2nd International Conference on Engineering Optimization
September 6 - 9, 2010, Lisbon, Portugal

ON THE MULTI-MODE, MULTI-SKILL RESOURCE CONSTRAINED PROJECT SCHEDULING PROBLEM (MRCPSP-MS)

Mónica A. Santos[1], Anabela P. Tereso[2]

[1] University of Minho, 4800-058 Guimarães, Portugal, pg13713@alunos.uminho.pt
[2] University of Minho, 4800-058 Guimarães, Portugal, anabelat@dps.uminho.pt

Abstract
In this paper we describe an extension of the Resource-Constrained Project Scheduling Problem (RCPSP). A literature review is presented to place our research in its proper context. The problem presented here belongs to the class of the optimization scheduling problems with multi-level (or multi-mode) activities. This means that the activities can be scheduled at different modes, each mode using a different resource level, implying different costs and durations. Each activity must be allocated exactly one unit of each required resource and the resource unit may be used at any of its specified levels. The processing time of an activity is given by the maximum of the durations that would result from a specific allocation of resources. The objective is to find the optimal solution that minimizes the overall project cost, while respecting a delivery date. A penalty is included for tardiness beyond the specified delivery date. We present a formal description of the problem and a mathematical model for it. We also introduce the implementation algorithm for the problem. The implementation was designed using the JAVA language, and the algorithm proposed is based on a branch and bound procedure, using breadth-first search (BFS) project network traversing, among some heuristic rules to filter large subsets of fruitless candidates relative to resource levels combinations.
Keywords: Project Scheduling Problem, RCPSP, Activity Networks, Multi-level, Multi-mode, Resource Allocation.

1. Introduction

This paper is concerned with an extension of the Resource-Constrained Project Scheduling Problems (RCPSP) which belongs to the NP-hard class of problems. In the several resource constrained scheduling problem models found in the literature, there are two important aspects present in any model: the objective and the constraints. The objective may be based on time, such as minimize the project duration, or on economic aspects, such as minimize the project cost. However, success relative to time does not imply success in economic terms. Often, time-based objectives are in conflict with cost-based objectives. A recurrent situation encountered in practice is the need to complete a project by its due date *and* maximize profit. Ozdmar and Ulusoy [1] reported in their survey of the literature, studies where the NPV is maximized while the due date is a 'hard' constraint (Patterson et al. [2] [3]). There are several other multi-objective studies in the literature where efficient solutions regarding time and cost targets are generated. Guldemond et al. [4] presented a distinctive study related to the problem of scheduling projects with hard deadline jobs, which they called Time-Constrained Project Scheduling Problem (TCPSP). They used a non-regular objective function.
Researchers agree that a project cannot be insulated from its costs, or executed without the scheduling of activities. As the costs depend on the activities in progress and scheduling is related to other constraints than monetary, the researchers explicitly included cash-flows-resources-constraints in their formulations. Elmaghraby and Herroelen [5] lay down the following property of an optimal solution that maximizes the NPV: the activities with positive cash flows should be scheduled as soon as possible and those with negative cash flow as late as possible. They concluded that the faster conclusion of the project is not necessarily the optimal solution with regard to maximizing the NPV. In Mika et al. [6] study, a positive flow is associated to each activity. The objective is to maximize the NPV of all cash flows of the project. They use two meta-heuristics that are widely used in research: Simulated Annealing (SA) and Tabu Search (TS).
Tereso et al's. research ([7] [8] [9] [10] [11] [12]) is included in the minimum-cost class problems. Tereso et al. [7] implemented a dynamic programming model for multimodal activities projects, with stochastic work content using Matlab. A recent metaheuristic, the Electromagnetism-Like Mechanism (EM), developed by Birbil and Fang [13], was implemented in Tereso et al. [8] in Matlab for the same class of projects, obtaining improved results, although authors mention that the computational performance could still be improved. Such improvement was presented later in Tereso et al. [9] with an enhanced implementation using the JAVA programming language. In Tereso et al. [10] [11] a dynamic programming model was developed on a distributed platform, with better results in terms of

computing performance than the previous implementation in Matlab. Another application to the same problem may also be found in Tereso et al. [12], the Evolutionary Algorithm (EA), that is a generic population-based metaheuristic optimization algorithm.
Constraints complicate the efficient optimization of problems, and the more accurately they describe the real problem, the more difficult it is to handle. Recent models include most of the requirements described by Willis [14] for modeling realistic resources. These requirements include the variable need of resources according to the duration of activity, variable availability of resources over the period of the project and different operational modes for the activities.
A discrete time/resource function implies the representation of an activity in different modes of operation. Each mode of operation has its own duration and amount of renewable and non renewable resources requirement.
Boctor [15] presented a heuristic procedure for the scheduling of non-preemptive resource-limited projects, although renewable from period to period. Each activity had a set of possible durations and resource requirements. The objective was to minimize the project duration. A general framework to solve large-scale problems was suggested. The heuristic rules that can be used in this framework were evaluated, and a strategy to solve these problems efficiently was designed. Heilmann [16] also worked with the multi-mode case in order to minimize the duration of the project. In his work, besides the different modes of execution of each activity, there is specified a maximum and minimum delay between activities. He presented a priority rule-based heuristic. Basnet [17] presented a filtered beam search technique to generate makespan minimizing schedule, for multi-mode single resource constrained projects, where there is a single renewable resource to consider and the multi-mode consists essentially of how many people can be employed to finish an activity.
In this paper we present a multi-mode case, the MRCPSP-MS, where activities can be scheduled at different modes, each mode using a different resource level/skill, implying different costs and durations.

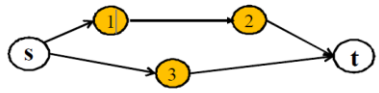
2. Problem description

Consider a project network in the activity-on-arc (AoA) representation: $G = (N, A)$, with $|N| = n$ (representing the events) and $|A| = m$ (representing the activities). Each activity may require the simultaneous use of several resources with different resource consumption according to the selected execution mode - each resource may be deployed at a different level. It is desired to determine the optimal resources allocation to the activities that minimizes the total cost of the project (resources + penalty for tardiness + bonus for earliness). We follow the dictum that an activity should be initiated as soon as it is sequence-feasible.
There are $|R| = \rho$ resources. A resource has a capacity of several units (say w workers or m/c's) and may be used at different levels, such as a 'resource' of electricians of different skill levels, or a 'resource' of milling machines but of different capacities and ages. A *level* may also be the amount of hours used by a resource; for example, half-time, normal time or extra-time. The different levels of a resource $r \in R$ may be represented as $L(r) = \{r_1, \dots r_{L(r)}\}$; the number of levels varies with the resource. If resource r is utilized at level l for activity j then the processing time of the activity shall be denoted by $p(j, l(j, r))$. An activity normally requires the simultaneous utilization of more than one resource for its execution.
To better visualize the problem one can summarize its characteristics in a matrix format as shown in table 1. For illustrative purposes we assume that any resource may have at most three levels: low (level 1), average or normal (level 2), and high (level 3). A cell entry in the matrix is the processing time $p(j, l(j, r))$ of activity j when accorded resource r at level $l(j, r)$. Due to space limitations, table 1 exhibits the information as $(j, (r, l))$; the symbol "p" is forfeited. If an activity does not require a resource; this is indicated in the matrix by the symbol $\emptyset$ (null). The symbol a_r gives the number of units available of resource r.

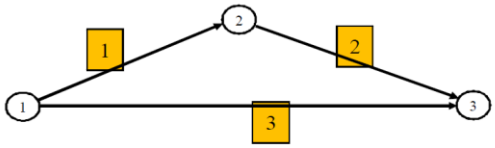
Table 1: Problem Characteristics

↓Act\Res→	1 (a_1)	……	$\|R\| = \rho$ (a_ρ)
1	$(1,1,1)$ $(1,1,2)$ $(1,1,3)$	…	$(1,\rho,1)$ $(1,\rho,2)$ $(1,\rho,3)$
2	$(2,1,1)$ $(2,1,2)$ $(2,1,3)$	…	$(2,\rho,1)$ $(2,\rho,2)$ $(2,\rho,3)$
⋮	⋮	⋮	⋮
$\|A\| = m$	$(m,1,1)$ $(m,1,2)$ $(m,1,3)$	…	$(m,\rho,1)$ $(m,\rho,2)$ $(m,\rho,3)$

The processing time of an activity is given by the maximum of the durations that would result from a specific allocation of the various resources. To better understand this representation, consider the miniscule project of figure 1 with three-activities. Assume that the project requires the utilization of four resources.



Project with 3 activities. AoN



AoA representation.

Figure 1: Project with three-activities

With little additional data processing the problem can be enriched with the inclusion of the cost of the resource utilization at each level. Let $\gamma_{l(r)}$ be the unitary cost of resource r at level l. Then in each cell there shall be a $2 \times |L(j,r)|$ matrix in which the first row gives the duration $p(j,r,l(j,r))$ and the second row the cost $c(j,r,l(j,r))$. The total cost of a resource, at each level, is obtained as the product of the resource unitary cost with the activity processing time.

Table 2: Resource Requirements, Processing Times and Resource Costs of Project.

↓Act	Resources →	1 (2)	2 (1)	3 (3)	4 (2)	η_j
	Res. Unit cost	(1,3)	(2,5,10)	(1,3,5)	(2,4,6)	
1	$p(j,r,l(j,r))$	(14,6)	Ø	(12,8,5)	(18,12,7)	3
	$c(j,r,l(j,r))$	(14,18)		(12,24,25)	(36,48,42)	
2	$p(j,r,l(j,r))$	Ø	(7,5,3)	Ø	(8,5,4)	2
	$c(j,r,l(j,r))$		(14,25,30)		(16,20,24)	
3	$p(j,r,l(j,r))$	(20,12)	(22,16,10)	Ø	Ø	2
	$c(j,r,l(j,r))$	(20, 36)	(44,80,100)			

η_j= count of resources required for activity j.

For brevity, table 2 gives only the processing times and costs at each level of the resources; the identity of the activity and the resource are suppressed.
The top row indicates that there are 4 resources with varying availability: resources 1 and 4 have $b1 = 2 = b4$ units each, resource 2 has only one unit ($b2 = 1$), and resource 3 has $b3 = 3$ units. The following row specifies a cost rate for each level of the resource. A positive entry in the row corresponding to activity j indicates the resource(s) required by the activity - each activity must be allocated exactly one unit of each resource. However, the resource may be utilized at any of its specified levels. For instance, resource 2 has only 1 unit available, which can be utilized at any of its 3 levels: if, for activity 2, it is utilized at level 1 (the lowest) then the processing time is 7 (i.e.; $p\ (2;\ 2,1) = 7$) and the cost is 14 (i.e.; $c\ (2;\ 2,1) = 14$); if it is utilized at level 2 (the intermediate) then the processing time is 5 and the cost is 25; finally, if it is utilized at level 3 (the highest) then the processing time is 3 and the cost is 30; etc.
Suppose that all four resources are allocated at their respective level 2 (normal intensity). Letting p_j denote the duration of activity j and c_j de total cost of activity j, we get:

- activity 1 shall take $p1 = max\{6, 8, 12\} = 12$ time units; $c_j = 18 + 24 + 48 = 90$ monetary units;
- activity 2 shall take $p2 = max\{5, 5\} = 5$ time units; $c_j = 25 + 20 = 45$ monetary units;
- activity 3 shall take $p3 = max\{12, 16\} = 16$ time units; $c_j = 36 + 80 = 116$ monetary units;

and the project shall consume $max\{12 + 5, 16\} = 17$ time units to complete, however due to resource restrictions, activities 2 and 3 cannot be executing at the same time since resource 2 has only one unit which must be allocated to either activity. So if allocated first to activity 2 the project should consume

$max\{12 + 5, 17 + 16\} = 33$ time units; if allocated first to activity 3 the project should consume $max\{16 + 5, 16\} = 21$ time units. The latter decision should be the preferred one. So the total resource cost of the project shall be 251 monetary units and will be finished at time $T = 21$ (assuming the project started at time $T = 0$). We also consider lateness costs and earliness gains (negative costs). If the project specified due date is $Ts = 24$, the project will finish early. If the unitary cost for earliness is equal to -10, the total cost will be $251 - 10 * 3 = 221$.

2.1. Mathematical Model

Briefly, the constraints of this problem are:

- Respect the precedence among the activities.
- A unit of the resource is allocated to at most one activity at any time (the unit of the resource may be idle during an interval) at one level.
- Respect the capacity of the resource availability: The total units allocated at any time should not exceed the capacity of the resource to which these units belong.
- An activity can be started only when it is sequence-feasible and all the requisite resources are available, each perhaps at its own level, and must continue at the same levels of all the resources without interruption or preemption.

The objective adopted is to find the optimal solution that minimizes the overall project cost, while respecting a specified delivery date. A penalty is incurred for tardiness beyond the specified delivery date, and a reward is secured for early completion.

Let:

$G(N, A)$: Project network in AoA representation, with a set of N nodes, representing the events and A activities.
n: number of nodes; $n = |N|$.
m: number of arcs or number of activities; $m = |A|$.
a : activity, which may also be represented by arc (i, j).
r: resource $r \in |R|$
C^k: the *k*th uniformly directed cutset *(udc)* of the project network that is traversed by the project progression; $k = 1, \dots, K$.
l: level at which a resource is applied to an activity.
$x(a, r, l)$: a binary variable, of value 1 if resource r is allocated to activity a at level l, and 0 otherwise.
$p(a, r, l)$: the processing time of activity a when resource *r* is allocated at level l.
$p(a)$: processing time of the activity a (considering all resources).
$c(a, r, l)$: resource cost of activity a when resource *r* is allocated at level l.
$c_R(a)$: resource cost of the activity a (considering all resources).
η_a : the count of resources required by activity a.
ρ : number of resources, $\rho = |R|$.
b_r : capacity of resource r.
$\gamma(r, l)$: : marginal cost of resource r at level l.
γ_E : marginal gain from early completion of the project.
γ_L : marginal loss (penalty) from late completion of the project.
t_i : time of realization of node i (AoA representation), where node 1 is the "start node" of the project and node n its "end node".
T_s : target completion time of the project.
c_E: earliness cost.
c_T: tardiness cost.
c_{ET}: earliness-tardiness cost.
c_R : total resource cost for all project activities.
TC: total cost of the project.

The constraints are enumerated next.

We begin by defining the processing time of an activity $p(a)$ as the maximum of the processing times imposed by the different resources. These processing times will be a function of the levels at which the resources required by

the activity are allocated, so

$$p(a) = \max_{all\ r}\{p(a,r,l)\}, \forall\ a\ \in A \tag{1}$$

Respect the precedence among the activities: an activity cannot start before all the preceding activities have finished.

$$t_j - t_i \geq p(a), \forall\ a\ \in A \tag{2}$$

Respect the capacity of the resource availability: the total units of a resource allocated at any time to all the activities should not exceed the capacity of the resource to which these units belong. This restriction is applicable to the activities that are concurrently active, (i.e., on-going), which must lie in the same *uniformly directed cutset (udc).*

The total allocation of resource *r* to the active activities in the "current" *udc* C^k is given by

$$\sum_{a\in C^k} x(a,r,l)\ \leq\ b_r, \forall\ r \in R \tag{3}$$

A unit of a resource is allocated to an activity at only one level (the unit of the resource may be idle during an interval of time):

$$\sum_{forall\ l} x(a,r,l) = 1, \forall\ a\ , \forall\ r \in R \tag{4}$$

An activity must be allocated all the resources it needs at some level, at which time it can be started and must continue at the same level for all the resources without interruption or preemption. This requirement is represented by the equation:

$$\eta_a - \sum_{r\in R}\sum_{forall\ l} x(a,r,l) = 0, \forall\ a\ \in C^K \tag{5}$$

The difficulty in implementing this constraint set stems from the fact that we do not know a priori the identity of the *udc's* that shall be traversed during the execution of the project, since that depends on the resource allocations. A circularity of logic is present here: the allocation of the resources is bounded by their availabilities at each *udc*, but these latter cannot be known except after the allocations have been determined. Unfortunately, this vicious cycle cannot be broken by a blanket enumeration of all the *udc's* of the project because that would over-constrain the problem. There are several ways to resolve this circularity, formal as well as heuristic. The formal ones are of the integer programming genre which, when combined with the nonlinear mathematical programming model presented above, present a formidable computing burden. The heuristic approaches are more amenable to computing. Such a heuristic is used to solve this problem, as presented in section 3.
The objective function is composed of two parts: the cost of use of the resources, and the gain or loss due to earliness or tardiness; respectively, of the project completion time (t_n) relative to its due date.
Earliness and Tardiness are defined by:

$$e\ \geq \mathrm{T}_s - t_n \tag{6}$$

$$d \geq \mathrm{t}_n - T_s \tag{7}$$

$$e, d \geq 0 \tag{8}$$

The costs may be evaluated as follows:

a) the cost of resource utilization in the selected level, for each activity is:

$$c_R(a) = \sum_{r\in R} c(a,r,l) \tag{9}$$

$$c(a,r,l) = \gamma(r,l) * p(a,r,l) \tag{10}$$

b) the earliness-tardiness costs are:

$$c_{ET} = c_E + c_T = \gamma_E \cdot e + \gamma_L \cdot d \tag{11}$$

c) total resources cost for all activities of project:

$$c_R = \sum_{a \in A} c_R(a) \tag{12}$$

d) total cost of project:

$$TC = C_R + C_{ET} \tag{13}$$

The desired objective function may be written simply as:

$$\min \text{TC} \tag{14}$$

3. Description of the procedure used

The initial procedure we adopted, applied to a small project, was based in a breadth first search (BFS) algorithm. All the nodes (partial solutions) in the search tree are evaluated at each stage before going any deeper, subsequently implementing an *exhaustive search* that visits all nodes of the search tree. This strategy can be applied for small projects but becomes infeasible for larger ones.
The branch and bound search technique allows reducing the number of nodes being explored. It can be seen as a polished breadth first search, since it applies some criteria in order to reduce the BFS complexity. Usually it consists of keeping track of the best solution found so far and checking if the solution given by that node is greater than the best known solution. So if that node cannot, at best, offer a better solution than the solution obtained so far, the node is discarded. It is composed of two procedures, the *branching*, where a set of nodes build up smaller sets, creating the tree structure through a recursive application; and the *bounding*, where upper and lower bounds are calculated. The bounding function can be 'strong', which is usually harder to calculate but searches a smaller domain hence faster in finding the solution, or 'weak' which is easier to calculate but searches a larger domain and therefore may be slower in finding the solution. Branch and bound is really efficient if the bounds can be made very tight. In our case, the objective of our problem is to minimize the total cost of the project, that gets a bonus or a penalty cost depending on whether the project is completed earlier or later than a specified due date; respectively. As a result, finding a strong bounding function would be dependent on the project variables referred to above (penalty cost, bonus cost and due date). The feat of the bounding function is simply in reducing the search while not discarding potentially desirable branches. A filtered beam search is a non-rigorous (in the sense of not guaranteeing optimality) branch and bound procedure that uses breadth first search and where only the top few 'best' nodes are kept. At each stage of the tree, it generates all successors for each node at the current stage, but only stores a predetermined (smaller) number of descendents at each node, called the beam width, evaluating them according to some heuristic. We have decided to study an adaptation of the initial algorithm, presented below, to a filtered beam search algorithm that is still in development.

The first approach we exploit to arrive at the solution for this problem begins by the analyzing the network and the resource requirements. We started by considering the small network represented in figure 1 (it will be called network 1 from now on). We consider that activities go through 4 states: "to begin", "pending"," in progress" and "finished". To get the first activities to begin with, we search all activities that do not have any predecessors. These activities are set to state "to begin". All the others are set to the state "pending".
Activities in the state "to begin" are analyzed in order to check resources availability. If we have enough resources, all activities in the state "to begin" modify the state to "in progress", otherwise we apply, in sequence, the following rules, until resources conflict are resolved:

a) Give priority to activities that are precedent of a larger number of "pending activities".

b) Give priority to activities that use fewer resources.

c) Give priority to activities in sequence of arrival to the state "to begin".

So without resource conflicts, the project would initiate with activities 1 and 3 active, and after activity 1 finishes, activity 2 could be initiated and be active at the same time as activity 3, if activity 3 hadn't finished yet, by that time. However, activities 2 and 3 both need resource 2 that has only 1 unit available. Activity 2 has to wait until activity 3 finishes, to begin executing.
As introduced in the model an event represents the starting time of one or more activities. The project begins at event 0, with both activities 1 and 3 active (state "in progress"). There are enough resources to initiate these activities. Each activity must be allocated exactly one unit of each resource. For each active activity, we calculate all the possible combinations of resources levels. Then we join all activities combinations, getting the initial combinations of allocation modes for all active activities. These initial combinations form branches through which we will get possible solutions for the project. All combinations have a copy of resources availability information, and activities current state.

We continue applying the following procedure to each combination:

1. To all activities in progress, we find the ones that will be finished first, and set that time as the next event.
2. We update activities found in step 1 to state "finished", and release all the resources being used by them.
3. For all activities in the state "to begin", we check if they can begin, the same way we did when initiating the project. Activities in the state "to begin" are analyzed in order to check resources availability. If there are no resource conflicts, all activities in the state "to begin" are set to state "in progress" and resources are set as being used, otherwise we apply in sequence, the rules described above.
4. For all activities in the state "pending", we check for precedence relationships. For all activities that are precedence-feasible their state is updated to state "to begin".
5. If there are resources available, and any pending activities were set "to begin" we apply step 3 again.
6. For all new activities "in progress" we set its start time to next event found in step 1, and determine all the possible combinations of its resources levels. Then we join all found combinations for these activities, getting new combinations to join to the actual combination being analyzed. This forms new branches to process in order to get the project solution.
7. We continue by applying step 1 to each new combination until all activities are set to state "finished".
8. If all activities in a combination are set to state "finished", we have a valid project solution.

When all project solutions are found, we evaluate for each one, the finishing time of the project and the total project cost, finding the best one. This procedure was implemented in JAVA.

Consider the following data (table 3) for network 1.

Table 3: Resource Requirements, Processing Times and Resource Costs of the Project.

RESOURCE →	**1**			**2**			**3**			**4**			
AVAILABILITY	2			1			3			2			
↓ Activity \ Levels →	**1**	**2**		**1**	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**	
Unitary costs	2	4		3	5	7	1	4	5	1	3	5	η_j
A1(Processing time)	14	6	...	...	...	...	12	8	5	18	12	7	3
A1(Resource cost)	28	24	...	...	...	...	12	32	25	18	36	35	
A2(Processing time)	...	...	...	7	5	3	...	...	...	8	5	4	2
A2(Resource cost)	...	...	...	21	25	21	...	...	...	8	15	20	
A3(Processing time)	20	12	...	22	16	10	...	...	...	...	...	...	2
A3(Resource cost)	40	48	...	66	80	70	...	...	...	...	...	...	

Assume the following rates for earliness and lateness costs: $\gamma_E = -10$, $\gamma_L = 20$ and the due date $T_S = 24$.
Activity A1 has 18 combinations of resources levels ($2 * 3 * 3$) evaluated as a product of the number of levels of

each resource required for A1.
Activity A3 has 6 combinations of resources levels (2*3).
The project initiates with activity A1 an A3 active, so with 108 (18*6) branches to search solutions.
Activity 2 can only begin when activities A1 and A3 have finished, because of the precedence relationship with activity A1 and resource restrictions of resource 2, being used by activity A3.
When activity 2 is ready to begin we will generate 9 possible combinations of its resources levels (3*3).
These activity combinations will be joining existent project combinations, obtaining finally 972 valid solutions for the project (108*9).

After enumerating all possibilities, we can evaluate finishing time and total project cost for each solution. For the project represented on table 3 the solution obtained is:

Activity 1:

$x(1,1,1)$	$x(1,1,2)$	$c(1,1,1)$	$p(1,1,2)$	$x(1,3,1)$	$x(1,3,2)$	$x(1,3,3)$	$c(1,3,3)$	$p(1,3,3)$	$x(1,4,1)$	$x(1,4,2)$	$x(1,4,3)$	$c(1,4,3)$	$p(1,4,3)$	$C_R(1)$	$p(1)$
0	1	24	6	1	0	0	12	12	0	0	1	35	7,0	71	12,0

Activity 3:

$x(3,1,1)$	$x(3,1,2)$	$c(3,1,2)$	$p(3,1,1)$	$x(3,2,1)$	$x(3,2,2)$	$x(3,2,3)$	$c(3,2,3)$	$p(3,2,3)$	$C_R(3)$	$p(3)$
0	1	48	12	0	0	1	70	10,0	118	12,0

Activity 2:

$x(2,2,1)$	$x(2,2,2)$	$x(2,2,3)$	$c(2,2,3)$	$p(2,2,3)$	$x(2,4,1)$	$x(2,4,2)$	$x(2,4,3)$	$c(2,4,2)$	$p(2,4,2)$	$C_R(2)$	$p(2)$
0	0	1	21	3,0	0	0	1	20	4,0	41	4,0

t_n	$T_s - t_n$	$e >= T_s - t_n$	C_E	$t_n - T_s$	$d >= t_n - T_s$	C_T	$C_E + C_T$	C_R	$TC = C_{ET} + C_R$
16,0	8,0	8,0	-80,0	-8,0	0,0	0,0	-80,0	230	150,0

So the project finishing time is $t_n = 16$. Since the due date was 24, a bonus will be applied. The project cost without bonus or penalty is equal to the cost of the resources $C_R = 230$. The total cost of project is $TC = 150$.

Activities execution modes are:

- A1 start: 0; R1 (level 2; cost: 24; duration: 6); R3 (level 1; cost: 12; duration: 12); R4 (level 3; cost: 35; duration: 7).
- A3 start: 0; R1 (level 2; cost: 48; duration: 12); R2 (level 3; cost: 70; duration: 10).
- A2 start: 12; R2 (level 3; cost: 21; duration: 3); R4 (level 3; cost: 20; duration: 4).

4. Conclusions

In this paper we provide a formal model to the MRCPSP-MS problem and a breath-first procedure description, for an optimal allocation of resources in a project, with multi-mode activities, minimizing its total cost, while respecting all the restrictions. We implemented the application using the object oriented paradigm language, JAVA.
We achieved the optimal solution for a simple 3 activities project network, by obtaining all possible solutions and search the best between them. Although our procedure implemented heuristics rules, there application doesn't obstruct the achieving of the optimal solution.
In future work, we intend to complete the adaptation of a filtered beam search algorithm to this problem, analyzing and obtaining results for more complex project networks. We intend to study as well the computational performance and acquire statistics about the procedure efficiency, since this problem is a very difficult problem to solve (NP-hard).

References

[1] L. Ozdamar and G. Ulusoy, A Survey on the Resource-Constrained Project Scheduling Problem, *IIE Transactions*, 27, 574-586, 1995.

[2] J.H. Patterson, R. Slowinski, F.B. Talbot and J. Weglarz, An algorithm for a general class of precedence and resource constrained scheduling problems, *Advances in Project Scheduling*, Amsterdam, 3-28, 1989.

[3] J.H. Patterson, F.B. Talbot, R. Slowinski and J. Weglarz, Computational experience with a backtracking algorithm for solving a general class of precedence and resource constrained scheduling problems, *European Journal of Operational Research*, 49, 68-7, 1990.

[4] T. Guldemond, J. Hurink, J. Paulus and J. Schutten, Time-constrained project scheduling, *Journal of Scheduling*, 11 (2), 137-148, 2008.

[5] S.E. Elmaghraby and W.S. Herroelen, The scheduling of activities to maximize the net present value of projects, *European Journal of Operational Research*, 49, 35-40, 1990.

[6] M. Mika, G. Waligora and G. Weglarz, Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models, *European Journal of Operational Research*, 164 (3), 639-668, 2005.

[7] A.P. Tereso, M.M. Araújo and S.E. Elmaghraby, Adaptive Resource Allocation in Multimodal Activity Networks, *International Journal of Production Economics*, 92, 1-10, 2004.

[8] A.P. Tereso, M.M. Araújo and S.E. Elmaghraby, The Optimal Resource Allocation in Stochastic Activity Networks via The Electromagnetism Approach, *Ninth International workshop on Project Management and Scheduling (PMS'04)*, Nancy-France, 26-28 April, 2004.

[9] A.P. Tereso, M.M. Araújo and S.E. Elmaghraby, Optimal resource allocation in stochastic activity networks via the electromagnetic approach: a platform implementation in Java, *Control & Cybernetics*, 38, 745-782, 2009.

[10] A.P. Tereso, J.R Mota and R.J. Lameiro, Adaptive Resource Allocation Technique to Stochastic Multimodal Projects: a distributed platform implementation in JAVA, *IFORS Triennial Conference (IFORS'05)*, Honolulu-Hawaii-USA, 11-15 July, 2005.

[11] A.P. Tereso, J.R Mota and R.J. Lameiro, Adaptive Resource Allocation Technique to Stochastic Multimodal Projects: a distributed platform implementation in JAVA, *Control and Cybernetics*, 35, 661-686, 2006.

[12] A.P. Tereso, L. Costa, R. Novais and M.M. Araújo, The Optimal Resource Allocation in Stochastic Activity Networks via the Evolutionary Approach: a platform implementation in Java, *International Conference on Industrial Engineering and Systems Management*, Beijing – China, 2007.

[13] S.I. Birbil and S.C Fang, An Electromagnetism like Mechanism for Global Optimization, *Journal of Global Optimization*, 25, 263-282, 2003.

[14] R.J. Willis, Critical path analysis and resource constrained project scheduling theory and practice, *European Journal of Operational Research*, 21, 149-155, 1985.

[15] F.F. Boctor, Heuristics for scheduling projects with resource restrictions and several resource-duration modes, *International Journal of Production Research*, 31, 2547-2558, 1993.

[16] R. Heilmann, Resource–constrained project scheduling: a heuristic for the multi–mode case, *OR Spectrum*, 23(3), 335–357, 2001.

[17] C. Basnet, G. Tang and T. Yamaguchi, A Beam Search Heuristic for Multi-Mode Single Resource Constrained Project Scheduling, *Proceedings of the 36th Annual Conference of the Operational Research Society of New Zealand*, Christchurch, NZ, Nov-Dec, 1-8, 2001.