B. ; Iwamura, K.: Topological Optimization Model for Communication Network with Multiple Reliability Goals. In: *Computer and Mathematics with Applications* 39 (2000), S. 59–69

[131] Lloyd-Evans, R.: *Wide Area Network Performance and Optimization: Practical Strategies for Success*. Harlow : Addison-Wesley, 1996

[132] Lourenço, H. R. ; Martin, O. C. ; Stuetzle, T.: Iterated Local Search / Department of Economics and Business, Universitat Pompeu Fabra. 2000 (513). – Arbeitsbericht. Url: `http://ideas.repec.org/p/upf/upfgen/513.html` abgerufen am: 15.11.03

[133] M. Cantoni, E. Z.: Genetic algorithms and Monte Carlo simulation for optimal plant design. In: *Reliability Engineering and System Safety* 68 (2000), S. 29–38

[134] Manzi, E. ; Labbe, M. ; Latouche, G. ; Maffioli, F.: Fishman's sampling plan for computing network reliability. In: *IEEE Transactions on Reliability* 50 (2001), Nr. 1, S. 41–46

[135] Marathe, M. V. ; Ravi, R. ; Sundaram, R. ; Ravi, S. S. ; Rosenkrantz, D. J. ; H. B. Hunt, III: Bicriteria network design problems. In: *Journal of Algorithms* 28 (1998), Nr. 1, S. 142–171

[136] Marseguerra, M. ; Zio, E. ; Coit, L. Podofilliniand D.: Optimal design of reliable network systems in presence of uncertainty. In: *IEEE Transactions on Reliability* 54 (2005), Nr. 2, S. 245–253

[137] Marseguerra, M. ; Zio, E. ; Podofillini, L.: Optimal Reliability/Availability of Uncertain Systems via Multi-Objective Genetic Algorithms. In: *IEEE Transactions on Reliability* 53 (2004), Nr. 3, S. 424–434

[138] Mertens, P. ; Griese, J.: *Integrierte Informationsverarbeitung 2 - Planungs- und Kontrollsysteme in der Industrie*. 9. Wiesbaden : Gabler-Verlag, 2002

[139] Mertens, P. ; Griese, J.: *Integrierte Informationsverarbeitung 1 - Operative Systeme in der Industrie*. 14. Wiesbaden : Gabler-Verlag, 2004

[140] Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*. 3. Berlin : Springer-Verlag, 1996

[141] Michalewicz, Z. ; Fogel, D. B.: *How to Solve It: Modern Heuristics*. 2. Berlin : Springer-Verlag, 2004

[142] Mitchell, M.: *An Introduction to Genetic Algorithms*. 1. Cambridge : MIT Press, 1998

[143] Monticone, L.C.: An implementation of the Buzacott algorithm for network global-reliability. In: *IEEE Transactions on Reliability* 42 (1993), Nr. 1, S. 46–49

[144] Mönch, L. ; Rose, O.: Shifting-Bottleneck-Heuristik für komplexe Produktionssysteme: softwaretechnische Realisierung und Leistungsbewertung. In: *Proceedings Multi-Konferenz Wirtschaftsinformatik, Teilkonferenz Quantitative Methoden in ERP und SCM*, 2004, S. 145–159

[145] Nissen, V.: *Einführung in evolutionäre Algorithmen: Optimierung nach dem Vorbild der Evolution*. Braunschweig, Wiesbaden : Vieweg-Verlag, 1997

[146] Nägeli, H.-P.: Management der Informationssicherheit - Erfahrungen eines Finanzdienstleisters. In: *HMD – Praxis der Wirtschaftsinformatik* 232 (2003), S. 79–88

[147] Pelikan, M.: *Bayesian optimization algorithm: From single level to hierarchy.* Urbana, IL, University of Illinois at Urbana-Champaign, Diss., 2002

[148] Picot, A. ; Reichwald, R. ; Wigand, R. T.: *Die grenzenlose Unternehmung.* 5. Wiesbaden : Gabler-Verlag, 2003

[149] Pietsch, T. ; Martiny, L. ; Klotz, M.: *Strategisches Informationsmanagement.* 4. Berlin : Erich Schmidt-Verlag, 2004

[150] Prasad, T. D. ; Park, Nam-Sik: Multiobjective Genetic Algorithms for Design of Water Distribution Networks. In: *Journal of Water Resources Planning and Management* 130 (2004), Nr. 1, S. 73–82

[151] Prim, R. C.: Shortest Connection Networks and Some Generalizations. In: *Bell System Technical Journal* 36 (1957), S. 1389–1401

[152] Raepple, M.: *Sicherheitskonzepte für das Internet.* 2. Heidelberg : dpunkt-Verlag, 2001

[153] Raepple, M.: Sicherheit in elektronischen Marktplätzen. In: *HMD – Praxis der Wirtschaftsinformatik* 223 (2002), S. 63–74

[154] Rechenberg, I.: *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* 1. Stuttgart : Frommann-Holzboog-Verlag, 1973

[155] Rechenberg, I.: *Evolutionsstrategie '94.* Stuttgard-Bad Cannstadt : Friedrich-Frommann-Verlag, 1994

[156] Reichelt, D. ; Gmilkowsky, P. ; Linser, S.: A Study of an Iterated Local Search on the Reliable Communication Networks Design Problem. In: Rothlauf, F. (Hrsg.) u. a.: *Applications of Evolutionary Computing, EvoWorkshops 2005: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, and EvoSTOC, Lausanne, Switzerland, March 30 - April 1, 2005, Proceedings* Bd. 3449, Springer-Verlag, 2005, S. 156–165

[157] Reichelt, D. ; Knauf, M.: Ein multikriterielles Verfahren zum Entwurf zuverlässiger und ökonomischer Netzwerktopologien / Technische Universität Ilmenau. 2004. – Forschungsbericht. URL: `http://www.wirtschaft.tu-ilmenau.de/~reichelt/pubs/workingpaper_sa_moo.%pdf`, abgerufen am 29.6.2005

[158] Reichelt, D. ; Rothlauf, F.: Designing Reliable Communication Networks with an Evolutionary Algorithms. In: *International Journal of Computational Intelligence and Applications* 5 (2005), Nr. 2, S. 251–266

[159] Reichelt, D. ; Rothlauf, F. ; Gmilkowsky, P.: Designing Reliable Communication Networks with a Genetic Algorithm Using a Repair Heuristic. In: Gottlieb, J. (Hrsg.) ; Raidl, G. R. (Hrsg.): *Evolutionary Computation in Combinatorial Optimization, 4th European Conference, EvoCOP 2004, Coimbra, Portugal, April 5-7, 2004, Proceedings* Bd. 3004, Springer-Verlag, 2004, S. 177–187

[160] Rommelfanger, H. J. ; Eickemeier, S. H.: *Entscheidungstheorie.* 1. Berlin u.a. : Springer-Verlag, 2002

[161] Rosenberg, R. S.: *Simulation of genetic populations with biochemical properties.* Ann Harbor, Michigan, University of Michigan, Diss., 1967

[162] Rothlauf, F.: *Representations for genetic and evolutionary algorithms.* Heidelberg : Physica-Verlag, 2002

[163] Rothlauf, F. ; Goldberg, D. E.: Redundant representations in evolutionary computation. In: *Evolutionary Computation* 11 (2003), Nr. 4, S. 381–415

[164] Sachs, L.: *Angewandte Statistik.* 10. Berlin u.a. : Springer-Verlag, 2002. – ISBN 3–540–40555–0

[165] Sayoud, H. ; Takahashi, K. ; Vaillant, B.: Designing communication network topologies using steady-state genetic algorithms. In: *IEEE Communications Letters* 5 (2001), Nr. 3, S. 113–115

[166] Schaffer, J. D.: *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*, Vanderbilt University, Diss., 1984

[167] Schöneburg, E. ; Heinzmann, F. ; Feddersen, S.: *Genetische Algorithmen und Evolutionsstrategien.* 1. Addison-Wesley, München u.a., 1994

[168] Schumacher, C. ; Vose, M. D. ; Whitley, L. D.: The No Free Lunch and Problem Description Length. In: Spector, L. (Hrsg.) u. a.: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001).* San Francisco, California, USA : Morgan Kaufmann, 7-11 July 2001, S. 565–570

[169] Schuster, E. ; Wilhelm, S.: Informationsmanagement in der Produktion - Neue Dienstleistungen durch E-Service. In: *HMD – Praxis der Wirtschaftsinformatik* 219 (2001), S. 43–53

[170] Schwefel, H.-P.: *Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik*, Technische Universität Berlin, Diplomarbeit, 1965

[171] Schwefel, H.-P.: *Evolutionsstrategie und numerische Optimierung*, Technische Universität Berlin, Diss., 1975

[172] Serafini, P.: Simulated Annealing for multiple objective optimization problems. In: *Multiple Criteria Decision Making. Expand and Enrich the Domains of Thinking and Application* (1994), S. 283–292

[173] Shooman, M. L.: *Reliability of Computer Systems and Networks: Fault Tolerance, Analysis and Design.* New York : John Wiley & Sons Inc., 2002

[174] Silverman, B.: *Density Estimation for Statistics and Data Analysis.* London : Chapman and Hall, 1986

[175] Sinz, E. ; Mertens, P.: Kopplung von Anwendungssystemen. In: *Wirtschaftsinformatik* 44 (2002), Nr. 5

[176] Smith, A. E. ; Dengiz, B.: Evolutionary Methods for the Design of Reliable Networks. In: Corne, David W. (Hrsg.) u. a.: *Telecommunications Optimization: Heuristic and Adaptive Techniques.* Wiley and Sons, 2000, S. 17–34

[177] Soni, S. ; Narasimhan, S. ; LeBlanc, L.J.: Telecommunication access network design with reliability constraints. In: *IEEE Transactions on Reliability* 53 (2004), Nr. 4, S. 532–541

[178] Spohn, D. L. ; Brown, T. ; Grau, S.: *Data network design.* 3. New York : McGraw-Hill, 2002 (Computer communication)

[179] Srinivas, N. ; Deb, K.: Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. In: *Evolutionary Computation* 2 (1994), Fall, Nr. 3, S. 221–248

[180] Stannat, A. ; Petri, C.: Trends in der Unternehmens-IT: Mittelfristige Entwicklungen in der Informationstechnologie und in IT-Organisationen aus Sicht von Unternehmen und öffentlichen Organisationen. In: *Informatik Spektrum* 27 (2004), Nr. 3, S. 227–237

[181] Statistisches Bundesamt (Hrsg.): *Statistisches Jahrbuch für die Bundesrepublik Deutschland.* Bd. 2003. Verlag Metzler-Poeschel, 2003

[182] Stoer, M. ; Wagner, F.: A Simple Min Cut Algorithm. In: *Algorithms - ESA '94 Second Annual European Symposium.* Berlin at al. : Springer-Verlag, 1994, S. 141–147

[183] Sucky, E.: Softwaregestützte Verhandlungen in Supply-Chains. In: *Wirtschaftsinformatik* 46 (2004), Nr. 5, S. 459–469

[184] Thiel, C.: Ein Reifegradmodell für das IT-Sicherheitsmanagement. In: *HMD – Praxis der Wirtschaftsinformatik* 236 (2004), S. 52–58

[185] Turau, V.: *Algorithmische Graphentheorie.* Bonn : Addison-Wesley, 1996

[186] Ulungu, E. L. ; Teghem, J. ; Frontemps, Ph. ; Tuyttens, D.: MOSA method: a tool for solving multiobjective combinatorial optimization problems. In: *Journal of Multi-Criteria Decision Analysis* 8 (1999), S. 221–236

[187] Van Veldhuizen, D. A.: *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations.* Ohio, Air Force Institute of Technology, Department of Electrical and Computer Engineering, Diss., 1999

[188] Wall, M.: *GALib: A C++ Library of Genetic Algorithm Components.* 1998. – URL: `http://lancet.mit.edu/ga` abgerufen am 22.7.2005

[189] Weicker, K.: *Evolutionäre Algorithmen*. Stuttgart : Teubner-Verlag, 2002

[190] Whitley, L. D. ; Gordon, V. S. ; Mathias, K. E.: Lamarckian Evolution, The Baldwin Effect and Function Optimization. In: *Proceedings of Parallel Problem Solving from Nature - PPSN III* Bd. 866, Springer-Verlag, 1994. – ISBN 3–540–58484–6, S. 6–15

[191] Wolpert, D. H. ; Macready, W. G.: No Free Lunch Theorems for Search / The Santa Fe Institute. Santa Fe, NM, 1995 (SFI-TR-95-02-010). – Forschungsbericht. `citeseer.nj.nec.com/wolpert95no.html` abgerufen am 4.2.2004

[192] Wolpert, D. H. ; Macready, W. G.: No Free Lunch Theorems for Optimization. In: *IEEE Transactions on Evolutionary Computation* 1 (1997), April, Nr. 1, S. 67–82

[193] Ying, L.: Analysis method of survivability of probabilistic Networks. In: *Military Communication Technology Magazine* 48 (1993)

[194] Zitzler, E. ; Laumanns, M. ; Bleuler, S.: A Tutorial on Evolutionary Multiobjective Optimization. In: *Metaheuristics for Multiobjective Optimisation* Bd. 535. Berlin : Springer-Verlag, 2004, S. 3–38

[195] Zitzler, E. ; Laumanns, M. ; Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In: *Proceedings of Evolutionary Methods for Design, Optimisation, and Control*, 2002, S. 95–100