KATHOLIEKE UNIVERSITEIT
LEUVEN

Faculty of Economics and Applied Economics

A heuristic methodology for solving spatial a resource-constrained project scheduling problems

Eline De Frene, Damien Schatteman, Willy Herroelen and Stijn Van de Vonder

DEPARTMENT OF DECISION SCIENCES AND INFORMATION MANAGEMENT (KBI)

KBI 0722

A heuristic methodology for solving spatial resource-constrained project scheduling problems

Eline De Frene Damien Schatteman Willy Herroelen
Stijn Van de Vonder
Department of Decision Sciences and Information Management
Research Center for Operations Management
Faculty of Economics and Applied Economics
Katholieke Universiteit Leuven (Belgium)
email: <first name>.<family name>@econ.kuleuven.be

June, 2007

Abstract

In this paper we present a heuristic methodology for solving resource-constrained project scheduling problems with renewable and spatial resources. We especially concentrate on spatial resources that are encountered in construction projects, but our analysis can easily be generalized to other sectors. Our methodology is based on the application of a schedule generation scheme on a priority list of activities. We explain why the parallel schedule generation scheme is not applicable for projects with spatial resources. We introduce a procedure for transforming priority lists into precedence and resource feasible lists that avoid deadlocks on the spatial resources. We conclude from a computational experiment on two sets of instances that priority rules performing well for the regular resource-constrained project scheduling problem also perform well in the presence of spatial resources and allow to effectively solve large problems in very short CPU time.
Keywords: Resource-constrained project scheduling, construction, spatial resources

1 Introduction

Our recent experience with field-testing an integrated risk management approach developed under a joint research contract with the *Belgian Building Research Institute* (Schatteman et al. (2006)) revealed the need for the development of scheduling procedures able to cope with so-called *spatial resources* (de Boer (1998)). Spatial resources are assigned to a group of activities rather than to a single activity and are monopolized by the group from the start of the first activity till the completion of the last activity in the group. The objective of this paper is to introduce an effective and efficient heuristic methodology for scheduling resource-constrained projects in the presence of both renewable and spatial resources.

The paper is organized as follows. In Section 2, we review the literature and introduce the type of spatial resources dealt with in this paper through problem settings that may occur in construction projects. A mathematical programming formulation is given in Section 3. In Section 4 we demonstrate why traditional project scheduling methodology fails to deal effectively with the presence of spatial resources. Our heuristic solution methodology is presented in Section 5. Sections 6 and 7 are devoted to the results of a computational experiment performed on two datasets. The last section provides overall conclusions.

2 Spatial resources

The concept of spatial resources has been introduced by de Boer (1998). He explored a project scheduling problem with the Royal Netherlands Navy Dockyards where the dry docks of shipyards served as spatial resources. A spatial resource is not required by a single activity, as is the case with renewable resources, but by a group of activities, called a spatial resource activity group (or just activity group). The spatial resource is occupied from the first moment an activity from such a group starts until the last activity in the group finishes. As long as other resources and precedence constraints permit, all activities in the same group can be scheduled simultaneously. However, if two activity groups require the same spatial resource unit, at most one group can be in progress at the same time.

The solution procedure introduced by de Boer (1998) aims at minimizing maximum lateness and is based on a parameterized regret-based random sampling, analogous to the adaptive search method of Kolisch and Drexl (1996). It has been applied to rather simple spatial resource networks with only one spatial resource (a dry dock) with single unit availability. As a

result, all activities of all spatial resource groups have a constant spatial resource requirement. The procedure has been tested on a small number of generated networks with only 15% of the activities belonging to a spatial activity group.

Paulus and Hurink (2006) and Hurink et al. (2006) added an adjacency constraint to the definition of spatial resources, specifying that the spatial resource units that are assigned to an activity group need to be adjacent in space.

In this paper, we deal with a different and much broader project setting. For the spatial resources used in construction work - formworks, cranes, scaffolds, temporary girders, large machinery - adjacency is not a factor. Some of these are treated as spatial resources because of their inherent immobility. Withdrawing such spatial resources from an activity group during idle times in order to assign them to a different activity group, is only unfeasible due to the required time and effort to move and set-up these resources.

In this paper we drop the adjacency requirement. We consider project networks that require an arbitrary number of spatial resources in more than a single unit. Activities within a group that occupies spatial resource units, may call other spatial resource units and as such belong to different groups. de Boer (1998) makes the crucial assumption that if a project has more than one activity group, precedence relations are set between the groups such that no cycles occur. We will in contrast transform a priority list into a precedence and resource feasible priority list in which cyclical spatial resource calls cannot occur.

We assume that projects are represented by an activity-on-the-node network $G(N, A)$, where the set of nodes N represent the activities and the set of arcs A denotes the finish-start, zero-lag precedence relations. We assume a single dummy start and end node. R^ρ denotes the set of renewable resources and we assume a constant availability of renewable resource type $k \in R^\rho$, denoted as a_k^ρ. The per period required renewable resource units of type k by activity j is a constant number r_{jk}^ρ. The set of spatial resources is denoted as R^σ and we assume a constant availability of spatial resource type $k \in R^\sigma$, denoted as a_k^σ.

We define an activity group $g \in \mathcal{G}$, with $\mathcal{G}$ as the set of activity groups, as a subnetwork of the project network that starts with a so-called *call activity* j_g^c, ends with a *release activity* $j_g^{r_r}$ and contains 0, 1 or more *intermediate activities*, which can be release activities. We will denote an intermediate activity that serves as a release activity for group g as $j_g^{r_i}$. Likewise, an intermediate activity that does not change the spatial resource requirement of the group will be denoted as j_g^i. A *call activity* is the first activity of

the activity group that reserves a number $r^{\sigma}_{j^c_g k} > 0$ of spatial resource units of type $k \in R^{\sigma}$. A call activity $j^c_g \in g$ will obtain the requested spatial resource units at its scheduled starting time $s_{j^c_g}$ so that it can occupy the reserved units for its entire duration $p_{j^c_g}$. The total amount of claimed spatial resource units remain reserved for the group until release activity $j^{r_i}_g$ or $j^{r_r}_g$ releases part respectively all of the remaining claimed resource units of the corresponding spatial resource type. A release activity $j^{r_l}_g$ (with l equal to r or i) of an activity group g releases spatial resource units, i.e. $r^{\sigma}_{j^{r_l}_g k} < 0$, at its completion time $f_{j^{r_l}_g}$, so that they can become available to other groups. Any activity belonging to a particular group can act as call or release activities for other groups. This implies that the call activity of group g claiming resource type $k \in R^{\sigma}$ may at the same time be a call activity for another group $g' \neq g$ for the same resource type $k \in R^{\sigma}$.

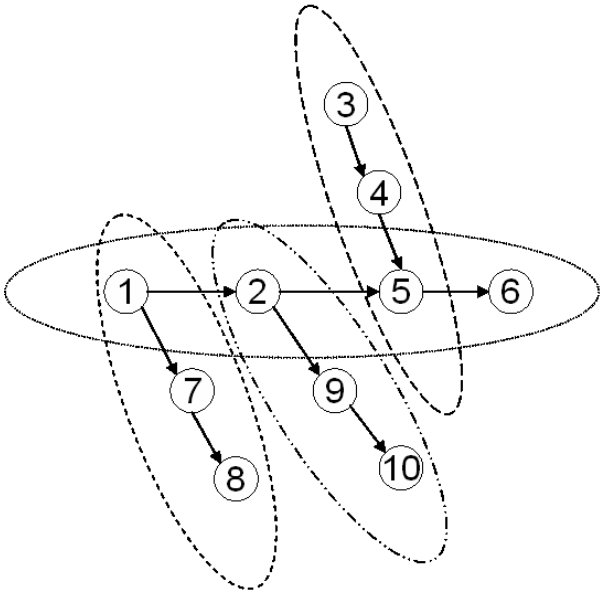

Figure 1: Activity groups

As shown in Figure 1, the activities of an activity group can also be a member of other activity groups. For simplicity, we assume that intermediate activities of activity groups do not serve as release activity. Activity 1, for example, is a call activity that claims a number of units of spatial resource type 1 that will be occupied by the activities 1, 2, 5 and 6 of activity group 1 until they are released by release activity 6. Activity 1 is also a call activity that claims a number of units of resource type 1 and resource type 2 to be occupied by the second group activities 1, 7 and 8 until they are released

by the release activity 8. Activity 2, being a member of activity group 1, is also a call activity requesting a number of units of resource type 1 and 3 to be used by the activities 2, 9 and 10 of activity group 3. Activity 5 is a release activity of a fourth activity group that releases the resource units of the resource types that were claimed for this group by its call activity 3.

The subset $X \subset N$ unites all call activities in the project. Likewise $Y \subset N$ groups all release activities. We remark that an activity can belong to X as well as to Y.

3 Formal problem statement

The basic *resource-constrained project scheduling problem* (RCPSP) can be conceptually formulated as:

$$\text{minimize } s_n \tag{1}$$

subject to

$$\sum_{j \in S_t} r_{jl}^{\rho} \leq a_l^{\rho} \qquad \forall l \in R^{\rho};\ \forall t \tag{2}$$

$$s_j \geq s_i + d_i \qquad \forall (i,j) \in A \text{ with } i,j \in N \tag{3}$$

$$s_j \geq 0 \qquad \forall j \in N \tag{4}$$

The objective function in Eq. (1) is to minimize the project makespan, given by the start time s_n of the dummy end activity n. Eqs. (2) are the renewable resource constraints. For each renewable resource type $l \in R^{\rho}$, we assume that a finite amount a_l^{ρ} is available on a period-per-period basis. Resource feasibility implies that for each resource type l and for each time period t the sum of the resource requirements r_{jl}^{ρ} of the activities that are in progress during period t ($S_t = \{j | s_j \leq t < s_j + p_j\}$) cannot exceed the availability a_l^{ρ}. Eqs. (3) specify the precedence constraints. Whenever $(i,j) \in A$, activity j cannot start before immediate predecessor i has finished. s_j represents the starting time of activity j and p_j its deterministic duration. Eqs. (4) impose non-negativity constraints on the decision variables s_j.

The basic RSPSP (problem $m,1|cpm|C_{\max}$ in the notation of Herroelen et al. (2000)) has been shown by Blazewicz et al. (1983) to be NP-hard in the strong sense.

The RCPSP only takes resource constraints for renewable resources into account. The *spatial resource-constrained project scheduling problem* (problem $m,1\sigma|cpm|C_{\max}$ in the notation of Herroelen et al. (2000)) additionally requires the specification of spatial resource constraints, as follows:

$$\sum_{g \in G_t} r^{\sigma}_{gk} \leq a^{\sigma}_{k} \qquad \forall k \in R^{\sigma};\, \forall t \qquad (5)$$

As we already mentioned before, each spatial resource type $k \in R^{\sigma}$ has a constant availability of a^{σ}_{k} per time period t. For each spatial resource type k and for each time period t the sum of the resource requirements r^{σ}_{gk} of the activity groups that are in progress during period t ($G_t = \{g|s_g \leq t < f_g\}$) cannot exceed the availability a^{σ}_{k}. Hereby, s_g and f_g denote the start, respectively the completion times of activity group g, i.e. $s_g = s_{j^c_g}$ and $f_g = f_{j^{r_r}_g}$.

The spatial resource unit requirements to be taken into account in each time period t will be determined as follows:

$$r^{\sigma}_{gk} = \begin{cases} r^{\sigma}_{j^c_g k} & \text{if } j^c_g \in S_t \\ \sum_{o \in Z_i} r^{\sigma}_{ok} & \text{otherwise} \end{cases}$$

In other words, if a call activity j^c_g belonging to activity group g is in progress in period t, i.e. $j^c_g \in S_t$, then the number of spatial resource units occupied by the group in period t is defined as $r^{\sigma}_{gk} = r^{\sigma}_{j^c_g k}$. If no activity j^c_g of the activity group g is in progress during period t, i.e. $j^c_g \notin S_t$, or the activity in progress is an intermediate activity j^i_g or a release activity $j^{r_l}_g$, then r^{σ}_{gk} is set equal to $\sum_{o \in Z_i} r^{\sigma}_{ok}$, where Z_i is the set of predecessors of activity $i|i = j^i_g$ or $i = j^{r_l}_g$, with $i \in g$ and $f_i \leq t$.

The spatial resource-constrained project scheduling problem adds spatial resource constraints to the strongly NP-hard RCPSP. Its complexity justifies the use of heuristics. The literature on the general type of spatial RCPSP introduced in this paper is to the best of our knowledge void.

4 Spatial resources and standard project scheduling procedures

Before introducing our heuristic scheduling approach for solving the spatial resource-constrained project scheduling problem (see Section 5), we will first illustrate why spatial resources require special treatment.

4.1 Spatial resources modeled as renewable resources

It would be tempting to treat spatial resources as if they were regular renewable resources which boils down to solve the basic RCPSP model consisting of Eqs. (1) - (4), supplemented with the following spatial resource constraints:

$$\sum_{j \in S_t} r_{jk}^{\sigma} \leq a_k^{\sigma} \qquad \forall k \in R^{\sigma}; \forall t \tag{6}$$

In our problem setting, however, the activities of a spatial activity group, contrary to the setting of de Boer (1998), cannot be scheduled simultaneously. Moreover, even when the spatial resources that are reserved for an activity group are not used at a particular point in time, they cannot be considered as available for other groups.

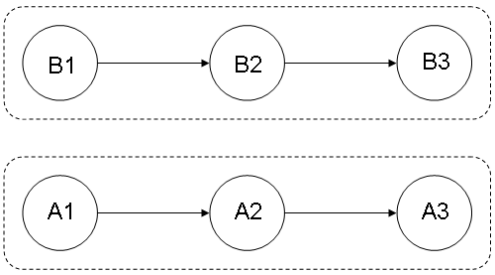

Figure 2: Project network

Consider the construction of two separate concrete walls, wall A and wall B, as shown in Figure 2. In order to be erected, both walls require a supporting formwork with single unit availability. The construction of a concrete wall can thus be modeled as a spatial activity group that is a chain of three activities, i.e. placing the formwork (activities A1 for wall A and B1 for wall B), pouring the wall (activities A2 and B2), and removing the formwork (activities A3 and B3). Solving this small problem as an RCPSP could result in the schedule shown in Figure 3. In this schedule, the execution of the activity group for wall A is preempted upon completion of activity A1 by activity B1 that belongs to the second activity group. Although the single unit resource availability constraint is not violated at any time, this schedule is clearly unfeasible. When activity 1 finishes, activity B1 cannot reserve the spatial resource unit as long as it has not been released by release activity A3. Modelling spatial resources as renewable resources may clearly lead to infeasible schedules.

It would be tempting to avoid this type of schedule infeasibility by imposing precedence constraints between the activity groups. Our experience

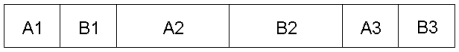

Figure 3: An RCPSP solution

(Schatteman et al. (2006)) shows this to be a common practice among Belgian construction companies. In the example of Figure 2, introducing a minimal finish-start, zero-lag precedence relation between release activity A3 and call activity B1, or between release activity B3 and call activity A1 would avoid a spatial resource conflict and would avoid the infeasibility shown in Figure 3.

Avoiding schedule infeasibilities by introducing extra precedence relations, however, may prevent the generation of high quality schedules. Consider the project shown in Figure 4 that consists of two spatial resource activity groups: building a wall and making a ledger. The erection of the wall requires formwork 1 while making a ledger requires formwork 2. The concrete wall has to be poured (activity A2) before the ledger can be made (activity B1). Activities A1, A2, A3 and B1, shown shaded in the figure, all require the same single renewable resource unit.

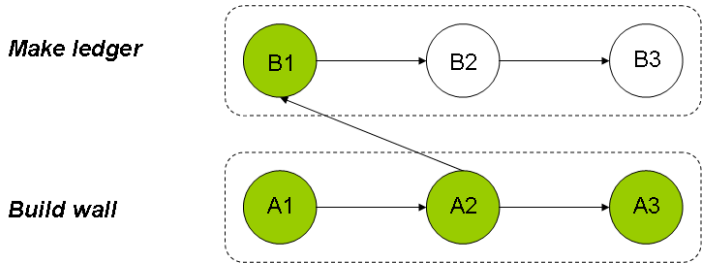

Figure 4: Project network

Translating the finish-start zero-lag precedence relation between activity A2 and B1 into a finish-start, zero-lag precedence relation between the two activity groups, i.e. between activity A3 and B1, or imposing maximal zero-lag precedence relations between the three activities of each activity group (or equivalently, an extra time-lag between activity A2 and B1 equal to the duration of activity A3), leads to the schedule shown in Figure 5.

Figure 5: Solution

Because both activity groups do not require the same spatial resource,

however, it is possible to start B1 before A3. This minimal makespan solution is presented in Figure 6.

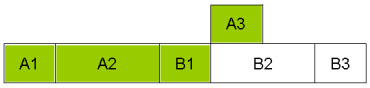

Figure 6: Optimal solution

4.2 Replacing spatial activity groups by a single aggregate activity

Infeasible preemption of activity groups, the problem illustrated in Figure 3, can be prevented by aggregating the activities of a spatial activity group into a single non-preemptable aggregate activity. However, such an approach may also overlook many potentially good schedules when the project activities also require renewable resources.

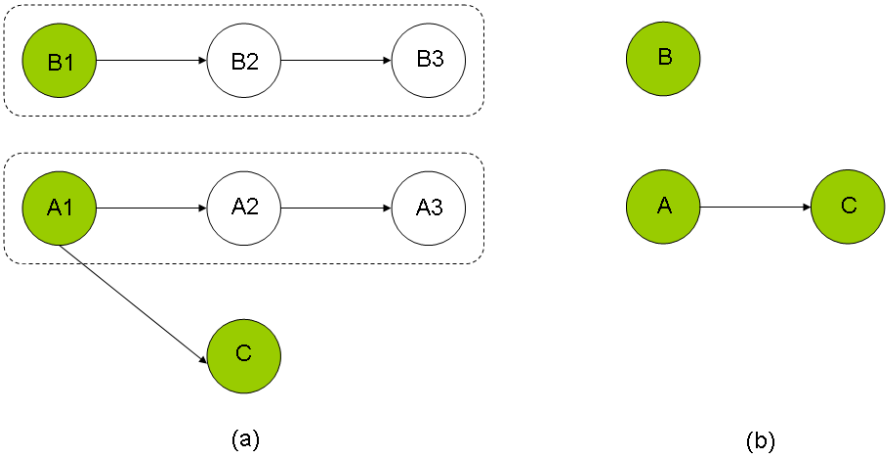

Figure 7: Network (a) without aggregation (b) with aggregation

Consider again the construction of two separate concrete walls, wall A and wall B, which require the same formwork. Activity C (make ledger) can start as soon as activity A1 (placing the formwork) is finished. Activities A1, B1 and C require one unit of a single renewable resource type. Figure 7(a) represents the corresponding network (the activities requiring the renewable resource unit are shaded). If the activities of the activity groups are aggregated into a single non-preemptable activity (Figure 7(b)), the renewable resource unit will be in use for the entire duration of the aggregated

activities A and B, forcing activity C to be scheduled in series with the non-preemptable aggregated activities A and B (creating either the chains <B,A,C>, <A,B,C> or <A,C,B>).

Figure 8(b) shows a resulting schedule with activity C scheduled after the two groups. Figure 8(a) gives a minimal makespan schedule in which activity C is scheduled in parallel with activity A2, which is feasible because activities A2 and A3 do not require the renewable resource unit.

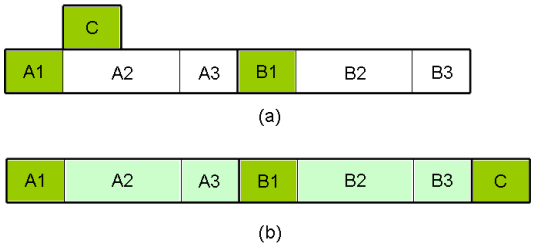

Figure 8: Solution (a) without aggregation (b) with aggregation

5 Algorithm

The heuristic solution procedure described in this section involves the application of a schedule generation scheme on a priority list of activities. The literature on priority list based schedule generation schemes for the RCPSP is very extensive (for overviews see Kolisch (1996a) and Kolisch (1996b)).

When spatial resources that are to be reserved for all the activities of activity groups come into play, however, it must be guaranteed that the priority lists do not result in deadlocks. A *deadlock* is a situation where two or more competing actions are waiting for the other to finish, and thus neither ever does. In the context of spatial resources, deadlock refers to a specific condition when two or more activities are each waiting for another to release spatial resource units or more than two activities are waiting for resources in a circular chain.

Consider the modified example of Figure 2, where both walls are now the lower and upper part of the same large wall. Activity 6 denotes the extra activity corresponding to the placement of the upper part on top of the lower part by means of a crane. Both parts still require the presence of a supporting formwork until the part is completed. For the lower part, the formwork can only be removed after the upper part has been placed on top of the lower part. The upper and lower part can be regarded as two activity groups. The construction of the lower part consists of setting up the formwork (activity

1), pouring the concrete (activity 2) and removing the formwork (activity 7). Analogously, the construction of the upper part consists of setting up the formwork (activity 3), pouring the concrete (activity 4) and removing the formwork (activity 5). Mounting the formwork for the lower part (activity 1) and mounting the formwork for the upper part (activity 3) are the call activities. Removing the formwork from the lower part (activity 7) and from the upper part (activity 5) are the release activities. Figure 9 shows the precedence relationships for this project.

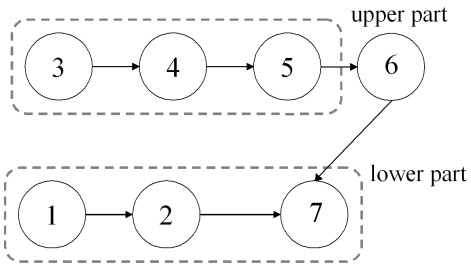

Figure 9: A project with possible deadlock

Assume a single formwork is available. If we schedule this project by applying a *serial generation scheme* (SGS) on the priority list $\lambda = (1, 2, 3, 4, 5, 6, 7)$, starting with activity 1 would result in a deadlock. After scheduling activities 1 and 2, activity 7 cannot be started yet because it waits for activity 6 to finish and thus transitively for activity 3 to start. Activity 3 on the other hand, waits for activity 7 to release the formwork. This results in a circular wait between activities 3 and 7. Applying a different priority list, for example list $\lambda = (3, 4, 5, 1, 2, 6, 7)$, avoids the deadlock and leads to a feasible schedule.

In order to avoid deadlocks, we need a procedure for transforming an input priority list λ into a precedence and resource feasible priority list μ.

5.1 Transforming the priority list

In this section a procedure is proposed that transforms any priority list λ into a precedence and spatial resource feasible list μ. The procedure consists of two algorithms. In Algorithm 1, we start from a list χ that includes all call activities in the set X sorted in the order dictated by priority list λ. The algorithm transforms χ into a precedence and spatial resource feasible list β. Algorithm 2 generates a precedence feasible priority list μ, taking into account the precedence relations between the call activities imposed by the spatial resource feasible list generated by Algorithm 1. The schedule

generation scheme to transform the priority list into a feasible schedule is described in Section 5.4.

5.1.1 Constructing a feasible call activity list

We denote by $Card_X$ the cardinality of the set of call activities X, and we let x denote the position in list χ, B the set of activities that are already added to β and b the position in list β.

The algorithm initializes $B = \emptyset$. For every spatial resource type $k \in R^\sigma$, the variable V_k that denotes the current availability of that resource, is initialized to a_k^σ.

For each release activity $j \in Y$, we define the set P_j with cardinality $card_{P_j}$, as the set of call activities that are direct or transitive predecessors of j.

For each spatial resource $k \in R^\sigma$, we define Y_k as the set of release activities for that spatial resource:

$$\forall k \in R^\sigma, i \in Y : i \in Y_k \Leftrightarrow r_{ik}^\sigma < 0. \tag{7}$$

We also initialize the variables $nc_{P_j,k}$ and $re_{P_j,k}$, where $nc_{P_j,k}$ stands for *not called* and $re_{P_j,k}$ for *released*. $nc_{P_j,k}$ is initialized by:

$$\forall k \in R^\sigma, j \in Y_k, i \in P_j : nc_{P_j,k} = \sum_{i=1}^{card_{P_j}} r'_{ik} \text{ with } r'_{ik} = max(0, r_{ik}). \tag{8}$$

$re_{P_j,k}$ is set equal to 0 for $\forall j$, $\forall k$.

We also initialize every $nc'_{P_j,k} = nc_{P_j,k}$. Analogously every $re'_{P_j,k}$ is set equal to $re_{P_j,k}$. The role of the variables $nc'_{P_j,k}$ and $re'_{P_j,k}$ will immediately become clear. The procedure runs as depicted in Algorithm 1. The variable initializations for this algorithm are given in Table 1.

$B = \emptyset$
$b = 1$
$x = 1$
$\forall k \in R^\sigma : V_k = a_k^\sigma$
$\forall k \in R^\sigma, j \in Y_k, i \in P_j : nc'_{P_j,k} = nc_{P_j,k} = \sum_{i=1}^{card_{P_j}} r'_{ik}$
$\forall k \in R^\sigma, j \in Y_k : re'_{P_j,k} = re_{P_j,k} = 0$

Table 1: Variable initializations Algorithm 1

Algorithm 1 Making the call activity list feasible

WHILE $(b \neq (card_X + 1))$ DO

WHILE $(b \neq 0)$ DO

WHILE $(x \neq (card_X + 1)$ and $(X \setminus B) \neq \emptyset)$ DO

1. *Check whether already in β*

IF $(\chi_x \in B)$ THEN $x = x + 1$

2. *Check precedence feasibility*

ELSE IF $(\exists i \in X | (i, \chi_x) \in A$ and $i \notin B)$ THEN $x = x + 1$

3. *Check spatial resource feasibility*

ELSE $\forall k \in R^\sigma : V'_k = V_k - r_{\chi_x,k}$

IF $(\exists k \in R^\sigma | V'_k < 0)$ THEN $x = x + 1$

ELSE

$\forall k \in R^\sigma :$

$\forall j \in Y_k :$

IF $((P_j \setminus B) \neq \emptyset$ and $\chi_x \in P_j)$ THEN

$nc'_{P_j,k} = nc_{P_j,k} - r_{\chi_x,k}$

IF $(nc'_{P_j,k} = 0$ and $P_j \setminus (B \cup \chi_x) = 0)$ THEN $re'_{P_j,k} = re_{P_j,k} - r_{j,k}$

$V''_k = V_k$

IF $(r_{\chi_x,k} \geq 0)$ THEN $V''_k = V''_k - r_{\chi_x,k}$

IF $(re_{P_j,k} \neq 0)$ THEN $V''_k = V''_k + re_{P_j,k}$

IF $(\nexists j \in Y_k | (P_j \setminus (B \cup \chi_x) \neq 0$ and $nc'_{P_j,k} \leq V''_k)$ THEN

$x = x + 1$

break

4. *Add activity to β*

IF not break THEN

$\beta_b = \chi_x$

$B = (B \cup \chi_x)$

$\forall k \in R^\sigma :$

$\forall j \in Y_k :$

$nc'''_{P_j,k} = nc_{P_j,k}$ and $re'''_{P_j,k} = re_{P_j,k}$

$nc_{P_j,k} = nc'_{P_j,k}$ and $re_{P_j,k} = re'_{P_j,k}$

$V'''_k = V_k$

$V_k = V''_k$

$x = 1$

$b = b + 1$

END DO

5. *Backtrack*

IF $(x = card_X + 1)$ THEN

$b = b - 1$

IF $(b \neq 0)$ THEN

$B = (B \setminus \beta_b)$

$\forall k \in R^\sigma :$

$\forall j \in Y_k :$

$nc_{P_j,k} = nc'''_{P_j,k}$ and $re_{P_j,k} = re'''_{P_j,k}$

$V_k = V'''_k$

$x = \chi(\beta_b) + 1$

ELSE no feasible list exists

END DO

END DO

The procedure tries to add activities to B one by one, by considering activities in the order of list χ. As long as $x \neq card_X + 1$ and $(X \setminus B) \neq \emptyset$, it can be checked if activity χ_x can be added to B. Otherwise, no activity can be currently assigned and a deadlock has been detected. The procedure backtracks to the previous activity in β.

Step 1 considers the first unscheduled activity in χ.

Precedence feasibility is checked in *Step* 2. If an activity has call activity predecessors that are not assigned to B, the next activity in χ is considered.

When the first eligible activity has been found, the algorithm checks in *Step* 3 for spatial resource feasibility. It is first checked whether sufficient units are available for each spatial resource. If not, the next activity in χ is considered.

If sufficient units are available for the current activity, the procedure goes through a spatial resource loop. The first spatial resource is examined. For each $j \in Y_k$, the procedure checks if the current activity is a predecessor of j. If so, the value of $nc_{P_j,k}$ is reduced by the spatial resource requirement of the current activity. Remark that we do not change the value of $nc_{P_j,k}$ but store the change in an intermediate variable $nc'_{P_j,k}$. The procedure also checks if all call activities in P_j are already assigned to B. If so, the release activity j may release its spatial resources units. Next, the algorithm calculates the potential availability V''_k of spatial resource k after current activity χ_x would be scheduled. V''_k is equal to V_k minus the called number of units and plus the number of released units.

Now we check whether we will not be entangled by adding this activity to B. That is to say, if for this spatial resource k, there is at least one release activity j with not all predecessors definitively or tentatively scheduled for which it holds that $nc'_{P_j,k} \leq V''_k$, the loop is re-executed for the next spatial resource. If no such release activity can be found, the procedure considers the next activity in list χ. If the loop can be executed for all spatial resources and if for the last spatial resource the while-loop does not need to be broken, the algorithm can move to the next step.

In *Step* 4, activity χ_x is added to β. The procedure makes backtrack copies $nc'''_{P_j,k}$, $re'''_{P_j,k}$ and V'''_k of variables $nc_{P_j,k}$, $re_{P_j,k}$ and V_k. Upon adding χ_x it could indeed be the case that no other activity can be added to β forcing the procedure to backtrack. By working this way, we can thus easily delete the activity from the output list β. Step 4 will also update the variables $nc_{P_j,k}$, $re_{P_j,k}$ and V_k by setting them equal to $nc'_{P_j,k}$, $re'_{P_j,k}$ and V''_k. If an activity is added to β, we set x equal to 1, increase b and re-execute the while-loop. If an activity is deleted from the output list in *Step* 5, the algorithm is continued for the next x in the input list.

Ultimately, a precedence and spatial resource feasible list β will be found,

if one exists. If there does not exist such a list, there is no need to execute the second algorithm, because no feasible solution exists for the network.

5.1.2 Constructing a feasible priority list

Algorithm 2 accepts the priority list λ with n activities and changes the position of the activities until a precedence and spatial resource feasible list μ is obtained. As mentioned before, we will take the resulting order of list β of Algorithm 1 into account.

$M = \{0\}$
$m = 1$
$l = 1$
$b = 1$
$\forall k \in R^{\sigma} : V_k = a_k^{\sigma}$

Table 2: Variable initializations Algorithm 2

The algorithm starts by initializing the set M of activities that are already added to list μ as $M = \{0\}$, with activity 0 being the dummy start activity. Analogously to the use of subscripts x and b in Algorithm 1, we use the subscripts l and m to indicate the positions in lists λ and μ respectively. For every spatial resource $k \in R^{\sigma}$, V_k is again initialized to a_k^{σ}. Furthermore we set b, the position of the considered activity in β, equal to 1.

In *Step* 1, we look for the first unscheduled activity. *Step* 2 checks the spatial resource availability after taking into account the requirement for the considered activity λ_l. If this availability becomes negative for at least one spatial resource, the procedure takes the next activity of the priority list λ. In *Step* 3 precedence feasibility is checked. If the activity has predecessors that are not assigned to M, the next activity in priority list λ is considered. In order to obtain spatial resource feasibility, it is checked whether the sequence of the activities in β is preserved. More specifically, if λ_l is a call activity and if there exist one or more unscheduled activities that precede λ_l in β, then the procedure continues with the next activity in λ. The activity is added to μ in *Step* 4. If a call activity is added to μ, b is increased by one.

The combination of these two algorithms leads to a precedence and resource feasible priority list μ. This list can be used to schedule activities by a schedule generation scheme as will be explained in Section 5.4.

Algorithm 2 Making the priority list feasible

WHILE $(m \neq n)$ DO

1. *Check whether already in μ*

IF $(\lambda_l \in M)$ THEN $l = l + 1$

2. *Check spatial resource availability*

ELSE $\forall k \in R^\sigma : V'_k = V_k - r_{\lambda_l,k}$

IF $(\exists k \in R^\sigma | V'_k < 0)$ THEN $l = l + 1$

3. *Check precedence feasibility*

a. *Common precedence feasibility*

ELSE IF $(\exists i \in N | (i, \lambda_l) \in A$ and $i \notin M)$ THEN $l = l + 1$

b. *Spatial resource feasibility*

ELSE IF $(\lambda_l \in X$ and $\lambda_l \neq \beta_b)$ THEN $l = l + 1$

4. *Add activity to μ*

ELSE $\mu_m = \lambda_l$

$M = (M \cup \lambda_l)$

$\forall k \in R^\sigma : V_k = V'_k$

IF $(\lambda_l \in X)$ THEN $b = b + 1$

$l = 1$

$m = m + 1$

END DO

5.2 Dominance rules

For speeding up our procedure, two dominance rules are added. The dominance rules will both limit the amount of backtracking required by Algorithm 1.

The first dominance rule is the use of *a higher backtrack level*. In Algorithm 1 we used level 0 as backtrack level. If in our backtrack procedure we arrive at level 0, we know that no feasible list β exists. But whenever we add an activity to β we can do the following. If for every spatial resource $k \in R^\sigma : V_k = a_k^\sigma$, we can raise our backtrack level to b, the number of activities already added to B. If we then have to backtrack and arrive at level b, we know we can stop backtracking because we are sure we will not find a feasible list.

The second dominance rule is *the addition of an extra control array*, whereby the number of elements is equal to 2^{card_X}. Every element in the array is initialized to 0. Each time we arrive at the backtrack step, Step 5 in Algorithm 1, we save the set of activities to which we cannot add one of the remaining unassigned activities. Concretely, for every saved set of activities we generate a corresponding binary value. For example, if set $\{1, 2\}$ is fathomed, the value of element 5 $(2^1 + 2^2)$ is set equal to 1. Later on, if we place activity 2 in the first position of β and we want to check if we can put activity 1 in the next position in the output list, we first have to determine whether it is still useful to examine this branch of the tree. The value of our new set $\{2, 1\}$ will also be equal to 5 $(2^2 + 2^1)$. We can see in our array that the value of element 5 is already 1, so that we do not have to go on with this branch. This dominance rule thus prevents the branching of already considered sets. The higher the total number of different call activities over all spatial resources, the more effective this second dominance rule. The results of this dominance rule will be given in Section 7.

5.3 An illustrative example

Figure 10 shows a 10-activity project that requires a single spatial resource type with $a^\sigma = 2$. The curved lines in the figure identify the activity groups. Each group requires one unit of the spatial resource. For example for activity group $\{2, 3\}$, $r_2^\sigma = +1$ and $r_3^\sigma = -1$. We observe that the set of call activities $X = \{2, 4, 5, 7\}$ and the set of release activities $Y = \{3, 6, 8\}$. We have $P_3 = \{2\}$, $P_6 = \{2, 4, 5\}$ and $P_8 = \{2, 4, 5, 7\}$.

For every P_j, we derive the values of nc_{P_j}, nc'_{P_j}, re_{P_j} and re'_{P_j} :

$nc'_{P_3} = nc_{P_3} = r_2 = 1$ $\qquad$ $re'_{P_3} = re_{P_3} = 0$

$nc'_{P_6} = nc_{P_6} = r_2 + r_4 + r_5 = 3$ $\qquad$ $re'_{P_6} = re_{P_6} = 0$

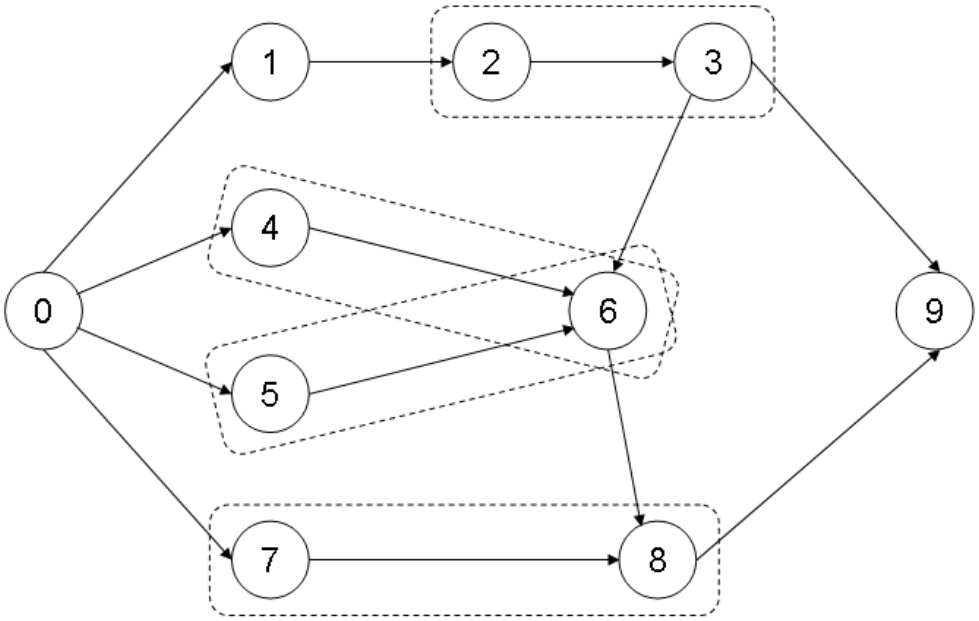

Figure 10: A project network with spatial resources

$nc'_{P_8} = nc_{P_8} = r_2 + r_4 + r_5 + r_7 = 4 \qquad re'_{P_8} = re_{P_8} = 0$

We illustrate the procedure on the activity list $\lambda = (7, 4, 2, 1, 5, 3, 8, 6, 9)$.

The procedure is initialized by setting $B = \emptyset$, $b = 1$, $x = 1$ and $V = a^\sigma = 2$. The procedure starts from the ordered list of call activities $\chi = (7, 4, 2, 5)$ and runs as follows.

$\underline{\mathbf{b = 1}}$

$b \neq card_X + 1 = 5; b \neq 0$

$x = 1; x \neq card_X + 1 = 5$ and $X \setminus B = \{2, 4, 5, 7\} \neq \emptyset$

$\chi_x = 7$ (activity 7 is the first activity in the ordered list of call activities)

1. 7 is not an element of B
2. Activity 7 has no predecessor call activities
3. Check spatial resource availability

   $V' = V - r_7^\sigma = 2 - 1 = 1$

   Check spatial resource feasibility

   $nc'_{P_3} = 1$ and $nc'_{P_6} = 3$

   Activity 7 precedes release activity 8: $nc'_{P_8} = 4 - 1 = 3$

   $V'' = 2 - 1 + 0 = 1$

   $nc'_{P_3} \leq V''$
4. Add activity 7 to B

$\beta_1 = 7$
$B = \{7\}$
$nc'''_{P_3} = nc_{P_3} = 1$ $\quad re'''_{P_3} = re_{P_3} = 0$
$nc'''_{P_6} = nc_{P_6} = 3$ $\quad re'''_{P_6} = re_{P_6} = 0$
$nc'''_{P_8} = nc_{P_8} = 4$ $\quad re'''_{P_8} = re_{P_8} = 0$
$nc_{P_3} = nc'_{P_3} = 1$ $\quad re_{P_3} = re'_{P_3} = 0$
$nc_{P_6} = nc'_{P_6} = 3$ $\quad re_{P_6} = re'_{P_6} = 0$
$nc_{P_8} = nc'_{P_8} = 3$ $\quad re_{P_8} = re'_{P_8} = 0$
$V''' = 2$ and $V = 1$
$x = 1$
$b = 2$

$\underline{\mathbf{b = 2}}$

$x = 1; \chi_x = 7$
1. $7 \in B$; $x = x + 1 = 2$

$x = 2; \chi_x = 4$
1. $4 \notin B$
2. Activity 4 has no predecessor call activities
3. $V' = V - r_4^\sigma = 1 - 1 = 0$
   $V'' = 1 - 1 + 0 = 0$
   $nc'_{P_3} = 1$
   Activity 4 precedes release activities 6 and 8: $nc'_{P_6} = 3 - 1 = 2$ and $nc'_{P_8} = 3 - 1 = 2$
   $nc'_{P_3}, nc'_{P_6}, nc'_{P_8} > V''$; $x = x + 1 = 3$

$x = 3; \chi_x = 2$
1. $2 \notin B$
2. Activity 2 has no predecessor call activities
3. $V' = 1 - 1 = 0$
   Activity 2 precedes activities 3, 6 and 8: $nc'_{P_3} = 1 - 1 = 0$, $nc'_{P_6} = 3 - 1 = 2$ and $nc'_{P_8} = 3 - 1 = 2$
   $nc'_{P_3} = 0$ and $P_3 \backslash (B \cup \chi_x) = 0$: $re'_{P_3} = re_{P_3} - r_3^\sigma = 0 - (-1) = 1$
   $V'' = 1 - 1 + 1 = 1$
   $nc'_{P_6}, nc'_{P_8} > V''$; $x = x + 1 = 4$

$x = 4; \chi_x = 5$
1. $5 \notin B$
2. Activity 5 has no predecessor call activities
3. $V' = 1 - 1 = 0$
   $nc'_{P_3} = 1$, $nc'_{P_6} = 2$ and $nc'_{P_8} = 2$
   $V'' = 1 - 1 + 0 = 0$
   $nc'_{P_3} = 1$
   Activity 5 precedes activities 6 and 8: $nc'_{P_6} = 3 - 1 = 2$

and $nc'_{P_8} = 3 - 1 = 2$

$nc'_{P_3}, nc'_{P_6}, nc'_{P_8} > V''$; $x = x + 1 = 5$

$x = 5$

5. Backtrack

$b = b - 1 = 1$

$b \neq 0$:

$B = \emptyset$

$nc_{P_3} = nc'''_{P_3} = 1$ $\quad$ $re_{P_3} = re'''_{P_3} = 0$

$nc_{P_6} = nc'''_{P_6} = 3$ $\quad$ $re_{P_6} = re'''_{P_6} = 0$

$nc_{P_8} = nc'''_{P_8} = 4$ $\quad$ $re_{P_8} = re'''_{P_8} = 0$

$V = V''' = 2$

$x = \chi(\beta_b) + 1 = 1 + 1 = 2$

$\underline{\mathbf{b = 1}}$

$x = 2; \chi_x = 4$

1. $4 \notin B$
2. Activity 4 has no predecessor call activities
3. $V' = 2 - 1 = 1$

$nc'_{P_3} = 1$ and $nc'_{P_6} = 2$ and $nc'_{P_8} = 3$

$V'' = 2 - 1 + 0 = 1$

$nc'_{P_3} \leq V''$

4. $\beta_1 = 4$

$B = \{4\}$

$nc'''_{P_3} = nc_{P_3} = 1$ $\quad$ $re'''_{P_3} = re_{P_3} = 0$

$nc'''_{P_6} = nc_{P_6} = 3$ $\quad$ $re'''_{P_6} = re_{P_6} = 0$

$nc'''_{P_8} = nc_{P_8} = 4$ $\quad$ $re'''_{P_8} = re_{P_8} = 0$

$nc_{P_3} = nc'_{P_3} = 1$ $\quad$ $re_{P_3} = re'_{P_3} = 0$

$nc_{P_6} = nc'_{P_6} = 2$ $\quad$ $re_{P_6} = re'_{P_6} = 0$

$nc_{P_8} = nc'_{P_8} = 3$ $\quad$ $re_{P_8} = re'_{P_8} = 0$

$V''' = 2$ and $V = 1$

$x = 1$

$b = 2$

$\underline{\mathbf{b = 2}}$

$x = 1; \chi_x = 7$

1. $7 \notin B$
2. Activity 7 has no predecessor call activities
3. $V' = 1 - 1 = 0$

$nc'_{P_3} = 1$ and $nc'_{P_6} = 2$ and $nc'_{P_8} = 2$

$V'' = 1 - 1 + 0 = 0$

$nc'_{P_3}, nc'_{P_6}, nc'_{P_8} > V''$; $x = x + 1 = 2$

$x = 2; \chi_x = 4$

1. $4 \in B$; $x = x + 1 = 3$

$x = 3; \chi_x = 2$

1. $2 \notin B$
2. Activity 2 has no predecessor call activities
3. $V' = 1 - 1 = 0$

   $nc'_{P_3} = 0$ and $nc'_{P_6} = 1$ and $nc'_{P_8} = 2$

   $nc'_{P_3} = 0$ and $P_3 \backslash (B \cup \chi_x) = 0$: $re'_{P_3} = re_{P_3} - r_3^\sigma = 0 - (-1) = 1$

   $V'' = 1 - 1 + 1 = 1$

   $nc'_{P_6} \leq V''$
4. $\beta_2 = 2$

   $B = \{2, 4\}$

   $nc'''_{P_3} = nc_{P_3} = 1$ $\quad$ $re'''_{P_3} = re_{P_3} = 0$

   $nc'''_{P_6} = nc_{P_6} = 2$ $\quad$ $re'''_{P_6} = re_{P_6} = 0$

   $nc'''_{P_8} = nc_{P_8} = 3$ $\quad$ $re'''_{P_8} = re_{P_8} = 0$

   $nc_{P_3} = nc'_{P_3} = 0$ $\quad$ $re_{P_3} = re'_{P_3} = 1$

   $nc_{P_6} = nc'_{P_6} = 1$ $\quad$ $re_{P_6} = re'_{P_6} = 0$

   $nc_{P_8} = nc'_{P_8} = 2$ $\quad$ $re_{P_8} = re'_{P_8} = 0$

   $V''' = 1$ and $V = 1$

   $x = 1$

   $b = 3$

$\underline{\mathbf{b = 3}}$

$x = 1; \chi_x = 7$

1. $7 \notin B$
2. Activity 7 has no predecessor call activities
3. $V' = 1 - 1 = 0$

   $nc'_{P_6} = 1$ and $nc'_{P_8} = 1$

   $V'' = 1 - 1 + 0 = 0$

   $nc'_{P_6}, nc'_{P_8} > V''$; $x = x + 1 = 2$

$x = 2; \chi_x = 4$

1. $4 \in B$; $x = x + 1 = 3$

$x = 3; \chi_x = 2$

1. $2 \in B$; $x = x + 1 = 4$

$x = 4; \chi_x = 5$

1. $5 \notin B$
2. Activity 5 has no predecessor call activities
3. $V' = 1 - 1 = 0$

   $nc'_{P_6} = 0$ and $nc'_{P_8} = 1$

   $nc'_{P_6} = 0$ and $P_6 \backslash (B \cup \chi_x) = 0$: $re'_{P_6} = re_{P_6} - r_6^\sigma = 0 - (-2) = 2$

   $V'' = 1 - 1 + 2 = 2$

   $nc'_{P_8} \leq V''$

4. $\beta_3 = 5$

$B = \{2, 4, 5\}$

$nc'''_{P_3} = nc_{P_3} = 0$ $\quad$ $re'''_{P_3} = re_{P_3} = 1$

$nc'''_{P_6} = nc_{P_6} = 1$ $\quad$ $re'''_{P_6} = re_{P_6} = 0$

$nc'''_{P_8} = nc_{P_8} = 2$ $\quad$ $re'''_{P_8} = re_{P_8} = 0$

$nc_{P_3} = nc'_{P_3} = 0$ $\quad$ $re_{P_3} = re'_{P_3} = 1$

$nc_{P_6} = nc'_{P_6} = 0$ $\quad$ $re_{P_6} = re'_{P_6} = 2$

$nc_{P_8} = nc'_{P_8} = 1$ $\quad$ $re_{P_8} = re'_{P_8} = 0$

$V''' = 1$ and $V = 2$

$x = 1$

$b = 4$

$\underline{\mathbf{b = 4}}$

$x = 1; \chi_x = 7$

1. $7 \notin B$
2. Activity 7 has no predecessor call activities
3. $V' = 2 - 1 = 1$

   $nc'_{P_8} = 0$

   $re'_{P_8} = re_{P_8} - r_8^{\sigma} = 0 - (-1) = 1$

   $V'' = 2 - 1 + 1 = 2$
4. $\beta_4 = 7$

   $B = \{2, 4, 5, 7\}$

   $nc'''_{P_3} = nc_{P_3} = 0$ $\quad$ $re'''_{P_3} = re_{P_3} = 1$

   $nc'''_{P_6} = nc_{P_6} = 0$ $\quad$ $re'''_{P_6} = re_{P_6} = 2$

   $nc'''_{P_8} = nc_{P_8} = 1$ $\quad$ $re'''_{P_8} = re_{P_8} = 0$

   $nc_{P_3} = nc'_{P_3} = 0$ $\quad$ $re_{P_3} = re'_{P_3} = 1$

   $nc_{P_6} = nc'_{P_6} = 0$ $\quad$ $re_{P_6} = re'_{P_6} = 2$

   $nc_{P_8} = nc'_{P_8} = 0$ $\quad$ $re_{P_8} = re'_{P_8} = 1$

   $V''' = 2$ and $V = 2$

   $x = 1$

   $b = 5$

$b = (card_x + 1) = 4 + 1 = 5$. The algorithm stops with the precedence and spatial resource call activity feasible list $\beta = (4, 2, 5, 7)$.

Now we use this feasible call activity list to determine μ, the precedence and spatial resource feasible priority list. We hereby start from our original priority list $\lambda = (7, 4, 2, 1, 5, 3, 8, 6, 9)$. We initialize $M = \{0\}$, $m = 1$, $l = 1$, $b = 1$ and $V = 2$.

$\underline{\mathbf{m = 1}}$

$m \neq n$

$l = 1; \lambda_l = 7$

1. $7 \notin M$
2. $V' = 2 - 1 = 1 \geq 0$

3a. All predecessors of activity 7 are in M

3b. Activity 7 belongs to X and is different from $\beta_1 = 4$;
$l = l + 1 = 2$

$l = 2; \lambda_l = 4$

1. $4 \notin M$
2. $V' = 2 - 1 = 1 \geq 0$

3a. All predecessors of activity 4 are in M

3b. Activity 4 belongs to X and equals $\beta_1 = 4$, so activity 4 will be added to list μ

4. $\mu_1 = 4$
$M = \{0, 4\}$
$V = 1$
Activity 4 belongs to X thus $b = b + 1 = 2$
$l = 1$
$m = m + 1 = 2$

$\underline{\mathbf{m = 2}}$

$l = 1; \lambda_l = 7$

1. $7 \notin M$
2. $V' = 1 - 1 = 0 \geq 0$

3a. All predecessors of activity 7 are in M

3b. Activity 7 belongs to X and is different from
$\beta_2 = 2$; $l = l + 1 = 2$

$l = 2; \lambda_l = 4$

1. $4 \in M$; $l = l + 1 = 3$

$l = 3; \lambda_l = 2$

1. $2 \notin M$
2. $V' = 1 - 1 = 0 \geq 0$

3a. Not all predecessors of activity 2 are in M; $l = l + 1 = 4$

$l = 4; \lambda_l = 1$

1. $1 \notin M$
2. $V' = 1 - 0 = 1 \geq 0$

3a. All predecessors of activity 1 are in M

3b. Activity 1 does not belong to X

4. $\mu_2 = 1$
$M = \{0, 1, 4\}$
$V = 1$
$l = 1$
$m = m + 1 = 3$

$\underline{\mathbf{m = 3}}$

$l = 1; \lambda_l = 7$

Activity 7 remains ineligible

$l = 2; \lambda_l = 4$

1. $4 \in M$; $l = l + 1 = 3$

$l = 3; \lambda_l = 2$

1. $2 \notin M$
2. $V' = 1 - 1 = 0 \geq 0$

3a. All predecessors of activity 2 are in M

3b. Activity 2 belongs to X and equals $\beta_2 = 2$

4. $\mu_3 = 1$

$M = \{0, 1, 2, 4\}$

$V = 0$

Activity 2 belongs to X thus $b = b + 1 = 3$

$l = 1$

$m = m + 1 = 4$

$\underline{\mathbf{m = 4}}$

$l = 1; \lambda_l = 7$

Activity 7 remains ineligible

$l = 2; \lambda_l = 4$ and $l = 3; \lambda_l = 2$ and $l = 4; \lambda_l = 1$

1. $4, 2, 1 \in M$

$l = 5; \lambda_l = 5$

1. $5 \notin M$
2. $V' = 0 - 1 = -1 < 0$; $l = l + 1 = 6$

$l = 6; \lambda_l = 3$

1. $3 \notin M$
2. $V' = 0 - (-1) = 1 \geq 0$

3a. All predecessors of activity 3 are in M

3b. Activity 3 does not belong to X

4. $\mu_4 = 3$

$M = \{0, 1, 2, 3, 4\}$

$V = 1$

$l = 1$

$m = m + 1 = 5$

$\underline{\mathbf{m = 5}}$

Activity 7 remains ineligible

Activities 4, 2 and 1 already belong to M

$l = 5; \lambda_l = 5$

1. $5 \notin M$
2. $V' = 1 - 1 = 0 \geq 0$

3a. All predecessors of activity 5 are in M

3b. Activity 5 belongs to X and equals $\beta_3 = 5$

4. $\mu_5 = 5$
$M = \{0, 1, 2, 3, 4, 5\}$
$V = 0$
Activity 5 belongs to X thus $b = b + 1 = 4$
$l = 1$
$m = m + 1 = 6$

$\underline{\mathbf{m = 6}}$

$l = 1; \lambda_l = 7$
1. $7 \notin M$
2. $V' = 0 - 1 = -1 < 0$; $l = l + 1 = 2$

Activities 4, 2, 1, 5 and 3 already belong to M

$l = 7; \lambda_l = 8$
1. $8 \notin M$
2. $V' = 0 - (-1) = 1 \geq 0$
3a. Not all predecessors of activity 8 are in M; $l = l + 1 = 8$

$l = 8; \lambda_l = 6$
1. $6 \notin M$
2. $V' = 0 - (-2) = 2 \geq 0$
3a. All predecessors of activity 6 are in M
3b. Activity 6 does not belong to X
4. $\mu_6 = 6$
$M = \{0, 1, 2, 3, 4, 5, 6\}$
$V = 2$
$l = 1$
$m = m + 1 = 7$

Continuing the procedure results in the precedence and spatial resource feasible list $\mu = (4, 1, 2, 3, 5, 6, 7, 8, 9)$.

5.4 The schedule generation scheme

In project scheduling literature, two basic schedule generation schemes are often used: the parallel schedule generation scheme (parallel SGS) and the serial schedule generation scheme (serial SGS). Both can be applied to a priority list in order to build a resource and precedence feasible project schedule. We will illustrate both schemes on the project network of Figure 10. For illustrative purposes, we assume that no other renewable resources are required and that all activities have a single unit duration.

The *parallel* SGS iterates over time and starts at each decision time t as many unscheduled activities as possible in accordance with the precedence and resource constraints. The priority list dictates the order in which the activities are considered.

If we apply the parallel SGS to the list $\mu = (4, 1, 2, 3, 5, 6, 7, 8, 9)$ for the example project, we start by scheduling activities 4 and 1 at time 0. The next two activities in μ, activities 2 and 3, cannot be scheduled at time 0, because activity 1 has not been finished. Activity 5 is the next activity to be started at time 0. The remaining activities in μ cannot be started because either a predecessor has not yet finished (activities 6, 8 an 9 are in this case) or the required resources are not available (activity 7). Despite the feasible priority list, a deadlock still occurs. We are in a situation in which activity 2 is waiting for activity 6 to release spatial resources and activity 6 is waiting for activity 2 (and 3) to be scheduled. The parallel SGS is not applicable for generating spatial resource schedules since it does not guarantee a deadlock-free schedule.

The *serial* SGS selects in each iteration the next unscheduled activity in the priority list and assigns the first possible starting time that satisfies both the precedence and resource constraints.

When we apply this SGS to $\mu = (4, 1, 2, 3, 5, 6, 7, 8, 9)$, we first schedule activities 4 and 1 to start at time 0. Activity 1 is shown unshaded in Figure 11 because it does not require spatial resources. Activity 4 requests a spatial resource unit. Activity 2 can be scheduled when its predecessor activity 1 ends, i.e. at time 1. Next, activity 3 starts at time 2 and releases the spatial resource unit that activity 2 called. The next activity in priority list μ, activity 5, can only be scheduled at time 3 when activity 3 releases its spatial resource unit. The schedule is completed as shown in Figure 11.

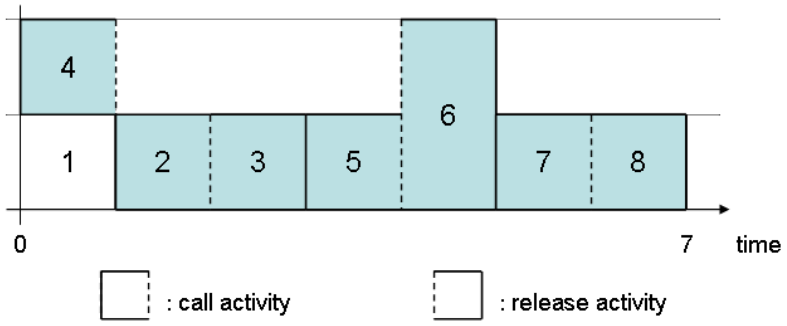

Figure 11: Resource profile for example project

6 Computational experiment

We extended the J30 and J120 instance sets of PSPLIB (Kolisch and Sprecher (1997)) to generate networks with spatial resources. For each instance we

kept the first two renewable resources - i.e. we preserved the total availability of the resource and the requirements of the activities - and we added one or two spatial resources with availability a_k^σ and a certain number of call and release activities. Furthermore, we made a distinction in the spatial resource intensity by using 3, 6 or 9 different call activities per spatial resource. For each J30 and J120 instance, we generated six different spatial resource networks: three with one spatial resource (setting 1SR) and 3, 6 or 9 call activities, and three networks with two spatial resources (setting 2SR), each having 3, 6 or 9 call activities. So for networks with two spatial resources and 9 call activities, there can be in theory 18 call activities in total. However, this number cannot always be obtained because of two reasons. First of all, not every network allows to generate 9 different call activities per spatial resource because of the inherent structure of the network. Second, an activity can be a call activity for the first as well as the second spatial resource type. It is easier to reach the maximum number of call activities for the J120 than for the J30 instances.

The fifteen priority rules shown in Table 3 are tested in the experiment. A detailed explanation for each of them can be found in Demeulemeester and Herroelen (2002). We report on the results obtained on 425 feasible networks for each combination of the total number of activities (30 or 120), the number of spatial resources (1SR or 2SR) and the number of call activities (approximately 3, 6 or 9).

7 Computational results

All computational results have been obtained on a personal computer equipped with a Pentium D 2.8 GHZ processor. All algorithms have been coded in C.

7.1 Makespan performance of the serial priority rules

The average makespan over six groups of 425 network instances (namely one spatial resource (1SR) and 3 call activities (3call), 1SR and 6call, 1SR and 9call, 2SR and 3call, 2SR and 6call and 2SR and 9call) obtained by the serial SGS equipped with each of the 15 priority rules listed in Table 3 are presented in Figure 12. The latest start time priority rule (LST) - ranking among the best priority rules for the classical RCPSP - consistently yields the smallest makespan. It seems fair to conclude that the priority rules which perform well for the RCPSP also perform well for the spatial RCPSP.

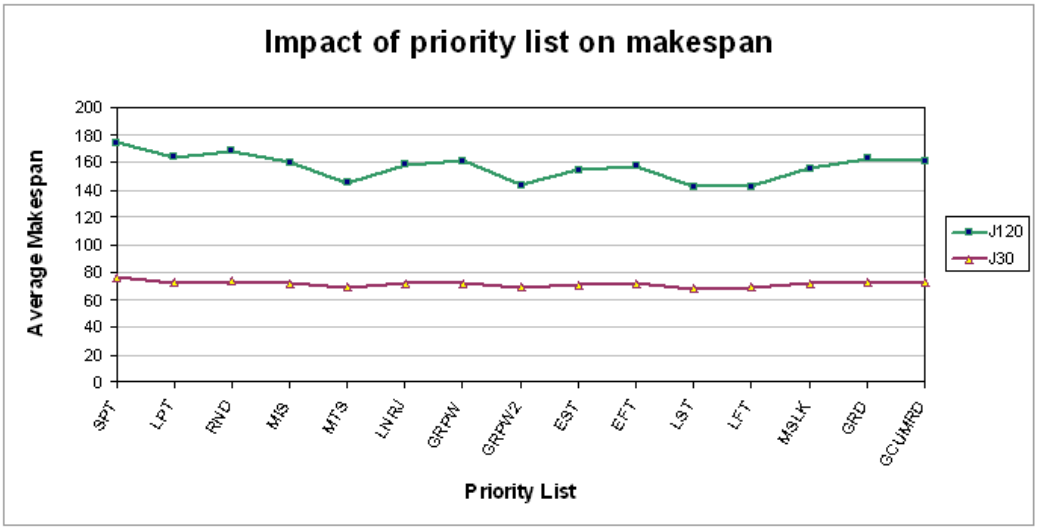

Figure 12: Comparison of makespan performance

7.2 CPU time requirements

In the following, we report on the computational requirements of the serial SGS equipped with the LST rule for the networks with 120 activities and two spatial resources, the most difficult instances to solve. The results are given in Table 4 and are plotted in Figure 13. The mentioned times are the total times (in seconds) needed to solve all 425 networks.

The computational times are very small. The results obtained by the second dominance rule (the addition of the extra control array) are much better than the results obtained by the basic algorithm without the dominance rule. For instances with three call activities for each spatial resource, the use of the dominance rule results in an increase of the CPU time needed. For the networks with the same characteristics, but with approximately 9 call activities, we observe a very large increase in the computation time needed for the 425 networks by the algorithm when the second dominance rule is not used.

8 Conclusions

In this paper, we developed a solution procedure for solving spatial resource-constrained project scheduling problems. Our spatial resource setting is different from previous settings described in the literature. We consider project networks that may require both multiple renewable resources in arbitrary amounts and an arbitrary number of spatial resources in more than a single

SPT	Shortest Processing Time
LPT	Longest Processing Time
RND	Random
MIS	Most number of Immediate Successors
MTS	Most number of Total Successors
LNRJ	Least number of Non-Related Jobs
GRPW	Greatest Rank Positional Weight (only immediate successors taken into account)
GRPW2	Greatest Rank Positional Weight 2 (all successors taken into account)
EST	Earliest Start Time
EFT	Earliest Finish Time
LST	Latest Start Time
LFT	Latest Finish Time
MSLK	Minimum Slack
GRD	Greatest Resource Demand
GCUMRD	Greatest Cumulative Resource Demand

Table 3: Priority Rules

	dominance rule	no dominance rule
J120 - 2SR - 3st	3.34	3.30
J120 - 2SR - 6st	3.78	3.78
J120 - 2SR - 9st	5.19	7171.96

Table 4: Computation times for LST for solving 425 networks (in seconds)

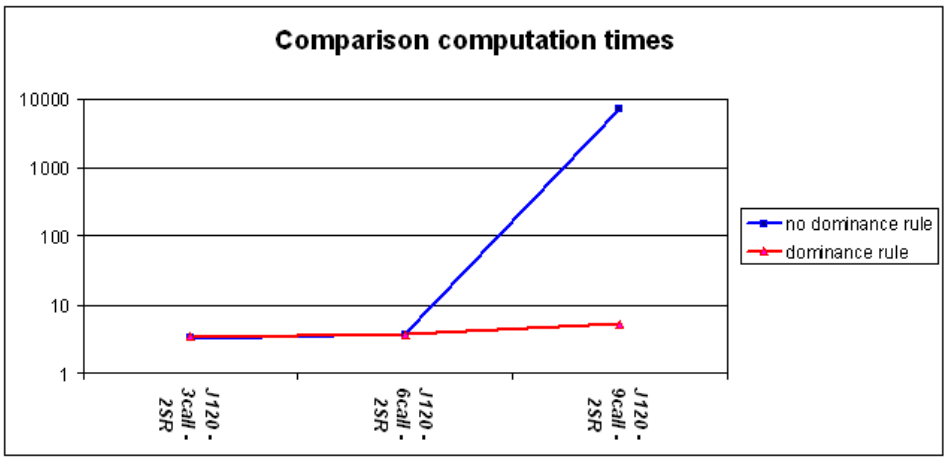

Figure 13: Comparison computation times for LST

unit. Activities within a group of activities that occupy spatial resource unit types, may serve as call or release activities that reserve, respectively release, other spatial resource types. We also drop the requirement that the spatial resources should be adjacent. The objective is to schedule the project activities, satisfying both the precedence and resource constraints, under the minimum makespan objective.

We have demonstrated that spatial resources need special treatment in that treating them as renewable resources may lead to infeasibilities and/or low quality schedules. We have presented a new heuristic solution method for the problem based on the application of a schedule generation scheme that operates on a priority list of activities. We have developed two algorithms for transforming a given priority list into a precedence and resource feasible priority list, that if combined with a serial SGS, avoids the occurrence of circular waits for the spatial resources. We showed that the parallel SGS is not applicable for projects with spatial resources because despite the use of a precedence and resource feasible priority list, deadlocks can still occur.

Our computational experiment with 15 different priority rules on adapted J30 and J120 PSPLIB instances revealed that the priority rules which perform well for the classical RCPSP also do well on the spatial RCPSP in very small computation times.

References

Blazewicz, J., Lenstra, J. and Kan, A. R. (1983). Scheduling subject to resource constraints - classification and complexity. *Discrete Applied Math-*

emetics, 5, pp 11–24.

de Boer, R. (1998). *Resource-Constrained Multi-Project Management - A Hierarchical Decision Support System*. PhD thesis. University of Twente, The Netherlands.

Demeulemeester, E. and Herroelen, W. (2002). *Project scheduling - A research handbook*. Vol. 49 of *International Series in Operations Research & Management Science*. Kluwer Academic Publishers, Boston.

Duin, C. and Van der Sluis, E. (2006). On the complexity of adjacent resource scheduling. *Journal of Scheduling*, 9(1), pp 49–62.

Herroelen, W., De Reyck, B. and Demeulemeester, E. (2000). On the paper "resource-constrained project scheduling: notation, classification, models and methods" by brucker et al.. *European Journal of Operational Research*, 128(3), pp 679–688.

Hurink, J., Kok, A. and Paulus, J. (2006). Decomposition method for project scheduling with spatial resources. *Research report*. University of Twente, The Netherlands.

Kolisch, R. (1996a). Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management*, 14, pp 179–192.

Kolisch, R. (1996b). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, 90, pp 320–333.

Kolisch, R. and Drexl, A. (1996). Adaptive search for solving hard project scheduling problems. *Naval Research Logistics*, 43(1), pp 23–40.

Kolisch, R. and Sprecher, A. (1997). PSPLIB - a project scheduling library. *European Journal of Operational Research*, 96, pp 205–216.

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

Paulus, J. and Hurink, J. (2006). Adjacent-resource scheduling - why spatial resources are so hard to incorporate. *Electronic Notes in Discrete Mathematics*, 25, pp 113–116.

Schatteman, D., Herroelen, W., Van de Vonder, S. and Boone, A. (2006). A methodology for integrated risk management and proactive scheduling of construction projects. *Research Report KBI 0622*. Department of Decision Sciences and Information Management, Katholieke Universiteit Leuven, Belgium.