北京

International Conference
on Industrial Engineering and Systems Management
IESM 2007
May 30 - June 2, 2007
BEIJING - CHINA

二〇〇七

The Optimal Resource Allocation in Stochastic Activity Networks via the Evolutionary Approach: A Platform Implementation in Java*

Anabela P. Tereso, Lino A. Costa, Rui A. Novais, Madalena T. Araújo

University of Minho
Production and Systems Department
4800-058 Guimarães
PORTUGAL

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 Algorithm had also been adapted and implemented, with better results than the Dynamic Programming Model. This paper presents another philosophy for solving the same problem, based on an Evolutionary Algorithm. This approach was implemented using an Object Oriented language, Java, and its results were compared to the Electromagnetism Algorithm. A distributed version was also developed, to be run in a computer network, in order to take advantage of available computational resources.

Key words: Project Management and Scheduling, Stochastic Activity Networks, Resource Allocation, Evolutionary Algorithms, Electromagnetism Algorithm.

1 Introduction

1.1 Problem Definition

The problem treated in this paper may be stated as follows. Given an activity-on-arc network [7] that defines a project, we wish to find the resource allocation that optimizes the total cost. Each activity *a* has stochastic work content W_a, assumed to be exponentially distributed with a parameter θ, which may be different for different activities. The total project cost is the sum of two costs: (*i*) the resource cost (*RC*), which is proportional to the square of resource usage for the duration of the activity, with constant of proportionality equal to c_R, and (*ii*) the

* This paper was not presented at any other revue. Corresponding author: Anabela P. Tereso. Email address: anabelat@dps.uminho.pt (Anabela P. Tereso), lac@dps.uminho.pt (Lino A. Costa), rui.fafe@gmail.com (Rui A. Novais), mmaraujo@dps.uminho.pt (Madalena T. Araújo)

"tardiness cost" which is proportional to the amount of tardiness from a specified due date *T*, with constant of proportionality equal to c_L, representing the marginal cost per period. The duration of an activity *a*, denoted by Y_a, depends on the work content and the amount of resource allocated to the activity; $Y_a = W_a / x_a$ in which x_a is the intensity of resource allocation, restricted with lower and upper bounds $x_a \in [l_a, u_a]$ with $0 \leq l_a \leq u_a < \infty$. There is only one resource of unlimited availability so that it does not impose any limitations on 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 (see [10] for more details).

In chapter 1 of this paper, after defining the problem (section 1.1) we present a brief introduction to the Evolutionary Algorithm (EVA) (section 1.2). In section 1.3 we present the model description. In chapter 2, we describe the rationale for the application development, namely the reason for choosing the Java programming language (section 2.1) including some implementation details and the data structures used (section 2.2). In section 2.3, we present the Evolutionary Algorithm in a distributed mode, in order to take advantage of paralleled programming. We conclude the paper presenting the results of comparing the Electromagnetism Algorithm (EMA) [2,3,11,14] with the Evolutionary Algorithm (chapter 3), drawing some conclusions and pointing some future directions of research (chapter 4).

1.2 The Evolutionary Approach

The Evolutionary Algorithms are frequently used to solve computational and optimization problems. These algorithms are particular suited to optimization problems that, due to their nonlinear and constrained nature, are multimodal and nonconvex. Therefore, if these problems are solved using standard local optimization methods, it is possible that the solution obtained will be of local nature. Thus, it is important to use global optimization techniques in order to obtain an approximation to the global optimum. Evolutionary algorithms mimic the natural evolution of the species in biological systems and they can be used as a robust global optimization tool. In past years, several distinct approaches have emerged namely, Evolution Strategies (ESs) and Genetic Algorithms. In this work, we consider an Evolutionary Algorithm based on Evolution Strategies. These algorithms do not require any continuity or convexity property of the problem being solved. Moreover, unlike conventional algorithms, only information regarding the objective function and constraints is required to perform the search.

ESs, in the past, proved to be powerful single objective optimizers. Moreover, several studies indicate that ESs are, usually, more efficient than genetic algorithms in terms of number of objective function evaluations, specially, with continuous optimization problems [1,5,6,9].

They start searching from an initial population (a set of points) and transition rules between generations are deterministic. Thus, two populations are maintained: one parent population of size μ and one offspring population of size λ. The individuals of the populations are vectors of real coded decision variables that are potential optimal solutions. Each generation, λ offspring are generated from μ progenitors by mutation and recombination. The best individuals are then selected for next generation. Important features of evolution strategies are the self adaptation of step sizes for mutation during the search and the recombination of individuals that is performed between ρ individuals. In general, in ESs nomenclature, this algorithm is referred as $(\mu/\rho+\lambda)$-ES or $(\mu/\rho,\lambda)$-ES according to type of selection used. In $(\mu/\rho+\lambda)$-ES (figure 1), selection is performed among the μ parents and λ offspring. On the other hand, in $(\mu/\rho,\lambda)$-ES (figure 2), selection takes into account only the λ offspring generated. Thus, selection pressure, in $(\mu/\rho+\lambda)$-ES, is inferior to selection pressure of $(\mu/\rho,\lambda)$-ES.

Basically, the recombination operator consists on, before mutation, to recombine a set of chosen parents to generate a new solution. A given number ρ $(1 \leq \rho \leq \mu)$ of parents are randomly chosen for recombination. When $\rho = 1$ then there is no recombination. Two types of recombination are, mainly, considered: intermediate and discrete recombination. Since, in this work, the recombination implemented was the discrete recombination, only this recombination will be described in detail. In the discrete recombination, each component of the

offspring is chosen from one of the ρ parents at random. Thus, for ρ chosen parents (randomly selected from population), the offspring x_p is given by

$$x_p = (x_{u_1,1}, \ldots, x_{u_n,n})$$

with $u_1, \ldots, u_n$ generated uniformly at random, i.e., $u_1 \sim U(1,\rho), \ldots, u_n \sim U(1,\rho)$ and $p = 1, \ldots, \mu$. In discrete recombination, the integer uniform random values $u_1, \ldots, u_n$ allow the selection of which of the ρ parents will give the value of decision variable i. This procedure allows different combinations of the values of the decision variables from existing solutions in the population. Standard deviations are similarly recombined.

During the search, the step sizes for mutation are adapted before performing mutation. Several self-adaptation schemes are possible. One possibility is to actualize the standard deviations σ_i (for each decision variable) according to the equation:

$$\sigma_i^{(k+1)} = \sigma_i^{(k)} e^{z_i} e^{z}$$

where $z_i \sim N(0, \Delta\sigma_A{}^2)$, $z \sim N(0, \Delta\sigma_B{}^2)$ and $\Delta\sigma_A{}^2$ and $\Delta\sigma_B{}^2$ are parameters of the algorithm. In all experiments conducted only this non-isotropic adaptation rule was considered.

After recombination, mutation is applied to all μ individuals, generating λ ($\lambda > \mu$) offspring, i.e., $x_m = x_p + z$. Usually, the random numbers z are generated according to a Gaussian or Normal distribution. Besides, it is convenient that small changes occur frequently, but large ones only rarely. So, two requirements arise together for the generation of the random numbers z:

the expected value of the components z_i must be equal to zero, i.e., $E(z_i) = 0$ for $i = 1, \ldots, n$ and

the variances σ_i^2 must be small, for $i = 1, \ldots, n$.

In this sense, the random numbers z_i can be generated according to a Normal distribution with mean zero and variance σ_i^2:

$$z_i \sim N(0, \sigma_i^2)$$

So, mutation consists on adding random numbers with mean zero and variance σ_i^2 to the vector of decision variables.

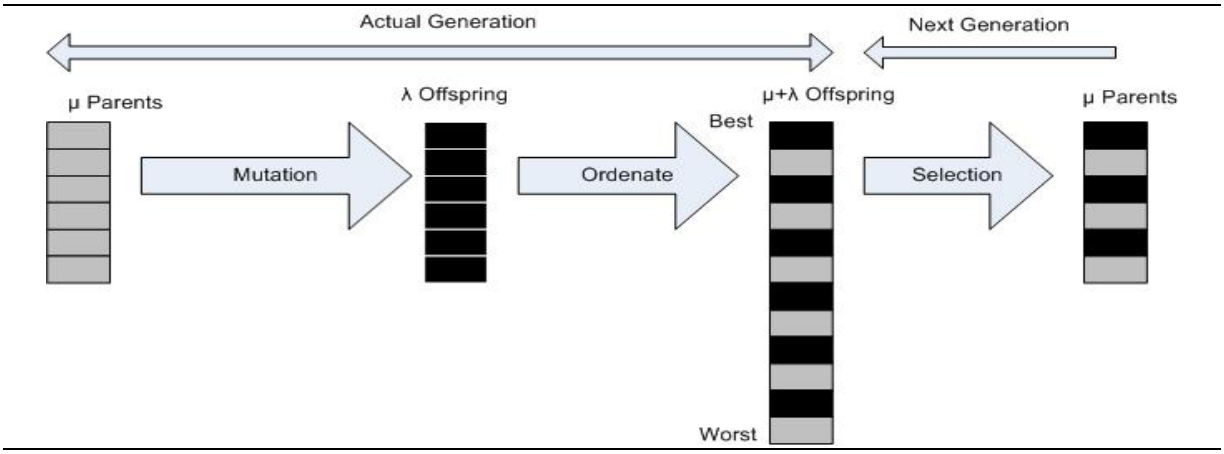

Fig. 1. (μ+λ) nomenclature

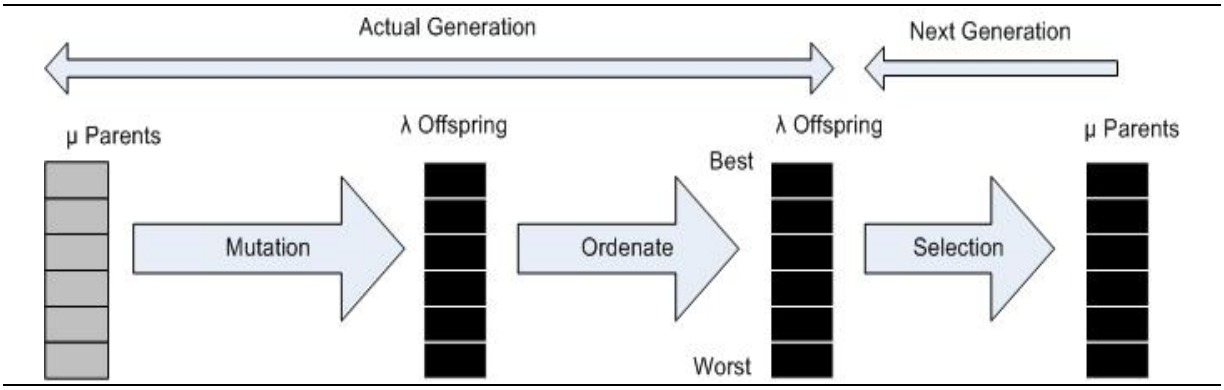

Fig. 2. (μ,λ) nomenclature

1.3 Model Description

The application of the EVA to our problem was done in the following way: each individual is a vector containing real variables corresponding to a potential optimal solution of the problem. These variables are the amount of resource allocated to each activity of the project network. The goal of EVA adapted to our problem is to minimize the project cost associated with each particle (assignment of resources) in the population by changing that assignment. One has to be very careful in setting the algorithm parameters. For instance, if we choose a large number of particles, the algorithm will need more time to find the best solution, but there is a higher probability that the best solution will be lower in cost, because we tried a larger number of solutions. So, one may assert that there are a few parameters that influence the algorithm's run time. They are:

Number of sampled work contents (k) – The EVA 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.

Parent Population size (popsiezpar) – A large number of parents will result in testing a large number of solutions in each iteration.

Offspring Population size (popsizeoff) – A large number of offspring will result in testing a large number of solutions in each iteration.

Number of network activities (n) – A project with a large number of activities leads to a larger and more complex network. The EVA will take more time finding or calculating the CPM (Critical Path Method) value (used to calculate the total cost).

So, given an activity network, for each activity *a*, there is an associated work content (W_a). This work content is generated by a negative exponential distribution with parameter θ. We consider the existence of a single resource, with x_a being the amount of resource allocated to activity *a*, such that

$$0 \le l_a \le x_a \le u_a < \infty, \qquad (1)$$

where l_a and u_a are, respectively, the lower and upper bound of the resource allocated to the activity. When we allocate a quantity x_a of resource *a* to an activity, the resource cost is assumed to be proportional to the square of the intensity of the resource allocation for the duration of the activity, i.e.:

$$RC_a = \mathrm{c_R} \times x_a{}^2 \times Y_a \ , \qquad (2)$$

in which c_R is the constant of proportionality. As the duration of the activity is given by

$$Y_a = \frac{W_a}{x_a} \ , \qquad (3)$$

we end up with the resource cost being linear in the work content,

$$RC_a = c_{Ra} \times x_a \times W_a \, . \qquad (4)$$

Since W_a is a random variable, RC_a and Y_a will be random variables too. Each project has a due date (T) and may have a Tardiness Cost (TC). There is a penalty constant (c_L) that represents the cost per unity past the due date. The objective is to find the amount of resource allocation to each activity in order to minimize the total cost (C). The total cost is equal to:

$$C = \sum_{a=1}^{n} RC_a + TC \ . \qquad (5)$$

Where *TC* is the tardiness cost given by

$$TC = c_L \times \max(0, t_n - T) \, , \qquad (6)$$

in which, t_n is the time of realization of the last node of the project (project completion), determined as the maximum of the project completion time secured by the CPM calculations [15,16] using the durations given by any realization of the random variable $\{Y_a\}$. 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.

2 Application development

In this section, we discuss the main issues that arose during the application development using Java.

2.1 The choice of a Programming Language

We selected Java as the programming language because we wanted to create an application GP2006 [13] that uses the EMA, the Dynamic Programming Model and the EVA with equal facility. The GP2006 application allows us to experiment with the different approaches and compare the results, weighting the "pros and cons" of each. The Java language has also some computational advantages which will be explained in the next section.

2.2 Data Structures and Input Parameters

To represent the "list of activities", we defined three Classes in Java:

- **Node** – to represent each node of the graph with information about immediately preceding and immediately succeeding nodes and the activities that connects to the node;
- **Activity** – To represent one activity with information about the parameter θ, the lower and upper bounds on the resource allocation; and
- **Network** – which contains a list of activities and nodes

The class **Individual** was defined to represent the solutions to our problem. This class holds a number of attributes, namely attributes to store the Project Cost (solution cost), a vector containing a set of standard deviations used by the algorithm and the particles coordinates (the resource allocation to each activity of the project network).

The class **Project Cost** is used to calculate the cost of the project associated with one realization of the work content for each activity. It uses the CPM and the work content generated at the beginning of the algorithm to calculate the resource cost and the delay cost that together constitute the project cost.

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 of recombination and mutation.

Finally we have another class named **Configuration**, which holds all the parameters used by the algorithm. These parameters are the parent population size, offspring population size, the number of activities presented in the network, the due date, the penalty cost and the number of work contents to be generated.

One of the most important considerations when we use Java is the speed of accessing the data structures. In our case we used two data structures called *Vector* and *HashMap*. The operations of *search, remove, add* and *travel* are very fast when we use *HashMaps*. The sort operation is quicker when we use a *Vector.* The process of decoding and execution of a code written in Java is very efficient. We used two *Vectors* and one *HashMap* to support the algorithm. The two *Vectors* hold the population of parents and offspring (feasible solutions to the problem). The *HashMap* holds the work content necessary to calculate the Project Cost associated with each particle.

Figure 3 represents diagrammatically the various classes defined. More information on these classes and on the developed code is presented in [8].

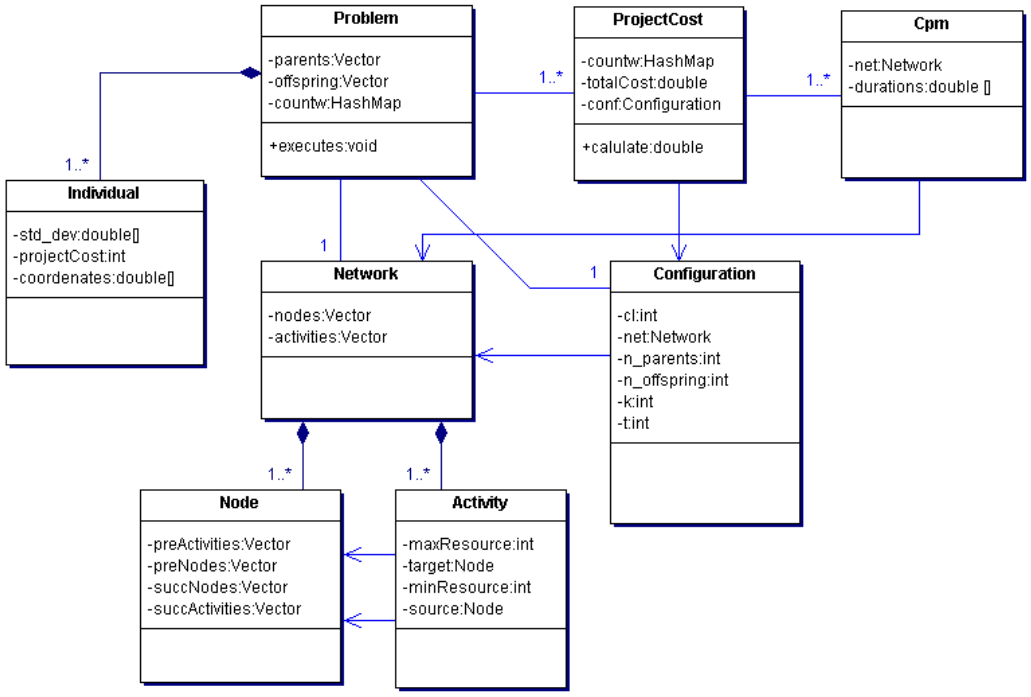

Fig. 3. Class diagram

This diagram is written in the Unified Model Language (UML) and represents a class diagram. The UML is a graphical language used to specify, build and visualize Object Oriented (OO) information systems. Class diagrams are the most common diagrams found 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 [4]).

2.3 Distributed implementation of the Evolutionary Algorithm

In this section we take the next step using this algorithm. The distributed implementation consists of the following: we have a machine called server that begins by creating the problem configuration, setting up the parameters, namely, work contents, population size, activity network and number of clients. The server then sends this configuration to all the clients in the network. In this step the server also sends the work contents necessary for the clients to calculate the project cost associated with each individual belonging to the population. Before sending the problem configuration, the server must also set up all data communication channels with the clients. The clients send one message telling if the configuration and work contents have been well received. The server now sends all individuals one by one to each client in order to calculate the associated project cost. The clients, after calculating the project cost value, return the solution to the server, which will evaluate the mean of all the results, associating the value to the individual´s project cost. This process is repeated for all individuals and all iterations of the algorithm. After collecting all population values, the server sends a "kill" signal to all clients telling them to shutdown. After that, the server stores the best solution and terminates itself. We use sockets and data streams to perform the communication between server and clients. The server creates a communication socket over an IP (Internet Protocol) address. To connect to the server, all clients must have this address in a configuration file that is set up manually. The server knows how many clients will perform the connection by looking at its own configuration file

3. Results

We are now going to compare the EVA with the EMA. In order to fully understand the advantages and disadvantages of each algorithm, we needed to take some measures. These two optimization algorithms work over the same problem (objective function), so to measure the time and quality of the solution returned they must have the same input. So, we tested the two algorithms with the same work contents. The performance measures used were the time and the number of project evaluations. We also compared the distributed version of these algorithms in order to realize their impact on larger networks. To perform the tests we used Pentium IV 3Ghz computers with 512 MB of memory.

3.1 Experimental layout

We used a set of 14 networks with a number of activities ranging from 3 to 76. The networks chosen allowed us to analyze a spectrum with different network complexities. These networks had also been previously tested [14] allowing us to compare performances and results. The parameter T (due date) was selected using CPM and the duration of the longest path, assuming the mean work content and the quantity of resource x_a equal to 1. This leads the duration of each activity to be $Y = \frac{\overline{W}}{1}$ which is equal to $\overline{W}$. T was selected to be slightly larger than the longest path (in the CPM calculations). c_L was set up in order to allow for some tardiness cost if the quantity of resources used is low.

Table 1 shows the characteristics of each network tested.

Table 1 – Characteristics of each network

Network	Number of activities	T	c_L
1	3	16	2
2	5	120	8
3	7	66	5
4	9	105	4
5	11	28	8
6	11	65	5
7	12	47	4
8	14	37	3
9	14	188	6
10	17	49	7
11	18	110	10
12	24	223	12
13	38	151	5
14	76	121	4

3.2 Parameters used

The parameters used by the EMA and the EVA can be seen on table 2. The characteristics of all networks tested can be obtained upon e-mail, through the corresponding author.

Table 2 – Parameters used

Parameter	EMA	EVA
m	15	-----
popsizepar	------	15
popsizeoff	------	15
poprecomb	------	15
delta	0.05	-----
pertpar	0.25	-----
localiterations	1	-----

The meaning of the EMA parameters is the following:

- *m* is the number of sample points used in the algorithm;
- *delta* is a parameter used to perform local search, and it must be between 0 and 1;
- *pertpar* is a perturbation parameter used by the algorithm;
- *localiterations* is the maximal number of local search iterations allowed.

The meaning of the EVA parameters is the following:

- *popsizepar* is the number of individuals in parent population;
- *popsizeoff* is the number of individuals in offspring population;
- *poprecomb* is the number of individuals of parent population that participate in recombination;

More information can be found in [2,3,5,6].

3.3 Single Mode results

In the next table we can see the results given by the two algorithms. We fixed the number of evaluations at 50,000 and register the resource allocation that minimizes the total cost.

Table 3 – Results for 50,000 evaluations, k=10 (EMA and EVA)

Network	Total Cost (EMA)	Run Time (EMA)	Total Cost (EVA)	Run Time (EVA)
1	24.39	4.64s	24.24	3.41s
2	492.00	8.25s	491.53	7.00s
3	200.10	14.52s	206.35	13.42s
4	395.62	24.75s	392.26	23.68s
5	132.76	33.94s	132.80	32.89s
6	473.90	40.14s	458.68	39.28s
7	327.18	53.36s	326.09	52.21s
8	122.32	58.95s	119.03	58.20s
9	242.18	1m 26s	242.91	1m 25s
10	134.47	2m 02s	128.38	2m 01s
11	238.87	3m 14s	232.42	3m 14s
12	110.89	6m 42s	71.66	6m 46s
13	996.74	18m 45s	1028.89	18m 57s
14	526.53	2h 11m 24s	518.86	2h 13m 19s

We also measured the optimization levels achieved by the two algorithms. The optimization level is evaluated by

$$Optim.Level = 1 - (\frac{BestOF}{BestIP}), \qquad (7)$$

where $BestOF$ is the best value achieved by the objective function and $BestIP$ is the best solution chosen from the initial population.

In table 4 we can see the number of evaluations of the objective function which show us the speed achieved by the algorithms engine. We fixed the algorithms run time at 150 seconds and register the number of function evaluations made by each algorithm.

Table 4 – No. Evaluations (Time = 150 seconds, k=10)

Network	EMA	EVA	Network	EMA	EVA
1	2,222,356	3,612,175	8	214,256	221,340
2	1,334,885	1,764,640	9	142,514	145,860
3	811,970	923,315	10	102,543	106,260
4	487,511	531,325	11	64,758	62,800
5	370,623	391,170	12	31,590	32,340
6	293,973	313,290	13	11,476	11,900
7	228,148	244,900	14	1,786	2,540

3.4 Distributed Mode Results

For the distributed mode, the results obtained are the same as in the single mode, but the run times are different. On the next tables we show the results of the distributed mode. Since the goal is to decrease the run time of bigger networks, we only show the results for these networks. For the smallest networks, the single mode returns better results than the distributed mode. The values are very high because we used a heavy objective function (k=600) in order to fully understand the difference between the Single Mode (SM) and the Distributed mode (DM).

Table 5 – Distributed Mode (EMA, k=600)

Network	Run Time (SM)	Run Time (DM, Cli[1]=4)	Run Time (DM, Cli=6)
10	2h 12m 03s	7h 59m 06s	6h 10m 50s
11	3h 28m 32s	8h 01m 48s	6h 13m 10s
12	4h 00m 29s	8h 09m 40s	6h 18m 55s
13	20h 36m 02s	10h 20m 07s	7h 19m 10s
14	27h 56m 40s	23h 30m 48s	18h 12m 02s

Table 6 – Distributed Mode (EVA, k=600)

Network	Run Time (SM)	Run Time (DM, Cli=4)	Run Time (DM, Cli=6)
10	2h 19m 12s	8h 14m 24s	6h 22m 48s
11	3h 32m 24s	8h 15m 00s	6h 23m 28s
12	4h 08m 13s	8h 17m 51s	6h 25m 20s
13	21h 18m 04s	10h 25m 32s	7h 24m 36s
14	28 h 10m 02s	24h 31m 38s	18h 59m 48s

4. Conclusions and Directions for Future Research

4.1 Conclusions

Results Obtained: As we can see from table 3, the results are not exactly the same because the search methods are different, but they are very similar. The two algorithms have reached approximately the same cost in 29% of the cases (difference less then 1). EVA reached a smaller value 57% of the cases and EMA 14% of the cases. We can say that EVA is slightly better than the EMA when concerning the cost value obtained.

Optimization Values: The EMA and the EVA are very similar when we look at the optimization levels. Figure 4 and table 7 shows us the average optimization levels for all networks considered. This average optimization levels are the mean values of the optimization levels achieved for all networks.

[1] Number of Clients

Table 7 – Average Optimization Levels

No. Evaluations	Average EMA Optimization Level	Average EVA Optimization Level
1,000	0.075	0.048
2,000	0.082	0.070
4,000	0.111	0.090
10,000	0.111	0.111
50,000	0.117	0.148

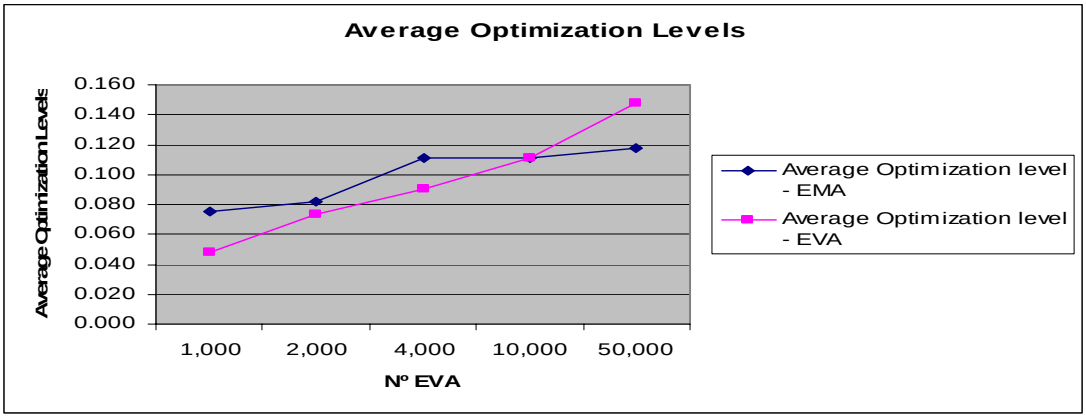

Fig. 4. Optimization Values: K=10, No. EVA=50,000

Note that, for a small number of evaluations the EMA returns better results. At 10,000 evaluations the algorithms return similar values. For 50,000 evaluations the EVA gives better results. We may conclude that when increasing the number of evaluations the optimization levels increase as well. If the number of evaluations is higher, the EVA returns better results. Figure 5 shows us the performance achieved by the two algorithms over different number of evaluations.

Algorithms speed engine: We may conclude by taking a look at table 4 and figure 5 that the speed engine of these two algorithms is very similar. The number of project evaluations done by the EVA is slightly higher than the EMA. This happens because the EMA spends extra time calculating the forces acting on each solution. As the number of activities increases, the number of project evaluations decreases because the network becomes heavier.

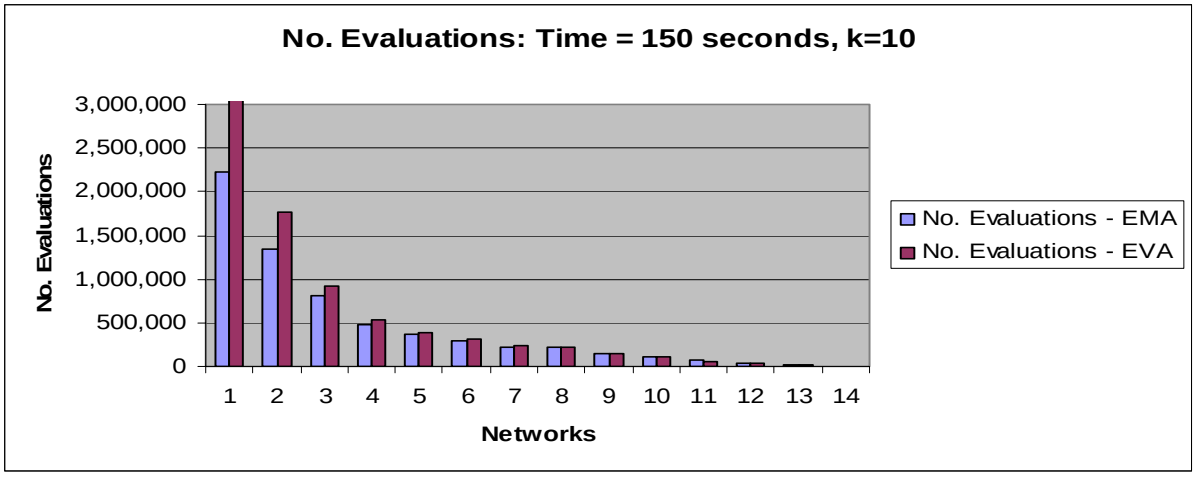

Fig. 5. No. Evaluations, Time = 150 seconds, k = 10

Single Mode vs. Distributed Mode: As we can see by the tables shown in section 3.4, the distributed mode returns a better run time for larger networks. The worst behavior in smaller networks can be explained, because the communication protocol used takes time sending and receiving data. For the last two networks tested, the run time decreases significantly in the distributed mode. In this case we can say that the use of the distributed mode is indispensable when dealing with large networks, as we can see from figure 6 and 7. We may also say that if we are going to run the algorithms in this mode, its better to use a higher number of clients. The figures show us that running with a high number of clients decreases the run time spent.

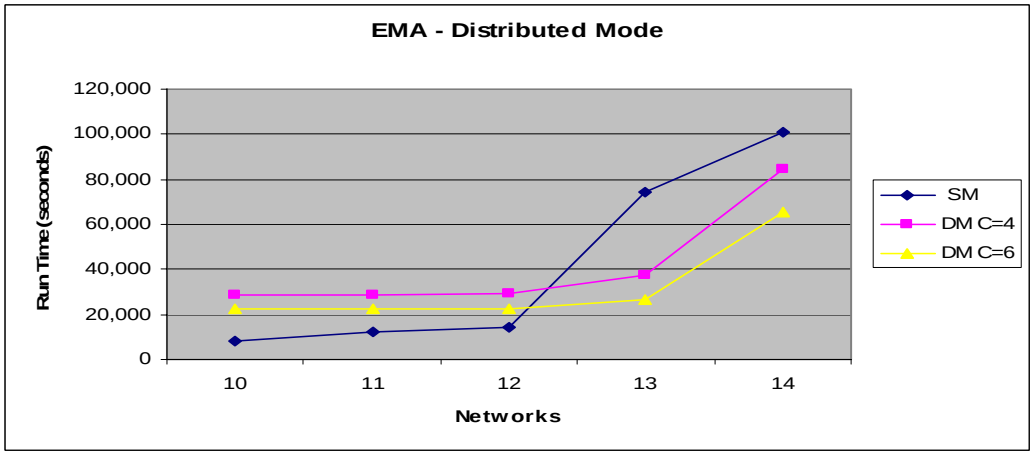

Fig. 6. EMA – Distributed Mode

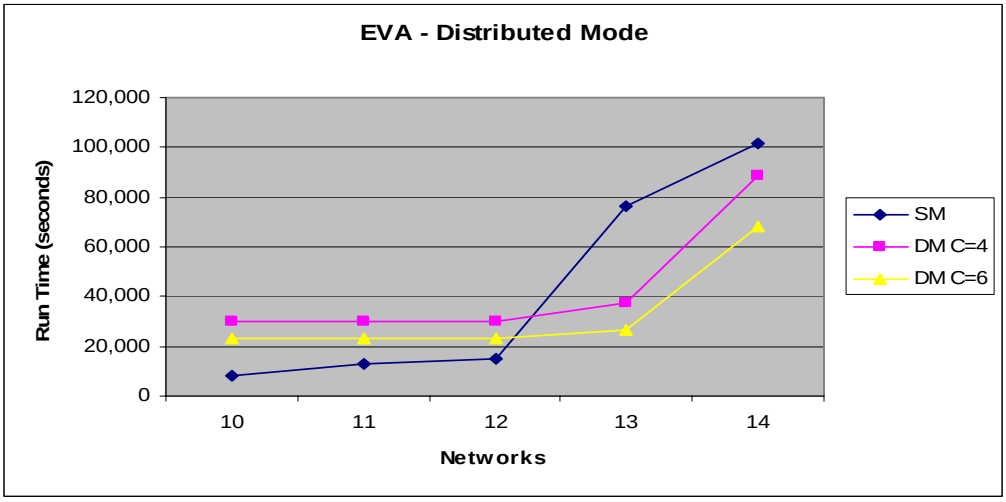

Fig. 7. EVA – Distributed Mode

Comparing the EMA with the EVA in its distributed mode, we may say that the EVA is a little slower than the EMA.

Parameter k influence: The k parameter is one of the most important parameters presented in these algorithms. It tells how heavy a function evaluation is. When increasing the k parameter, the run time increases as well. We have to be very careful when choosing the k parameter. If a greater k brings precision to the solution, it also brings additional run time.

Computer resources: During the tests made in the distributed mode, we paid no attention to the performance of computer resources. For instance the computer's CPU (Central Processing Unit) is not running at 100% when, for example, the server is expecting values or the clients are expecting the individuals. This and other optimizations can be performed to all machines that belong to the network. We can do this by putting two or more clients in a single machine or putting one client in the server's machine. We also can adjust and tuning each machine in the network.

EMA vs. EVA: After running all the tests we may conclude that the EMA and the EVA are very similar in solving this particular problem. The EMA produces better results when running with a low number of evaluations and the EVA produces better results when running with a high number of evaluations. As the EVA does a higher number of evaluations than the EMA, for the same running time, we may conclude that the EVA engine is better. Running in the distributed mode, the EMA produces better results. Compared to the performance of the Dynamic Programming Model [10,12] we may conclude that these algorithms are very good in solving this particular problem, and the results obtained encourage us continue using this algorithms in future research.

4.2 Directions and Future Research

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, still assuming the exponential distribution. 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 money. 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 minimize 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 that is valid throughout the duration of the project.

5 References

[1] Bäck, T. (1996). "The Evolutionary Algorithms in Theory and Pratice", Oxford University Press, Inc., |NY|.

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

[3] 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.

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

[5] Costa, L. A. (2003) "Algoritmos Evolucionários em Optimização Uni objectivo e Multiobjectivo" (in Portuguese). Thesis (Ph.D.), Universidade do Minho, Portugal.

[6] Costa, L. A. and Oliveira P. (2001). "Evolutionary Algorithms Approach to the Solution of Mixed Integer Non-Linear Programming Problems", Computers chem. Engng., 25, 257-266.

[7] Elmaghraby, S.E. (1977). "Activity Networks – project planning and control by network models", John Wiley and Sons, Inc., New York.

[8] 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.

[9] Schewefel, H.-P. (1995). "Evolution and Optimum Seeking", Wiley, New York.

[10] Tereso, A. P., Araújo, M.M. and Elmaghraby, S.E. (2004) "Adaptive Resource Allocation in Multimodal Activity Networks", International Journal of Production Economics, 92:1-10, 2004.

[11] Tereso, A. P., 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.

[12] 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", Dynamic Programming Special Issue of the Journal of Control and Cybernetics, June 2005.

[13] Tereso, A. P., Novais, R. (2006). "GP2006", Software Application being developed. Report in preparation, Universidade do Minho, Guimarães, Portugal.

[14] Tereso, A. P., Novais, R. and Araújo M. M. (2006) "The Optimal Resource Allocation in Stochastic Activity Networks via the Electromagnetism Approach: A Platform Implementation in Java", Universidade do Minho, Guimarães, Portugal.

[15] Kelley J. E. (1960). "Critical-path Planning and Scheduling: Mathematical Basis", Mauchly Associates Inc, Ambler, Pennsylvania.

[16] Kelley J. E. (1963). "The critical path method: Resource planning and scheduling", in J. F. Muth, G. L. Thompson (Eds), Industrial scheduling, Prentice-Hall, Englewood Cliffs, NJ, PP 347-365.