Jena Research Papers in Business and Economics

A heuristic solution framework for the resource constrained multi-project scheduling problem with sequence-dependent transfer times

Doreen Krüger, Armin Scholl

16/2007

Jenaer Schriften zur Wirtschaftswissenschaft

Working and Discussion Paper Series
School of Economics and Business Administration
Friedrich-Schiller-University Jena

ISSN 1864-3108

Publisher:

Wirtschaftswissenschaftliche Fakultät
Friedrich-Schiller-Universität Jena
Carl-Zeiß-Str. 3, D-07743 Jena

www.jbe.uni-jena.de

Editor:

Prof. Dr. Hans-Walter Lorenz
h.w.lorenz@wiwi.uni-jena.de

Prof. Dr. Armin Scholl
armin.scholl@wiwi.uni-jena.de

www.jbe.uni-jena.de

A heuristic solution framework for the resource constrained multi-project scheduling problem with sequence-dependent transfer times

Doreen Krüger, Armin Scholl

Friedrich-Schiller-Universität Jena
Fakultät für Wirtschaftswissenschaften
Lehrstuhl für Betriebswirtschaftliche Entscheidungsanalyse
Carl-Zeiß-Straße 3, D-07743 Jena

e-Mail: {d.krueger, a.scholl}@wiwi.uni-jena.de

Abstract

We consider the problem of scheduling multiple projects subject to joint resource constraints. All approaches proposed in the literature so far are based on the assumption that resources can be transferred from one project to the other without any expense in time or cost. In many real-world settings this assumption is not realistic. For example, cranes have to be transported to another location and reinstalled there. In order to consider this additional aspect, we generalise the multi-project scheduling problem by additionally including transfer times which represent transportation, installation, adjustment, (re-) learning and other setup activities necessary when a resource is removed from one project and reassigned to another (or from one job to another within the same project).

In this paper, we define the modified multi-project scheduling problem with transfer times (called RCMPSPTT), formulate it as an integer linear programme, propose heuristic solution procedures and present results of comprehensive computational experiments.

Keywords:

project scheduling – combinatorial optimisation – mathematical model – transfer times

1 Introduction

1.1 Multi-project scheduling

Project scheduling has been playing a vital role in scheduling literature for some decades now. The common *resource constrained project scheduling problem (RCPSP)* has been studied extensively. However, single-project environments are rarely encountered in business today. Usually, companies run more than one project simultaneously. According to Payne (1995) up to 90 % of all projects (measured by their value) worldwide are executed in a multi-project environment. This finding goes along with Lova and Tormos (2001), who questioned 202 Spanish companies and found that 84 % of them run multiple projects in parallel. Nonetheless, single-project management concepts are by no means irrelevant as they provide a solid basis for multi-project concepts. For excellent introductions to resource constrained single-project scheduling see Kolisch (1995), Klein (2000), Demeulemeester and Herroelen (2002) and Neumann et al. (2003). For surveys of solution approaches see e.g. Brucker et al. (1999), Kolisch and Padman (2001) as well as Kolisch and Hartmann (1999, 2006).

The *resource constrained multi-project scheduling problem (RCMPSP)* as an extension of the RCPSP is considered as the simultaneous scheduling of two or more projects which demand the same scarce resources. Precedence constraints are defined only within projects. Projects are linked by the usage of the same restricted resources of the company. An objective function on company level often has to be considered although objectives of single projects may also be regarded (Kurtulus and Davis 1982, pp.161). The company objective as e.g. maximising profit is aimed at by managing the whole project portfolio or multi-project of the company by a resource manager, whereas project targets are set by single project managers. The latter aim to minimise project delay, project cost, etc.

Multi-project scheduling as considered here has been a research topic since the late 1960s. However, it has been studied not nearly as comprehensively as single-project scheduling. One may distinguish two main research fields in multi-project scheduling - the static and the dynamic project environment (Dumond and Mabert 1988, p. 102). The *static environment* view assumes a closed project portfolio. All projects of the company are summarised to a super-project (portfolio) and scheduled once. The multi-project is unequivocal and no rescheduling necessary. After the last project of a multi-project has been completed, a new multi-project may start. On the contrary, the *dynamic environment* view considers an open project portfolio. While scheduled projects are executed, new projects arrive to the system and have to be integrated because the portfolio is changing over time. Rescheduling of the system becomes necessary frequently.

Research mainly focuses on the static environment. Scheduling in such a static environment has been researched amongst others by Fendley (1968), who was the first discussing the modelling of a complete multi-project scheduling system and proposing methods for assigning due dates to incoming projects and priority rules for sequencing individual activities. Pritsker et al. (1969)

present a zero-one integer programme for the problem considering various possible constraints, e.g. job splitting or resource substitutability and objectives like, e.g., total throughput time, makespan or total lateness. Kurtulus and Davis (1982) introduce the single-project approach, whereas Kurtulus and Narula (1985) add the multi-project approach for the multi-project scheduling problem (see below). In both papers, special priority rule based solution procedures are developed and tested. Lawrence and Morton (1985) test resource price based priority rules to minimise weighted tardiness costs within a multi-project. Lova and Tormos (2001) analyse existing priority rules for the single- and multi-project approach. Moreover, they present new two-phase rules for the multi-project approach. Vercellis (1994) suggests a decomposition technique. However, these are only some examples for research on static multi-project scheduling, further references are, amongst others, Patterson (1973), Mohanty and Siddiq (1989), Wiley et al. (1998), and Lova et al. (2000).

Dynamic environments are researched by Dumond and Mabert (1988). Their study is based on priority rules for static environments. Due date assignment rules, which are tested by simulation, are added. Dumond (1992) as well as Dumond and Dumond (1993) extend the former study by introducing different resource availability levels. Bock and Patterson (1990) allow resource pre-emption in the multi-project. Yang and Sum (1993, 1997) give attention to dynamic project environments by establishing a dual-level management structure for assigning resources to projects on a higher level and operative project scheduling on a lower level. Ash and Smith-Daniels (1999) put emphasis on the learning, forgetting and relearning cycle in dynamic multi-project environments while Anavi-Isakow and Golany (2003) apply queuing theory and adapt the production management concept of CONWIP (constant work in progress) to the multi-project environment.

Another approach to multi-project scheduling to be mentioned is hierarchical planning. This aspect is not to be discussed here because it leads to a completely different way of scheduling. We refer to Hans et al. (2003) and De Boer (1998) for further reading.

The following of this paper extends the concept of static project environments. That is why a short summary of this concept is given in the following. As already mentioned, in literature a single- and a multi-project approach to static multi-project scheduling are classified. On the one hand, the single-project approach merges all projects of the multi-project to an artificial super-project with a dummy start and end activity for time and resource scheduling (Kurtulus and Davis 1982, p. 162). Hence, in this case multi-project scheduling is identical to single-project scheduling of large projects. In comparison to the RCPSP, the multi-project duration, which is given by the realised finishing time of the last activity of the latest project, is minimised. On the other hand, the multi-project approach keeps the projects separate for (resource unconstrained) time scheduling. Consequently, critical path analysis identifies a critical path for each project. Afterwards, the projects are merged for resource scheduling. The objective of this approach is to minimise the mean project delay, which is often measured by the deviation from the critical path bound for each project if no due dates are given.

The only scheduling procedures applied to both approaches so far are heuristics using priority rules. According to Lova and Tormos (2001), a multi-project consists of about 120 to 480 activities. Exact procedures cannot handle problems of this size so far since already the basic RCPSP is NP-hard. Hence, heuristics are the only realistic possibility for solving the RCMPSP.

Even though the static project environment is emphasised in literature, it still fails to pay attention to some important aspects of multi-projects. One of the aspects is resource transfers within a multi-project. Virtually all papers neglect resource transfers between projects or assume them having zero duration. Resources may be shifted between projects unlimitedly, without any restrictions and any control. Time delays and costs caused by these transfers are not taken into account. In reality, transfers may take time

- when a resource is physically moved from one location to another, e.g. heavy machines, specialists that fly around the world, and/or
- when a resource has to be adjusted in respect to content, e.g. setup times for machines, human resources that have to get acquainted with new projects. Especially for human resources the learning, forgetting and relearning life-cycle plays a vital role in transfer time considerations.

This problem is being tackled in our paper to resolve this criticism of the state-of-the-art scheduling approaches. Resource transfers are going to be considered as an additional aspect to the RCMPSP in a static environment.

1.2 Setup times in project scheduling

First and foremost, a review of papers that already considered transfer or setup times in the single-project case and first attempts to consider resource transfers among projects in a multi-project case is presented.

Setup times which are a variant of transfer times have already been investigated in production scheduling and lot sizing extensively. In single-project scheduling some research on the field of setup times has been done. Kolisch (1995) develops a zero-one integer programme for the RCPSP with sequence-independent setup times. Setup times are incorporated by introducing two modes for activity execution. If a setup is needed the setup time is integrated into the task duration (mode 2), otherwise the activity is scheduled in mode 1 with its normal duration. It is assumed that the capacity of each resource is one unit. Moreover, only one of the resources that are required by an activity can demand a setup. Debels and Vanhoucke (2006) consider setup times in the pre-emptive resource-constrained project scheduling. They assume that a setup is always needed when a pre-empted activity is continued. The setup times are considered to be sequence-independent and added to the duration of the pre-empted activities. Neumann et al. (2003, ch. 2.14) extend the approach of Trautmann (2001) and present the RCPSP with time windows and sequence-dependent changeover times. Unlike Trautmann, who assumes that resource requirements can only take values of 0 or 1, Neumann et al. allow for arbitrary resource capacities and resource requirements of activities. They split the problem into two interdependent subproblems. In a first step, a precedence-feasible schedule is determined. In a second step it is checked whether this schedule is changeover feasible. Changeover feasibility is given when all resource constraints are met while setup times are considered. Mika et al. (2006) give a more extensive literature review on setup times in project scheduling. They classify setups into several categories but do not focus on resources which have to changeover to other activities or projects. Instead, they classify setups from an activity perspective.

In the context of multi-project scheduling, setup times or transfer times are rarely encountered in literature up to now. Yang and Sum (1993, 1997) consider resource transfer times in a dynamic multi-project environment with dual-level management structure. A central resource pool manager assigns resources to projects, whereas a project manger schedules activities within his project using the allocated resources. Resource changeovers can only be handled via a central pool. No transfer times from the project to the pool occur, while transfers from the pool to any project are assumed to take a constant time for all resource types, i.e., the times are sequence- as well as resource type-independent. Neumann (2003) points out that the problem formulation presented in Neumann et al. (2003) can be applied to multi-project scheduling with distributed locations, too.

In the following, we consider general resource transfers with sequence- and resource-dependent transfer times with emphasis on the resources which changeover between activities in the same projects or between activities in different projects. In Section 2, the new resource constrained multi-project scheduling problem with transfer times (RCMPSPTT) is defined. Section 3 presents a mathematical model for this novel optimisation problem. In Section 4 and 5, heuristic solution frameworks based on priority rules are described. Computational experiments and their result are presented in Section 6. Conclusions are given in Section 7.

2 Problem description

In the new problem RCMPSPTT, we consider a multi-project made up of a set of single projects $P = \{1 \ldots m\}$. Each project $p \in P$ consists of a set J_p of real jobs as well as a dummy start activity s_p and a dummy end activity e_p. The multi-project comprises all project activities as well as a global super source s_0 and sink e_0, which are all summarised in set $J' = \{1 \ldots n\}$.

The activities and precedence constraints of the multi-project can be represented by a finish-to-start activity-on-node network without time lags. A set of direct predecessors A_j is given for each job $j \in J'$. The global source s_0 has no predecessor, whereas each local project source is successor of s_0 and the global sink e_0 succeeds each local project sink. Except for this global linkage of all projects, precedence constraints exist only between jobs of the same project but not between different projects. To ease presentation, we (re-)number all jobs $j \in J'$ topologically, i.e. $j > i$ for each pair $j \in J'$ and $i \in A_j$. Thus, job 1 is the global source node s_0 and job n the global sink node e_0.

An activity j with integer duration d_j must not be interrupted once it has been started. Several types of renewable resources $r \in R$ with constant capacity a_r are available for project execution in every period. Executing activity $j \in J'$ requires a constant integer number of resource units u_{jr} of resource type $r \in R$ per period. The required amount must be transferred from other activities to job j and, thus, often from other projects to the job that is to be executed. This transfer takes time Δ_{ijr} depending on the originating activity $i \in J'$ and receiving activity $j \in J' - \{i \cup A_i\}$ as well as the resource type $r \in R$ irrespective of the number of resource units transferred. Transfer times may occur within projects but also and, more importantly, among projects. All transfer times are assumed to fulfil the triangular inequality. Due to the static view, transfers from the global source and to the global sink are assumed to consume no time. However, in case of physical resources like cars and machines, these nodes can be considered to represent a global resource pool where all (physical) resources are stored before the multi-project starts and to which all resources must return after having terminated the multi-project. Hence, non-zero transfer times from and to the pool might occur as well. However, transfers which flow from the global source s_0 to the global sink e_0 directly take no time because these flows absorb redundant resource units and are not executed in reality.

Dummy activities are characterised by a duration and resource usage of zero for each resource type with exception of the global source s_0 and sink e_0. These two dummy jobs take no time but their resource usage is defined by $u_{s_0 r} = u_{e_0 r} = a_r$ for $r \in R$ because the global source needs to provide all resources to the multi-project while the global sink collects them.

The RCMPSPTT is constituted by determining finishing times F_j for all activities $j \in J'$ and corresponding resource transfers x_{ijr} such that a multitude of constraints is met. All precedence and resource constraints must be observed while sequence- and resource-type-dependent transfer times for resources changing to other activities are considered.

In the single-project approach, the multi-project duration MPD, i.e., the finishing time F_{e_0} of the global sink e_0, or its relative increase MPDI is minimised. MPDI is measured as the relative deviation of MPD from the time of the multi-project's critical path from s_0 to e_0. The critical path time, which represents a simple lower bound LB1 on the multi-project duration, is used to evaluate the delay of the multi-project caused by resource restrictions.

In the multi-project approach, the objective consists in minimising mean project delay MD defined as average relative deviation of the realised finishing time F_{e_p} from the critical path time $LB1_p$ over all projects $p \in P$. $LB1_p$ is used as a surrogate for a due date of each project to evaluate their delays, because due dates are not pre-determined in this problem version.

Both objective functions, one for each scheduling approach, are commonly used in existing literature on the basic multi-project scheduling problem (Pritsker et al. 1969, Kurtulus and Davis 1982, Kurtulus and Narula 1985 as well as Lova and Tormos 2001). However, these objectives for scheduling multiple projects should not be taken for granted. This paper uses them for a first analysis of the new problem but will point out other possibilities in the conclusion.

3 Mathematical model

We develop a mixed-integer linear programme for RCMPSPTT, which is based on combining the traditional model for RCPSP by Pritsker et al. (1969) with a network flow based formulation of the single-project scheduling problem with sequence-dependent setup times as proposed by Neumann et al. (2003, ch. 2.14). The model includes the typical assumptions of the RCPSP and extends it by considering transfer times between activities of different or the same project(s).

The following *parameters* are used:

P	set of projects; index: p
J_p	set of real activities of project $p \in P$; index: j
J	set of real activities within all projects; $J = \bigcup_{p \in P} J_p$
J'_p	activities of project $p \in P$ including dummy start activity s_p and dummy end activity e_p; $J'_p = J_p \cup \{s_p, e_p\}$
J'	set of activities within all projects plus single global source s_0 and global sink e_0, i.e., $J' = \bigcup_{p \in P} J'_p \cup \{s_0, e_0\}$
n	number of jobs; $n = \vert J' \vert$
d_j	duration of activity $j \in J'$ (with $d_{s_p} = d_{e_p} = 0$ for $p \in P \cup \{0\}$)
T	upper bound on the project duration
t	index for periods; t = 0,...,T
A_j	set of direct predecessors of job $j \in J'$; $A_{s_0} = \{\ \}$, $A_{e_0} = \bigcup_{p \in P} \{e_p\}$, $A_{s_p} = \{s_0\}$ for $p \in P$; for $j \in J_p$ with $p \in P$: $A_j \subseteq J_p - \{j\}$ (precedence constraints only within a project)
A_j^*	set of direct and indirect predecessors of job $j \in J'$ (predecessors in the transitive closure of the graph)
S_j	set of direct successors of job $j \in J'$; $S_{e_0} = \{\ \}$, $S_{s_0} = \bigcup_{p \in P} \{s_p\}$, $S_{e_p} = \{e_0\}$ for $p \in P$; for $j \in J_p$ with $p \in P$: $S_j = \{i \mid i \in J_p \cup \{e_p\} \wedge j \in A_i\}$
S_j^*	set of direct and indirect successors of job $j \in J'$
EF_j	earliest finishing time of activity $j \in J'$; $EF_{s_p} = 0$ for $p \in P \cup \{0\}$ (forwards path)
LF_j	latest finishing time of activity $j \in J'$ (backwards path)
TI_j	time window for finishing activity $j \in J'$; $TI_j = [EF_j, LF_j]$
LB1	lower bound of multi-project duration (critical path time)
$LB1_p$	lower bound of project $p \in P$ (individual critical path time of project p)
R	set of resources; index: r
a_r	number of units of resource $r \in R$ available per period
u_{jr}	number of units of resource $r \in R$ required for performing activity $j \in J'$ per period
Jr_j	set of real activities to which resources might be transferred after having performed $j \in J$; $Jr_j := J - \{j\} - A_j^*$
Δ_{ijr}	time for transferring units of resource $r \in R$ from task $i \in J \cup \{s_0\}$ to task $j \in Jr_i \cup \{e_0\}$, $\Delta_{s_0 e_0 r} = 0 \ \forall r \in R$

We define the following *variables*:

$$f_{jt} = \begin{cases} 1 & \text{if activity j is terminated at the end of period t} \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } j \in J' \text{ and } t \in TI_j$$

F_j realised finishing time of activity $j \in J'$

$$z_{ijr} = \begin{cases} 1 & \text{if resource r is transferred from activity i to j} \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } r \in R,\ i \in J \cup \{s_0\},\ j \in Jr_i \cup \{e_0\}$$

x_{ijr} number of units of resource $r \in R$ transferred from activity $i \in J \cup \{s_0\}$ to activity $j \in Jr_i \cup \{e_0\}$

The *model* is given by the objective function (1) and the set of constraints (2) to (13).

$$\text{Minimise } \Phi(\mathbf{F},\mathbf{f},\mathbf{x},\mathbf{z}) \quad \text{s.t.} \tag{1}$$

(I) Time scheduling	$\sum_{t \in TI_j} f_{jt} = 1$	for all $j \in J'$	(2)
	$F_j = \sum_{t \in TI_j} t \cdot f_{jt}$	for all $j \in J'$	(3)
	$F_j - F_i \geq d_j$	for $j \in J'$ and $i \in A_j$	(4)
Linking (I) and (II)	$F_i + \Delta_{ijr} + d_j \leq F_j + T \cdot (1 - z_{ijr})$	for $i \in J \cup \{s_0\}$, $j \in Jr_i \cup \{e_0\}$ and $r \in R$	(5)
	$x_{ijr} \leq z_{ijr} \cdot \min\{u_{ir};u_{jr}\}$	for $i \in J \cup \{s_0\}$, $j \in Jr_i \cup \{e_0\}$ and $r \in R$	(6)
	$z_{ijr} \leq x_{ijr}$	for $i \in J \cup \{s_0\}$, $j \in Jr_i \cup \{e_0\}$ and $r \in R$	(7)
(II) Resource scheduling	$\sum_{h \in J - \{i\} - S_i^*} x_{hir} = u_{ir}$	for $i \in J \cup \{e_0\}$ and $r \in R$	(8)
	$\sum_{j \in Jr_i \cup \{e_0\}} x_{ijr} = u_{ir}$	for $i \in J \cup \{s_0\}$ and $r \in R$	(9)
Variables	$f_{jt} \in \{0,1\}$	for $j \in J'$ and $t \in TI_j$	(10)
	$F_j \geq 0$	for $j \in J'$	(11)
	$z_{ijr} \in \{0,1\}$	for $r \in R$, $i \in J$, $j \in Jr_i$	(12)
	$x_{ijr} \geq 0$	for $r \in R$, $i \in J \cup \{s_0\}$, $j \in Jr_i \cup \{e_0\}$	(13)

The objective function (1) depends on the approach applied. In the single-project perspective, the multi-project duration increase MPDI is minimised, which is equivalent to minimising the multi-project duration MPD:

$$\text{Minimise } \text{MPDI}(\mathbf{F},\mathbf{f},\mathbf{x},\mathbf{z}) = \frac{F_{e_0} - LB1}{LB1} \cdot 100\% \tag{14}$$

In the multi-project approach, the mean project delay MD is used as performance measure:

$$\text{Minimise } \text{MD}(\mathbf{F},\mathbf{f},\mathbf{x},\mathbf{z}) = \frac{1}{|P|} \cdot \sum_{p \in P} (F_{e_p} - LB1_p) \tag{15}$$

Time scheduling constraints (2) - (4) are well-known from the RCPSP formulation. Constraints (8) - (9) represent resource flows in the multi-project. Inequalities (5) to (7) link the time scheduling and the resource flow parts of the model. Moreover, they handle resource transfers. Eventually, (10) - (13) define decision variables of the problem.

Constraints (2) ensure that each job is terminated in a unique period. Equations (3) transform the binary decision variables f_{jt} into realised finishing times for model reduction reasons. Inequations (4) represent the explicit precedence relationships within a project while (5) express implicit precedence relationships, which are necessary for possible resource transfers.

From a time feasibility point of view, a transfer of resource r from activity i to j ($i \rightarrow j$) may only occur ($z_{ijr} = 1$) if activity i ends before activity j is started and the time lag between both activities is larger than or equal to the required transfer time for resource type r (constraints (5)). From the resource feasibility point of view, a resource transfer of resource r from i to j can only take place if both activities have a positive resource demand. The amount of resource type r, which is actually transferred, thus, depends on whether a time-feasible and resource-feasible transfer is possible ($z_{ijr} = 1$ and $\min\{u_{ir}, u_{jr}\} > 0$) as expressed in constraints (6). Yet, it is limited by the demand of activity i or j, whatever is lower. Constraints (7) ensure that the indicator variable z_{ijr} is set only to 1 if a transfer really takes place ($x_{ijr}>0$). Since triangular inequations are true for all transfer times, it is not useful to transfer more resources than needed. While equations (8) satisfy that the incoming flows of resource r to activity i sum up to its resource demand u_{ir}, (9) guarantee that this very quantity of resource r flows out of activity i. The incoming flow of resource r can only be provided by activities preceding i directly or indirectly (including s_0), whereas the outgoing flow can only be directed to succeeding activities (including e_0). Moreover, constraint (9) for $i = s_0$ guarantees that the global source provides resource capacity a_r of each resource type r to the multi-project, whereas constraint (8) for $i = e_0$ ensures that all resources are collected at the end of the project.

Since RCMPSPTT is a generalisation of RCPSP, the problem is NP hard. As expected, preliminary experiments indicate that modelling and solving the problem with modern optimisation software like XPress-MP or CPlex is not sufficient to solve problems of realistic size. Therefore, we develop heuristic solution frameworks which combine elements for time and resource scheduling. Basics for time scheduling are the two common schedule generation schemes and priority rules well-known for RCPSP and RCMPSP (cf. e. g. Kurtulus and Davis 1982, Klein 2000, Lova and Tormos 2001) which have to be adapted adequately. For resource and transfer scheduling new scheduling rules are presented. In Section 4, we define a solution framework based on the parallel scheduling scheme. In Section 5, the framework is adapted to the serial scheme.

4 Parallel scheduling framework

4.1 Algorithm

The time oriented parallel scheduling scheme creates a feasible schedule by considering increasing decision times t. At each decision time t as many activities as possible are started such that precedence and resource feasibility is given (Lova and Tormos 2001, p. 267). The sequence in which jobs are considered is determined by a priority rule. This common scheduling scheme is adapted to the case of sequence- and resource type-dependent transfer times.

In order to ease presentation, the dummy source nodes s_p and end nodes e_p of all single projects are eliminated by directly linking the projects to the global source node s_0 and the global sink node e_0. Finally, the starting and finishing times of those nodes can simply be set as follows:

$$s_p = \min\{F_j - d_j \mid j \in J_p\}\,,\; e_p = \max\{F_j \mid j \in J_p\} \text{ for } p = 1,...,P$$

The algorithmic representation is based on that of Kolisch and Hartmann (1999). It uses the following additional parameters complementing the notation of Section 3:

J'	due to temporarily deleting the dummy sources and sinks of the projects, we get $J' = J \cup \{s_0, e_0\}$; $n = \lvert J' \rvert$
$\mathcal{C}$	set of completed jobs at scheduling time t ; $\mathcal{C} = \{j \in J' \mid F_j \le t\}$
$\mathcal{A}$	set of jobs active in period t; $\mathcal{A} = \{j \in J' \mid F_j - d_j \le t < F_j\}$
$\mathcal{S}$	set of jobs scheduled up to time t; $\mathcal{S} = \mathcal{A} \cup \mathcal{C}$
$\tilde{a}_r(t)$	available units of resource type $r \in R$ at scheduling time t with $\tilde{a}_r(t) = a_r - \sum_{j \in \mathcal{A}} u_{jr}$
$\mathcal{D}$	set of eligible jobs at scheduling time t with $\mathcal{D} = \{j \in J' - \mathcal{S} \mid A_j \subseteq \mathcal{C} \wedge u_{jk} \le \tilde{a}_r(t) \forall r \in R\}$
y_{jr}	number of units of resource r delivered to job j and still awaiting to be delivered to a another job
TJ^r	set of transfer-feasible activities able to deliver units of resource type r to job j up to time t; $TJ^r = \{\ i \in \mathcal{S} \mid F_i + \Delta_{ijr} \le t \ \wedge\ y_{ir} > 0\}$
s^r	total supply of resource type $r \in R$ by transfer-feasible jobs at time t ; $s^r = \sum_{i \in TJ^r} y_{ir}$

Algorithm 1 starts out with an initialisation of the problem instance. All quantities of resource transfers x_{ijr} and activity resource stocks y_{jr} are initialised to zero. The dummy global source $s_0 = 1$ is scheduled at time 0 and put into the set of completed activities. Afterwards, the modified parallel scheduling scheme is applied.

For each trial scheduling time t, as many eligible activities as possible are scheduled provided resource and precedence constraints including transfer times are not hurt. From the set $\mathcal{D}$ of eligible jobs, the jobs j to be examined are chosen one after another in a sequence defined by an appropriate *job rule* (see Section 4.2). For each considered j, the sets TJ^r, which contain all activities that can deliver at least one unit of resource type r to job j up to time t, are determined. If these jobs supply sufficient resource units of each resource type $r \in R$, job j is scheduled and delivering jobs i are chosen from TJ^r by a resource transfer rule until the demand of job j is satisfied (see Section 4.3). Simultaneously, transfer quantities x_{ijr} from i to j are calculated for each r. Should any determined set TJ^r offer insufficient resource amounts to the selected job j, this job is removed from the set $\mathcal{D}$ of eligible jobs. If there does not remain any job in the set of eligible activities $\mathcal{D}$ for which feasible resource transfers can be determined, scheduling time t is increased step by step until a non-empty set $\mathcal{D}$ is determined and at least one job j is contained for which all required resources can be provided in time. When all jobs have been scheduled, the algorithm stops.

4.2 Job rules

Job selection rules (job rules for short) are the original priority rules known for the RCPSP or the RCMPSP. A job rule assigns priority values to all jobs and orders them according to non-increasing or non-decreasing values. Within a scheduling scheme, the jobs are examined and, if possible, scheduled in this order. A plethora of such rules are known for the RCPSP. Overviews can be found in, e.g., Kolisch and Hartmann (1999), Kolisch and Padman (2001), Klein (2000, ch. 5.2+7.4).

```
Initialise  $t := 0$ ; $\mathcal{A} := \emptyset$ ; $\mathcal{S} := \mathcal{C} := \{1\}$ ; $F_1 := 0$ ; $\tilde{a}_r(t) := a_r$ for all $r \in R$ and $t = 0,\ldots,T$ ;
            $y_{jr} := 0$ for all $j \in J'$ and $r \in R$ ; $x_{ijr} := 0$ for all $i,j \in J'$ and $r \in R$
            $\mathcal{D} := \{j \in J' \mid A_j = \{1\}\}$ ;          /* set of eligible jobs at t=0
While $|\mathcal{S}| < n$ do                                 /* iterate on increasing t until all jobs are assigned
    While $\mathcal{D} \neq \emptyset$ do                             /* assign as many jobs as possible to current time t
        Select best job $j \in \mathcal{D}$ ;                         /* apply *job rule* (Section 4.2)
Resource scheduling
        For $r \in R$ do                                       /* for each resource type ...
            $TJ^r = \{ i \in \mathcal{S} \mid F_i + \Delta_{ijr} \leq t \wedge y_{ir} > 0\}$ ;      /* ... compute transfer-feasible set
            $s^r := \sum_{i \in TJ^r} y_{ir}$ ;                          /* ... compute total supply of resource r for j in t
        If ( $s^r \geq u_{jr}$ for all $r \in R$ ) do                  /* if supply is sufficient for all resources, then ...
            For $r \in R$ do                                   /* ... find delivering jobs for all r as follows:
                While $y_{jr} < u_{jr}$ do                        /* repeat until resource demand of j for r is satisfied
                    Select and remove best job $i \in TJ^r$ ;         /* apply *resource transfer rule* (Section 4.3)
                    $x_{ijr} := \min\{y_{ir}, u_{jr} - y_{jr}\}$ ;             /* compute transferable amount of resource r
                    $y_{jr} := y_{jr} + x_{ijr}$ ; $y_{ir} := y_{ir} - x_{ijr}$          /* update resource stocks
            $F_j := t + d_j$ ; $\mathcal{S} := \mathcal{S} \cup \{j\}$ ; $\mathcal{A} := \mathcal{A} \cup \{j\}$    /* ... schedule job j starting in t and finishing in $t + d_j$
            For $\tau \in [t, t + d_j[$ and $r \in R$ do
                $\tilde{a}_r(\tau) := \tilde{a}_r(\tau) - u_{jr}$ ;                   /* ... reduce available capacity of resources
        $\mathcal{D} := \mathcal{D} - \{j\}$ ;                                /* remove j from the set of (unexamined) eligible jobs
    $t := t+1$;                                          /* increase current scheduling time t
    $\mathcal{C}' = \{i \in \mathcal{A} \mid F_i \leq t\}$ ;                       /* set of tasks just completed in current t
    $\mathcal{A} := \mathcal{A} - \mathcal{C}'$ ; $\mathcal{C} := \mathcal{C} \cup \mathcal{C}'$ ;                 /* update set of active and completed jobs
    $\mathcal{D} := \{j \in J' - \mathcal{S} \mid A_j \subseteq \mathcal{C} \wedge u_{jk} \leq \tilde{a}_r(t) \forall r \in R\}$   /* compute set of eligible tasks for current t
```

Algorithm 1. Modified parallel scheduling scheme

	job rule (job j)	extremum	priority value π_j
multi-project approach	**minSASP**	$\min_j$	$\pi_j = LB1_{p(j)} + d_j$
	minSLK_MP	$\min_j$	$\pi_j = LF_j(CP_{p(j)}) - EF_j(CP_{p(j)})$ with $CP_{p(j)}$ as critical path of project p(j)
	minLFT_MP	$\min_j$	$\pi_j = LF_j(CP_{p(j)})$ with $CP_{p(j)}$ as critical path of project p(j)
	maxTWK(dyn)	$\max_j$	$\pi_j = d_j \cdot \sum_{r=1}^{\lvert R\rvert} u_{jr} + \sum_{k \in \mathcal{S} \cap J_{p(j)}} d_k \cdot \sum_{r=1}^{\lvert R\rvert} u_{kr}$
	maxRD_MP	$\max_j$	$\pi_j = d_j \cdot \sum_{r=1}^{\lvert R\rvert} u_{jr} + \sum_{k=1}^{\lvert J_{p(j)}\rvert} (d_k \cdot \sum_{r=1}^{\lvert R\rvert} u_{kr})$
single-project approach	**minSLK_SP**	$\min_j$	$\pi_j = LF_j(CP) - EF_j(CP)$ with CP as critical path of the super-project
	minSLK_SP(dyn)	$\min_j$	$\pi_j = LF_j(PS) - EF_j(PS)$ with PS as partial schedule of the super-project
	minLFT_SP	$\min_j$	$\pi_j = LF_j(CP)$ with CP as critical path of the super-project
	maxRD_SP	$\max_j$	$\pi_j = d_j \cdot \sum_{r=1}^{\lvert R\rvert} u_{jr}$
inde-pendent	**FCFS(dyn)**	$\min_j$	$\pi_j = \max\{EF_i \mid i \in A_j \cap \mathcal{S}\}$
	Random	$\min_j$	$\pi_j = \text{random}(\,)$

Table 1. Job rules for parallel and serial scheme

Yet, in the context of multi-project scheduling specialised job rules have been developed and applied both in the parallel and the serial scheme. The rules can be classified according to the approach they are designed for. There are special rules for the single- and the multi-project approach as well as rules that are independent of these approaches. The most common rules for the multi-project scheduling problem are summarised in Table 1 (see Lova and Tormos 2001, Pritsker et al. 1969, Kurtulus and Davis 1982, Kurtulus and Narula 1985). Column 1 classifies the presented rules regarding the approach they are used with. For the notation used see Section 3; additionally $p(j)$ indicates the project to which a job j belongs, i.e., $j \in J_{p(j)}$. Shortest activity from shortest project (minSASP), minimum slack (minSLK), minimum latest finishing time (minLFT), maximum total work content (maxTWK) as well as maximum resource demand (maxRD) are rules for the multi-project approach because they rely on information about each project within the multi-project (extension "MP"). Apart from minSASP and maxTWK, the rules can be used for the single-project approach as well (extension "SP"). They use information of the super (multi-)project for priority calculation. In this context, dynamic rules ("dyn") are possible, too. Priority values are updated in each iteration depending on the already scheduled activities (partial schedule PS) and their fixed starting and finishing times instead of being computed only once at the beginning of the whole procedure (static rules). The rules first come first served (FCFS) and Random are independent of the approach to multi-project scheduling.

4.3 Resource transfer rules

When a job j is selected to be scheduled at time t and all sets TJ^r supply sufficient resource units, it must be decided which of the jobs in each TJ^r is chosen to deliver resources to job j. This is obviously only necessary if there is a surplus of provided resource units.

To find the delivering jobs for each resource r required by job j, all jobs in TJ^r are sorted by a priority rule which we call *resource transfer ru*le (to be more specific, it is a transfer-from rule, cf. Section 5.2). Even if the same rule is applied to each r, different priority lists will result.

We suggest three rules, divided into time and resource oriented ones (Table 2). A time oriented rule is minTT, which selects a job i with lowest transfer time from i to job j first, regarding the considered resource type r. The rule minGAP prefers jobs with minimal gap between the earliest delivery time (= finishing time F_i + transfer time Δ_{ijr}*)* and the start time t of j. The resource oriented rule RS sorts the jobs of TJ^r according to their free resource stock after task execution, i.e., the amount of resource units that are not transferred to another job yet. minRS selects the job with minimum and maxRS with maximum stock first.

	transfer-from rule	**extre mum**	**priority value π'^r_i for $i \in TJ^r$**
time oriented	**minTT**	$\min_i$	$\pi'^r_i = \Delta_{ijr}$
	minGAP	$\min_i$	$\pi'^r_i = gap_{ijr} = t - (F_i + \Delta_{ijr})$
resource oriented	**minRS**	$\min_i$	$\pi'^r_i = y_{ir} = u_{ir} - \sum_{k \in \mathcal{S}} x_{ikr}$
	maxRS	$\max_i$	

Table 2. Resource transfer rules for parallel scheme

After the transfer-feasible activities have been sorted, the actually delivering tasks are selected. We propose two different ways, forward and backward selection.

Forward selection uses the original priority list. The activity with highest priority is chosen and delivers as many free resource units to job j as possible or necessary, whatever is the lower. Activities from the top of the list are selected until the resource demand of job j is satisfied. *Backward selection* sums the provided free resources along the priority list up, starting with the highest priority activity. The first activity in the list which causes equality or surplus of resource units when added is used as the starting point for backward transfer scheduling. From this starting point on, the resource demand of job j is satisfied by the activities along the list downwards to the task with the highest priority. In each step as many free resources are provided to job j as required and available, whatever comes first.

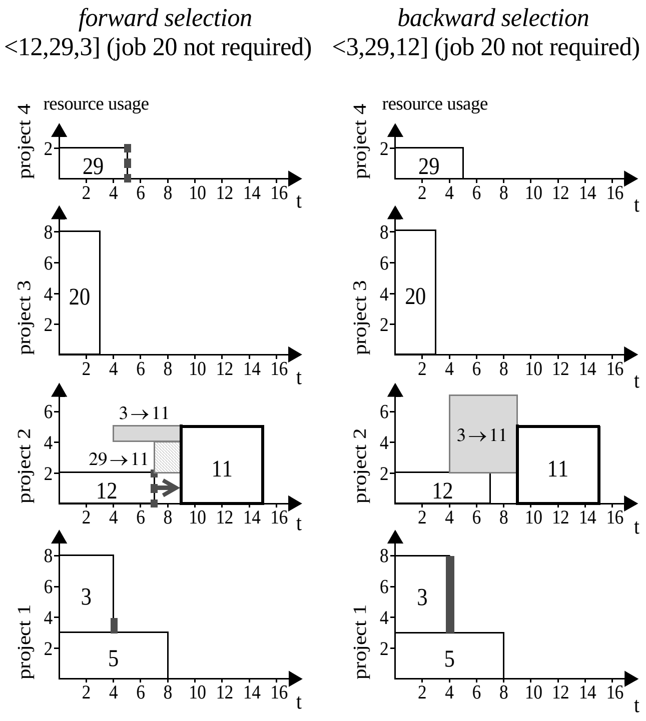

Figure 1. Forward and backward selection

Figure 1 illustrates both selection principles by means of an example with P=4 projects and a single resource with capacity $a_1 = 20$. Each job is visualised as a rectangle with its duration defining the horizontal and its resource usage the vertical extent. Job j = 11 out of project 2 with a resource demand of $u_{11,1} = 5$ is selected by some job rule for the current scheduling time $t = 9$. All jobs in $\mathcal{S} = \{3, 5, 12, 20, 29\}$ are already scheduled. As transfer times we consider $\Delta_{3,11} = 5$, $\Delta_{5,11}=5$, $\Delta_{12,11} = 0$, $\Delta_{20,11} = 6$, and $\Delta_{29,11} = 2$ (last index for r=1 omitted). Due to $F_5 + \Delta_{5,11} = 8 + 5 = 13 > 9$, we obtain $TJ^1 = \{3, 12, 20, 29\}$. The minTT rule determines the priority list <12, 29, 3, 20]. To fulfil the resource needs of job 11, the first three jobs of the list, which can provide 9 resource units, are sufficient. The actual quantities sent by those jobs are different depending on the selection principle used. Forward selection results in transfers from all three jobs as illustrated in Figure 1 while for backward selection only job 3 delivers resource units to task 11.

Assuming that transfer times within the same project are smaller than between different projects, the forward selection principle together with the minTT rule ensures that resources are kept and used within the same project if possible. However, as the example shows, this may lead to splitting up transfers (job 3 delivers only a single unit). It could be of advantage if job 3 provided all its free resource units to task 11 since a resource transfer from project 1 to project 2 takes place anyway. Then the resource supply of jobs 12 and 29 could be fully used for other transfers as achieved by backward selection here.

5 Serial scheduling framework

5.1 Basic algorithm

Contrary to the parallel scheduling scheme, the serial one is activity oriented and builds the schedule in n stages with n activities to be scheduled. At each stage g, an activity is selected and scheduled as early as possible such that precedence and resource feasibility is given (Lova and Tormos 2001, p. 267).

Algorithm 2 illustrates the time scheduling frame of the adapted serial scheduling scheme based on the representation of Kolisch and Hartmann (1999). After initialisation, one activity j, which is selected by a job rule, is scheduled at each stage g. For this purpose, the same rules as in case of the parallel scheme can be used (cf. Section 4.2). However, the set of eligible jobs $\mathcal{D}$ is now defined by $\mathcal{D} = \{j \in J' - \mathcal{S} | A_j \subseteq \mathcal{S}\}$. For the selected job j the earliest precedence-feasible finishing time EF_j is determined by usual time calculation. The earliest resource-feasible finishing time ERF_j of job j considers all resource capacities additionally. Finally, the earliest transfer-feasible finishing time ETF_j is calculated. It is the earliest time at which job j can be finished when precedence, resource and transfer constraints are taken into account. Thus, it is used as realised finishing time F_j in the solution schedule. Obviously, $EF_j \le ERF_j \le ETF_j$ holds. When an activity j has been scheduled, it is inserted in the set of scheduled activities $\mathcal{S}$ while available resources and the set $\mathcal{D}$ of eligible jobs are updated.

```
Initialise $\tilde{a}_r(t) = a_r$ for $r \in R$, $t = 0, \dots, T$; $F_1 = 0$; $\mathcal{S} = \{1\}$;
           $y_{jr} := 0$ for $j \in J'$, $r \in R$; $x_{ijr} := 0$ for $i, j \in J', r \in R$;
           $\mathcal{D} := \{j \in J' - \{1\} | A_j = \{1\}\}$;  /* set of eligible jobs
For g := 1 to n do
    Select the best job $j \in \mathcal{D}$      /* apply *job rule*
    $EF_j := \max\{F_h | h \in A_j\} + d_j$
    $ERF_j := \min\{t \ge EF_j | u_{jr} \le \tilde{a}_r(\tau) \; \forall r \in R, \tau \in [t-d_j;t[\}$;
    $ETF_j :=$ CalcResTrans(j, $ERF_j$);
    $\mathcal{S} := \mathcal{S} \cup \{j\}$; $F_j := ETF_j$;   /* schedule job j
    For $\tau \in [F_j - d_j, F_j[$ and $r \in R$ do
        $\tilde{a}_r(\tau) := \tilde{a}_r(\tau) - u_{jr}$;        /* reduce available capacity
    $\mathcal{D} := \{j \in J' - \mathcal{S} | A_j \subseteq \mathcal{S}\}$;     /* update eligible set
```

Algorithm 2. Modified serial scheduling scheme

The function CalcResTrans(j, ERF_j) for determining a transfer-feasible finishing time ETF_j and corresponding resource flows is given in Algorithm 3. It begins with trial starting time $t = ERF_j - d_j$ of the selected job j and increases t incrementally until transfer-feasibility is achieved for all resource types. As in case of the parallel scheme, it has to be ensured that each resource unit required by job j arrives in t or earlier. Moreover, if activity j is inserted between already scheduled jobs, it must be ensured that the broken resource flows can be repaired feasibly. Therefore, for each already scheduled i and each resource r, any transfer from i to an also scheduled k is examined. If it is possible to deliver resource units from i to j at time t or earlier and to transfer them to k after finishing j without delaying job k, a *breakable transfer* is found and k is added to the set K_{ir}. The set TJ^r of jobs potentially delivering resource r to j in-time consists of all jobs which have still undelivered resource units ($y_{ir} > 0$) and of those from which at least one breakable transfer starts ($K_{ir} \ne \emptyset$). All those possible deliveries are summed up in s^r.

```
Initialise $z_r := 0\ \forall r \in R$ ; $t := ERF_j - d_j$ ;          /* $z_r$ adds up the resource units delivered to j
Repeat                                                                /* repeat until feasible starting time t is found
    For each $r \in R$ do                                             /* consider each resource r separately
        For each $i \in \mathcal{S}$ do                               /* set of receiving jobs in breakable transfers
            $K_{ir} := \{s \in \mathcal{S} - \{i\} \mid x_{isr} > 0 \wedge F_i + \Delta_{ijr} \le t \le F_s - d_s - \Delta_{jsr} - d_j\}$;
        $TJ^r := \{i \in \mathcal{S} \mid (y_{ir} > 0 \wedge F_i + \Delta_{ijr} \le t) \vee K_{ir} \ne \emptyset\}$ ;   /* set of potentially delivering jobs
        $s^r := \sum_{i \in TJ^r} \left( y_{ir} + \sum_{k \in K_{ir}} x_{ikr} \right)$                 /* total supply of resource r for j
    If ( $s^r \ge u_{jr}$ for each $r \in R$ ) then                  /* if supply is sufficient for all resources then ...
resource scheduling
        For each $r \in R$ do                                         /* ... generate deliveries for all resource types r
            While $z_r < u_{jr}$ do                                   /* repeat as long as units of r are missing
                Select and remove best delivering job $i \in TJ^r$    /* apply transfer-from rule
                While ($K_{ir} \ne \emptyset$) and ($z_r < u_{jr}$) do   /* repeat for all breakable transfers
                    Select the best receiving job $k \in K_{ir}$      /* apply transfer-to rule
                    $x' := \min\{x_{ikr}, u_{jr} - y_{jr}\}$ ; $K_{ir} := K_{ir} - \{k\}$ ;   /* set transferable quantity, reduce $K_{ir}$
                    $x_{ijr} := x_{ijr} + x'$ ; $x_{ikr} := x_{ikr} - x'$ ; $x_{jkr} := x'$ ; $z_r := z_r + x'$   /* modify transfers
                $x' := \min\{y_{ir}, u_{jr} - y_{jr}\}$ ;             /* transfer remaining supply of i to j
                $x_{ijr} := x_{ijr} + x'$ ; $y_{ir} := y_{ir} - x'$ ; $y_{jr} := y_{jr} + x'$ ; $z_r := z_r + x'$   /* modify transfers
        Return $ETF_j = t + d_j$ ;                                    /* ... terminate function, return feasible time for j
    Else t := t+1;                                                    /* increase trial time t and repeat
```

Algorithm 3. Function CalcResTrans(ERF_j) for feasible starting time of selected job j and resource flows

When enough units are available for all resources, j can be scheduled to start at time t. The delivering activities i are selected from TJ^r one after another according to a transfer-from rule (cf. Section 5.2.1). The receiving jobs $k \in K_{ir}$ of breakable flows are chosen one after another according to a transfer-to rule (cf. Section 5.2.2). The current flow x_{ikr} from i to k is broken up and activity j receives this very quantity from job i while activity j delivers the same amounts back to job k. If still necessary, the remaining resource stocks of activity i (y_{ir}) are used to satisfy the demands of job j. This process is repeated until all resource needs of job j are satisfied.

Figure 2 represents a cutout of a multi-project schedule with 2 projects and a resource capacity of $a_1 = 18$ units of the single resource 1. Activity $j=23$ as part of project 1 is to be scheduled. It requires 6 units of the resource. Regarding all precedence relations, it may start after job 22 is finished. Jobs 21 and 22 deliver all their resource units to already scheduled jobs, which are not part of the cutout (indicated by the solid bar at the end of those jobs). Moreover, it has already been decided that jobs 26 and 27 provide all their resource supply to activities 28 and 29, which for some reasons cannot start earlier. Since transfer times within a project are supposed to be zero, the resources supplied by jobs 26 and 27 are waiting for their usage by 28 and 29 (this slack is indicated by a dotted bar). The right Gantt chart shows that a part of these waiting resources can be transferred to job 23, used to perform it and return to 28 and 29 in-time. If this would not be possible without delaying already scheduled activities, insertion must be forbidden due to the logic of the serial scheme to schedule each task finally. Notice that activity inser-

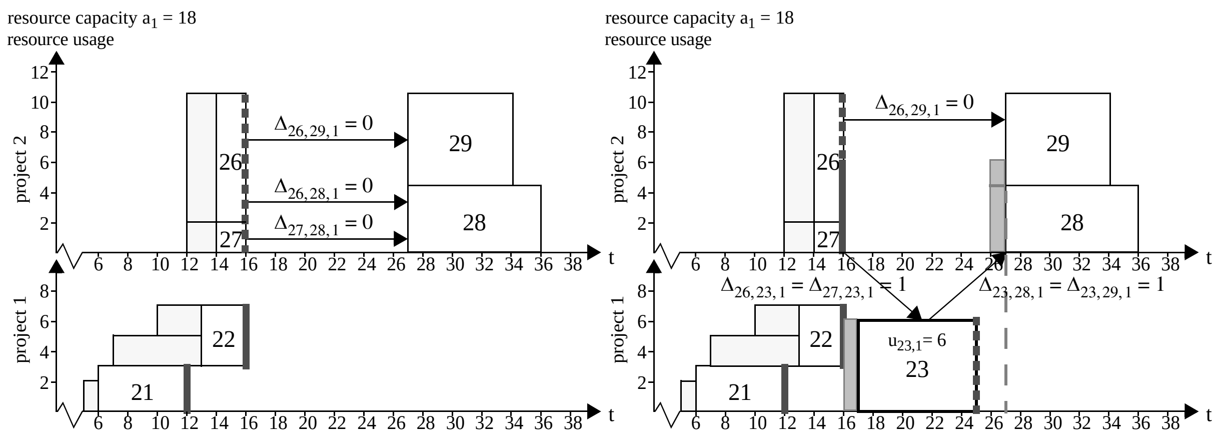

Figure 2. Activity insertion (serial schedule generation scheme)

tion cannot occur in the parallel scheduling scheme, because the starting time t of jobs is increased monotonic from iteration to iteration in this scheme.

5.2 Resource transfer rules

In contrast to the parallel scheme, two types of transfer rules are necessary in the serial scheme because resource transfers to be broken up must be priorised as well.

5.2.1 Transfer-from rules

Like for the parallel scheduling scheme, it must be determined by a priority rule which of the transfer-feasible jobs $i \in TJ^r$ deliver resources to the selected job j. Again, this is to be considered only, if there is a surplus of available resources. The rules that are established in Table 2 for the parallel scheme can also be applied for the serial scheme. Yet, additional rules are possible and summarised in Table 3.

	transfer-from rule	**extremum**	**priority value π'^r_i for $i \in TJ^r$**
time oriented	**minES**	$\min_i$	$\pi'^r_i = F_i + \Delta_{ijr}$
resource oriented	**minTRS, maxTRS**	$\min_i, \max_i$	$\pi'^r_i = y_{ir} + \sum_{k \in K_{ir}} x_{ikr}$
compound	**maxCV**	$\max_i$	$\pi'_i = \lambda_1 \cdot \left(\frac{\overline{\Delta}_{jr} - \Delta_{ijr}}{\overline{\Delta}_{jr} - \underline{\Delta}_{jr}}\right) + \lambda_2 \cdot \left(\frac{\overline{FA}_{jr} - (F_i + \Delta_{ijr})}{\overline{FA}_{jr} - \underline{FA}_{jr}}\right) + \lambda_3 \cdot \left(\frac{y_{ir} - \underline{y}_r}{\overline{y}_r - \underline{y}_r}\right)$ with $\underline{\Delta}_{jr} = \min\{\Delta_{hjr} \mid h \in TJ^r\}$, $\overline{\Delta}_{jr} = \max\{\Delta_{hjr} \mid h \in TJ^r\}$ $\underline{FA}_{jr} = \min\{F_h + \Delta_{hjr} \mid h \in TJ^r\}$, $\overline{FA}_{jr} = \max\{F_h + \Delta_{hjr} \mid h \in TJ^r\}$ $\underline{y}_r = \min\{y_{hr} \mid h \in TJ^r\}$, $\overline{y}_r = \max\{y_{hr} \mid h \in TJ^r\}$

Table 3. Additional transfer-from rules for resource transfers (serial scheduling scheme)

Besides minTT and minGAP, minES is another time oriented rule. It selects that job i which leads to the earliest starting time of job j. The group of resource oriented rules is extended by the total resource supply (TRS) rule, which considers not only still freely available resource stocks of a job but also the resources that can be provided by breaking up existing resource flows feasibly. For the parallel scheme, this rule does not differ from the resource stock (minRS, maxRS) rule since breaking up resource flows (activity insertion) is impossible.

Finally, a compound priority rule was built. It joins time and resource oriented aspects by considering relative transfer times, relative earliest starting times and relative resource stocks. In each category, the relative values are obtained by normalising the absolute values to the interval [0,1] with the best job in this category set to 1 and the worst one set to 0. Finally, the ratings of the three categories are weighted and summed up. The weights can be set according to the preferences of the decision maker, whatever he assumes most important. For the parallel scheme, the relative earliest starting time component is irrelevant but can be replaced by the relative gap (with the biggest gap as best value) to form a compound rule for this scheme, too.

The delivering jobs for the resource transfer can be selected from the priority list *forwards* or *backwards* just as described in Section 4.3 for the parallel scheduling scheme until the resource demand of activity j is fulfilled.

	transfer-to rule	**extremum**	**priority value** π''^{r}_{k}
time oriented	**minTT**	$\min_k$	$\pi''_k = \Delta_{jkr}$
resource oriented	**maxFlow**	$\max_k$	$\pi''_k = \begin{cases} x_{ikr} & \text{if } F_j + \Delta_{jkr} \le F_k - d_k \\ 0 & \text{else} \end{cases}$

Table 4. Transfer-to rules for breaking up flows

5.2.2 Transfer-to rules

If an activity i, which has been selected by a transfer-from rule as described in the preceding section, delivers more units of a resource r to already scheduled activities than still needed by job j, i.e., $\sum_{k \in K_{ir}} x_{ikr} > u_{jr} - z_r$ (for z_r see Algorithm 3), the sequence of breaking up resource flows becomes relevant. Thus, an additional priority rule (called *transfer-to rule*) is required to determine which of the existing resource flows from j to other activities $k \in K_{ir}$ are broken up to satisfy the demand of job j for resource r. Table 4 presents two transfer-to rules.

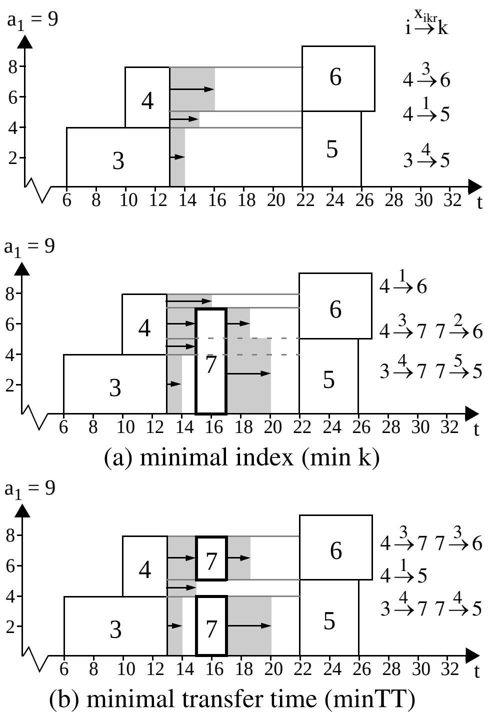

Figure 3. Example for breaking up flows

Figure 3 illustrates the effect of the rules. Transfer times are indicated by grey areas. Job 7, with a resource demand of $u_{7,1} = 7$ for the only resource $r = 1$, is to be inserted between jobs 3, 4 and 5, 6. When job 3 is selected first as delivering activity, the insertion $3 \rightarrow 7 \rightarrow 5$ does not need a priority rule because all resource units of the flow $3 \xrightarrow{4} 5$ must be redirected via job 7. When job 7 breaks the flows from task 4 afterwards, an additional rule is necessary since these flows provide one more resource unit than needed. A simple rule based on increasing job numbers breaks flow $4 \xrightarrow{1} 5$ first and then $4 \xrightarrow{2} 6$ (Figure 3 (a)). The remaining flow $4 \xrightarrow{1} 6$ is not disturbed. The minTT rule breaks flows which lead to a minimal transfer time from job j to k primarily. Due to $\Delta_{7,6} = 2 < 3 = \Delta_{7,5}$, $4 \xrightarrow{3} 6$ is broken and $4 \xrightarrow{1} 5$ is kept. The advantage of the minTT rule is that the gap between the inserted job j and the receiving job k will be relatively large. Thus, other jobs may be inserted between j and k in later scheduling steps. In case of the rule maxFlow large flows are broken first in order to break as few flows as possible when job j is inserted. For the example, this rule would also result in schedule (b).

5.3 Resource based version of the serial scheme

The serial scheme as well as our adaptation to RCMPSPTT described above rely on minimising the starting time for each job j selected to be scheduled (called *time based serial scheme*). This might lead to arranging resource flows in a disadvantageous manner. In order to overcome this problem, another approach may be used which focuses on resource flows primarily.

This *resource based serial scheme* works as follows: For each job j, resource transfers from all scheduled jobs are considered possible, defining the extended set $ETJ^r = \{i \in \mathcal{S} \mid u_{ir} > 0\}$ of all potentially delivering jobs i. From this set, jobs are selected using a transfer-from rule one after another until the supplied quantity of all resources is just sufficient for job j. For this minimal subset I^r of the most promising potentially delivering jobs, the earliest feasible starting time of job j is computed using a transfer-to rule for sorting the flows to be broken as described in Algorithm 3. If not enough flows are breakable, a feasible schedule cannot be generated. For resources r that cause infeasibility, a further potentially delivering job i is added to I^r according to the transfer-from rule used. Finding a feasible (earliest) starting time for j is tried again. This process stops with a feasible starting time t and the corresponding end time $ETF_j = t + d_j$ for job j as in case of the time based version of the serial scheme (see Algorithm 3).

transfer-from rule	extre-mum	priority value π'^r_i for $i \in ETJ^r$
minGAP	$\min_i$	$\pi'^r_i = \begin{cases} gap_{ijr} & \text{if } gap_{ijr} \geq 0 \\ \lvert gap_{ijr} \rvert + T & \text{else} \end{cases}$
min|GAP|	$\min_i$	$\pi'^r_i = \lvert gap_{ijr} \rvert$

Table 5. Modified transfer-from rules for resource transfers (resource based serial scheduling scheme)

Within the resource based serial scheme, the same transfer-from rules as collected in Table 2 and 3 can be applied. However, for the minGap rule the question of a trial time t which serves as reference point for gap computation arises: An initial trial time is given by $t := ERF_j - d_j$. For this trial time, $gap_{ijr} := t - (F_i + \Delta_{ijr})$ is calculated for each job in $ETJ^r = \{i \in \mathcal{S} \mid u_{ir} > 0\}$ for the first resource r to be examined. Now, as many jobs as necessary for satisfying the resource demand of job j for the considered resource r are chosen from the priority list and collected in I^r. If only jobs with positive gaps are chosen, trial time t can be achieved for resource r. If jobs with negative gaps are also necessary to satisfy the demand of j for r, trial time t will not be a feasible scheduling time. The modified priority rule minGAP, therefore, prefers jobs which will increase the intended scheduling time least (see Table 5). If sufficient resources can be provided by free resource stocks and feasibly broken flows, i. e., I^r offers enough resources, job j is (temporarily) scheduled as early as possible at time $\tau := \max\{F_i + \Delta_{ijr} \mid r \in R \wedge i \in I^r\}$ for the examined resource r. To schedule transfers for another resource type r' the trial time t is set to $t := \tau$ and the above process is repeated. If not enough resources of r' can be delivered up to t, this trial time needs to be increased. This might result in unscheduling the resource flows of already examined resource types. If for any of these resource types r flows from i to k were broken, it must be calculated whether the insertion of job j for resource type r is still feasible for the new trial scheduling time t. If not, they must be rescheduled assuming this new trial time. This process is repeated until for all resource types feasible resource transfers can be scheduled.

An additional transfer-from rule $\min|GAP|$ is introduced for the resource based serial scheme, which is a an adaptation of minGap and considers only the absolute deviations from the scheduling time t. It is assumed that resources should be delivered as closely to trial time t (before and after) as possible. Hence, positive and negative deviations from t are rated equally (see Table 5).

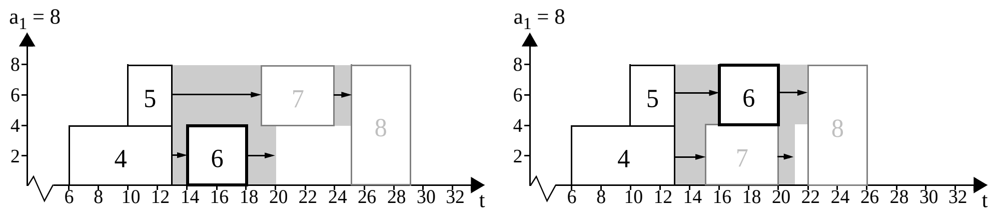

Figure 4. Time based versus resource based serial scheme

The potential advantage of the resource based view is illustrated in Figure 4. Job 6 is to be scheduled after task 4 and 5. The time based serial scheme schedules job 6 at the earliest time t = 14 by providing it with resources from job 4 because it is the only one allowing this minimal starting time ($TJ^1 = \{4\}$). In a next step, activity 7 is selected by the job rule. Its earliest possible starting time is t = 19 after having received its resources from task 5. Finally, job 8 is scheduled at t = 25. Using the resource based serial scheme, we get $ETJ^1 = \{4,5\}$ for j=6. Thus, the transfer $5 \rightarrow 6$ may be chosen by an appropriate transfer-from rule instead of $4 \rightarrow 6$. Job 6 does not start earliest possible but the project duration is, nevertheless, shortened since job 7 can be scheduled in t = 15 due to a shorter transfer time when it receives it resources from task 4 instead of 5. Using ETJ^r obviously allows a greater range of possible schedules, because there are more possibilities for resource flows than with TJ^r. Notice that there is no difference if all transfer times are zero, i.e., in case of RCPSP and RCMPSP because each job is scheduled the earliest by both approaches.

6 Computational experiments

In order to examine the performance of the different heuristic procedures developed for the new problem RCMPSPTT, we carry out a number of computational experiments. On the one hand, we consider the effect of transfer times within single projects as that has not been done before. On the other hand, we examine the consequences of transfer times in a multi-project environment. The heuristic frameworks were coded in the programming language C and the tests have been performed on an Intel Pentium 4 processor with 3.2 GHz and 1 GB RAM.

6.1 Classification scheme for tested procedures

The heuristic frameworks presented in Section 4 and Section 5 consist of different generation schemes and priority rules which can be combined to a considerable number of concrete heuristic procedures. In order to reference these procedures in a comfortable manner, we use a classification tuple $(a|b|c|d|e)$ with the following meanings:

a	generation scheme; $a \in \{par, tbser, rbser\}$ for parallel scheme, time based and resource based serial scheme
b	job rule (see Table 1)
c	transfer-from rule (see Table 2, Table 3, Table 5)
d	planning direction; $d \in \{fwd, bwd\}$ for forward and backward selection of delivering jobs (cf. Section 4.3)
e	transfer-to rule (see Table 4); left empty for the parallel scheme

For example, (tbser | minLFT_SP | minTT | bwd | maxFlow) specifies a procedure using the time based serial scheduling scheme, the job rule minLFT_SP, the transfer-from rule minTT, backward selection and the transfer-to rule maxFlow.

6.2 Experiments for single projects

At first, we evaluate the heuristic frameworks for single projects with and without transfer times. This experiment allows conclusions for real single projects and combined multi-projects within the single project approach of multi-project management.

6.2.1 Generating test data

The experiment is based on the standard benchmark data set J60 of the ProGen library (Kolisch et al. 1995). To get a stable and reliable basis of comparison, all 480 project instances of the J60 set have been solved by Scatter PROGRESS (Klein and Scholl 1999, 2000). 400 of these instances could be solved optimally within 3600 s. These 400 instances and the corresponding optimal solutions were used to build a data set containing transfer times. Transfer times have been introduced for each resource type such that optimality of the solution found by Scatter PROGRESS is retained. This is done in order to have available optimal solutions for RCMPSPTT such that the quality of heuristic solutions can be judged in the best possible manner.

Moreover, the generated transfer times comply with triangular inequalities for all (h,i,j)-combinations with $h,i,j \in J$ and $h \neq i \neq j$. This means that the time for a transfer from h to j cannot be reduced by sending the resource via i. Under these conditions, symmetric transfer times $\Delta_{ijr} = \Delta_{jir}$ are generated systematically and randomly for each resource type $r \in R$. In a first step, the optimal time schedule is complemented by a corresponding feasible resource flow. This is done by solving the resource flow problem contained in the mathematical programme given the optimal finishing times F_j of all jobs (cf. Section 3). When a job i delivers a resource r to another job j in this resource flow, feasible transfer times are bounded from above by $F_j - d_j - F_i$. In order not to get a very tight and unrealistic problem, the maximal value allowed is, however, set to $\overline{\Delta}_{ijr} := 0.5 \cdot (F_j - d_j - F_i)$. For relations (i,j) without an actual resource flow, a maximal transfer time $\overline{\Delta}_{ijr}$ is sampled uniformly from an interval [0, $\max_{ijr}$]. The upper bound $\max_{ijr}$ depends on the time span between the first and the last usage of resource r in the optimal schedule as well as the number of activities that require r during this period. It can be interpreted as the average time an activity that requires the considered resource r could use this resource in the optimal solution until r is set free ultimately. This value is used to limit transfer times to realistic durations which may be lower or higher than real activity durations but will not be extraordinary high. Transfers from the global source and to the global sink node are set to zero because resources are assumed to be already available at the respective jobs when the project starts and need not be transferred elsewhere after the project has been finished ($\overline{\Delta}_{e_0ir} = \overline{\Delta}_{ie_0r} = 0 \ \forall i \in J', r \in R$). Furthermore, $\overline{\Delta}_{iir} = 0 \ \forall i \in J', r \in R$. Given these maximal transfer times for each potential relation (i,j), a linear programme is formulated and solved which sets all transfer times as large as possible such that the triangular conditions and the upper limits are fulfilled. Due to the triangular conditions, in many cases smaller transfer times than $\overline{\Delta}_{ijr}$ must be chosen such that no more than 80 % non-zero elements are contained in the transfer time matrices of the 400 modified J60 projects within the data set on average.

6.2.2 Selected results

As a reference point, we apply the heuristic frameworks to the 400 selected instances of the original J60 data set (all transfer times are 0). In this case, both serial schemes are identical (tbser = rbser) and only job rules are required.

scheduling scheme	**tbser**			**par**		
job rule	avg	min	max	avg	min	max
minSLK_SP	6.69	0	55.17	4.64	0	42.71
minSLK_SP(dyn)	6.92	0	44.83	3.49	0	26.19
minLFT_SP	2.85	0	31.11	3.87	0	28.99
maxRD_SP	11.01	0	51.72	8.11	0	45.33
FCFS	8.40	0	38.55	7.11	0	40.96
Random	13.15	0	60.94	8.71	0	52.38

Table 6. Selected job rules applied to original J60 data set (relative deviation from optimum in %)

Table 6 summarises the mean relative deviations from optimum for selected job rules. The performance obtained is consistent with test results by Klein (2000). The experiments show that the parallel scheme often generates better results than the serial one. The only exception is minLFT_SP which proves to be the best rule of the test in combination with the serial scheme. An average deviation from optimum of 2.85 % has been reached. For the parallel scheme minSLK(dyn) outperformed other rules with a deviation of 3.49 %. However, minLFT is not far behind with 3.87 %. Concluding one can remark that critical path based rules like minLFT or minSLK generally turn out to be more efficient than resource based rules like maxRD.

When resource transfer times are added to the problem, the results get significantly worse. At first, several transfer-from rules were evaluated. Here, the results only for some of the tests are presented, a large number of further tests confirm the obtained statements. Since minLFT proved to be the best job rule for the serial scheduling scheme and one of the best ones for the parallel scheme in absence of transfer times, it is selected as a basis of this very first analysis. In order to make a ceteris paribus comparison of different transfer-from rules, we choose backward selection and the transfer-to rule maxFlow. Table 7 summarises the results (omitted values indicate the corresponding rule being inappropriate for that scheme).

scheduling scheme	**rbser**			**tbser**			**par**		
transfer-from rule	avg	min	max	avg	min	max	avg	min	max
minTT	141.65	0	375.00	13.91	0	93.20	10.36	0	45.92
minES	18.12	0	94.18	17.98	0	94.18	–	–	–
minGAP	11.04	0	86.41	10.98	0	82.52	10.47	0	56.48
min |GAP|	89.65	4.48	263.51	–	–	–	–	–	–
minRS	236.76	41.00	470.31	13.62	0	96.12	10.37	0	56.48
maxRS	241.59	21.21	583.33	15.19	0	93.20	10.40	0	54.63
minTRS	202.83	27.47	433.33	14.50	0	93.20	–	–	–
maxTRS	257.08	31.88	601.85	14.18	0	94.18	–	–	–
maxCV	126.40	0	322.03	14.59	0	93.20	10.42	0	51.61
Random	142.11	23.08	324.07	14.56	0	94.18	10.43	0	51.61

Table 7. J60 test with (* | minLFT_SP | * | bwd | maxFlow)-combinations (relative deviation from optimum in %)

The results show huge differences for the resource based serial scheme. The mean deviation from optimum lies between 11.04 % and 257.08 % while the maximum deviation even reaches 601.85 % for the maxTRS rule. The same is true for tests considering other combinations of job rules, transfer-to rules and planning directions. It becomes clear that for the resource based serial scheme only minGAP and minES are good transfer-from rules. The four resource based rules, however, show very poor performance as the extended set of potentially delivering jobs together with a resource oriented point of view leads to unnecessarily enlarged starting times.

The time based serial scheme avoids these large deviations and gets rather acceptable results for most of the rules and slightly better results even for minGAP and minES. Here, the different sets of potentially delivering jobs have only a minor effect due to preferring resource transfers that allow for an early starting of job j. Since minGAP is the rule which tries to avoid unproductive times of resources most consequently, it performs best in both serial schemes.

However, the parallel scheme turns out to be better whatever transfer-from rule is used. The differences between these rules are marginal. Even the random rule gets the same performance level. Obviously, the strict manner of using resources in each period as completely as possible reduces the degree of freedom in making transfer decisions thereby reducing the potential of making wrong decisions.

As a first result, we can state that in case of single projects with transfer times applying the parallel scheme is usually preferable to applying one of the serial schemes. Concerning the serial scheme, the time based version clearly outperforms the resource based one. In both cases, selecting a transfer-from rule is a sensitive decision.

In any case, the transfer-from rule minGAP is a good decision. Therefore, we fix this rule for a further test which examines job rules for all schemes. The results are presented in Table 8.

scheduling scheme	**rbser**			**tbser**			**par**		
job rule	avg	min	max	avg	min	max	avg	min	max
minSLK_SP	32.56	0	111.63	32.23	0	124.00	6.09	0	49.07
minSLK_SP(dyn)	34.48	0	133.33	34.33	0	133.33	11.48	0	60.22
minLFT_SP	11.04	0	86.41	10.98	0	82.52	10.47	0	56.48
maxRD_SP	31.82	0	108.33	34.98	0	114.46	9.60	0	46.94
FCFS	13.50	0	71.58	13.63	0	71.58	13.77	0	59.14
Random	36.15	0	133.33	35.66	0	133.33	10.53	0	58.95

Table 8. J60 test with (* | * | minGAP | bwd | maxFlow)-combinations (relative deviation from optimum in %)

It reveals that in case of both versions of the serial scheme only two rules are acceptable, namely minLFT_SP and FCFS. The latter has indeed a larger average deviation but a smaller maximal one. The performance of rbser is for most rules slightly worse than that of tbser, except for FCFS and maxRD_SP where it is even better. Obviously, the chosen transfer-from rule minGAP makes the schemes almost identical as it prefers resource transfers that enable early starting times for the job j to be scheduled. However, for maxRD_SP the difference between both serial schemes is a lot larger than for other rules. Thus, one can assume that for resource based job rules the resource based serial scheme is more suitable than the time based one.

The overall best results are achieved by the parallel scheduling scheme again. The best job rule for this scheduling method is minSLK_SP with a mean deviation of only 6.09 %. It must also be noticed that the resource based job rule maxRD_SP is often the 2rd or 3nd best rule for all scheduling schemes, which is a completely different result to the evaluation without transfer times where it was one of the worst. The differences between all scheduling schemes diminish when FCFS is used as a job rule since it aims at scheduling jobs in order of increasing starting times even in the serial scheme.

In order to find the overall best scheme-rule combination, a further systematic test is performed using the job rule minSLK_SP and the parallel scheme by varying the transfer-from rules (c) and the planning direction (d) as summarised in Table 9. It turns out that only small differences exist. The best combination is (parallel | minSLK_SP | minGAP | fwd | ∅) with an average relative deviation of 5.81 % closely followed by the combination (parallel | minSLK_SP | minTT | bwd | ∅). However, even the random selection of delivering jobs gives reasonable result.

c \ d	**fwd**	**bwd**
minTT	5.92	5.82
minGAP	5.81	6.09
minRS	5.99	5.99
maxRS	6.39	6.14
maxCV	6.15	6.11
Random	6.40	6.08

Table 9. Results for combinations (par | minSLK_SP | * | * | ∅)

To summarise, the presented results allow the following conclusions. Basically, the parallel scheme should be preferred. If a serial scheme should be applied the time based one is slightly preferable. In both cases, selecting an appropriate transfer-from rule is more essential than selecting the job rule. On the contrary, the job rule is more important for the parallel scheme.

Transfer-to rules and the planning direction have much less influence as further test series, which are not documented here, show. Perhaps, the following recommendation may be given: If the aim of project management is to keep resources in the project as long as possible, forward selection (fwd) should be applied together with the transfer-to rule minTT. If transferring as many resource units as possible when a resource must change projects anyway is the approach to project management, backward selection (bwd) together with the transfer-to rule maxFlow will be preferable.

6.3 Experiments in a multi-project environment

In the following, experiments concerning the performance of the heuristic frameworks within a multi-project environment are reported. A corresponding data set is generated and selected results are given and interpreted.

6.3.1 Generating test data

The multi-project data set is based on the well-known Patterson data set (Patterson 1984) as it contains different projects of different sizes which enables to construct realistic multi-projects. A total of 100 multi-project instances are generated with $|J| \in [93, 204]$ and $|R| = 3$ by randomly choosing five Patterson instances, respectively, each forming a project within the multi-project, i. e. $|P| = 5$. The common capacity a_r of any resource $r \in R$ in the new multi-projects is sampled uniformly from the interval

$$[\max\{\max\{u_{jr}| \, j \in J'\}, \min\{a_{pr}| \, p \in P\}\}; \textstyle\sum_{p=1}^{5} a_{pr} - \max\{a_{pr}| \, p \in P\}]$$

with a_{pr} denoting the capacity of resource r in the original project p. To ensure existence of a feasible solution, the lower bound of the interval for resource type r is given by the maximal resource usage of any activity within the multi-project or the minimal resource capacity of any original single project whatever is greater. The upper bound ensures that the projects actually compete for the resources.

Transfer times are once again generated randomly depending on minimal and maximal activity durations and considering triangular inequalities. In order to reflect realistic conditions in many multi-projects, it is assumed that non-zero times only occur for transfers between projects. Transfers from the sink and to the source are given zero times as well.

This data set is used to test the single- and the multi-project perspective based on the multi-project duration increase MPDI and the mean project delay MD, respectively. Yet, as optimal solutions are not known, both objective values are calculated using the critical path lower bounds LB1 and $LB1_p$, respectively, as commonly recommended (see Section 3).

6.3.2 Selected results

To examine the performance of the heuristic frameworks depending on job rules and scheduling schemes, we again fix the transfer-to rule to minGAP, the transfer-to rule to minFlow, and use the backwards selection. In order to judge the influence of transfer times, the same test is performed completely ignoring these times on the one hand and considering them on the other. The results are summarised in Table 10, the best rules in each group are highlighted.

The results for the transfer-less data set (columns 2-4) are largely consistent with that reported in Lova and Tormos (2001). For minimising the objective MPDI of the single-project approach, the parallel scheme performs always better than the serial one. However, for minimising the objective MD of the multi-project approach, it depends on the job rule which scheme is better. Moreover, for minimising MPDI, which is the objective for the single-project approach, all rules developed for this very approach (lower part of Table 10) perform better than their multi-project counterparts (upper part of Table 10). This is not only true for the serial scheme but also for the parallel one, with exception of maxRD. Yet, for minimising MD, rules specialised on the multi-project approach perform better than the corresponding single-project rules.

	no transfer times considered				**transfer times considered**					
	MPDI [%]		**MD [days]**		**MPDI [%]**			**MD [days]**		
scheme / job rule	**tbser**	**par**	**tbser**	**par**	**rbser**	**tbser**	**par**	**rbser**	**tbser**	**par**
minSASP	141.63	126.89	31.41	30.05	210.98	208.33	150.67	47.81	47.36	39.78
minSLK_MP	117.06	115.91	41.60	41.23	197.91	193.76	143.52	71.84	69.98	45.06
minLFT_MP	105.12	101.95	39.32	38.41	198.63	196.72	135.04	76.01	75.28	41.21
maxTWK	121.18	113.76	36.08	35.18	179.31	184.34	151.91	46.94	48.14	47.53
maxRD_MP	112.34	107.21	39.73	39.91	175.48	179.20	136.51	57.54	58.22	45.03
minSLK_SP	105.23	103.71	44.78	43.69	183.74	180.31	135.73	74.30	73.42	46.73
minSLK_SP(dyn)	110.41	89.67	41.49	49.19	182.40	180.33	147.21	65.50	64.75	52.33
minLFT_SP	93.77	90.40	50.35	48.47	193.79	192.28	138.77	96.29	95.51	52.43
maxRD_SP	111.41	111.95	45.80	43.10	179.31	184.34	140.19	46.94	48.14	45.55
FCFS	103.84	102.19	44.70	43.37	206.82	205.31	145.64	88.46	87.59	44.99
Random	126.65	116.39	43.79	42.61	199.06	195.50	144.67	62.33	60.58	45.93

Table 10. Results for (* | * | minGAP | bwd | maxFlow)-combinations

The overall best combination for MPDI uses the parallel scheme together with the rule minSLK_SP(dyn), directly followed by minLFT_SP. The latter rule is the best one combined with the serial scheme. The rules maxTWK, maxRD_MP and minSASP, as rules for the multi-project approach, are among the worst job rules for this objective, whereas they turn out to be among the best ones for minimising MD.

When resource transfers are considered (columns 5-10), the parallel scheme is now generally superior to the serial ones irrespective of the objective confirming our results in Section 6.2. However, job rule performance is quite different to the transfer-less test. Likewise the results of Section 6.2.2, for both objectives, it is confirmed that the time based serial scheme is (slightly) better than the resource based one for time oriented job rules. Moreover, for both objectives, it can be affirmed that the resource based serial scheduling scheme performs better than the time based one for resource oriented job rules. It emerges that the resource based scheduling scheme together with maxTWK or maxRD_SP achieves best results and outperforms the time based scheme for MPDI and MD in general. Therefore, it can be stated that in presence of transfer times, the resource orientation of serial scheduling procedures has a positive effect.

Moreover, it can be observed that for minimising MPDI for both serial schemes single-project time oriented rules are once again better than their multi-project counterparts. However, when it comes to resource oriented rules, the approach, for which the rules are designed, is irrelevant. MaxRD_MP, maxTWK and maxRD_SP, which perform poorly when no transfer times occurred, produce best results now. The other way around, minLFT_SP, FCFS and minLFT_MP, which are pretty good rules before, turn out to be some of the worst now. For the parallel scheme, minLFT_MP, which is a multi-project rule, appears to be the best rule. Thus, the preference of single-project rules for minimising MPDI with the parallel scheme cannot be supported when transfer times occur. Yet, minSLK_SP and minLFT_SP are only slightly worse. Consequently, independent of the considered approach the rules are intended for, one can say that minLFT_MP, minLFT_SP and minSLK_SP show a good performance again. Yet, minSLK_SP(dyn) which is best in the transfer-less test is one of the worst rules in presence of transfer times.

The conclusions for minimising MD are quite different. For both versions of the serial scheme and the parallel scheme, maxTWK and SASP, which already obtain good results without resource transfers, and now additionally maxRD_SP outperform other rules again. The SASP rule is once again the best rule with the parallel and the time based serial scheme.

A final experiment indicates once more that the remaining components c, d, and e of the heuristic frameworks, in particular the transfer-from-rules, have considerable influence on the performance of the serial schemes and minor influence on the parallel scheme. This is achieved by selecting the actual transfer-from rule, transfer-to rule and the planning direction randomly. Furthermore, in each iteration of the serial scheme it is decided randomly, if the normal or extended set of potentially delivering jobs is determined (random selection between tbser or rbser). Table 11 summarises the results. Obviously, the serial scheme performs considerably worse while the parallel one is effected only slightly.

	MPDI [%]		MD [days]	
job rule \ **scheme**	***ser**	**par**	***ser**	**par**
minSASP	538.99	153.8	129.28	39.45
minSLK_MP	627.26	148.04	226.40	45.32
minLFT_MP	524.55	139.19	195.47	41.12
maxTWK	529.78	156.11	159.93	48.03
maxRD_MP	542.84	139.33	177.57	47.16
minSLK_SP	600.84	139.78	220.16	47.70
minSLK_SP(dyn)	573.78	151.58	186.00	53.50
minLFT_SP	526.86	141.42	237.18	52.40
maxRD_SP	612.01	142.43	255.08	46.40
FCFS	483.10	149.34	195.88	49.40
Random	552.84	148.28	178.34	45.71

Table 11. (* | * | Random | Random | Random)-combinations

Further extensive tests not reported here indicate that the transfer-from rule minGAP is the best transfer-from rule in (almost) all cases and that the influence of the transfer-to rule and the planning direction is negligible. Thus, the results presented in Table 10 indeed show the best available performance of the tested combinations of job rule and scheduling scheme.

To summarise the most important findings, transfer times should not be neglected when scheduling projects, because they have an important influence on objective values. They must be considered when scheduling a multi-project but the usage of special transfer priority rules is not vital for the parallel scheme, which is in marked contrast to the results for the serial scheme. Moreover, when transfer times occur, resource based information and, thus, resource orientation of the scheduling process gain in importance for priority calculation and the following job selection, especially in the serial scheduling scheme.

7 Conclusions and future research

In this paper, aspects of resource transfers in multi-project scheduling are considered for the first time. We present a mathematical model for the resulting new multi-project scheduling problem with resource transfers. Furthermore, a heuristic framework based on the well-known priority rule based heuristics for the RCPSP and RCMPSP has been developed and tested. The results illustrate that resource transfers should not be neglected since they increase the multi-project duration or the mean project delay whichever is the aim to be minimised by project managers. Yet, it has been shown that applying sensible transfer scheduling rules is of great importance for the serial scheduling scheme whereas for the parallel scheme emphasis must be put on selecting appropriate job rules. Especially resource based rules gain in importance. However, the obtained results are still not satisfying. Deviations from optimum of about 5% on average are still relatively high. It should be aimed at improving the results by developing new and improved solution procedures as meta heuristics, e .g. genetic algorithms. The presented heuristic solution procedure can build an essential basis for this.

Finally, other aspects should be taken into account. For this paper, it is assumed that precedence constraints are defined only within the projects. This assumption holds for multi-project scheduling as researched here. However, the developed models and procedures can be transferred to programme scheduling as well, which requires that precedence constraints between projects become relevant. This can easily be integrated. Moreover, we only analyse time oriented objective functions in this paper. However, in practice cost aspects play an important role as well. Resource transfers increase cost measures. Thus, they should be integrated into the models and solution procedures in next research steps. In a wider context additional considerations should be remarked. The underlying static environment assumption often does not hold for multi-projects in practice. In a real multi-project environment, dynamic aspects as stochastically arriving projects must be considered as well. Furthermore, the changing of project and task portfolios is frequently caused by change requests. These requests are an even more important reason for the dynamic nature of projects. Change requests express a formalised wish to change the characteristics of the project content which the parties to a contract already agreed on. The aspect of projects arriving stochastically to the system are already considered in the literature on dynamic environments. But the way they are dealing with this problem is still not satisfying for applicable project scheduling in the real world. Finally, they are not considering the changing of the structure of projects that are already in the portfolio. In this case of changing project requisitions, the concept of resources changing between projects becomes even more important.

Last but not least, the absence of uncertainty in multi-project scheduling, e.g. in activity durations or resource availability, up to now is a point of criticism that should not be neglected. In single-project scheduling this aspect is already part of research. In multi-project scheduling it has not been considered yet. However, one can assume that deterministic scheduling deals with the uncertainty problem by working with expected activity durations or resource requirements as one-point estimators. Incorporating all aspects of uncertainty into models and solution procedures will certainly lead to very complex systems, especially since the problem becomes more complex in the context of multiple interdependent projects.

Despite all the mentioned open questions, a first step to resolve the criticism of the state-of-the-art research is being made in this paper by considering the sequence- and resource type-dependent resource transfers as an additional aspect to the RCMPSP in a static environment.

References

Anavi-Isakow, S. and B. Golany (2003): Managing multiple-project environments through constant work-in-progress. International Journal of Project Management 21/1, pp. 9-18.

Ash, R. and D. E. Smith-Daniels (1999): The effects of learning, forgetting, and relearning on decision rule performance in multiproject scheduling. Decision Sciences 30/1, pp. 47-82.

Bock, D. B. and J. H. Patterson (1990): A comparison of due date setting, resource assignment, and job preemption heuristics for the multiproject scheduling problem. Decision Sciences 21/2, pp. 387-402.

Brucker, P.; A. Drexl, R. Möhring, K. Neumann and E. Pesch (1999): Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research 112, pp. 3-41.

De Boer, R. (1998): Resource-constrained multi-project management - A hierarchical decision support system. PhD thesis, University of Twente, Enschede.

Debels, D. and M. Vanhoucke (2006): Pre-emptive resource-constrained project scheduling with setup times. Working paper 2006/391, Universiteit Gent.

Demeulemeester, E. L. and W. S. Herroelen (2002): Project scheduling: A research handbook. Kluwer Academic Publishers, Boston.

Dumond, E. J. and J. Dumond (1993): An examination of resourcing policies for the multi-resource problem. International Journal of Operations & Production Management 13/5, pp. 54-76.

Dumond, J. (1992): In a multi-resource environment, how much is enough? International Journal of Production Research 30/2, pp. 395-410.

Dumond, J. and V. A. Mabert (1988): Evaluating project scheduling and due date assignment procedures: An experimental analysis. Management Science 34/1, pp. 101-118.

Fendley, L. G. (1968): Towards the development of a complete multiproject scheduling system. Journal of Industrial Engineering 19/10, pp. 505-515.

Hans, E. W.; W. Herroelen, R. Leus and G. Gullink (2003): A hierarchical approach to multi-project planning under uncertainty. Omega 35/5, pp. 563-577.

Klein, R. (2000): Scheduling of resource-constrained projects. Kluwer Academic, Boston.

Klein, R. and A. Scholl (1999): Scattered branch and bound. An adaptive search strategy applied to resource-constrained project scheduling. European Journal of Operations Research 7/3, pp. 177-201.

Klein, R. and A. Scholl (2000): PROGRESS: Optimally solving the generalized resource-constrained project scheduling problem. Mathematical Methods of Operations Research 52, pp. 467-488.

Kolisch, R. (1995): Project scheduling under resource constraints. Physica, Heidelberg.

Kolisch, R. and R. Padman (2001): An integrated survey of deterministic project scheduling. Omega 29/3, pp. 249-272.

Kolisch, R. and S. Hartmann (1999): Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis. In: J. Weglarz (ed.): Project scheduling: Recent models, algorithms, and applications. Kluwer, Boston, pp. 147-178.

Kolisch, R. and S. Hartmann (2006): Experimental investigation of heuristics for resource-constrained project scheduling: An update. European Journal of Operational Research 174, pp. 23-37.

Kolisch, R.; Sprecher, A. and A. Drexl (1995); Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science 41, pp. 1693-1703.

Kurtulus, I. and E. W. Davis (1982): Multi-project scheduling: Categorization of heuristic rules performance. Management Science 28/2, pp. 161-172.

Kurtulus, I. S. and S. C. Narula (1985): Multi-project scheduling: Analysis of project performance. IIE Transactions 17/1, pp. 58-66.

Lawrence, S. R. and T. E. Morton (1993): Resource constrained multi-project scheduling with tardy costs: Comparing myopic, bottleneck, and resource pricing heuristics. European Journal of Operational Research 64, pp. 168-187.

Lova, A. and P. Tormos (2001): Analysis of scheduling schemes and heuristic rules performance in resource-constrained multiproject scheduling. Annals of Operations Research 102, pp. 263-286.

Lova, A; C. Maroto and P. Tormos (2000): A multicriteria heuristic method to improve resource allocation in multiproject scheduling. European Journal of Operational Research 127, pp. 408-424.

Mika, M.; G. Waligóra and J. Weglarz (2006): Modelling setup times in project scheduling. In: Jósefowska, J. and J. Weglarz (eds.): Perspectives in modern project scheduling. Springer, New York, pp. 131-163.

Mohanty, R. P. and M. K. Siddiq (1989): Multiple projects-multiple resources-constrained scheduling: Some studies. International Journal of Production Research 27/2, pp. 261-280.

Neumann (2003): Project scheduling with changeover times - Modelling and applications. In: Proceedings of the international conference on industrial engineering and production management 1, Porto/Portugal, May 26-28, pp. 30-36.

Neumann, K.; C. Schwindt and J. Zimmermann (2003): Project scheduling with time windows and scarce resources. Springer, Berlin.

Patterson, J. H. (1973): Alternate methods of project scheduling with limited resources. Naval Research Logistics Quarterly 20/4, pp. 767-783.

Patterson, J. H. (1984): A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem. Management Science 30/7, pp. 854-867.

Payne, J. H. (1995): Management of multiple simultaneous projects: A state-of-the-art review. International Journal of Project Management 13/3, pp. 163-168.

Pritsker, L. J.; A. A. B. Watters and P. M. Wolfe (1969): Multiproject scheduling with limited resources: A zero-one programming approach. Management Science 16, pp. 93-108.

Trautman (2001): Anlagenbelegungsplanung in der Prozessindustrie. Gabler, Wiesbaden.

Vercellis, C. (1994): Constrained multi-project planning problems: A lagrangean decomposition approach. European Journal of Operational Research 78, pp. 267-275.

Wiley, V. D.; R. F. Deckro and J. A. Jackson (1998): Optimization analysis for design and planning multi-project programs. European Journal of Operational Research 107, pp. 492-506.

Yang, K. K. and C. C. Sum (1993): A comparison of resource allocation and activity scheduling rules in a dynamic multi-project environment. Journal of Operations Management 11/2, pp. 207-218.

Yang, K. K. and C. C. Sum (1997): An evaluation of due date, resource allocation, project release, and activity scheduling rules in a multiproject environment. European Journal of Operational Research 103/1, pp. 139-154.