Kommunizierende Genetische Algorithmen: Durch Evolution zur Kooperation

Jochen Schoof, 3SOFT GmbH, Wetterkreuz 19a, 91058 Erlangen
(Jochen.Schoof@3SOFT.de)

Einleitung

Die Kooperation zwischen Menschen und Computern gewinnt in zahlreichen Problemstellungen mehr und mehr an Bedeutung. Ein wesentlicher Grund hierfür ist die ständig wachsende Komplexität relevanter Problemstellungen. Dadurch bedingt sind weder der Mensch noch der Computer alleine in der Lage, zufriedenstellende Lösungen zu entwickeln. Die Kombination der individuellen Fähigkeiten hat sich in vielen Bereichen als gewinnbringend erwiesen. Beispiele aus dem täglichen Leben sind die Ermittlung einer nach bestimmten Kriterien optimalen Bahnverbindung oder auch die Suche nach bestimmten Inhalten im Internet. Hierbei werden aber zumindest auf Computerseite typischerweise Programme genutzt, die nicht für diese Kooperation vorgesehen sind und deshalb nur bedingt deren Anforderungen genügen.

Genetische Algorithmen (GA) als Repräsentanten der „Evolutionary Computation“ stellen einen Ansatz zur Lösung hochkomplexer Optimierungsaufgaben dar, der sich an den Vorgängen der Evolution orientiert. Im Gegensatz zu vielen anderen Optimierungsverfahren bringen sie einige Eigenarten mit, die kooperative Erweiterungen einfach und erfolgversprechend machen. Die praktische Erprobung zeigt, dass der Einsatz einer Variante Genetischer Algorithmen in Kooperation mit menschlichen Bedienern deutlich verbesserte Ergebnisse liefern kann, ohne die vom Programm allein gelieferten Ergebnisse zu gefährden. Diese Robustheit ist ein wesentliches Zielkriterium für viele Anwendungsfälle.

Kooperation zwischen Mensch und Computer

Versuche, schwierige Probleme in Kooperation von Mensch und Computer lösen zu lassen, blicken auf eine lange Tradition zurück. Um die aus einer solchen Kooperation erwachsenden Potenziale zu verdeutlichen, sollen zwei relativ bekannte Beispiele kurz genannt werden.

So wurde bereits 1971 in [KFM71] ein Verfahren zur kooperativen Lösung des Travelling Salesperson Problems (TSP) vorgestellt. Bei diesem Problem geht es darum, dass ein Handlungsreisender eine vorgegebene Zahl von Städten besuchen muss. Die Entfernungen aller Städte zueinander sind bekannt. Es soll nun die Streckenführung gefunden werden, die jede Stadt genau einmal erreicht und dabei die geringstmögliche Länge hat. Dieses Problem gilt in der Informatik als eine der schwierigsten Aufgabenstellungen, da der zur Lösung erforderliche Aufwand sehr viel schneller wächst als die Anzahl zu besuchender Städte. Heute sind dank leistungsfähiger Rechner Lösungen für sehr große Probleme bekannt, doch 1971 war mit damaliger Hardware noch nicht einmal ein Problem mit 100 Städten lösbar.

Die Vorgehensweise zur Lösung bestand darin, dass der Computer zunächst sehr nahe beieinander liegende Städte paarweise verband und so Gruppen bildete. Der Bediener gab nun an, welche Gruppen verbunden werden sollten. Das Programm ermittelte die kürzeste Verbindung dieser Gruppen und bildete so größere Gruppen. Durch iteratives Vorgehen kam man so schließlich zu einer Lösung.

Auf diese Weise wurden Probleme mit 100 Städten in etwa siebzig Minuten gelöst, während die besten exakten Lösungen von Programmen alleine mindestens die zwanzigfache Zeit benötigten. Die Kooperation zwischen Mensch und Computer führte also zu einem Ergebnis, dass Computer alleine erst Jahre später liefern konnten.

Der zweite Bereich von Arbeiten beschäftigt sich mit dem Schachspiel. Hier wurde von Althöfer 1985 das Dreihirn-Prinzip vorgestellt (siehe z.B. [Alt98]). Dabei koordiniert ein Mensch die Aktionen zweier Schach-Computer oder Schach-Programme. Dieses Gespann tritt als ein Schachspieler auf, indem die beiden Programme in jeder Stellung jeweils einen Zug berechnen. Wollen beide denselben Zug spielen, so führt der Koordinator diesen aus. Besteht zwischen den Programmen keine Einigkeit, so wählt er aus den beiden Vorschlägen den seiner Ansicht nach besseren Zug aus. Schon erste Ergebnisse zeigten ‚dass das Dreihirn besser spielt, als jeder seiner Bestandteile. Durch die Leistungsverbesserung der Schachprogramme und die in vielen Begegnungen geschulten Koordinationsfähigkeiten war eine Dreihirn-Variante schließlich in der Lage gegen einen Top Ten Schachgroßmeister in einem Match über acht Partien mit 5:3 zu gewinnen. Hierbei wurden handelsübliche Schachprogramme und PCs im Wert von wenigen tausend Mark benutzt. Zu vergleichbaren Leistungen war zur selben Zeit nur das Millionenprojekt Deep Blue von IBM in der Lage.

Beide Beispiele machen deutlich, dass durch Kooperation noch bestehende Defizite von Hard- und Software kompensiert werden können. Ein Team aus Mensch und Computer kann sowohl dem Menschen als auch dem Computer allein etwas voraus sein.

Genetische Algorithmen

Genetische Algorithmen (GA, siehe z.B. [Gol89]) sind nur ein Vertreter zahlreicher algorithmischer Varianten, die normalerweise unter dem Begriff „Evolutionary Computation“ zusammengefasst werden. Die Gemeinsamkeit aller Ansätze ist die Lösung von Problemen durch Programme, die sich der Prinzipien der Evolution bedienen. Die Arbeitsweise eines GA ist problemunabhängig und folgt einem festen Schema:

1. Zufälliges Erzeugen einer Reihe potentieller Problemlösungen („Initialpopulation“)
2. Bewerten dieser Lösungs-Kandidaten mit einer Bewertungsfunktion
3. Bilden neuer Kandidaten aus den bestehenden unter Bevorzugung der am besten bewerteten
4. Bilden einer neuen Population durch Selektion aus alten und neuen Kandidaten
5. Falls Abbruchkriterium noch nicht erreicht, bei 2. Fortfahren

Zentrales Erfordernis eines GA ist also eine Bewertungsfunktion, die es erlaubt, die Güte von Lösungs-Kandidaten zu ermitteln. Diese Funktion wird vom GA je nach Anforderung maximiert oder minimiert. Beim bereits zuvor geschilderten TSP würde die Bewertungsfunktion zum Beispiel einfach die Länge einer vorgeschlagenen Streckenführung ausrechnen.

Beim Bilden neuer Lösungen wird die Nähe zur Evolution deutlich: Hier werden Funktionen eingesetzt, die zum Beispiel Mutationen und Rekombinationen nachbilden. Während die Mutation als Veränderung einer Lösung – oder im GA-Sprachgebrauch eines Individuums – nur lokale Wirkung zeigt, werden bei der Rekombination wie bei einer Kreuzung die Eigenschaften von zwei oder mehr Individuen zu Bildung neuer herangezogen. Auch die im nächsten Schritt folgende Selektion bildet in stark vereinfachter Form die biologische Selektion nach, d.h. im Sinne der Bewertungsfunktion bessere Lösungen bleiben erhalten,

während schlechte verschwinden. In Analogie zur Natur werden die Schritte 2. bis 5. Als eine Generation des GA bezeichnet. So einfach das Vorgehen auf den ersten Blick erscheint, so beeindruckend sind die damit erzielten Ergebnisse (siehe z.B. [Gol89]).

Interaktive GA: Symbiosen zwischen Mensch und GA

Erste Versuche, GA um Interaktion zu erweitern, konzentrierten sich auf eine Bewertung durch den Menschen. Dieser Ansatz ist naheliegend, da es bei vielen Problemen schwer ist, eine exakte Bewertungsfunktion anzugeben. Ein erstes Beispiel [Smi91] war die interaktive Generierung von geometrischen Kurven, die einem Käfer ähnlich sehen sollten. Dem Bediener wurden jeweils sechzehn Vorschläge gemacht, unter denen er diejenigen Kurven auswählte, die einem Käfer am meisten ähnelten. Aus diesen wurde dann die nächste Generation erzeugt. Tatsächlich ähnelten die entstehenden Kurven immer mehr Käfern.

Das Problem dieses interaktiven GA (IGA) wurde aber deutlich, sobald der Bediener „falsche" Individuen – also solche, die nicht wie Käfer aussahen – auswählte. In diesem Falle wurden auch keine vernünftigen Ergebnisse erreicht. Außerdem funktioniert das gesamte Verfahren nur, wenn der Bediener tatsächlich eine Auswahl trifft. Tut er das nicht, so steht der gesamte Vorgang still. Das Problem der „Sabotage" des Verfahrens tritt bei dieser einfachen Aufgabenstellung sicher überwiegend bewusst durch den Bediener auf.

Bei komplizierteren Problemen ist es aber möglich, dass ein Bediener nach bestem Wissen kooperiert und trotzdem nur falsche Angaben macht. Ein Grund kann die ungenügende Vertrautheit mit dem Problem sein. Der Einsatz des IGA, der stark einer Symbiose zwischen Bediener und Programm gleicht, ist daher bei technischen Problemstellungen, wo objektive Maßstäbe gelten, wenig sinnvoll.

Das Vorgehen hat aber in vielen Bereichen durchaus seine Berechtigung. So wurde der beschriebene Ansatz zur sogenannten „Interactive Evolution" weiterentwickelt (siehe z.B. [GB95]). Diese Verfahren wird im Design-Bereich erfolgreich eingesetzt.

Der kommunizierende GA: Ein kooperationsfähiger GA

Will man den technisch orientierten Sektor bedienen, so muss ein anderer Ansatz zur Interaktion mit GA gefunden werden. In seiner vorgestellten Funktionsweise ist ein GA nicht zur Kooperation geeignet. Vielmehr wird er typischerweise mit bestimmten Parametern gestartet und läuft dann ohne irgendeine Interaktion ab, bis sein Abbruchkriterium (z.B. das Erreichen einer gewissen Näherung der Lösung) erreicht ist. Der Bediener kann höchstens einen vorzeitigen Abbruch vornehmen.

Um Kooperation zu ermöglichen muss der GA um Fähigkeiten zur Kommunikation erweitert werden. Im wesentlichen muss er in die Lage versetzt werden selbst Informationen auszugeben und andererseits Informationen von außen aufzunehmen. Die Ausgabe ist dabei unkritisch, da dies ohne Veränderungen am grundsätzlichen Ablauf leicht möglich ist. Zum Beispiel könnte man bei 2. die jeweils aktuellen Individuen ausgeben. Tatsächlich bietet ein großer Teil der GA-Implementierungen diese Möglichkeit an.

Bei der Eingabe ist nun aber eine Einflussnahme auf den Ablauf des GA unvermeidlich. Es ist denkbar, dass dadurch einige Eigenschaften des GA verändert werden. Besonders kritisch ist

hierbei die in [Rud94] gezeigte Konvergenzgarantie. Dort wurde gezeigt, dass sobald sichergestellt ist, dass durch Mutation jede Lösung erreichbar ist und das jeweils bisher beste Individuum in die Folgegeneration übernommen wird ein GA sicher gegen ein globales Optimum konvergiert. Obgleich die praktische Bedeutung nicht all zu hoch ist, wäre der Verlust dieser Eigenschaft ein hoher Preis für die Fähigkeit zur Kommunikation.

Der kommunizierende GA (KGA) erweitert daher zur Ausgabe interner und Aufnahme externer Informationen den zuvor genannten Ablauf folgendermaßen:

1. Zufälliges Erzeugen einer Reihe potentieller Problemlösungen („Initialpopulation“)
2. Bewerten dieser Lösungs-Kandidaten mit einer Bewertungsfunktion *und Ausgabe*
3. Bilden neuer Kandidaten aus den bestehenden unter Bevorzugung der am besten bewerteten
4. *Aufnahme eines extern vorgeschlagenen Kandidaten, falls ein solcher vorliegt*
5. Bilden einer neuen Population durch Selektion aus alten und neuen Kandidaten
6. Falls Abbruchkriterium noch nicht erreicht, bei 2. Fortfahren

Es lässt sich zeigen, dass hierbei die Konvergenzgarantie unter gleichen Bedingungen erhalten bleibt (siehe [Sch99a], S. 72 ff). Wesentlich hierfür ist, dass Mutation und Selektion nicht verändert werden. Somit kann der GA jetzt neben solchen Individuen, die er mittels genetischer Operationen erzeugt hat auch solche nutzen, die auf irgendeine Art und Weise extern entstanden sind. Denkbare Möglichkeiten sind der Vorschlag eines menschlichen Bedieners oder ein anderes Programm. Die gewählte Beschränkung auf einen Vorschlag pro Generation ist nicht zwingend, erleichtert aber die oben genannten technischen Beweise.

Anwendungsszenarien

Wie sieht nun die Kooperation mit dem KGA aus? In der Regel wird das Programm die bisher ermittelte beste Lösung in einer für den Bediener informativen Form darstellen und diesen so mit Informationen über seinen Fortschritt versorgen. Gleichzeitig ist eine Möglichkeit anzubieten, in ähnlicher Weise eigene Vorschläge zu erarbeiten und diese dem KGA übermitteln zu können. Da die Interpretation eines Individuums für die reale Welt auch als der „Phänotyp“ dieses Individuums bezeichnet wird, spricht man von einem Phänotyp-Editor. Wie dieser für ein Tourenplanungsproblem aussehen könnte zeigt die aus [Gra96] entnommene Abbildung 1.

Das Benutzerinterface nimmt bei der Kooperation eine zentrale Rolle ein, da es die völlig unterschiedlichen Sichtweisen von Software und Bediener in geeigneter Weise zusammenführen muss. Diese Aufgabe kann in Abhängigkeit vom jeweiligen Problem auch deutlich schwieriger sein als im zuvor gezeigten Fall, der eine naheliegende graphische Darstellung benutzt.

Die mit der Ermöglichung der Kooperation verbundene Hoffnung ist es, dass gute Vorschläge vom KGA aufgenommen werden und seine weitere Arbeit positiv beeinflussen, während schlechte Vorschläge durch die Selektion verworfen werden und keinerlei bleibenden Einfluss haben. Ein Vorteil, der so erreich würde, ist die Überprüfung der Bedienervorschläge anhand der Bewertungsfunktion, die als das wesentliche Kriterium für die Güte eines Vorschlags angesehen wird.

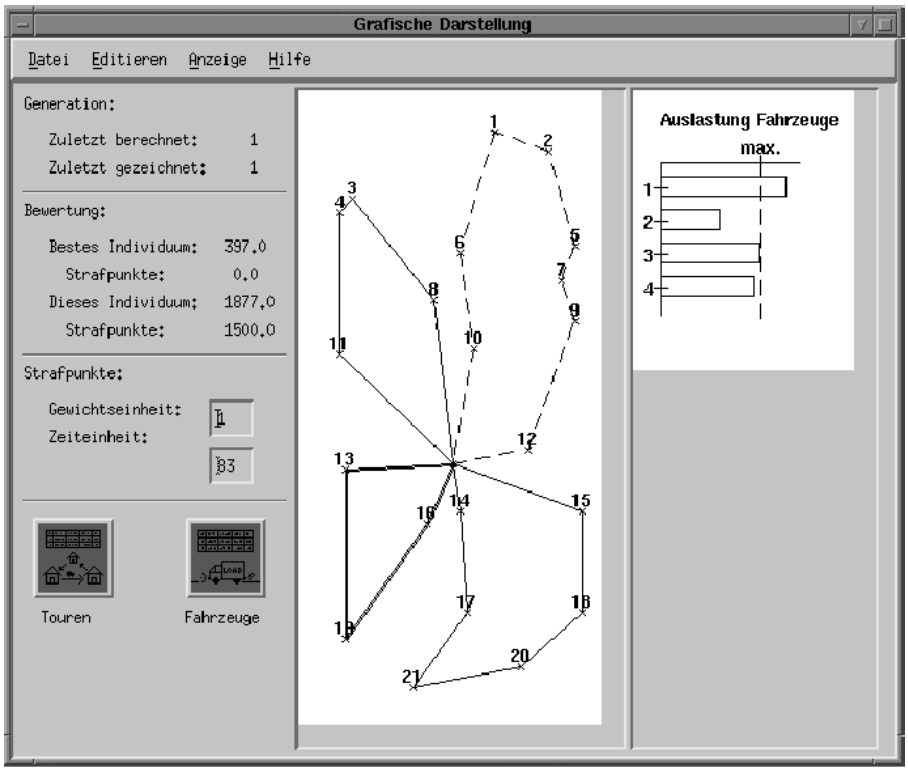

Abbildung 1: Beispiel eines Phänotyp-Editors für Tourenplanungsprobleme

Praktische Ergebnisse

Um zu einer stochastisch soliden Bewertung des Einflusses externer Vorschläge auf die zum Auffinden der optimalen Lösung benötigte Zeit zu kommen, wurde die Vorschläge eines menschlichen Bedieners durch ein Programm simuliert. Dieses Programm war in der Lage, zu bestimmten Zeitpunkten sowohl zuvor genau festgelegte Vorschläge als auch solche mit einer vorgegebenen Bewertung oder völlig zufällige zu machen. Die Erprobung erfolgte an einer Reihe von Problemen, wobei das erwähnte TSP und das Counting Ones Problem näher untersucht wurden. Bei letzterem wird ein als Bit-String repräsentiertes Individuum durch die Anzahl der mit einem vorgegebenen Muster übereinstimmenden Bitstellen bewertet. Diese ist zu maximieren, d.h. eine Übereinstimmung mit dem vorgegebenen Muster zu erreichen.

Abbildung 2 zeigt, wie sich Vorschläge verschiedener Qualität auf zur Ermittlung des Optimums benötigte Zeit auswirken. Es wird deutlich, dass genau der erhoffte Effekt eintritt: gute Vorschläge beschleunigen die Suche nach einer Lösung während schlechte keinen feststellbaren Einfluss haben. Diese Erkenntnis ist für den Bediener sehr beruhigend, denn er muss nicht damit rechnen, durch spontane Vorschläge das Ergebnis zu verschlechtern. Er kann sich vielmehr ganz von seiner Intuition und Erfahrung leiten lassen.

Implementierungen des KGA sind u.a. in der Werkstoffkunde und der Epidemiologie eingesetzt worden. In diesen beiden Bereichen ging es darum, das momentane Wissen, das als individuelle Erfahrung einzelner Personen algorithmisch nicht nachbildbar ist, mit einer systematischen Suchmethode zu kombinieren. In beiden Fällen wurde dabei erstmalig unter Zuhilfenahme eines Computers sinnvolle Ergebnisse erzielt. Details über diese Projekte sind in [ARP96] und [Sch99b] zu finden.

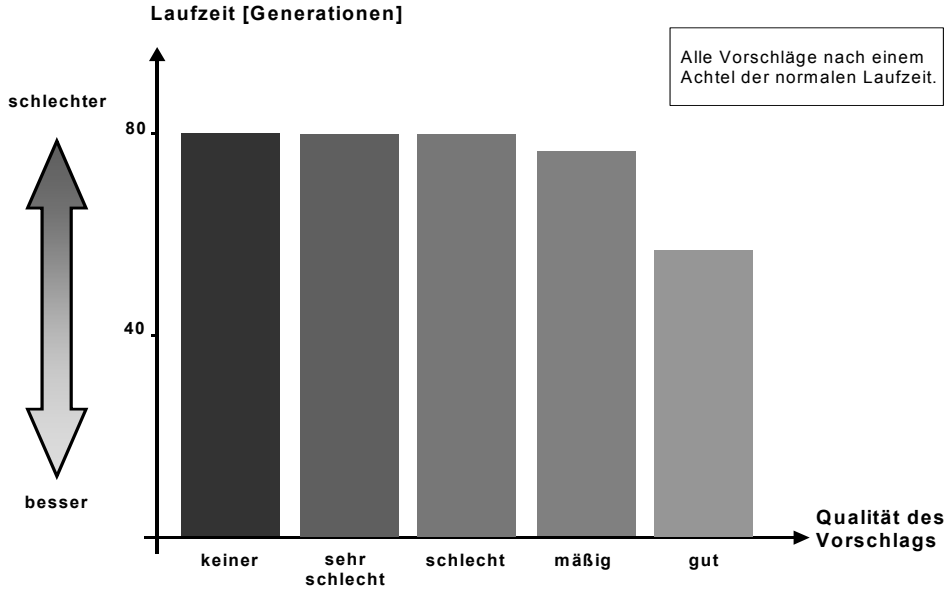

Abbildung 2: Laufzeitverbesserungen durch Vorschläge verschiedener Qualität

Ausblick

Genetische Algorithmen stellen eine interessante Basis zur Entwicklung kooperierender Software dar. Ihr großer Vorzug ist eine hohe Robustheit gegen – gewollt oder ungewollt – schlechte Vorschläge. Da sie auch für die Interaktion mit anderen Programmen offen sind, kann man sie als zentrale Instanz kooperierender Problemlösungs-Systeme einsetzen. Der praktische Einsatz kommunizierender Genetischer Algorithmen zeigt über die verkürzten Optimierungszeiten und verbesserten Ergebnisse hinaus positive Effekte, wie etwa eine signifikante Verminderung der bei klassischen GA beobachteten Schwankungen der Ergebnisse. Ein Einsatz in anderen Bereichen erscheint durchaus vielversprechend.

Literatur

[Alt98] I. Althöfer: 13 Jahre 3-Hirn, Selbstverlag, Jena 1998

[GB95] J. Graf, W. Banzhaf: Interactive Evolution for simulated Natural Evolution, in: M. Alliot et al. (Eds.), Artificial Evolution (LNCS 1063), Springer 1995

[Gol89] D. Goldberg: Genetic Algorithms in Search, Optimisation, and Machine Learning, Addison Wesley 1989

[Gra96] A. Grau: Einsatz hybrider Techniken in Genetischen Algorithmen am Beispiel von Tourenplanungen, Diplomarbeit, Informatik II, Universität Würzburg 1996

[KFM71] P. Krolak, W. Felts, G. Marble: A Man-Machine Approach Toward Solving the Travelling Salesman Problem, in: Comm. of the ACM (14) 5, 1971, S. 327-334

[Sch99a] J. Schoof: Diagnostic Support in Allergology Using Interactive Learning Clasifier Systems, in: W. Gaul, H. Locarek-Junge (Eds.), Classification in the Information Age (Proc. GfKl Conf. Dresden 1998), Springer 1999, S. 588-595

[Sch99b] J. Schoof: Kooperative Optimierung mit kommunizierenden Genetischen Algorithmen, Tectum Verlag 1999

[Smi91] J.R. Smith: Designing Biomorphs with an Interactive Genetic Algorithm, in: R.K. Belew, L.B. Booker (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, 1991, S. 535-538