Computers & Operations Research 38 (2011) 310–319

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Time-constrained project scheduling with adjacent resources

J.L. Hurink [a], A.L. Kok [b], J.J. Paulus [c], J.M.J. Schutten [d,*]

[a] Department of Applied Mathematics, Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands
[b] ORTEC, Algorithmic R&D, P.O. Box 490, 2800 AL Gouda, The Netherlands
[c] CQM bv, P.O. Box 414, 5600 EK Eindhoven, The Netherlands
[d] Department of Operational Methods for Production and Logistics, School of Management and Governance, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands

ARTICLE INFO

Available online 4 June 2010

Keywords:
Time-constrained project scheduling
Adjacent resources
Project scheduling
Integer programming

ABSTRACT

We develop a decomposition method for the Time-Constrained Project Scheduling Problem (TCPSP) with adjacent resources. For adjacent resources the resource units are ordered and the units assigned to a job have to be adjacent. On top of that, adjacent resources are not required by single jobs, but by job groups. As soon as a job of such a group starts, the adjacent resource units are occupied, and they are not released before all jobs of that group are completed. The developed decomposition method separates the adjacent resource assignment from the rest of the scheduling problem. Test results demonstrate the applicability of the decomposition method. The presented decomposition forms a first promising approach for the TCPSP with adjacent resources and may form a good basis to develop more elaborated methods.

1. Introduction

We develop a decomposition method for project scheduling problems with adjacent resources. Adjacent resources are resources for which the units assigned to a job are required to be in some sense adjacent. Possible examples of adjacent resources are dry docks, shop floor spaces, and assembly areas. We focus on the Time-Constrained Project Scheduling Problem (TCPSP), Guldemond et al. [1], with one one-dimensional adjacent resource. However, the presented concepts and methods can be easily extended to more general models, e.g. multiple one-dimensional adjacent resources or two-dimensional adjacent resources.

The Time-Constrained Project Scheduling Problem (TCPSP) with an adjacent resource is defined as follows. We are given a set of jobs, a set of renewable resources and one one-dimensional adjacent resource. Each job is characterized by a release date, processing time, deadline and its resource requirements, and has to be scheduled without preemption. The processing of jobs is further restricted by precedence relations. The adjacent resource is a special type of resource that is characterized by two properties. First, the resource units of adjacent resource are somehow topologically ordered (in this case ordered on a line) and the resource units assigned to a job have to be neighbored/adjacent and reassignment is not allowed. Second, motivated by the occurrence of adjacent resources in real life problems, we consider the more general case that the adjacent resource is not required only by a single job but by groups of jobs (called job groups or simply groups). As soon as a job of such a job group starts, the assigned adjacent resource units are occupied, and they are not released before all jobs of that group are completed. In the considered model, it is only possible to hire additional capacity for the renewable resources, and not for the adjacent resource. The objective is to find a feasible assignment of the job groups to the adjacent resources and a feasible job schedule that minimizes the cost of hiring additional capacity.

The consideration of adjacent resources in the above-mentioned form is motivated by a cooperation with a Dutch consultancy company. They encountered at several of their clients adjacent resource requirements. Since the project scheduling models in the literature do not cover these requirements, the company either assigns the adjacent resources in advance based on simple rules or they relax the adjacency requirements and repair the achieved solutions afterwards. However, since both approaches do not lead to satisfactory solutions, the company strives to incorporate adjacent resources in their planning software for project scheduling. One practical application is from the ship building industry that we use to illustrates the adjacency requirements. In this problem the docks form one-dimensional adjacent resources, and all jobs related to building a single ship form a job group. Clearly, the part of the dock assigned to one ship has to satisfy the adjacency requirement. As soon as the construction of a ship starts, the assigned part of the dock is occupied until the construction is finished and the ship is removed from the dock. Removal or repositioning of a partially assembled ship is in practice too cumbersome and time consuming and therefore not an option. The other resources required to build the ships (like machines, equipment and personnel) can be

* Corresponding author. Tel.: +31 53 489 4676; fax: +31 53 489 2159.
E-mail address: m.schutten@utwente.nl (J.M.J. Schutten).

0305-0548/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2010.05.006

modeled as renewable resources. The capacity of the dock is fixed but the capacity of renewable resources can be increased, e.g. by hiring additional personnel.

Adjacent resources have some relation with other special resource types considered in the literature. Spatial resources, as introduced in De Boer [2], are also resources which are not only required by a single job but also by a group of jobs. However, no adjacency of the assigned resource units is required. Make-to-order assembly problems under assembly area constraints, see e.g. Hess and Kolisch [3] and Kolisch [4], form a special case of project scheduling problems with spatial resources where each job group requires exactly one unit of the spatial resource. In this case the adjacency requirement is automatically fulfilled. Without the adjacency requirement on the resources, the spatial resource can also be modeled with the concept of cumulative resources, see e.g. Beck [5], Neumann and Schwindt [6] and Neumann et al. [7]. Cumulative resources are, for example, used to incorporate storage facilities into project scheduling problems. When a job group starts the cumulative resource is depleted by a given amount, and replenished as soon as a job group completes. In Section 3.1 we show why an adjacent resource cannot be modeled as a cumulative resource.

The literature that does consider an adjacency requirement on resources, only considers the special case in which exclusively adjacent resources are considered and groups consist of a single job. In this case the scheduling problem can be seen as a two-dimensional packing problem. Examples of this can be found in literature on berth allocation at container terminals, e.g. Guan et al. [8] and Lim [9], reconfigurable embedded platforms, e.g. Fekete et al. [10] and Steiger et al. [11], and check-in desks at airports, see Duin and Van der Sluis [12]. In Hartmann [13] such packing problems are modeled by introducing a mode for each possible placement of a job on the adjacent resource. Consequently, one has to solve a multi-mode project scheduling problem with possibly an exponential number of modes (see Section 3.2).

Relaxing the group and adjacency requirements, the considered problem reduces to the TCPSP as considered in Guldemond et al. [1]. The study of such types of Time-Constrained Project Scheduling Problems started with Möhring [14] and Deckro and Herbert [15], for an overview see Neumann et al. [7].

Summarizing, the concepts of job groups and adjacency requirement on resources have been treated in the literature, but never in a combined manner. To the best of our knowledge this work is the first to consider this combination.

The outline of this paper is as follows. In Section 2 we formally state the Time-Constrained Project Scheduling Problem with one one-dimensional adjacent resource. Additionally, we provide an illustrative example which we use throughout this paper. Before we present the developed decomposition method, we discuss in Section 3 why existing modeling and solution techniques for related problems are not applicable when we are dealing with an adjacent resource. Section 4 describes the decomposition method, the main contribution of this paper. In this approach, first the groups are assigned to the adjacent resource and then the jobs are scheduled. The solution of the group assignment problem implies additional precedence relations between the jobs. Once these precedence relations are added, the scheduling of the jobs can be done with a method for the TCPSP, e.g. the method of Guldemond et al. [1]. In Section 4.3 we introduce objective functions for the group assignment problem in order to steer the assignment to a promising one. Section 5 reports on computational tests. In Section 6 we give some concluding remarks.

2. The TCPSP with adjacent resources

In this section, we start by giving a detailed description of the time-constrained project scheduling problem with adjacent resources (TCPSP with adjacent resources). As mentioned before, we restrict ourselves to a single one-dimensional adjacent resource and, thus, each job group has a requirement for exactly one adjacent resource. However, the presented concepts and methods can be easily extended to more general models, e.g. multiple one-dimensional adjacent resources or two-dimensional adjacent resources. In Section 2.2, we show how an instance of the scheduling problem can be represented by an Activity-on-Node network (AoN network) and at the end of this section, we give an example project with its corresponding AoN network and a corresponding solution to illustrate the problem. We use this illustrative example throughout this paper.

2.1. Formal model description

For a project, we are given a set $\mathcal{J}$ of n jobs, i.e. $\mathcal{J}=\{J_1,\ldots,J_n\}$. Each job J_j has a release date r_j, a processing time p_j, and a deadline d_j. W.o.l.g. we assume that all these and following input parameters are integer. Preemption of jobs is not allowed. The time horizon is divided into T time buckets, $t=0,\ldots,T-1$, where time bucket t represents the time interval $[t,t+1)$ and $T=\max\{d_1,\ldots,d_n\}$. Thus, for each job J_j, time bucket r_j is the first and time bucket d_j-1 the last in which the processing of job J_j can take place. We assume the time windows for each job to be large enough to process the job, i.e. $d_j-r_j\geq p_j$, since otherwise no feasible schedule exists. The processing of the jobs is further restricted by precedence relations, which are given by sets $P_j\subset\mathcal{J}$, denoting all direct predecessors of job J_j. With each precedence relation $J_i\in P_j$ there is an associated non-negative time lag τ_{ij} indicating that there have to be at least τ_{ij} time buckets between the completion of job J_i and the start of job J_j. We assume w.l.o.g. that all release dates and deadlines are consistent with the precedence relations, i.e. $r_i+p_i+\tau_{ij}\leq r_j$ and $d_j-p_j-\tau_{ij}\geq d_i$ for all $J_i\in P_j$. (If this is not the case, it can be achieved by a simple preprocessing.)

For the processing of the jobs there is a set $\mathcal{R}$ of renewable resources, $\mathcal{R}=\{R_1,\ldots,R_K\}$, and one one-dimensional adjacent resource $\overline{R}$ available. Each renewable resource $R_k\in\mathcal{R}$ has a capacity $Q_{k,t}$ in time bucket t, and the adjacent resource has capacity $\overline{Q}$ in all time buckets. Job J_j has a resource requirement q_{jk} for renewable resource R_k during its processing. Additionally, we are given a set $\mathcal{G}$ of job groups, i.e. $\mathcal{G}=\{G_1,\ldots,G_m\}$. A job group $G_g\in\mathcal{G}$ represents a subset of the jobs ($G_g\subset\mathcal{J}$) and has a requirement of $\overline{q}_g$ adjacent resource units. The assigned resource units to group G_g are occupied from the first moment a job in G_g starts, and are released as soon as all jobs in G_g are completed. In principle a job can belong to any number of job groups.

In the considered model we assume the adjacent resource to have a fixed capacity. However, we do allow an increase of the capacity of the renewable resources. Increasing the capacity of renewable resource R_k in time bucket t by one unit, incurs a cost of c_{kt}. The objective is to find a feasible assignment of groups to the adjacent resource units, and at the same time a feasible schedule of jobs on the renewable resources, such that the total costs of increasing the capacity of the renewable resources is minimized.

2.2. Activity-on-Node representation

Representing an instance of a project scheduling problem by an Activity-on-Node (AoN) network is a well known modeling technique from the literature, see e.g. Neumann et al. [7]. In such a network nodes correspond to jobs (or sometimes called activities, hence the name) and arcs to precedence relations. Two dummy jobs having zero processing time and no resource requirements, J_0

Table 1
Example project: job characteristics.

Job	Cleaning crew (R_1)	Painting crew (R_2)	Safety inspector (R_3)	Processing time	Release date	Deadline	Direct predecessors	Job group
J_1	1	0	0	2	0	7	$\emptyset$	G_1
J_2	0	1	0	4	2	11	$\{J_1\}$	G_1
J_3	1	0	0	2	1	7	$\emptyset$	G_2
J_4	0	1	0	4	3	11	$\{J_1,J_3\}$	G_2
J_5	1	0	0	2	4	8	$\emptyset$	G_3
J_6	0	2	0	3	6	11	$\{J_5\}$	G_3
J_7	0	0	1	2	4	11	$\emptyset$	G_3

and J_{n+1}, are added to represent the start and completion of the project, respectively. So, $\{J_0J_1,\ldots J_{n+1}\}$ is both the node and the job set. Each precedence relation $J_i \in P_j$ is represented by an arc from node J_i to J_j with weight τ_{ij}. The release date of job J_j is modeled by an arc from node J_0 to J_j with weight equal to r_j.

The job groups and the deadlines can be incorporated into the AoN network construction for project scheduling problems as follows. To be able to identify the start and completion of the job groups, we add to the project dummy jobs with zero processing time and no resource requirement. For each group G_g we introduce a start job J_g^s and a completion job J_g^c, respectively. Job J_g^s is a predecessor of all jobs in G_g with zero time lag, and has release date $\min_{J_j \in G_g}\{r_j\}$ and deadline $\min_{J_j \in G_g}\{d_j - p_j\}$. All jobs in G_g are predecessors of J_g^c. J_g^c has release date $\max_{J_j \in G_g}\{r_j + p_j\}$ and deadline $\max_{J_j \in G_g}\{d_j\}$. To model the deadlines of the jobs, we fix the scheduling of job J_{n+1} at time T (the largest deadline). The deadline of job J_j now can be modeled by an arc from node J_j to J_{n+1} with weight equal to $T - d_j$.

2.3. Example project

To emphasize the impact of an adjacent resource on the scheduling problem, consider the following illustrative example project from the yacht building industry. There are three yachts, each to be cleaned and painted in the next few days. The third yacht is also due for safety inspection. There is one adjacent resource, the dock ($\overline{R}$) with a length of 30 m ($\overline{Q} = 30$). There are three renewable resources: cleaning crews (R_1), painting crews (R_2), and safety inspectors (R_3). There is one crew for cleaning ($Q_{1,t}=1$), one crew for painting ($Q_{2,t}=1$), and no safety inspector ($Q_{3,t}=0$) available during the entire project horizon. These capacities can (and have to) be extended in some time periods by hiring personnel, e.g. the safety inspector. Table 1 displays the characteristics of the seven jobs. All time lags are of zero length. Job group G_g corresponds to yacht g, and all yachts have to be placed on the one available dock. The lengths of the yachts are given by $\overline{q}_1 = 10$, $\overline{q}_2 = 10$ and $\overline{q}_3 = 20$.

Fig. 1 shows the AoN network corresponding to this example project. All precedence relations implied by transitivity are omitted from the AoN network. Fig. 2 gives a feasible solution for this project. The schedules for the renewable resources R_1 to R_3 indicate that for time buckets 4, 5, 8, 9, and 10 an additional painting crew and for time buckets 7 and 8 a safety inspector has to be hired. For the adjacent resource there are two different schedules displayed in Fig. 2. Both schedules for $\overline{R}$ satisfy the timing constraints (both schedules for $\overline{R}$ have the exact same start and completion times) and both are consistent with the schedules for the renewable resources R_1 to R_3, but only the second satisfies the adjacency requirement. For the renewable resources it suffices to specify the start of each job, but for the adjacent resources the specific assignment of resource units is also necessary. The necessity of specifying the assignment of the groups to the adjacent resources is also the topic of the next section.

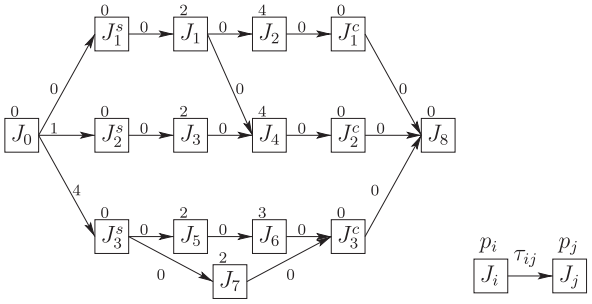

Fig. 1. Example project: Activity-on-Node representation.

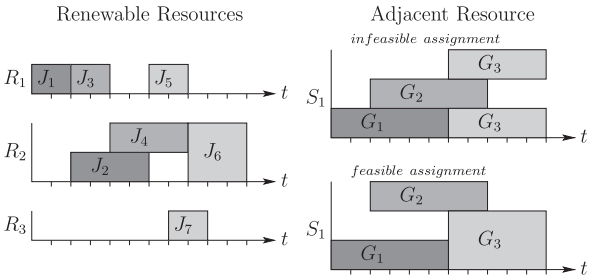

Fig. 2. Example project: Gantt-charts of a solution.

3. Failing modeling techniques

In this section we review two modeling techniques, *cumulative resource* modeling and *multi-mode* modeling, and comment on the use of *sequential planning heuristics* for scheduling problems having job groups. These techniques and methods seem at first glance useful for solving problems with adjacent resources. However, as we show, the additional computational complexity introduced by the adjacency requirement causes these techniques and methods to fail. The TCPSP with adjacent resources contains several elements which make the problem *NP*-hard. If we remove the adjacent resources and the notion of groups, we get the TCPSP which is *NP*-hard even if all processing times are equal to 1. However, by assuming all time windows to be large enough, i.e. $r_j + p_j \le d_j$ for all J_j, and assuming unlimited hiring possibilities, at least there always exists a feasible solution for the TCPSP. In contrast to this, with the addition of only one adjacent resource the problem of deciding whether or not a feasible solution exists, turns out to be NP-complete. Furthermore, it is NP-complete to decide whether or not given group start and completion times that respect the adjacent resource capacity constraints can be extended to a feasible solution respecting also the adjacency requirements without changing the start and completion times of the job groups (see Section 3.1). In Section 3.2 we discuss why multi-mode modeling should not be used and in Section 3.3 we discuss what the pitfalls are when jobs are planned sequentially in the presence of job groups.

3.1. Cumulative resources modeling

Relaxing the adjacency requirement gives a problem that can be modeled with a cumulative resource replacing the adjacent resource. At the start of a group the cumulative resource is depleted and at completion the resource units are again available. Cumulative resources are used to model, for example, inventory levels, see Neumann et al. [7]. This is, however, a proper relaxation of the problem and not a different formulation, i.e. there is no guarantee that a solution for the 'problem with a cumulative resource' can be transformed into a solution for the 'problem with an adjacent resource'.

Modeling groups with cumulative resources, relaxing the adjacency requirements and solving it as such, gives us start and completion times of the jobs and groups, but no assignment of particular adjacent resource units to groups. It is only guaranteed that in each time bucket at most $\overline{Q}$ units of the adjacent resource are used. Determining whether there exist a feasible assignment of adjacent resources given these start and completion times, is a strongly *NP*-complete problem, see Duin and Van der Sluis [12]. As a consequence, there is no guarantee that the start and completion times found with a solution method based on cumulative resource modeling, are such that there exists a feasible assignment of the groups to the adjacent resource units. It is even an NP-complete problem to determine whether there exists such a feasible assignment.

3.2. Multi-mode representation

An possible approach to model the adjacency requirements is to represent each possible placement of a group on the adjacent resource by a different mode, as done in Hartmann [13]. To explain this construction, we assume that each job group consists of exactly one job. We introduce for each possible placement a mode for the job (job group), where a placement is an interval of adjacent resource units of the required length. To be precise, we introduce a set of renewable resources Z_l (with $l=1,\ldots,\overline{Q}$) all with a capacity of 1. Each resource Z_l represents one resource unit of the adjacent resource. For job j with adjacent resource requirement $\overline{q}_j$ we introduce modes m_{ij} (with $i=1,\ldots,\overline{Q}-\overline{q}_j+1$). Mode m_{ij} represents job j being placed on adjacent resource units i to $i+\overline{q}_j-1$. Thus, the resource requirement of job j in mode m_{ij} for resource Z_l is 1 if $i\le l\le i+\overline{q}_j-1$ and 0 otherwise.

The problem with this multi-mode representation is that the number of new resources and modes we have to introduce depends on the input data of a specific problem instance, and not on the input size of the problem. More precisely, with this transformation the instance size grows exponential, since the adjacent resource capacity is encoded in size $\log\overline{Q}$ in the original formulation, and after the transformation we need to specify $O(\overline{Q})$ modes for each job. Thus, this problem transformation is not a polynomial time transformation. From a computational time perspective, but also from a practical point of view, this should be avoided.

A second problem occurs when a constructive planning heuristic is used in combination with such a multi-mode representation. Then, not finding a feasible solution does not mean that the instance is infeasible (see Section 3.3). The decomposition method presented in Section 4 does not have this drawback and gives proof of infeasibility if it occurs.

3.3. Sequential planning heuristic

A sequential planning heuristic includes the jobs one by one into a schedule. This approach is the basis of almost all constructive heuristics for project scheduling problems. However, there is a large pitfall when we consider instances of the TCPSP with job groups, whether it be with an adjacent resource or a cumulative resource. As soon as the first job of a group is selected to be scheduled next, the group has to be assigned to the adjacent resource (or a given amount of the cumulative resource is depleted). These resource units stay assigned (or depleted) until the last job of the group is completed. However, in general it is hard to estimate when this will be. Thus, it is unclear how long other jobs may be delayed.

As a consequence, there is no mechanism to predict whether or not starting a certain group will cause jobs of other groups to miss their deadline. Thus, for a partially created schedule one cannot ensure that it can be extended to a complete feasible schedule.

For the decomposition method described next, this is not an issue. The assignment of the groups to adjacent resources is done such that a feasible job schedule exists.

4. The decomposition method

The considerations in the previous section indicate that for the TCPSP with adjacent resources no direct and simple heuristic can be found, since the problem of finding a feasible assignment of the adjacent resource for a given timing is already *NP*-complete. Furthermore, this fact also implies that it does not make sense to first treat the timing of the jobs and then the assignment of adjacent resources. Even more, it indicates that the problem of getting a feasible assignment of the adjacent recourse units to groups should play a central role and be treated first. Therefore, in this section, we present a decomposition method which considers first the feasibility of the assignment to the adjacent resource, and second the timing of the jobs. By considering the assignment of the groups first, we can use a sequential planning heuristic to schedule the jobs in the second stage and do not run into the problems mentioned in the previous section.

Since already the feasibility question of the assignment of the adjacent resource is NP-complete, we choose to use an exact approach in this first stage. It is based on an ILP formulation.

The outline of the decomposition method for the TCPSP with adjacent resources (which we refer to as the *original problem*) is as follows. The decomposition method separates the adjacent resource assignment from the problem of scheduling the jobs. The first step is to determine an assignment of the groups to the adjacent resource units, and an ordering between those groups that are assigned to at least one common adjacent resource units. We call this the group assignment problem (GAP). Let F_{GAP} denote the set of all feasible solutions of the GAP. For a solution $a\in F_{\mathrm{GAP}}$, we have an ordering of the groups that are assigned to the same adjacent resource. This ordering implies additional precedence relations in the original problem, i.e. if groups g and h share an adjacent resource unit in a solution of the GAP and g is scheduled before h then no job of group h can start before the completion of all jobs in group g. After adding these implied precedence relations, the adjacent resources do not have to be considered anymore. The resulting problem is a TCPSP without adjacent resources (which we refer to as the *resulting TCPSP*). Denote this resulting TCPSP for $a\in F_{\mathrm{GAP}}$ by TCPSP(a). The second step is to find a low cost solution of the resulting TCPSP, which can be done by employing existing methods, e.g. the method proposed in Guldemond et al. [1].

By the above construction we can rewrite the TCPSP with adjacent resources as

$$\min_{a\in F_{\mathrm{GAP}}} Opt(\mathrm{TCPSP}(a))$$

where $Opt(\cdot)$ denotes the optimal value of the resulting TCPSP.

In the following we first describe the GAP. We design the GAP in such a way that the original problem has a feasible solution if and only if the GAP has a feasible solution and such that each feasible solution of the GAP can be extended to a feasible solution of the original problem. Afterwards, we present an ILP formulation of the GAP which can be used to solve it. Finally, we treat the resulting TCPSP problem resulting after fixing the assignment of the adjacent resource units according to a solution of the GAP.

4.1. The group assignment problem (GAP)

The feasibility of the original problem depends on the adjacent resource capacity and requirements, and on the timing constraints of the jobs. It does not depend on the renewable resources, since we assume that we can hire unlimited additional capacity of the renewable resources. For each group G_g we have to determine an adjacent resource assignment and a start and completion time, denoted by s_g and c_g, respectively. The start and completion time of a group implies a time window $[s_g,c_g]$ in which it should be possible to process all jobs of group G_g, respecting the timing constraints of the jobs. Note that the processing time of a group is not a priori fixed.

The start and completion time of a group have to be consistent with the release dates and deadlines of the jobs in this group. This gives rise to the following definitions: earliest start time of a group $EST_g := \min_{J_j \in G_g}\{r_j\}$, latest start time of a group $LST_g := \min_{J_j \in G_g}\{d_j-p_j\}$, earliest completion time of a group $ECT_g := \max_{J_j \in G_g}\{r_j+p_j\}$, and latest completion time of a group $LCT_g := \max_{J_j \in G_g}\{d_j\}$. In order to guarantee processing of all jobs in a group, the start and completion of a group should be such that $s_g \in [EST_g,LST_g]$ and $c_g \in [ECT_g,LCT_g]$.

It is, however, not possible to choose s_g and c_g independently within the mentioned intervals. For example, if we choose the start time of a group G_g to be large, then choosing the completion time of group G_g equal to ECT_g might not be feasible. The reason for this is that there can be a path in the AoN network from J_g^s to J_g^c such that adding all processing times and time lags on this path exceeds the value c_g-s_g. This means that the time window $[s_g,c_g]$ is not large enough to process the jobs in G_g. We define therefore a minimum processing time $p_g^{\min}$ for group G_g. To obtain the value of $p_g^{\min}$, we schedule J_g^s as late as possible (i.e. set $s_g=LST_g$) and given this start time of J_g^s we schedule J_g^c as early as possible. The difference between these start and completion times gives $p_g^{\min}$. The renewable and adjacent resource capacity is not considered, only timing constraints play a role. Note that $p_g^{\min}$ is determined by some critical path in the AoN network or by the release dates and deadlines. Only in the first case the minimum processing time requirement forms an additional constraint.

The above constraints focus on a single group. But there are also restrictions between different groups. Whenever there is a path in the AoN network from J_g^s to J_h^c we call group G_g a predecessor of group G_h, and by $P'_h \subset \mathcal{G}$ we denote the set of predecessors of group G_h. Again, if the start of group G_g is chosen large, it might not be possible to schedule the completion of group G_h at ECT_h. This gives rise to the definition of start-completion time lags τ'_{gh} between groups, i.e. $s_g+\tau'_{gh} \le c_h$ for $G_g \in P'_h$. As before with the calculation of $p_g^{\min}$, to calculate τ'_{gh}, we schedule group G_g to start as late as possible, and given this start time, we schedule group G_h as early as possible. The difference between these values gives τ'_{gh}. (In particular $\tau'_{gg}=p_g^{\min}$.) Since we have start-completion time lags, it is possible that G_g is a predecessor of G_h and simultaneously that G_h is a predecessor of G_g. This implies that these groups have to be scheduled in parallel for some duration.

The group assignment problem (GAP) can now be formally stated as follows: *We are given a set $\mathcal{G}$ of groups, i.e. $\mathcal{G}=\{G_1,\ldots,G_m\}$, precedence relations among the groups, and an adjacent resource $\overline{R}$. The adjacent resource has a capacity of $\overline{Q}$ resource units. Each group $G_g \in \mathcal{G}$ has to be assigned to $\overline{q}_g$ adjacent resource units for its entire duration. Group G_g has to start between EST_g and LST_g, and to complete between ECT_g and LCT_g with a duration of at least $p_g^{\min}$. Whenever there is a precedence relation between two groups G_g and G_h, i.e. $G_g \in P'_h$, group G_g has to start at least τ'_{gh} time units before the completion of G_h. A solution of the GAP is an assignment of the groups to the adjacent resource units, and a schedule of the groups that respect the time windows and the precedence relations.*

The following theorem shows that the feasibility of the GAP and the original problem are equivalent.

Theorem 1. *The group assignment problem has a feasible solution if and only if the TCPSP with adjacent resources has a feasible solution.*

Proof. Suppose that the original problem has a feasible solution. This solution of the TCPSP with adjacent resources specifies the adjacent resource assignment of the groups and the start times of the jobs. From the start times of the jobs we can derive the start and completion times of the groups. By definition, these start and completion times satisfy all the time restrictions for the groups in the GAP. So, the GAP has a feasible solution.

Now suppose that the GAP has a feasible solution. From the GAP solution we have an adjacent resource assignment and start and completion times of the groups. Since the renewable resources can be hired, the feasibility of the original problem only depends on the timing constraints on the jobs. By definition of the time windows of the groups, their minimum duration, and the time lags it is ensured that feasible start times of jobs exists, i.e. by scheduling the jobs as early as possible. □

As mentioned in the previous section, the problem of finding a feasible solution for the overall problem is already *NP*-complete. Since this feasibility is equivalent to the feasibility of the GAP, the given decomposition has the advantage that the hard feasibility question is already treated at an early stage.

4.1.1. ILP formulation of the GAP

To find a solution for the GAP we model it as an ILP. For this, we define the following variables. The variable $a_g \in [0,\overline{Q}-\overline{q}_g]$ gives the adjacent resource assignment of group G_g, i.e. group G_g is assigned to the interval $[a_g,a_g+\overline{q}_g]$. Variables $s_g \in [EST_g,LST_g]$ and $c_g \in [ECT_g,LCT_g]$ are the start and completion time of group G_g as before. Finally, binary variables x_{gh}, y_{gh}, z_{gh} are used in the modeling to avoid overlap in the adjacent resource assignment.

The GAP is represented by the following set of constraints (directly followed by an explanation):

$$c_g-s_g \ge p_g^{\min} \quad \forall G_g \tag{1}$$

$$c_h-s_g \ge \tau'_{gh} \quad \forall G_g \in P'_h \tag{2}$$

$$\overline{Q}\cdot(z_{gh}+y_{gh}) \ge a_h+\overline{q}_h-a_g \quad \forall g<h \tag{3}$$

$$\overline{Q}\cdot(1+z_{gh}-y_{gh}) \ge a_g+\overline{q}_g-a_h \quad \forall g<h \tag{4}$$

$$T\cdot(1-z_{gh}+x_{gh}) \ge c_h-s_g \quad \forall g<h \tag{5}$$

$$T\cdot(2-z_{gh}-x_{gh}) \ge c_g-s_h \quad \forall g<h \tag{6}$$

The constraints (1) and (2) are clear from the definition of $p_g^{\min}$ and τ'_{gh}. What remains it to ensure that no two groups using a common adjacent resource unit, overlap in time. It is sufficient to check this for each index pair (g,h) with $g<h$. Whenever the groups overlap on the adjacent resource, the right hand sides of

(3) and (4) are both larger than 0, implying that $z_{gh}=1$. If they do not overlap, at least one of the right hand sides is at most 0. Due to the free choice for the variable y_{gh} the variable z_{gh} is now unrestricted. Whenever the groups overlap in time the right hand sides of (5) and (6) are both larger than 0, implying that $z_{gh}=0$. Again, due to the variable x_{gh} the variable z_{gh} is unrestricted otherwise. Thus, constraints (3)–(6) ensure that no two groups have a conflict, since it is impossible for groups G_g and G_h to overlap in time and on the adjacent resource simultaneously (the two possibilities lead to different values of z_{gh}).

It is not necessary to restrict the variables s_g, c_g, and a_g to be integer. If the obtained solution contain a non-integer value for one of these variables we can round the value down without violating the constraints, since all input parameters are integers. No objective function is needed since we are at this stage only looking for a feasible assignment.

4.1.2. Adding cutting planes

The ILP given by (1)–(6) gives a complete description of the GAP. By adding additional valid inequalities (cutting planes) we can reduce the computation time required to solve the ILP. We propose two types of cutting planes.

For a subset of groups that have a cumulative adjacent resource requirement more than $\overline{Q}$, we know that at least two group have a resource overlap in any solution. We call such a subset a *resource conflicting set*. A *minimum resource conflicting set* is a resource conflicting set that does not remain a resource conflicting set if any of the groups are removed from it. Let S be such a minimum resource conflicting set. Then we can add the following constraint to our model:

$$\sum_{G_g,G_h \in S, g<h} z_{gh} \geq 1 \quad \forall S \tag{7}$$

Similarly we can look at a *minimum time conflicting set*. When a set of groups have a cumulative minimum duration larger than T, we know that at least two groups share time units, and thus not adjacent resource units. This implies that not all z_{gh} values can be 1. Let S' be such a minimum time conflicting set. We can add

$$\sum_{G_g,G_h \in S', g<h} z_{gh} \leq \frac{1}{2}|S'|(|S'|-1)-1 \quad \forall S' \tag{8}$$

As we show in Section 5, adding constraints (7) and (8) to the ILP formulation (1)–(6) can significantly reduce the computational time needed to solve the GAP.

4.2. Deriving and solving the resulting TCPSP

In this section we treat the problem that remains after assigning the groups to the adjacent resource. We show how the solution of the GAP can be incorporated into the AoN network, such that the group structure and adjacent resource do not have to be considered anymore when searching for a low cost schedule for the jobs.

A solution of the GAP gives us for each group G_g a start time (s_g) and a completion time (c_g), and an adjacent resource assignment ($[a_g, a_g+\overline{q}_g]$). We could impose these start and completion times of a group on the jobs within that group by redefining the release dates and deadlines of the jobs, i.e. $\overline{r}_j := \max\{r_j, s_g\}$ and $\overline{d}_j := \min\{d_j, c_g\}$ for all $J_j \in G_g$. However, this would unnecessarily restrict the resulting TCPSP, i.e. hiring a lot of renewable resources might be unavoidable, since these aspects did not play a role in the GAP. It is sufficient to keep the order in which two groups G_g and G_h are assigned to the same adjacent resource unit. Thus, if groups G_g and G_h share an adjacent resource unit and group G_g completes before group G_h starts (all jobs of group G_g complete before any job of group G_h starts), we add a precedence relation from jobs J_g^c to J_h^s. In this way, different solutions for the resulting TCPSP may result in different start and completion times of the groups, however, the assignment of the groups to the adjacent resources remains valid to all these solutions. Therefore, adding precedence relations restricts the resulting TCPSP less than imposing the start and completion times from the GAP solution, and still guarantees feasibility of the adjacent resource assignment. Note that the release dates and deadlines of jobs may require adjustment such that they are consistent with the added precedence relations. The dummy jobs J_g^s and J_g^c now can be removed from the AoN network. When removing a dummy job all predecessors of the dummy job become predecessors of all the successors of the dummy job.

To find a solution of the resulting TCPSP any known method from the literature can be employed. We employ the method of Guldemond et al. [1] since it fits the resulting TCPSP best. The method is designed to deal with hiring of renewable resources and work in overtime. Since our model does not include working in overtime, the method boils down to the following two stage procedure. The first stage of the heuristic constructs schedules by means of randomized sampling. In this stage many schedules are constructed by adding jobs one by one into the schedule. Jobs can only be put into the schedule if all their predecessors have been scheduled. The selection of the job to be scheduled next is based on a randomization which is biased to jobs which get close to their deadline. The second stage consists of a neighborhood search. This neighborhood search repeatedly removes a small subset of the jobs from the schedule and reinsert them by means of an ILP solution. This last technique is based on the work of Palpant et al. [16].

4.3. Objective function

Up to now, we considered just an arbitrary feasible solution $a \in F_{\mathrm{GAP}}$ of the GAP. However, if we choose different feasible solutions of the GAP we obtain different resulting TCPSP's. The following example illustrates how different solutions of the GAP imply different precedence relations in the resulting TCPSP, and hence different solutions for the original problem.

Consider the example project as given in Section 2.3, but now with the duration of job J_4 reduced to 2, i.e. $p_4=2$. In Fig. 3 two different assignments of the adjacent resource together with the corresponding schedules of the renewable resources are given. Since a second painting crew has to be hired in time buckets 4 and 5 for Assignment 1 and not for Assignment 2, Assignment 2 results in a better solution of the resulting TCPSP. Therefore, it is a better solution of the original problem.

This raises the question whether it is possible to predict somehow the quality of the overall solution by assigning some quality measure to the assignments within the ILP. The above example illustrates that it may be beneficial to have long durations for the groups. An assignment of groups with long durations allows more flexibility in scheduling the jobs in the resulting TCPSP. One may expect to find better job schedules when there is more room for the jobs. To get a GAP solution with long group durations, we propose the following two objective functions for the GAP.

The first objective function simply maximizes the total absolute group duration:

$$\mathrm{ABS} := \text{maximize} \sum_{G_g \in \mathcal{G}} (c_g - s_g)$$

The second objective function maximizes the total group duration relative to the minimum duration of the group:

$$\mathrm{REL} := \text{maximize} \sum_{G_g \in \mathcal{G}} \left(\frac{c_g - s_g}{p_g^{\min}} \right)$$

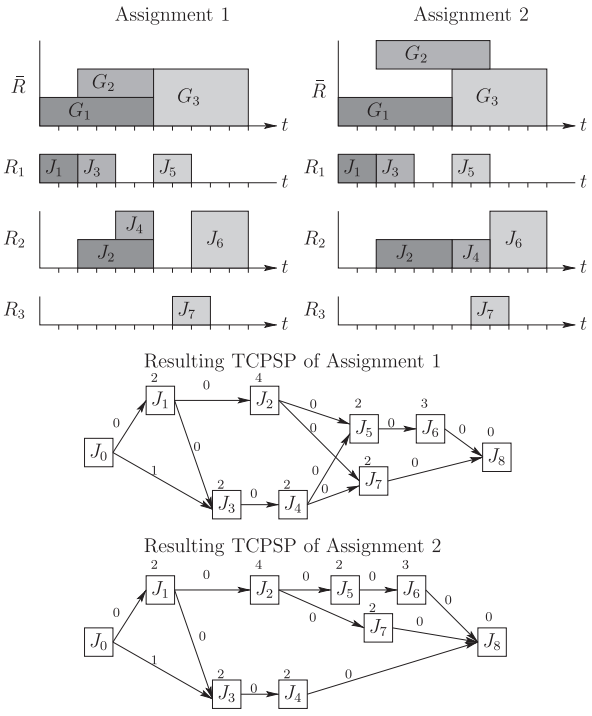

Fig. 3. Example project: two alternative assignments.

For the assignments given in Fig. 3 both measures have a preference to Assignment 2, i.e. for Assignment 1 we have ABS=6+4+5=15 and REL=$\frac{6}{6}+\frac{4}{4}+\frac{5}{5}=3$, and for Assignment 2 we have ABS=6+6+5=17 and REL=$\frac{6}{6}+\frac{6}{4}+\frac{5}{5}=3\frac{1}{2}$.

One may think of various other objective criteria for assignments. However, our goal is not to investigate many different objectives, but to see if the objectives influence the overall outcome and if one objective dominates the other. Therefore, in next section we study the effect of adding these objective functions to the GAP on the quality of the solution found for the original problem.

5. Computational tests

In this section we report on computational experiments for the presented decomposition approach. The aim of this section is to show that the presented decomposition approach gives a flexible approach to handle the TCPSP with adjacent resources. Since no solution methods for this problem are known and since also the possible relaxations of the problem (e.g. relaxing the adjacency constraints of the adjacent resources) do not lead to useful lower bounds, it is hard to judge the overall quality of the achieved solutions. However, we show that by using different objective functions for the GAP we obtain different solutions for the original problem. The test results show no dominance between the proposed GAP objective functions.

The computations are performed on a computer with a dual 3.40 GHz processor and 1.00 GB of RAM memory. The ILP's are solved with CPLEX version 11.1.

5.1. Generating instances

For the generation of the instances we make use of the project generator ProGen, described in Kolisch and Sprecher [17,18] and Kolisch et al. [19], which generates instances for the RCPSP. This generator uses various parameters to govern the properties of the precedence network and the resource requirements of the instances generated. In this work, we do not fully explore all possible parameter variations but restrict ourself to the parameter values that are mentioned in Kolisch et al. [19] as leading to instances that are interesting to study. Further below we mention the precise settings.

The instances of the TCPSP with one one-dimensional adjacent resource are generated in three steps. To construct an instance, we first generate a set of groups with their adjacent resource requirement and the precedence relations among them, and determine the capacity of the adjacent resource. In the second step, we generate a set of renewable resources, for each group a set of jobs with corresponding renewable resource requirements and precedence relations, and the renewable resource capacities. In the final step, we convert the precedence relations between groups into precedence relations between jobs of those groups. Whenever there is a precedence relation from group G_g to G_h we add a precedence relation from a randomly selected job from G_g to a randomly selected job from G_h.

We generate four different sets of instances that vary in the number of groups and the number of jobs per group. However, for all instances we let the total number of jobs be equal to 120. This allows us to somehow compare the computation times. In the four different sets of instances the number of groups are 8, 10, 12, and 15 with each group having 15, 12, 10 and 8 jobs, respectively. For each set of instances we generate 100 instances.

The parameters of ProGen are chosen as follows. The network complexity (the average number of non-redundant arcs per node) of the group and job networks is 1.5 with an allowed deviation of 0.3. Each group requires up to 10 units of the adjacent resource and the resource strength (a measure of the resource scarceness through a scaling in the convex combination of the minimum and maximum resource demand) of the adjacent resource is 0.2. The number of renewable resources varies between 5 and 10 and each job requires up to 5 of these resources for processing. The renewable resource factor (the average portion of resources required per job) is 0.5 and the renewable resource strength is 0.2. The processing times of the jobs are drawn uniformly from $\{1,\ldots,15\}$. For details on these measures we refer to Kolisch et al. [19].

In order to transform these instances to instances of the time-constrained project scheduling problem, we define the cost of hiring one additional renewable resource unit in one time bucket to be 1, i.e. $c_{kt}=1$ for all k,t. Furthermore, we define the release date of the project as 0 and derive a deadline that applies to all jobs. The deadline of an instance is determined in the same way as projects generated by ProGen have due dates. Let MP denote the minimum project length, which is defined by the longest path in the AoN network, and let $\overline{T}$ be the sum of all processing times. $\overline{T}$ is thereby an upper bound on the project length. Now we define the project deadline as $MP+\delta(\overline{T}-MP)$. The scaling parameter δ is called the due date factor. The smaller this value the smaller the deadline becomes, and as a consequence the resource requirements play a more important role. We vary the value of δ between 0.05 and 0.20. Varying δ while keeping the remaining data fixed, leads to different instances, which we treat as different versions of the same instance. Using these general release date and deadline, release dates and deadlines of the jobs are derived by making them consistent with the precedence relations.

The parameter values are chosen in this way to achieve instances that are useful for testing, that is, instances that are not easy to solve. By varying the due date factor we can move from sets with many instances having 0 cost solutions (large due date factor) to sets with many infeasible instances (small due date factor). Due to the adjacency requirements of the adjacent resource, there is no guarantee that there exist feasible solutions for the generated instances. However, infeasibility of an instance can already be determined after solving the GAP.

Table 2
Solving the GAP: eight groups.

Number of groups	Due date factor	GAP objective	Using CP1	Using CP2	Time	Number feasible	Time feasible	Number infeasible	Time infeasible	Number time exceeded
8	0.05	NO	TRUE	TRUE	0.01	1	0.02	99	0.01	0
			TRUE	FALSE	0.01	1	0.02	99	0.01	0
			FALSE	TRUE	24.07	1	0.00	98	6.20	1
			FALSE	FALSE	20.95	1	0.09	98	3.00	
		ABS	TRUE	TRUE	0.01	1	0.02	99	0.01	0
			TRUE	FALSE	0.01	1	0.02	99	0.01	0
			FALSE	TRUE	0.33	1	0.02	99	0.33	0
			FALSE	FALSE	0.17	1	0.03	99	0.17	0
		REL	TRUE	TRUE	0.01	1	0.02	99	0.01	0
			TRUE	FALSE	0.00	1	0.02	99	0.00	0
			FALSE	TRUE	0.16	1	0.02	99	0.16	0
			FALSE	FALSE	0.23	1	0.02	99	0.23	0
8	0.10	NO	TRUE	TRUE	25.78	50	0.15	49	15.72	1
			TRUE	FALSE	26.88	50	0.11	50	53.64	0
			FALSE	TRUE	317.51	49	48.70	36	65.67	15
			FALSE	FALSE	271.58	50	1.13	38	144.76	12
		ABS	TRUE	TRUE	0.41	50	0.08	50	0.74	0
			TRUE	FALSE	0.54	50	0.08	50	1.00	0
			FALSE	TRUE	3.48	50	1.05	50	5.90	0
			FALSE	FALSE	3.45	50	0.25	50	6.64	0
		REL	TRUE	TRUE	0.35	50	0.10	50	0.59	0
			TRUE	FALSE	0.46	50	0.08	50	0.83	0
			FALSE	TRUE	6.21	50	0.55	50	11.88	0
			FALSE	FALSE	12.07	50	0.20	50	23.94	0
8	0.15	NO	TRUE	TRUE	18.32	95	0.33	4	0.01	1
			TRUE	FALSE	3.63	95	0.05	5	71.60	0
			FALSE	TRUE	29.26	95	11.85	4	0.02	1
			FALSE	FALSE	18.51	95	0.54	4	0.01	1
		ABS	TRUE	TRUE	0.07	95	0.06	5	0.33	0
			TRUE	FALSE	0.08	95	0.06	5	0.48	0
			FALSE	TRUE	0.55	95	0.33	5	4.82	0
			FALSE	FALSE	0.35	95	0.13	5	4.52	0
		REL	TRUE	TRUE	0.14	95	0.06	5	1.78	0
			TRUE	FALSE	0.13	95	0.06	5	1.45	0
			FALSE	TRUE	0.69	95	0.32	5	7.82	0
			FALSE	FALSE	2.21	95	0.15	5	41.43	0
8	0.20	NO	TRUE	TRUE	0.02	99	0.02	1	0.09	0
			TRUE	FALSE	0.02	99	0.02	1	0.13	0
			FALSE	TRUE	0.05	99	0.04	1	1.11	0
			FALSE	FALSE	0.03	99	0.03	1	0.34	0
		ABS	TRUE	TRUE	0.02	99	0.02	1	0.06	0
			TRUE	FALSE	0.02	99	0.02	1	0.05	0
			FALSE	TRUE	0.05	99	0.05	1	0.42	0
			FALSE	FALSE	0.03	99	0.03	1	0.16	0
		REL	TRUE	TRUE	0.02	99	0.02	1	0.06	0
			TRUE	FALSE	0.02	99	0.02	1	0.05	0
			FALSE	TRUE	0.06	99	0.06	1	0.58	0
			FALSE	FALSE	0.03	99	0.03	1	0.11	0

5.2. Solving the GAP

In the first series of tests we concentrate on the group assignment problem (GAP) only. For the four sets of instances (100 instances with 8, 10, 12, and 15 groups, respectively), we explore the effect of varying the due date factor on the number of feasible solutions, and the effect of using cutting planes in combination with the proposed objective functions on the computation time.

For each of the 400 instances we solve the GAP 48 times; four different due date factors, three different objective functions (NO

Table 3
Solving the TCPSP: comparing objectives.

Number of groups	GAP objective	Number feasible	Average time GAP (feasible)	Average Time resulting TCPSP	Average TCPSP objective value	Average value best found solution
8	NO	95	0.05	42.42	54.43	37.6
	ABS	95	0.05	40.59	57.28	
	REL	95	0.06	40.92	56.85	
10	NO	100	8.30	36.53	56.17	29.99
	ABS	100	0.07	35.37	50.21	
	REL	100	0.08	36.89	43.27	
12	NO	100	0.26	34.05	53.79	33.23
	ABS	100	0.17	32.51	54.34	
	REL	100	0.19	33.18	53.41	
15	NO	99 (1 time exceeded)	1.89	30.51	60.04	32.64
	ABS	100	0.84	30.67	60.87	
	REL	100	0.81	30.57	53.65	

Table 4
Solving the TCPSP: times best.

8 groups			10 groups		
NO	ABS	REL	NO	ABS	REL
62	58	62	34	47	44
12 groups			15 groups		
NO	ABS	REL	NO	ABS	REL
39	32	44	39	34	37

meaning without objective, ABS and REL) each with four different settings of the cutting planes. For these computations we have set a time limit of 1800 s on the computation time spent on one instance. Table 2 summarizes the results for the instances with eight groups per instance. The table displays the number of feasible and infeasible instances and the number of instances for which the time limit of 1800 s is exceeded. Additionally, it contains the average computation time per instance, and, more specific, also the average computation time for the feasible and infeasible instances separately.

As mentioned earlier and now demonstrated by the computational tests, a decrease of the due date factor leads to a growing fraction of infeasible instances in the set. By increasing the due date factor from 0.05 to 0.10, the number of feasible instances grows a lot, but still half of the instances are infeasible in the set with 8 groups in each instance. Further increasing the due date factor to 0.15 seems to lead to interesting instances for testing the resulting TCPSP, i.e. we expect many of the instances to be feasible but also that hardly any of the instances has a feasible job schedule that does not hire additional renewable resources.

Regarding the cutting planes we can draw a firm conclusion. It is best to employ the cutting planes of (7), called CP1, and not the cutting planes of (8), called CP2. The improvement obtained by adding CP1 is particularly large for the computation time spent on the infeasible instances.

We have tested instances with up to 15 groups. The results are similar to the ones in Table 2. Obviously, the computation time increases as the number of groups increases. Tests have shown that the computation time spent becomes large, i.e. the time limit of 1800 s is exceeded more often, but by tuning CPLEX this can be dealt with. We point out that for practical applications 15 groups are already quite a lot. We end this section by a note on the use of CPLEX.

Note. It is surprising to see that CPLEX 11.1 is able to determine the feasibility of an instance faster if there is an objective function in the GAP. CPLEX's search strategies are influenced by the objective function.

For a subset of instances that exceed the time limit we have tested different settings of CPLEX 11.1. For each of these instances we are able to reduce the required computation time to a fraction of a second by changing the settings. However, the optimal setting differs from instance to instance. We use therefore the default settings for all computations.

5.3. Solving the resulting TCPSP

In the second series of tests we compare the cost of the resulting TCPSP when different objective function are used in the GAP. In these tests we use a due date factor of 0.15 and use only the cutting planes of (7), that is CP1, as motivated in the previous section. For each of the 100 instances of the four sets, we solve the GAP and the resulting TCPSP 3 times, once without an objective for the GAP and next with the two proposed objectives. As a consequence of the chosen cost for hiring extra renewable resources, the objective value of the resulting TCPSP equals the amount of resource units hired in total. Solving the resulting TCPSP takes on average 35 s and never more than 3 min.

Table 3 displays the results of these tests. Note that only one instance in the set of 15 groups exceeds the time limit of 1800 s in the GAP. When we consider the cost of the TCPSP solution, we see that all GAP objectives result in about the same average values (second to last column).

To see whether the different GAP solutions found by the three objectives result in large differences, we can compare them with the overall best found solution. The rightmost column of Table 3 gives the average values of these best found solution. There is a significant difference between the average best found solution and the average solution found by using just one specific GAP objective. So, the overall solution found heavily depends on the GAP solution found.

In Table 4 we count the number of times that the best found solution is due to either the use of no objective, the ABS objective or the REL objective. The results show that the use of either

objective is equally good, the number of times they lead to the best found solution is about the same. So, no choice of objective function dominates the other.

The above results show that it is hard to predict just on the basis of the GAP solution, what the objective value of the resulting TCPSP will be. Therefore, it is worth to generate not only one solution for the GAP, but also to generate a number of different GAP solutions by using different objectives, and solve for each of them the corresponding TCPSP.

6. Concluding remarks

We presented a decomposition approach for the time-constrained project scheduling Problem with adjacent resources, which focuses in an early stage on the issue of a feasible assignment of groups to adjacent resource units. This is important since the NP-completeness of the feasibility problem forms the main obstacle to develop fast heuristic solution approaches for the overall problem. The presented approach detects infeasibility of an instance of the TCPSP with adjacent resources by solving the corresponding GAP in the first step. In case an instance is feasible, the first step gives also a solution for the group assignment and the order of the groups on the adjacent resources, which can be extended to an overall feasible solution in the second step. The test results show no clear dominance among the presented GAP objective functions. Finding good solutions by one specific objective remains problematic, but by solving each instance multiple times with different GAP objective functions, the quality of the generated schedules improves significantly.

The presented method can easily be extended to include two-dimensional adjacent resources and multiple adjacent resources, by modifying the ILP formulation. However, the computational time required to solve the GAP's will become a bigger issue.

For future research it would be interesting to see whether other GAP objectives are more successful. In particular, one might choose an objective function that is more dependent on the job characteristics. However, preliminary tests have shown that using weighted versions of the presented objectives does not improve the results (the weights of a group corresponds to the renewable resource requirements of the jobs in that group). Besides exploring a fixed objective, the presented decomposition can be the basis of a feedback between the GAP and the resulting TCPSP, where the outcome of the resulting TCPSP can influence the GAP objective before resolving. This may lead to a local search approach, where the weights and the different type of objectives of the GAP can be used as a solution space. Adapting the weights can be seen as some sort of intensification phase and the change of the objective as some sort of diversification phase of the search process. To make such an approach successful, intelligent ways of changing the weights based on the outcome of the TCPSP have to be developed.

References

[1] Guldemond TA, Hurink JL, Paulus JJ, Schutten JMJ. Time-constrained project scheduling. Journal of Scheduling 2008;11(2):137–48.
[2] De Boer R. Resource-constrained multi-project management, a hierarchical decision support system. PhD thesis, University of Twente; 1998.
[3] Hess K, Kolisch R. Efficient methods for scheduling make-to-order assemblies under resource, assembly area and part availability constraints. International Journal of Production Research 2000;38(1):207–28.
[4] Kolisch R. Integrated scheduling, assembly area- and part-assignment for large scale make-to-order assemblies. International Journal of Production Economics 2000;64:127–41.
[5] Beck JC. Heuristics for scheduling with inventory: dynamic focus via constraint criticality. Journal of Scheduling 2002;5(1):43–69.
[6] Neumann K, Schwindt C. Project scheduling with inventory constraints. Mathematical Methods of Operations Research 2002;56(3):513–33.
[7] Neumann K, Schwindt C, Zimmermann J. Project scheduling with time windows and scarce resources. Lecture notes in economics and mathematical systems, vol. 508. Springer; 2002.
[8] Guan Y, Xiao W-Q, Cheung RK, Li C-L. A multiprocessor task scheduling model for berth allocation: heuristic and worst-case analysis. Operations Research Letters 2002;30(5):343–50.
[9] Lim A. The berth planning problem. Operations Research Letters 1998;22(2–3):105–10.
[10] Fekete SP, Köhler E, Teich J. Higher-dimensional packing with order constraints. SIAM Journal on Discrete Mathematics 2006;20:1056–78.
[11] Steiger C, Walder H, Platzner M. Operating systems for reconfigurable embedded platforms: online scheduling of real-time tasks. IEEE Transactions on Computers 2004;53(11):1393–407.
[12] Duin CW, Van der Sluis E. On the complexity of adjacent resource scheduling. Journal of Scheduling 2006;9(1):49–62.
[13] Hartmann S. Packing problems and project scheduling models: an integrative perspective. Journal of the Operational Research Society 2000;51(9):1083–92.
[14] Möhring RH. Minimizing costs of resource requirements in project networks subject to a fixed completion time. Operations Research 1984;32(1):89–120.
[15] Deckro RF, Herbert JE. Resource constrained project crashing. OMEGA International Journal of Management Science 1989;17(1):69–79.
[16] Palpant M, Artigues C, Michelon P. LSSPER: solving the resource-constrained project scheduling problem with large neighbourhood search. Annals of Operations Research 2004;131(1):237–57.
[17] Kolisch R, Sprecher A. Project scheduling library—PSPlib 〈http://129.187.106.231/psplib/〉; 1997.
[18] Kolisch R, Sprecher A. PSPLIB a project scheduling problem library. European Journal of Operational Research 1997;96(1):205–16.
[19] Kolisch R, Sprecher A, Drexl A. Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science 1995;41(10):1693–703.