Decomposition Method for Project Scheduling with Spatial Resources

J.L. Hurink, A.L. Kok, J.J. Paulus

University of Twente, PO box 217, 7500 AE Enschede, The Netherlands

Abstract

Project scheduling problems are in many practical instances restricted by a limited availability of spatial resources. In this paper we develop a decomposition method for the Time-Constrained Project Scheduling Problem (TCPSP) with Spatial Resources. Spatial resources are resources that are not required by single activities, but by groups of activities. As soon as an activity of such a group starts, the spatial resource units are occupied, and they are not released before all activities of the that group are completed. On top of that, the spatial resource units that are assigned to a group have to be adjacent. [**Tests, Results**]

1 Introduction

In many practical instances of project scheduling, there is a limited amount of physical space available. For example, one has to deal with dry docks, shop floor spaces or assembly areas. In this paper we discuss the complications occurring in project scheduling problems where such spatial resources play a role, and present a decomposition method that separates the spatial resource assignment from the scheduling of the other resources.

Project Scheduling Problems are comprehensive descriptions of scheduling problems from practice. From the literature two variants can be distinguished. The Resource-Constrained Project Scheduling Problem (RCPSP) aims to schedule the activities such that the makespan is minimized without exceeding the resource capacities. There is a large amount of literature available on the RCPSP. For an overview see Herroelen et al. [6], Kolisch and Harmann [9], and Kolisch and Padman [10]. The other variant is the Time-Constrained Project Scheduling Problem (TCPSP), in which strict deadlines on the activities have to be met, but resource levels may be exceeded, or work in overtime is allowed, see Guldemond et al. [5]. The objective is to minimize the usage of these additional resources.

In the context of project scheduling problems, De Boer [1] introduced the concept of spatial resources through activity groups. A first characteristic is, that the spatial resource is not required by a single activity but by a group of activities (*activity group* or simply *group*). Second, a spatial resource is characterized by an adjacency requirement. The units of a spatial resource that are assigned to a group have to be adjacent. The spatial resource units are occupied from the first moment an activity of such a group starts until the last activity in the group finishes. Whether activities within a group can be processed simultaneously depends on the availability of the other resources and precedence relations.

The concept of spatial resources is important since spatial resources occur often in practice. One example is given by the ship building industry. The dry docks are the spatial resources and all activities involved in building one ship form a group. As soon as we start building a ship, a certain length of the dry dock is required. This part of the dry dock is occupied until the whole ship is finished, i.e., all activities of the group are completed. Other examples can be found in berth allocation in a container terminal, see Guan et al. [4] and Lim [14], shop floor spaces and assembly areas, see Hess and Kolisch [7] and Kolisch [8], reconfigurable computing, see Fekete et al. [3] and Steiger et al. [15], and check-in desks at airports, see Duin and Van Sluis [2]. The literature on the examples mentioned above, only consider special cases, in which spatial resources are considered exclusively and groups consist of only one activity. To the best of our knowledge there does not exists any literature on the general problem of project scheduling with spatial resources.

Besides the connection with scheduling, there is a connection with packing problems. However, since the problem is set in a scheduling background we have to consider things as precedence relations, release dates and deadlines. Therefore, general packing techniques cannot be applied. Hardly any literature concerns ordered packing, [3] is an exception.

The rest of the paper is organized as follows. Section 2 gives a detailed problem description of the Time-Constrained Project Scheduling Problem (TCPSP) with spatial resources, and it explains why existing solution methods fail with the addition of spatial resources. To solve the TCPSP with spatial resources, we develop in Section 3 a decomposition method. The decomposition leads to a group assignment problem (GAP), for which an ILP formulation is given. After solving the GAP, we are left with an ordinary TCPSP. For the GAP an objective function has to be chosen such that the resulting TCPSP allows for high quality solutions. Section 4 presents different possible objective functions for the GAP. Section 5 concerns the computational tests, in which a comparison is made between the quality of the solutions of the resulting TCPSP, when different objective functions for the GAP are used. Section 6 concludes this paper.

2 The Project Scheduling Problem with Spatial Resources

In this section, we start by giving a detailed description of the time-constrained project scheduling problem with spatial resources (TCPSP with spatial resources). Hereby, we only consider 1-dimensional spaces. At the end of this section, we discuss why existing solution methods for project scheduling problems fail on the extension with spatial resources, motivating the decomposition approach.

We are given a set $\mathcal{A}$ of n activities, i.e., $\mathcal{A} = \{A_1, \ldots, A_n\}$. Each activity A_j has a release date r_j and a deadline d_j, and has to be scheduled without preemption between r_j and d_j for a duration p_j into the time horizon $[0, T]$. This time horizon is divided into T time buckets, $t = 0, ..., T-1$, where time bucket t represents the time interval $[t, t+1]$. Thus, for activity A_j time bucket r_j is the first available, and $d_j - 1$ the last. We assume the time windows for each activity to be large enough, i.e., $d_j - r_j \geq p_j$. For the processing of the activities there is a set $\mathcal{R}$ of renewable resources, $\mathcal{R} = \{R_1, \ldots, R_K\}$ and a set $\mathcal{S}$ of spatial resources, $\mathcal{S} = \{S_1, \ldots, S_L\}$, available. Each renewable resource $R_\kappa \in \mathcal{R}$ has a capacity $K_{\kappa,t}$ in time bucket t, and each spatial resource $S_\lambda \in \mathcal{S}$ has capacity L_λ. The processing of the activities is restricted by precedence relations, which are given by sets $P_j \subset \mathcal{A}$, denoting all direct predecessors of activity A_j. With each precedence relation there is associated a non-negative time lag τ_{ij} indicating that there are at least τ_{ij} time buckets between the completion of activity A_i and the start of activity A_j. We assume that all release dates and deadlines of the activities are consistent with the precedence relations**, see [cite]?**. Activity A_i has a resource requirements $q_{i\kappa}$ for renewable resource R_κ during its processing.

Additionally, we are give a set $\mathcal{G}$ of m groups, i.e. $\mathcal{G} = \{G_1, \ldots, G_m\}$. Each group $G_g \in \mathcal{G}$ is a subset of the activities ($G_g \subset \mathcal{A}$) and groups are pairwise disjunct. Group G_g has a spatial resource requirement of l_g adjacent units of the spatial resource $\sigma(G_g) \in \mathcal{S}$, from the start of the first activity in G_g until the completion of the last activity in G_g.

In the considered model the spatial resource have fixed capacities which cannot be extended. However, we do allow an increase of the capacity of the renewable resources. Increasing the capacity of renewable resource R_κ in time bucket t by one unit, incurs a cost of $c_{\kappa t}$. The objective is to find a feasible assignment of groups to the spatial resource units, and at the same time a feasible schedule of activities on the renewable resources, such that the total costs of increasing the capacity of the renewable resources is minimized.

The fact that all time windows are large enough, implies that there exist feasible solutions if the spatial resources are relaxed, since we can increase the capacity levels of the renewable resources. The problem described is NP-hard since it contains both the traditional TCPSP [5], see Guldemond et al. [5], and strip packing, **see [..]** as special cases. Due to the spatial resources, which have fixed capacity and require an adjacent assignment, determining whether there exists a feasible schedule or not is NP-hard. It reduces to bin packing **[?, see Papadimitriou]**.

The TCPSP with spatial resources can be represented by an activity on node network, see Figure 1. Each node corresponds to an activity and each arc to a precedence relation. The activities within one circle are members of one and the same group.

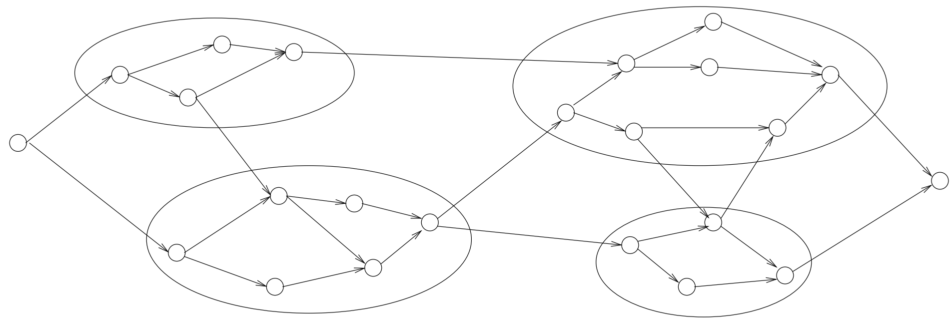

Fig. 1. Activity on node representation of the TCPSP with spatial resources.

If we remove the spatial resources, and thereby also the notion of groups, we have exactly the Time-Constrained Project Scheduling Problem. Guldemond et al. [5] give a heuristic solution method for this problem.

Most of the existing methods for project scheduling problems are constructive heuristics. However, the specific assignment of the groups to the spatial resource units cannot be included by a simple placement rule, due to the adjacency requirement. Assigning the groups to the spatial resource units in a sequential manner is very unlikely to give a feasible solution. There are two reasons why such a sequential assignment does not work. Firstly, the completion time of a group is unknown until all activities are scheduled. Until then it is not known when the spatial resource units will be available again. Secondly, without looking ahead one might assign a group such that the spatial resource, at a later point in time, becomes only occupied in the middle. The space is divided into two parts, such that groups with large spatial requirement cannot fit, delaying the assignment of these large groups. Therefore, a different approach is needed when spatial resources are involved.

In the next section we present a decomposition approach that first solves the groups to spatial resource assignment, such that the resulting problem is a ordinary TCPSP for which solution methods exist.

3 Decomposition Approach

In this section, we present a decomposition approach for the TCPSP with spatial resources (which we refer to as the *original problem*) that separates the spatial resource assignment from the rest of the scheduling problem. We are going to determine an assignment of the groups to the spatial resource units,

and an ordering between the groups assigned to the same spatial resource units. We call this the Group Assignment Problem (GAP). The ordering is going to imply additional precedence relations in the original problem, such that the spatial resources do not have to be considered anymore. If groups G_g and G_h share a spatial resource unit, and G_g completes before G_h starts all activities in G_g become predecessors of each activity in G_h. We are left with an ordinary TCPSP, which we refer to as the *resulting TCPSP*. This decomposition allows the use of existing methods to solve the resulting TCPSP.
We define the GAP such that feasibility of the original problem only depends on feasibility of the GAP. The GAP has a solution if and only if the original problem has one. The solution of the GAP implies additional precedence relations, restricting the solution space of the resulting TCPSP. Therefore, the assignment of the groups to the spatial resource units should anticipate these implications of the additional precedence relations. Do do this through the objective of the GAP.
In the following, we state the Group Assignment Problem (GAP), followed by an explanation of the parameters, why they are necessary and how to derive them from the original problem.
We are given a set $\mathcal{G}$ of groups, $\mathcal{G} = \{G1, ..., G_m\}$ and a set $\mathcal{S}$ of spatial resources, $\mathcal{S} = \{S_1, ..., S_L\}$. A spatial resource S_λ has a capacity of L_λ spatial resource units. Each group $G_g \in \mathcal{G}$ has to be scheduled on l_g adjacent spatial resource units of $\sigma(g) \in \mathcal{S}$, for a duration of at least d_g^{min}. Group G_g has to start between its earliest start time (EST_g) and latest start time (LST_g), and complete between its earliest completion time (ECT_g) and latest completion time (LCT_g). Whenever there is a precedence relation between two groups G_g and G_h, i.e., $G_g \in P_h$, there is a positive time lag τ_{gh}. This time lag implies that the start of group G_g is at least τ_{gh} time units before the completion of group G_h. A solution of the GAP is an assignment of the groups to the spatial resources, and a schedule of the groups that respect the time windows and the precedence constrained.
The parameters of the GAP have to be defined such that the GAP is feasible if and only if the TCPSP with spatial resource is feasible, and if a feasible solution exist, the resulting TCPSP has a feasible solution. The minimum duration d_g^{min} of a group G_g has to be chosen such that it is long enough to accommodate all the activities of group G_g, if we relax the availability of the renewable resources. Each pair of activities (A_i, A_j) in group G_g that is connected by a directed path from A_i to A_j in the activity on node network, impose a lower bound on the duration of the group G_g. This lower bound equals the minimum time difference between the start of activity A_i and the completion of activity A_j, satisfying the release dates, deadlines and time lags. We define d_g^{min} to be equal to the maximum of all these lower bounds. If a group is scheduled for less than this minimum duration we lose feasibility of the resulting TCPSP.
The time lag τ_{gh} between two groups G_g and G_h can be derived similarly. Whenever there is a directed path from an activity A_i in group G_g to an ac-

tivity A_j in group G_h in the activity on node network, we can calculate the minimum difference between the start of activity A_i and the completion time of activity A_j. The time lag τ_{gh} is now the maximum of all these minimum differences. Note that the precedence relation is on a start and a completion time. A precedence relation states that G_g start at least τ_{gh} time units before G_h completes. Again, if any of these precedence relations is violated we lose feasibility in the resulting scheduling problem.
The release dates and deadlines of the activities restrict the start and completion times for the groups. Since we assumed the release dates and deadlines to be consistent with the precedence relations of the activities, the earliest start time of group G_g (EST_g) equals the minimum release date of the activities in G_g. The latest start time of group G_g (LST_g) equals the minimum of the deadline minus the processing time of the activities, i.e., $LST_g = \min_{A_i \in G_g} d_i - p_i$. The latest completion time of group G_g (LCT_g) equals the maximum deadline of the activities in G_g, and the earliest completion time of group G_g (ECT_g) equals the maximum of the release dates plus the processing time of the activities, i.e., $ECT_g = \max_{i \in g} r_i + p_i$.
The GAP is NP-hard since it contains 2-dimensional bin packing as a special case. Nevertheless, we present an ILP formulation of the problem and use this to solve it to optimality, since the number of groups is very limited in most practical applications.
To model the GAP as an ILP, we employ in the dimension time and space, a start and completion variable for each group. Variables s_g^{time} (s_g^{space}) and c_g^{time} (c_g^{space}) define the start and completion of group G_g in the dimension time (space). Feasible placement of the groups is controlled by the binary variables w_{gh}, v_{gh}, y_{gh} and x_{gh}. If groups G_g and G_h overlap in space, variable w_{gh} equals 1 and if groups G_g and G_h overlap in time v_{gh} equals 1. To get a feasible placement for any two groups G_g and G_h, not both w_{gh} and v_{gh} can be 1. Variables y_{gh} and x_{gh} have no interpretation, but only serve the modelling.
Now the GAP can be represented by the following ILP (directly followed by an explanation of the constraints):

$$maximize: \quad f(s_g^{time}, c_g^{time}) \tag{1}$$

subject to:

$$\begin{aligned}
c_g^{time} - s_g^{time} &\geq d_g^{min} && \forall G_g \in \mathcal{G} && (2)\\
c_h^{time} - s_g^{time} &\geq \tau_{gh} && \forall G_h \in \mathcal{G}, G_g \in P_h && (3)\\
c_g^{space} - s_g^{space} &= l_g && \forall G_g \in \mathcal{G} && (4)\\
L_{\sigma(G_g)} \cdot (w_{gh} + y_{gh}) &\geq c_h^{space} - s_g^{space} && \forall G_g, G_h \in \mathcal{G}, \sigma(G_g) = \sigma(G_h) && (5)\\
L_{\sigma(G_g)} \cdot (1 + w_{gh} - y_{gh}) &\geq c_g^{space} - s_h^{space} && \forall G_g, G_h \in \mathcal{G}, \sigma(G_g) = \sigma(G_h) && (6)\\
T \cdot (v_{gh} + x_{gh}) &\geq c_h^{time} - s_g^{time} && \forall G_g, G_h \in \mathcal{G}, \sigma(G_g) = \sigma(G_h) && (7)\\
T \cdot (1 + v_{gh} - x_{gh}) &\geq c_g^{time} - s_h^{time} && \forall G_g, G_h \in \mathcal{G}, \sigma(G_g) = \sigma(G_h) && (8)\\
w_{gh} + v_{gh} &\leq 1 && \forall G_g, G_h \in \mathcal{G}, \sigma(G_g) = \sigma(G_h) && (9)\\
s_g^{time} &\in [EST_g, LST_g] && \forall G_g \in \mathcal{G} && (10)\\
c_g^{time} &\in [ECT_g, LCT_g] && \forall G_g \in \mathcal{G} && (11)\\
s_g^{space} &\in [0, L_{\sigma(G_g)} - l_g] && \forall G_g \in \mathcal{G} && (12)\\
c_g^{space} &\in [l_g, L_{\sigma(G_g)}] && \forall G_g \in \mathcal{G} && (13)\\
w_{gh}, v_{gh}, y_{gh}, x_{gh} &\in \{0, 1\} && \forall G_g, G_h \in \mathcal{G}, \sigma(G_g) = \sigma(G_h) && (14)
\end{aligned}$$

The objective function is not part of the GAP, but is used later, to ensure that the resulting TCPSP has a good structure. Possible options for the objective function (1) are discussed in the next section. Constraint (2) ensures that each group is scheduled for at least its minimum duration. Due to constraint (3), the time lags between the groups are satisfied. Constraint (4) defines the spatial requirement of the group. Constraints (5) to (9) manage the feasibility of the placement of the groups. Whenever two groups make use of the same spatial resource units, both right hand sides of (5) and (6) are strictly positive, and independently of the value of y_{gh} the value of w_{gh} has to be equal to 1. Note that the right hand side is at most $L_{\sigma(G_g)}$. Whenever two groups do not make use of the same spatial resource units, exactly one of the right hand sides of (5) and (6) will be strictly positive and one negative. So with the right choice of y_{gh}, we can set $w_{gh} = 0$. Similarly, constraints (7) and (8) imply that if groups G_g and G_h overlap in time the variable v_{gh} equals 1, and is unrestricted otherwise. Through constraint (9) not both v_{gh} and w_{gh} can equal 1, implying that no two groups overlap in time and space simultaneously. Thus, we have a feasible assignment. Finally, constraints (10) to (14) define the domain of the variables.

With this ILP formulation we can obtain a solution for the GAP. Having a solution of the GAP gives us for each group a start and a completion time, and the spatial resource assignment. One way to combine this solution with the resulting scheduling problem would be to impose the start and completion times of a group on the activities within that group. However, this would unnecessarily restrict the resulting scheduling problem. We only look at the order in which two groups G_g and G_h are assigned when they share a spatial resource unit. If group G_g completes before group G_h starts, all activities of group G_g complete before any activity of group G_h starts. This only implies precedence relations in the resulting scheduling problem, thereby restricting it less. More precisely, if groups G_g and G_h share a spatial resource unit and G_g completes before G_h starts, then all activities in group G_g become predecessors of each activity in group G_h.

4 Objective functions for the GAP

To be able to use the ILP formulation, presented in the previous section, we need an objective function. As stated before, the assignment of the groups should be such that the resulting TCPSP allows a high quality solution. In this section we present 5 different types of objective functions for the GAP. In each of these objective functions there is a weight W_g associated with each group G_g. The weight of a group denotes the importance of having a long duration for this group. We consider 3 different types of weights, so combined this gives a total of 15 different objective functions. We compare these objective functions in the next section.
First the different types of objective functions are discussed and second the different type of weights. The different types of objective functions are:

- maximize total weighted duration,
- maximize the minimum weighted flexibility on each spatial resource,
- minimize the weighted shortage of processing time,
- minimize the total weighted conflict,
- minimize the maximum weighted conflict on each spatial resource.

To have flexibility in scheduling an activity, the duration for which its group is scheduled should be large. Flexibility is most imported when there is a high requirement for (a particular) renewable resource by the activities within one group. So the most intuitive objective function for the GAP is *maximize total weighted duration*, given by (15).

$$maximize: \quad \sum_{G_g \in \mathcal{G}} W_g \cdot (c_g^{time} - s_g^{time}) \tag{15}$$

The objective *maximize total weighted duration* (15) has the disadvantage that a group with a high weight can dominate other groups. As a result, groups with a slightly lower weight might get scheduled for their minimum duration. To overcome this problem we propose a second objective function. By expressing flexibility as $\frac{c_g^{time} - s_g^{time}}{W_g}$, we get the objective function *maximize the minimum flexibility on each spatial resource*, given by (16). Each group will have at least some flexibility relative to its weight.

$$maximize: \quad \sum_{S_\lambda \in \mathcal{S}} \min_{G_g | \sigma(G_g) = S_\lambda} \left(\frac{c_g^{time} - s_g^{time}}{W_g} \right) \tag{16}$$

A second way to prevent that some groups get large durations, and thereby dominating the other groups, is to compare its scheduled duration with the total processing time of the activities within the group. Scheduling a group for a larger duration than the sum of the processing times of its activities is often unnecessary. The objective function *minimize the weighted shortage of processing time*, given by (17), is therefore the third type of objective function.

$$minimize: \quad W_g \cdot \frac{\left| \sum_{A_i \in G_g} p_i - (c_g^{time} - s_g^{time}) \right|}{\sum_{A_i \in G_g} p_i} \tag{17}$$

Whenever two groups are allocated to the same spatial resource units we call them *conflicting*. Since conflicting groups imply precedence relations in the resulting TCPSP, the execution of these groups are related. When If groups G_g and G_h are conflicting, and we take more time for the activities in group G_g, when solving the resulting TCPSP, it reduces the time for scheduling the activities of group G_h, and vice versa, see Figure 2.

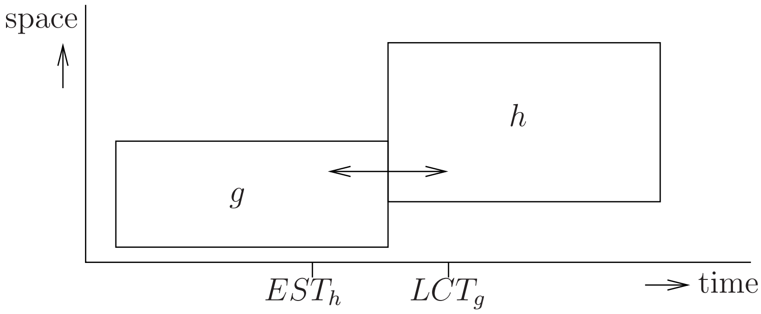

Fig. 2. Two groups in conflict.

We express the conflict between groups G_g and G_h by p_{gh}. The duration of group G_g is bounded by the difference between the latest completion time (LCT_g) and the earliest start time (EST_g). The conflict is measured as the scheduled duration relative to the longest possible duration. Again the W_g is indicating the importance of a long duration for group G_g. So, for p_{gh} the following should hold:

$$p_{gh} \geq \frac{(LCT_g - EST_g) \cdot w_{gh} - (c_g^{time} - s_g^{time})}{LCT_g - EST_g} \cdot W_g \quad \forall G_g, G_h \in \mathcal{G}, \sigma(G_g) = \sigma(G_h), \tag{18}$$

$$p_{gh} \geq 0 \quad \forall G_g, G_h \in \mathcal{G}. \tag{19}$$

Whenever there is a conflict ($w_{gh} = 1$), the right hand side of (18) is positive. And if $w_{gh} = 0$, then p_{gh} is only restricted by (19).

The last two types of objective functions for the GAP become *minimize the total conflict* (20) and *minimize the maximum conflict on each spatial resource* (21). When using these objective functions, constraints (18) and (19) have to be added to the ILP-formulation.

$$minimize: \quad \sum_{G_g, G_h \in \mathcal{G} | \sigma(G_g) = \sigma(G_h)} p_{gh} \tag{20}$$

$$minimize: \quad \sum_{S_\lambda \in \mathcal{S}} \max_{G_g, G_h \in \mathcal{G} | \sigma(G_g) = \sigma(G_h) = S_\lambda} p_{gh} \tag{21}$$

The weights W_g can be determined in numerous ways. We present three of them. The first is simply putting all weights equal to 1. This is useful in the comparisons later, to determine the effect of weights.

$$W_g = 1, \quad \forall G_g \in \mathcal{G} \tag{22}$$

The other two types of weights depend on the resource requirement of the activities within the groups. The groups with a high requirement of (one particular) renewable resource will need to have a larger duration. The total resource requirement $q_{g\kappa}$ of a group G_g for renewable resource R_κ equals the sum of the total request of its activities for this resource, i.e., $q_{g\kappa} = \sum_{A_i \in G_g} q_{i\kappa} \cdot p_i$. If this amount is high relative to the availability $K_{\kappa t}$, then the weight should be large as well.
The weights W_g for a group can be defined as the *total resource requests relative to the resource capacity*:

$$W_g = \sum_{R_\kappa \in \mathcal{R}} \frac{q_{g\kappa}}{\max_t K_{\kappa t}}, \quad \forall G_g \in \mathcal{G}. \tag{23}$$

or as the *maximum of the resource requests relative to the resource capacity*:

$$W_g = \max_{R_\kappa \in \mathcal{R}} \frac{q_{g\kappa}}{\max_t K_{\kappa t}}, \quad \forall G_g \in \mathcal{G}. \tag{24}$$

5 Computational Results

This section describes the setup of the computational tests and its results. After generating instances for the TCPSP with spatial resources, we solve the corresponding GAP's with the 15 different objective functions presented in Section 4, each leading to a different resulting TCPSP. After solving the resulting TCPSP's, we are able to compare the effect of the different objective functions. For the generation of the instances we make use of the project generator ProGen, see Kolisch and Sprecher [11, 12] and Kolisch et al. [13], an instance generator for the RCPSP. The GAP's are solved with CPLEX and the resulting TCPSP with the method from [5].

5.1 Generating Instances

The instances of the TCPSP with spatial resources are generated in three steps. To construct an instance, we first generate a set of spatial resources with their capacities, and a set of groups with their spatial resource requirement and precedence relations. The precedence network and spatial resource availability and requirement are generated with ProGen. In the second step, we generate a set of renewable resources with their capacities, and for each group a set of activities with renewable resource requirements and precedence relations. Again, this is done with ProGen. In the final step, we convert the precedence relations between groups into precedence relations between activities of those groups. Whenever there is a precedence relation from group G_g to group G_h we add a precedence relation from a randomly selected activity from G_g to a randomly selected activity from G_h. Figure 3 displays the three steps of generating an instance.

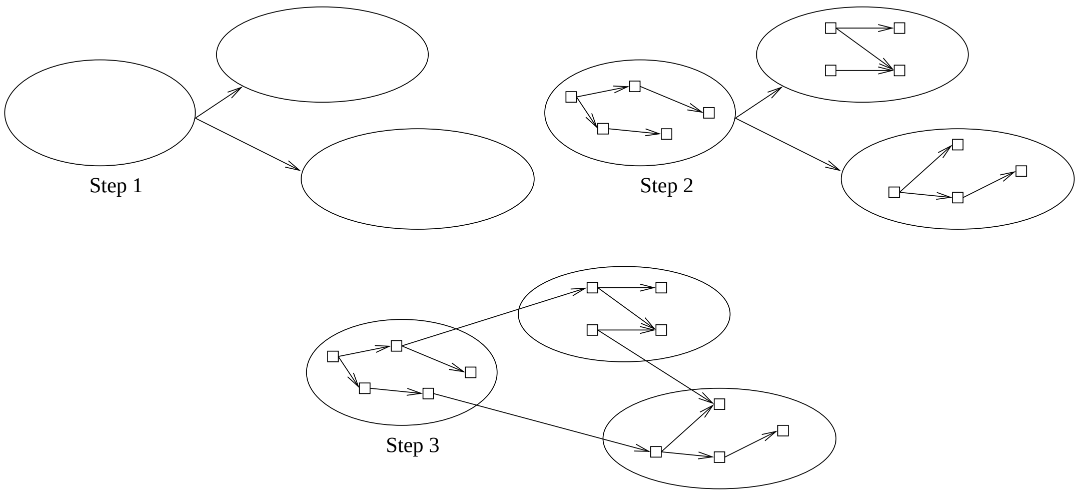

Fig. 3. The three steps of generating an instance for the TCPSP with spatial resources.

Let the release date of each activity be equal to 0. We also derive one deadline that applies to all activities. Let MP denote the minimum project length, which is defined by the longest path in the activity on node network, and let $\bar{T}$ denote the upper bound on the project length, $\bar{T} = \sum_{A_j \in \mathcal{A}} p_j$. We define the project deadline, and thereby the deadline of all the activities, be $MP + 0.1(\bar{T} - MP)$.

Due to the adjacent assignment of the spatial resource units to the groups, there is no guarantee there exist feasible solutions for the generated instances.

However, after solving the GAP, infeasibility of an instance becomes clear and we remove it from our test set.

5.2 Solving the GSP

From each of the generated instances, we derive 15 different GAP's. Each with a different objective function and weight combination. The GAP instances are solved with CPLEX through the ILP formulation implemented in AIMMS. It turns out that even for instances with a small number of groups, the computational time can grow large. Therefore, we have to restrict ourself to instances with 7 groups and 1 spatial resource. From the solutions of the GAP's, we observe that the use of different objective functions results in solutions that are different in assignment and ordering. This means that the additional precedence relations implied by the GAP solutions are different, and also the resulting TCPSP's are different. We can expect different solutions for the original problem.

5.3 Solving the resulting TCPSP

Each generated instance combined with the solution of the GAP, leads to a resulting TCPSP. We solve these TCPSP's with the method from [5]. This is a two stage approach, in which first a feasible schedule is constructed with a randomized sampling technique, and than improved by a local search.
We let the cost of increasing the capacity of a renewable resource by one unit in one time unit be 1, i.e., $c_{\kappa t} = 1$.
This method gives us a schedule of the activities and the due to the precedence relations implied by the GAP, the spatial resource assignment is still feasible. The solution of the GAP and the solution of the resulting TCPSP together give a feasible solution of the TCPSP with spatial resources.

5.4 Comparing the different objective functions

We have generated 100 instances for the TCPSP with spatial resources. Together with the 15 different objective functions this gives 1500 instances for the GAP, each leading to an instance for the resulting TCPSP.
The instances are generated such that the spatial resources are tight, they have to play a role. As a consequence only 63 of the 100 instances generated turned out to have feasible solutions. In the remainder we only consider these 63 feasible instances.
Table 1 displays the average objective value for the resulting TCPSP with the

Table 1
Average objective values of the resulting TCPSP

Objective	Weight: all equal to 1	total resource requests	maximum resource request	best
maximize total weighted duration	409.71	401.16	395.40	381.67
maximize the minimum weighted flexibility	418.67	401.84	402.62	379.08
minimize the weighted shortage of processing time	399.89	393.13	391.22	381.92
minimize the total weighted conflict	411.37	403.00	406.84	384.68
minimize the maximum weighted conflict	410.73	407.02	410.87	377.87
best	366.27	365.33	363.40	352.79

Table 2
Number of times it gives the best solution

Objective	Weight: all equal to 1	total resource requests	maximum resource request
maximize total weighted duration	12	14	17
maximize the minimum weighted flexibility	7	10	17
minimize the weighted shortage of processing time	15	13	16
minimize the total weighted conflict	9	13	10
minimize the maximum weighted conflict	14	9	14

different objective functions. Given an objective, and we take for each of the three weights the best solution we get the average objective value depicted in the rightmost column. Given a weight, and we take for each of the five objectives the best solution we get the bottom row. The bottom right cell give the average objective value, if we take for each instance the best found schedule. In Table 2, we see the number of times each objective gives the best found schedule.

From these two tables we can conclude that there is no dominant objective weight combination. Taking weights dependent on the regular resources improves the quality of the schedules, so putting all weights equal to one is not a good approach. No matter which weight used, the objective *minimize the weighted shortage of processing time* has the best average objective values, but gives for many instances not the best found solution.

[**Pairwise compair the 10 objectives (not with weight=1), and determine which is best.**]

If we combine two objective functions, meaning that we solve each instance twice and take the best solution, we can improve a lot. Combining *maximize the minimum weighted flexibility* and *minimize the weighted shortage of processing time*, both with weight *maximum resource request*, we get an average objective value of 373.06.

6 Conclusions

The project scheduling problem cannot be solved with existing methods since the spatial resource units are required to be assigned in an adjacent manner. Therefore, we have developed a decomposition method for the TCPSP with spatial resources. First we solve the groups to spatial resource assignment, the GAP. The solution found implies additional precedence relations such that we are left with an ordinary TCPSP. With the use of different objective functions in the GAP we want to anticipate the implications for the resulting TCPSP. Although the test results show no dominance among the presented GAP objective functions, it is shown that taking the requirement of regular resources into account does help to find good schedules.

With our decomposition approach, we are able to detect infeasibility in the first step or quite easily construct feasible solutions. Finding good solutions remains problematic, but by taking combinations of objective functions the quality of the generated schedules improves. The computational time of the GAP remains a drawback, it limits the number of groups it can handle.

For future research is would be interesting to see under which conditions which objective function performs well, and to explore different methods to solve the GAP. The presented method gives a decomposition for the TCPSP with spatial resources, and it requires quite some effort to modify it into a method for the RCPSP with spatial resource. Due to the fixed capacities of the regular resources in a RCPSP, it will be much more difficult to anticipate the effect of the additional precedence relations.

References

[1] R. de Boer. *Resource-Constrained Multi-Project Management, A hierarchical decision support system.* PhD thesis, University of Twente, 1998.

[2] C.W. Duin and E. van Sluis. On the complexity of adjacent resource scheduling. *Journal of Scheduling*, 9(1):49–62, 2006.

[3] S.P. Fekete, E. Köhler, and J. Teich. Higher-dimensional packing with order constraints. *To appear in SIAM Journal on Discrete Mathematics*, 2006.

[4] Y. Guan, W-Q Xiao, R.K. Cheung, and C-L Li. A multiprocessor task scheduling model for berth allocation: heuristic and worst-case analysis. *Operations Research Letters*, 30:343–350, 2002.

[5] T.A. Guldemond, J.L. Hurink, J.J. Paulus, and J.M.J. Schutten. Time-constrained project scheduling. *to appear*, 2006.

[6] W. Herroelen, B. De Reyck, and E. Demeulemeester. Resource-constrained project scheduling: a survey of recent developments. *Computers and Operations Research*, 25:279–302, 1998.

[7] K. Hess and Kolisch R. Efficient methods for scheduling make-to-order assemblies under resource, assembly area and part availability constraints. *International Journal of Production Research*, 38(1):207–228, 2000.

[8] R. Kolisch. Integrated scheduling, assembly area-, and part-assignment for large scale make-to-rder assemblies. *International Journal of Production Economics*, 64:127–141, 2000.

[9] R. Kolisch and S. Hartmann. Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis. In J. Weglarz, editor, *Handbook on Recent Advances in Project Scheduling*, pages 197–212. Kluwer, 1999.

[10] R. Kolisch and R. Padman. An integrated survey of deterministic project scheduling. *Omega*, 29:249–272, 2001.

[11] R. Kolisch and A. Sprecher. Project scheduling library - PSPlib. http://129.187.106.231/psplib/.

[12] R. Kolisch and A. Sprecher. PSPLIB a project scheduling problem library. *European Journal of Operational Research*, 96:205–216, 1997.

[13] R. Kolisch, A. Sprecher, and A. Drexl. Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science*, 41, 1995.

[14] A. Lim. The berth planning problem. *Operations Research Letters*, 22:105–110, 1998.

[15] C. Steiger, H. Walder, and M. Platzner. Operating systems for reconfigurable embedded platforms: Online scheduling of real-time tasks. *IEEE Transactions on Computers*, 53(11):1393–1407, 2004.