An efficient evolutionary algorithm for the deadline problem in project management

Matías Galnares and Sergio Nesmachnow

Universidad de la República, Uruguay
{mgalnares,sergion}@fing.edu.uy

Abstract. This article presents an efficient evolutionary algorithm applied to the deadline scheduling in project management, a NP-hard problem with major relevance in software engineering and scheduling activities. The evolutionary algorithm has been specifically designed to provide accurate and efficient solutions, by using operators that allow realistic problem instances to be solved. Efficient numerical results are reported in the experimental analysis performed on standard problem instances. The experimental results demonstrate that the proposed evolutionary algorithm is able to outperform one of the best well-known deterministic techniques for the problem in reduced execution times, specially on highly complex instances.

1 Introduction

In general, project management involves planning and organizing a set of activities in order to generate a product or offer a service in the best possible way [11]. A project duration can often be reduced by accelerating some of its activities by employing additional resources that increase the cost of the entire project. In this case, each activity can be performed by using a set of alternatives (*modes*) which are defined by a time-cost pair. Usually, only a reduced number of modes are taken into account for each activity. A key problem consists in finding a schedule that assigns modes to activities, providing a good tradeoff between the duration and cost of each activity, enabling the best project performance. In this article, the scheduling problem addressed is the Deadline Problem in Project Management (DPPM), which accounts for both precedence between activities and deadline for its execution. In the related literature, it is also known as the Discrete Time/Cost Trade-off Problem (DTCTP).

Traditional scheduling problems are NP-hard [8], thus classic exact methods are only useful for solving problem instances of reduced size. Heuristics and metaheuristics are promising methods for solving scheduling problems, since they are able to get efficient solutions in reasonable time, even for large problem instances. Evolutionary algorithms (EAs) have emerged as flexible and robust metaheuristic methods for solving this kind of complex problems, achieving the high level of accuracy and efficiency also shown in many other application areas [3].

The main contributions of this manuscript are: i) to introduce an efficient EA to solve the DPPM, implemented to compute results for realistic instances in reasonable execution times, and ii) to efficiently compute accurate results, which outperform previous results in literature, for a set of problem instances with tight deadline constraints.

Overall, the proposed EA was able to compute **ten new best solutions** for the set of 36 problem instances tackled.

The manuscript is structured as follows. The next section describes the paradigm of evolutionary computation. The DPPM formulation is introduced in Section 3. Section 4 reviews previous works on EAs applied to solve the DPPM and related variants. The implementation details of the proposed EA are described in Section 5. The experimental analysis and the discussion of the results are presented in Section 6, while the conclusions and main lines for future work are formulated in Section 7.

2 Evolutionary algorithms

EAs are non-deterministic methods that emulate the evolutionary process of species in nature, in order to solve optimization, search, and other related problems [3]. In the last twenty-five years, EAs have been successfully applied for solving optimization problems underlying many real applications of high complexity.

The generic schema of an EA is shown in Algorithm 1. An EA is an iterative technique (each iteration is called a *generation*) that applies stochastic operators on a pool of individuals (the population P) in order to improve their *fitness*, which is a measure related to the objective function. Every individual in the population is the encoded version of a solution for the problem. The initial population is generated by a random method or by using a specific heuristic for the problem. An evaluation function associates a fitness value to every individual, indicating its suitability to the problem. Iteratively, the probabilistic application of *variation operators* like the *recombination* of parts from two individuals or random changes (*mutations*) are guided by a selection-of-the-best technique to tentative solutions of higher quality.

The stopping criterion usually involves a fixed number of generations or execution time, a quality threshold on the best fitness value, or the detection of a stagnation situation. Specific policies are used to select the groups of individuals to recombine (the *selection* method) and to determine which new individuals are inserted in the population in each new generation (the *replacement* criterion). The EA returns the best solution ever found in the iterative process, taking into account the fitness function.

Algorithm 1 Schema of an evolutionary algorithm.

```
 1: initialize(P(0))
 2: generation ← 0
 3: while not stopcriteria do
 4:    evaluate(P(generation))
 5:    parents ← selection(P(generation))
 6:    offspring ← variation operators(parents)
 7:    newpop ← replacement(offspring, P(generation))
 8:    generation ++
 9:    P(generation) ← newpop
10: end while
11: return best solution ever found
```

3 Deadline Problem in Project Management

The DPPM formulation considers the following elements:

- Every project has a set of activities $A = \{a_1, a_2, \ldots, a_N\}$. Since some activities may require the completion of some other activities before they begin, a *precedence function P* is defined, where P_i is the set of immediate predecessors of activity a_i.
- A set of execution modes $M_i = \{m_{i1}, m_{i2}, \ldots, m_{iR_i}\}$ is defined for each activity a_i, where each activity must be assigned to exactly one mode.
- For each activity a_i and each mode m_{ik}, the time/cost pair (t_{ik}, c_{ik}) is defined, where t_{ik} is the duration and c_{ik} is the cost. For any two modes m_{ik_1} and m_{ik_2} of a given activity a_i, $t_{ik_1} < t_{ik_2}$ implies $c_{ik_1} > c_{ik_2}$, which means that in order to speed up the time of a given activity additional resources are needed, i.e. higher costs are demanded. In addition, $k_1 < k_2$ implies $t_{ik_1} > t_{ik_2}$ for all activities a_i, that is, the activity modes are ordered by decreasing order of duration.
- S_i denotes the starting time of activity a_i.
- A deadline T for the project duration is established.

The goal of the DPPM is to find a *schedule*, i.e. a function $f : M^R \rightarrow A^N$ that assigns modes to the activities, which minimizes the total cost while fulfilling the precedence constraints and subject to that the entire project duration cannot exceed the deadline T.

For the DPPM mathematical formulation, lets consider two dummy activities, a_0 which precedes all those real activities with no predecessors, and a_{N+1}, which is performed after all activities having no successors are finished (thus, S_{N+1} is the entire project duration), and the binary decision variables y_{ik}, whose values are given by Eq. 1.

$$y_{ik} = \begin{cases} 1, & \text{if activity } a_i \text{ is assigned to mode } m_{ik} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

So, the DPPM formulation as optimization problem is presented in Eq. 2.

(2)

$$\begin{aligned}
\text{minimize} \quad & \textstyle\sum_{i=1}^{N} \sum_{k=1}^{R_i} c_{ik} \times y_{ik} && (2.1)\\
\text{subject to} \quad & \textstyle\sum_{k=1}^{R_i} y_{ik} = 1,\ i = 1 \ldots N && (2.2)\\
& S_i \geq S_j + \textstyle\sum_{k=1}^{R_j} t_{jk} \times y_{jk},\ j \in P_i,\ i = 1 \ldots N+1 && (2.3)\\
& S_{N+1} \leq T && (2.4)\\
& S_i \geq 0,\ i = 1 \ldots N+1 && (2.5)\\
& y_{ik} \in \{0,1\} i = 1 \ldots N,\ k = 1 \ldots R_i && (2.6)
\end{aligned}$$

The objective of the DPPM is the minimization of the total cost (2.1). The constraints of the problem are: each activity must be assigned to exactly one mode (2.2); an activity cannot start before all its immediate predecessors are completed (2.3); the entire project duration cannot exceed the deadline T (2.4); the starting time of the activities must be non-negative (2.5); and the values of y_{ik} are binary (2.6).

4 Related work: heuristics and metaheuristics for the DPPM

This subsection presents a brief review of previous works that have proposed applying heuristics and metaheuristics to the DPPM and related variants of the problem.

Pioneering works in scheduling demonstrated that when the time/cost functions are linear, the DPPM can be solved by traditional methods such as Maximum Flow or Cut Search algorithms. Dunne et al. [6] proved that all versions of the DTCTP are NP hard in the strong sense. They also show that some special structures like pure parallel and pure series are solvable in polynomial times.

Demeulemeester et al. [5] solved the Time/Cost Curve Problem by applying a horizon-varying approach using iterative solutions of the DPPM, computed with a Branch and Bound (BAB) algorithm using Linear Relaxation based lower bounds (LB). The proposed approach solved small-sized instances up to 30 activities and four modes easily, but failed to solve most of the tackled instances with 40 activities. Deineko et al. [4] proved that there cannot exist a polynomial time approximation algorithm with a performance guarantee better than 3/2 for any versions of the DTCTP.

Akkan et al. [1] computed LB for the DPPM using column generation techniques based on a network decomposition approach. The proposed techniques are also applied to construct feasible solutions. An extensive computational study revealed the satisfactory behavior of the algorithm, which obtained solutions with average gap less than 7% in only 6 seconds.

Hafizoglu and Azizoglu [9] introduced algorithms based on linear programming relaxation to solve the DPPM, and defined two LB on the optimal total cost, namely Naive Bound and LPR-Based LB. These LB were used in their BAB algorithm to define the branching strategy and to eliminate non-promising partial solutions. The BAB method solved instances with up to 150 activities and 10 modes in reasonable execution times, showing a satisfactory behavior for loose deadline time constraints. However, when faced with tighter constraints, the execution time of BAB increased considerably, reaching an hour of computing time. Up to our knowledge, the BAB algorithm is one of the best method for solving the DPPM, although it demands a large computing time for instances with tight deadlines.

Anagnostopoulos et al. [2] developed five variants of a simulated annealing algorithm for solving the problem. Although these variants differ on the number of iterations in each cycle and on the stopping criterion, all of them achieve feasible solutions in a few seconds, and all methods are able to solve large-sized instances up to 300 activities with 4 modes. However, the quality of the computed solutions are only evaluated with estimations of the global optima within a certain confidence interval.

Hazir et al. [10] proposed an effective exact algorithm to solve the time minimization version of the DTCTP by decomposing it in to two simpler subproblems. The *master problem* (MP) solves a relaxed version of DTCTP and generates trial values for the integer variables and a LB for minimization. The *subproblem* is the original problem with the values of the integer variables fixed by the MP. Instances from [1] with 85 to 136 activities, 2 to 10 modes, and tight deadline constraints are solved. According to the results, 74% of these tight instances could be solved exactly in 10 minutes, 96% in an hour and all the instances are solved in 90 minutes. Up to our knowledge, this is the best method for solving the time minimization DTCTP with tight deadlines.

Zhang et al. [13] extended the DTCTP by considering renewable and non renewable resource-constrains simultaneously. A genetic algorithm was implemented for this problem, which is able to solve instance with two renewable and two nonrenewable resources and with up to 30 activities and three modes. The computation experiments show that the algorithm solved these instances in no more than four seconds.

Recently, Fallah-Mehdipour et al. [7] included a new parameter, the quality of the project, to previously considered time and cost parameters. Two EAs are used to solve the proposed problem, namely Multi-objective particle swarm optimization (MOPSO) algorithm and Non-dominated sorting genetic algorithm (NSGA-II). To evaluate both algorithms, two different instances are considered with two objectives and 18 activities, and three objectives and 7 activities are solved. The results show that both algorithms are able to achieve feasible solutions, but no computing time are reported.

Summarizing, the analysis of the related works shows that solving DPPM/DTCTP instances with tight deadline constraints in reduced execution times is a hard task. Thus, there is still room to contribute in this line of research, by developing efficient and accurate methods to solve the DPPM/DTCTP, able to handle the increasing complexity of realistic instances with tight constraints in reduced execution times.

5 An efficient evolutionary algorithm for the DPPM

This section introduces the implementation details of the proposed EA, designed to compute accurate solutions in reduced time and to provide a good exploration pattern, by using ad-hoc variation operators.

The GAlib library. The algorithm was implemented on GAlib, a library for EAs developed in C/C++ using the object oriented paradigm [12]. The library includes tools to implement EAs and offers the possibility of developing user-defined representations and operators.

Solution encoding. In canonical GAs, bits are used to represent a solution. However, in the proposed EA a more complex encoding is defined to consider the precedence relations between activities and the different modes in which each activity can be performed. Each individual in the population is encoded as an array $I = (I_0^0, I_1^a, ..., I_i^k, ..., I_N^b, I_{N+1}^0)$ where I_i represents the activity a_i, and I_i^k denotes the activity a_i in mode m_{ik}. The execution order of activities is from left to right: all of the predecessors of activity I_i are located before—i.e. at the left—of I_i. Fig. 1 shows an example of solution encoding for a given set of activities and precedence relations modeled according to a directed acyclic graph.

Fitness function. Let $sol = (I_0^0, I_1^{a_1}, ..., I_i^{a_i}, ..., I_N^{a_N}, I_{N+1}^0)$ be a solution. The fitness function is given by Eq. 3, where C is the maximum cost of the project, i.e. the sum of all activity costs assuming that all activities are in the costliest mode (the last mode).

$$fitness(sol) = C - \sum_{i=1}^{i=N} cost(I_i^{a_i}) \tag{3}$$

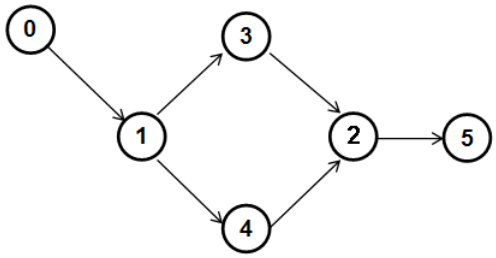


Fig. 1. Proposed encoding for DPPM solutions.

Feasibility check and repair mechanism. The operators used in the proposed EA can generate non-feasible solutions that do not fulfill the deadline constraints. Thus, a feasibility check/repair method is applied to correct non-feasible solutions. The total time of the project cannot exceed the deadline T. Therefore, any individual $(I_0^0, I_1^{a_1}, ..., I_i^{a_i}, ..., I_N^{a_N}, I_{N+1}^0)$ must fulfill that $time(I_1^{a_0}) + ... + time(I_i^{a_i}) + ... + time(I_N^{a_N}) \leq T$. When this condition is not verified, the feasibility repair mechanism works by randomly selecting an activity and changing its mode in order to demand a shorter execution time. This procedure is iteratively applied until the deadline constraint is met.

Initialization. An initial population of feasible solutions is generated by applying an ad-hoc randomized construction operator. Starting from $I = (I_0^0, \emptyset, ...\emptyset, ...\emptyset)$, a new activity I_i in mode m_{ik} is randomly selected. If all the predecessors of activity I_i have already been inserted in the solution, then I_i^k is inserted in the first available empty location. Otherwise, another activity is selected. When the whole schedule has been constructed, the feasibility check/repair mechanism is applied in order to meet the deadline constraints.

Selection. The standard proportional selection method (*roulette wheel*) is used. An individual i is selected to recombine/mutate with probability P_{SEL}, given by Eq. 4, where $popSize$ is the size of the population in the EA.

$$P_{SEL}(i) = \frac{fitness(i)}{\sum_{j=1}^{popSize} fitness(j)} \tag{4}$$

Exploitation: recombination. An ad-hoc single point crossover operator [3] is used to recombine solutions in order to preserve the precedence relations between activities. Two parents $A = (A_0^0, A_1^a, ..., A_{r-1}^b, A_r^c, ..., A_{N+1}^0)$, and $B = (B_0^0, B_1^d, ..., B_{r-1}^e, B_r^f, ..., B_{N+1}^0)$ are selected from the population, using the selection operator, and an integer r is selected randomly from the interval $[1, N]$. Then, from A and B two offspring C and D are generated according to r: $C = (\overline{A_0^0}, \overline{A_1^{a'}}, ..., \overline{A_{r-1}^{b'}}, A_r^c, ..., A_{N+1}^0)$ where $\overline{A_0^0}, \overline{A_1^{a'}}, ..., \overline{A_{r-1}^{b'}}$ are the elements $A_0^0, A_1^a, ..., A_{r-1}^b$ in A, but using the modes in B,and arranged in the order they are in B, and $D = (\overline{B_0^0}, \overline{B_1^{d'}}, ..., \overline{B_{r-1}^{e'}}, B_r^f, ..., B_{N+1}^0)$ where $\overline{B_0^0}, \overline{B_1^{d'}}, ..., \overline{B_{r-1}^{e'}}$ are

the elements $B_0^0, B_1^d, ..., B_{r-1}^e$ in B, but using the modes in A, and arranged in the order they are in A. An example of recombination is presented in Fig. 2(a).

The crossover operator can generate solutions that exceed the maximum time allowed, so the feasibility check/repair is used to assure that all constraints are met.

Exploration: mutation. Let $A = (A_0^0, ..., A_p^*, ..., A_r^m, ..., A_s^*, ..., A_{N+1}^0)$ be the individual that is selected for mutation, where A_r^* denotes the activity A_r in any (fixed) mode. An integer r is selected at random from the interval $[1, N]$. Let activity A_p be the last predecessor of activity A_r, and let activity A_s be the first successor of activity A_r. An integer d is selected randomly from the interval $[p+1, s-1]$, then for $d \leq r \rightarrow A' = (A_0^0, ..., A_p^*, A_r^{m'}, A_{p+1}^*, ..., A_{r-1}^*, A_{r+1}^*, ..., A_{N+1}^0)$, and for $d > r \rightarrow A' = (A_0^0, ..., A_{r-1}^*, A_{r+1}^*, ..., A_{s-1}^*, A_r^{m'}, A_s^*, ..., A_{N+1}^0)$.

The mutation operator maintains the precedence relations between activities, but the new mode is randomly selected, so the feasibility check/repair is applied to guarantee that the deadline constraint is met. An example of mutation is presented in Fig. 2(b).

Local search. The local search operator attempts to improve a given solution $A = (A_0^0, A_1^a, ..., A_r^k, ..., A_N^b, A_{N+1}^0)$ by evaluating to change the activities to a mode that reduces the total cost. Starting on a randomly selected activity a_r represented by A_r, the operator cyclically analyzes all the N activities, attempting to change it to mode $m_{r(k-1)}$, assuming that the activity is assigned to execute in mode m_{rk}. The process is iteratively applied until all activities have been analyzed and no change has been applied, because making such changes will imply to violate the deadline constraint. Thus, a new schedule $A' = (A_0^0, A_1^{a'}, ..., A_r^{k'}, ..., A_N^{b'}, A_{N+1}^0)$ is generated.

The local search operator maintains the precedence relations between activities and it also guarantees that the deadline constraint is not violated.

This LS is applied to all new individuals obtained from recombination and mutation operators in each generation. In addition, LS is a time-consuming operator and has an important influence in EA performance.

$r = 3$

$A = (0^{a_0}, 4^{a_1}, 1^{a_2}, 2^{a_3}, 3^{a_4}, 5^{a_5}, 6^{a_6})$

$B = (0^{b_0}, 1^{b_1}, 2^{b_2}, 4^{b_3}, 5^{b_4}, 3^{b_5}, 6^{b_6})$

$C = (0^{b_0}, 1^{b_1}, 4^{b_3}, 2^{a_3}, 3^{a_4}, 5^{a_5}, 6^{a_6})$

$D = (0^{a_0}, 1^{a_2}, 2^{a_3}, 4^{b_3}, 5^{b_4}, 3^{b_5}, 6^{b_6})$

(a) Crossover.

$M = (0^{m_0}, 4^{m_1}, 1^{m_2}, 2^{m_3}, 5^{m_4}, 3^{m_5}, 6^{m_6})$

$r = 4 \rightarrow A_r = 5$

$A_p = 4 \rightarrow p = 1$

$A_s = 6 \rightarrow s = 6$

$random(2,5) \rightarrow d = 3$

$M' = (0^{m_0}, 4^{m_1}, 5^{m'_5}, 1^{m_2}, 2^{m_3}, 3^{m_4}, 6^{m_6})$

(b) Mutation.

Fig. 2. Example of the crossover and mutation operators

6 Experimental analysis

This section introduces the set of DPPM instances and the platform used in the experimental evaluation. After that, the parameter setting experiments are commented. The last subsection presents and discusses the numerical results of the proposed EA.

6.1 DPPM instances

Thirty-six instances in the benchmark from [1] are used to evaluate the proposed EA. We selected the most complex instances, regarding two metrics: Coefficient of Network Complexity (CNC), the ratio between number of activities and number of modes, and Complexity Index (CI), that evaluates how close is an instance to a series-parallel one.

In the selected instances, the number of modes and the durations of each activity are chosen using a discrete uniform distribution in $[1,10]$ and $[3,123]$, respectively. The minimum cost $c_{i,1}$ is uniformly selected in [5,15] for all i, and $c_{i,k+1}$ is defined recursively by $c_{i,k+1} = c_{i,k} + \alpha(t_{i,k} - t_{i,k+1})$, where α is also taken from a uniform distribution. The deadline values are defined by $T = T_{min} + \theta(T_{max} - T_{min})$, being T_{min} and T_{max} the shortest and longest project durations respectively, and $\theta \in \{0.15, 0.30, 0.45\}$.

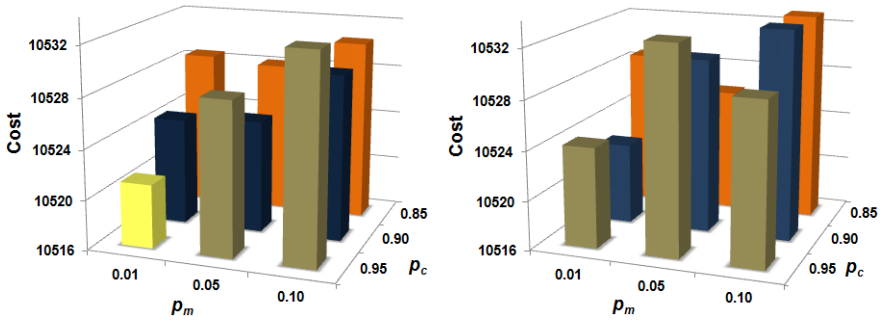
6.2 Development and execution platform

The experimental analysis was performed on a Dell PowerEdge server with QuadCore Xeon E5430 processor at 2.66 GHz, 8GB RAM, and CentOS Linux 5.2.

6.3 Parameter setting experiments

A study was performed to determine the best values of three EA parameters: *popSize*, and the crossover (p_C) and mutation (p_M) probabilities. The candidate values were: $popSize \in [50, 75, 100]$, $p_C \in [0.85, 0.90, 0.95]$, and $p_M \in [0.01, 0.05, 0.1]$.

Thirty executions of the proposed EA were performed for two average-size DPPM instances, with $CI = 14$, $CNC = 6$, $N = 102$, and $\theta = 0.15$. Fig. 3 presents two samples of the parameter configuration results for $popSize = 50$ and $popSize = 100$.



(a) Population size = 50 individuals. (b) Population size = 100 individuals.

Fig. 3. Two samples of the parameter configuration results.

The best results were obtained when using the configuration $popSize = 50$, $p_C = 0.95$, $p_M = 0.01$, showing the importance of the crossover operator to compute accurate solutions. No significant improvements were detected when increasing *popSize*, suggesting that using a larger population is not useful to improve the results quality.

6.4 Empirical analysis

This subsection presents the experimental results of the proposed EA for the DPPM.

Comparison against previous results. Fifty executions of the proposed EA were performed to solve each of the 36 DPPM instances studied. We perform a comparison against the BAB method, the best known deterministic method for the problem. The EA results are also compared with LB for the problem computed in [9] using CPLEX.

We defined the *GAP* metric in order to compare the quality of the solutions computed by the proposed EA against BAB and LB. The GAP metrics are defined in Eq. 5, where $best_{EA}$ is the cost of the best solution computed by the proposed EA.

$$GAP_{BAB} = \frac{best_{EA} - best_{BAB}}{best_{BAB}} \qquad GAP_{LB} = \frac{best_{EA} - best_{LB}}{best_{LB}} \tag{5}$$

Table 1 presents the experimental results obtained by the proposed EA, and a comparison with BAB and LB regarding both the quality of solutions and the execution time. Those cases where the proposed EA outperformed the BAB method, regarding the cost of the best solution found or the execution time, are marked in bold.

The results in Table 1 show that the proposed EA is an accurate method to solve the DPPM. It outperformed the BAB algorithm in **10** instances—regarding the solution quality—, and it also computed similar results than BAB but requiring significant less execution time in other **12** instances, especially when tight deadlines are imposed. A maximum cost improvement of **4.7%** over BAB was obtained for an instance with 128 activities. The comparison with LB indicates that the EA computed **21** optimal solutions for the problem. Fig. 4(a) summarizes the improvements achieved by the proposed EA over BAB, regarding the type of deadline constraints (loose, medium, and tight), and Fig. 4(b) compares the average execution times of EA and BAB.

Fitness evolution and execution time. This subsection analyzes the fitness evolution of the proposed EA and the trade-off between the solution quality and the required execution time. Fig. 5 presents a representative case for the fitness evolution, showing that despite starting from low-quality solutions, the EA is able to find well-suited schedules in a very short time.

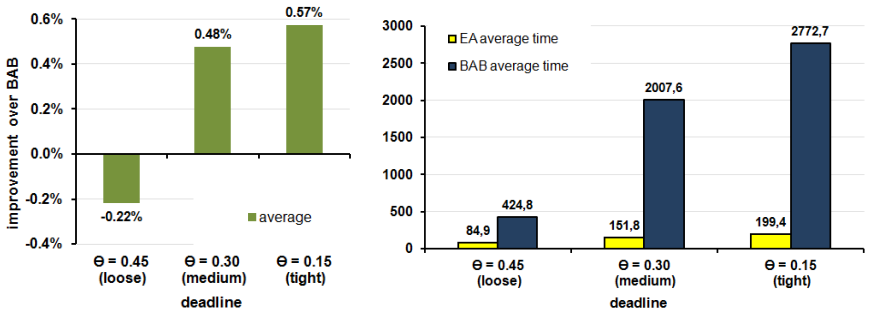
7 Conclusions and future work

This article presented an accurate and efficient EA to solve the DPPM, an important problem in project management. The DPPM proposes to assign modes to activities in order to provide a good tradeoff between duration and cost, enabling the best project performance, while fulfilling deadline constraints on the total project duration.

CNC	N	Θ	EA					BAB			LB		
			avg	t_{AVG}	σ	best	t_{BEST}	best	t_{BAB}	GAP_{BAB}	best	t_{LB}	GAP_{LB}
CI = 13													
		0.15	13027.1	37.8	43.9	12951	**39.8**	12951	2094.7	0.0%	12951	1.9	**0.0%**
5	85	0.30	7859.9	29.6	45.3	7790	**24.7**	7790	158.7	0.0%	7790	0.4	**0.0%**
		0.45	5134.9	23.2	65.5	5064	25.0	5054	65.5	0.2%	5054	0.5	0.2%
		0.15	15538.4	124.5	55.3	15440	**128.4**	15555	3600.0	**-0.7%**	15440	87.8	**0.0%**
6	102	0.30	10615.6	88.1	58.2	10445	**91.8**	10547	3600.0	**-1.0%**	10326	190.0	1.2%
		0.45	6076.7	68.2	32.6	6024	66.7	6010	294.9	0.2%	6010	3.3	0.2%
		0.15	27641.0	222.0	28.0	27603	**229.6**	27693	3600.0	**-0.3%**	27526	2305.2	0.3%
7	117	0.30	17620.2	134.0	75.4	17537	**134.3**	17537	3600.0	0.0%	17537	112.1	**0.0%**
		0.45	11542.7	75.0	92.6	11422	102.2	11389	3600.0	0.3%	11308	2.0	1.0%
		0.15	23941.5	251.7	20.0	23895	**247.2**	24406	3600.0	**-2.1%**	23895	3601.4	**0.0%**
7	119	0.30	14542.9	170.9	42.2	14465	208.6	14425	3600.0	0.3%	14425	32.4	0.3%
		0.45	8769.7	69.1	67.4	8701	103.1	8625	398.9	0.9%	8625	1.4	0.9%
		0.15	22031.1	430.1	173.8	21663	**399.7**	22125	3600.1	**-2.1%**	21603	260.3	0.3%
8	128	0.30	13046.2	301.0	206.1	12735	**309.7**	13357	3600.0	**-4.7%**	12696	14.2	0.3%
		0.45	7507.2	135.4	25.7	7428	**165.1**	7428	210.4	0.0%	7428	0.6	**0.0%**
		0.15	17149.6	247.4	89.1	17005	**308.8**	17118	3600.0	**-0.7%**	17005	11.2	**0.0%**
8	129	0.30	11081.3	137.2	6.6	11077	**197.4**	11077	950.6	0.0%	11077	5.8	**0.0%**
		0.45	7108.8	133.9	29.5	7092	87.6	7092	48.8	0.0%	7092	0.3	**0.0%**
CI = 14													
		0.15	13204.3	32.2	60.6	13057	**29.8**	13057	247.0	0.0%	13057	0.3	**0.0%**
5	85	0.30	8553.5	23.5	46.3	8504	25.5	8481	157.6	0.3%	8481	0.2	0.3%
		0.45	5670.3	19.8	63.1	5573	19.1	5573	10.4	0.0%	5573	0.1	**0.0%**
		0.15	19085.5	78.7	95.5	18948	**89.3**	18948	238.7	0.0%	18948	0.9	**0.0%**
6	102	0.30	12820.5	65.8	275.0	12558	**69.5**	12558	82.4	0.0%	12558	0.5	**0.0%**
		0.45	7676.4	46.3	23.9	7610	35.4	7610	9.5	0.0%	7610	0.1	**0.0%**
		0.15	19794.2	152.8	137.9	19585	**169.9**	19585	1891.3	0.0%	19585	9.7	**0.0%**
7	116	0.30	11218.4	148.9	50.5	11123	141.0	11123	127.6	0.0%	11123	0.7	**0.0%**
		0.45	7429.3	43.3	56.3	7364	35.4	7364	57.6	0.0%	7364	0.3	**0.0%**
		0.15	9764.7	276.9	24.4	9693	**249.8**	9736	3600.1	**-0.4%**	9693	482.0	**0.0%**
7	119	0.30	5943.0	182.3	24.1	5886	**169.0**	5886	3600.0	0.0%	5884	13.5	**0.0%**
		0.45	3879.9	111.9	13.3	3841	117.4	3834	3.6	0.2%	3834	0.3	0.2%
		0.15	8113.5	283.2	21.0	8082	**252.9**	8082	3600.1	0.0%	8082	92.3	**0.0%**
8	128	0.30	5377.8	255.8	23.3	5286	**273.4**	5326	3600.0	**-0.8%**	5263	11.3	0.4%
		0.45	3643.0	125.5	9.1	3604	123.2	3575	160.7	0.8%	3575	0.5	0.8%
		0.15	18858.6	170.7	81.6	18697	**247.0**	18794	3600.0	**-0.5%**	18642	96.3	0.3%
8	129	0.30	11893.6	155.3	33.8	11819	176.4	11808	1014.1	0.1%	11808	3.6	0.1%
		0.45	7914.1	141.5	55.9	7850	**138.8**	7850	237.5	0.0%	7850	0.8	**0.0%**

Table 1. Experimental results of the proposed EA for the DPPM.

The proposed EA was designed to provide accurate solutions, by using operators that allow realistic problem instances to be solved in reduced execution times. Ad-hoc recombination, mutation, and local search operators have been specifically proposed to achieve this goal. In addition, a feasibility check/repair method is incorporated in order to assure that the tentative solutions meet the deadline constraints.



(a) Solution quality. (b) Execution time.

Fig. 4. Comparison between the proposed EA and BAB.

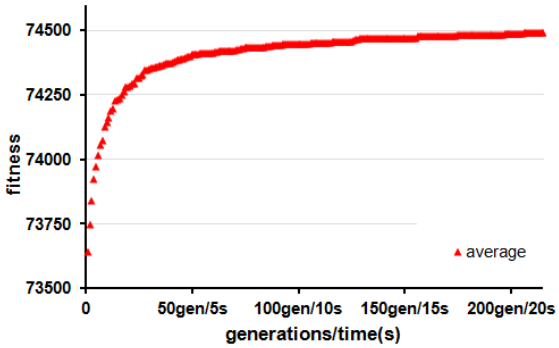


Fig. 5. Representative case for the fitness evolution of the proposed EA.

The experimental evaluation of the proposed EA was performed on a set of 36 complex benchmark instances from [1], regarding standard metrics for complexity. A comparative study against the BAB algorithm [9], one of the best well-known deterministic techniques for the problem, was performed. The numerical results demonstrated that the proposed EA is able to outperform BAB in terms of both quality of solutions and computational efficiency, especially when solving instances with tight deadlines.

The proposed EA was able to find **10** new best-known solutions for the benchmark instances solved, obtaining cost improvement of up to **4.7%** over BAB. The EA also computed similar results than BAB but requiring significant less execution time in other **12** instances. The comparison with LB indicates that the EA was able to compute **21** optimal solutions out of the 36 problem instances tackled.

The previous results indicate that the proposed EA is an accurate and efficient method to solve the DPPM, especially when dealing with tight deadlines.

The main lines for future work are related with improving the evolutionary search and the computational efficiency of the proposed EA. New evolutionary operators can be designed to improve the results quality. In addition, parallel models of the proposed EAs can be applied in order to further improve the efficiency of the method.

Bibliography

[1] C. Akkan, A. Drexl, and A. Kimms. Network decomposition based benchmark results for the discrete time-cost tradeoff problem. *European Journal of Operational Research*, 165:339–358, 2005.

[2] K. Anagnostopoulos and L. Kotsikas. Experimental evaluation of simulated annealing algorithms for the timecost trade-off problem. *Applied Mathematics and Computation*, 217(1):260–270, 2010.

[3] T. Bäck, D. Fogel, and Z. Michalewicz, editors. *Handbook of evolutionary computation*. Oxford University Press, 1997.

[4] V. Deineko and G. Woeginger. Hardness of approximation of the discrete time-cost tradeoff problem. *Operations Research Letters*, 29:207–210, 2001.

[5] E. Demeulemeester, B. Reyck, B. Foubert, W. Herroelen, and M. Vanhoucke. New computational results on the discrete time/cost trade-off problem in project networks. *Operations Research Letters*, 49:1153–1163, 1998.

[6] E. Dunne, J. Ghosh, and C. Wells. Complexity of the discrete timecost tradeoff problem for project networks. *Operations Research*, 45:302306, 1997.

[7] E. Fallah-Mehdipour, O. Bozorg, M. Rezapour, and M. Mario. Extraction of decision alternatives in construction management projects: application and adaptation of NSGA-II and MOPSO. *Expert Systems with Applications*, 39(3):2794–2803, 2012.

[8] M. Garey and D. Johnson. *Computers and intractability*. Freeman, 1979.

[9] B. Hafizoglu and M. Azizoglu. Linear programming based approaches for the discrete time/cost trade-off problem in project networks. *Journal of the Operational Research Society*, 61(4):676–685, 2010.

[10] O. Hazir, M. Haouari, and E. Erel. Discrete time/cost trade-off problem: A decomposition-based solution algorithm for the budget version. *Computers and Operations Research*, 37(4):649–655, 2010.

[11] J. Nicholas and H. Steyn. *Project Management for Engineering, Business, and Technology*. Routledge, 2011.

[12] M. Wall. GAlib: A C++ library of genetic algorithm components. *Mechanical Engineering Department, Massachusetts Institute of Technology*, 1996.

[13] J. Zhang and H. Shan. Multi-resource constrained discrete time/cost trade-off problem and its improved genetic algorithm. In *International Conference on Management Science and Engineering*, pages 123–128, 2010.

Acknowledgments

The work of S. Nesmachnow has been partially supported by PEDECIBA and ANII, Uruguay. The authors would like to thank Baykal Hafizoglu for providing the problem instances used in the experimental analysis.