WCCI 2012 IEEE World Congress on Computational Intelligence
June, 10-15, 2012 - Brisbane, Australia IEEE CEC

A Modified NSGA-II for the Multiobjective Multi-mode Resource-Constrained Project Scheduling Problem

Sanderson C. Vanucci
Minerconsult Engenharia Ltda
Avenida Raja Gabaglia 1255
Belo Horizonte, MG, Brazil, 30380-435
email: Sanderson@ottimah.com

Rafael Bicalho
Dep. Mathematics
Universidade Federal de Minas Gerais
Av. Antônio Carlos, 6627
Belo Horizonte, MG, Brazil, 31270-901
email: rbmbika@gmail.com

Eduardo G. Carrano
Dep. Electrical Engineering
Universidade Federal de Minas Gerais
Av. Antônio Carlos, 6627
Belo Horizonte, MG, Brazil, 31270-901
email: egcarrano@ufmg.br

Ricardo H. C. Takahashi
Dep. Mathematics
Universidade Federal de Minas Gerais
Av. Antônio Carlos, 6627
Belo Horizonte, MG, Brazil, 31270-901
email: taka@mat.ufmg.br

***Abstract*—This work studies a multiobjective version of the Multi-mode Resource-Constrained Project Scheduling Problem (MRCPSP), in which both the total time of execution and the total cost of assignment are treated as objective functions. An NSGA-II based genetic algorithm is employed for the estimation of the Pareto-optimal solution set. An encoding/decoding scheme guarantees that only feasible individuals are represented in the population, and problem-specific mutation and crossover operators are employed in order to enhance the algorithm efficiency. A case study of the engineering design of the facilities of a new mining plant located in the northern Brazilian territory illustrates the application of the proposed methodology. In this case study, several scenarios of project task assignment to a team of workers with different skills and different hiring costs are generated by the proposed algorithm. This quantitative description of the trade-off between project term and project cost can be particularly useful in the preliminary stage of price negotiation between the engineering consulting firm that develops the project and the client mining company.**

I. Introduction

A *project* is a series of interrelated activities or procedures that are conducted toward the accomplishment of a definable goal [1]. This kind of problem is among the first topics that have been studied by the field of operational research. The first statements of those problems were concerned only with the precedence constraints between tasks, and the typical solutions tried to find the sequencing of tasks that allowed the project execution in the smallest possible time [1]. In the decade of 1960's, the issue of constrained resources has been raised [2], leading to the statement of a longstanding problem: the Resource-Constrained Project Scheduling Problem (RCPSP). In this problem, the execution of different tasks will depend on some limited resources, which will force some tasks to be executed sequentially, not by virtue of precedence constraints, but due to the need of using some resource that cannot be employed simultaneously in the tasks. This problem has been found to be NP-hard [3], requiring the application of heuristic search methods in order to be solved in instances involving hundreds of tasks or resources. A generalization of the RCPSP, the *Multi-mode Resource Constrained Project Scheduling Problem* (MRCPSP) has been studied in the last 20 years [4], [5]. The MRCPSP considers that each task may be completed using different resources (different modes), which will incur in different completion times. This version of the project scheduling problem is still more difficult, and has been solved up to the optimality only for instances of the order of 20 tasks, up to now [6].

A large number of different approaches has been developed for dealing with the MRCPSP, in recent years. [7] used a simulated annealing algorithm for solving the problem. [8] developed another simulated annealing algorithm for the same problem. [9] presented a first attempt to solve the MRCPSP using genetic algorithms. [10] further studied the applicability of genetic algorithms to the problem, now investigating also several different objective functions. [11] developed a particle swarm algorithm for the MRCPSP. [6] has developed an estimation of distribution algorithm that performs forward-backward local searches in order to enhance the search ability. All those studies have been developed in a single-objective framework, considering only one objective function. The most common objective function considered so far has been the total time (makespan).

In this paper, a multiobjective approach for the MRCPSP is proposed, considering, in addition to the total time of the project (makespan), the total cost of task assignment as another objective[1]. This approach is motivated by the practical problem of task assignment in the context of teams of skilled workers, which have different skills, meaning both different costs of hiring and different times for task completion. This additional objective may be important because there is a trade-off between project term and project cost. The availability of quantitative information about this trade-off may be particularly important in the negotiation between a firm that will develop a project and a client firm that contracts the project. The case study presented here is related to the design of the facilities of a new mining plant located in the northern Brazilian territory, executed by an engineering consulting firm which was contracted by a mining company.

The proposed algorithm is based on the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [13], which is a standard multiobjective genetic algorithm widely employed in applications. A particular encoding/decoding procedure is proposed, such that only feasible solutions are generated, avoiding the usual difficulties with problem constraints. Some problem-specific mutation and crossover operators are also employed.

This work is organized as follows: Section II presents the formulation of the problem. Section IV presents the proposed algorithm. Finally, section V presents the results in a real case study in a Brazilian company.

II. The Multiobjective Multi-mode Resource Constrained Project Scheduling Problem

The *Multiobjective Multi-mode Resource Constrained Project Scheduling Problem* (MMRCPSP) is stated in this section. Let $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ represent a set of agents, and $\mathcal{K} = \{k_1, k_2, \ldots, k_m\}$ a set of tasks. In most of the cases, $m > n$. The time is represented by variable $t \in \mathbb{R}_+$. The time of starting of task k_j is represented by t_j, and the duration of execution of task k_j by agent $a_i \in \mathcal{A}$ is given by $d_{ij} \in \mathbb{R}_+$. A schedule $\mathcal{S}$ is composed of m triples (k_j, a_i, t_j) meaning that the task k_j starts at time t_j and is assigned to agent a_i:

$$\mathcal{S} = \{(k_j, t_j, a_i) \mid j = 1, \ldots, m\} \tag{1}$$

It is assumed that each agent can be assigned with only one task in any moment. Assuming that $t_{j_1} < t_{j_2}$, this means that:

$$(k_{j_1}, t_{j_1}, a_i), (k_{j_2}, t_{j_2}, a_i) \in \mathcal{S} \Rightarrow t_{j_2} \geq t_{j_1} + d_{ij_1} \tag{2}$$

The tasks in the set $\mathcal{K}$ are inter-related by precedence constraints expressed by matrix $\mathbf{P} \in \{1, 0\}^{m \times m}$, for which $p_{j_2 j_1} = 1$ means that the task k_{j_2} should be preceded by task k_{j_1} (k_{j_2} should start after k_{j_1} is finished), and $p_{j_2 j_1} = 0$ means that such a precedence relation does not need to hold. This constraint may be expressed as:

$$\begin{aligned} &(k_{j_1}, t_{j_1}, a_{i_1}), (k_{j_2}, t_{j_2}, a_{i_2}) \in \mathcal{S} \text{ and } p_{j_2 j_1} = 1 \\ &\Rightarrow t_{j_2} \geq t_{j_1} + d_{i_1 j_1} \end{aligned} \tag{3}$$

[1]The authors of this paper thank an anonymous reviewer that pointed out that there were some former attempts to develop a multiobjective approach for the MRCPSP using evolutionary computation techniques, although not published in mainstream technical literature, for instance in reference [12].

The decision variables of MMRCPSP may be synthesized in an assignment matrix, $\mathbf{X} \in \{0, 1\}^{n \times m}$, in which x_{ij} is a binary decision variable which is 1 if task $k_j \in \mathcal{K}$ is assigned to agent $a_i \in \mathcal{A}$ and 0 otherwise, and in a starting time vector $\mathbf{t} = [\ t_1 \ \ \ldots \ \ t_m\ ]'$. Using this notation, the requirement that all tasks should be executed is expressed as:

$$\sum_{i=1}^{n} x_{ij} = 1 \ \ , \ \ \forall\ j = 1, \ldots, m \tag{4}$$

One of the objective functions of MMRCPSP is the function $f_c(\mathbf{X})$, which may represent the cost of execution of the tasks. The cost of assignment of task $k_j \in \mathcal{K}$ to agent $a_i \in \mathcal{A}$ is given by c_{ij}. Therefore, the total project cost f_c becomes defined as:

$$f_c(\mathbf{X}) = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} \cdot x_{ij} \tag{5}$$

In situations in which the agents represent costs only when they are executing the tasks, this function models the agent-related financial cost of project execution, which should be minimized. A similar function could also represent a quality factor, with the quality of execution of task $k_j \in \mathcal{K}$ by agent $a_i \in \mathcal{A}$ is given by c_{ij}. In this case, this function should be maximized.

The other objective function of MMRCPSP, $f_t(\mathbf{X}, \mathbf{t})$, is the total execution time of the project. This function is given by the termination time of the latest task:

$$f_t(\mathbf{X}, \mathbf{t}) = \max_{j} (t_j + d_{ij} \cdot x_{ij}) \tag{6}$$

The *Multiobjective Multi-mode Resource Constrained Project Scheduling Problem* (MMRCPSP) is stated as:

Multiobjective Multi-mode Resource Constrained Project Scheduling Problem (MMRCPSP): *minimize functions $f_c(\mathbf{X})$ and $f_t(\mathbf{X}, \mathbf{t})$, given by expressions (5) and (6), subject to constraints (2), (3) and (4).*

In order to properly define the simultaneous minimization of f_c and f_t, the decision variables are merged in a variable $\mathbf{Z} = (\mathbf{X}, \mathbf{t})$, and the objective functions are put together in a vector function $f(\mathbf{z}) = [\ f_c(\mathbf{X}) \ \ \ f_t(\mathbf{X}, \mathbf{t})\ ]'$. Define the relational operator $\leq$ between vectors $\mathbf{v}$ and $\mathbf{w}$ as true when $v_i \leq w_i$ is true componentwise, and the relational operator $\neq$ between vectors $\mathbf{v}$ and $\mathbf{w}$ as true when $v_i \neq w_i$ is true for some component i. The feasible set Ω represents all $\mathbf{z}$ for which the constraints (2), (3) and (4) hold. The Pareto-optimal solution set $\mathcal{P}^*$ is defined as:

$$\begin{aligned} \mathcal{P}^* = \{&\mathbf{z}^* \in \Omega \mid \ \nexists\ \mathbf{z} \in \Omega \text{ such that} \\ &\mathbf{f}(\mathbf{z}) \leq \mathbf{f}(\mathbf{z}^*) \text{ and } \mathbf{f}(\mathbf{z}) \neq \mathbf{f}(\mathbf{z}^*)\} \end{aligned} \tag{7}$$

This paper is concerned with the problem of finding estimates (sample sets) of the Pareto-optimal set $\mathcal{P}^*$ of MMR-CPSP.

III. Solution Encoding and Decoding

It should be noticed that the MMRCPSP, as formulated in the former section, is a problem which is difficult to be solved mainly due to the structure of precedence constraints (3) and total time objective function (6), which are stated in terms of a time variable. However, it is possible to partly eliminate this difficulty by stating a new decision variable set which determines the ordering of task execution by the agents, instead of directly determining the time of execution starting. The precedence constraints are also treated within the encoding scheme, using a correction mechanism. Therefore, all encoded solutions become feasible, and the encoding transforms the problem into an unconstrained permutation problem.

In the proposed scheme, each solution is encoded as a $2 \times m$ matrix, with each column representing a task. The matrix rows contain the information about: (1) the agent to which the task is assigned; (2) the task priority. The task priority list is a permutation of the set of numbers $\mathbf{y} = \{1, \ldots, m\}$. The task priority is used as a decision variable that determines the order in which the agent will execute the tasks which are assigned to it: the next task to be executed by an agent is the task k_i for which $y(i)$ has smallest value, in the list of assignments of this agent. The time of starting of a task becomes defined implicitly: a task is started as soon as the agent finishes all tasks with more priority in its list of assignments.

The precedence constraints are treated by a simple correction mechanism: any time when a priority list $\mathbf{y}$ is generated, it is examined from 1 to m. If a task k_j, which is preceded by task k_l, is found first than the later, the priority of k_j is assigned to k_l, and the priorities of k_j and of all other tasks until the one with priority immediately smaller than the former priority of k_l are increased by one. The algorithm 1 shows this mechanism. In this algorithm:

- $\phi(j)$ is the priority of the task of higher priority that precedes task k_j. If no task precedes task k_j, $\phi(j) = 0$.
- $v(j)$ is a vector which contains the indices of the tasks with priority smaller than $\phi(j)$ and greater than or equal to $y(j)$.

Algorithm 1 Pseudocode for Precedence Correction

```
1:  procedure PrecedenceCorrection($\mathbf{X}$,$\mathbf{y}$,$\mathbf{P}$)
2:      $j \leftarrow 1$
3:      while $j < m$ do
4:          if $\phi(j) > y(j)$ then
5:              $\phi(j) \leftarrow y(j)$
6:              $y(v(j)) \leftarrow y(v(j)) + 1$
7:          end if
8:          $j \leftarrow j + 1$
9:      end while
10: end procedure
```

An example of solution encoding for $n = 2$ and $m = 4$ is shown in table I, in which the tasks have precedence relations presented in table II. Figure 1 shows the representation of those precedence relations as a graph. In this example, the tasks 1 and 2 are the first ones to be executed, since they don't depend on any other task. As those tasks are executed by the same agent, the task 1 is executed first, because it has higher priority. As soon as task 1 becomes ready, the task 3 is started by agent 2 and agent 1 goes to task 2. After finishing task 2, agent 1 starts task 4. Finally, task 5 is executed by agent 2.

TABLE I: Solution encoding

Task	k_1	k_2	k_3	k_4	k_5
Agent	1	1	2	1	2
Priority	1	2	3	4	5

TABLE II: Precedence relations

Task	Preceding tasks
k_1	-
k_2	-
k_3	k_1
k_4	k_1, k_2
k_5	k_3, k_4

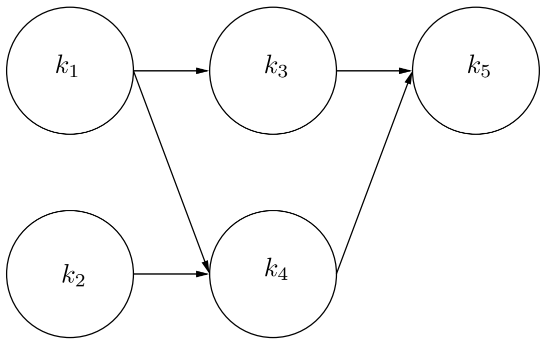

Fig. 1: Precedence relations

Each encoded solution is evaluated (decoded) according to the procedure described in algorithm 2. In this algorithm:

- Θ represents the set of indices of uncompleted tasks;
- Ψ represents the subset of Θ composed of the indices of the tasks that are the next ones in the priority list of any agent which is not busy and do not depend on any task in Θ;
- $\delta(\cdot)$ is the duration of the tasks included in the argument set.
- $\mathbf{T}_s$ is an $1 \times m$ vector with the time of starting of each task.
- $\mathbf{T}_f$ is an $1 \times m$ vector with the time of termination of each task.

IV. Proposed Algorithm

The algorithm proposed in this work is based on the NSGA-II [13], with specific crossover and mutation operators inspired in the ones presented in [14], and a new encoding scheme that represents only feasible solutions. The algorithm pseudo-code is presented in Algorithm 3.

In this algorithm:

Algorithm 2 Pseudocode for Decoding

```
1:  procedure DECODE($\mathbf{X}$,$\mathbf{y}$)
2:      $t \leftarrow 0$
3:      $\Theta \leftarrow \{1, \dots, m\}$
4:      while $\Theta \neq \emptyset$ do
5:          evaluate $\Psi$
6:          $\mathbf{T}_s(\Psi) \leftarrow t$
7:          $\mathbf{T}_f(\Psi) \leftarrow t + \delta(\Psi)$
8:          $t \leftarrow t + \min(\delta(\Psi))$
9:          $\Theta \leftarrow \Theta - \arg\min(\delta(\Psi))$
10:     end while
11: end procedure
```

Algorithm 3 Pseudocode for NSGAII

```
1:  procedure NSGAII($N$,$N_A$)
2:      $t \leftarrow 0$
3:      $P_t \leftarrow$ new_population($N$)
4:      $Q_t \leftarrow \emptyset$
5:      $A \leftarrow$ non_dominated($P_t$)
6:      while not stop criterion do
7:          $R_t \leftarrow P_t \cup Q_t$
8:          $\mathcal{F} \leftarrow$ fast_non_dominated_sorting($R_t$)
9:          $P_{t+1} \leftarrow \emptyset$
10:         $i \leftarrow 1$
11:         while $|P_{t+1}| + |\mathcal{F}_i| \leq N$ do
12:             $\mathcal{C}_i \leftarrow$ crowding_distance_assignment($\mathcal{F}_i$)
13:             $P_{t+1} \leftarrow P_t \cup \mathcal{F}_i$
14:             $i \leftarrow i + 1$
15:         end while
16:         $\mathcal{F}_i \leftarrow$ sort($\mathcal{F}_i$,$\mathcal{C}_i$, 'descending')
17:         $P_{t+1} \leftarrow P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$    ▷ fill $P_{t+1}$
    with the $N - |P_{t+1}|$ less crowded individuals of $\mathcal{F}_i$
18:         $Q_{t+1} \leftarrow$ selection($P_{t+1}, N$)
19:         $Q_{t+1} \leftarrow$ crossover($Q_{t+1}$)
20:         $Q_{t+1} \leftarrow$ mutation($Q_{t+1}$)
21:         $t \leftarrow t + 1$
22:         $A \leftarrow$ non_dominated($A \cup Q_t$)
23:     end while
24: end procedure
```

- $P \leftarrow$ **new_population**(N) generates a random population with N individuals following the encoding scheme which has been adopted for the problem;
- $A \leftarrow$ **non_dominated**(P) returns the individuals which lie in the first front of the population P;
- $\mathcal{F} \leftarrow$ **fast_non_dominated_sorting**(P) employs *fast non-dominated sorting* [13] procedure to find the front of each solution in population P;
- $\mathcal{C}_i \leftarrow$ **crowding_distance_assignment**$(\mathcal{F}_i)$ employs *crowding distance assignment* procedure to estimate how the solutions of front i are spread in the objective space. Solutions which are in less populated areas receive higher crowding values than the ones which are located in more populated areas of the objective space;
- $\mathcal{F}_i \leftarrow$ **sort**$(\mathcal{F}_i, \mathcal{C}_i,$ 'descending'$)$ sorts the solutions of front i in descending order of $\mathcal{C}_i$;
- $Q \leftarrow$ **selection**(P, N) uses binary stochastic tournaments for performing selection of the population P. The outcome of this procedure is a population Q, with N individuals which have been selected from P with replacement.

A. Crossover

The Uniform Crossover operation is employed only over the portion of chromosomes that encode the agents who execute the tasks, as exemplified in Figure 2. A crossover pointer vector, containing 0's and 1's, is generated randomly with uniform probability. In the parent individuals, the agents pointed by the 1's of the crossover pointer are changed between parents, while the agents corresponding to the 0's are kept unchanged, giving rise to the offspring individuals.

It should be noticed that, in the crossover operation, the priority of tasks are not changed. This procedure was found to be less disruptive than the other alternative, of changing all variables.

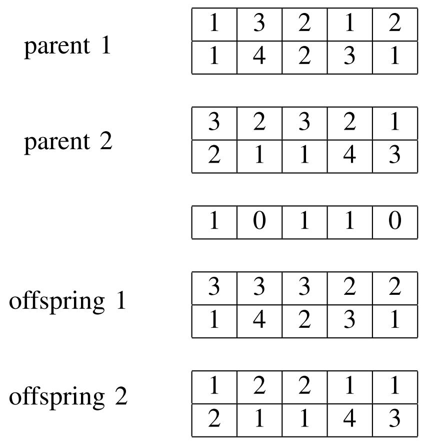

Fig. 2: Crossover operation. The upper row of the individual encoding of the parents (which encode the agents) are interchanged in the positions indicated by 1's in the crossover pointer vector. The positions corresponding to 0's in the crossover pointer, and also all the task priorities (the second row of encoding) are kept unchanged.

B. Mutation

Four different mutation operators have been employed, in order to exploit specific characteristics of the problem. The overall mutation operation is structured as:

1) A real-valued random number $m \in [0, 1]$ and an integer random number $r \in \{1, \dots, T\}$ are generated.
2) If $m \leq 0.2$: change randomly the agent which executes task r;
3) If $0.2 < m \leq 0.4$: change randomly the priority of task r;
4) If $0.4 < m \leq 0.7$: replace the agent of task r with the agent which takes less time to execute that task;
5) If $m > 0.7$: replace the agent of task r with the agent that executes this task with the smallest cost.

TABLE III: List of tasks to be executed in the project.

Task	Description
k_1	Design Criteria
k_2	Preliminary master plan
k_3	Consolidated master plan
k_4	Primary crushing arrangements
k_5	Secondary crushing arrangements
k_6	Arrangements: conical pile / homogenization
k_7	Arrangements: Secondary screening
k_8	Tertiary crushing: mechanical arrangements
k_9	Intermediary stock pile: mechanical arrangements
k_{10}	Load station: mechanical arrangements
k_{11}	Sampling system: mechanical arrangements
k_{12}	Water treatment plant: mechanical arrangements
k_{13}	Compressors: mechanical arrangements
k_{14}	Belt conveyors: mechanical arrangements
k_{15}	Memory calculation: dimensioning of mechanical equipment
k_{16}	Subsidies for composing the investment estimate in mechanics
k_{17}	Subsidies for the final report of the mechanical project
k_{18}	3D model: general arrangement of mechanical plant
k_{19}	Consolidation of mechanical equipment list

TABLE IV: Precedence relations between tasks.

Task	Preceding tasks
1	
2	1
3	2
4	3
5	3
6	3
7	3
8	3
9	3
10	3
11	3
12	3
13	3
14	3
15	4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14
16	15
17	15
18	4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14
19	15

TABLE V: Cost of task assignment to each agent. The agents are designated by A1 - A7.

Task	A1	A2	A3	A4	A5	A6	A7
k_1	600	420	240	210	180	120	120
k_2	3600	2520	1440	1260	1080	720	720
k_3	210	300	120	105	90	60	60
k_4	180	420	300	300	180	120	120
k_5	157	367	262	262	157	105	105
k_6	142	285	237	237	380	380	237
k_7	150	300	250	250	400	400	250
k_8	165	330	275	275	440	440	275
k_9	120	240	200	200	320	320	200
k_{10}	187	375	312	312	500	500	312
k_{11}	105	210	175	175	280	280	175
k_{12}	180	360	300	300	480	480	300
k_{13}	90	180	150	150	240	240	150
k_{14}	180	360	300	300	480	480	300
k_{15}	300	600	500	500	800	800	500
k_{16}	450	400	300	300	150	100	100
k_{17}	1500	2400	2400	2400	900	900	900
k_{18}	610	1220	3050	3050	6100	6100	6100
k_{19}	180	270	180	180	180	30	30

TABLE VI: Time of each agent for completing the tasks. The agents are designated by A1 - A7.

Task	A1	A2	A3	A4	A5	A6	A7
k_1	6	8.6	15	17.1	20	30	30
k_2	36	51.4	90	102.9	120	180	180
k_3	2.1	3	7.5	8.6	10	15	15
k_4	1.8	8.6	12	12	20	30	30
k_5	1.6	7.5	10.5	10.5	17.5	26.3	26.3
k_6	1.4	7.9	9.5	9.5	5.9	5.9	9.5
k_7	1.5	8.3	10	10	6.3	6.3	10
k_8	1.7	9.2	11	11	6.9	6.9	11
k_9	1.2	6.7	8	8	5	5	8
k_{10}	1.9	10.4	12.5	12.5	7.8	7.8	12.5
k_{11}	1.1	5.8	7	7	4.4	4.4	7
k_{12}	1.8	10	12	12	7.5	7.5	12
k_{13}	0.9	5	6	6	3.8	3.8	6
k_{14}	1.8	10	12	12	7.5	7.5	12
k_{15}	3	16.7	20	20	12.5	12.5	20
k_{16}	4.5	6.3	8.3	8.3	16.7	25	25
k_{17}	15	37.5	37.5	37.5	100	100	100
k_{18}	6.1	305	122	122	61	61	61
k_{19}	1.8	3.3	5	5	5	30	30

V. Results

The proposed algorithm has been tested for task assignment in some sub-projects related to the design of the facilities of a new mining plant located in the northern Brazilian territory, executed by an engineering consulting firm. In all cases, the projects were constituted of the design tasks of sub-systems of the plant. The sets of agents were always constituted by teams of engineers with different skills and which were more or less experienced. Therefore, the cost of execution of each task varied from agent to agent, and the time needed for task completion also varied.

This section presents some essays of the task assignment in the project of the mechanical sub-system of the facility. The list of tasks is presented in table III. The precedence relations between tasks is presented in table IV. The cost of assignment of each task to each agent is presented in table V. The time that would be consumed by each agent for completing each task is presented in table VI.

The Pareto-optimal front obtained by the proposed algorithm for this case study is presented in figure 3. This solution set was found after 100 generations, using a population with 100 individuals.

In figure 3 it becomes clear the existence of assignment options that are too different, both in terms of cost and of time for execution. The most expensive solution costs about three times the price of the less expensive one, but takes only nearly one fifth of the time needed for project completion. There is a group of solutions that are very cheap and that demand a long time for project completion. These solutions involve using only the non-experienced low-income workers. There is also a group of solutions that are very expensive and very fast. Those solutions involve using the most experienced and skilled workers all the time, and using the less experienced workers only for avoiding bottlenecks in project development, guaranteeing that the most critical tasks are assigned preferentially to the most experienced workers. These two groups

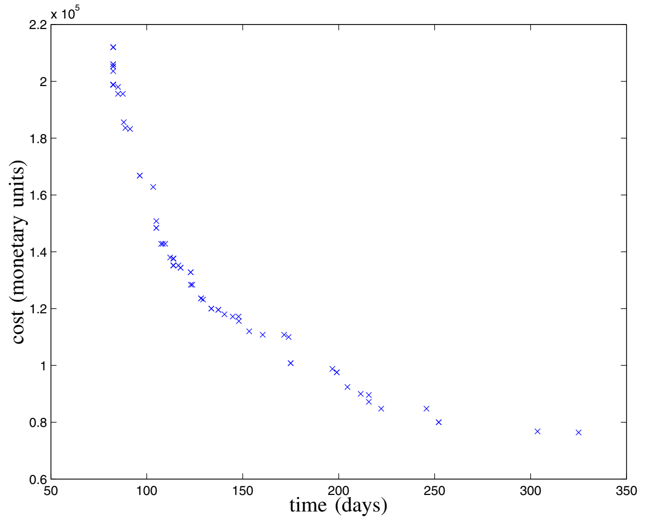

Fig. 3: Pareto-optimal front of the case study.

of solutions that were found by the proposed algorithm are relatively obvious, and usually are not the solutions that are adopted, because the costumers usually don't accept a very large time of execution for the project, and also don't accept very expensive costs.

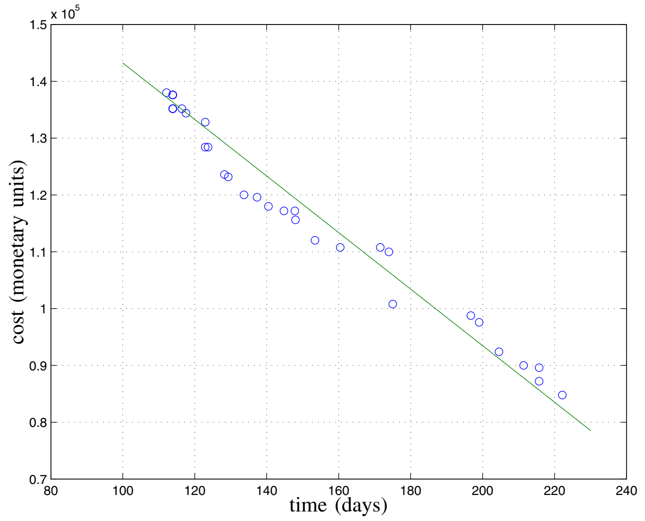

Fig. 4: Central region of the Pareto-optimal front of the case study. The line obtained by linear regression is superimposed.

The most interesting solutions, in this case, have costs between 0.85×10^5 and 1.4×10^5, and durations between 110 and 220 days. These solutions are much more complex to be synthesized, involving different combinations of usage of the available workforce. This portion of the Pareto-front, which has been found to be useful as a reference for price negotiation with the client companies, is shown in Figure 4. This portion of the Pareto-front was found, in this case, to be close to a line segment with equation:

$$y = -497.6x + 7.855 \times 10^4 \quad , \quad 110 \leq x \leq 220 \qquad (8)$$

This means that the contract between the consulting firm and the client company could set deadlines that were between 110 and 220 days for project execution. Each day less than the maximum of 220 days would represent 497.6 monetary units for the cost incurred by the consulting firm. This explicit trade-off between cost and time allows the employment of a systematic procedure for price determination, based on a well-informed cost estimation of the service, for a given contracted term.

VI. Conclusions

In recent years, there has been a steady interest in the study of the Multi-mode Resource-Constrained Project Scheduling Problem (MRCPSP). In most of the research works dealing with this subject, the objective function has been the total time of execution of the project. This work stated a multiobjective version of MRCPSP. In this formulation, both the total time of execution and the total cost of assignment are treated as objective functions.

An NSGA-II based genetic algorithm is employed for the estimation of the Pareto-optimal solution set. An encoding/decoding scheme guarantees that only feasible individuals are represented in the population, and problem-specific mutation and crossover operators are employed in order to enhance the algorithm efficiency.

The new formulation proposed here was motivated by the problem of task assignment in teams of skilled workers that are committed to execute some design project. A case study of the engineering design of the facilities of a new mining plant located in the northern Brazilian territory illustrates the application of the proposed methodology. In this case study, several scenarios of project task assignment to a team of workers with different skills and different hiring costs are generated by the proposed algorithm. This quantitative description of the trade-off between project term and project cost can be particularly useful in the preliminary stage of price negotiation between the engineering consulting firm that develops the project and the client mining company.

Several further developments of the proposed methodology will be conducted in the near future, including: the inclusion of the composition of the agent set (the design team) as a decision variable, the inclusion of uncertainty in the task duration (with agent-dependent uncertainty), and the inclusion of a functionality for the optimization of task re-assignment in a project that has already been started.

References

[1] M. M. Klein, "Scheduling project networks," *Communications of the ACM*, vol. 10, no. 4, pp. 225–231, 1967.

[2] J. D. Wiest, "Some properties of schedules for large projects with limited resources," *Operations Research*, vol. 12, no. 3, pp. 395–418, 1964.

[3] K. S. Naphade, S. D. Wu, and R. H. Storer, "Problem space search algorithms for resource-constrained project scheduling," *Annals of Operations Research*, vol. 70, pp. 307–326, 1997.

[4] A. Drexl and J. Gruenewald, "Nonpreemptive multimode resource-constrained project scheduling," *IIE Transactions*, vol. 25, no. 5, pp. 74–81, 1993.

[5] R. Kolisch, A. Sprecher, and A. Drexl, "Characterization and generation of a general class of resource-constrained project scheduling problems," *Management Science*, vol. 41, no. 10, pp. 1693–1703, 1995.

[6] L. Wang and C. Fang, “An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem,” *Computers and Operations Research*, vol. 39, pp. 449–460, 2012.

[7] F. F. Boctor, “Resource-constrained project scheduling by simulated annealing,” *International Journal of Production Research*, vol. 34, no. 8, pp. 2335–2351, 1996.

[8] J. Jozefowska, M. Mika, R. Rozycki, G. Waligora, and J. Weglarz, “Simulated annealing for multi-mode resource-constrained project scheduling,” *Annals of Operations Research*, vol. 102, pp. 137–155, 2001.

[9] M. Mori and C. C. Tseng, “A genetic algorithm for multi-mode resource constrained project scheduling problem,” *European Journal of Operational Research*, vol. 100, no. 1, pp. 134–141, 1997.

[10] J. Alcaraz, C. Maroto, and R. Ruiz, “Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms,” *Journal of the Operational Research Society*, vol. 54, no. 6, pp. 614–626, 2003.

[11] B. Jarbouia, N. Damaka, P. Siarry, and A. Rebaic, “A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems,” *Applied Mathematics and Computation*, vol. 195, no. 1, pp. 299–308, 2008.

[12] H. Iranmanesh, M. R. Skandari, and M. Allahverdiloo, “Finding Pareto optimal front for the multimode time, cost quality trade-off in project scheduling,” *World Academy of Science, Engineering and Technology*, vol. 40, pp. 346–350, 2008.

[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: Nsga ii,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[14] R. F. Subtil, E. G. Carrano, M. J. F. Souza, and R. H. C. Takahashi, “Using an enhanced integer NSGA-II for solving the multiobjective generalized assignment problem,” in *Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC 2010)*, Barcelona, Spain, July 2010, pp. 1–7.