Experimental Results of an Adaptive Resource Allocation Technique to Stochastic Multimodal Projects

Anabela P. Tereso
anabelat@dps.uminho.pt
M. Madalena T. Araújo
mmaraujo@dps.uminho.pt
Universidade do Minho 4800-058 Guimarães PORTUGAL
Salah E. Elmaghraby
elmaghra@eos.ncsu.edu
North Carolina State University
Raleigh, NC 27695-7906 USA

Abstract. We describe the structure and the implementation aspects of the dynamic programming procedure that was proposed in a previous report (Tereso *et al.*, 2001) for the optimal resource allocation to activities under the assumption of stochastic work content. We illustrate its implementation to five projects of varying size, which exhibit the computational complexity of the procedure. The pseudo-code may be obtained by request from the lead author.

Key Words: Operations Planning and Scheduling, Activity Networks, Resource Allocation, Dynamic Programming

1 Introduction and Background

This paper reports on experimental results on an approach for the resolution of the problem of adaptive resource allocation in stochastic multimodal project networks. The approach is based on a dynamic programming (DP) model introduced in a previous paper by the same authors (Tereso *et al.*, 2001).

The problem tackled can be briefly stated as follows. We are given a multimodal activity network[1] in which the work content of each activity is defined as a random variable (*r.v.*) of known distribution which may differ for different activities. The duration of the activity depends on its work content and the amount of resource allocated to it (we assume a single resource that is demanded by all the activities). It is desired to determine the optimal resource allocation to the activities of the project, so that the total expected cost is minimized. This cost is composed of two parts: the cost of resource utilization, assumed quadratic in the resource intensity, and a penalty for tardiness, assumed linear in the tardiness[2]. To the best of our knowledge this problem has not been treated before. Contributions to the classical 'resource constrained project scheduling problem' (RCPSP) and its variants are numerous; the interested reader may wish to consult the two most recent books on the subject by Demeulemeester and Herroelen (2002) and Neumann *et al.* (2002), and the references cited therein to gain a complete picture of developments in that aspect of project scheduling.

[1] That is, each activity can be performed in any number of levels of resource intensity applied to it, with resulting shorter or longer duration.
[2] Naturally, this presumes the prior definition of a due date for the completion of the project.

For simplicity of exposition and ease of computing we assume that the work content of each activity is a continuous *r.v.* that is exponentially distributed. This has the salutary effect of rendering the residual distribution of work content invariant with the passage of time of undertaking the activity, thus avoiding the complications surrounding the determination of such distributions which are probabilistic arguments that have little to do with the focus of the optimization procedure of interest to us here. We also assume that the resource is a continuous variable that may be allocated in any intensity within an interval between lower and upper bounds [l, u]. The availability of the resource is abundant so that it does not impose any limitation on the number of activities that are in progress simultaneously.

The DP model will be introduced using a simple example with only three activities as depicted in the network of figure 1 (activity-on-arc representation).

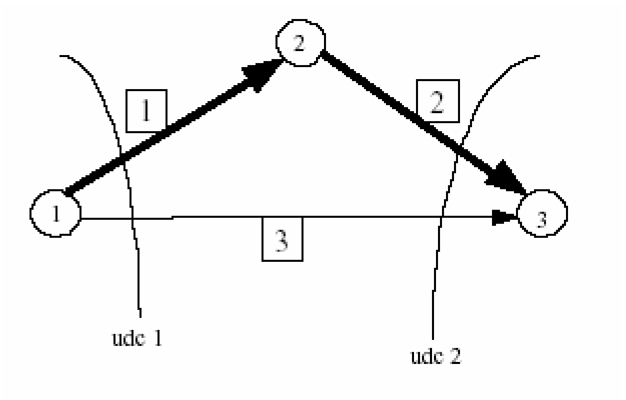

Figure 1: Example network with its uniformly directed cutsets.

In this example the due date of the project is $T = 14$; and the unit cost per period tardy is $c_L = 2$. The resource allocation to each activity is denoted by x_i for i = 1, 2, 3; with lower limit $l_i = 0.5$ and upper limit $u_i = 1.5$ for all *i*. The x_i*'s* are the *decision variables* of this problem. The parameters $\{\lambda_i\}$ of the distributions of the work content of the activities are as shown in (1).

Activity *i* :	**1**	**2**	**3**
λ_i :	0.2	0.1	0.07

(1)

It is clear that at any point in time the manager must cope with a subset of activities that lie on a uniformly directed cutset (*u.d.c.*). In this simple example there are only two *u.d.c.*'s: $C_1 = \{1, 3\}$ and $C_2 = \{2, 3\}$. To be sure, at the outset the project manager is concerned with activities 1 and 3, which lie on C_1. Then, depending on the progress in these two activities, he may eventually be concerned with activities 2 and 3, which lie on C_2. If the resource allocation to activity 3 is (temporarily) fixed at, say, $\hat{x}_3$, the problem reduces to the optimal determination of the resource allocation to activities 1 and 2, which can be readily resolved by standard DP recursion. The set of "fixed" activities is denoted by F; in this example $F = \{3\}$: Finally, searching over the values of *x3* with repeated optimization at each value would yield the (unconditional) optimum allocation to all three activities. In general, our procedure determines the *u.d.c.*'s of the network (which define the stages of the DP iterative scheme), and the cutset intersection index (*c.i.i.*), which represents the variables to be (temporarily) fixed; see Tereso *et al.* (2001) for details.

We continue with our illustrative example without much detail for the sake of economy in space. The expected resource cost of the fixed variables is denoted by *rcf*, which in this case is the expected cost of activity 3

$$rcf = \hat{x}_3 \cdot \varepsilon(W_3) = \frac{\hat{x}_3}{0.07} , \tag{2}$$

where W_3 is the work content of activity 3, $\mathcal{E}$ (W_3) denotes its expected value, and $\hat{x}_3$ the amount of resource allocated to it. Reverse numbering of the DP stages yields

$$f_1(t_2 \mid \mathsf{F} = \{3\}) = rcf + \min_{x_2 \in [0.5,1.5]} \mathcal{E}\{x_2 w_2 + 2\mathcal{E}(U)\} \tag{3}$$

where

$$\mathrm{U} = \max\ \{0, \mathbf{Y_3} - \mathrm{T}\}, \tag{4}$$

and

$$\mathbf{Y_3} = \max\ \{t_2 + \frac{W_2}{x_2}, \frac{W_3}{\hat{x}_3}\}. \tag{5}$$

The second and last stage would be defined as follows:

$$f_2(t_1 = 0 \mid \mathsf{F} = \{3\}) = \min_{x_1 \in [0.5,1.5]} \mathcal{E}\{x_1 W_1 + \mathcal{E}[f_2(Y_2)]\} \tag{6}$$

where

$$\mathbf{Y}_2 = \frac{W_1}{x_1}. \tag{7}$$

The solution for this network, obtained in 4.8 seconds[3], is:

$$\{x_1^*, x_3^*\} = \{1.0, 1.0\} \tag{8}$$

with an expected cost of 45.53.

The optimal value of x_2 depends on the state of node 2, when it is reached, and can be obtained by the previously developed optimal policy for stage 1, as defined in equation (3).

It is evident that the Achilles heel of this DP approach resides in the definition of the state space of the DP recursion and in the search over the "fixed" variables to determine the unconditional optimum. Our pursuit for efficient ways to reduce these two aspects of the optimizing procedure to manageable dimensions is the *raison d'étre* of this research.

2 The Application Development

This model was implemented in Matlab. The pseudo-code can be accessed on the internet[4], or upon request by e-mail[5]. In this section, we discuss the main issues that arose during the development of this application.

[3] The computer used to do the experimentations was a Pentium III, 650MHz, 128 MB.

2.1 Data Structure and Input Parameters

To allow the introduction of any network of arbitrary nodes and activities, we had to have a format to store all the information of the network. This was accomplished by defining a structure (*Net*) where each column represents an activity. This structure contains five fields which represent for each activity, respectively, the origin node, the terminal node, the parameter λ (which can be replaced by any set of parameters required to specify the probability distribution if it is different from the exponential), the lower bound, and the upper bound on the resource allocation.

The main program of this application is *Dp*. This program begins with calling the *InputNetwork* procedure, which allows the user to input the data for the project. These are: (*i*) the number of activities of the network (*n*), (*ii*) the data that defines the activities. The program automatically checks for correctness of the number of activities and for the uniqueness of an activity between any two nodes; and (*iii*) the values of c_L and *T*.

2.2 The Stages of the Dynamic Program

The number of stages of the DP iterations is the same as the number of decision variables. This is determined by evaluating the longest path (by the number of activities) in the network. The variables along this path define the set D of decision variables, and the complementary set of activities defines the set F of "fixed" variables. To accomplish this, we constructed two procedures. The first is *DecisionVars*, which determines the decision variables by determining the longest path in the network. The second is the function *SumNodesLen* which evaluates the maximal length of paths to reach a given node from the project start node. Next, another function is called that defines the nodes that belong to the longest path (*NodesLP*) which, in turn, defines the stages of the DP iterations. This process results, for the example in figure 1, in the following set of "nodes on the longest path": *NLP* = [1, 2, 3]. Finally, the decision variables are defined by the "activities on the longest path". The final result for the example in figure 1 is *ALP* = {1, 2}.

2.3 The Discretization of the Work Content

For the sake of computational feasibility, it is necessary to discretize each *W*, using k_2 values (we have opted for $k_2 = 4$ values for all the activities), all of equal probability. The function that generates these values is called *GenerateW*. The function receives the parameter λ of the exponential distribution, and returns *W*, an array with 4 values. The mathematical formulation that evaluates these four values is presented in the Appendix, and reflects the simplicity of the exponential distribution.

2.4 Determination of Bounds on Node Realization Times

The evaluation of the minimal and maximal duration of each activity is accomplished using the procedure *Durations*. Then it is possible to evaluate the bounds on the realization times of the network nodes, using procedures *GenerateTlimits* and *GenerateTvalues*. The array *Tlim* contains the

[4] www.eng.uminho.pt/~dps/anabelat (Topic: research).
[5] anabelat@dps.uminho.pt

limits of the interval of the node realization times, and the array *Tval* stores, in one field, the discretized list of the times of realization of the nodes. In another field, the "step" used between them will also be stored.

Arrays *Tlim(1)* and *Tlim(2)* store the lower and upper limits on the node realization times, respectively. The minimal duration of an activity is given by $Y_{\min} = \frac{w_{\min}}{x_{\max}}$, and the maximal duration is $Y_{\max} = \frac{w_{\max}}{x_{\min}}$. The *Tlim's* are generated in the following way. For node 1, *Tlim* = 0. For the following nodes, taken in topological order, their origins are determined, and the following formula is applied (in which the notation $i \,\pi\, j$ indicates node i immediately precedes node j):

$$T_{\lim j}(1) = \max_{i\pi j}\{T_{\lim i}(1) + Y(i,j)_{\min}\}; \qquad j = 2,\ldots,m. \tag{9}$$

$$T_{\lim j}(2) = \max_{i\pi j}\{T_{\lim i}(2) + Y(i,j)_{\max}\}; \quad j = 2,\ldots,m. \tag{10}$$

Next, it is necessary to discretize the nodes realization times. For that purpose, a function *CalculateTvalues* was constructed which, with the interval limits, generates a set of values and the interval step between them. This is done for all the nodes, except the first and the last.

2.5 The DP Iterations

The code for implementing the DP iterations depends on the network topology and the derived number of stages. If the network has *N*-1 stages[6] and *K* fixed variables, we need a main program with *K* nested cycles, one for each fixed variable. Inside each cycle we need to call *N*-1 different procedures, one for each stage. The content of these procedures depends on the topology of the network. This necessitates that one composes the code dynamically after the inputting of the network parameters. We briefly describe the approach used to accomplish this task.

After determining the *ALP* (the activities on the longest path), we know how many stages there shall be in the DP iterations. Initially, we generate the main program, using the procedure *generateMainCode*. The instructions are then composed in the form of strings, with a fixed and a variable part, namely the part for the cycles needed and the calling of the procedures that will optimize each stage of the DP model.

The next step is to generate the file *dps1*, which is the procedure that evaluates the contribution for stage 1 (the last stage in the network, which is the first stage in the DP iteration as we iterate backwards). This procedure is different from the others (*dpsn* for *n* = 2*, ..., N*-1*)*, because it contains the tardiness cost and the cost of the fixed resources. The procedure that generates this file is *generateDps1Code*.

As described above, the state of the DP model is given by the times of realization of the origin nodes (events) of the *u.d.c.* that defines the stage. Each *u.d.c.* contains one decision variable, by construction; all other activities have their allocations "fixed". For stage 1 (the final stage), they are the origin nodes of the activities that terminate at the last node.

[6] N is the number of notes

In stage 1, the activity to optimize is the last activity of the set *ALP*. After determining the origins of the activities terminating at the last node, a cycle is initiated for each origin node (arranged topologically) which now assumes the role of the last node, utilizing the realization times that have been previously generated for its state nodes. This results in the determination of the time of realization of the node under consideration as being the maximum of the times of realization of the state nodes plus the duration of the corresponding activities (which depend on the resource allocation). Then, for the activity containing the decision variable to optimize, one enumerates all its possible values, determining the total expected cost (which includes the delay cost and the cost of the resource, as well as the cost of the fixed resources). At the end, for every possible realization times of the state, a minimal cost is evaluated.

The procedures that generates the remaining stages is called *generateDpsNCode*. This is a general program that can be applied to generate the files $\{dpsn\}_{n=2}^{N-1}$.The instructions for this procedures are similar. The procedure evaluates the cost of the resource for the decision variable, which is added to the expected cost of the preceding stage.

As the coding was done in Matlab, to be able to use all the new files generated it is necessary to use the instruction *rehash*, which puts them in memory. This way they will be ready to be called to optimize the introduced network.

3 Examples

The program outlined above was implemented on a set of four projects that range in size from 5 to 18 activities. Due to limitation in space, these examples could not be presented here but they can be seen in the internet[7], or requested by e-mail[8]. For each project, we present the network and its parameters, together with the solution given by the program. The solution times varied from a few seconds to five *days* on a Pentium III, 650 MHz, 128 MB. The program output indicates the "optimal" cost and the "best" values for the variables that emanate from node 1, as well as the "best" values for the fixed variables. The values of the remaining decision variables depend on the state of the project, at the time of the decision, and can be determined from the optimal policies developed for the corresponding stage. The words *optimal* and *best* are put between quotation marks because they are not the *absolute* optima and best due to the discretization of the work content and the times of node realizations. Finer meshes may result in improved optima, at the price of (greatly) increased computational effort.

These set of experiments were done to demonstrate "proof of concept": armed with a powerful computer the proposed model can be applied to any network. But for the larger ones, the time necessary to get results is prohibitive with the facilities at our disposal. Our future research shall focus on the development of approximations that do not detract much from the optimum, but are more modest in their computing requirements.

[7] www.eng.uminho.pt/~dps/anabela (Topic: research).
[8] anabelat@dps.uminho.pt.

4 Complexity of the DP Model

The order of complexity of the DP model may be calculated as follows.

In a project containing *A* activities and *N* nodes, there are at most *N*-1 stages of the DP iterations which correspond to the longest path. At each stage there are, on average, *m=A/N* source nodes.

If the time of realization of a node is discretized at k_1 discrete points, there shall be k_1^m states at each stage. For each state one must consider k_2 possible allocations of the resource. There are (*A*-*N*+1) "fixed" activities, which gives rise to $k_2^{(A-N+1)}$ enumerations to secure the unconditional optimum. Therefore the complexity of the procedure is $O\left(N.m.k_1^m.k_2^{(A-N+1)}\right)$. Since *A* is bounded from above by N^2, one can state the order of complexity as $\mathrm{O}\left(N^2.k_1^N.k_2^{N^2}\right)$.

References

Adlaka, V.G., Kulkarni, V.G. (1989). A Classified Bibliography of Research on Stochastic PERT Networks: 1966-1987. INFOR, Vol. 27, nº 3.

Angus, R.B., Gunderson, N.A. (1997). Planning, Performing, and Controlling Projects: principles and applications. Prentice Hall, London.

Bellman, R.E. (1957). Dynamic Programming. Princeton University Press, New Jersey.

Bellman, R.E., Dreyfus, S.E. (1962). Applied Dynamic Programming. Princeton University Press, New Jersey.

Burke, R. (1992). Project Management: planning and control.. 2nd ed., Wiley, Chichester.

Brooks, G.H., White, C.R. (1965). An Algorithm for Finding Optimal or Near Optimal Solutions to the Production Scheduling Problem. Journal of Industrial Engineering, January-February Issue, 34-40.

Christofides, N., Alvarez-Valdés, R., Tamarit, J.M. (1987). Project Scheduling with Resource-Constraints: a branch-and-bound approach. European Journal of Operartional Research 29, 262-273.

Davis, E.W. (1973). Project Scheduling under Resource Constraints: historical review and categorization of procedures. AIIE Transactions, Vol. 5, nº 4, 297-313.

Demeulemeester, E.L., Herroelen, W.S. (2002). Project Scheduling: A Research Handbook. Kluwer Academic Publishers, Boston. ISBN 1-40207-051-9.

Elmaghraby, S.E. (1993). Resource Allocation via Dynamic Programming in Activity Networks. European Journal of Operational Research 64, 199-215.

Elmaghraby, S.E., Fathi, Y., Taner, M.R. (1998). On The Sensitivity of Project Variability to Activity Mean Duration. International Journal of Production Economics 62, 219-232.

Elmaghraby, S.E. (1977). Activity Networks - project planning and control by network models. John Wiley and Sons, Inc., New York.

Elmaghraby, S.E. (2000). Optimal Resource Allocation and Budget Estimation in Multimodal Activity Networks. Private Communication, North Carolina State University, Reileigh - North Carolina - USA.

Held, M., Karp, R.M. (1962). A Dynamic Programming Approach to Sequencing Problems. Journal of the Society for Industrial and Applied Mathematics, March.

Lewis, J.P. (1995). Project Planning, Scheduling & Control: a hands-on guide to bringing projects in on time and budget. New York: McGraw-Hill.

Lewis, J.P. (1997). Fundamentals of Project Management. Amacon, New York.

The MathWorks, Inc (1997). Using Matlab.

Moder, J.J., Phillips, C.R., Davis, E.W. (1983). Project Management with CPM, PERT, and Precedence Diagramming, 3d ed., Van Nostrand, New York.

Neumann, K., Schwindt, C., Zimmermann, J. (2002). Project Scheduling with Time Windows and Scarce Resources. Springer-Verlag, Berlin. ISBN 3-540-4263-6.

Patterson, J.H., Slowinski, F.B., Talbot, F.B., Weglarz, J. (1989). An Algorithm for a General Class of Precedence and Resource Constrained Project Scheduling Problems. In Slowinski, R., Weglarz, J. (Eds). Advances in Project Scheduling. Elsevier, Amsterdam, 3-29.

Sprecher, A., Hartmann, S., Drexl, A. (1997). An Exact Algoritm for Project Scheduling with Multiple Modes. OR Spektrum 19, 195-203.

Tereso, A.P., Araújo, M.M., Elmaghraby, S.E. (2001). Adaptive Resource Allocation in Multimodal Activity Networks. Research Report, Universidade do Minho, Portugal, submitted for publication.

Tereso, A.P. (2002). Project Management - Adaptive Resource Allocation in Multimodal Activity Networks. PhD Thesis, Universidade do Minho, Portugal.

Appendix

Assuming that W is exponentially distributed, with parameter $\lambda = 0.1$, then the probability density function is given by (11),

$$p(t) = \lambda e^{-\lambda t} \text{ with } \lambda = 0.1. \tag{11}$$

For $t_1 \le t \le t_2$, we must have, iteratively,

$$\int_{t1}^{t2} \lambda e^{-\lambda t} dt = e^{-\lambda t2} - e^{-\lambda t1} = 1/K, \tag{12}$$

$$\text{initiated at } t_1 = 0 \text{ and terminated at } t_2 = \infty, \tag{13}$$

which immediately yields the desired values of the intermediate t_2's.

With $\lambda = 0.1$ and according to the assumption of four values of W, all with the same probability (=0.25), the *t* axis is divided into four zones, all with the same area under the curve, as depicted in figure 2.

The limits for these four zones are 0, t_1, t_2, t_3, $+\infty$. The evaluation of t_1, t_2 and t_3 is done by simple inversion of the exponential function to yield,

$$t_1=2.8768, \; t_2=6.9315, \; t_3=13.8629, \tag{14}$$

which result in the intervals:

$$[0, 2.8768[; \; [2.8768, 6.9315[; \; [6.9315, 13.8629[; \; [13.8629, \infty]$$

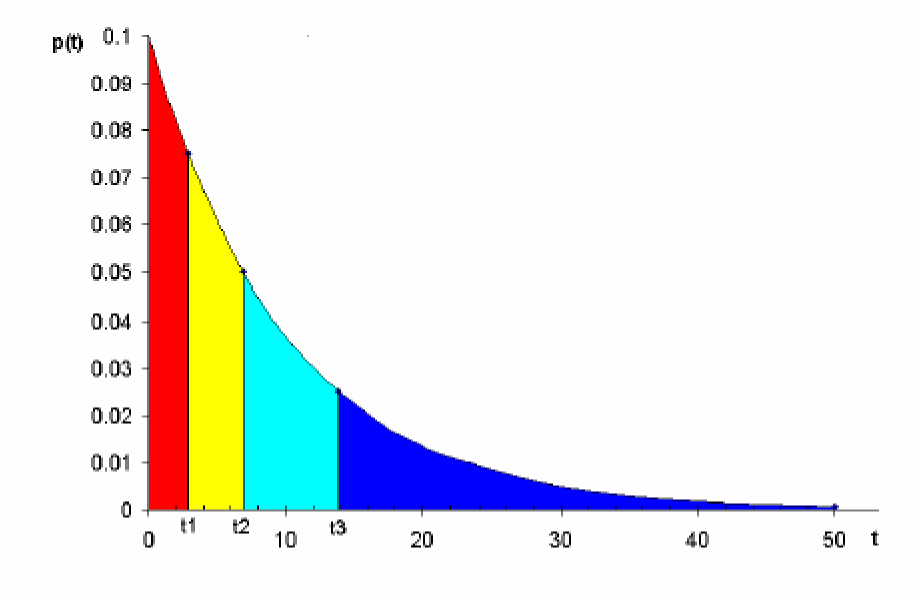

Figure 2: Exponential distribution divided in four zones, with the same area

The discrete probabilities will be positioned at the means of the intervals obtained above. The global mean is known to be

$$\text{Global Mean} = \int_0^{+\infty} tf(t)dt = \frac{1}{\lambda}. \quad (15)$$

Similarly, the partial means are evaluated using the expression (16), because the area corresponds to a probability of 0.25.

$$\text{Segment Mean } [t_i, t_j] = \frac{1}{0.25}\int_{tj}^{ti} tf(t)dt. \quad (16)$$

So, we have the discrete work content values:

$$w_1 = 1.3695,\ w_2 = 4.7675,\ w_3 = 10.0000,\ w_4 = 23.8629 \quad (17)$$

The program confirms the correctness of these evaluations by comparing the Global Mean to the mean of the Segment Means. The mean of the four values in (17) is indeed equal to the global mean $\left(\frac{w_1 + w_2 + w_3 + w_4}{4} = \frac{1}{\lambda} = 10\right)$, having all equal probability and being representative of the exponential distribution. These are the values that are going to be evaluated by the function *Generate*W.