Les Cahiers du GERAD ISSN: 0711–2440

An iterative algorithm based on evolutive cuts for project scheduling with material storage constraints

A. Piveteau, M. Gamache,
R. Pellerin

G–2015–106

October 2015

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

An iterative algorithm based on evolutive cuts for project scheduling with material storage constraints

Alban Piveteau [a]
Michel Gamache [b]
Robert Pellerin [a]

[a] *Department of Mathematics and Industrial Engineering, Jarislowsky/SNC-Lavalin Research Chair, Polytechnique Montréal, Montréal (Québec) Canada, H3C 3A7*

[b] *GERAD & Department of Mathematics and Industrial Engineering, Jarislowsky/SNC-Lavalin Research Chair, Polytechnique Montréal, Montréal (Québec) Canada, H3C 3A7*

alban.piveteau@polymtl.ca
michel.gamache@polymtl.ca
robert.pellerin@polymtl.ca

October 2015

Les Cahiers du GERAD
G–2015–106

Abstract: This paper deals with the resource constrained project scheduling problem, which consists of scheduling a set of activities with minimal duration that are subject to precedence constraints and the limited availability of resources. In addition to these constraints, we also take into account work space limitations and storage capacity, which are major challenges in construction projects. To solve this NP-Hard problem, we proposed an algorithm based on an evolutive lower bound which permits the branching procedure to be driven with more efficiency whilst minimizing the problem size by updating the time horizon all along iterations. Tests on instances based on project examples generated from the *PSPLib* library are presented.

Key Words: Project scheduling, logistics constraints, RCPSP, branch & bound, optimization, integer programming, lower bound.

Résumé : Cet article traite du problème de gestion de projet avec contraintes de ressources qui consiste à ordonnancer des activités de façon à minimiser le temps de complétion tout en respectant des contraintes de précédence et de ressources. En plus de ces contraintes, on doit tenir compte d'un espace de stockage limité, ce qui est une contrainte fréquemment rencontrée dans les projets de construction. Pour résoudre ce problème NP-difficle, nous proposons un algorithme basé sur la mise à jour fréquente d'une borne inférieure, ce qui rend plus efficace la procédure de branchement en réduisant l'horizon de temps, ce qui réduit également la taille du problème au fur et à mesure que les itérations progressent. Des tests sur des instances générées à partir de la librairie PSPLib sont présentés.

1 Introduction

The aim of the Resource Constrained Project Scheduling Problem (RCPSP) is to schedule a set of activities with minimal duration subject to precedence constraints and the limited availability of resources. This problem is NP-hard [2] because of the cumulative nature of resource consumption, which allows for the execution of parallel activities.

The RCPSP has been widely studied in the literature. The proposed approaches differ mainly in their objective function (minimize makespan, minimize costs, maximize solution robustness), and their resolution strategy [3]. Much attention has been put forth to resolve resource conflicts occurring when the total amount of required employees by activities that can be run in parallel exceeds the number of available employees. However, the realization of projects can face other restrictions that may affect the project schedule. For example, work space limitations and storage capacity are major challenges in construction projects. Conflicts of space often occur when the necessary space for an activity interferes with that of other activities or with previously-accomplished works. For example, Riley and Sanvido [14] studied a building site during a period of two months and recorded 71 cases of space conflicts for only four labor types. Several studies also confirmed that space limitation is one of the main reasons of productiveness loss in construction [13, 10, 9, 5]. The difficulty of managing work space is mainly a result of the dynamic character of the space necessary to carry out a task, since this space changes as plans advance.

In a similar manner, equipment and material storage capacity has a major impact on project scheduling and procurement activities. The management of building materials is a crucial activity as the costs of the equipment and material represent a major portion of total project costs [15]. According to several studies, building materials generally account for 40 to 60 % of the complete project budget [1, 12, 16]. Furthermore, bad planning of storage space utilization can draw away a lack of necessary equipment to carry out tasks and or postpone construction works, or conversely, needlessly augment the inventory costs by accepting materials at the building site too far in advance. Furthermore, the acquisition of material before their current needs can cause serious problems such as congestion and deterioration of the quality of some materials.

Recognizing the difficulty of managing projects with material space constraints, this research explores the use of an iterative algorithm based on evolutive cuts to determine optimal project plan under resource and storage space constraints. The reminder of the paper is organized as follows. First, in Section 2 we introduce the integer linear programming model that includes new logistics constraints for classic scheduling problems. Then Section 3 demonstrates performances of this model on instances from *PSPLib* on which we generate new attributes. The aim of those tests is to identify integer program (*IP*) limits (size, resolution time). Finally, we present a heuristic approach based on the resolution of the *IP* at which we introduce a new contraint acting as a lower bound for the Z value in Section 4. The lower bound value will evolve gradually at each iteration until a stopping criterion detection.

2 The mathematical model

We introduce the integer linear programming model that includes new logistics constraints for classic scheduling problems. Among those additional constraints, we consider a procurement time delay which requires the reception of materials before the execution of the associated task. This procurement policy involves warehousing constraints where materials have to be stored before and during the task process. Moreover, warehousing constraints must respect space availability in stock areas or warehouses. We assume that warehousing space occupied by materials run until the end of associated tasks. Consequently, warehousing space occupied by materials required for a specific task stays constant over a time corresponding to the sum of task duration and the warehousing duration required before the starting date of the task (see Figure 1).

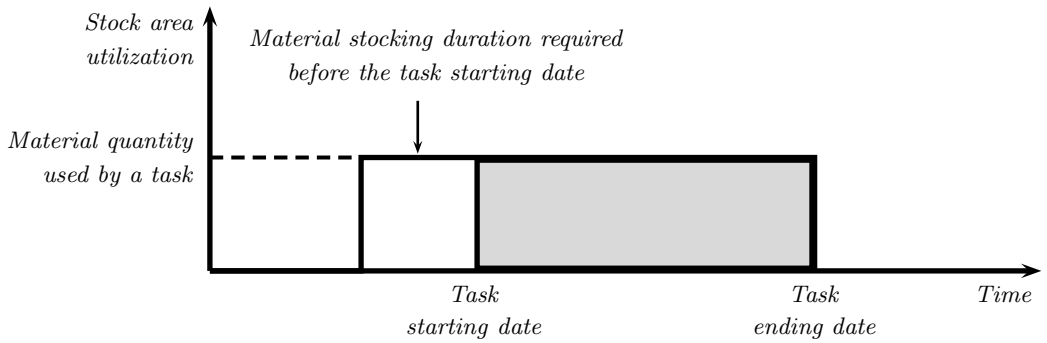

Figure 1: Stockage area utilization for a task

We will introduce notations used in our model described in the next section.

i	an index representing task number such as $i \in \mathcal{I} = \{0, 1, 2, \ldots, n+1\}$ where activities 0 et $n+1$ represent the project start and end, respectively
Γ_i	the set of immediate successors of the task i
p_i	the processing time of task i where $p_0 = p_{n+1} = 0$
d_i	the material stocking duration required before task i starting date where $d_0 = d_{n+1} = 0$
t	an index representing the treatment period such as $t \in \mathcal{T} = \{0, \ldots, T\}$ where T represents project maximal duration estimated by $T = \sum_{i \in \mathcal{I}} p_i$
ES_i	the earliest starting date of task i (found with preprocessing by resolving the IP considering precedence constraints only)
LS_i	the latest starting date of task i ((found with preprocessing by resolving the IP considering precedence constraints and the horizon only)
h	an index representing a kind human resource such as $h \in \mathcal{H}$
R_h	the number of human resources h available per period
$r_{i,h}$	the number of human resources h required for task i per period where $r_{0,h} = r_{n+1,h} = 0 \quad \forall h \in \mathcal{H}$
m	the index representing a kind material resource such as $m \in \mathcal{M}$
a	an index representing a stockage area such as $a \in \mathcal{A}$
S_a	the number of materials able to be stored on stocking area a
$s_{i,m}$	the number of material m required for task i per period where $s_{0,m} = s_{n+1,m} = 0 \quad \forall m \in \mathcal{M}$
$\gamma_{m,a}$	an integer parameter equals to 1 if material m can be stored on stocking area a and 0 otherwise.

In the model, to determine if a task i is active in period t, we must check if it began during the period $[t - p_i + 1, t]$. In order to avoid periods verification where $t - p_i + 1 < 0$, we use $\omega_i = \max\{0, t - p_i + 1\}$. By analogy, we introduce $\delta_i = \min\{t + d_i, T\}$ in order to avoid periods verification where $t + d_i > T$.

The decision variables for this model are the following:

$$x_{i,t} = \begin{cases} 1, & \text{if task } i \text{ starts on period } t \\ 0, & \text{otherwise} \end{cases}$$

The linear program is defined by:

$$\min \; Z = \sum_{t=ES_{n+1}}^{LS_{n+1}} t x_{n+1,t} \tag{1}$$

sous les contraintes :

$$\sum_{\tau=ES_j}^{LS_j} tx_{j,\tau} - \sum_{\tau=ES_i}^{LS_i} tx_{i,\tau} \geq p_i \qquad \forall i \in \mathcal{I},\ \forall j \in \Gamma_i \tag{2}$$

$$\sum_{i\in\mathcal{I}} \sum_{\tau=\omega_i}^{t} r_{i,h} x_{i,\tau} \leq R_h \qquad \forall h \in \mathcal{H},\ \forall t \in \mathcal{T} \tag{3}$$

$$\sum_{i\in\mathcal{I}} \sum_{m\in\mathcal{M}} \sum_{\tau=\omega_i}^{\delta_i} s_{i,m}\gamma_{m,a} x_{i,\tau} \leq S_a \qquad \forall a \in \mathcal{A},\ \forall t \in \mathcal{T} \tag{4}$$

$$\sum_{\tau=0}^{ES_i-1} x_{i,\tau} = 0 \qquad \forall i \in \mathcal{I} \tag{5}$$

$$\sum_{\tau=LS_i+1}^{T} x_{i,\tau} = 0 \qquad \forall i \in \mathcal{I} \tag{6}$$

$$\sum_{t\in\mathcal{T}} x_{i,t} = 1 \qquad \forall i \in \mathcal{I} \tag{7}$$

$$x_{i,t} \in \{0;1\} \qquad \forall i \in \mathcal{I},\ \forall t \in \mathcal{T} \tag{8}$$

The objective-function (1) minimizes the starting date of the last task of the project ((i.e. $i = n + 1$). The set of constraints (2) forces the start date of all immediate successors of task i (i.e. tasks j) after the end date of task i. Human resource constraints are defined by the set of constraints (3). On each period t and for each resource h, the human resources needed for all activities i in treatment have to be inferior to the human resources availability R_h. The new logistics constraints are defined by the set of constraints (4). In each period t and for each storage area a, the space needs for all activities i in treatment and the ones for which materials are already delivered because of the procurement policy must be inferior to the stockage area space availability S_a. To ease the resolution process, the set of constraints (5) and (6) set to zero the value of variables $x_{i,t}$ when the period t is less than the earliest starting date of task i and when period t is greater than the latest starting date of task i. The set of constraints (7) imposes that each activity occurs only once. Finally, the set (8) declares all variables as binary.

3 Tests and results

In this section, we first describe the instances that were generated in order to test our model. Results of these tests are then presented and analyzed.

3.1 Description of instances

Our instances are based on project examples generated from the *PSPLib* library. A tool has been developed to add materials needs, restrictions and space availability on each of the storage areas. It permits to simulate the material management context in our examples. The model has been tested on four different project sizes: 30, 60, 90 and 120 tasks.

For each task, the duration, the immediate successors as well as the human resource requirements are derived from *PSPLib* and correspond to the following examples:

Number of tasks	Project
30	*J*3022_5
60	*J*6022_4
90	*J*9022_9
120	*J*12030_3

For each project, five difficulty levels, with respect to material constraints (4), were created (the first level being the most difficult). In all, we tested the model on twenty instances.

The three parameters qualifying projects in *PSPLib* (i.e., *network complexity nc*, *resource factor RF* and *resource strength RS*) are fixed to their medium value: $nc = 1.8$, $RF = RS = 0.5$. The storing constraints are defined by two parameters: RFN (*material force*) and RSN (*material strength*). The RFN parameter represents the average number of material types needed for a task. This parameter is standardized between 0 and 1. For instance, if $RFN = 0.5$ and if our tool generated 4 types of material, then the medium number of material type required by task will be equal to 2. The RSN parameter is used to generate the maximum storing area space. The parameter is also standardized between 0 and 1 (the larger this value is, the less warehousing space restrictions there are).

Considering RSN as the most influent parameter (cf, similarity between *material force/strength* and *resource force/strength* and Kolisch [6] works), RFN is fixed to the medium value ($RFN = 0.5$). Thereby, five instances are generated for each project (30, 60, 90 and 120 tasks) by varying the RSN parameter from 0.1 to 0.9 with an increment of 0.2.

Mathematical model scripts were elaborated with *AMPL* and solved using *CPLEX solver* (12.2.0.0 version). In our tests, we limited the solution time to 5000 seconds. All our computations were made using a computer with an AMD Athlon 64 X2 Dual Core Processor 4200+, 2.22 GHz and 3 Go RAM.

3.2 Results

Tables 1 and 2 show the results obtained for the 20 instances. Times are indicated in seconds. An X indicates that no integer solution was found, Z_{LP} is the value of the optimal solution of the linear program relaxation whereas Z_{ILP} is the value of best integer solution found within the time delay. Finally, a star indicates that the optimal solution was found.

Table 1: Results for projects with 30 and 60 tasks

Level	30 tasks				60 tasks			
	LP		ILP		LP		ILP	
	Z_{PL}	Time	Z_{ILP}	Time	Z_{LP}	Time	Z_{ILP}	Time
1	52.14	2.03	76	5000	92.09	19.12	X	5000
2	47.23	0.98	56*	69.81	19.09	254	5000	
3	58.53	1.91	87	5000	65.18	9.09	111	5000
4	46.65	0.74	52*	67	65.18	5.48	359	ROM
5	46.65	0.76	52*	16	65.18	7.44	83	5000

ROM: Run Out of Memory

Table 2: Results for projects having 90 and 120 tasks

Level	90 tasks				120 tasks			
	LP		ILP		LP		ILP	
	Z_{PL}	Time	Z_{ILP}	Time	Z_{LP}	Time	Z_{ILP}	Time
1	136,60	29.81	X	5000	141.03	100.89	X	5000
2	99,03	43,92	X	5000	108	51.56	X	5000
3	97	25,34	X	5000	108	10.08	659	ROM
4	97	16,66	X	5000	108	10.19	X	5000
5	97	7.80	234	5000	108	10.70	659	ROM

ROM: Run Out of Memory

Tables 3 and 4 show the details of the solution process; more precisely, they indicate for each instance the number of nodes visited in the branching tree and the number of simplex iterations required to find the LP solution.

Table 3: Number of branching nodes and simplex iterations

	30 tasks		60 tasks	
Level	Nodes	Simplex iterations	Nodes	Simplex iterations
1	243538	19276212	5049	1871946
2	241705	17768261	4486	2722956
3	125347	18962814	36181	9899318
4	10469	77793	ROM	ROM
5	530	4503	212815	23831283

ROM: Run Out of Memory

Table 4: Number of branching nodes and simplex iterations

	90 tasks		120 tasks	
Level	Nodes	Simplex iterations	Nodes	Simplex iterations
1	1	516045	63	399034
2	917	771265	134	452039
3	5704	756471	ROM	ROM
4	8107	794698	240	803888
5	4471	1390234	ROM	ROM

ROM: Run Out of Memory

The results illustrated in these four tables demonstrate that the current approach cannot quickly converge towards optimal solutions, especially for large scale projects. A detailed analysis of solutions at each node of the branching tree shows the presence of a very large integrality gap (more than 15%). Moreover, the value of the LP solution at each node remains very close to the one obtained at the root node. Thereby, a huge amount of nodes are generated. When a feasible integer solution is found, the value of this solution doesn't give an efficient upper bound and thus, the number of nodes in the branching tree remains quite large. Based on these observations, we propose a new approach that will alleviate these difficulties. This new approach is described in the following section.

4 A new solution strategy based on an evolutive cut

Considering the large integrality gap observed, we propose a two-phase approach. The first one consists of finding a feasible solution for the IP, whereas the second phase attempts to improve this solution to converge towards the optimal solution. This iterative approach involves the addition of a new constraint (denoted the evolutive cut) which forces the solution to be higher or equal to a defined value called the temporary lower bound, denoted by LB^k_{temp}, where k represents the iteration number. The LB^k_{temp} value is modified at each iteration: $LB^{k+1}_{temp} = \alpha_1 * LB^k_{temp}$, where $0 < \alpha_1 < 1$ is a coefficient used to reduce the temporary lower bound value. Thereby, at the beginning, the lower bound on the value of Z_{LP} is very high, which eases the search for an feasible integer solution, even though the gap between the value of this solution and the one of the optimal solution is large.

4.1 The new constraint

Let $\mathcal{P}$ be the IP described by the objective-function (1) and constraints (2) to (8). As explained previously, a cut is introduced into problem $\mathcal{P}$ in order to force the value of the objective function at iteration k to be greater or equal to LB^k_{temp}. The new constraint at iteration k can be written as follows:

$$\sum_{t \in \mathcal{T}} t x_{n+1,t} \geq LB^k_{temp} \tag{9}$$

There is another way to implement this constraint:

$$\sum_{t=1}^{\rho} x_{n+1,t} = 0 \tag{10}$$

where $\rho = LB^k_{temp} - 1$.

This last constraint is more restrictive than the previous one during the solution of the *LP*. Moreover, this constraint is equivalent to set $ES_{n+1} = Z^k_{PLNE}$. This latter formulation is of greater interest given it simply requires modifying constraint (5). In fact, giving the objective-function (1), changing the upper limit of the summation in the constraint (5) has the same effect as adding the lower bound on the value of the solution. This formulation is the one that has been implemented.

4.2 Two theorems

Before giving details on the solution approach, we introduce two theorems that enable us to prove that a solution at iteration k is optimal under certain conditions.

Theorem 1 *Let $\mathcal{P}$ be the problem described in the previous subsection. If a feasible solution, called $\mathcal{S}_1$, exists for $\mathcal{P}$ such as $x^1_{n+1,t_1} = 1$, then another feasible solution, called $\mathcal{S}_2$, also exists for $\mathcal{P}$ such as $x^2_{n+1,t_2} = 1 \quad \forall t_2 > t_1$.*

Proof. Let $\mathcal{S}_1$ be a feasible solution for problem $\mathcal{P}$ and such as $x^1_{n+1,t_1} = 1$. Let $\mathcal{S}_2$ be a solution such as:

$$\begin{aligned} x^1_{i,t} &= x^2_{i,t} \quad \forall i \neq n+1;\ \forall t \\ x^1_{n+1,t_1} &= 1 \\ \text{and} \quad x^2_{n+1,t_2} &= 1 \qquad \text{where} \quad t_2 > t_1 \end{aligned}$$

Since $\mathcal{S}_1$ is feasible, $t_2 > t_1$, and task $n+1$ does not consume any resources, does not have successor and its processing time is null, then all constraints of $\mathcal{P}$ will remain satisfied with solution $\mathcal{S}_2$, which proves that $\mathcal{S}_2$ is also a feasible solution. □

This theorem indicates that it is always possible to find a feasible solution from another one simply by delaying the starting date of task $n+1$.

Theorem 2 *If $\mathcal{S}_2$ is an optimal solution of problem $\mathcal{P}^k$ such as $x^2_{n+1,t_2} = 1$ and $t_2 > LB^k$, then $\mathcal{S}_2$ is the optimal solution of problem $\mathcal{P}$.*

Proof. Let $\mathcal{S}_1$ be an optimal solution of problem $\mathcal{P}$ such as $x^1_{n+1,t_1} = 1$ for $t_1 < LB^k$. Let $\mathcal{S}_2$ be an optimal solution of problem $\mathcal{P}^k$ such as $x^2_{n+1,t_2} = 1$ where $t_2 > LB^k$. From theorem 1, we know that it is possible to create a solution $\mathcal{S}_3$ from $\mathcal{S}_1$ such as:

$$\begin{aligned} x^1_{i,t} = x^3_{i,t} \qquad & \forall i \neq n+1;\ \forall t \\ \text{and} \quad x^3_{n+1,t_3} = 1 \qquad & \text{where} \quad t_3 = LB^k \end{aligned}$$

Since $\mathcal{S}_3$ is a feasible solution and $\mathcal{Z}_3 < \mathcal{Z}_2$ then $\mathcal{S}_3$ is an optimal solution for problem $\mathcal{P}^k$, which contradicts the first assertion. □

4.3 An iterative solution approach using a decreasing lower bound

To understand the notation used in the proposed algorithm, we introduce the following definitions.

LB^k_{temp}	A lower bound value fixed at iteration k.
UB^k	An upper bound value equals to $\mathcal{Z}^k$, the current best admissible solution (optimal or not) for problem $\mathcal{P}^k$.
β	The global stopping criteria that limits the total solution time (in seconds).
β_ℓ	A stopping criteria that limits the solution time (in seconds) required for finding a feasible solution during operation ℓ (where $\ell = 0$ or 1) of the algorithm. When this period is exceeded, then it is better to tighten the bounds by increasing the value of the lower bound than to continue the resolution. Those delays are included within β but are not additive to it.
α_0 and α_1	Two parameters used to reduce the value of the lower bound during the first (α_0) phase and the second phase (α_1) of the algorithm.
$\mathcal{Z}^k$	The best integer solution found at iteration k.

Other parameters used in the algorithm were defined in previous sections. Figure 2 presents a pseudo-code for the algorithm.

1 $k := 0$
2 $UB^0 := T$
3 $LB^0_{temp} := \lfloor \alpha_0 T \rfloor$
4 Solve $\mathcal{P}^0$ for a time limit β_0
5 If no admissible solution for problem $\mathcal{P}^0$ is found during the time period β_0 then
6 $\quad LB^0_{temp} := \lfloor 0.5(LB^0_{temp} + T) \rfloor + 1$ and go back to step 4
7 Else (*an admissible solution is found*)
8 $\quad k := k + 1$
9 $\quad UB^k := \mathcal{Z}^{k-1}$
10 $\quad LB^k_{temp} := \lfloor \alpha_1 UB^k \rfloor$
11 $\quad$ If $UB^k - LB^k_{temp} = 1$
12 $\quad\quad$ Solve $\mathcal{P}^k$ with a limited resolution time $\beta_1 = \beta_{1.1}$
13 $\quad$ Else
14 $\quad\quad$ Solve $\mathcal{P}^k$ with a limited time $\beta_1 = \beta_{1.2}$
15 $\quad$ If $\mathcal{Z}^k = LB^k_{temp}$
16 $\quad\quad$ Go back to step 8 (*we have to decrease* LB^k_{temp})
17 $\quad$ Else (*then we have* $\mathcal{Z}^k > LB^k_{temp}$)
18 $\quad\quad$ If resolution time is inferior to β_1
19 $\quad\quad\quad$ Stop: the optimal solution is found
20 $\quad\quad$ Else (*limited resolution time is reached*)
21 $\quad\quad\quad LB^k_{temp} := \lfloor 0.5(LB^k_{temp} + UB^k) \rfloor$ and go back to step 8

Figure 2: Research algorithm with a decreasing lower bound

The algorithm is processed in two phases. Phase 1, described by steps 1 to 6, consists of finding a feasible solution to problem $\mathcal{P}$. A maximal delay of β_0 seconds is devoted to the resolution of this problem. If delay β_0 is reached and no integer solution is found, then it is better to increase the temporary lower bound by choosing the midpoint between LB^k_{temp} and T in order to reduce the research domain rather than pursue the resolution (steps 5 and 6). The value of β_0 is also accounted for in the global resolution time which is compared to β, i.e the stopping criteria when no feasible or optimal solution is found for the entire problem. Choosing the values of parameters α_0 and β_0 is not trivial and it depends on the problem difficulty. For an easy problem to solve, it is adequate to fix α_0 at a lower level in order to converge faster towards the optimal solution; while a more difficult problem requires an higher initial lower bound. For our tests, we fixed $\alpha_0 = 0,8$ and $\beta_0 = 600$ seconds.

When a feasible integer solution is found, we proceed to phase 2, described in steps 7 to 21. This phase consists of finding the optimal solution or the best integer solution before exceeding overall time limit β

(set a priori) to which we subtract the time used in Phase 1. At each iteration of this phase, we start with an adjustment of the research domain by updating the two bounds LB^k_{temp} and UB^k (steps 9 and 10). We set $BS^k = \mathcal{Z}^{k-1}$ and we decrease the lower bound value by a factor α_1 which acts similar to α_0. A preprocessing phase is undertaken to set the value of some variables according to the new bounds. First, we set $ES_{n+1} = B^k_{temp}$, which involves $x_{n+1,t} = 0 \quad \forall t \leq LB^k_{temp} - 1$. Secondly, knowing that the last planning period, T, can be substituted by UB^k, we can recompute for all tasks i, the LS_i values which will set several decision variables to zero. Finally, we underline the fact that a feasible solution found at iteration $k-1$ can be used as a feasible solution for solving the problem $\mathcal{P}^k$.

Thereafter, at each iteration k, four cases may occur.

Case 1: An optimal solution is found for the problem $\mathcal{P}^k$ within the time limit $\beta_{1.2}$ such as $\mathcal{Z}^k = BI^k_{temp}$. This case tallies with steps 13 to 16. We have to proceed to another iteration and adjust bounds, i.e, set $LB^{k+1}_{temp} = \alpha_1 * LB^k_{temp}$ and update the upper bound to $\mathcal{Z}^k$ (see steps 8 to 10).

Case 2: An optimal solution is found for the problem $\mathcal{P}^k$ within the time limit $\beta_{1.2}$ such as $\mathcal{Z}^k > BI^k_{temp}$. This case tallies with steps 17 to 19. The solution found is also the optimal solution for problem $\mathcal{P}$ (see theorem 2).

Case 3: The time limit is exceeded and we get a feasible solution for problem $\mathcal{P}^k$ such as $\mathcal{Z}^k > BI^{k+1}_{temp}$. This case tallies with steps 13, 14, 17, 20 and 21. Another iteration is needed and the lower bound value BI^{k+1}_{temp} must be increased in order to expedite the resolution. The idea consists of allowing $\beta_{1.2}$ seconds to find the optimal solution for problem $\mathcal{P}^k$ regarding the defined lower bound. If within $\beta_{1.2}$ seconds, it is impossible to confirm that the solution found is optimal, we prefer to increase the lower bound value because we presume that it will be quicker to confirm the optimality in this manner.

Case 4: The limited resolution time $B_{1.2}$ is reached and we get a feasible solution for problem $\mathcal{P}^k$ such as $\mathcal{Z}^k = BI^{k+1}_{temp} + 1$. This case tallies with steps 11 and 12. This case transpires to be similar to case 3. However, here we can't increase the lower bound. We need to allocate more time (here $\beta_1 = \beta_{1.1} \gg \beta_{1.2}$) to the algorithm to confirm whether or not the optimality of the solution can be found.

Finally, if an optimal solution of problem P is not found before exceeding β seconds, we stop the algorithm. The best solution that was found is then printed.

Figure 3 illustrates cases 1 (Figure 3.b), 3 (Figure 3.c), and 4 (Figure 3.d). In this figure, LB is the lower bound of problem $\mathcal{P}$; i.e. the value of the linear relaxation of problem $\mathcal{P}$.

An interesting aspect of this iterative algorithm is that the feasible solution (optimal or not) found at iteration k is a also a feasible solution at iteration $k+1$. So, the time devoted to finding a feasible solution at iteration $k+1$ is reduced to zero.

4.4 Results with the new algorithm

Presented results were acquired with the following parameter values:

$$\begin{aligned} \alpha_0 &= 0,8 \\ \alpha_1 &= 0,9 \\ \beta_0 &= 600 \\ \beta_{1.1} &= 90 \\ \beta_{1.2} &= 3600 \\ \beta &= 5000 \end{aligned}$$

The following tables sum up acquired results using the old and the new algorithms. We first notice that the new algorithm permits a better solution to be obtained, but optimality remains difficult to prove. Even though optimality is not always proven, it is noteworthy to observe the convergency speed growth towards good solutions when using the new algorithm. Indeed, with this method, solutions that would be found in 5000 seconds using the old approach (i.e. without the evolutive lower bound) can be found much quicker.

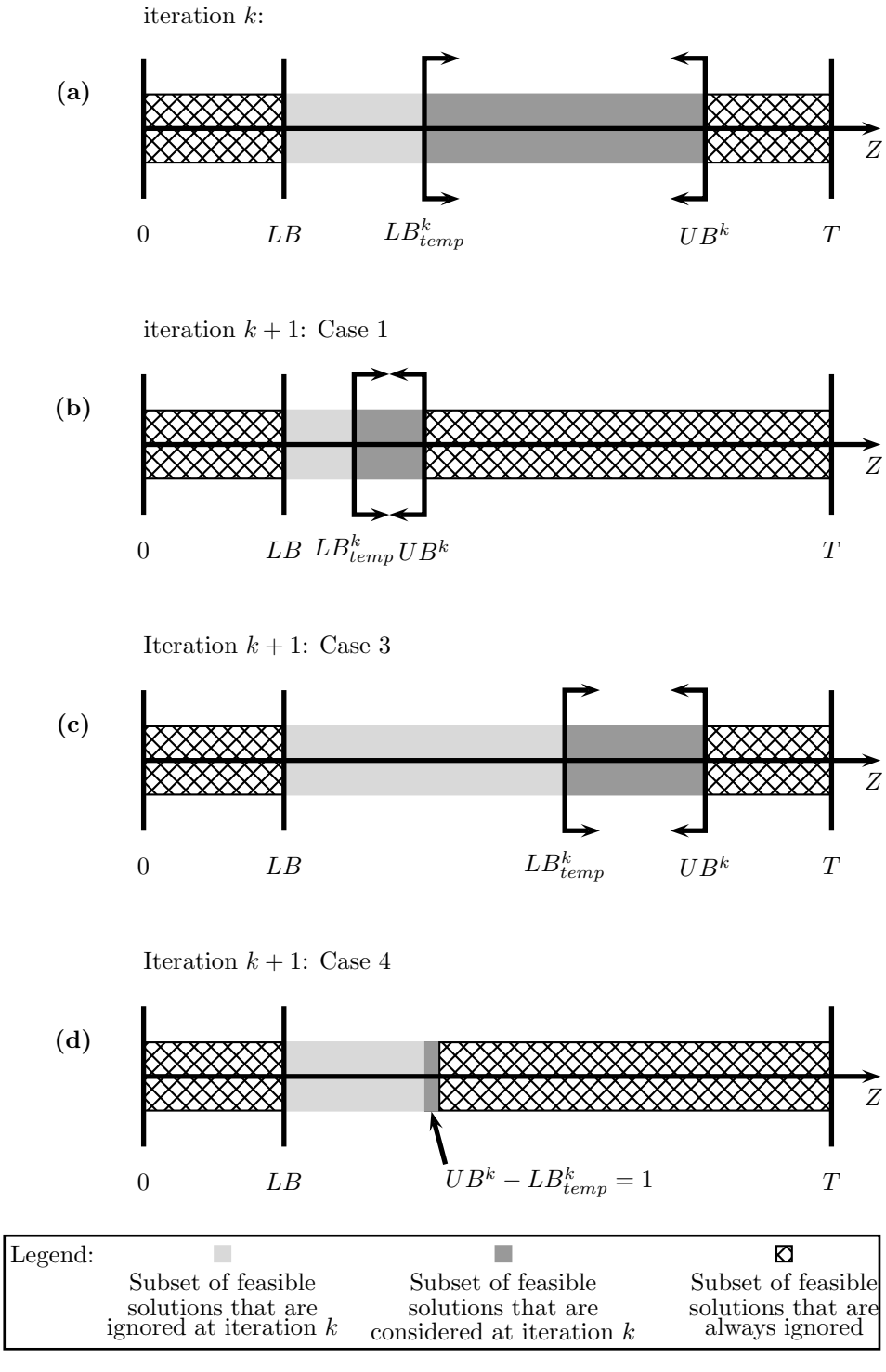

Figure 3: Research areas at iteration k

Table 5: Results for a 30 tasks project

Level	Old algorithm		New algorithm	
	Solution	Time	Solution	Time
1	76	5000	72	5000
2	56*	1453	56*	375
3	87	5000	81	5000
4	52*	67	52*	20
5	52*	16	52*	20

Table 6: Results for a 60 tasks project

Level	Old algorithm		New algorithm	
	Solution	Time	Solution	Time
1	X	5000	142	5000
2	254	5000	87	5000
3	111	5000	74	5000
4	359	ROM	73	5000
5	83	5000	73	5000

Table 7: Results for a 90 tasks project

Level	Old algorithm		New algorithm	
	Solution	Time	Solution	Time
1	*X*	5000	373	5000
2	*X*	5000	198	5000
3	*X*	5000	375	5000
4	*X*	5000	104	5000
5	234	5000	349	5000

Table 8: Results for a 120 tasks project

Level	Old algorithm		New algorithm	
	Solution	Time	Solution	Time
1	*X*	5000	659	5000
2	*X*	5000	381	5000
3	659	ROM	427	5000
4	*X*	5000	108*	2811
5	659	ROM	322	5000

Figure 4 shows advantages of this evolutive cut algorithm for the problem with 30 tasks at level 1 of difficulty (similar results were observed with other examples). We notice on Figure 4 that the new approach using fictive lower bound allows a better solution to be found in 1 minute ($Z = 72$) than a solution found using the traditional approach ($Z = 76$) in 1 hour.

Table 9 presents detailed results for the new approach. Among the other, we indicate the time (in seconds) required to find the best integer solution. For the 9 out of 16 problems in which we did not find optimal solution, we notice that the difference between the upper bound and the fictive lower bound is inferior or equals to 1. Without being able to prove it, we believe that the best solution found is actually the optimal one for those cases. Indeed, the set up we made forces the branch & bound algorithm to search for an integer solution which value is equal to the lower bound. We believe that the domain of feasible solutions respecting integrity constraints is empty while this is not the case for the domain of feasible solutions where

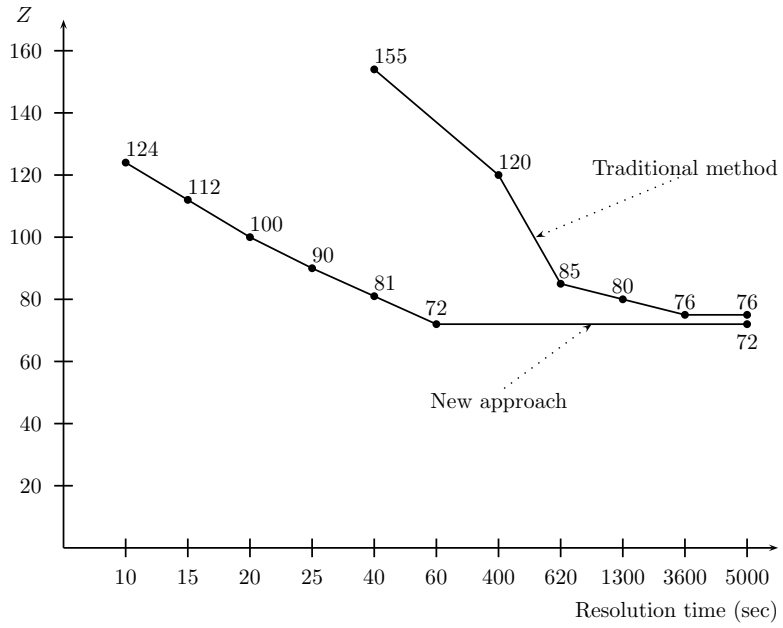

Figure 4: Example of both resolution approaches for the defined problem: 30 tasks, level 1

integrity constraints are relaxed. Therefore, the branch & bound algorithm uses the remaining time trying to prove that no integer solution exists for the lower bound value; it explores a huge branching tree since it has to explore each branch in deep in order to record that branching decision does not lead to a feasible solution. The time limit of 5000 seconds is exceeded before all the leaves in the branch & bound tree have been explored.

Table 9: Detailed results for the new approach

Tasks number	Leve	Z_{LP}	Best integer solution	fictive lower bound	Time (sec)
30	1	53	72	71	63
30	2	48	56*	—	375
30	3	59	81	80	375
30	4	47	52*	—	20
30	5	47	52*	—	20
60	1	93	142	141	2003
60	2	70	87	86	2353
60	3	66	74	43	2830
60	4	66	73	72	2838
60	5	66	73	73	5000
90	1	137	373	354	4873
90	2	100	198	178	4913
90	3	97	375	356	4838
90	4	97	104	103	3861
90	5	97	349	314	4925
120	1	142	659	658	852
120	2	108	381	361	4900
120	3	108	427	427	5000
120	4	108	108*	—	2811
120	5	108	322	319	4600

5 Conclusions and perspectives

Through this work, we have defined a new scheduling problem that allows projects which are subject to material space constraints to be scheduled. To solve these NP-Hard problems, we have proposed an algorithm based on a evolutive lower bound which allows the branching procedure to be driven with more efficiency whilst minimizing the problem size by updating the time horizon and LS_i parameters all along the iterations.

New research perspectives may be considered in future works. First of all, we want to work on the problem modeling by setting up an event-base modeling [7] allowing the number of variables to be minimized. A specific work has also to be done on the branching decisions method. Particularly, Demeulemeester et al. [4] works show a branching method based on priority rules allowing for optimal solutions to be reached quickly on small-size problems. Coupled with our algorithm, those two ideas may be exploited to reach an optimal solution on larger size projects.

Finally, new constraints representing the material management context of modern projects may be studied. For example, delivery activities may be looked at by defining constraints which group deliveries on fixed date. Indeed, it is common to receive and stock material on a fixed date as defined by contractual agreement between suppliers and transporters in order to minimize total procurement and transportation costs.

References

[1] Agapiou, A., Clausen, L.E., Flanagan, R., Norman, G., and Notman, D. The role of logistics in the materials flow control process. Construction Management and Economics, 16, 131–137, 1998.

[2] Blazewicz, J., Lenstra, J.K., and Rinnooy Kan, A.H.G. Scheduling subject to resource constraints: classification and complexity. Discrete Applied Mathematics, 5(1), 11–24, 1983.

[3] Demeulemeester, E.L. and Herroelen, W.S. Project Scheduling: A Research Handbook. (49), International Series in Operations research and Management Science, Kluwer Academic Publishers, 2002.

[4] Demeulemeester, E. and De Reyck, B. The discrete time/resource trade-off problem in project networks: a branch-and-bound approach. Open Access publications from Katholieke Universiteit Leuven, Katholieke Universiteit Leuven, 1997.

[5] Guevremont, M. Ordonnancement de projet sous contraintes de ressources, matériels et d'entreposage. Master report, Polytechnique Montréal, April, 2009.

[6] Kolisch, R., Sprecher, A., and Drexl, A. Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science, 41(10), 1693–1703, October 1995.

[7] Kone, O., Artigues, C., Lopez, P. and Mongeau, M. Event-based MILP models for resource-constrained project scheduling problems. Computers and Operations Research, 38(1), Project Management and Scheduling, January 2011.

[8] Lientz, B.P., and Rea, K.P. International Project Management, Academic Press, San Diego, USA, 2003.

[9] Muehlhausen, F. Construction Site Utilization: Impact of Material Movement and Storage on Productivity and Cost. AACE Transactions, 35, L.2.1–L.2.9, 1991.

[10] Oglesby, C.H., Parker, H.W., and Howell, G.A. Productivity improvement in building. New York: Mcgraw-Hill, 1989.

[11] Pellerin, R., Sadr, J., Guevremont, M. and Rousseau, L.M. Planning of international projects with material constraints. Proceedings of the International Conference on Industrial Engineering and Systems Management, Montréal, Canada, 2009.

[12] Polat, G., and Arditi, D. The JIT materials management system in developing countries. Construction Management and Economics, 23, 697–712, 2005.

[13] Rad, P. Analysis of working space congestion from scheduling dated. American Association of Cost Engineer Transactions, F4.1–F4.5, 1980.

[14] Riley, D.R., and Sanvido, V.E. Space planning method for multistory building construction. Journal of Construction Engineering and Management, ASCE 123(2), 171–180, 1997.

[15] Tserng, H.P., Yin, S.Y., and Li, S. Developing has planning resource supply chain system for building projects. Newspaper of Building Engineering and Management, 132(4), 393–407, 2006.

[16] Wong, E.T.T., and Norman, G. Economic evaluation of materials planning systems for building. Construction Management and Economics, 15, 39–47, 1997.