# On The Multi-Mode, Multi-Skill Resource Constrained Project Scheduling Problem – A Software Application

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

**Abstract** We consider an extension of the Resource-Constrained Project Scheduling Problem (RCPSP) to multi-level (or multi-mode) activities. 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 which includes a penalty for tardiness beyond the specified delivery date as well as a bonus for early delivery. We give some of the most important solution details and we report on the preliminary results obtained. The implementation was designed using the C# language.

## 1 Introduction

This paper is concerned with an extension of the Resource-Constrained Project Scheduling Problem (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 study related to the problem of scheduling projects with hard deadline jobs, defined as a 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

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

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].

Constraints complicate the efficient optimization of problems, and the more accurately they describe the real problem, the more difficult it is to handle it. 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 the activities, variable availability of resources over the project duration 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 schedules, 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 a previous paper [18] we provided a formal model to the multi-mode, multi-skill resource constrained project scheduling (MRCPSP-MS) problem and a breadth-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 a procedure using the object oriented paradigm language, JAVA and achieved the optimal solution for a simple 3 activities project network, by obtaining all possible solutions and search the best between them. The plan was to complete an adaptation of a "filtered beam" search algorithm to this problem in the future; this report addresses this issue.

## *1.1 Problem description*

Consider a project network in the activity-on-arc (AoA) mode of 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. An activity normally requires the simultaneous utilization of more than one resource for its execution.

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 the resources required by the activity. The objective is to find the optimal solution that minimizes the overall project cost, while respecting a delivery date. 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 at a particular level (the unit of the resource may be idle during an interval).
- 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.

Figure 1 presents the mathematical model for the problem. For more information on this model refer to our previous paper [18].

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).
- $\eta_a$ : the count of resources required by activity $a$.
- $\rho$ : number of resources, $\rho = |R|$.
- $b_r$ : capacity of resource $r$.
- $\gamma(r,l)$:: marginal cost of resource $r$ at level $l$.
- $\gamma_E$ : marginal gain from early completion of the project.
- $\gamma_L$ : 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.

**Minimize TC**

**Subject to:**

$$p(a) \geq p(a,r,l) \text{ for all a, r and l}$$

$$t_j - t_i \geq p(a), \forall\, a \in A$$

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

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

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

**Where:**

$$TC = C_R + C_{ET}$$

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

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

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

$$c(a,r,l) = \gamma(r,l) * p(a,r,l)$$

$$e \geq T_s - t_n$$

$$d \geq t_n - T_s$$

$$e, d \geq 0$$

**Fig. 1** Mathematical Model

## 2 Solution Details

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 were 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 (BaB) 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 offer a better solution than the solution obtained so far, the node is discarded. The BaB process consists of two procedures: subset generation and subset elimination. The former (the subset generation) is accomplished by *branching,* where a set of descendent nodes, form a tree-like structure. The latter (subset elimination) is realized through *bounding*, where upper and lower bounds are calculated for the "value" of each node. The bounding function can be strong, which is usually harder to calculate but faster in finding the solution, or weak, which is easier to calculate but slower in finding the solution. The BaB approach is more 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 while respecting or exceeding the specified due date; respectively. As a result, finding a strong bounding function would depend on the three project parameters cited: the 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 heuristic BaB procedure that uses breadth first search but only the top "best" nodes are kept. At each stage of the tree, it generates all successors for the selected nodes at the current stage, but only stores a predetermined number of descendent nodes at each stage, called the *beam width*. This paper is concerned with the study of the adaptation of the initial algorithm, presented below, to a "filtered beam" search procedure.

## *2.1 Procedure description*

The procedure to be executed can be based either on the BFS algorithm or on the Beam Search Algorithm. If the latter is the one adopted a beam width value must be defined. We consider that activities can be in one of four states: "to begin", "pending", "active" and "finished". To get the first activities with which to initiate the process, we search all activities that do not have any predecessors. These activities are set to state "to begin". All 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:

1. Give priority to activities that are precedent of a larger number of "pending activities".
2. Give priority to activities that use fewer resources.
3. Give priority to activities in sequence of arrival to the state "to begin".

An "*event*" represents the starting time of one or more activities and the project begins at event 0. 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.

If the algorithm set to find the best solution is the *Beam Search Algorithm*, then:

1. If the number of combinations is less than the beam width value, all combinations are kept.
2. Otherwise, the set of combinations must be reduced to the beam width value. In this case some combinations need to be discarded using a defined rule to evaluate the ones in the top best. The possible rules for selection are:

   Select top best combinations that have:
   - Minimum Duration.
   - Minimum Cost.
   - Minimum Cost/Duration.

In either case, we continue applying the following procedure to each combination:

3. To all activities in progress, we find the ones that will be finished first, and set that time as the next *event*.
4. We update activities found in step 1 to state "finished", and release all the resources being used by them.
5. 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 "active" and resources are set as being used, otherwise we apply in sequence, the rules described above.
6. 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". These activities aren't combined to the previous set of "to begin" activities to give priority to activities that entered first in this state.
7. If there are resources available, and any pending activities were set "to begin" we apply step 5 again.
8. For all new activities "in progress" we set their start time to the next *event* found in step 3, 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.
9. We continue by applying step 1 (or 3) to each new combination until all activities are set to state "finished".
10. Once all activities in a combination are set to state "finished", we have a valid project solution.

When the project final solutions are found, we evaluate, for each one, the finishing time of the project and the total project cost, choosing the best one.

The BaB and the Beam Search procedures are typical methods applied to the RCPSP. The differentiating aspects of our approach are, on one hand the definition of the set of states followed by the activities, combined with the priority rules used to solve resource conflicts, and on the other hand the alternative evaluation rules used to discard undesirable "branches".

## *2.2 Application Development*

The software was developed in C# language using Visual Studio 2010. To construct the project network (in AoN), we use Graph#, an open source library for .Net/WPF applications that is based on a previous library QuickGraph. These libraries support GraphML that is an XML-based file format for graphs, although we didn't make use of this format. The graph is automatically generated for each project loaded in the application. To save/load existing projects we define an *xml* file that embodies all project characteristics for this problem.

### 2.1.1 Data Model

Three main classes were defined for the application. The base *class* is *NetProject* that keeps all project required information: name, activities, resources, due date, bonus and penalty cost. Then we have the *Resource class* that keeps the resource identification availability and levels. Each resource level has a unitary cost. The *Activity class* has activity identification, resources requirement and its precedents. The referred classes are the most relevant to represent the project, additional classes are used to support the evaluation of the project solution.

### 2.2.2 Functionalities

The application provides the functionalities described next.

- Load a Project.
  - The project must be saved as an xml file, using a structure that represents the project components (activities, resources, etc.).
- Create a Project:
  - There are two main steps to create a new project:
    - First the project "skeleton" is built through a wizard that initiates asking the project name and the number of resources and activities. Next the resource data is introduced namely the availability of each resource and the number of associated levels. Finally

the activities information is introduced namely the identification and precedents of each activity.
  - o Secondly it generates the project graph and a project grid where the remaining project information can be introduced.

- Edit/Save a Project.
- Determinate best solution:
  – This can be achieved using a BFS based Algorithm or a Beam Search Algorithm.
- Save solution to a txt file.

We present next the application look, using some prints.

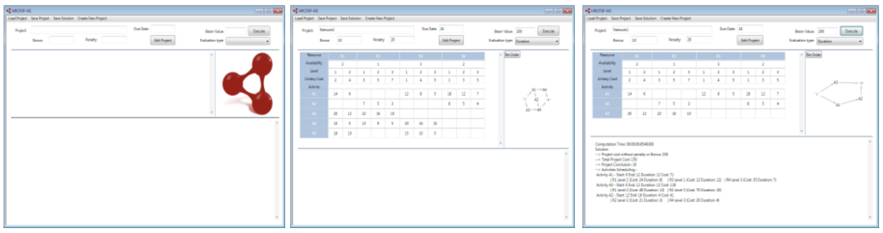

**Fig. 2** Application prints

## 3 Preliminary results

The next computational tests were performed on an Intel® Pentium® M @1.20GHz 1.25GB RAM.

Consider the following data (table 1) for a three activities network.

**Table 1** 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 | $\eta_j$ |
| 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 $T_S = 24$.

Using the BFS Algorithm the project obtained solution is presented in table 2.

**Table 2** Solution totals, obtained using BFS Algorithm.

| $t_n$ | $C_E$ | $C_T$ | $C_R$ | TC | Runtime (ms) |
|---|---|---|---|---|---|
| 16,0 | 80,0 | 0,0 | 230 | 150,0 | 66 |

Since the due date was 24, a bonus is applied. Activities execution modes are:

- Activity A1 - Start: 0 End: 12 Duration: 12 Cost: 71
  | R1 Level 2 (Cost: 24 Duration: 6) | R3 Level 1 (Cost: 12 Duration: 12) | R4 Level 3 (Cost: 35 Duration: 7)
- Activity A3 - Start: 0 End: 12 Duration: 12 Cost: 118
  | R1 Level 2 (Cost: 48 Duration: 12) | R2 Level 3 (Cost: 70 Duration: 10)
- Activity A2 - Start: 12 End: 16 Duration: 4 Cost: 41
  | R2 Level 3 (Cost: 21 Duration: 3) | R4 Level 3 (Cost: 20 Duration: 4)

**Table 3** Solution totals, obtained using Beam Search Algorithm.

| Beam Width | Evaluation Type | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cost | | | | | | Duration | | | | | | Cost/Duration | | | | | |
| | $t_n$ | $C_E$ | $C_T$ | $C_R$ | TC | Runtime (ms) | $t_n$ | $C_E$ | $C_T$ | $C_R$ | TC | Runtime (ms) | $t_n$ | $C_E$ | $C_T$ | $C_R$ | TC | Runtime (ms) |
| 150 | 26 | 0 | 40 | 201 | 241 | 33 | 16 | 80 | 0 | 230 | 150 | 52 | 26 | 0 | 40 | 201 | 241 | 42 |
| 200 | 26 | 0 | 40 | 201 | 241 | 48 | 16 | 80 | 0 | 230 | 150 | 68 | 26 | 0 | 40 | 201 | 241 | 69 |
| 700 | 20 | 40 | 0 | 240 | 200 | 253 | 16 | 80 | 0 | 230 | 150 | 319 | 20 | 40 | 0 | 240 | 200 | 372 |
| 900 | 16 | 80 | 0 | 231 | 151 | 419 | 16 | 80 | 0 | 230 | 150 | 482 | 16 | 80 | 0 | 231 | 151 | 607 |

The BFS Algorithm generates 972 combinations for the three activity network. We used a beam width between 150 and 900. As we can see by the results exhibited in table 3, the duration evaluation type was the best for this network, achieving the same result as the BFS Algorithm, even with the lowest beam width. The other evaluation types gave both the same result.

## 4 Conclusions

We developed a practical tool, useful to represent multi-mode projects, and to find a solution for the problem on hand – select the best mode for each resource in each activity in order to minimize the total cost, considering the resource cost, a penalty for tardiness and a bonus for early completion. We must continue testing the tool, in order to evaluate the quality of the solution obtained, since the heuristic used doesn't guarantee the optimum. Further experiments will also allow specifying the limits of its applicability in terms of the number of activities, the number of resources, and the number of alternative levels of resource application. Another useful effort is to compare as well the solutions obtained with both algorithms, trying to define a recommended "beam width" and evaluation type.

## References

1. Ozdamar L, Ulusoy G (1995) A Survey on the Resource-Constrained Project Scheduling Problem. IIE Transactions 27:574-586.
2. Patterson JH, Slowinski R, Talbot FB, Weglarz J (1989) An algorithm for a general class of precedence and resource constrained scheduling problems. Advances in Project Scheduling Amsterdam 3-28.
3. Patterson JH, Talbot FB et al (1990) Computational experience with a backtracking algorithm for solving a general class of precedence and resource constrained scheduling problems. Eur J Oper Res 49:68-7.
4. Guldemond T, Hurink J, Paulus J, Schutten J (2008) Time-constrained project scheduling. J Sched 11(2):137-148.
5. Elmaghraby SE, Herroelen WS (1990) The scheduling of activities to maximize the net present value of projects. Eur J Oper Res 49:35-40.
6. Mika M, Waligora G, Weglarz G (2005) Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models. Eur J Oper Res, 164 (3):639-668.
7. Tereso AP, Araújo MM, Elmaghraby SE (2004) Adaptive Resource Allocation in Multimodal Activity Networks. Int J Prod Econ 92:1-10.
8. Tereso AP, Araújo MM, Elmaghraby SE (2004) The Optimal Resource Allocation in Stochastic Activity Networks via The Electromagnetism Approach. Ninth International workshop on PMS'04 Nancy-France 26-28.
9. Tereso AP, Araújo MM, Elmaghraby SE (2009) Optimal resource allocation in stochastic activity networks via the electromagnetic approach: a platform implementation in Java. Control Cybern 38:745-782.
10. Tereso AP, Mota JR, Lameiro RJ (2005) Adaptive Resource Allocation Technique to Stochastic Multimodal Projects: a distributed platform implementation in JAVA. IFORS'05 Honolulu-Hawaii-USA 11-15.
11. Tereso AP, Mota JR, Lameiro RJ (2006) Adaptive Resource Allocation Technique to Stochastic Multimodal Projects: a distributed platform implementation in JAVA. Control Cybern 35:661-686.
12. Tereso AP, Costa L, Novais R, Araújo MM, (2007) The Optimal Resource Allocation in Stochastic Activity Networks via the Evolutionary Approach: a platform implementation in Java. ICIESM 2010 Beijing China.
13. Birbil SI, Fang SC (2003) An Electromagnetism like Mechanism for Global Optimization. J Global Optim 25:263-282.
14. Willis RJ (1985) Critical path analysis and resource constrained project scheduling theory and practice. Eur J Oper Res 21:149-155.
15. Boctor FF (1993) Heuristics for scheduling projects with resource restrictions and several resource-duration modes. Int J Prod Res 31:2547-2558.
16. Heilmann R (2001) Resource–constrained project scheduling: a heuristic for the multi–mode case. OR Spectrum 23(3):335–357.
17. Basnet C, Tang G, Yamaguchi T (2001) A Beam Search Heuristic for Multi-Mode Single Resource Constrained Project Scheduling. Proceedings of the 36th Annual ORSNZ Conference Christchurch NZ Nov-Dec 1-8.
18. Santos MA, Tereso AP (2010) On the Multi-Mode, Multi-Skill Resource Constraint Project Scheduling Problem (MRCPSP-MS). EngOpt 2010 Lisbon Portugal September 6-9.