International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:6, 2007

Flexible Heuristics for Project Scheduling with Limited Resources

Miloš Šeda

***Abstract*—**Resource-constrained project scheduling is an NP-hard optimisation problem. There are many different heuristic strategies how to shift activities in time when resource requirements exceed their available amounts. These strategies are frequently based on priorities of activities. In this paper, we assume that a suitable heuristic has been chosen to decide which activities should be performed immediately and which should be postponed and investigate the resource-constrained project scheduling problem (RCPSP) from the implementation point of view. We propose an efficient routine that, instead of shifting the activities, extends their duration. It makes it possible to break down their duration into active and sleeping subintervals. Then we can apply the classical Critical Path Method that needs only polynomial running time. This algorithm can simply be adapted for multiproject scheduling with limited resources.

***Keywords*—**Project management, resource-constrained scheduling, NP-hard problem, CPM, heuristic method.

I. Introduction

THE scheduling problem is a frequent task in the control of various systems such as manufacturing processes [3], project management [7], and service system control (reservation systems, timetabling).

A classical task of project management is to create the network graph of a project and, on the basis of knowledge or an estimation of time of activities, to determine critical activities that would have influenced a project delay. Each activity draws on certain resources, e.g. financial resources, natural resources (energy, water, material, etc.), labour, managerial skills. The solution of this task is well known in the case when we do not take limited resources into consideration.

However, in real situations, available capacities of resources are constrained to certain limits. Project planning under limited resources [3,7] is difficult because of

- interdependence among activities due to sharing the same resources,
- dependence of resource consumption on the manner in which the activity is subdivided and
- dependence on the limits of availability of the various resources

The character of all the three dependencies is basically non-linear. In general, scheduling problems are *NP-hard*, consequently no polynomial- time algorithms are known to guarantee an optimal solution.

Classical methods of network analysis offer some approaches which depend on whether activities that have a lack of resource(s) may be suspended or not and whether activities will be put off until the moment when it is possible to perform them. A question is: Which concurrent activities should be suspended or put off because of a lack of resource(s)?

An excellent survey [5] with 203 references presents classification, models and methods for solving the *resource-constrained project scheduling problem* (RCPSP) and, in [8], it is critically reviewed and completed. There are single-mode and multi-mode variants of the problem, resources can be non-renewable or renewable, activities of project can be interrupted or not and so on.

With respect to NP-hardness of the RCPSP, mainly heuristic methods are used for its solving. Papers frequently present applications of stochastic heuristics such as simulated annealing [4,12], genetic algorithms [2,9,10], tabu-search [1,12,14], ant systems [13] and swarm optimization method [15].

In this paper we assume that, when resources are not sufficient, a suitable method for shifting one or more activities has been selected and we touch another problem. In real situations, the durations of activities can only be estimated and are subject to change. If we included this new information in our considerations, this could result in quite a different optimal schedule. Since, obviously, the dates of activities in the past cannot be changed, it is necessary for the calculation of the entire project with new data to modify the dates of only those activities that have not yet been finished.

II. Basic Notions

In this paragraph we introduce the notation used in this paper and the basic concepts of CPM. We consider a network graph G with n *topologically ordered* vertices [6], which means that, for each edge (i,j), i appears before j $(i < j)$ and the starting vertex has number n_0=1 and ending vertex number

Manuscript received July 20, 2007. This work was supported in part by the Ministry of Education, Youth and Sports of the Czech Republic under research plan MSM 0021630518 "Simulation Modelling of Mechatronic Systems".

Miloš Šeda works in the Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology, Technická 2896/2, CZ 616 69 Brno, Czech Republic (phone: +420-54114 3332; fax: +420-54114 2330; e-mail: seda@fme.vutbr.cz).

n. This ordering can be gained as follows:

1. Start from the origin and assign n_0 to it.
2. Leave all edges outgoing from n_0 and assign numbers $n_0+1, \dots, n_0+k_1$ to k_1 vertices that have no input edge.
3. Leave all edges outgoing from the vertices numbered in the previous step and assign numbers $n_0+k_1+1, \dots, n_0+k_1+k_2$ to k_2 vertices that have no input edge.
4. Continue this way until all vertices are numbered.

Assume that edges represent activities of a project and vertices correspond to beginnings or ends of activities. Denote $E(G)$ the set of edges of a graph G, and $V(G)$ its set of vertices. If $e = (i, j)$ is an edge in $E(G)$, then we denote its duration by t_{ij} or $t(i,j)$, or t_e in short. Similarly, requirements of activity (i,j) for a resource r will be denoted by r_{ij} etc.

The following notions refer to start and end vertices of the network graph: $T_i^{(0)}$ represents the earliest possible start time of vertex i, $T_j^{(1)}$ the latest allowable finish time of vertex j and $T_S(i,j) = T_j^{(1)} - T_i^{(0)} - t_{ij}$ is the *total slack of activity* (i,j) or *total* (*activity*) *float* (the amount of time by which the start of a given activity can be delayed without delaying the completion of the project).

Finally, assume that $V_i(e)$ denotes the starting vertex of an edge $e=(i,j)$ and $V_j(e)$ is its ending vertex.

Further notation will be introduced when needed.

III. Algorithm

The algorithm is based on time shifting of activities when their total requirements are higher than the resource limit. This is implemented by prolonging their duration but distinguishing, for each activity, its starting duration and current duration which equals the length of shift and starting duration. The greatest advantage of this access is that whenever we need to compute new earliest possible start times and latest allowable finish times for activities after shifts or update the actual time duration of some activities, we can compute the whole project using a simple CPM method and, in spite of this, the dates of finished activities remain unchanged in the result of the new calculation. In other words, any change in the present has no effect on results in the past.

Let us denote

t_{ij}^s …	starting duration of activity (i,j)
t_{ij}^c …	current duration of activity (i,j)
$\delta_{ij} = t_{ij}^c - t_{ij}^s$ …	interval when activity has no requirements ("sleeps")

Now we will formulate an algorithm. The symbol := stands for the assignment operator.

1. [*Initialization*]

Using CPM method, we determine for each edge the earliest possible start time and the latest allowable finish time. Let us assign

$$\tau_1 := 0, \tag{1}$$

$$\delta_{ij} := 0,\ t_{ij}^c := t_{ij}^s \text{ for every } (i,j) \in E(G) \tag{2}$$

2. [*Test for finishing*]

If $\tau_1 = T_n^{(0)}$ then algorithm finishes else we continue to step 3.

3. [*Determination of interval* $[\tau_1, \tau_2]$]

The left bound is given and the right bound we determine from the following formula

$$\tau_2 = \min_{(i,j)\in E(G)} \left(\left\{ T_i^{(0)} \mid T_i^{(0)} > \tau_1 \right\} \cup \left\{ T_i^{(0)} + t_{ij}^c \right\} \right) \tag{3}$$

4. [*Determination of activities requiring resource in* $[\tau_1, \tau_2]$]

Let

$$A = \left\{ (i,j) \in E(G) \mid [\tau_1, \tau_2] \subseteq [T_i^{(0)} + \delta_{ij}, T_i^{(0)} + t_{ij}^c] \right\} \tag{4}$$

Let us determine the total requirements q of activities from A.

$$q = \sum_{(i,j)\in A} r_{ij} \tag{5}$$

If $q >$ resource limit, then we continue to step 5 else to step 7.

5. [*Ordering activities from A by their priorities*]

Let $A = \{e_1, \dots, e_m\}$. We order all m activities in A into a non-ascending sequence B as follows. If b_k, b_l are any elements from B, then the order relation is defined as follows:

$$\begin{aligned} b_k \succ b_l \Leftrightarrow\ & T_{V_i(b_k)}^{(0)} + \delta_{b_k} < \tau_1 \\ & \text{else if } T_S(b_k) < T_S(b_l) \\ & \quad \text{then if } T_S(b_k) = T_S(b_l) \\ & \quad\quad \text{else if } r_{b_k} > r_{b_l} \end{aligned} \tag{6}$$

The first condition on the right-hand side means that b_k has begun in the previous interval.

6. [*Right-shifting*]

Because of step 4, there exists $j < m$ such that

$$\sum_{i=1}^{j} r_{b_i} \le \text{limit and } \sum_{i=1}^{j+1} r_{b_i} > \text{limit} \tag{7}$$

We shift activity b_{j+1} so that new values of its parameters are obtained from the old values by the following assignments:

$$\begin{aligned} t_{b_{j+1}}^c &:= t_{b_{j+1}}^c + \tau_2 + \left(T_{V_i(b_{j+1})}^{(0)} + \delta_{b_{j+1}} \right), \\ \delta_{b_{j+1}} &:= t_{b_{j+1}}^c - t_{b_{j+1}}^s \end{aligned} \tag{8}$$

for the other activities in B (according to their priorities), we either add their resource requirements and place them into the schedule or shift them if limit has been exceeded. Finally, we apply the CPM method again.

7. [*Next interval*]

$$\tau_1 := \tau_2 \tag{9}$$

and return to step 2

IV. Example

Let us have a network graph described by the following table including also the activity requirements for one resource and the destination of project parameters obtained by the first use of CPM.

In the example, the starting time of the project equals 0 and all times are relative to this start. In practice, the start of the project is given in the form of real dates and times.

TABLE I
PROJECT WITH LIMITED RESOURCE *r*

i	j	t_{ij}	r_{ij}	$T_i^{(0)}$	$T_j^{(1)}$	$T_S(i,j)$
1	2	2	6	0	2	0
1	3	4	3	0	7	3
1	4	5	5	0	6	1
2	4	4	4	2	6	0
2	5	3	7	2	12	7
3	6	7	4	4	14	3
4	5	6	5	6	12	0
4	6	4	3	6	14	4
5	6	2	5	12	14	0

The following figures show the results obtained by computation. In Fig. 1 we see the result of the first application of CPM and the interval in which resource requirements exceed the limit. The currently investigated interval is marked by two vertical lines.

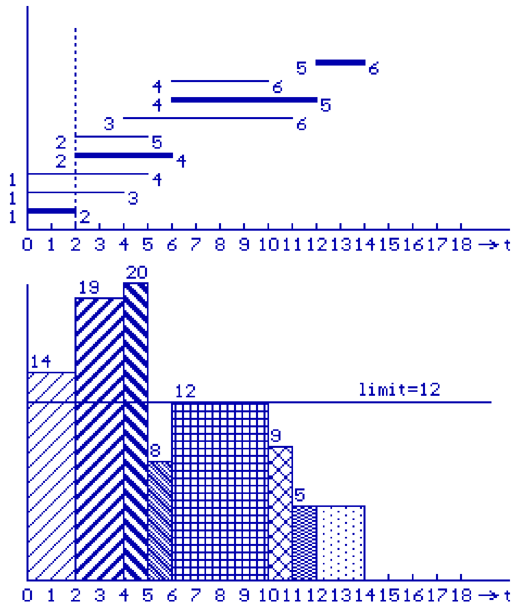

Fig. 1 Initial state

In Fig. 1 we can see that three activities that should run in time interval [0, 2] are lacking resource *r* and thus the requirements of at least one of them cannot be satisfied and its start must be postponed to the next interval. This interval is defined by the nearest two times representing the earliest possible start time or the latest admissible finish time of some of the activities. The priority of activities in terms of its resource requirements being satisfied is determined by (6) in step 5 of the algorithm.

After several iterations we will get the result as shown in Fig. 2. The project duration was prolonged from 14 to 17 time units. The horizontal dotted line in the Gantt chart represents the "sleeping" part of activities (= interval with no resource requirements) that corresponds to the length of their shift. Critical activities are highlighted by a bold line. As the sleeping parts of activities do not require any resources, any shift and shortening of the beginning of a sleeping part has no effect on the length of the project and cannot be included in the critical path.

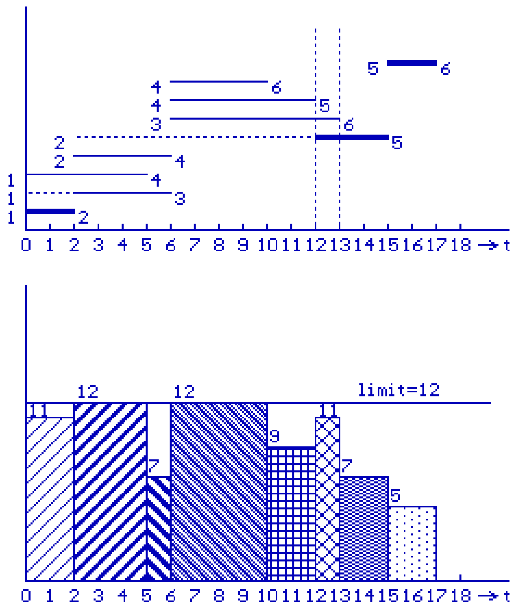

Fig. 2 Final schedule

V. Generalisation for Multiproject Schedule

The algorithm described in the previous section can be very simply adapted for multiproject scheduling with limited resources. The steps of this generalised algorithm must take into account that resources are shared by all the projects. If the number of projects is *N*, then, e.g., Step 3 can be modified as follows

$$\tau_2 = \min_{\substack{(i,j)\in E(G_k)\\ k\in[1,N]}} \left(\left\{ T_i^{(0)} \mid T_i^{(0)} > \tau_1 \right\} \cup \left\{ T_i^{(0)} + t_{ij}^{c} \right\} \right) \qquad (3')$$

The other steps may be adapted in a similar way.

VI. Possible Improvements

If we consider a schedule as a sequence of time intervals where neighbouring intervals based on the time an activity starts or finishes, the scheduling problem may be reduced to a set of the optimal choices of activities in intervals by the order relation (6).

In the network analysis literature, we may find many other strategies describing how the ranking of competing activities is accomplished, for instance:

- *greatest remaining resource demand*, which schedules as first those activities with the greatest amount of work remaining to be completed,
- *least total float*, which schedules as first those activities possessing the least total float,
- *shortest imminent activity*, which schedules as first those activities requiring the least time to complete,
- *greatest resource demand*, which schedules as first those activities requiring the greatest quantity of resources from the outset.

Using defined priorities as prices, the task investigated in one interval corresponds to the well-known *knapsack problem*. If the number of the activities sharing a source in an interval is low, e.g. up to 20, and this is satisfied in most of real situations, then this problem can be solved exactly by a branch and bound method or by dynamic programming.

Assuming only one resource with a limited capacity, we can deal with the 0-1 *knapsack problem* (0-1 KP), which is defined as follows: A set of n items is available to be packed into a knapsack with capacity of C units. Item i has value v_i and uses up to w_i of capacity. We try to maximise the total value of packed items subject to capacity constraint.

$$\max\left\{\sum_{i=1}^{n} v_i x_i \mid \sum_{i=1}^{n} w_i x_i \le C, x_i \in \{0,1\}, i=1,\dots,n\right\} \tag{10}$$

Binary decision variables x_i specify whether or not item i is included in the knapsack.

The simplest way is to generate all possible choices of items and to determine the optimal solution among them. This strategy is of course not effective because its time complexity is $O(2^n)$.

Finding the solution may be faster using the *branch and bound method* [11], which restricts the growth of the search tree. Avoiding much enumeration depends on the precise upper bounds (the lower the upper bounds, the faster the finding of the solution is). Let items be numbered so that

$$\frac{v_1}{w_1} \ge \frac{v_2}{w_2} \ge \dots \ge \frac{v_n}{w_n} \tag{11}$$

We place items in the knapsack by this non-increasing sequence. Let $x_1, x_2, \dots, x_p$ be fixed values of 0 or 1 and

$$M_k = \left\{\mathbf{x} \mid \mathbf{x} \in M, x_j = \xi_j, \xi_j \in \{0,1\}, j=1,\dots,p\right\} \tag{12}$$

where M is a set of feasible solutions. If

$$(\exists q)(p < q \le n): \sum_{j=p+1}^{q-1} w_j \le C - \sum_{j=1}^{p} w_j \xi_j < \sum_{j=p+1}^{q} w_j \tag{13}$$

then the upper bound for M_k can be determined as follows:

$$U_B(M_k) = \sum_{j=1}^{p} v_j \xi_j + \sum_{j=p+1}^{q-1} v_j + \frac{v_q}{w_q}\left(C - \sum_{j=1}^{p} w_j \xi_j - \sum_{j=p+1}^{q-1} w_j\right) \tag{14}$$

In more complex and more frequent situations when we have more than one limited resource, we transform the resource-constrained scheduling into a sequence of *multi-knapsack problem* (MKP) solutions. MKP is defined as follows:

$$\begin{aligned} &\text{maximise} && \sum_{i=1}^{n} v_i x_i \\ &\text{subject to} && \sum_{i=1}^{n} w_{ri} x_i \le C_r, r=1,\dots,m, \\ & && x_i \in \{0,1\}, i=1,\dots,n \end{aligned} \tag{15}$$

For the solution of the task with multiple constraints, we must generalise the approaches mentioned above.

The combinatorial approach can be applied without any changes, but using the branch and bound method, we must redefine the upper bound. For its evaluation we use the following formula

$$U_B(M_k) = \min\left\{U_B^1(M_k), \dots, U_B^m(M_k)\right\} \tag{16}$$

where auxiliary bounds $U_B^i(M_k), i=1,\dots,m$ correspond to the given constraints and are determined as in the 0-1 knapsack problem. Before evaluation of these auxiliary bounds, the other variables must be sorted again by decreasing values v_j/w_{ij}. Evidently, the run time will increase substantially.

VII. Conclusion

In the paper, a new implementation of the computing of the resource-constrained project scheduling was proposed. The strategy of activity-shifting was replaced by prolonging their duration and dividing them into active and sleeping parts. It makes it possible to apply a simple CPM algorithm. The proposed algorithm was designed in a mathematical form and verified for a single version of RCPSP.

We also sketched how to adapt the proposed algorithm for multiproject scheduling with limited resources.

Finally, we discussed possible improvements based on more flexible ways choosing activities for intervals where resource requirements exceed their available amounts.

Further investigation will include fuzzy versions of the problem and the case of a multi-project scheduling problem.

References

[1] C. Artigues, P. Michelon and S. Reusser, “Insertion Techniques for Static and Dynamic Resource-Constrained Project Scheduling,” *European Journal of Operational Research*, vol. 149, pp. 249-267, 2003.

[2] A. Azaron, C. Perkgoz and M. Sakawa, "A Genetic Algorithm for the Time-Cost Trade-off in PERT Networks," *Applied Mathematics and Computation*, vol. 168, pp. 1317-1339, 2005.

[3] J. Blazewicz, K.H. Ecker, G. Schmidt and J. Weglarz, *Scheduling Computer and Manufacturing Processes*. Berlin: Springer-Verlag, 1996.

[4] K. Bouleimen and H. Lecocq, "A new Efficient Simulated Annealing Algorithm for the Resource-Constrained Project Scheduling Problem and Its Multiple Mode Version," *European Journal of Operational Research*, vol. 149, pp. 268-281, 2003.

[5] P. Brucker, A. Drexl, R. Möhring, K. Neumann and E. Pesch, "Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods," *European Journal of Operational Research*, vol. 112, pp. 3-41, 1999.

[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, *Introduction to Algorithms*. Cambridge, Massachusetts: MIT Press, 2001.

[7] S.E. Elmaghraby, *Activity Networks: Project Planning and Control by Network Models*. New York: John Wiley & Sons, 1977.

[8] W. Herroelen, E. Demeulemeester and B.D. Reyck, "A Note on the Paper "Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods" by Brucker et al," *European Journal of Operational Research*, vol. 128, pp. 679-688, 2001.

[9] K.W. Kim, M. Gen and G. Yamazaki, "Hybrid Genetic Algorithm with Fuzzy Logic for Resource-Constrained Project Scheduling," *Applied Soft Computing*, vol. 2/3F, pp. 174-188, 2003.

[10] K.W. Kim, Y.S. Yun, J.M. Yoon, M. Gen and G. Yamazaki, "Hybrid Genetic Algorithm with Adaptive Abilities for Resource-Constrained Multiple Project Scheduling," *Computers in Industry*, vol. 56, pp. 143-160, 2005.

[11] J. Klapka, J. Dvořák and P. Popela. *Methods of Operational Research* (in Czech). Brno: VUTIUM, 2001.

[12] M. Mika, G. Waligóra and J. 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*, vol. 164, pp. 639-668, 2005.

[13] L.-Y. Tseng and S.-C. Chen, "A Hybrid Metaheuristic for the Resource-Constrained Project Scheduling Problem," *European Journal of Operational Research*, 2005, article in press.

[14] V. Valls, S. Quintanilla and F. Ballestin, "Resource-Constrained Project Scheduling: A Critical Activity Reordering Heuristic," *European Journal of Operational Research*, vol. 149, pp. 282-301, 2003.

[15] H. Zhang, X. Li, H. Li and F. Huang, "Particle Swarm Optimization-Based Schemes for Resource-Constrained Project Scheduling," *Automation in Construction*, vol. 14, pp. 393-404, 2005.