Lower Bounds for the Multi-skill Project Scheduling Problem with Hierarchical Levels of Skills

Odile Bellenguez and Emmanuel Néron

Laboratoire d'Informatique de l'Université de Tours,
64 avenue Jean Portalis, 37200 Tours, France
odile.bellenguez@etu.univ-tours.fr
emmanuel.neron@univ-tours.fr

Abstract. In this paper, we introduce an extension of the classical Resource-Constrained Project Scheduling Problem: the Multi-skill Project Scheduling Problem. We consider a project made up of activities that must be implemented by a staff: every member of this staff masters one or more skill(s). An activity needs a given amount of each skill with a fixed minimum level of mastering. For each unit of a skill needed, we have to assign an employee who masters the required level of this skill during the whole processing time of the activity. The objective is to minimize the duration of the project, i.e. the makespan. We introduce here two lower bounds used to evaluate the minimum duration.

1 Introduction

This paper deals with the Multi-skill Project Scheduling Problem (MSPSP) with hierarchical levels of skill. In this problem, there is a project to schedule, in which activities need to be done by staff members who master specific skills at specific level of ability. It can be seen as an extension of the classical Resource-Constrained Project Scheduling Problem with multiple modes of execution (MM-RCPSP), but in our problem the number of modes allowed for each activity corresponds to the number of subsets of staff members that are able to satisfy needs, and this number can be very large. This is the reason why methods used for the MM-RCPSP, for example in [7], [13], [16], [12], [18], [23], cannot be used directly for the MSPSP. Here the goal is to find some efficient lower bounds that are not too time-consuming, in order both to evaluate heuristic methods (as for example in [6]) and to be used in a branch-and-bound method. Section 2 presents the problem; Section 3 is devoted to lower bounds, then we show experimental results (Section 4), before concluding in Section 5.

2 Multi-skill Project Scheduling

The MSPSP is a model that can be applied to different cases of project management. We focus on a particular case of multi-skill project scheduling that appears when we have to take into account different levels of skill abilities.

2.1 General Case

As in the classical project scheduling problem (RCPSP, MM-RCPSP), we want to schedule a project in a minimum elapsed time, respecting resource and precedence constraints. A project is made up of a set of activities A_i, $i \in \{0, \ldots, n\}$, that represent all the steps of the project. Each activity has to be processed without preemption. Between the activities of the project, there exist precedence relations (A_i, A_j) that can be modelled using an activity-on-node graph $G = (A, E, d)$. In this precedence graph, we include the dummy source A_0 and the dummy sink A_n that represent the beginning and the end of the project, respectively. Each activity has given needs of resources. In our case, the resources are staff members that have to be assigned to activities.

Moreover, the resources, which are staff members P_m, $m \in \{0, \ldots, M\}$, cannot be chosen arbitrarily. To satisfy each need of an activity A_i, we have to assign a person according to his/her skill that must match the required skill during the whole processing time p_i of activity A_i. Actually, a staff member can master or not each skill needed in the project. Thus, to schedule an activity we have to choose among all the staff members who possess the required skill for each need. Besides that, the employees may not be available all along the time horizon; this means we are allowed to assign a person for the need of an activity A_i starting at time t_i only if he/she is available during the total duration of the activity, i.e. during $[t_i, t_i + p_i[$.

2.2 Hierarchical Levels of Skills

There has been considerable research on workforce planning and project scheduling. For example, the nurse rostering problem [8] is well studied: it assigns employees to satisfy all the needs on all the shifts, trying to find a fair solution for everybody. The course timetabling problem has also been studied (for example in [1]). It consists in finding a timetable for each lesson, respecting teachers availabilities and the rooming constraints. The authors in [14] present a problem related to course scheduling but to minimize the total cost. The paper [2] treats simultaneously the problem of minimizing the project duration and the associated manpower cost. Although the notion of skill has been studied with respect to some workforce planning problems (as in [25]), to the best of our knowledge, very few papers deal with hierarchical levels of skills in the field of project scheduling [15].

For the MSPSP with hierarchical levels of skills, we evaluate for each employee the level l of quality he/she guarantees regarding skill S_k according to his/her experience. In the same way, we evaluate for each activity its level of difficulty. Then, we can deduce S_k^l, the level l of mastering skill S_k that will be required, and the number $b_{i,k}^l$ of employees that will be required for level l of this skill S_k to process A_i. So, we have to assign to each need somebody that masters this skill at least at the required level. We also know the total number of persons required, for each level of each skill, allowing this number to be equal to zero.

The notation used in this paper is defined in Table 1.

Table 1. Input data and auxiliary notation

Activity data	
n	the index of the last activity,
A_i, $i \in \{0,\ldots,n\}$	the set of activities of the project: A_0 is dummy start node and A_n the dummy end of the project,
p_i, $i \in \{0,\ldots,n\}$	the processing time of activity A_i,
$(A_i, A_j) \in E$	if there exists a precedence relation between A_i and A_j,
$G = (A, E, d)$	the precedence graph.
Resource data	
K	number of skills,
L	number of levels per skill,
M	number of staff members,
S_k, $k \in \{0,\ldots,K\}$	set of skills: k is the number of the skill,
S_k^l, $k \in \{0,\ldots,K\}$, $l \in \{0,\ldots,L\}$	set of levels of skills; k is number of the skill and l the level (1 is first level of mastering skill and L is the highest level),
P_m, $m \in \{0,\ldots,M\}$	staff members,
$S_{m,k} = l$, $m \in \{0,\ldots,M\}$, $k \in \{0,\ldots,K\}$	the maximum level l at which P_m can do S_k,
to simplify: $S_{m,k}^{l'} = 1$	$\forall l' \leq l$, if P_m can do S_k at level l, 0 otherwise,
$A(P_m, t)$, $m \in \{0,\ldots,M\} = 1$ $t \in \{0,\ldots,T_{\max}\}$	if P_m is available at time t, 0 otherwise,
$b_{i,k}^l$, $i \in \{0,\ldots,n\}$ $k \in \{0,\ldots,K\}$, $l \in \{0,\ldots,L\}$	number of persons able to do S_k at level l, required to execute A_i.
Auxiliary notation	
r_i	release date of activity A_i,
$\tilde{d}_i$	deadline of activity A_i,
t_i	starting time of activity A_i,
$A(P_m, t_1, t_2) = \sum_{t=t_1}^{t_2-1} A(P_m, t)$, $m \in \{0,\ldots,M\}$	total time P_m is available between t_1 and t_2.

2.3 Example and Integer Linear Program

The MSPSP model can be applied to some project scheduling problems that arise in the software development industry, where employees are programmers, analysts, designers, etc [15]. In this section we give an example of such a problem. The project to be implemented is made up of four activities linked by the precedence relationship presented in Figure 1.

Needs of the activities are summarized in Table 2 and skills of employees are presented in Table 3. There are two levels per skill. We are interested in schedules that minimize the makespan of the project. Finding a solution consists in fixing starting times of all the activities of the project and assigning a subset of staff

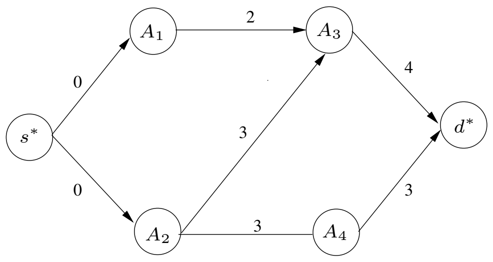


Fig. 1. Precedence graph of the project of the example

Table 2. Needs of activities of the example

	Skill S_1		Skill S_2		Skill S_3		Skill S_4	
$b^l_{i,k}$	S^1_1	S^2_1	S^1_2	S^2_2	S^1_3	S^2_3	S^1_4	S^2_4
A_1	1	1	0	0	1	0	1	0
A_2	0	0	0	1	0	0	1	0
A_3	0	0	1	0	2	0	0	0
A_4	1	0	0	0	0	1	0	0

Table 3. Skills of employees of the example

	Skill S_1		Skill S_2		Skill S_3		Skill S_4		Unavailability period(s)	
	S^1_1	S^2_1	S^1_2	S^2_2	S^1_3	S^2_3	S^1_4	S^2_4	no.	(start, end)
P_0	1	1	1	1	0	0	0	0	1	$(2, 4)$
P_1	0	0	1	1	1	1	0	0	0	–
P_2	0	0	1	0	1	1	1	1	0	–
P_3	1	0	1	0	1	0	0	0	2	$(2, 3)$; $(6, 8)$
P_4	1	0	0	0	0	0	1	1	0	–
P_5	1	0	0	0	1	1	0	0	1	$(7, 10)$

members to each activity, according to their needs. We present a feasible solution for this problem in Figure 2.

The MSPSP can be modelled by the integer LP formulation below, where

– $x_{i,m,t} = 1$ if P_m begins to work for A_i at time t, 0 otherwise
– $\delta^l_{i,m,k} = 1$ if P_m does S_k at level l for A_i, 0 otherwise:

$$\forall (i,j) \in E, \frac{\sum_{m=0}^{M} \sum_{t=0}^{T_{\max}} x_{i,m,t} \cdot t}{\sum_{k=0}^{K} \sum_{l=0}^{L} b^l_{i,k}} + p_i \leq \frac{\sum_{m=0}^{M} \sum_{t=0}^{T_{\max}} x_{j,m,t} \cdot t}{\sum_{k=0}^{K} \sum_{l=0}^{L} b^l_{j,k}}, \tag{1}$$

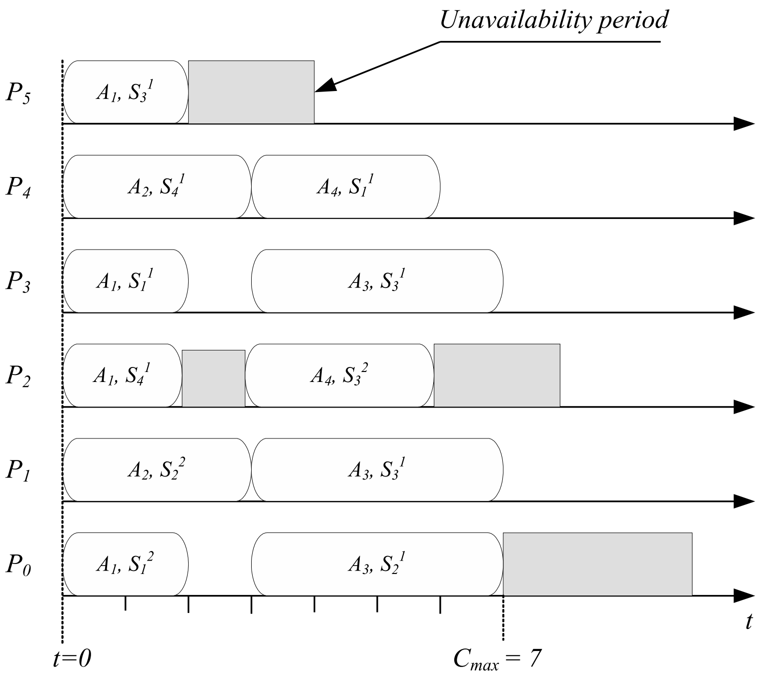


Fig. 2. One feasible solution for the example

$$\forall i \in \{1,\dots,n\}, \forall m \in \{1,\dots,M\}, \sum_{t=0}^{T_{\max}} x_{i,m,t} \leq 1\,, \tag{2}$$

$$\forall i \in \{1,\dots,n\}, \forall m \in \{1,\dots,M\}, \sum_{t=0}^{T_{\max}} x_{i,m,t} \cdot t \leq \frac{\sum_{h=0}^{M}\sum_{t=0}^{T_{\max}} x_{i,h,t} \cdot t}{\sum_{k=0}^{K}\sum_{l=0}^{L} b_{i,k}^{l}}\,, \tag{3}$$

$$\begin{aligned}&\forall i \in \{1,\dots,n\}, \forall m \in \{1,\dots,M\}, \forall k \in \{1,\dots,K\},\\ &\forall l \in \{1,\dots,L\}, \delta_{i,m,k}^{l} \leq S_{m,k}^{l}\,,\end{aligned} \tag{4}$$

$$\forall i \in \{1,\dots,n\}, \sum_{m=0}^{M}\sum_{t=0}^{T_{\max}} x_{i,m,t} \cdot t = \sum_{k=0}^{K}\sum_{l=0}^{L} b_{i,k}^{l}\,, \tag{5}$$

$$\forall i \in \{1,\dots,n\}, \sum_{m=0}^{M} \delta_{i,m,k}^{l} = b_{i,k}^{l}\,, \tag{6}$$

$$\forall m \in \{1,\dots,M\}, \sum_{i=0}^{n}\sum_{d=t-p_i+1}^{t} x_{i,m,d} \leq 1\,, \tag{7}$$

$$\forall i \in \{1,\dots,n\}, \forall m \in \{1,\dots,M\}, \sum_{t=0}^{T_{\max}} x_{i,m,t} = \sum_{k=0}^{K}\sum_{l=0}^{L} \delta_{i,m,k}^{l}\,, \tag{8}$$

$$\min C_{\max} = \frac{\sum_{t=0}^{T_{\max}} \sum_{m=0}^{M} x_{n,m,t} \cdot t}{\sum_{k=0}^{K} \sum_{l=0}^{L} b_{n,k}^{l} + p_n} .$$

Equation (1) ensures that two activities that respect the precedence relation do not overlap. Equation (2) allows staff members to be assigned to an activity only once. Equation (3) obliges all the staff members assigned to a common activity to start at the same time. According to Equation (4), a person cannot be assigned to a need if he/she does not master the level of the skill needed. Equation (5) ensures that the number of persons that do an activity is equal to the sum of the need of this activity. Equation (6) obliges the number of persons that are assigned to a level of a skill to be equal to the needs for this level of this skill, for each activity. Equation (7) ensures that a person will not start an activity during the whole processing time of an activity he/she is already assigned to. Finally, Equation (8) ensures that a person participates in all the activities he/she starts.

This model is a time-indexed one, and in the general case a simple relaxation of this kind of model does not provide good lower bounds as it has been demonstrated for RCPSP. It is necessary to use constraint programming and /or a cutting plane technique [4], [10] to have good bounds. But these methods are too time-consuming: our goal is to develop an efficient lower bound that may be used both to compute a global lower bound and that may be used at each node of a branch-and-bound method. Thus, we focus on other types of lower bound formulation.

2.4 Literature Review

The MSPSP is a kind of problem very close to the classical MM-RCPSP [24], [12], [13], [7] and can be considered as a particular case of it. Actually, in the multi-mode RCPSP, every activity has different possible modes of execution. One mode is defined by a given quantity of each resource and an associated processing time, and for each activity there exists a finite number of modes (less than ten in the classical instances). If we apply this model to our problem, a mode corresponds to a feasible subset of staff members that can be assigned to an activity according to the required skills, while the processing time remains the same for all modes.

To the best of our knowledge, most of the methods that exist for the MM-RCPSP are based on explicit enumeration of all possible modes (even for computing lower bounds the authors of [7] use variables $\xi_{i,m}$ that are equal to 1 if and only if A_i is processed in mode m). For the MSPSP the number of different modes, i.e. the number of different subsets of staff members that can be assigned to an activity, may become huge. For example if we want to solve an instance with three skills with only one level for each skill, and 10 employees, some activities may have more than 1,000 modes each. Then, according to the LP formualtion proposed by [7], the number of variables grows quickly and the problem becomes intractable even for small size instances.

The problem we want to solve can also be seen as a particular case of the multi-resource shop [9], where every activity has specific fixed needs for each type of resource, a type of resource being defined by a set of machines. The difficulty is that the sets of resources corresponding to each different type are not disjoint. In our problem the resources are always the employees and the set of resources is those who master the required level of the skill needed.

3 Lower Bounds

As mentioned previously, the lower bounds we are looking for must provide a good trade-off between their efficiency and their time-consumption because they will be used in a branch-and-bound method at each node of the search tree. We have adapted two such bounds that have been proposed for RCPSP. Notice that the two bounds that we have adapted, namely from [22] and [3], are two of the most efficient ones to be used in branch-and-bound methods, considering respectively instances with high ratio of disjunction between activities and instances with few disjunctions between activities (see [11] and [3] for further details).

Here we present two classical lower bounds for the RCPSP that we adapt for the MSPSP. Both of them are destructive, which means that we fix a value D we want to test and either we detect a contradiction, i.e. we prove that the project cannot end before D and then $D+1$ is a valid lower bound, or we cannot detect an infeasibility and then D is decreased. Practically, we use binary search between the value given by the critical path and a valid upper bound [6].

For these two methods it is necessary that each activity has a time window. So, first of all, we use the precedence graph to compute the release date r_i of each activity A_i, according to the release dates and the durations of all its predecessors. Setting $r_i = L(0, A_i)$ as the longest path from the source to the activity A_i, then the value D, which is a lower bound for the project duration we are testing, is propagated from the last activity to the first in order to get deadlines $\tilde{d}_i(D)$ for the activities. Here $\tilde{d}_i(D)$ is equal to $L(A_i, A_n) - p_i$, the longest path from an activity to the sink of the project. Notice that these two lower bounds are mainly based on $[r_i, \tilde{d}_i(D)]$, thus they can be applied even if generalized precedence relationships are considered.

Notice that we have adapted these two lower bounds to MSPSP because their efficiency on a large range of RCPSP instances has been proved.

3.1 Graph of Compatibility

The first lower bound that we have adapted from RCPSP was introduced by [22] and is based on the notion of a block. A block is a feasible subset of activities that can be processed simultaneously, i.e. violating neither the resource constraints nor the precedence constraints. In the MSPSP framework, the precedence constraints can be simply verified, and checking whether a subset J of activities does not violate the resource constraints if its activities are in progress at the same time can be done using a max-flow formulation (see Figure 5). Thus, the bound of Mingozzi can be adapted to the MSPSP.

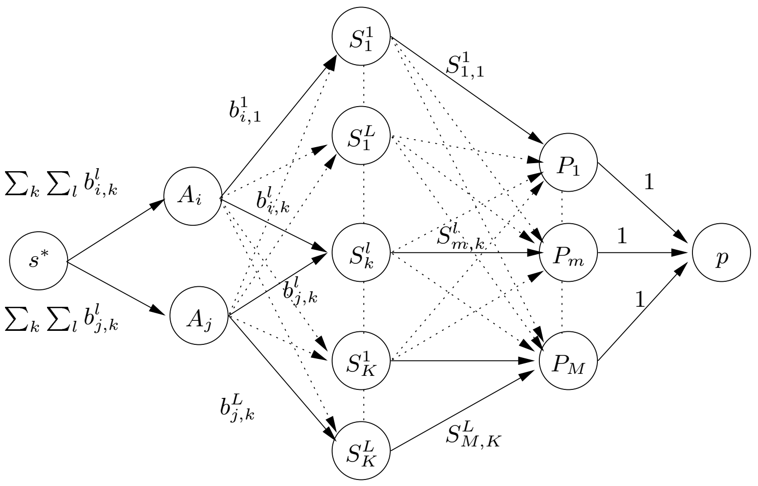


Fig. 3. Graph G_1 used for testing the needs of A_i and A_j

More precisely, we have adapted the third bound (LB_3) of [22] that is based on the use of disjunction between a couple of activities due to the resource constraint. For each couple of activities A_i and A_j (except the dummy activities), we test if it is possible for them to be in progress at the same time. First of all, we have to check that there is no precedence relationship between the two activities. Then we have to test if their time windows overlap, i.e. if there exists an intersection between $[r_i,\tilde{d}_i(D)]$ and $[r_j,\tilde{d}_j(D)]$, and finally we have to check the resource constraints. To find out if there are enough resources to schedule the two activities A_i and A_j during a common period, we solve the corresponding assignment problem using a max-flow formulation. The graph $G_1 = (X_1, F, c)$, $X_1 = \{A_i, A_j\} \bigcup \{S_k^l | \forall k \in \{0, \dots, K\}, \forall l \in \{0, \dots, L\}\} \bigcup \{P_m | \forall m \in \{0, \dots, M\}\}$ in which we search for a maximum flow is presented in Figure 3.

In this graph, the two nodes of the first column symbolize the activities A_i and A_j we are testing. The nodes of the second column represent each level of each skill S_k^l needed by the activities, and the third column of nodes corresponds to the staff members P_m that can be assigned to one of those activities. Edges have maximum capacity: for those from activity A_i to a level of a skill S_k^l this capacity is equal to the number of required staff members: $b_{i,k}^l$. From a staff member to the sink (p) this capacity is equal to 1 in order to limit a person to do one thing at a time. The edge between a level of a skill S_k^l and a person P_m exists if the person is able to do this level of this skill, i.e. if $S_{m,k}^l = 1$. We compute the maximum flow in order to compare it to the sum of the needs of the two activities ($\sum_{k=0}^{K} \sum_{l=0}^{L} (b_{i,k}^l + b_{j,k}^l)$) and conclude if the two activities can be in progress at the same time.

Property 1. If the maximum flow found in graph G_1 is strictly less than the sum of the needs of the two activities ($\sum_{k=0}^{K} \sum_{l=0}^{L} (b_{i,k}^l + b_{j,k}^l)$), there is a contradiction for the activities A_i and A_j to be in progress at the same time, and we can conclude that those two activities have to be scheduled in two time-intervals that do not overlap.

Proof. The proof is obvious due to the graph construction.

Once all those verifications have been done, we know exactly which couple of activities can be in progress at the same time, i.e. $(A_i,\, A_j) \in E'$. We can then build the graph of compatibility. Based on the precedence graph $G = (A, E, d)$ we define a graph $G' = (A, E')$, in which there is an edge between two activities (A_i, A_j) if no infeasibility has been detected and they can be scheduled in a common interval. We associate a weight with each node, equal to the duration p_i of the corresponding activity A_i. Then, we search for a maximum-weighted stable set in this graph, in order to determine the longest set of activities that cannot be in progress at the same time, which is a lower bound of the project duration.

But finding a maximum-weighted stable set in a graph is $\mathcal{NP}$-hard in the strong sense. This is the reason why we first try to solve it through a heuristic method as in [22]. This method is a greedy algorithm. First, we take all the nodes corresponding to the activities of the critical path, and then we add possible nodes one by one, in order of decreasing weights, until we cannot add any other node. A second way to compute the stable set is based on a MIP formulation solved with Cplex (1). Practical results show that this problem is well-solved. The model used is the following, where $x_i \in \{0, 1\}$, $x_i = 1$ if A_i is in the stable set:

$$\max \sum_{i=0}^{n} p_i \cdot x_i \qquad \text{s.t. } \forall (A_i, A_j) \in E' \; x_i + x_j \leq 1\,. \tag{9}$$

3.2 Energetic Reasoning Based Lower Bound

In [3] and [21] satisfiability tests and time-bound adjustments were introduced for the classical RCPSP. Satisfiability tests can notably be used to find if a given schedule can complete before a global deadline D and then can be used to compute a destructive lower bound. Energetic reasoning is based on the fact that in a given time interval $[t_1, t_2]$, we are able to detect if all the mandatory parts of the activities that have to be processed in this time interval can be done or not. The mandatory part of activity A_i that has to be scheduled between t_1 and t_2 is computed either by left-shifting or right-shifting the activity in its time window $[r_i, \tilde{d}_i(D)]$. Then we are sure that if there exists at least one time interval where those mandatory parts cannot be satisfied, i.e. there are not enough resources, then the lower bound can be increased to $D + 1$.

In order to use energetic reasoning, we compute all time intervals $[t_1, t_2]$, where $t_1 \in \{r_i, r_i + p_i, \tilde{d}_i(D) - p_i, \forall i \in \{1, \ldots, n\}\}$ and $t_2 \in \{r_i + p_i, \tilde{d}_i(D) - p_i, \tilde{d}_i(D), \forall i \in \{1, \ldots, n\}\}$, $t_1 < t_2$. All these time intervals are to be tested. This set of time points is a subset of those that have been proved to be relevant for bounding RCPSP [3]. They have not been proved to be relevant for the MSPSP, but the two problems are very close, so we assume that at least those time points

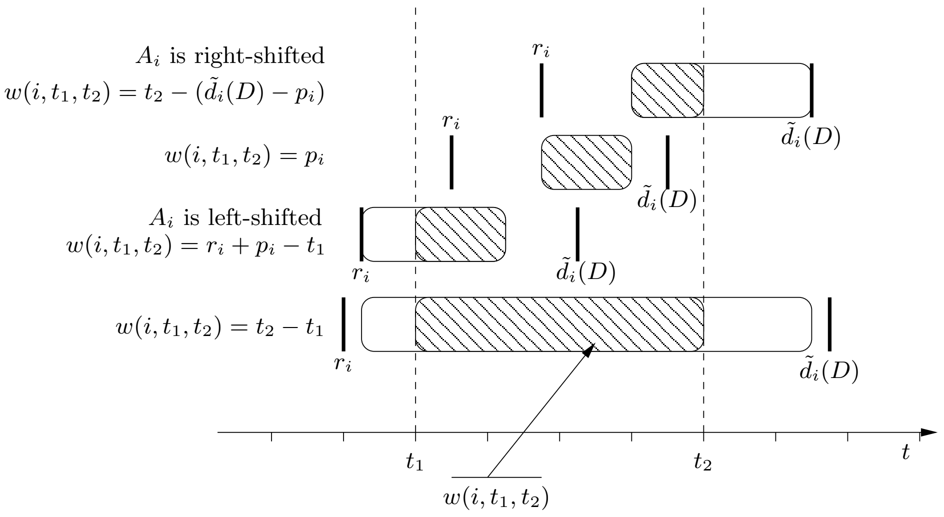


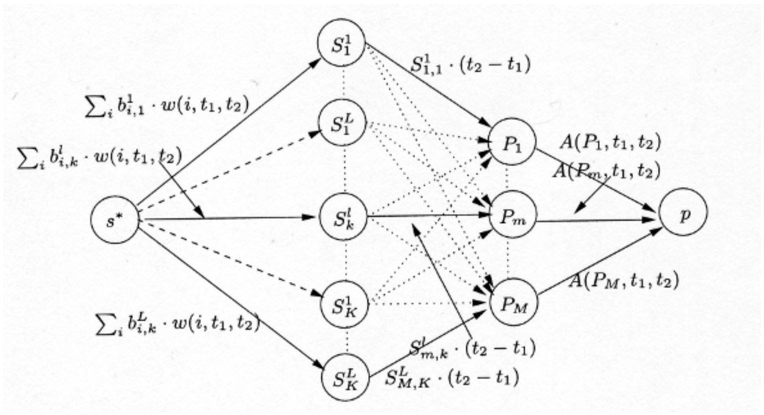
Fig. 4. Computation of the mandatory part of an activity

are interesting for our problem, and we cannot add all the possible time points because this lower bound is already as slow as acceptable, as is proved in Section 4.2.

To test the interval $[t_1, t_2]$, we have to compute the mandatory part of each activity in this interval. The mandatory part of an activity is the minimum part of this activity we have to schedule in this interval if we do not want to violate the time window of the activity. The mandatory part of A_i between t_1 and t_2, $w(i, t_1, t_2)$, is (see Figure 4)

$$w(i,t_1,t_2) = \min(\max(0, r_i + p_i - t_1), \max(0, t_2 - (\tilde{d}_i(D) - p_i)), p_i, t_2 - t_1) .$$

Once all the mandatory parts are computed we check if there are enough available resources in this interval to execute at least all these mandatory parts. As above, this problem can be modelled as an assignment problem that can be solved using a max-flow formulation. To do this, we use the graph $G_2(t_1, t_2, D)$ presented in Figure 5, where we search for a maximum flow in order to verify property 2. This graph is made of the first column of nodes that represent each level of each skill S_k^l and of the second column of nodes that corresponds to the staff members. Each edge from s to a level of a skill S_k^l has a maximum capacity equal to the mandatory parts times the number of persons needed for this level l of this skill $(\sum_{i=0}^{n} \sum_{k=0}^{K} \sum_{l=0}^{L} (w(i,t_1,t_2) \cdot b_{i,k}^l))$. Edges between levels of skills S_k^l and staff members P_m exist if the staff member masters this level of the skill, i.e. $S_{m,k}^l = 1$. These edges have the maximum capacity equal to the length of the time interval $[t_1, t_2]$. Finally, the edge between person P_m and p has the maximum capacity equal to the total time this person is available between t_1 and t_2, which is equal to $A(P_m, t_1, t_2)$.



S_1^1
S_1^L
S_k^l
S_K^1
S_K^L
s^*
P_1
P_m
P_M
p
$\sum_i b_{i,1}^1 \cdot w(i,t_1,t_2)$
$\sum_i b_{i,k}^l \cdot w(i,t_1,t_2)$
$\sum_i b_{i,k}^L \cdot w(i,t_1,t_2)$
$S_{1,1}^1 \cdot (t_2-t_1)$
$S_{m,k}^l \cdot (t_2-t_1)$
$S_{M,K}^L \cdot (t_2-t_1)$
$A(P_1,t_1,t_2)$
$A(P_m,t_1,t_2)$
$A(P_M,t_1,t_2)$

Fig. 5. Solving the assignment problem for energetic lower bound

Property 2. If there is at least a time-interval $[t_1, t_2]$ where the maximum flow in $G(t_1, t_2, D)$ is strictly less than the sum of the mandatory parts times the number of persons needed $(\sum_{i=0}^{n} \sum_{k=0}^{K} \sum_{l=0}^{L} (w(i, t_1, t_2) \cdot b_{i,k}^l))$, then D+1 is a valid lower bound.

Proof. The proof is obvious due to the graph construction.

4 Experimental Results

4.1 Instances Generation

Since there are no standard benchmark instances for the problem this paper deals with, we have generated some instances to test our methods. In fact, the problem is really close to other project management problems like the MM-RCPSP and multi-resource shop, but to validate our lower bounds we have to take instances where the number of "modes" or possible subsets of persons are much more numerous. Thus, we decided to keep the precedence graph from the PSPlib instances [19], [20], and build the data sets on those graphs. We have taken 180 instances from the single mode RCPSP, with 30, 60 and 90 activities, and a network complexity equal to 1.5, 1.8 or 2.1. Then for each instance, we have randomly generated between three and six different skills with three levels of ability. Then we have generated a number of persons between five and 30, and defined their skills in order to have at least one feasible solution. The instances we have generated in this way represent a wide range of instances, according to the *disjunctive ratio on precedence* and *disjunctive ratio on resource* defined in [3]. Some of them are strongly constrained so there are very few blocks of activities that can be scheduled in parallel, and other have many activities that can be in progress at the same time.

Table 4. Deviation between the two lower bounds and the upper bound given by the tabu search

No. of	No. of	LB with energetic reasoning (%)		LB with graph of compatibility (%)		Best of the two LB (%)	
activities	instances	Av.	Max.	Av.	Max.	Av.	Max.
30	60	3.06	16.66	2.95	13.63	2.53	13.636
60	60	8.03	20.58	11.34	28.2	7.35	18.6
90	60	11.47	26.47	22.34	47.17	11.17	25
Total	180	7.52	26.47	12.21	47.17	7.02	25

Table 5. Computational time of the lower bounds

No. of activities	No. of instances	LB with energetic reasoning (s)	LB with graph of compatibility (s)
30	60	0.938	0.012
60	60	3.013	0.04
90	60	5.99	0.15
Total	180	3.31	0.07

4.2 Results

Using 180 generated data sets, we have obtained the results presented in Table 4. In this table, the lower bounds are compared to the best known upper bound for each instance. These upper bounds have been obtained by a tabu search we have applied to the specific problem with hierarchical levels of skills, inspired by [6]. This tabu search has been adapted from the one for RCPSP [17]. Thus, in the first step, activities are sorted in a list, according to a classical priority rule (Minimum Slack Time, Latest Starting Time,...), and we apply a dispatching rule adapted from the Serial Schedule Scheme to define a solution. Then a neighbour is defined by swapping two activities in the priority list, respecting precedence constraints. Moves that do not modify the current solution in terms of starting times of activities are not allowed. All generated solutions are stored using a hashtable, with the index of the iteration in which the solution has been explored. These solutions are considered to be tabu during a given number of iterations, in order to avoid cycling.

The deviation presented in Tables 4 and 5 is equal to: $\frac{upper\ bound - lower\ bound}{upper\ bound}$. In these tables we do not show the results given by the lower bound based on the graph of compatibility when the stable set is computed by the greedy algorithm because this method is not efficient: it is clearly dominated by the two other lower bounds. Notice that the minimum deviation between LB and UB is not reported because it is always equal to zero.

These results clearly show that the lower bound given by energetic reasoning is better than the lower bound given by the graph of compatibility for every instance. However, Table 5 shows that this lower bound is much more time consuming. Although the average deviation is more important for the method based on the graph of compatibility, there exist some instances where this lower bound is more efficient than the other one. This is why the third column has been included in Table 4 in order to take the best one from the two lower bounds. It appears that the deviation we get considering the best from the two lower bounds is better than the one obtained by the energetic reasoning-based lower bound. So, the two lower bounds appear to be complementary. This is due to the fact that they do not consider the same aspect of the problem: the graph of compatibility first considers the activities two by two, and checks if there is a resource conflict between them in the whole time horizon, whereas the energetic lower bound checks a set of activities but only for restricted time intervals.

Notice that even if the lower bounds seem very time consuming, they will be used in this way, i.e. included into binary search on D, only at the root node of the search tree. In every other node of a branch-and-bound method, it will not be useful to search for lower bound by binary search, it is only necessary to check if the current value of the best upper bound minus one is feasible or not. If this value cannot be respected, that means that at least one of the two lower bounds detects infeasibility, so the node is pruned. Thus, the time needed at each node will be drastically reduced.

In order to find out if the lower bound limitations are particularly linked to one feature (or more) of the instances, as it appears for the RCPSP [3], we have tried to classify our instances and find a link between the average deviation and the *ratio of resource disjunction* between activities, the *ratio of precedence disjunction* between activities or the average number of couple of activities that have no contradiction to be assigned in parallel. No conclusion can be drawn because none of these features appears to be directly influential.

Finally, the average deviation increases dramatically when the size of the instances grows, but this gap may be due to the quality of the upper bound used. Actually, the tabu search we used is not guaranteed to be really efficient for the kind of instances with hierarchical levels of skill we consider, and the gap with the optimum is unknown. On some instances we have generated for the MSPSP without hierarchical levels of skills, i.e. certainly less difficult, the average deviation between the best lower bounds and the upper bound is around 4%.

5 Conclusion

This paper deals with the MSPSP, which is an extension of the RCPSP. In this particular scheduling problem, the resources are staff members with specific skills and levels of skills that allow them to be assigned to different kind of activities of the project. Each activity has a specific need for each level of each skill.

We have proposed a model for this problem, and introduced two lower bounds, adapted from lower bounds known for the RCPSP. The results show that the lower bounds are complementary and efficient.

The research direction we focus on now is to design an exact method in order to compute optimal solutions for small and medium size instances. This method will be used to determine whether the gap between lower bounds and upper bound is due to the upper bound or not. Our two lower bounds will be used in the branch-and-bound method. This exact method is already in progress [5].

References

1. Alvarez-Valdes, R., Martin, G., Tamarit, J. M.: Constructing Good Solutions for a School Timetabling Problem. J. Oper. Res. Soc. **47** (1996) 1203–1215
2. Bailey, J.: Optimization and Heuristic Models to Integrate Project Task and Manpower scheduling. Comput. Indust. Eng. **29** (1995) 473–476
3. Baptiste, P., Le Pape, C., Nuijten, W.: Satisfiability Tests and Time Bound Adjustments for Cumulative Scheduling Problems. Ann. Oper. Res. **92** (1999) 305–333
4. Baptiste, P., Demassey, S.: Tight LP Bounds for Resource Constrained Project Scheduling. OR Spectrum **26** (2004) 251–262
5. Bellenguez, O., Néron, E.: An Exact Method for Solving the Multi-skill Project Scheduling Problem. Operations Research Conference, Tilburg (2004) 97–98
6. Bellenguez, O., Néron, E.: Methods for Solving the Multi-skill Project Scheduling Problem. 9th Int. Workshop on Project Management and Scheduling, Nancy (2004) 66–69
7. Brucker, P., Knust, S.: Lower Bounds for Resource-Constrained Project Scheduling Problems. Eur. J. Oper. Res. **149** (2003) 302–313
8. Burke, E. K., De Causmaecker, P., Vanden Berghe, G., Van Landeghem, H.: The State of the Art of Nurse Rostering. J. Scheduling **7** (2004) 441–499
9. Dauzère-Pérès, S., Roux, J., Lasserre, J. B.: Multi-resource Shop Scheduling with Resource Flexibility. Eur. J. Oper. Res. **107** (1998) 289–305
10. Demassey, S., Artigues, C., Michelon, P.: Constraint-Propagation-Based Cutting Planes: An Application to the Resource-Constrained Project-Scheduling Problem. INFORMS J. Comput. (2003) In press
11. Demeulemeester, E., Herroelen, W.: New Benchmark Results for the Resource-Constrained Project Scheduling Problem. Manage. Sci. **43** (1997) 1485–1492
12. De Reyck, B., Herroelen, W.: The Multi-mode Resource-Constrained Project Scheduling Problem with Generalized Precedence Relations. Eur. J. Oper. Res. **119** (1999) 538–556
13. Hartmann, S., Drexl, A.: Project Scheduling with Multiple Modes: A Comparison of Exact Algorithms. Networks **32** (1998) 283–297
14. Haase, K., Latteier, J., Schirmer, A.: The Course Scheduling Problem at the Lufthansa Technical Training. Eur. J. Oper. Res. **110** (1998) 441–456
15. Hanne, T., Nickel, S.: A Multiobjective Evolutionary Algorithm for Scheduling and Inspection Planning in Software Development Projects. Eur. J. Oper. Res. **167** (2005) 663–678
16. Josefowska, J., Mika, M., Rozycki, R., Waligora, G., Weglarz, J.: Simulated Annealing for Multi-mode Resource-Constrained Project Scheduling Problem. Ann. Oper. Res. **102** (2001) 137–155

17. Klein, R.: Project Scheduling with Time-Varying Resource Constraints. Int. J. Prod. Res. **38** (2000) 3937–3952
18. Kolisch, R., Drexl, A.: Local Search for Non-preemptive Multi-mode Resource-Constrained Project Scheduling Problem. IIE Trans. **29** (1997) 987–999
19. Kolisch, R., Sprecher, A.: PSPLIB—A Project Scheduling Problem Library. Eur. J. Oper. Res. 96 (1997) 205–216
20. Kolisch, R., Sprecher, A.: PSPLIB—A Project Scheduling Problem Library. ftp://ftp.bwl.uni-kiel.de/pub/operations-research/psplib/html/indes.html (2000)
21. Lopez, P., Erschler, J., Esquirol, P.: Ordonnancement de Tâches sous Contraintes: Une Approche Énergétique. RAIRO-APII **26** (1992) 453–481
22. Mingozzi, A., Maniezzo, V., Riciardelli, S., Bianco, L.: An Exact Algorithm for the Resource-Constrained Project Scheduling Problem Based on a New Mathematical Formulation. Manage. Sci. **44** (1998) 714–729
23. Özdamar L.: A Genetic Algorithm Approach to a General Category Project Scheduling Problem. IEEE Trans. on Systems, Man, and Cybernetics, forthcoming
24. Sprecher, A., Drexl, A.: Multi-mode Resource Constrained Project Scheduling Problem by a Simple, General and Powerful Sequencing Algorithm. Eur. J. Oper. Res. **107** (1998) 431–450
25. Toroslu, I.: Personnel Assignment Problem with Hierarchical Ordering Constraints. Comput. Indust. Eng. (2003) 493–510