1st International Conference on Project Economic Evaluation
ICOPEV'2011, Guimarães, Portugal

# ON THE MULTI-MODE, MULTI-SKILL RESOURCE CONSTRAINED PROJECT SCHEDULING PROBLEM – COMPUTATIONAL RESULTS

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

[1] Department of Production and Systems, University of Minho, Portugal

* Corresponding author: pg13713@alunos.uminho.pt, University of Minho, 4800-058 Guimarães, Portugal

**KEYWORDS**
RCPSP, Multi-mode, Beam search

**ABSTRACT**

This paper is concerned with an extension of the Resource-Constrained Project Scheduling Problem (RCPSP) which belongs to the class of the optimization scheduling problems with multi-level (or multi-mode) activities. 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 implemented an adaptation of a filtered beam search (FBS) algorithm to this problem, using the C# programming language. A "filtered beam" search is a heuristic Branch and Bound (BaB) procedure that uses breadth first search but only the top "best" nodes are kept. We give some of the most important solution details and we report on further computational results, by testing the application for different problem sizes.

## INTRODUCTION

The resource-constrained project scheduling problem (RCPSP) has been demonstrated to be in the class of NP-hard problems (Blazewicz et al. 1983). The need to solve real problems in reasonable time led researchers to develop heuristic procedures.

The heuristics being used belong to one of two classes: the class of priority rule-based methods or the class of meta-heuristics approaches. The priority rule-based methods build a plan by selecting activities from a range of activities available successively so that all activities are sequenced (Boctor, 1993; Dean et al. 1992; Heilmann, 2001). The meta-heuristics based methods begin with an initial solution and try to improve it. The improvement of a solution is obtained by transforming one or several solutions into new ones. There are still two types of heuristics, series heuristics where the priority of the activities is predetermined and remains fixed, and parallel heuristics where the priority is updated each time an activity is scheduled for processing.

Another type of heuristics found in the literature, are sub-fields of meta-heuristics such as tabu search, simulated annealing (Mika et al., 2005) and genetic algorithms. Gonçalves et al. (2004) presented a genetic algorithm for the RCPSP problem. They used a chromosome representation based on random keys. Tseng (2008) also discussed the use of genetic algorithms applied to the multi-project, multi-mode RCPS problems (MMRCMPSP).

It is also usual to find integer programming models applied to single and multi-project environments. Multi-project problems are indirectly analyzed using single project procedures and considering the parallel projects as parallel sequences of activities with the same start and end nodes.

The profusion of binary variables and constraints has led researchers to develop branch-and-bound (BaB) procedures for the problem. The success of this technique depends on the branching technique and on the tightness of its lower limit.

Kis (2005) concentrates on the scheduling problem where the need for resources for each activity varies in proportion to the intensity of the activity itself. To formalize the problem he used an integer linear programming model and proposed a BaB algorithm to find the optimal solution. However BaB procedures are inadequate for real size problems despite their efficiency relative to a frontal attack on the discrete optimization problem.

Another recent paradigm is the Electromagnetism-like Mechanism (EM). Tereso et al. (2004b) presented an application of the EM to stochastic multimodal projects that had been studied before using Dynamic Programming (DP) (Tereso et al., 2004a). The DP model was developed later on a distributed platform (Tereso et al., 2006). Improved results of the EM, in terms of computing performance, with an enhanced application using JAVA, were also obtained (Tereso et al., 2007).

In the several resource constrained scheduling problem models found in the literature, there are two important elements: the objective function and the constraints. Constraints complicate the efficient optimization of problems, and the more accurately they describe the real problem, the more difficult it is to handle. Willis (1985) described requirements for modeling realistic resources.

These requirements include the variable need of resources according to the duration of the 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 resource requirements.

Boctor (1993) presented a heuristic procedure for the scheduling of non-preemptive resource-limited projects, when the resources are renewable over time. 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 (2001) 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. The problem presented here also belongs to the class of the optimization scheduling problems with multi-level (or multi-mode) activities; i.e., the activities can be scheduled in different modes, each mode using a different resource level/skill, implying different costs and durations.

The objective may be based on time, such as minimize the project duration (Boctor, 1993; Heilmann, 2001; Basnet et al., 2001; Guldemond et al., 2008) or on economic aspects, such as minimize the project cost (Mika et al., 2005; Tereso et al., 2004a; Tereso et al., 2006). However, success relative to time does not imply success in economic terms. A recurrent situation encountered in practice is the need to complete a project by its due date and maximize profit. Özdamar and Ulusoy (1995) reported in their survey of the literature, studies where the NPV is maximized while the due date is a 'hard' constraint (Patterson et al., 1989, 1990). 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 (1990) 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 earlier conclusion of the project is not necessarily the optimal solution with regard to maximizing the NPV. In Mika et al. (2005) study, a positive flow is associated with each activity. The objective was to maximize the NPV of all cash flows of the project.

**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. An activity may be initiated as soon as it is sequence-feasible, subject to resources availability. 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 over-time. The processing time of an activity is given by the maximum of the durations that would result from a specific allocation of resources. Each activity must be allocated exactly one unit of each required resource and the resource unit may be used at any of its stated levels. The objective is to determine the optimal allocation of the resources to the activities that minimizes the total project cost (resources + penalty for tardiness + bonus for earliness), while respecting a delivery date. Briefly, the constraints of this problem are:

a) Respect the precedence among the activities.
b) A unit of the resource is allocated to at most one activity at any time at a particular level.
c) Respect the capacity of the resource availability.
d) 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 resources levels without interruption or preemption.

In a previous paper we provided a formal model to the multi-mode, multi-skill resource constrained project scheduling problem (MRCPSP-MS) (Figure 1) and a breadth-first procedure description (Santos and Tereso, 2010a).

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; $k = 1, \ldots, K$.
- $l$: 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:**

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

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

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

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

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

**Where:**

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

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

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

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

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

$$e \geq T_s - t_n$$

$$d \geq t_n - T_s$$

$$e, d \geq 0$$

Figure 1: Mathematical Model

Then in Santos and Tereso (2010b) we presented an adaptation of a filtered beam search (FBS) algorithm to this problem, using C# programming language and reported on the preliminary results obtained for small project networks. The application developed allowed determining the project solution using either the Breadth First Search (BFS) algorithm or the Beam Search Algorithm procedure. In this paper we report on further computational results, by testing the application for different problem sizes.

## SOLUTION DETAILS

In the BFS algorithm, all the nodes (partial solutions) in the search tree are evaluated at each stage before going any deeper, subsequently realizing an exhaustive search that visits all nodes of the search tree. The branch and bound (BaB) search technique can be seen as a polished BFS, since it applies some criteria in order to reduce the BFS complexity. 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 are generated, creating a tree-like structure. The latter (subset elimination) is realized through either bounding, where upper and lower bounds are evaluated for the "value" of each node, or feasibility, where the extension of a partial solution is deemed infeasible, and the branch is aborted. The bounding function can be strong, which is usually harder to calculate but faster in finding the optimal 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 (which includes a bonus for early completion or a penalty for exceeding the specified due date). As a result, finding a strong bounding function would depend on the three project parameters cited: the penalty cost, the bonus cost and the due date. A FBS is a heuristic BaB procedure that uses BFS 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 stores only a preset number of descendent nodes at each stage, called the *beam width*. Basnet et al. (2001) presented a FBS approach 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 basically of how many people can be employed to finish an activity.

The BaB and the Beam Search procedures are typical methods applied to the RCPSP (Basnet et al., 2001; Demeulemeester and Herroelen, 1996; Kis, 2005). The differentiating aspects of our approach are, first, the definition of a set of states defined by the condition of the activities, combined with the priority rules used to solve resource conflicts, and second, the alternative evaluation rules used to discard "undesirable branches".

### Procedure description

The procedure to be executed can be based either on the BFS algorithm or on the Filtered 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*"

(i.e., on-going) 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*" are modified to the state "*active*"; otherwise we apply, in sequence, the following rules, until resources conflict are resolved:

1. Give priority to activities that are precedents to 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 in which no activity has started yet. 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 the resources availability information, and activities' current state.

If the algorithm selected 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 the possible rules to evaluate the ones in the top best:

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

   Not all combinations of the set can be directly compared, because the number of activities that have been scheduled in each combination may differ. So the combinations are grouped by the number of activities that have been already schedule.

   Then the combinations are compared with the others that belong to the same group. The final set is composed by a share of combinations of each group formed before.

   The ratio of each group in the final combinations set is calculated by:

$$ratio = groupCount / totalCombinations \quad (1)$$

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 seek the ones that 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 no resource conflicts exists, 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 are not combined with 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 "*active*" activities we set their start time to the next event found in step 3, and determine all the possible combinations of its resource levels. Then we join all found combinations for these activities, getting new combinations to add to the actual combination being analyzed. This forms new branches to the 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*", at which time we have a valid project solution.

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

**Application Development**

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. These classes are the most relevant to represent the project. Additional classes are used to support the evaluation of the project solution.

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.

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:
  1. 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.
  2. 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 Breadth First Search based Algorithm or a Beam Search Algorithm.
- Save solution to a txt file.

Figure 2 shows the application appearance:

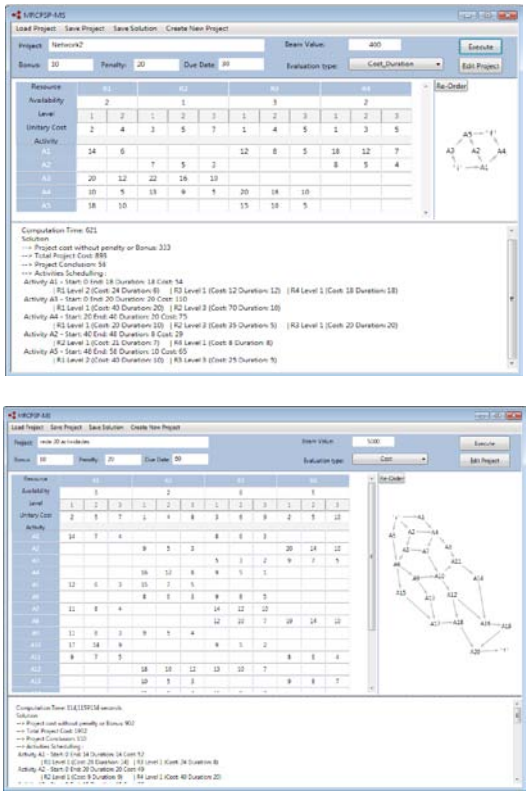

Figure 2: Application appearance

## COMPUTATIONAL RESULTS

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

Consider a three activities network, using 4 resources, one with 2 levels and the others with 3 different levels respectively. Assume the following rates for earliness and lateness costs: $\gamma_E = -10$, $\gamma_L = 20$ and the due date $T_S = 24$.

Table 1: Three activities network solution totals, obtained using BFS Algorithm.

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

Table 2: Three activities network 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 (s) | $t_n$ | $C_E$ | $C_T$ | $C_R$ | TC | Runtime (s) | $t_n$ | $C_E$ | $C_T$ | $C_R$ | TC | Runtime (s) |
| 150 | 26 | 0 | 40 | 201 | 241 | 0,33 | 16 | 80 | 0 | 230 | 150 | 0,05 | 26 | 0 | 40 | 201 | 241 | 0,04 |
| 200 | 26 | 0 | 40 | 201 | 241 | 0,48 | 16 | 80 | 0 | 230 | 150 | 0,06 | 26 | 0 | 40 | 201 | 241 | 0,07 |
| 700 | 20 | 40 | 0 | 240 | 200 | 0,25 | 16 | 80 | 0 | 230 | 150 | 0,03 | 20 | 40 | 0 | 240 | 200 | 0,37 |
| 900 | 16 | 80 | 0 | 231 | 151 | 0,42 | 16 | 80 | 0 | 230 | 150 | 0,48 | 16 | 80 | 0 | 231 | 151 | 0,60 |

The BFS Algorithm generates 972 combinations for the three activity network. We try to use a beam width between 150 and 900. As we can see by the results exhibited in table 2, the *Duration* evaluation type was the best for this network, achieving the same result as the BFS Algorithm, even for the smaller beam width. The other two evaluation types gave the same result.

Consider a five activities network, using the same resources of the three activities network above. Assume the following rates for earliness and lateness costs: $\gamma_E = -10$, $\gamma_L = 20$ and the due date $T_S = 30$.

Table 3: Five activities network solution totals, obtained using BFS Algorithm.

| $t_n$ | $C_E$ | $C_T$ | $C_R$ | TC | Runtime (s) |
|---|---|---|---|---|---|
| 36,0 | 0,0 | 120,0 | 400 | 520,0 | 13,6 |

Table 4: Five activities network 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 (s) | $t_n$ | $C_E$ | $C_T$ | $C_R$ | TC | Runtime (s) | $t_n$ | $C_E$ | $C_T$ | $C_R$ | TC | Runtime (s) |
| 150 | 53 | 0 | 440 | 384 | 844 | 0,09 | 36 | 120 | 0 | 413 | 533 | 0,11 | 49 | 0 | 380 | 366 | 746 | 0,36 |
| 1000 | 47 | 0 | 340 | 366 | 706 | 0,65 | 36 | 120 | 0 | 413 | 533 | 0,58 | 47 | 0 | 340 | 386 | 726 | 0,71 |
| 50000 | 44 | 0 | 280 | 385 | 665 | 13,0 | 36 | 120 | 0 | 400 | 520 | 17,4 | 44 | 0 | 280 | 385 | 665 | 13,2 |
| 100000 | 36 | 0 | 120 | 401 | 521 | 34,3 | 36 | 120 | 0 | 400 | 520 | 30,0 | 36 | 0 | 120 | 401 | 521 | 28,1 |

The BFS Algorithm generates 104976 combinations for the five activity network. We tried to use a beam width between 150 and 100000. As we can see by the results exhibited in Table 4, the *Duration* evaluation type achieved quicker results similar to the ones obtained with the BFS Algorithm. The other evaluation types are far from the solution obtained with the BFS algorithm using lowest beam widths, but achieve better $C_R$ (project cost without bonus or penalty) values.

Now consider a ten activities network, using 5 different resources, three of them with 2 possible levels, one having 5 levels and the left one with 3 elective levels. Assume the following rates for earliness and lateness costs: , $\gamma_E = -15$, $\gamma_L = 20$ and the due date $T_S = 30$.

The BFS solution couldn't be achieved in a reasonable time.

Table 5: Ten activities network 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 (s) | $t_n$ | $C_E$ | $C_T$ | $C_R$ | TC | Runtime (s) | $t_n$ | $C_E$ | $C_T$ | $C_R$ | TC | Runtime (s) |
| 1000 | 29 | 15 | 0 | 406 | 391 | 6,35 | 26 | 60 | 0 | 468 | 408 | 7,39 | 45 | 0 | 300 | 444 | 744 | 4,47 |
| 5000 | 27 | 45 | 0 | 417 | 372 | 13,99 | 26 | 60 | 0 | 468 | 408 | 18,37 | 44 | 0 | 280 | 447 | 727 | 19,46 |
| 10000 | 27 | 45 | 0 | 417 | 372 | 30,53 | 24 | 90 | 0 | 491 | 401 | 42,31 | 44 | 0 | 280 | 440 | 720 | 30,29 |
| 30000 | 27 | 45 | 0 | 417 | 372 | 95,5 | 23 | 105 | 0 | 482 | 377 | 106,52 | 44 | 0 | 280 | 440 | 720 | 88,38 |
| 50000 | 27 | 45 | 0 | 405 | 360 | 175,53 | 23 | 105 | 0 | 480 | 375 | 2114,3 | 42 | 0 | 120 | 451 | 691 | 179,1 |

We observe a performance decrease in runtime values. The evaluation type *Cost* provides the best solutions, with a TC = 360 for a beam width of 50000. Using *Duration* we achieve reasonable solutions, on the other hand using the *Cost/Duration* evaluation type provide "weak" solutions.

For a twenty activities network, using the 4 resources with 3 different levels each, we have assumed the following rates for earliness and lateness costs: $\gamma_E = -10$, $\gamma_L = 20$ and the due date $T_S = 60$.

Table 6: Twenty activities network 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 (s) | $t_n$ | $C_E$ | $C_T$ | $C_R$ | TC | Runtime (s) | $t_n$ | $C_E$ | $C_T$ | $C_R$ | TC | Runtime (s) |
| 500 | 125 | 0 | 1300 | 864 | 2164 | 2,3 | 69 | 0 | 180 | 1609 | 1789 | 20,5 | 132 | 0 | 1440 | 879 | 2319 | 4,3 |
| 1000 | 115 | 0 | 1200 | 885 | 2005 | 5,1 | 69 | 0 | 180 | 1519 | 1699 | 31 | 132 | 0 | 1440 | 879 | 2319 | 10,0 |
| 2000 | 115 | 0 | 1200 | 885 | 2005 | 12,34 | 69 | 0 | 180 | 1519 | 1699 | 89,1 | 126 | 0 | 1320 | 868 | 2188 | 17,5 |
| 3000 | 114 | 0 | 1080 | 916 | 1996 | 21,78 | 69 | 0 | 180 | 1519 | 1699 | 60,2 | 131 | 0 | 1000 | 896 | 2316 | 22,4 |

The performance executing this network for large beam width was too slow. We present solutions for a beam width of 500 and 3000. Again the evaluation type Cost/Duration gave the weakest solutions, and the Duration evaluation type achieved the better ones (TC =1699, $t_n$ =69).

## CONCLUSIONS AND FURTHER RESEARCH

The experiments done for the specific networks have shown that the tool provides feasible solutions, although it doesn´t guarantee the optimum. Three evalution types are available for the beam search procedure. For the tests run so far, the better solutions are achieved using the *Cost* evaluation type or *Duration* evaluation type. The *Cost/Duration* evaluation might be discarded or remodeled. The performance of the evaluation type is influenced by the specifications of the project, like bonus/penalty costs and due dates. The machine where the tests were run is obsolete nowadays (in terms of processor and in terms of memory capacity). For larger beam widths and larger networks, the runtimes obtained are several minutes. For most projects we obtained at least 12 solutions (some equal), with reasonable total project costs (TC) and due dates ($t_n$).
The algorithm and the code implemented should be revised and studied, in order to introduce performance

improvements. The creation of networks for the experiments is not easy using the project creation wizard of the application, since it is necessary for the user to introduce all project data, including resources data and activities characteristics, like resources required and precedents. In the future it will be useful to have a method to generate partially (or completely) valid networks in an automatic way, and run the experiments on powerful machines. Some enhanced techniques in terms of software design can be considered to improve the program implemented.

## REFERENCES

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

Blazewicz, J., Lenstra, J.K. and Rinnooy Kan, A.H.G. 1983. "Scheduling subject to resource constraints: classification and complexity." *Discrete Applied Mathematics*, Vol. 5, No. 1, 11-24.

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

Dean, B.V., Denzler, D.R. and Watkins, J.J. 1992. "Multiproject staff scheduling with variable resource constraints". *IEEE Transactions on Engineering Management*, Vol. 39, No. 1, 59-72.

Demeelemeeeter, E. L., Herroelen, W. S. 1996. "An Efficient Optimal Solution Procedure for the Preemptive Resource-Constrained Scheduling Problem". *European Journal of Operational Research*. 90, 334-348.

Elmaghraby, S.E. and Herroelen, W.S. 1990. "The scheduling of activities to maximize the net present value of projects". *European Journal of Operational Research*, Vol. 49, No. 1, 35-49.

Gonçalves, J. F, Mendes J. J. M. and Resende M. G. C. 2004. "A Genetic Algorithm for the Resource Constrained Multi-Project Scheduling Problem". Technical Report TD-668LM4. AT&T Labs Research.

Guldemond, T., Hurink J., Paulus J., Schutten J. 2008. "Time-constrained project scheduling". *Journal of Scheduling*, Vol. 11, No. 2. ,137-148.

Heilmann R. 2001. "Resource–constrained project scheduling: a heuristic for the multi–mode case". *OR Spektrum* 23: 335–357.

Kis, T. 2005. "A branch-and-cut algorithm for scheduling of projects with variable-intensity activities". *Mathematical Programming*, , Vol. 103, No. 3, 515-539.

Mika, M., Waligora, G. and Weglarz, G. 2005. "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*, Vol. 164, No. 3, 639-668.

Özdamar, L. and Ulusoy, G. 1995. "A Survey on the Resource-Constrained Project Scheduling Problem". *IIE Transactions*, Vol. 27, No. 5, 574-586.

Patterson, J. H., Slowinski, R., Talbot, F.B., Weglarz, J. 1989. "An algorithm for a general class of precedence and resource constrained scheduling problems", In: Slowinski, R. and Weglarz, J., Editors, 1989. Advances in Project Scheduling, Elsevier, Amsterdam, pp. 3–28.

Patterson, J. H., Talbot, F.B., Slowinski, R., Weglarz, J. 1990. "Computational experience with a backtracking algorithm for solving a general class of precedence and resource constrained scheduling problems". *European Journal of Operational Research*, Vol. 49, N0. 1, 68-79.

Santos, M.A., Tereso A.P. 2010a. "On the Multi-Mode, Multi-Skill Resource Constraint Project Scheduling Problem (MRCPSP-MS)". In *Proceedings of the 2nd International Conference on Engineering Optimization* (*EngOpt 2010)*, Lisbon – Portugal, September 6-9.

Santos, M.A., Tereso, A.P. 2010b. "On the Multi-Mode, Multi-Skill Resource Constrained Project Scheduling Problem – A Software Application", In *Proceedings of the WSC15 - The 15th Online World Conference on Soft Computing in Industrial Applications* WWW, November 15-27.

Tereso, A. P., Araújo, M.M., Elmaghraby, S.E. 2004a. "Adaptive Resource Allocation in Multimodal Activity Networks". *International Journal of Production Economics*, Vol. 92, No. 1, 1-10.

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

Tereso, A. P., Mota, J. R., Lameiro, R. J. 2006. "Adaptive Resource Allocation Technique to Stochastic Multimodal Projects: a distributed platform implementation in JAVA", *Control and Cybernetics*, Vol. 35, No. 3661-686.

Tereso, A.P., Costa, L., Novais R., Araújo, M.M., 2007. "The Optimal Resource Allocation in Stochastic Activity Networks via the Evolutionary Approach: a platform implementation in Java". In *Proceedings of the International Conference on Industrial Engineering and Systems Management* (IESM 2007), Beijing – China, May 30 – Jun 2.

Tseng, Ching-Chih 2008. "Two Heuristic Algorithms for a Multi-Mode Resource-Constrained Multi-Project Scheduling Problem". *Journal of Science and Engineering Technology*, Vol. 4, No. 2, 63-74.

Willis, R.J. 1985. "Critical path analysis and resource constrained project scheduling - theory and practice". *European Journal of Operational Research*, Vol. 21, No. 2, 149-155.