University of Minho
Engineering School
Production and Systems Department

International Conference on
Project Economic Evaluation

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

Mónica A. Santos, Anabela P. Tereso

Production and Systems Department

Engineering School

University of Minho – Portugal

pg13713@alunos.uminho.pt

anabelat@dps.uminho.pt

Topics

- Introduction
  - Problem Description
- Solution Details
  - Procedure Adopted
  - Application Development
- Computational Results
- Conclusions

Introduction

This work is concerned with an extension of the Resource-Constrained Project Scheduling Problem (RCPSP) which belongs to the NP-hard class of problems.

More precisely it consists of the optimization scheduling problem with multi-level (or multi-mode) activities. The activities can be scheduled at different modes, each mode using a different resource level, implying different costs and durations.

A resource has a capacity of several units (w workers or m/c's) and may be used at different levels.

Problem Description

An activity normally requires the simultaneous utilization of more than one resource for its execution. 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.

Mathematical Model

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.

Mathematical Model

- $c_R(a)$: 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.

Mathematical Model

Briefly, the constraints of this problem are:

Respect the precedence among the activities. $t_j - t_i \geq p(a), \forall\, a \in A$

A unit of the resource is allocated to at most one activity at any time at a particular level.

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

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.

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

An activity can be started only when it is sequence-feasible and all the requisite resources are available, and must continue at the same levels of all the resources without interruption or preemption.

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

Mathematical Model

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:

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

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

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

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

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

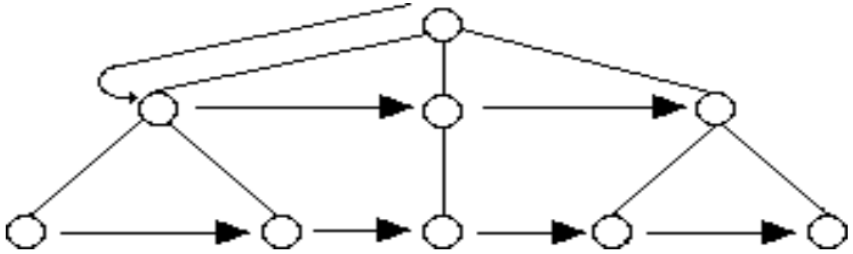
$$e \geq \mathrm{T}_s - t_n$$

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

$$e, d \geq 0$$

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.

Solution Details

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 better than the best known solution. If not, the node is discarded.

A “filtered beam” search is a heuristic BaB procedure that uses breadth first search but only the top “best” nodes are kept, depending on the *beam width*.

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”.

Procedure Description

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.

Procedure Description

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. For activities in the state "finished", the combination stores, for each required resource of the activity, the selected mode of execution.

Procedure Description

- 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.

Procedure Description

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. Activities in the state “to begin” are analyzed in order to check resources availability. If there are no resource conflicts, they are set to state “active” and resources are set as being used, otherwise we apply in sequence, the rules described above.

Procedure Description

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.

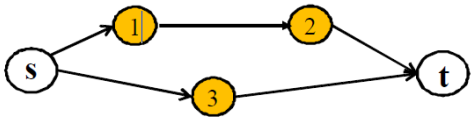
Procedure Description

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.

Procedure Description - Example



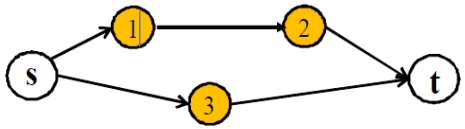
Project with 3 activities. AoN

$$\gamma_E = -10\,,\ \gamma_L = 20\,,\ T_S = 24.$$

Resource Requirements, Processing Times and Resource Costs of 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	n_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	-	-	-	-	-	-	

Procedure Description - Example



1. Activity A1: 18 combinations of resources levels (2*3*3).

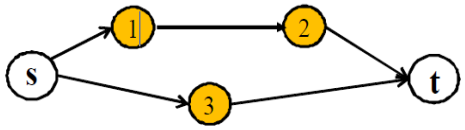
2. Activity A3: 6 combinations of resources levels (2*3).

3. The project initiates with activity A1 and A3 in state “active”, with 108 branches to search solutions if using de BFS procedure, otherwise if using the beam search procedure, these branches are evaluated and only the top bests are kept. The number of branches to keep is defined by the beam width (p.e. 50).

Procedure Description - Example

4. Activity 2 initiates in "pending" state. It can only begin when activities A1 and A3 have finished. When activity 2 is ready to begin, and is set to the state "active", generates 9 possible combinations of its resources levels (3*3).

5. These activity combinations will be joining existent project combinations, obtaining finally 972 valid solutions for the project (108*9) using BFS, or 450 valid solutions (50*9) if using BS.



Procedure Description - Example

Activity 1:

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

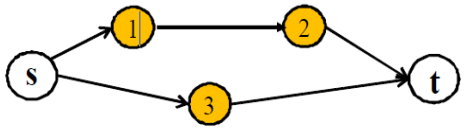
Activity 3:

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

Activity 2:

$x(2,2,1)$	$x(2,2,2)$	$x(2,2,3)$	$c(2,2,3)$	$p(2,2,3)$	$x(2,4,1)$	$x(2,4,2)$	$x(2,4,3)$	$c(2,4,3)$	$p(2,4,3)$	$C_R(2)$	$p(2)$
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

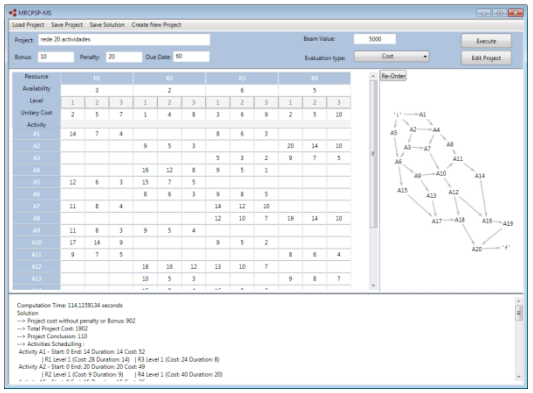


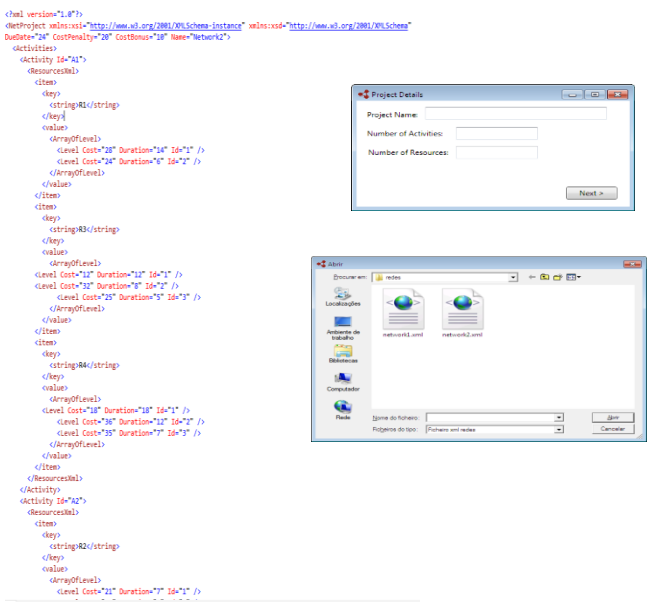
$\gamma_E = -10$, $\gamma_L = 20$, $T_S = 24$

Application Development

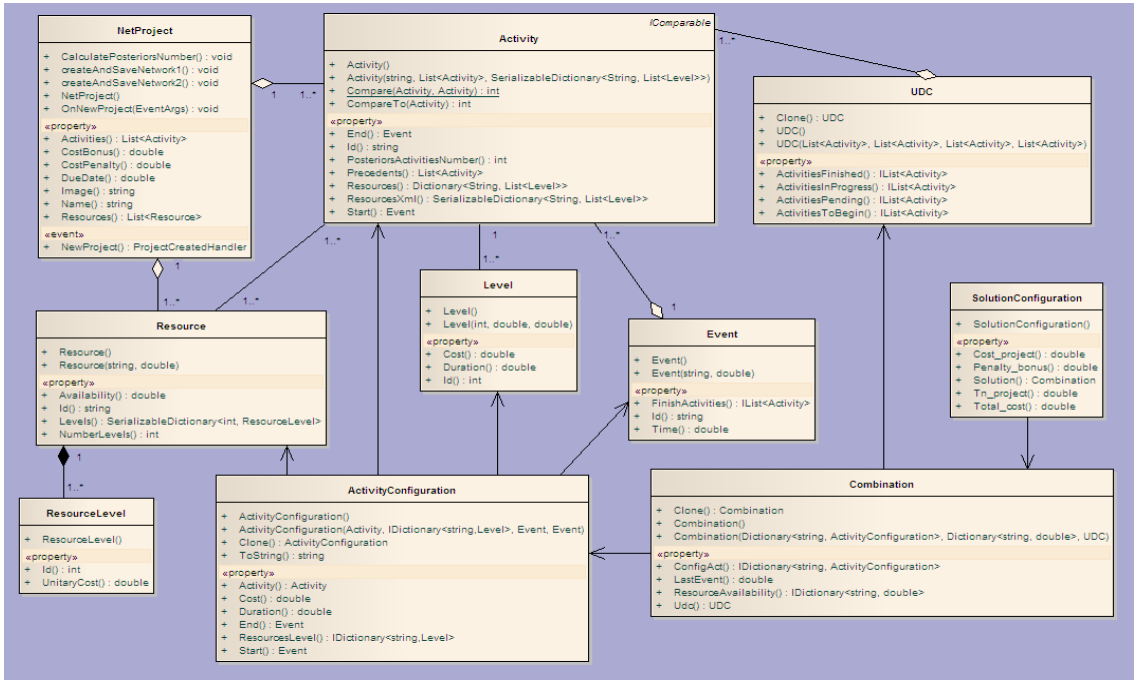
C# language - Visual Studio 2010.

To construct the project network (in AoN), we used Graph#.





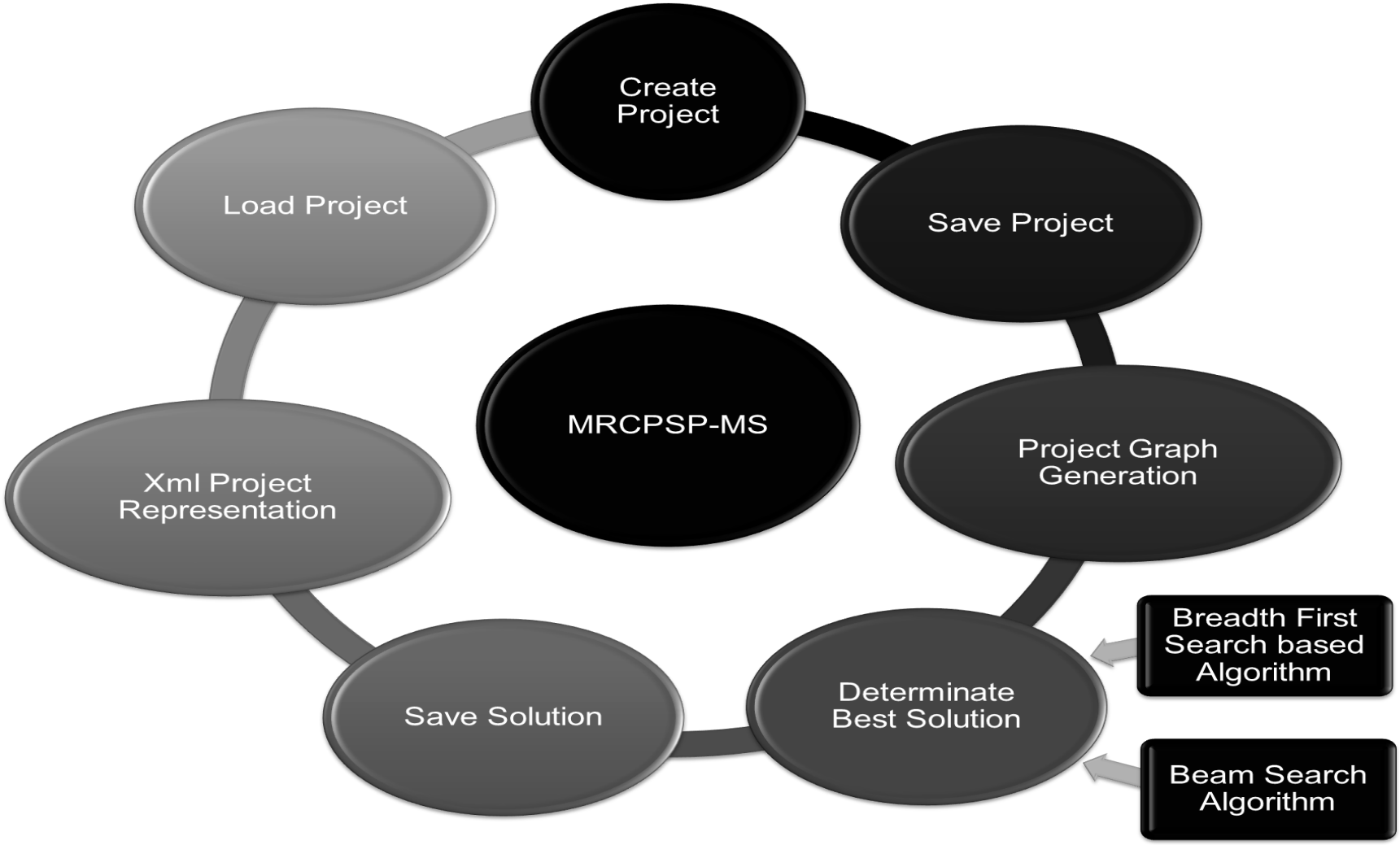
Data Model



Main Classes:

- *NetProject* keeps all project required information: name, activities, resources, due date, bonus and penalty cost.
- *Resource class* keeps the resource identification availability and levels.
- *Activity class* has activity identification, resources requirement and its precedents.
- Additional classes are used to support the evaluation of the project solution.

Functionalities



Computational results

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

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

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

972 combinations

Five Activities Network

Consider a five activities network, using the same resources of the three activities network above and with $\gamma_E = -10$, $\gamma_L = 20$, $T_S = 30$.

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

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

104976 combinations

Ten Activities Network

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.

$\gamma_E = -15$ $\gamma_L = 20$ $T_S = 30$

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

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)
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

Twenty Activities Network

Finally we tested a twenty activities network, using the 4 resources with 3 different levels each with: $\gamma_E = -10$ $\gamma_L = 20$ $T_S = 60$

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)
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

Conclusions

The experiments done for the specific networks have shown that the tool provides feasible solutions, although it doesn´t guarantee the optimum.

Three evaluation 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 the 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.

Conclusions

The algorithm and the code implemented can still be revised, 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.

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 (like parallel and distributed computing).

References

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.