Control and Cybernetics

vol. **38** (2009) No. 3

Optimal resource allocation in stochastic activity networks via the electromagnetism approach: a platform implementation in Java*

by

Anabela P. Tereso[1], Rui A. Novais[1], M. Madalena T. Araújo[1] and Salah E. Elmaghraby[2]

[1] Universidade do Minho, 4800-058 Guimarães, Portugal
[2] North Carolina State University, Raleigh
NC 27695-7906, USA
e-mail: anabelat@dps.uminho.pt, rui.fafe@gmail.com,
mmaraujo@dps.uminho.pt, elmaghra@eos.ncsu.edu

Abstract: An optimal resource allocation approach to stochastic multimodal projects had been previously developed by applying a Dynamic Programming model which proved to be very demanding computationally. A new approach, the Electromagnetism-like Mechanism, has also been adopted and implemented in Matlab, to solve this problem. This paper presents the implementation of the Electromagnetism approach using an Object Oriented language, Java, and a distributed version to be run in a computer network, in order to take advantage of available computational resources.

Keywords: resource allocation, project scheduling, project management, stochastic models, electromagnetism-like mechanism.

1. Problem definition and review of prior work

1.1. Introduction

The problem addressed in this paper, described in more detail in Section 1.2, has been treated via a dynamic programming model (DPM) in a Matlab implementation by Tereso, Araújo and Elmaghraby (2004), and in a Java implementation by Tereso, Mota and Lameiro (2005). This paper treats the same problem via the Electromagnetism-like Mechanism (EM) in a Java platform in two modes: a single processor mode and a multi-processor distributed mode. It compares the results with those of the DPM, and demonstrates the superiority of the EM in either mode for large projects.

*Submitted: September 2006; Accepted: January 2009.

The motivation for this research is as follows. The correct allocation of resources is crucial to the success of projects that otherwise would end up with time and cost overruns. Classical models of resource allocation assumed that each activity has a deterministic duration and known resource requirements, and attempted to "optimally" schedule the activities (in whichever sense optimality was defined, such as the minimization of total project duration or the minimization of the costs involved in project execution). This gave rise to the well known RCPSP (Resource-Constrained Project Scheduling Problem) which attracted a great deal of interest by a number of researchers, see the books by Neumann, Schwindt and Zimmermann (2001) and Demeulemeester and Herroelen (2002) for a comprehensive summary of the state of the art in these studies as of 2002. This perspective suffers from the serious flaw of ignoring the *uncertainty* present in real life projects. Unfortunately, the inclusion of uncertainty in these models seemed to constitute an insurmountable obstacle, and researchers had to increase the estimate of the time of realization of certain key events by an allowance (or "gap") that would act as buffer in case the preceding activities took longer than estimated. Such "fudge factor" is claimed to provide for robustness of the resulting schedules; see for instance the recent papers by Van De Vonder, Demeulemeester and Herroelen (2007) and by Van De Vonder et al. (2006).

There has been a rather large amount of research devoted to the problem of project scheduling under constrained availability of resources, in both the deterministic and stochastic contexts. Although tangential to the subject matter of this paper, because the vantage point of discussion is different from ours, we cite some of the research that dealt with the issues of job scheduling and the optimal allocation of (discretely or continuously) divisible resources to tasks (or jobs) in order to achieve some given objective, in both the general scheduling context as well as in the context of projects characterized by activities and resources. We limit the citation to papers and books that appeared within the last ten years. The reader is directed to the references cited in each of the contributions to gain a more comprehensive view of the various lines of research in prior years.

An excellent survey of the contributions to the famous (or rather infamous) "resource constrained project scheduling problem" (RCPSP) as of 1997 may be found in the paper by Herroelen, De Reyck and Demeulemeester (1998). The survey is updated in the book by Demeulemeester and Herroelen (2002). A more recent book, edited by Józefowska and Węglarz (2006), gives more contributions and perspectives on the RCPSP and related problems; see chapters: 4. "Due dates and RCPSP", 5. "RCPSP with variable intensity activities and feeding precedence constraints", 7. "Lower bounds for resource constrained project scheduling problem", 9. "A metaheuristic approach to the resource constrained project scheduling with variable activity durations and convex cost functions", 11. "Population learning algorithm for the resource-constrained project scheduling", 12. "Resource constrained project scheduling: A hybrid neu-

ral approach", and 15. "Resource-constrained project scheduling with time windows". The handbook edited by Błażewicz et al. (2007) has also some chapters related to the problem treated here; see in particular chapter 12. "Scheduling under resource constraints".

Janiak (1998) treats the two-machine flowshop problem when job processing times may be reduced linearly by the application of a limited, continuously divisible resource, such as financial outlay, energy, fuel, catalyzer; etc. He proves that the decision form of this problem is NP-complete even for the fixed job processing times on one of the machines and identical job reduction rates on another. He identifies some polynomially solvable cases of the problem (such as the case of singleton possible allocation to each job, which reduces the problem to the classical two-machine flowshop treated by Johnson, 1954, and the case when the sequence of performing the jobs is given), and provides four simple and modified approximate algorithms together with their worst case and experimental analysis. Also, he offers a fast exact algorithm of the branch and bound type based on some elimination properties of the problem, with some computational results and possible generalizations, such as a bicriterial approach. (See also some previous work of the same author – Janiak and Szkodny, 1994, and Janiak and Kovalyov, 1996.)

Then, Janiak and Portmann (1998) follow up on Janiak (1998) by extending the problem to general flowshops with more than two machines. They make the assumption that the processing times of jobs on some machines are linear, decreasing functions with respect to the amount of continuously divisible, non-renewable, locally and totally constrained resources, and they limit the discussion to permutation flow-shops. The objective of the study is to find a processing order of jobs and a resource allocation that minimizes the length of the makespan. Since the problem is strongly NP-hard, they propose a Genetic Algorithm approach to solve it. They give some characterizations of job interchanges that would lead to improvements in the objective function, and use these properties in their genetic algorithm. The results of some computational experiments are also given.

The problem treated by Ahn and Erenguc (1998) comes closest to the one treated here, except that: (i) they assume a finite number of discrete resource allocations and corresponding activity durations and costs, which they label as RCPSPMCM (for "the RCPSP with multiple crashable modes"), and (ii) the relationship between the resource allocated and the duration of the activity is not derived from the concept of a fixed work content. Similar to our scenario described below, the project is assumed to have a predetermined due date, $T > 0$. If the completion time of the project exceeds T, a predetermined penalty cost is incurred for each period the project is delayed beyond its due date. If the project due date cannot be violated, then the penalty is set to infinity. A schedule for RCPSPMCM consists of the triplets (finish time, mode, duration) for each of the activities of the project. A schedule is said to be feasible if it satisfies four conditions: (i) each activity is assigned a mode within its set of feasible modes,

(ii) for each activity the scheduled duration is between the normal and crash durations of the selected mode, (iii) all the precedence relations are satisfied, and (iv) resource requirements in each period do not exceed their respective capacities. Project cost is the sum of all activity costs and the penalty cost for completing the project beyond due date. The objective in this problem is to find a feasible schedule minimizing total project cost. The authors are led to an integer linear program model which they resolve heuristically using a two-stage procedure: in the first stage they generate a feasible schedule, and in the second they try to improve the feasible schedule by applying six improvement rules that they have developed. The second stage is repeated until application of these rules stops yielding further improvements. Then, the procedure goes back to the first stage, generates a new feasible schedule and moves to the second stage. Each time the procedure goes back to stage one, a new "pass" is started. The heuristic procedure terminates either when a predetermined number of passes is made, or when a predetermined computational time is reached. Computational experiments were performed with 100 problem instances randomly generated. The performance of their heuristic procedure was compared with the truncated exact solution procedure of Sprecher (1994) and Sprecher, Hartmann and Drexl (1994) (designed to solve the RCPSPMM, the RCPSP with multiple modes) and the truncated exact solution method of Ahn and Erenguc (1995) (designated as the RCPSPMCM) and was shown to outperform them (in the value of the objective function) for the problem instances considered. No measure of the time to reach the various solutions is given. The heuristic procedure of Ahn and Erenguc was later tested in the paper of Erenguc, Ahn and Conway (2001), which used a branch-and-bound procedure to achieve the exact optimum.

Other references cited here give a broader view of the problems addressed and the approaches used for their resolution, chronologically: Dauzere-Peres, Roux and Lasserre (1998), Tsai and Gemmil (1998), Artigues, Roubellat and Billaut (1999), Artigues and Roubellat (2000), Elmaghraby (2000), Penz, Rapine and Trystram (2001), Stork (2001), Bouleimen and Lecocq (2003), Bellenguez (2004), Buddhakulsomsiri and Kim (2006, 2007), Dieter et al. (2006), Lorenzoni, Ahonen and De Alvarenga (2006).

1.2. Problem definition

Our approach to the issue of optimal resource allocation under uncertainty differs radically from prior treatments in two respects: (*i*) it considers the *work content* of the activity (which embodies both resources *and* duration) as the fundamental variable of concern, and (*ii*) it imparts randomness to the work content, not duration. The duration of the activity is then *derived* from knowledge of the work content and the resource allocation; with the latter now becoming the *decision variable* of central importance, as it should be. It is our contention that the lack of practical implementations of the monumental research effort devoted to the RCPSP lies in the deficiencies in its fundamental premises: the

world is *not* deterministic, and managers are *not* confined to shifting activities in time to satisfy the limits of the resources availability. Rather, *managers manage, dynamically, the allocation of the available resources in a stochastically changing environment*. Adopting this point of view forces one to focus on the activity work content (referred to by some recent writers as the activity total "energy" requirements). According to the amount of the resource allocated, the duration will vary: more resources will result in a shorter duration, and conversely. Naturally, the amount of resource allocated to an activity, which we denote by x, is bounded from above (e.g., one cannot have more than so many men working on the activity) and from below (e.g., one cannot rent a truck for less than half a day). Putting all these notions together one ends up with the following functional relationship among the three variables of concern,

$$Y = \frac{W}{x}, \quad l \leq x \leq u \tag{1}$$

where W stands for the work content, a random variable (r.v.); x is the resource allocated to the activity (the *decision variable*) in units appropriate for the activity (personnel, machinery, funds, fuel; etc.), l and u are the bounds on the permissible allocation, and Y is the activity duration. Observe that Y is also a random variable, since it is a multiple ($= \frac{1}{x}$) of the work content W. A more general formulation of the relationship among the duration, the resource allocation and the work content may be stated as

$$Y = \frac{W}{x^{\alpha}}, \quad l \leq x \leq u \tag{2}$$

where α represents the degree of "interference" among the units of the resource if $0 < \alpha \leq 1$, and the degree of "synergism" if $\alpha > 1$. In the sequel we shall assume $\alpha = 1$, representing balanced response (no interference or synergism).

To illustrate, consider a simple case of W being a constant (formally, a "degenerate" random variable, assuming one value only with probability 1); say $w = 36$ *man-hrs* (about one week of effort by one worker)[1]. Then, if $x = 2$ (meaning 2 men are allocated to the activity) then it will take $y = 18$ *hrs* to complete the activity (about half a week). But if $x = 4$ *men*, then $y = 9$ *hrs* (slightly over a day); etc. Returning to our assumption of randomness, suppose that the work content W is in fact stochastic and that it can be anywhere between 24 and 48 *man-hrs*. Formally, one may say that W is uniformly distributed between 24 and 48, expressed as $W \sim U[24, 48]$. Then if $x = 2$ we shall have Y anywhere between 12 and 24 *hrs*, while if $x = 4$, Y shall be anywhere

[1]We reserve symbols in capitals to denote r.v.'s, while symbols in lower case represent their deterministic realizations.

between 6 and 12 *hrs*. Summarizing this example we have,

$$\begin{aligned} &W \sim U[24, 48] \\ &x = 2 \Rightarrow Y \sim U[12, 24], \text{and} \\ &x = 4 \Rightarrow Y \sim U[6, 12]. \end{aligned} \tag{3}$$

This example represents an ideal situation in which doubling the resource allocation halves the time. But suppose this is not the case: suppose that doubling the resource will indeed result in some improvement in duration, but the improvement is smaller than $1/2$, say, only 0.4 of its original value. Then we seek the exponent α in the equation

$$\frac{new}{old} = 0.6 = \frac{\dfrac{W}{(2x)^\alpha}}{\dfrac{W}{x^\alpha}} = \frac{1}{2^\alpha} \Rightarrow \alpha = \frac{\log(0.6)}{\log(0.5)} = 0.737. \tag{4}$$

Now we would have $Y = W/x^{0.737}$, which results in

$$\begin{aligned} &x = 2 \Rightarrow Y \sim U\left[\frac{24}{2^{0.737}} = 14.1, \quad \frac{48}{2^{0.737}} = 28.8\right], \text{ and} \\ &x = 4 \Rightarrow Y \sim U\left[\frac{24}{4^{0.737}} = 8.64, \quad \frac{48}{4^{0.737}} = 17.28\right]. \end{aligned} \tag{5}$$

Observe that not only the (lower and upper) limits are bigger than before, but also the range of the duration increased (from 12 to 14.7 for $x = 2$ and from 6 to 8.64 for $x = 4$), albeit is still smaller than the original range of 24. The impact of increased resource allocation to the activity has indeed diminished.

The problem treated in this paper may be more formally stated as follows. Given an AoA (Activity-on-Arc mode of representation) network defining a project, we wish to find the resource allocation minimizing total cost. This cost is the sum of two costs: (i) the "resource cost" (RC), proportional to the square of the resource usage for the duration of the activity, with constant of proportionality equal to c_R, being cost per resource unit, and (ii) the "tardiness cost" (TC), which is proportional to the magnitude of tardiness from a specified due date T, with constant of proportionality equal to c_L (the cost per time unit). Each activity a has stochastic work content W_a, assumed to be exponentially distributed (the reason for this choice shall be clarified next) with a parameter λ_a, which may vary for different activities. The duration of an activity a, denoted Y_a, depends on the work content and the amount of resource allocated to the activity as given by (1); $Y_a = W_a/x_a$; $0 < l_a \leq x_a \leq u_a < \infty$. There is only one resource of unlimited availability so that it does not impose any limitations of the number of concurrent activities. The goal is to minimize the total cost by selecting the optimal amount of resource allocated to each activity of the project.

Two elements in the above statement of the problem need some explanation. First, why assume exponentially distributed work content? The answer lies in our desire to relate the results secured under EM with the previously obtained results, which assumed the exponential distribution. Second, why the resource cost is assumed to be quadratic in the amount of resource assigned to the activity? The answer lies in that such assumption simplifies the analysis considerably by rendering the cost linear in work content. Indeed, if the resource cost is given by $c_R x^2 Y$, then substituting for Y from (1) we have

$$RC = c_R \cdot x^2 \left(\frac{W}{x} \right) = c_R \cdot x \cdot W. \tag{6}$$

Clearly, RC for any activity is an r.v. The tardiness cost TC is given by

$$TC = c_L \cdot \max\{0, t_{nn} - T\} \tag{7}$$

where t_{nn} is the random time of realization of node nn, the last node in the project, which signifies its completion.

Section 1.3 of this paper presents a brief introduction to the Electromagnetism-like Mechanism (EM) and its adaptation to our problem. In §1.4 we formally present the objective function and population analysis. Then we explain how the EM was adapted, in Matlab, to solve our problem (§1.5). In Section 2 we give the rationale for choosing the Java programming language (§2.1), the advantages of using Java instead of Matlab (§2.2), including some implementation details (the classes created) and the data structures used, and a description of the generic algorithm (§2.3). In §2.4 we present the EM in a distributed mode in order to take advantage of parallel computing. The results of our study and discussion of their significance are presented in Section 3. Section 4 draws some conclusions from our research and points out some future directions of research.

1.3. The Electromagnetism-like Mechanism

The Electromagnetism-like Mechanism (EM), developed by Birbil and Fang (2003), is based on the principles of electromagnetism. The algorithm begins by launching a population of "particles" into the feasible space. The "charge" of a particle determines the magnitude of the force of repulsion or attraction between it and each other particle in the population. If the value of the charge associated with a particle is low (in a minimization problem), this particle attracts all other particles of higher values; and conversely, a particle of high value repulses all particles of lower value. The direction of movement of a particle is determined as the resultant of all forces acting on it. The process moves the particle in the direction given by the resultant force, changing its coordinates by an increment. As particles are displaced from their original locations by these forces, a new configuration results and the process of determining forces and movement of

particles repeats. Asymptotic convergence to the optimum is guaranteed by Birbil, Fang and Sheu (2004).

In our case, we have a set of activities with associated stochastic work content, and a single resource to allocate to them. The correspondence between the above described EM concept and our problem may be conceived as follows. A "particle" is a specific vector of resource allocations. The "charge" of a particle is the value of the objective function, the sum of the resource and the tardiness costs. Implementation of EM requires definition of three structural parameters: (i) Size of the population of particles M. In our experimentation we fixed $M = 15$ for all networks. (ii) Number of samples K of the vector of work contents. We used different values of K, ranging from 10 to 1200. (iii) The maximum number of iterations I of the EM algorithm. In our experiments we limited the number of iterations to $25n$, as suggested by Birbil, Fang and Sheu (2004), since this number of iterations was found sufficient for convergence (n being the number of activities in the project network).

Recall that each particle $x^{(m)}$ is a vector of n elements representing the resource allocations to the project activities,

$$x^{(m)} = \left(x_1^{(m)}, \ldots, x_n^{(m)}\right) \quad m = 1, \ldots, M \tag{8}$$

Here, M is the size of the population of particles. In a project of, say, $n = 35$ activities, and $M = 15$, we take 15 points in the hypercube defined by the n inequalities $l_a \leq x_a \leq u_a$; $a = 1, \ldots, 35$. We select the M particles to span the feasible space of the resource allocation, as much as possible.

Fig. 1 illustrates a minuscule project of only $n = 2$ activities, $M = 5$ particles, and the forces acting on particle *#3*.

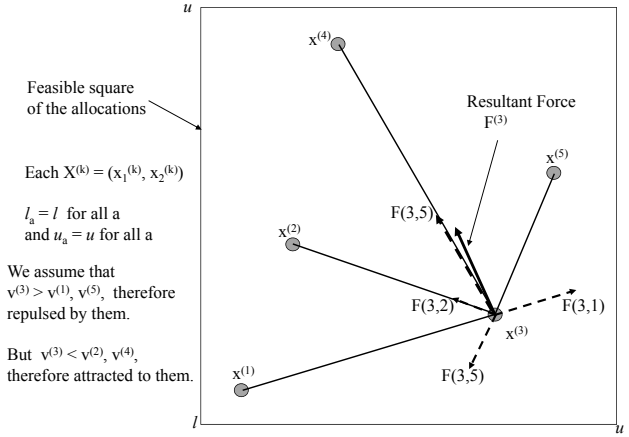

Figure 1. Forces acting on particle #3

Each particle in this example is represented by a two-dimensional vector: $x^{(m)} = \left(x_1^{(m)}, x_2^{(m)}\right)$, for $m = 1, ..., 5$. The "charge" of each particle is the average value of the objective function at that point, denoted by $\nu(x^{(m)})$,$m = 1, \ldots, M$. This value is determined through Monte Carlo sampling (there are K samples) of the vector of work contents $(W_a)_{a\in A}$ which, together with the allocation $x^{(m)}$, determine the "resource cost" as well as the time of project completion, denoted by $t_{nn}^{(m)}$ through standard critical path calculations. Knowledge of $t_{nn}^{(m)}$ enables determining the penalty for tardiness beyond the specified project due date T.

We define $\nu_{\min}$ as the minimal (average) value among all M points,

$$\nu_{\min} = \min_m \{\nu(x^{(m)})\} \tag{9}$$

It is important, for stability reasons, to "normalize" and "scale" these values, which result in the charge $q^{(m)}$ at point $x^{(m)}$. This charge is evaluated as follows,

$$q^{(m)} = \exp\left[-n \times \frac{\nu(x^{(m)}) - \nu_{\min}}{\sum_{k=1}^{M}\left[\nu(x^{(k)}) - \nu_{\min}\right]}\right], \quad m = 1, 2, ..., M. \tag{10}$$

Observe that a large $\nu(x^{(m)})$ results in a small $q^{(m)}$, and conversely, a small $\nu(x^{(m)})$ results in a large $q^{(m)}$. Indeed, at $\nu_{\min}$ the charge is 1, the maximum. The charge $q^{(j)}$ of particle j determines the force of attraction or repulsion between particle j and the other particles. For each pair of particles $x^{(j)}$ and $x^{(k)}$ suppose that $\nu(x^{(j)}) < \nu(x^{(k)})$, implying that $q^{(j)} > q^{(k)}$. Then, particle $x^{(k)}$ is "attracted" to particle $x^{(j)}$ by a force given by

$$F(j,k) = \left[(x^{(j)} - x^{(k)}) \times \frac{q^{(j)} \cdot q^{(k)}}{\left\|x^{(j)} - x^{(k)}\right\|^2}\right], \quad \forall j, k \tag{11}$$

and particle $x^{(j)}$ is "repulsed" by particle $x^{(k)}$ by the force of the same magnitude in the opposite direction. The direction of the attraction/repulsion force is along the line between the two particles with the arrow pointing from $x^{(k)}$ to $x^{(j)}$ for particle k and the reverse for j. The resultant vector force $F^{(m)}$ on each particle m is calculated by conventional methods (see Fig. 1). The force $F^{(m)}$ is then normalized to yield,

$$F^{(m)} = vector\ sum\ (F(j,m)),\ j \neq m,\ m = 1, \ldots, M. \tag{12}$$

This procedure is repeated for each particle $x^{(m)}$, $m = 1, \ldots, M$. Each particle $x^{(m)}$ is then moved in the specified direction by a random step given by

$$x^{(m')} \leftarrow x^{(m)} + \beta \cdot (RNG)^{(m)} \cdot F_{norm}^{(m)} \tag{13}$$

where β is selected randomly $\in (0, 1)$ and $(RNG)^{(m)}$ is the range of feasible movement of the particle towards the lower or upper bound. For instance,

suppose the allocation to activity a is bound by $l_a = 0.5 \leq x_a \leq 2.0 = u_a$, and that the current value of x_a of the particle is $x_a^{(m)} = 1.4$, and movement of the particle is towards increasing x_a. Then $(RNG)_a^{(m)} = 2 - 1.4 = 0.6$. If the random selection of β resulted in $\beta = 0.638$, then the coordinate shall change from x_a^m to $x_a^{(m+1)} = x_a^m + 0.638 \times 0.6 \times F_{norm}^{(m)} = x_a^m + 0.3828 F_{norm}^{(m)}$.

The movement of the particles continues $X^{(1)} \rightarrow X^{(2)} \rightarrow \cdots \rightarrow X^{(I)}$ until stopping condition is satisfied (with $I = 25n$). The allocation yielding $\nu_{\min}$ at algorithm stop is selected as the "optimal" allocation.

For each repetition of the experiment, a set of K vectors of work contents is generated and stays fixed thereafter. For each of the M particles generated during the EM process, the objective function is evaluated for each of the work contents generated. In other words, we calculate the value of the objective function for each particle, for each of the K vectors of work content. As mentioned before, the value of the particle (its "charge") is based on the average of the K values, $\nu^{(m)} = \sum_{k=1}^{K} \frac{\nu^{(k)}}{K}$. These average values are used in EM to decide on the forces acting on the particle. Finally, when we stop the movement of the particles we have one particle $\nu_{\min}$, whose average value is minimal. If more than one particle attains the minimum value, one is selected arbitrarily.

After a brief review of fundamental principles of the EM we now address the issue of the main differences between it and the DPM in dealing with randomness of the work content. For more insight into the DPM see Tereso, Araújo and Elmaghraby (2004).

DPM relies heavily on defining the state of the process which, in the case of interest to us, is the time of realization of the nodes in a uniformly directed cutset (*udc*) in the project network. Given the state of the project, the resource allocation vector to each activity in the $udc\{x_a | a \in udc\}$ defines the activity (stochastic) duration according to (1). The aggregate set of activities in the *udc* determines the (random) times of realization of the next adjacent *udc*'s. Applying the dynamic programming recursion equation, results in the recursive optimization of the resource allocation at each possible state of the process. This explains the need to discretize both the state space and the decision spaces in the DPM. It also leads to the estimation of the computational complexity as

$$O\left(|d_N|^N |d_D|^D |A|\right), \tag{14}$$

where N is the average number of "source nodes" in any *udc*, d_N is the number of discrete times of realization of any node, D is the average number of activities in any *udc*, d_D is the average number of discrete decisions for any activity, and A is the number of *udc*'s in the network. The form of (14) explains the diffculty in solving large scale projects via DPM with any meaningful discretization scheme. The EM confronts the randomness in the work content of the activities in a completely different manner. It simply samples the work content of each activity following standard Monte Carlo sampling procedures (by securing the inverse of

the cumulative distribution function of the random number generated). Then, for each sample, determines the various time and cost parameters in the network following standard CPM calculations. Repeated sampling of the work content generates the distributions of these parameters on the basis of which the various measures of performance are based. The complexity of the EM increases linearly in the number of activities in the project, and depends rather heavily on the size of the population (the parameter M) and the number of random samples taken (the parameter K).

1.4. Objective function and population analysis

The goal of the Electromagnetism approach in our problem is to minimize the expected project cost associated with each particle (assignment of resources) in the population by changing its resource assignment. One has to be very careful in choosing the algorithm parameters K, M and n. The parameters that influence the algorithm run time are:

- **Number of sampled work contents** (K) – The EM will calculate the solution for each vector of work contents generated, returning the mean value. A larger number of sampled work contents per activity will result in a more accurate value of the average cost.
- **Population size** (M) – A large number of particles will result in testing a large number of solutions in each iteration. A larger population yields a better result.
- **Number of network activities** (n) – A project with large number of activities leads to a larger and more complex network. The EM will take more time finding or calculating the CPM value (representing total duration of the project using the Critical Path Method, used to calculate the total cost).

We assumed that the project has a due date T and a Tardiness Cost (TC). The penalty constant c_L represents the cost per unit time after the due date. The objective is to find the resource allocation among activities so as to minimize total cost C. This cost is given by:

$$C = \sum_{a=1}^{n} RC_a + TC, \tag{15}$$

where RC_a is the resource cost of activity a (see (6)) and TC is the tardiness cost (see (7)).

If we increase the amount of resource to each activity, the TC will be lower, but the RC will be higher. The objective is to balance these two costs to achieve the minimal overall cost of the project.

1.5. The Matlab implementation

The EM had been previously implemented in Matlab (Tereso and Araújo, 2004). The pseudo-code can be accessed at www.dps.uminho.pt/pessoais/anabelat, or upon request by email from the first author. Here, we will only refer to the more important aspects of that implementation. There are two main procedures: (1) the procedure *Initialize.m* initializes the data structure that supports the population of particles. It begins by initializing the project cost and the coordinates associated with each particle. (2) The procedure *envis.m* is the main procedure for this implementation. It is composed of a primary loop that calculates the resultant force acting on a particle, moves the particle in corresponding direction, and executes local search to determine the best particle location each time. The procedures called in the main loop of *envis.m* are: (*i*) the procedure *calcF.m* calculating forces acting on particles, (*ii*) the procedure *move.m* moving the particles and (*iii*) the procedure *local.m* performing local search. The loop *envis.m* executes a number of iterations predefined at the beginning of the algorithm.

2. Application development

In this section, we discuss main issues of concern in the development of EM using Java.

2.1. The choice of a programming language

We chose Java as the programming language for implementation of EM because the dynamic programming model (DPM), proposed to solve the problem stated in §1.2, was written in both Matlab (Tereso, Araújo and Elmaghraby, 2004) and Java (Tereso, Mota and Lameiro, 2005), and we wanted to create an application enabling direct comparison of EM and DPM. The new application, dubbed GP2006, would allow us to experiment with both approaches and compare the results. Additionally, Java has also some computational advantages. It is an Object Oriented (OO) language, which allows for developing a more structured code. It is easier to program and a higher level language than, e.g., C++. It has the hyper threading technology, allowing for taking advantage of today's processors, and can be run in different operating systems (Linux, Windows, Macintosh), a huge advantage over C++, especially in a distributed implementation.

2.2. Matlab vs. Java

2.2.1. Data structures and input parameters

To represent the project we replaced the list of activities (in the Matlab implementation) containing five fields per activity (source node, target node, pa-

rameter λ, lower and upper bounds on resource allocation) by a more complex structure. To represent the "list of activities", we defined three classes in Java:

1. **Node** – to represent each node of the graph with information about immediately preceding and immediately succeeding nodes and the activities connected to the node.
2. **Activity** – to represent an activity with information about parameter λ, lower and upper bounds on the resource allocation; and
3. **Network** – which contains a list of activities and nodes.

One of the most important considerations when comparing implementations in Matlab and Java is the speed of accessing the data structures that support an algorithm written in Java. In our case we used the data structure called *HashMap*. The operations of *search, remove, add* and *travel* in this data structure are much faster than the ordinary list manipulations in Matlab. It is quicker for the processor to decode and execute a code in Java than in Matlab for these operations.

We used two *HashMaps* to support the algorithm. One that holds the populations of particles (feasible solutions to the problem) and another to hold the work content necessary to calculate the Project Cost associated with each particle, and determine the best particle(s) found in each run of the algorithm. This support is needed because then we can program the algorithm to execute N times, each with a different "seed" population of feasible solutions, record the best particle in each run, and thus secure a population of "best solutions". More information about this procedures and code developed can be found in Novais (2005).

2.2.2. Other important classes

Next we describe some other important classes used in this implementation.

1. We begin by introducing the class that represents the solutions to our problem, the class **Ion**. This class holds a number of attributes, namely attributes to store the Project Cost (solution cost), particle "charge" (force of attraction or repulsion of the particle to the rest of the population), the particle coordinates (resource allocation to each activity of the network project) and the force associated with that particle.
2. The class **Project Cost** used to calculate the cost of the project associated with one realization of the work content for each activity. It uses the CPM, the work content generated at the beginning of the algorithm and the activity network to calculate the resource cost and the delay cost that sum up to project cost.
3. The project is represented by an object from the class **Network**. The network is constructed using other classes that represent network components. These classes are **Node** and **Activity**, representing nodes and activities of the network, respectively.

4. The most important class is the class **Problem**. This class is responsible for holding all the data structures, and the activity network. It has a main routine responsible for executing a number of predefined operations such as *CalcF*, *Move* and *Local*. These operations are the ones mentioned above in Section 1.5.
5. Finally we have another class named **Configuration**, which holds all the parameters used by the algorithm. These parameters are the population size, the number of activities in the network, the due date, the penalty cost and the number of work contents to be generated.

For more information on these classes and on the code developed see Novais (2005). Fig. 2 presents diagrammatically the various classes that support the algorithm.

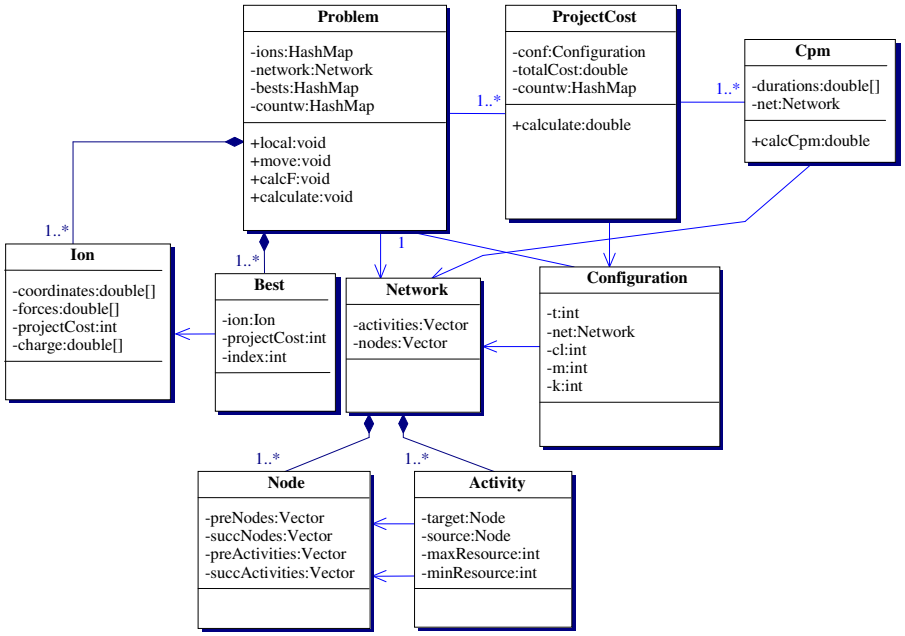

Figure 2. Class diagram

This diagram, written in UML (Unified Model Language) represents a class diagram. UML is a graphical language used to specify, build and visualize OO information systems. Class diagrams are most common in modeling OO systems. We use class diagrams to model the static design view of a system. For each object, the diagram describes its identity, the relationships with other objects and the internal attributes and operations (for more details see Booch, Rumbaugh and Jacobson, 1998).

2.3. The algorithm

In this section we will briefly describe how the algorithm works. To start the algorithm, we generate randomly K vectors of work content. We used values of K ranging from 10 to 1200. These vectors are not changed during the same execution of the algorithm, to keep the objective function stable. Then we generate M vectors of X ($M = 15$: size of the population). For each vector of particles (X) and for each vector of work contents (W) total cost is evaluated. The objective function value of each particle is the mean cost of all W's. Charges and forces are then evaluated. Points are moved to obtain a set of new m points. This process continues until the limit number of iterations is reached. In Fig. 3 we present the generic algorithm that describes the steps of this process.

1. Generate K vectors of $W=(w_1..w_n)$ randomly
2. Generate m vectors of $X=(x_1..x_n)$ to start with
3. For each vector X
4. For each vector W
5. $rc=\sum x_a W_a$
6. $tc = c_L max\ \{0, t_{nn}-T\}$
7. $c=rc+tc$
8. End for
9. $f = \sum c / K$
10. Evaluate charges
11. Evaluate forces
12. End for
13. Move the points
14. Go to step 3 until nº of iterations specified is reached

Figure 3. The algorithm

2.4. Distributed implementation of the Electromagnetism approach

In the distributed implementation we have a machine called Server that begins by creating the problem configuration, setting up the parameters (work contents, population size, activity network and number of clients) and setting up all data communication channels with the clients. The communication between the server and the clients can be described in the following steps: (1) The server sends the configuration to all the clients in the network. In this step the server also sends the work contents necessary for all clients to calculate the project cost associated with each *ion* belonging to the population. (2) The clients send one message each telling if the configuration and work contents have been well received. (3) The server now sends all *ions* one by one to each client in order to calculate the associated project cost. (4) The clients calculate project cost and return the solution to the Server, which evaluates the mean of all the results, associating the value to *ion*'s project cost. Steps 3 and 4 are repeated for

all *ion*s and all iterations of the algorithm. (5) After collecting all population values, the Server sends a signal to all clients telling them to shut down. After that, the server stores the best solution and terminates.

We use sockets and data streams for communication between server and clients. The Server creates a communication socket associated with an IP address. To connect to the server, clients must have this address in a configuration file, set up manually. The Server process expects a number of connections, which is set in its own configuration file.

The algorithm itself was not changed with respect to evaluating charges and forces. It was changed in that it distributes all the vectors of work contents generated over the available computers of the network for evaluating the total cost of each sample vector. In this way we could take advantage of the available resources. That is, if we generate 800 vectors of work contents, and we have 10 computers in a network, each one will evaluate the cost of 80 vectors. Strictly speaking this may be viewed as a distributed load allocation implementation for the EM rather than a distributed implementation of EM.

3. Results

3.1. Experiment layout

We used a set of 14 networks with the number of activities ranging from 3 to 49. The networks chosen enabled analysis of a spectrum of different network complexities. These networks were also used in prior studies (Tereso, Araújo and Elmaghraby, 2004; Tereso and Araújo, 2004; Tereso, Mota and Lameiro, 2005), allowing for comparison of performance and results. Table 1 shows the characteristics of each network tested.

The due date T was selected using CPM and duration of the longest path, assuming the mean work content and the quantity of resource x_a equal to 1. Thereby the duration of each activity is fixed at $y = \overline{W}$. T was selected to be slightly greater (1.04-1.09) than the length of the longest path (in the CPM calculations). c_L was set up so as to allow for some tardiness cost if the quantity of resources used is low. We normalized the marginal resource cost c_R at 1.

In the EM implementation we used different seeds for each run, to generate the Work Contents randomly. Then we selected the best result obtained for all the runs. The seed was generated based on the value of the actual time, in milliseconds, in order to ensure randomness.

3.2. Single mode results

The results reported here were obtained using an Intel Pentium IV E 2.6 GHz with 512MB of RAM under Microsoft Windows XP Professional SP2. The details of the networks tested are described in Appendix A (Net1 to Net13).

A common characteristic of the algorithms proposed for solving problems like the one under study is the presence of several random components, which

Table 1. Network characteristics

Network	N° of activities (n)	CP length	Due date (T)	Ratio ($\frac{T}{CP}$)	Unit Delay Cost (c_L)
1	3	15	16	1.067	2
2	5	115	120	1.043	8
3	7	62.9	66	1.049	5
4	9	100	105	1.050	4
5	11	26.67	28	1.050	8
6	11	62.08	65	1.047	5
7	12	44.72	47	1.051	4
8	14	35.5	37	1.042	3
9	14	178.58	188	1.053	6
10	17	44.98	49	1.089	7
11	18	106.11	110	1.037	10
12	24	212.05	223	1.052	12
13	38	143.99	151	1.049	5
14	49	210.12	221	1.052	5

leads to obtaining different results under different executions. Hence the need for extensive experimentation according to a well-designed scheme, in order to be able to reach meaningful results.

The experiments allowed us to compare the performance of the EM in Java with its performance in the previous Matlab implementation (Table 4). We were also able to compare the performance of the EM in Java with the DPM in Java as presented in Tereso, Mota and Lameiro (2005). Tables 2 and 3 give the results of EM and DPM in Java for different numbers of K. In Tables 2 and 3 we present, besides the network parameters (n, T and c_L), the total expected cost obtained (C), the run time ($RunT$) and the number of objective function evaluations done ($N.Ev$). $K1$ is a parameter used in the DPM, and represents the number of discretization points used for the decision variables.

Table 2 and 3 demonstrate that the EM has the advantage over DPM in computing speed, with the advantage growing for large networks. For networks with 20 or more activities DPM is incapable of reaching any conclusion in reasonable time, while EM succeeds in achieving the result. DPM has the upper hand over EM when dealing with small networks. This is because the EM has to create and initialize a population with M elements under all network sizes and configurations, which is a "fixed cost" in run time that adversely impacts the total run time for small networks, but is relatively insignificant for large networks.

In Table 4 we present EM run time for Matlab versus Java implementations.

As expected, Java is faster than Matlab. The effort spent on using a faster programming language allowed us to save a lot of time in obtaining the results (up to 43 minutes in Network 13).

Table 2. Results for selected networks ($K = 10$)

	n	T	c_L	EM (Java)			DPM (Java)		
				C	RunT	N.Ev	C	RunT	K1
Net1	3	16	2	23	0.23s	1125	44	0.02s	5
Net2	5	120	8	166	0.42s	1875	305	0.14s	5
Net3	7	66	5	66	0.84s	2625	194	0.19s	5
Net4	9	105	4	290	1.77s	3375	400	5.22s	5
Net5	11	28	8	66	3.02s	4125	130	22.42s	5
Net6	11	65	5	263	4.06s	4125	272	2m 33s	5
Net7	12	47	4	166	5.13s	4500	183	19m 13s	5
Net8	14	37	3	98	7.30s	5250	120	1h 36m 17s	5
Net9	14	188	6	202	10.31s	5250	1276	18h 16m 23s	5
Net10	17	49	7	54	18.94s	6375	141	4h 52m 23s	5
Net11	18	110	10	182	32.78s	6750	358	218h 50m	5
Net12	24	96	16	639	1m 03s	7875	*	*	*
Net13	38	151	5	771	1m 47s	9000	*	*	*

K: Number of work contents generated; n: Number of activities; c_L: Unit delay cost; C: Total cost; $RunT$: Run time; *N. ev.*: Number of project evaluations; $K1$: Number of discretization points; * Solution aborted because the network too big for DPM.

Table 3. Results for selected networks ($K = 100$)

	n	T	c_L	EM (Java)			DPM (Java)		
				C	RunT	N.Ev	C	RunT	K1
Net1	3	16	2	37	0.66s	1125	44	0.02s	5
Net2	5	120	8	321	2.16s	1875	305	0.14s	5
Net3	7	66	5	175	6.14s	2625	194	0.19s	5
Net4	9	105	4	312	14.61s	3375	400	5.22s	5
Net5	11	28	8	122	26.94s	4125	130	22.42s	5
Net6	11	65	5	253	31.57s	4125	272	2m 33s	5
Net7	12	47	4	160	47.77s	4500	183	19m 13s	5
Net8	14	37	3	120	1m 07s	5250	120	1h 36m 17s	5
Net9	14	188	6	810	1m 40s	5250	1276	18h 16m 23s	5
Net10	17	49	7	161	3m 05s	6375	141	4h 52m 23s	5
Net11	18	110	10	386	5m 22s	6750	358	218h 50m	5
Net12	24	96	16	622	10m 25s	7875	*	*	*
Net13	38	151	5	1580	17m 05s	9000	*	*	*

Table 4. Matlab vs. Java ($K = 100$)

	Matlab (EM)	**Java (EM)**
Net1	14.0s	0.7s
Net2	32.4s	2.2s
Net3	1m 6s	6.2s
Net4	1m 48s	14.6s
Net5	2m 18s	27.0s
Net6	2m 42s	32.0s
Net7	3m 30s	47.8s
Net8	4m 12s	1m 07s
Net9	5m 01s	1m 40s
Net10	7m 30s	3m 05s
Net11	9m 42s	5m 22s
Net12	18m 30s	10m 25s
Net13	60m 00s	17m 05s

3.3. Distributed mode results

Now, we review the results for the EM in the distributed mode. Note that these tests are different from those presented in the previous section because they were made using a network of computers. When comparing one algorithm to its distributed version we need to run the algorithm in a single mode version, using one computer that belongs to the network. The network used in the tests was composed of six Intel Pentium IV E 3.0 GHz with 496MB of RAM computers, under Microsoft Windows XP Professional SP2.

In this version, we tested networks 1 to 13 listed above plus network 14 with 49 activities, also presented in Appendix A. The tests demonstrate the advantage of the EM in its distributed version for large networks (#9 through #14): the processing times are significantly shorter and no network was aborted before completion. We used $K = 300$, $K = 600$ and $K = 1200$. The parameter M was fixed, as before, at 15. In this paper we tested only the impact of parameters K and n and added one more parameter to these tests, which is the number of clients (c) present in the network. The results obtained are presented in Table 5 and 6.

As can be seen, the results were obtained in a shorter time by the distributed version of EM algorithm, for the larger networks (Net10 to Net14). Even for the medium size networks, when K is high, the running times for the distributed mode are smaller than for the single mode. These results show the advantage of the distributed mode for the more complex objective functions and for large networks. For the smaller problems, using the distributed mode is not worth the effort due to the time spent on communication.

Table 5. Results for selected networks ($K = 300$, $K = 600$, $K = 1200$, SM, DM $(c = 6)$)

	n	**T**	$\mathbf{c}_L$	K=300	K=600	K=1200	K=300	K=600	K=1200
Net1	3	16	2	2.81s	5.36s	11.44s	3m 27s	3m 02s	2m 45s
Net2	5	120	8	11.70s	23.03s	46.47s	4m 21s	4m 21s	4m 27s
Net3	7	66	5	33.45s	1m 08s	2m 16s	5m 26s	5m 03s	5m 35s
Net4	9	105	4	1m 24s	2m 47s	5m 31s	6m 9s	5m 35s	6m 56s
Net5	11	28	8	2m 35s	5m 04s	10m 17s	7m 01s	7m 55s	7m 55s
Net6	11	65	5	3m 02s	6m 02s	11m 59s	7m 03s	8m 09s	7m 29s
Net7	12	47	4	4m 29s	8m 56s	18m 15s	8m 15s	8m 14s	7m 24s
Net8	14	37	3	6m 28s	12m 49s	25m 51s	9m 23s	9m 17s	8m 45s
Net9	14	188	6	9m 13s	18m 25s	37m 24s	9m 51s	9m 42s	9m 54s
Net10	17	49	7	17m 44s	35m 03s	1h 10m 24s	10m 26s	11m 10s	11m 53s
Net11	18	110	10	29m 59s	1h 23s	2h 1m 15s	11m 54s	11m 59s	12m 05s
Net12	24	96	16	58m 50s	1h 57m 05s	3h 58m 03s	13m 32s	13m 34s	17m 07s
Net13	38	151	5	1h 39m 27s	3h 18m 24s	6h 37m 58s	15m 21s	15m 23s	19m 41s
Net14	49	221	5	24h 30m **	38h 30m **	38h 30m **	1h 01m	1h 16m 59s	2h 06m

SM: single mode; DM:distributed mode; ** Test was aborted.

Table 6. Results for selected networks ($K = 600$, SM, $c = 2$, $c = 4$, $c = 6$)

	n	**T**	$\mathbf{c}_L$	**K**	**SM**	**c=2 DM**	**c=4 DM**	**c=6 DM**
Net1	3	16	2	600	5.36s	2m 45s	2m 38s	3m 02s
Net2	5	120	8	600	23.30s	3m 28s	3m 58s	4m 21s
Net3	7	66	5	600	1m 08s	4m 29s	4m 50s	5m 03s
Net4	9	105	4	600	2m 47s	6m 46s	6m 11s	5m 35s
Net5	11	28	8	600	5m 04s	8m 02s	7m 09s	7m 55s
Net6	11	65	5	600	6m 02s	7m 45s	6m 47s	8m 09s
Net7	12	47	4	600	8m 56s	8m 12s	8m 33s	8m 14s
Net8	14	37	3	600	12m 49s	9m 48s	9m 01s	9m 17s
Net9	14	188	6	600	18m 25s	9m 38s	9m 28s	9m 42s
Net10	17	49	7	600	35m 03s	11m 04s	11m 16s	11m 10s
Net11	18	110	10	600	1h 23s	14m 27s	11m 40s	11m 59s
Net12	24	96	16	600	1h 57m 05s	16m 53s	13m 33s	13m 34s
Net13	38	151	5	600	3h 18m 24s	19m 24s	18m 06s	15m 23s
Net14	49	221	5	600	38h 30m *	2h 30m	1h 45m	1h 16m 59s

In Table 6 one can see the difference of using a network with two, four or six computers, or only one computer. The single mode implementation is better when dealing with the small networks (Net1-Net6). For large networks (Net13-Net14), the use of a network with six computers is worth the effort, compared with the other solutions. For the medium networks, the results are better for the distributed mode implementation, but the use of six computers is not always the better choice.

4. Conclusions and directions for future research

4.1. Conclusions

Our experimental results indicate conclusively that EM is far superior to DPM for large networks: it is capable of solving large problems which are not solvable by DPM, and for those that can be solved by DPM it is much faster.

An intriguing observation is that EM almost always secured better values of the criterion function than DPM which is an *optimizing* method. How can that be? The answer lies in the fact that the solution obtained by DPM is in fact the best solution of a constrained version of the original problem. This is due to the need to discretize both the state and decision spaces since both are continuous. For example, using a 5 point discretization scheme and lower and upper bounds on the permissible allocation of 0.5 and 1.5; respectively, the DPM is in fact restricted to decision values in the set $\{0.5, 0.75, 1.0, 1.25, 1.5\}$. This means that the optimal solution given by the DPM is constrained to this set of values and may not be the "true" optimum. Actually, it constitutes an

experimental upper bound on the "true" optimum. Observe that using the EM yields a solution that may not be the "true" optimum, because we are using an heuristic approach and we limit the size of the population (M) and the number of iterations (to $25n$); hence the asymptotic properties of the EM need not be realized. However, the values tested by the EM belong to a larger set (in fact, the interval between the lower and upper bounds on the resource allocation) and hence do yield better values.

We may thus conclude that the use of EM opens a new chapter in the studies of the problem of resource allocation in activity networks. As to the use of the object oriented programming language Java, we may assert that the computational performance improved significantly and the run time spent on this implementation is much lower than with Matlab, as evidenced in Table 4 and in Fig. 4.

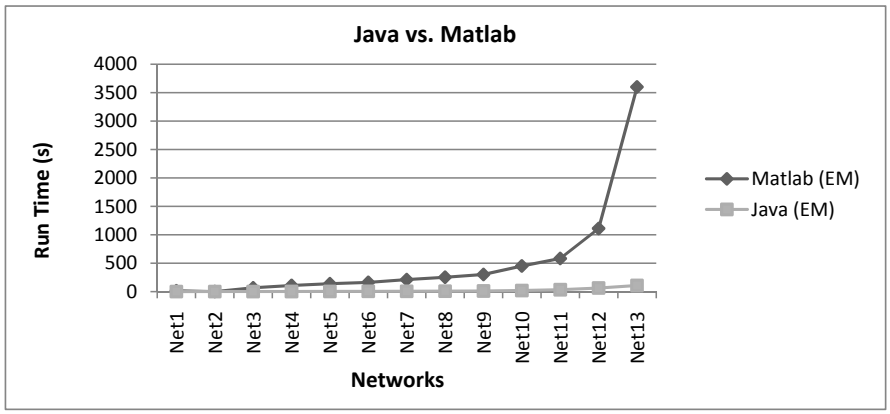

Figure 4. Java vs. Matlab

The difference between both running times is obvious.

The K parameter is one of the most important parameters in this algorithm. It tells how "heavy" an Ion evaluation is. Tables 2 and 3 indicate that at increasing K the run time increases as well. One has to choose carefully the value of parameter K: while a greater K brings precision to the solution, it also brings additional run time. Fig. 5 results from Table 5 and shows the influence of parameter K on the run time.

Fig. 5 indicates that larger samples of work content cause longer running times. The question is how many work content samples we need to find acceptable solution. This question is not covered by this paper, but is to be addressed in future work.

As shown in the tables of Section 3.3, the application running in the distributed mode returns a better run time for large networks. The worse behavior in small networks can be explained by the time consumed by the communica-

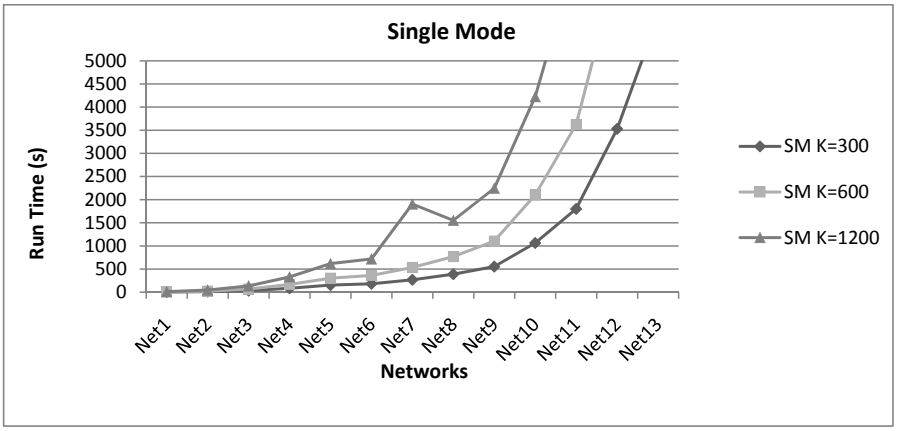

Figure 5. Single Mode, $K = 300$, $K = 600$, $K = 1200$

tion protocol in sending and receiving data. As to large networks, the run time decreases significantly in the distributed mode implementation. In this case we can say that the use of the EM in distributed mode is almost indispensable. Table 6 shows us the results when running with different number of clients. Based on this table, we produced the charts of Fig. 6 and Fig. 7, which show the advantages and disadvantages of running the EM on a computer network.

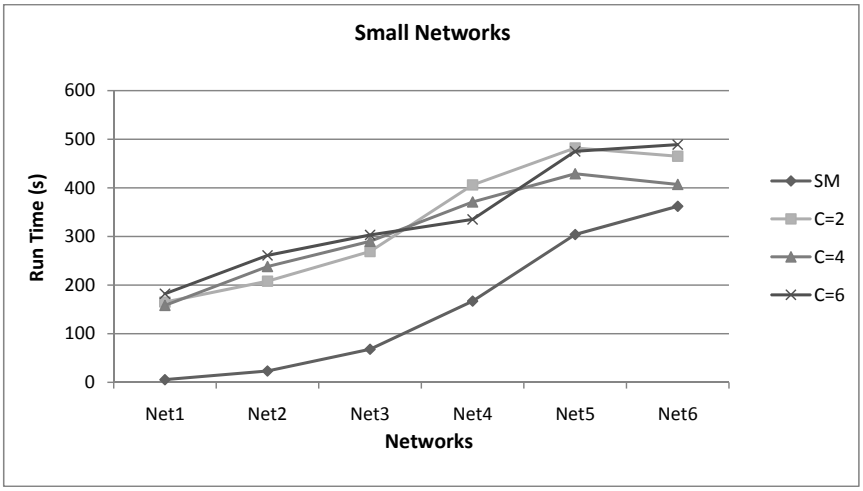

Figure 6. Comparative results for different number of clients (small networks)

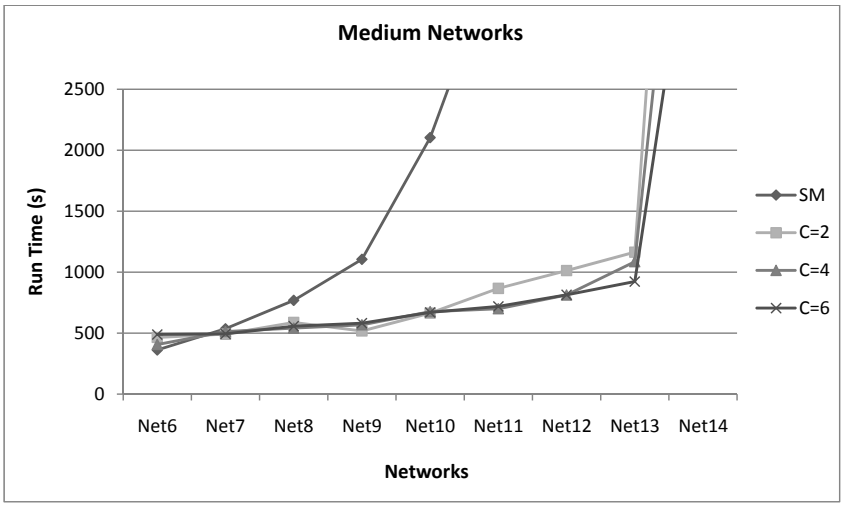

Figure 7. Comparative results for different number of clients (medium networks

Fig. 6 and Fig. 7 demonstrate that the use of the distributed mode is not necessary for small networks but its use in large networks is indispensable. For Net14 the algorithm in its single mode version did not stop in acceptable time, but the distributed mode version obtained one solution in one hour (using $K = 300$).

The decrease in computing time becomes less significant as the number of computers involved in the parallel implementation increases. For instance, in Table 6 there is less (proportionate) gain when the number of computers is increased from 4 to 6, especially for large problems (see also Figs. 6 and 7). This may be explained by the increased time spent on communication when the number of computers involved increases.

In this paper we pay more attention to the running time of the algorithm, than the solution quality, because this is a stochastic problem with random values generated. This leads to different results in different executions. The processes of generating work contents and initial ion's population have a huge random component. It is necessary sometimes to run the EM more than once to find a better solution. So, it is indispensable for the EM to have good running times.

After seeing the results given by Tables 2 and 3, we note that the solutions given by the EM with $K = 100$ were worse than the ones obtained with $K = 10$. This can be explained by the fact that when we choose a low K value the range of solutions given by the algorithm is larger (less precision). The range of solutions given by the algorithm when running with $K = 100$ is generally contained in the range given by the algorithm when running with $K = 10$ (see Fig. 8). The

best solution value will probably be smaller when running with a small K. In Tables 2 and 3, only networks 6, 7 and 12 returned a better result when running with $K = 100$.

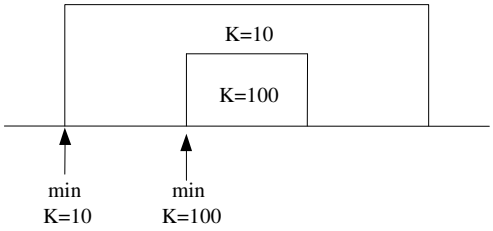

Figure 8. Influence of K on the result

During tests made in the distributed mode, we paid no attention to the performance of computer resources. For instance, the CPU is not running at 100% utilization when, for example, the server is expecting values or the clients are expecting the ions. This and other aspects can be subject to optimization on all machines belonging to the network. One can do this by putting two or more clients on a single machine or by putting one client on the server machine.

4.2. Future research

We would like to try other algorithms, with different philosophies, to solve this problem. So, a possible next step is to try Evolutionary Algorithms. The representation and computation of the project cost will be done in the same way as in the EM, but the strategies to reach the result will be implemented in a different way.

In our future research, we shall do experimentation using other than the exponential distribution, such as the uniform, the beta and the Weibull distributions.

In this research we have dealt with only one resource. We hope to extend this model to have more than one resource associated with each activity, assuming arbitrary distributions of the work content.

Finally, we have always assumed that an activity can start as soon as it is sequence feasible (all its predecessors have completed processing). But there are many instances in which one does not wish to start an activity at the time it is sequence feasible. An excellent example of that would be an activity that is not critical and involves a substantial outlay of resources. In such case, one wishes to postpone its initiation as much as possible. This injects the concept of intentional delays into the whole process. The question then becomes: what is the optimal delay in each activity to maximize the present value of the project? Under such scenario we will have to assume a stream of income and another stream of expenditure, and a discount factor valid over duration of the project.

References

Ahn, T. and Erenguc, S. (1995) Resource constrained project scheduling problem with multiple crashable modes - An exact solution method. *Working Paper Series #95-101*, Department of Decision and Information Sciences, University of Florida, Gainesville, FL.

Ahn, T. and Erenguc, S.S. (1998) The resource constrained project scheduling problem with multiple crashable modes: A heuristic procedure. *EJOR* **107**, 250-259.

Artigues, C., Roubellat, F. and Billaut, J.C. (1999) Characterization of a set of schedules in a resource-constrained multiproject scheduling problem with multiple modes. *Int. J. of Industrial Engineering Theory, Applications and Practice* **6**, 112-122.

Artigues, C. and Roubellat, F. (2000) A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes. *EJOR* **127**, 297-316.

Bellenguez, O. (2004) A multi-skill project scheduling problem. Laboratoire d'informatique de l'université de Tours.

Birbil, S.I. and Fang, S.C. (2003) An Electromagnetism like Mechanism for Global Optimization. *Journal of Global Optimization* **25**, 263-282.

Birbil, S.I., Fang, S.C. and Sheu, R.-S. (2004) On the Convergence of the Electromagnetism Method for Global Optimization. *Journal of Global Optimization*, **30**, 301-318.

Błażewicz, J., Ecker, K., Pesch, E., Schmidt, G. and Węglarz, J. (2007) *Handbook on Scheduling: From Theory to Applications*. Springer-Verlag, Berlin.

Booch, G., Rumbaugh, J. and Jacobson, I. (1998) *The Unified Modeling Language User Guide*. Addison-Wesley.

Bouleimen, K. and Lecocq, H. (2003) A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple modes version. *EJOR* **149**, 268-281.

Buddhakulsomsiri, J. and Kim, D.S. (2006) Properties of multi-mode resource constrained project scheduling problems with resource vacations and activity splitting. *EJOR* **175**, 279-295.

Buddhakulsomsiri, J. and Kim, D.S. (2007) Priority rule-based heuristic for multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. *EJOR* **178**, 374-390.

Dauzere-Peres, S., Roux, J. and Lasserre, J.B. (1998) Multi-resource shop scheduling with resource flexibility. *EJOR* **107**, 289-305.

Demeulemeester, E.L. and Herroelen, W.S. (2002) *Project Scheduling: A Research Handbook*, Kluwer.

Dieter, D., De Reyck, B., Leus, R. and Vanhoucke M. (2006) A hybrid scatter search/electromagnetism metaheuristic for project scheduling. *EJOR* **169**, 638-653.

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

Erenguc, S.S., Ahn, T. and Conway, D.G. (2001) The resource constrained project scheduling problem with multiple crashable modes: An exact solution method. *Naval Research Logistics* **48** 107-127.

Herroelen, W., De Reyck, B. and Demeulemeester, E. (1998) Resource-Constrained Project Scheduling: A survey of recent developments. *Computers and Operations Research* **25**, 279-302.

Janiak, A. and Szkodny, T. (1994) Job-shop scheduling with convex models of operations. *Mathematics of Computation Modelling* **20**, 59-68.

Janiak, A. and Kovalyov, M.Y. (1996) Single machine scheduling subject to deadlines and resource dependent processing times. *EJOR* **94**, 284-291.

Janiak, A. (1998) Minimization of the makespan in a two-machine problem under given resource constraints. *EJOR* **107**, 325-337.

Janiak, A. and Portmann, M.C. (1998) Genetic algorithm for the permutation flow-shop scheduling problem with linear models of operations. *Annals of Operations Research* **83**, 95-114.

Johnson, S.M. (1954) Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly* **1**, 61-68.

Józefowska, J. and Weglarz, J., eds. (2006) *Perspectives in Modern Project Scheduling*. Springer, New York.

Lorenzoni, L.L., Ahonen, H. and De Alvarenga, A.G. (2006) A multimode resource-constrained scheduling problem in the context of port operations. *Computers and Industrial Engineering* **50**, 55-65.

Neumann, K., Schwindt, C. and Zimmermann, J. (2001) *Project Scheduling with Time Windows and Scarce Resources. Lecture Notes in Economics and Mathematical Systems* **508**, Springer, New York.

Novais, R. (2005) *Gestão de Projectos*, Relatório de Estágio da Licenciatura em Engenharia de Sistemas e Informática (in Portuguese). Internal Report, Universidade do Minho, Braga, Portugal.

Penz, B., Rapine, C. and Trystram, D. (2001) Sensitivity analysis of scheduling algorithms. *EJOR* **134**, 606-615.

Sprecher, A. (1994) *Resource-Constrained Project Scheduling: Exact Methods for the Multi-Mode Case*. **LNEMS 409**, Springer-Verlag, Berlin.

Sprecher, A., Hartmann, S. and Drexl, A. (1994) Project scheduling with discrete time-resource and resource-resource tradeoffs. *Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel*, No. 357, Kiel, Germany.

Stork, F. (2001) Stochastic resource-constrained project scheduling. Ph.D. Thesis, TU Berlin.

Tereso, A.P., Araújo, M.M. and Elmaghraby, S.E. (2004) Adaptive Resource Allocation in Multimodal Activity Networks. *Int. J. of Production Economics* **92**, 1-10.

Tereso, A.P. and Araújo, M.M. (2004) The Optimal Allocation in Stochastic Activity Networks via the Electromagnetism Approach. *Proceedings of the Project Management and Scheduling'04*, Nancy – France.

Tereso, A.P., Mota, J.R. and Lameiro, R.J. (2005) Adaptive Resource Allocation Technique to Stochastic Multimodal Projects: a distributed platform implementation in Java. *Control & Cybernetics* **35**, 661-686.

Tsai, Y.W. and Gemmil, D.D. (1998) Using tabu search to schedule activities of stochastic resource-constrained projects. *EJOR*, **111**, 29-141.

Van De Vonder, S., Demeulemeester, E.L. and Herroelen, W.S. (2007) An investigation of efficient and effective predictive-reactive project scheduling procedures. *Journal of Scheduling* **10**, Special Issue on Project Scheduling under Uncertainty, edited by E.L. Demeulemeester and W.S. Herroelen.

Van De Vonder, S., Demeulemeester, E.L., Leus, R. and Herroelen, W.S. (2006) Proactive/reactive project scheduling - Trade-offs and procedures. In: J. Józefowska and J. Węglarz, eds., *Perspectives in Modern Scheduling*, *International Series in Operations Research & Management Science*, **92**.

Appendix A

All networks presented in this appendix are on A-o-A mode of representation.

Network 1

The first network tested is very simple, having only 3 activities. The due date T is 16 and the tardiness penalty c_L is 2 per unit time. The remaining parameters are given in Table A1. They are the origin and target node of each activity, the parameter (λ) of the exponential distribution, representing Work Content of each activity, and the minimal and maximal amount of resource to allocate to each activity. The expected duration of activity 1 is $1/\lambda = 1/0.2 = 5$, and for activities 2 and 3: 10 and 14.29, respectively. In this way, the PERT expected duration for this network is 15. The due date of the project is selected to be a value above the PERT expected duration (approximately by 5% more).

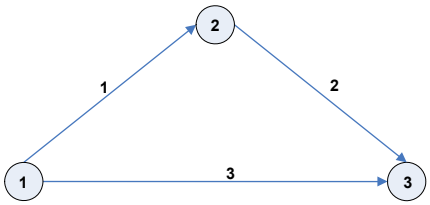

Figure A1. Network 1

Table A1. Parameters for network 1

Activity	**1**	**2**	**3**
Origin	1	2	1
Target	2	3	3
λ	0.2	0.1	0.07
$\mathbf{x}_{\min}$	0.5	0.5	0.5
$\mathbf{x}_{\max}$	1.5	1.5	1.5

Network 2

This network has 5 activities. The due date is $T = 120$ and tardiness cost is $c_L = 8$. Table A2 shows the remaining parameters. The PERT expected duration for this network is 115.

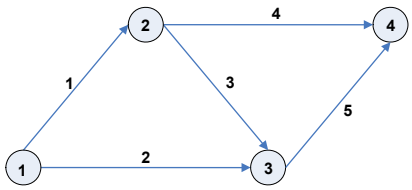

Figure A2. Network 2

Table A2. Parameters for network 2

Activity	**1**	**2**	**3**	**4**	**5**
Origin	1	1	2	2	3
Target	2	3	3	4	4
λ	0.02	0.03	0.04	0.024	0.025
$x_{\min}$	0.5	0.5	0.5	0.5	0.5
$x_{\max}$	1.5	1.5	1.5	1.5	1.5

Network 3

This network has 7 activities. The due date is $T = 66$ and tardiness cost is $c_L = 5$. The remaining parameters are given in Table A3. The PERT expected duration is 62.9.

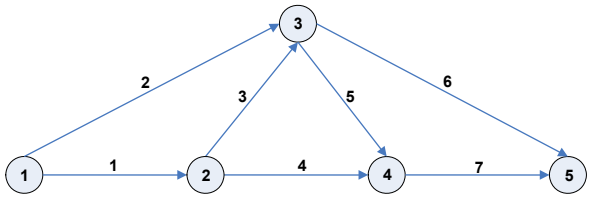

Figure A3. Network 3

Table A3. Parameters for network 3

Activity	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Origin	1	1	2	2	3	3	4
Target	2	3	3	4	4	5	5
λ	0.08	0.06	0.09	0.05	0.07	0.03	0.04
$x_{\min}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$x_{\max}$	1.5	1.5	1.5	1.5	1.5	1.5	1.5

Network 4

This network has 9 activities. For this network, $T = 105$ and $c_L = 4$. Table A4 shows the remaining parameters. The PERT expected duration is 100.

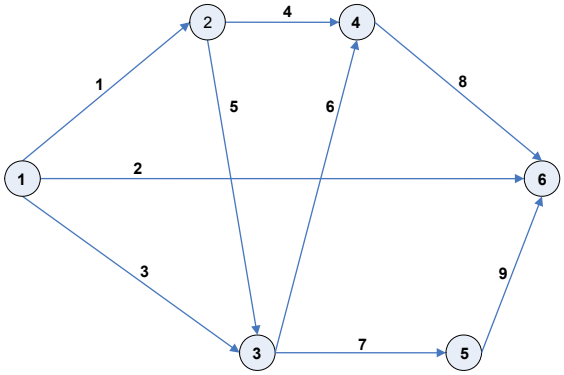

Figure A4. Network 4

Table A4. Parameters for network 4

Activity	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Origin	1	1	1	2	2	3	3	4	5
Target	2	6	3	4	3	4	5	6	6
λ	0.04	0.01	0.07	0.035	0.05	0.06	0.045	0.06	0.039
$\mathbf{x}_{\min}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$\mathbf{x}_{\max}$	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5

4.3. Network 5

Network 5 (Fig. A5) has 11 activities. For this network, $T = 28$ and $c_L = 8$. The remaining parameters are given in Table A5. The PERT expected duration is 26.67.

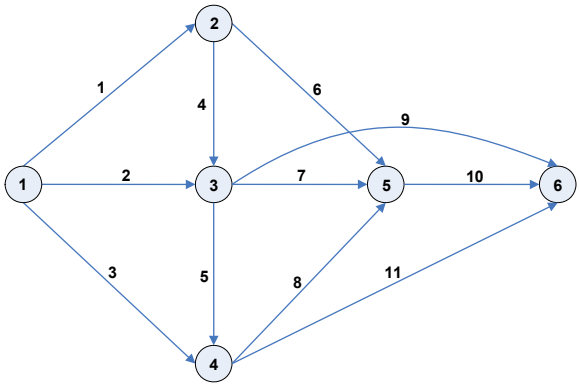

Figure A5. Network 5

Table A5. Parameters for network 5

Activity	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Origin	1	1	1	2	3	2	3	4	3	5	4
Target	2	3	4	3	4	5	5	5	6	6	6
λ	0.1	0.09	0.4	0.2	0.3	0.08	0.4	0.2	0.1	0.3	0.3
$x_{\min}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$x_{\max}$	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5

Network 6

This network has also 11 activities. The due date is $T = 65$ and the unit cost of tardiness is $c_L = 5$. See Table A6 for the rest of information. The PERT expected duration is 62.08.

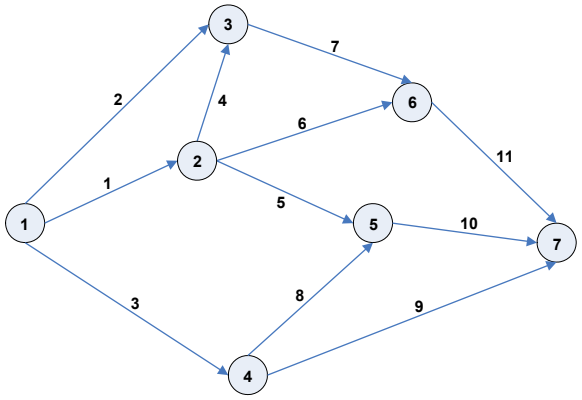

Figure A6. Network 6

Table A6. Parameters for network 6

Activity	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Origin	1	1	1	2	2	2	3	4	4	5	6
Target	2	3	4	3	5	6	6	5	7	7	7
λ	0.1	0.12	0.05	0.08	0.2	0.04	0.03	0.04	0.024	0.15	0.16
$\mathbf{x}_{\min}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$\mathbf{x}_{\max}$	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5

Network 7

Network 7 has 12 activities (Fig. A7), with a different topology. It has $T = 47$ and $c_L = 4$. The remaining parameters are shown in Table A7. The PERT expected duration is 44.72.

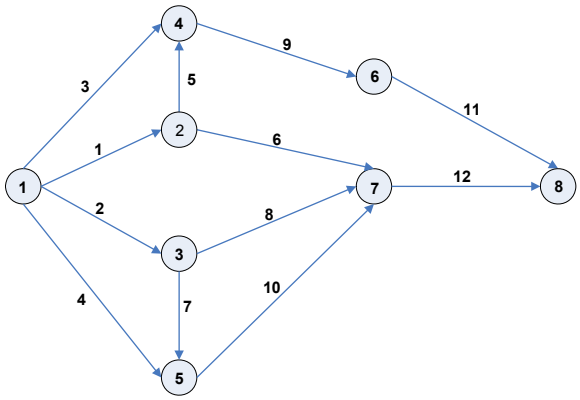

Figure A7. Network 7

Table A7. Parameters for network 7

Activity	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Origin	1	1	1	1	2	2	3	3	4	5	6	7
Target	2	3	4	5	4	7	5	7	6	7	8	8
λ	0.1	0.09	0.08	0.1	0.09	0.08	0.1	0.09	0.08	0.1	0.09	0.1
$\mathbf{x}_{\min}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$\mathbf{x}_{\max}$	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5

Network 8

This network has 14 activities. T is 37 and c_L is 3. The remaining parameters are presented in Table A8. The PERT expected duration is 35.5.

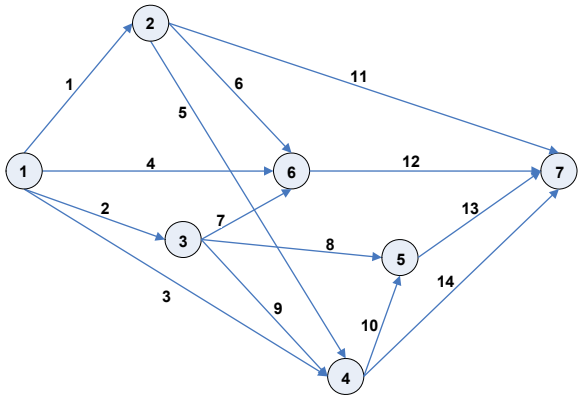

Figure A8. Network 8

Table A8. Parameters for network 8

Activity	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Origin	1	1	1	1	2	2	3	3	3	4	2	6	5	4
Target	2	3	4	6	4	6	6	5	4	5	7	7	7	7
λ	0.2	0.25	0.16	0.2	0.1	0.16	0.5	0.25	0.2	0.08	0.09	0.1	0.125	0.1
$\mathbf{x}_{\min}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$\mathbf{x}_{\max}$	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5

Network 9

Network 9 has the same number of activities as the previous one (14). Its due date is 188 and tardiness cost is 6. Other parameters are given in Table A9. The PERT expected duration is 178.57.

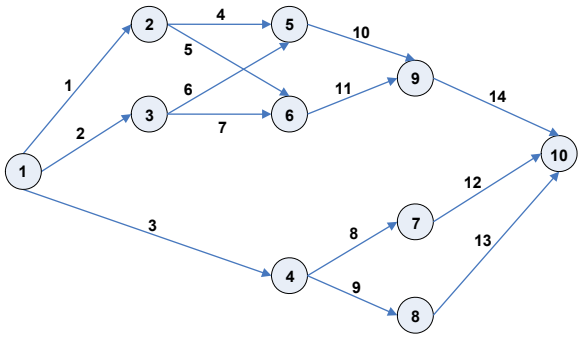

Figure A9. Network 9

Table A9. Parameters for network 9

Activity	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Origin	1	1	1	2	2	3	3
Target	2	3	4	5	6	5	6
λ	0.02	0.03	0.04	0.025	0.035	0.045	0.05
$\mathbf{x}_{\min}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$\mathbf{x}_{\max}$	1.5	1.5	1.5	1.5	1.5	1.5	1.5
Activity	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Origin	4	4	5	6	7	8	9
Target	7	8	9	9	10	10	10
λ	0.06	0.03	0.02	0.015	0.02	0.025	0.03
$\mathbf{x}_{\min}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$\mathbf{x}_{\max}$	1.5	1.5	1.5	1.5	1.5	1.5	1.5

Network 10

Network 10 has 17 activities. For this network, $T = 49$ and $c_L = 7$. The remaining parameters are shown in Table A10. The PERT expected duration is 44.98.

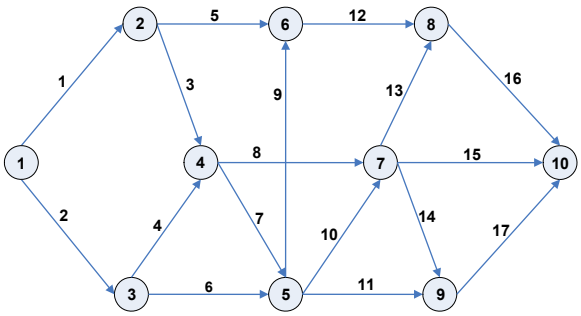

Figure A10. Network 10

Table A10. Parameters for network 10

Activity	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Origin	1	1	2	3	2	3	4	4	5
Target	2	3	4	4	6	5	5	7	6
λ	0.167	0.1	0.2	0.1	0.25	0.2	0.1	0.333	0.333
$\mathbf{x}_{\min}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$\mathbf{x}_{\max}$	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
Activity	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	
Origin	5	5	6	7	7	7	8	9	
Target	7	9	8	8	9	10	10	10	
λ	0.25	0.5	0.167	0.143	0.5	0.125	0.167	0.11	
$\mathbf{x}_{\min}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	
$\mathbf{x}_{\max}$	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	

Network 11

Network 11 (Fig. A11), has 18 activities. Here, $T = 110$ and $c_L = 10$. Other parameters are shown in Table A11. The PERT expected duration is 106.11.

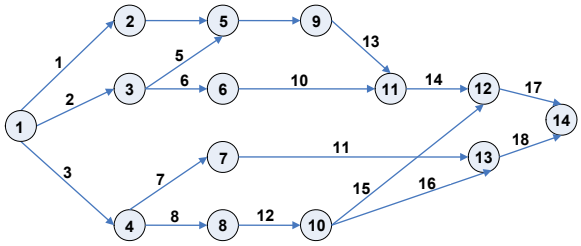

Figure A11. Network 11

Table A11. Parameters for network 11

Activity	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Origin	1	1	1	2	3	3	4	4	5
Target	2	3	4	5	5	6	7	8	9
λ	0.06	0.04	0.1	0.07	0.08	0.04	0.08	0.2	0.07
$\mathbf{x}_{\min}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$\mathbf{x}_{\max}$	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
Activity	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Origin	6	7	8	9	11	10	10	12	13
Target	11	13	10	11	12	12	13	14	14
λ	0.05	0.08	0.07	0.09	0.09	0.05	0.09	0.04	0.06
$\mathbf{x}_{\min}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$\mathbf{x}_{\max}$	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5

Network 12

Network 12, Fig. A12, is much bigger than the previous (24 activities). Here, $T = 223$ and $c_L = 12$. Other parameters are given in Table A12. The PERT expected duration is 212.05.

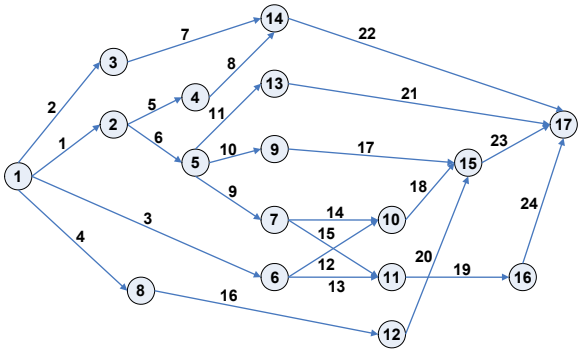

Figure A12. Network 12

Table A12. Parameters for network 12

Activity	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Origin	1	1	1	1	2	2	3	4
Target	2	3	6	8	4	5	14	14
λ	0.025	0.02	0.0222	0.0182	0.0333	0.0286	0.025	0.04
$\mathbf{x_{min}}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$\mathbf{x_{max}}$	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
Activity	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Origin	5	5	5	6	6	7	7	8
Target	7	9	13	10	11	10	11	12
λ	0.0333	0.025	0.02	0.0385	0.0357	0.05	0.04	0.0333
$\mathbf{x_{min}}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$\mathbf{x_{max}}$	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
Activity	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
Origin	9	10	11	12	13	14	15	16
Target	15	15	16	15	17	17	17	17
λ	0.05	0.0556	0.0278	0.0222	0.0667	0.0167	0.04	0.0217
$\mathbf{x_{min}}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$\mathbf{x_{max}}$	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5

Network 13

Network 13 (Fig. A13), is even bigger than the previous ones (38 activities). For this network, $T = 151$ and $c_L = 5$. The remaining parameters are presented in Table A13. The PERT expected duration is 143.99.

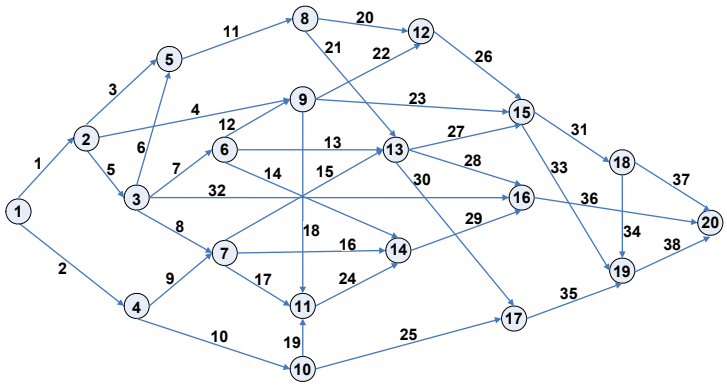

Figure A13. Network 13

Table A13. Parameters for network 13

Activity	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Origin	1	1	2	2	2	3	3	3	4	4
Target	2	4	5	9	3	5	6	7	7	10
λ	0.0625	0.0333	0.0625	0.05	0.1	0.0556	0.0667	0.0909	0.0833	0.025
$\mathbf{x_{min}}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$\mathbf{x_{max}}$	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
Activity	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Origin	5	6	6	6	7	7	7	9	10	8
Target	8	9	13	14	13	14	11	11	11	12
λ	0.05	0.0625	0.1	0.125	0.7143	0.0476	0.0625	0.05	0.0526	0.0588
$\mathbf{x_{min}}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$\mathbf{x_{max}}$	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
Activity	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	textbf
Origin	8	9	9	11	10	12	13	13	14	
Target	13	12	18	14	17	15	15	16	16	
λ	0.0667	0.0555	0.0476	0.0625	0.05	0.0555	0.0625	0.0833	0.0555	
$\mathbf{x_{min}}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	
$\mathbf{x_{max}}$	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	
Activity	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	
Origin	13	15	3	15	18	17	16	18	19	
Target	17	18	16	19	19	19	20	20	20	
λ	0.0417	0.0909	0.0909	0.0455	0.0417	0.0333	0.0667	0.125	0.1	
$\mathbf{x_{min}}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	
$\mathbf{x_{max}}$	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	

Network 14

Network 14 (Fig. A14) was the biggest network tested (49 activities). For this network, $T = 221$ and $c_L = 5$. The remaining parameters are shown in Table A14.

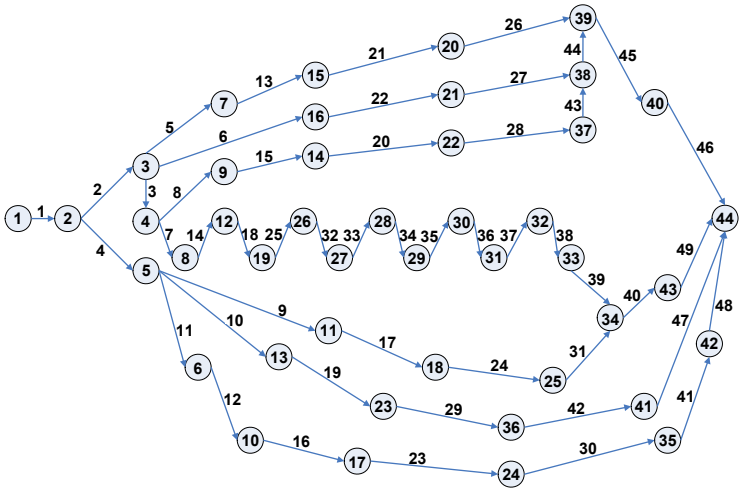

Figure A14. Network 14

Table A14. Parameters for network 14

Activity	1	2	3	4	5	6	7	8	9	10
Origin	1	2	3	2	3	3	4	4	5	5
Target	2	3	4	5	7	16	8	9	11	13
λ	0.01	0.125	0.25	0.16	0.11	0.125	0.11	0.2	0.16	0.11
x_{min}	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
x_{max}	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
Activity	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Origin	5	6	7	8	9	10	11	12	13	14
Target	6	10	15	12	14	17	18	19	23	22
λ	0.14	0.5	0.33	0.2	0.3	0.4	0.5	0.33	0.1	0.44
x_{min}	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
x_{max}	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
Activity	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
Origin	15	16	17	18	19	20	21	22	23	24
Target	20	21	24	25	26	39	38	37	36	35
λ	0.14	0.12	0.33	0.2	0.11	0.13	0.14	0.5	0.2	0.16
x_{min}	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
x_{max}	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
Activity	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**
Origin	25	26	27	28	29	30	31	32	33	34
Target	34	27	28	29	30	31	32	33	34	43
λ	0.14	0.09	0.14	0.14	0.09	0.14	0.5	0.33	0.33	0.08
x_{min}	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
x_{max}	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
Activity	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	
Origin	35	36	37	38	39	40	41	42	43	
Target	42	41	38	39	40	44	44	44	44	
λ	0.13	0.09	0.33	0.13	0.5	0.15	0.5	0.14	0.13	
x_{min}	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	
x_{max}	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	