ISSN 2395-8618

Project Scheduling: A Memetic Algorithm with Diversity-Adaptive Components that Optimizes the Effectiveness of Human Resources

Virginia Yannibelli and Analía Amandi

Abstract—**In this paper, a project scheduling problem is addressed. This problem supposes valuable assumptions about the effectiveness of human resources, and also considers a priority optimization objective for project managers. This objective is optimizing the effectiveness levels of the sets of human resources defined for the project activities. A memetic algorithm is proposed for solving the addressed problem. This memetic algorithm incorporates diversity-adaptive components into the framework of an evolutionary algorithm. The incorporation of these components is meant for improving the performance of the evolutionary-based search, in both exploitation and exploration. The performance of the memetic algorithm on instance sets with different complexity levels is compared with those of the heuristic search and optimization algorithms reported until now in the literature for the addressed problem. The results obtained from the performance comparison indicate that the memetic algorithm significantly outperforms the algorithms previously reported.**

Index Terms—**Project scheduling, human resource assignment, multi-skilled resources, memetic algorithms, evolutionary algorithms, simulated annealing algorithms.**

I. Introduction

IN MOST companies and organizations, project scheduling is recognized as a really central, difficult and time-consuming task [1, 2].

A project scheduling problem usually implies defining feasible start times (i.e., the precedence relations between the project activities must not be violated) and feasible human resource assignments (i.e., the human resource requirements of project activities must be satisfied) for project activities so that a given optimization objective is reached. In this context, the available knowledge of the effectiveness of human resources in relation to project activities is considered in order to define feasible human resource assignments for project activities. This is really important because both the development and the results of project activities depend mainly on the effectiveness of the human resources assigned to such activities [1, 2].

In the past 30 years, different kinds of project scheduling problems have been formally described and also addressed in the literature. However, to the best of the authors' knowledge, only few project scheduling problems take into consideration human resources with different effectiveness levels [3, 4, 5, 6, 10, 22, 23, 26], a main characteristic of real project scheduling problems. Such project scheduling problems supposes very different assumptions about the effectiveness of human resources.

The project scheduling problems formally described in [3, 4, 5, 26] suppose that human resources master one or several skills and have an effectiveness level for each mastered skill. Then, the effectiveness level of a human resource in a given project activity is determined by considering only the effectiveness level of the human resource in respect of one of the skills required for such activity. Therefore, it is supposed that the effectiveness level of a human resource depends only on their skills.

Unlike the project scheduling problems above mentioned, the project scheduling problem described in [6] suppose that the effectiveness level of a human resource depends on several factors inherent to its work context (i.e., the project activity to which the resource is assigned, the skill to which the resource is assigned within the project activity, the set of human resources assigned to the project activity, and the attributes of the resource). This assumption concerning the effectiveness levels of human resources is really important since human resources usually have very different effectiveness levels regarding different work contexts, and thus, the effectiveness level of a human resource is typically determined in respect of the factors inherent to its work context [1, 2]. In addition to this assumption, the project scheduling problem described in [6] considers a priority optimization objective for project managers. This objective implies optimizing the effectiveness of the sets of human resources defined for the project activities.

The project scheduling problem described in [6] is a variant of the RCPSP (Resource Constrained Project Scheduling Problem) [9] and so is recognized as an NP-Hard optimization problem. For this reason, heuristic search and optimization

Manuscript received on June 23, 2015, accepted on September 30, 2015, published on October 15, 2015.

The authors are with the ISISTAN Research Institute, UNICEN University, Tandil, Argentina, and also with CONICET, National Council of Scientific and Technological Research, Argentina (e-mail: {virginia.yannibelli, analia.amandi}@isistan.unicen.edu.ar).

ISSN 2395-8618

algorithms are required to solve problem instances with different complexity levels in a reasonable period of time.

In this paper, a memetic algorithm is proposed to solve the project scheduling problem described in [6]. This memetic algorithm incorporates diversity-adaptive components into the framework of an evolutionary algorithm. The incorporation of these diversity-adaptive components is meant for improving the performance of the evolutionary-based search [18, 19, 20].

This memetic algorithm is proposed for solving the problem described in [6] mainly because of the following. Memetic algorithms with diversity-adaptive components have been proven to be really effective in the resolution of a variety of NP-Hard optimization problems, and also have been proven to be much more effective than evolutionary algorithms and memetic algorithms with non diversity-adaptive components in the resolution of different NP-Hard optimization problems [18, 19, 20]. Therefore, the memetic algorithm proposed could outperform the heuristic search and optimization algorithms reported until now in the literature for solving the problem. To the best of the authors' knowledge, three heuristic search and optimization algorithms have been reported until now for solving the problem: a traditional evolutionary algorithm [6], a memetic algorithm [7] which incorporates a hill-climbing algorithm within the framework of an evolutionary algorithm, and a hybrid evolutionary algorithm [8] which incorporates a simulated annealing algorithm within the framework of an evolutionary algorithm.

The remainder of the paper is organized as follows. In Section 2, a review of reported project scheduling problems which consider the effectiveness of human resources is presented. In Section 3, the project scheduling problem addressed here is described. In Section 4, the memetic algorithm proposed for the problem is presented. In Section 5, the computational experiments carried out to evaluate the performance of the memetic algorithm are presented and also an analysis of the results obtained. Finally, In Section 6, the conclusions of the present work are presented.

II. Related Works

In the past 30 years, different kinds of project scheduling problems have been formally described and addressed in the literature. However, to the best of the authors' knowledge, only few of these project scheduling problems take into account human resources with different effectiveness levels [3, 4, 5, 6, 10, 22, 23, 26], a main characteristic of real project scheduling problems. Such project scheduling problems suppose very different assumptions about the effectiveness of human resources. This section reviews the assumptions supposed in different project scheduling problems described in the literature.

In [12, 13, 14, 17, 25], multi-skill project scheduling problems with different optimization objectives are formally described. In these problems, project activities require a given set of skills for their development, and also a given number of human resources for each skill into such set. The human resources available for the project activities master one or several skills, and these problems supposes that human resources that master a given skill have the same effectiveness level in respect of such skill.

In [15, 16, 24], skilled workforce project scheduling problems with different optimization objectives are formally described. In these problems, each project activity requires only one human resource with a given skill. Besides, the human resources available for project activities master one or several skills. These problems suppose that human resources that master a given skill have the same effectiveness level in respect of such skill.

In [3], a multi-skill project scheduling problem is formally described which considers hierarchical levels of skills. In this regard, this problem supposes that human resources that master a given skill have different effectiveness levels with respect to such skill. Besides, the project activities of this problem require a given set of skills for their development, a given minimum level of effectiveness for each skill of the set, and a given amount of human resources for each pair of skill and level. Then, this problem supposes that the human resource sets feasible for a given project activity have the same effectiveness level regarding the development of such activity.

In [4, 5, 26], multi-skill project scheduling problems with different optimization objective sets are formally described. In these problems, most project activities require only one human resource with a given skill. The human resources available for project activities master one or several skills. These problems suppose that human resources that master a given skill have different effectiveness levels in relation to such skill. Besides, these problems suppose that the effectiveness level of a human resource in a given activity only depends on the effectiveness level of the human resource in respect of the skill required for such activity.

Unlike the problems previously mentioned, the project scheduling problem formally described in [6] supposes that the effectiveness of a human resource depends on several factors inherent to its work context, and then different effectiveness levels can de defined for each human resource in relation to different work contexts. This assumption concerning the effectiveness of human resources is really important since human resources usually have very different effectiveness levels regarding different work contexts, and thus, the effectiveness level of a human resource is typically determined in respect of the factors inherent to its work context [1, 2]. Considering the previously mentioned, the project scheduling problem described in [6] supposes important assumptions concerning the effectiveness of human resources in the context of project scheduling problems.

ISSN 2395-8618

III. PROJECT SCHEDULING PROBLEM DESCRIPTION

In this paper, the project scheduling problem introduced in [6] is addressed. A description of this project scheduling problem is presented below.

A project contains a set A of N activities, $A = \{1, \ldots, N\}$, that has to be scheduled. Specifically, a starting time and a human resource set have to be defined for each project activity of the set A. The duration, human resource requirements, and precedence relations of each project activity are known.

The duration of each project activity j is notated as d_j. Besides, it is considered that pre-emption of project activities is not allowed. This means that, when a project activity starts, it must be developed period by period until it is completed. Specifically, the d_j periods of time must be consecutive.

Among the project activities, there are precedence relations. This is because usually each project activity requires results generated by other project activities. Thus, the precedence relations establish that each project activity j cannot start until all its immediate predecessors, given by the set P_j, have completely finished.

To be developed, project activities require human resources skilled in different knowledge areas. Specifically, each project activity requires one or several skills and also a given number of human resources for each skill required.

It is considered that a qualified workforce is available to develop the activities of the project. This workforce is made up of a number of human resources, and each human resource masters one or several skills.

Set SK contains the K skills required in order to develop the activities of the project, $SK = \{1,\ldots, K\}$, and set AR_k contains the available human resources with skill k. Then, the term $r_{j,k}$ represents the number of human resources with skill k required for activity j of the project. The values of the terms $r_{j,k}$ are known for each project activity.

It is considered that a human resource cannot take over more than one skill within a given activity, and also a human resource cannot be assigned more than one activity at the same time.

Based on the assumptions previously mentioned, a human resource can be assigned different project activities but not at the same time, can take over different skills required for a project activity but not simultaneously, and can belong to different possible sets of human resources for each activity.

Therefore, different work contexts can be defined for each available human resource. It is considered that the work context of a human resource r, denoted as $C_{r,j,k,g}$, is made up of four main components. In this respect, the first component refers to the project activity j which r is assigned (i.e., the complexity of j, the domain of j, etc.). The second component refers to the skill k which r is assigned within project activity j (i.e., the tasks associated to k within j). The third component is the set of human resources g that has been assigned j and that includes r (i.e., r must work collaboratively with the other human resources assigned to j). The fourth component refers to the attributes of r (i.e., his or her educational level regarding different knowledge areas, his or her level regarding different skills, his or her experience level regarding different tasks and domains, the kind of labor relation between r and the other human resources of g, etc.). In respect of the attributes of r, it is considered that these attributes could be quantified from available information about r (e.g., curriculum vitae of r, results obtained from evaluations made to r, information about the participation of r in already executed projects, etc.).

The four components previously mentioned are considered the main factors that determine the effectiveness level of a human resource. Because of this, it is assumed that the effectiveness level of a human resource depends on all the components of his or her work context. Then, different effectiveness levels can be considered for each human resource in respect of different work contexts.

The effectiveness level of a human resource r, in respect of a possible context $C_{r,j,k,g}$ for r, is notated as $e_{rCr,j,k,g}$. The term $e_{rCr,j,k,g}$ refers to how well r can take over, within activity j, the tasks associated to skill k, considering that r must work collaboratively with the other human resources of set g. The term $e_{rCr,j,k,g}$ takes a real value over the range [0, 1]. The values of the terms $e_{rCr,j,k,g}$ inherent to each human resource available for the project are known. It is considered that these values could be obtained from available information regarding the participation of the human resources in already carried out projects.

The problem of scheduling a project involves to determine feasible start times (i.e., the precedence relations between the project activities must not be violated) and feasible human resource assignments (i.e., the human resource requirements of project activities must be met) for project activities so that the optimization objective is reached. In this respect, an optimization objective priority for project managers is considered. This optimization objective implies optimizing the effectiveness of the sets of human resources assigned to the project activities. This objective is modeled by Formulas (1) and (2):

$$\max_{\forall s \in S} \left(e(s) = \sum_{j=1}^{N} e_{R(j,s)} \right) \tag{1}$$

$$e_{R(j,s)} = \frac{\sum_{r=1}^{|R(j,s)|} e_{rC_{r,j,k(r,j,s),R(j,s)}}}{|R(j,s)|} \tag{2}$$

Formula (1) optimizes the effectiveness of the sets of human resources assigned to the N project activities. In this formula, set S contains all the feasible schedules for the project in question. The term $e(s)$ refers to the effectiveness level of the sets of human resources assigned to the project activities by schedule s. The term $R(j,s)$ refers to the set of

ISSN 2395-8618

human resources assigned to activity *j* by schedule *s*. The term $e_{R(j,s)}$ refers to the effectiveness level corresponding to *R(j,s)*.

Formula (2) estimates the effectiveness level of the set of human resources *R(j,s)*. This effectiveness level is estimated by calculating the mean effectiveness level of the human resources belonging to *R(j,s)*.

For a more detailed discussion of the project scheduling problem described here and, in particular, of Formulas (1) and (2), readers are referred to the work [6] which has introduced this problem.

IV. Memetic Algorithm with Diversity-Adaptive Components

A memetic algorithm is proposed in order to solve the project scheduling problem addressed here. This memetic algorithm incorporates diversity-adaptive components into the framework of an evolutionary algorithm. The incorporation of these diversity-adaptive components is meant for improving the performance of the evolutionary-based search [18, 19, 20].

In the next sections, the general behavior and the main components of the memetic algorithm are described.

A. General Behavior of the Memetic Algorithm

Fig. 1 describes the general behavior of the memetic algorithm.

```
Memetic Algorithm
inputs: population_size, number_generations, $AP_{cLA}$, $AP_{cUA}$,
        $AP_{mLA}$, $AP_{mUA}$, $\lambda$ (replacement factor), number_iterations,
        $\alpha$ (cooling factor)
outputs: best_solution
procedure:
 1: population = generate_initial_population(population_size);
 2: generation = 1;
 3: while (generation $\leq$ number_generations) do
 4:     mating_pool = parent_selection_process(population);
 5:     offprings = crossover_process(mating_pool, $AP_{cLA}$, $AP_{cUA}$)
 6:     mutation_process(offsprings, $AP_{mLA}$, $AP_{mUA}$)
 7:     population = survival_selection_process(population,
        offsprings, $\lambda$)
 8:     simulated_annealing_stage(population,
        number_iterations, $\alpha$)
 9:     generation = generation + 1
10: end while
11: best_solution = get_best_solution_from(population)
12: return best_solution
```

Fig. 1. Main behavior of the memetic algorithm

As seen in Fig. 1, the memetic algorithm is an iterative process. This process starts from an initial population of solutions. Each solution of this population encodes a feasible schedule for the project to be scheduled. In addition, each solution has a fitness value which represents the quality of the related schedule with respect of the optimization objective of the addressed project scheduling problem. As was mentioned in Section III, this optimization objective involves optimizing the effectiveness of the sets of human resources assigned to the project activities. The iterative process ends when a given number of generations is reached. After this happens, the iterative process provides the best solution of the last population or generation as a solution to the project scheduling problem.

In each iteration, the memetic algorithm develops the following stages. First, a parent selection process is used in order to determine which solutions of the current population will compose the mating pool. The solutions of the current population with the best fitness values will have more probability of being selected.

After the mating pool is composed, the solutions in the mating pool are paired, and a crossover process is applied to each pair of solutions with a diversity-adaptive probability AP_c in order to generate new feasible ones.

Then, a mutation process is applied to each solution generated by the crossover process, with a diversity-adaptive probability AP_m. The mutation process is applied in order to introduce diversity in the new solutions generated by the crossover process.

Then, a survival selection process is used in order to define which solutions from the solutions in the current population and the solutions generated from the mating pool will compose the new population.

Finally, a diversity-adaptive simulated annealing algorithm is applied to each solution of the new population, except to the best solution of this population which is maintained into this population. Thus, the simulated annealing algorithm modifies the solutions of the new population.

B. Components of the Memetic Algorithm

In the next sections, the main components of the memetic algorithm are described. These components are: the encoding and decoding of solutions, the fitness function, the parent selection process, the crossover process, the mutation process, the survival selection process, and the simulated annealing algorithm.

1) Encoding and Decoding of Solutions

To encode the solutions of the population, the encoding introduced in [6] for project schedules was used. By this encoding, each solution is encoded by two lists with a length equal to *N*, considering that *N* is the number of activities in the project to be scheduled.

The first list is a traditional activity list. Each position on this list contains a different activity *j* of the project. Each activity *j* of the project can appear on this list in any position higher than the positions of all its predecessor activities. The activity list represents a feasible order in which the activities of the project can be added to the schedule.

The second list is an assigned resources list. This list contains information about the human resources of each skill *k*

ISSN 2395-8618

assigned to each activity of the project. Specifically, position *j* on this list contains a detail about the human resources of each skill *k* assigned to activity *j* of the project.

To build or decode the schedule related to the encoding above-described, the serial schedule generation method presented in [6] was used. This method adds the *N* project activities in the schedule one by one, considering the order defined by the activity list. For each project activity to be added in the schedule, this method defines the earliest feasible starting time. In this sense, the method considers that a project activity can start once all its predecessors have been completed and when all the human resources assigned to the project activity are available. Thus, the schedule obtained by this method from the encoding is always a feasible one.

With respect of the use of the serial schedule generation method, it is important to note that only one schedule can be obtained from a given encoded solution, however different encoded solutions could lead to the same schedule.

To generate the encoded solutions of the initial population considering the encoding previously-described, the random generation process introduced in [6] was used. By using this process, a very diverse initial population is obtained. This is meant in order to prevent the premature convergence of the evolutionary-based search developed by the algorithm.

2) *Fitness Function*

The fitness function is utilized in order to determine the fitness values of the encoded solutions. The fitness value of an encoded solution represents the quality of the related schedule with respect of the optimization objective of the addressed project scheduling problem. As mentioned in Section III, this optimization objective involves optimizing the effectiveness of the sets of human resources assigned to the project activities.

The detailed behavior of the fitness function is described as follows. Considering a given encoded solution *i*, the fitness function decodes the schedule *s* from the solution *i* by using the serial schedule generation method described in Section IV.B.1. Then, the fitness function calculates the value of the term *e(s)* corresponding to *s* (Formulas (1) and (2)). This value defines the fitness value *f(i)* of the solution *i*.

To calculate the value of term *e(s)*, the fitness function uses the values of the terms $e_{rCr,j,k,g}$ inherent to *s* (Formula 2). As was mentioned in Section III, the values of the terms $e_{rCr,j,k,g}$ inherent to each available human resource *r* are known.

Note that the term *e(s)* takes a real value over [0, …, *N*].

3) *Parent Selection Process*

The parent selection process is applied in order to determine which solutions of the current population will compose the mating pool. Then, the solutions in the mating pool, named parent solutions, will be utilized by the crossover process to generate new feasible solutions, named offspring solutions.

In order to carry out the parent selection process on the current population, the roulette wheel selection process [18] was applied here. This is one of the parent selection processes most applied in the literature [18].

The roulette wheel selection process considers the fitness values of the solutions of the current population and also is biased by a random factor. Thus, the solutions of the current population with the best fitness values will have more chances of being selected and so incorporated in the mating pool.

The roulette wheel selection process works as follows. Given the solutions of the current population, this process defines a selection probability *p(i)* for each solution *i* of the current population by Formula (3), where *f(i)* is the fitness value of solution *i* and *M* is the number of solutions into the current population.

Once defined the selection probabilities of the *M* solutions of the current population, the process calculates the cumulative probability vector *v* corresponding. Specifically, the process creates an empty vector *v* with a length equal to *M*, considering that position *i* on this vector *v* represents to solution *i*. Then, the process calculates a value *v(i)* for each position *i* on vector *v* by Formula (4):

$$p(i) = \frac{f(i)}{\sum_{i=1}^{M} f(i)} \tag{3}$$

$$v(i) = \begin{cases} p(i) & i = 1 \\ v(i-1) + p(i) & i = 2, \ldots, M \end{cases} \tag{4}$$

After the cumulative probability vector *v* is calculated, the process can start to select solutions from the *M* solutions of the current population.

To select one solution from the *M* possible solutions, the process randomly choose a real value *p* over the range [0, 1]. Then, the process goes through vector *v* in order to find the first position *i* on this vector with a value higher than *p*. Once such position *i* is found, solution *i* is considered by the process as the selected solution.

The above-described operation is repeated by the process until *M* solutions are selected to compose the mating pool.

4) *Diversity-Adaptive Crossover Process*

Once composed the mating pool, the solutions in the mating pool are paired, taking into account the order in which these solutions were incorporated into the mating pool. Then, a crossover process is applied to each of these pairs of solutions with a diversity-adaptive probability AP_c, in order to generate new feasible solutions. The crossover process applied here is described below.

Considering a given pair of solutions (*p*1 and *p*2) to be recombined, the crossover process develops the next stages. First, a crossover process feasible for activity lists is applied to the activity lists of *p*1 and *p*2, in order to generate two new

ISSN 2395-8618

activity lists. The first of these new activity lists is assigned to the first offspring solution (*o*1) of the pair, and the second of these new activity lists is assigned to the second offspring solution (*o*2) of the pair. Then, a crossover process feasible for assigned resources lists is applied to the assigned resources lists of *p*1 and *p*2, in order to generate two new assigned resources lists. The first of these new assigned resources lists is assigned to the first offspring solution (*o*1) of the pair, and the second of these new assigned resources lists is assigned to the second offspring solution (*o*2) of the pair. Therefore, the crossover process generates two new solutions (*o*1 and *o*2) from the pair of solutions (*p*1 and *p*2).

With respect of the crossover process applied to the activity lists of *p*1 and *p*2, the two-point crossover process for activity lists [21] was applied. This process works as follows. First, the process randomly chooses two crossover points *x*1 and *x*2, considering that $1 \leq x1 < x2 < N$. After that, the activities on positions [1, *x*1] of the list of *p*1 are positioned in the positions [1, *x*1] of the list of *o*1, in the same order. Then, the process selects the first ($x2 - x1$) activities of the list of *p*2 that are not included in the list of *o*1, and copies these activities in positions [*x*1+1, *x*2] of the list of *o*1, considering the order in which these activities appear in the list of *p*2. Finally, the process selects the ($N - x2$) activities of the list of *p*1 that are not included in the list of *o*1, and copies these activities in positions [*x*2+1, *N*] of the list of *o*1, considering the order in which these activities appear in the list of *p*1. Thus, the process generates the activity list of *o*1 from the activity lists of *p*1 and *p*2.

The generation of the activity list of *o*2 is similar to the generation of the activity list of *o*1. However, the roles of *p*1 and *p*2 are inverted to generate the list of *o*2.

With respect of the crossover process applied to the assigned resources lists of *p*1 and *p*2, the uniform crossover process [18] was applied. This process works as follows. First, the process creates an empty vector *u* with a length equal to *N*. Then, the process randomly defines a real value on [0, 1] for each position *i* of the vector *u*, considering $i = 1,\ldots,N$. After that, the process uses the vector *u* to define the assigned resources lists of *o*1 and *o*2. Specifically, for each position *i*, if $u(i) \leq 0.5$, the resource assignment for position *i* of the list of *o*1 (*o*2) is inherited from *p*1 (*p*2). Otherwise, if $u(i) > 0.5$, the resource assignment for position *i* of the list of *o*1 (*o*2) is inherited from *p*2 (*p*1).

The described crossover process is applied with a diversity-adaptive probability AP_c. In this respect, the diversity-adaptive crossover probability proposed in [11] was considered. This diversity-adaptive crossover probability is defined by Formula (5), where f_{max} is the maximal fitness of the current population, f_{avg} is the average fitness of the current population, and the term $(f_{max} - f_{avg})$ is considered as a measure of the diversity of the current population. Then, *f'* is the higher fitness of the two solutions to be recombined, and AP_{cLA} and AP_{cUA} are given values for the crossover probability, considering $0 \leq AP_{cLA} \leq 1$ and $0 \leq AP_{cUA} \leq 1$.

$$AP_c = \begin{cases} \dfrac{AP_{cUA}\,(f_{max} - f')}{(f_{max} - f_{avg})} & f' \geq f_{avg} \\ AP_{cLA} & f' < f_{avg} \end{cases} \tag{5}$$

By Formula (5), the crossover probability AP_c is adaptive according to the diversity of the current population. In this respect, when the diversity of the current population reduces, AP_c is increased in order to promote the exploration of the search space and thus to prevent the premature convergence of the evolutionary search developed by the memetic algorithm. Otherwise, when the current population is very diverse, AP_c is decreased in order to promote the exploitation of the search space. Thus, probability AP_c is adaptive according to the diversity of the current population, in order to promote either the exploitation or exploration of the search space.

5) *Diversity-Adaptive Mutation Process*

A mutation process is applied to each one of the solutions obtained by the crossover process, with a diversity-adaptive probability AP_m. The mutation process applied is described below.

Considering a given solution *p*1 to be mutated, the mutation process develops the next stages. First, a mutation process feasible for activity lists is applied to the activity list of *p*1. Thus, the activity list of *p*1 is mutated. Then, a mutation process feasible for assigned resources lists is applied to the assigned resources list of *p*1. Thus, the assigned resources list of *p*1 is mutated. By applying the described mutation process, solution *p*1 is mutated.

With respect of the mutation process applied to the activity list of *p*1, the adjacent pairwise interchange process [21] is applied. This mutation process works as follows. For each position $i = 1,\ldots, N\text{-}1$, the process swaps the activities on positions *i* and *i*+1 with a diversity-adaptive probability AP_m, considering that these activities can be swapped only when the activity on position *i* is not a predecessor of the activity on position *i*+1.

With respect of the mutation process applied to the assigned resources list of *p*1, the random resetting process [18] is applied. This mutation process works as follows. For each position $i = 1,\ldots, N$, the process defines a new resource assignment with a diversity-adaptive probability AP_m.

In relation to the diversity-adaptive probability AP_m utilized by the mutation process, the diversity-adaptive mutation probability proposed in [11] was considered. This diversity-adaptive mutation probability is defined by Formula (6), where f_{max} is the maximal fitness of the current population, f_{avg} is the average fitness of the current population, and $(f_{max} - f_{avg})$ is a measure of the diversity of the current population. Then, *f''* is the fitness of the solution to be mutated, and AP_{mLA} and

ISSN 2395-8618

AP_{mUA} are predefined values for the mutation probability, considering $0 \leq AP_{mLA}, AP_{mUA} \leq 1$.

$$AP_m = \begin{cases} \dfrac{AP_{mUA}\,(f_{max} - f'')}{(f_{max} - f_{avg})} & f'' \geq f_{avg} \\ AP_{mLA} & f'' < f_{avg} \end{cases} \quad (6)$$

By Formula (6), the mutation probability AP_m is adaptive based on the diversity of the current population. Specifically, when the diversity of the current population decreases, AP_m is increased, promoting the exploration of the search space and thus avoiding the premature stagnation of the evolutionary search developed by the memetic algorithm. In contrast, when the current population is diverse, AP_m is decreased, promoting the exploitation of the search space. Thus, probability AP_m is adaptive based on the diversity of the current population, to promote either the exploitation or exploration of the search space.

6) *Survival Selection Process*

The survival selection process is used in order to determine which solutions from the solutions in the current population and the solutions generated from the mating pool will compose the new population.

In order to carry out the survival selection process, the fitness-based steady-state selection process [18] was applied here. By applying this process, the best solutions achieved by the memetic algorithm throughout the evolutionary search are preserved [18].

The fitness-based steady-state selection process works as follows. First, the process sorts the *M* solutions of the current population according to their fitness values. After that, the process orders the *M* solutions generated from the mating pool, according to their fitness values.

Then, the process selects the best (M - λ) solutions from the current population, where λ is a parameter that takes an integer value over the range [1, M -1]. After that, the process selects the best λ solutions from the solutions generated from the mating pool by the crossover and mutation processes.

Finally, the *M* selected solutions are used by the process to compose the new population.

7) *Diversity-Adaptive Simulated Annealing Algorithm*

Once a new population is obtained by the survival selection process, a diversity-adaptive simulated annealing algorithm is applied to each solution of this population, excepting the best solution of this population. The best solution of the population is maintained into the population.

This diversity-adaptive simulated annealing algorithm is a variant of the simulated annealing algorithm described in [8].

Fig. 2 below presents the general behavior of the diversity-adaptive simulated annealing algorithm.

As seen in Fig. 2, the diversity-adaptive simulated annealing algorithm is an iterative process. This process starts from a given encoded solution *s* and a given initial value t_i for the temperature parameter. The iterative process ends when a given number of iterations is reached.

```
Simulated Annealing Algorithm
inputs: solution_s, temperature_t_i, number_iterations,
        α (cooling factor)
outputs: new_solution
procedure:
 1: s = solution_s;
 2: t = temperature_t_i;
 3: c = 1;
 4: while ( (t > 0) and (c ≤ number_iterations) ) do
 5:    s' = generate_new_solution_from(s);
 6:    if ( fitness(s) < fitness(s') ) then
 7:       s = s';
 8:    else
 9:       Δ = fitness(s) - fitness(s');
10:       acceptance_probability = exp(-Δ/t);
11:       x = random(0, 1);
12:       if (x < acceptance_probability) then
13:          s = s';
14:       end if
15:    end if
16:    t = t x α;
17:    c = c + 1;
18: end while
19: new_solution = s;
20: return new_solution;
```

Fig. 2. Main behavior of the simulated annealing algorithm

In each iteration, the process generates a new encoded solution *s'* from the encoded solution *s* by applying a move operator. Then, the process analyzes if the solution *s* should be replaced by the new solution *s'*. When the fitness value of the solution *s* is lower than that of the solution *s'*, the process replaces to the solution *s* by the solution *s'*. Otherwise, when the fitness value of the solution *s* is higher than or equal to that of the solution *s'*, the process replaces to the solution *s* by the solution *s'* with an acceptance probability *exp(-Δ/t)*. Note that the term *Δ* represents the difference between the fitness value of the solution *s* and the fitness value of the solution *s'*, and the term *t* represents the current value of the temperature parameter. The used acceptance probability is proportional to the current value of the temperature parameter. At the end of each iteration, the current value of the temperature parameter is reduced by a given cooling factor α.

To apply the described simulated annealing algorithm to the solutions of the population obtained by the survival selection process, an initial value t_i is defined for the temperature parameter. This value t_i is defined based on the diversity of the population. In particular, this value t_i is inversely proportional to the diversity of the population, and is calculated as detailed

ISSN 2395-8618

in Formula (7), considering that the term ($f_{max} - f_{avg}$) represents to the diversity of the population:

$$t_i = 1 / \left(f_{max} - f_{avg} \right). \tag{7}$$

By Formula (7), when the population is very diverse, the value t_i is very low, and so the acceptance probability of the simulated annealing algorithm is also low. Thus, the algorithm fine-tunes the solutions to which it is applied, promoting the exploitation of the search space. When the diversity of the population reduces, the value t_i increases, and the acceptance probability of the algorithm also increases. Thus, the algorithm tries to move away from the solutions to which it is applied, promoting the exploration of the search space.

Considering the above-mentioned, the simulated annealing algorithm is adaptive based on the diversity of the population, in order to promote either the exploitation or exploration of the search space.

With respect to the move operator used by the simulated annealing algorithm to generate a new solution from a given solution, this move operator is described below.

Considering a given encoded solution *s*, the move operator develops the next stages. First, a move operator for activity lists is applied to the activity list of *s*. Thus, a new activity list is obtained. Then, a move operator for assigned resources lists is applied to the assigned resources list of *s*. Thus, a new assigned resources list is obtained. By the described move operator, a new solution is obtained from the given solution *s*.

In relation to the move operator applied to the activity list of *s*, the simple shift operator [21] is applied. This operator works as follows. The operator selects randomly only one activity of the list, and then moves the selected activity from its current position to a new feasible position for this activity. In this respect, the new position is randomly selected from the set of feasible positions for the activity on the list.

In relation to the move operator applied to the assigned resources list of *s*, an operator which is a variant of the random resetting process [18] is applied. This operator works as follows. The operator selects randomly only one position of the list, and then defines a new resource assignment for the selected position.

V. Computational Experiments

In this section, the computational experiments carried out in order to evaluate the performance of the proposed memetic algorithm are described. Then, the results obtained by these experiments are presented and analyzed in detail. Finally, the performance of the memetic algorithm is compared with those of the heuristic search and optimization algorithms reported until now in the literature for solving the addressed problem.

A. Instance Sets

In order to carry out the computational experiments, the six instance sets introduced in [7] were used. Each of these instance sets contains 40 different instances. Besides, these instance sets have no instances in common. Table I presents the main characteristics of these six instance sets.

Each instance of these six instance sets includes a number of activities to be scheduled and a number of human resources available for developing these activities. For each activity, the instance specifies the duration, the predecessor activities, the required skills, and the number of human resources required for each of these skills. For each available human resource, the instance specifies the skills mastered by the human resource. Besides, the instance specifies the effectiveness level of each available human resource *r* with respect of each one of the possible work contexts for *r* in the instance. In particular, the instance includes all the terms $e_{rCr,j,k,g}$ corresponding to each available human resource *r* and a random value over the range [0,1] for each of these terms.

Each instance of these six instance sets has a known optimal solution with respect of the effectiveness level of the sets of human resources assigned to the project activities. These know optimal solutions are considered here as references to evaluate the performance of the memetic algorithm.

TABLE I.
MAIN CHARACTERISTICS OF THE INSTANCE SETS

Instance Set	Activities per instance	Possible sets of human resources per activity
j30_5	30	1 to 5
j30_10	30	1 to 10
j60_5	60	1 to 5
j60_10	60	1 to 10
j120_5	120	1 to 5
j120_10	120	1 to 10

B. Parameter Setting of the Memetic Algorithm

The memetic algorithm has the following nine parameters: population size, number of generations, probabilities AP_{cLA} and AP_{cUA} for the crossover process, probabilities AP_{mLA} and AP_{mUA} for the mutation process, replacement factor λ for the survival selection process, number of iterations and cooling factor α for the simulated annealing algorithm.

The setting of these nine parameters affects the behavior of the memetic algorithm, and thus, could affect the performance of the memetic algorithm. Because of this reason, preliminary experiments were developed with the aim of determining the parameter setting by which the memetic algorithm reaches the best performance on the six instance sets. These preliminary experiments are described below.

Different values were considered for each one of the nine parameters. The second column of Table II presents the range of values considered for each one of these parameters. In this respect, two values were considered for the population size (i.e., 90 and 180), and three values were considered for the number of generations (i.e., 300, 600 and 900). Different

ISSN 2395-8618

values were considered for the probabilities AP_{cLA} and AP_{cUA} (i.e., values over the range [0.5, 1]), and also different values were considered for the probabilities AP_{mLA} and AP_{mUA} (i.e., values over the range [0.01, 0.2]). Two values were considered for the replacement factor λ of the survival selection process (i.e., 45 and 75). Finally, three values (i.e., 25, 50 and 75) were considered for the number of iterations of the simulated annealing algorithm, and three values were considered for the cooling factor α (i.e., 0.7, 0.8 and 0.9). The combinations of these parameters values determined different possible settings for the parameters of the memetic algorithm.

The different possible settings were evaluated on the six instance sets. Specifically, each setting was run 10 times on each instance of the six instance sets. After that, the results obtained by the settings for each of the six instance sets were analyzed. In this respect, for each instance set, the average percentage deviation from the optimal solution was analyzed and also the percentage of instances for which the optimal solution is reached at least once among the 10 runs developed.

The third column of Table II presents the parameter setting by which the memetic algorithm reached the best performance on the six instance sets. This parameter setting is considered in the following sections.

TABLE II.
PARAMETER SETTING OF THE MEMETIC ALGORITHM

Parameter	Values considered	Values selected
Population Size	90, 180	90
Generations	300, 600, 900	300
Crossover		
AP_{cLA}	[0.5, 1]	0.9
AP_{cUA}	[0.5, 1]	0.6
Mutation		
AP_{mLA}	[0.01, 0.2]	0.1
AP_{mUA}	[0.01, 0.2]	0.05
Survival Selection		
λ (replacement)	45, 75	45
Simulated Annealing		
Iterations	25, 50, 75	25
α (cooling)	0.7, 0.8, 0.9	0.9

C. Main Results

The performance of the memetic algorithm was evaluated on the six instance sets. Specifically, the memetic algorithm was run a number of 30 times on each instance of the six instance sets. To develop these runs, the parameter setting detailed in the third column of Table II was used.

The results obtained by the memetic algorithm for each of the six instance sets were analyzed. In particular, for each instance set, the average percentage deviation from the optimal solution was analyzed and also the percentage of instances for which the optimal solution is reached at least once among the 30 runs carried out.

Table III presents the average percentage deviation from the optimal solution (Av. Dev. (%)) obtained by the memetic algorithm for each of the six instance sets.

TABLE III
RESULTS OBTAINED BY THE MEMETIC ALGORITHM

Instance Set	Av. Dev. (%)
j30_5	0
j30_10	0
j60_5	0
j60_10	0
j120_5	0.1
j120_10	0.36

With respect of the Av. Dev. (%) values obtained by the memetic algorithm for the first four instance sets, the memetic algorithm obtained Av. Dev. (%) values equal to 0%. These results mean that the memetic algorithm reached an optimal solution in each of the 30 runs developed on each instance of these four instance sets. Thus, the memetic algorithm reached an optimal performance for the first four instance sets.

With respect of the Av. Dev. (%) values obtained by the memetic algorithm for the last two instance sets, the memetic algorithm obtained Av. Dev (%) values equal to 0.1% and 0.36%, respectively. The meaning of these Av. Dev (%) values was analyzed taking into consideration that the known optimal solutions of the instances of these two sets have a fitness level equal to 120. Thus, Av. Dev (%) values equal to 0.1% and 0.36% mean that the average fitness level of the solutions reached by the memetic algorithm is 119.88 and 119.57, respectively.

Therefore, the memetic algorithm has reached very near-optimal solutions for the instances of the two sets. In addition, the memetic algorithm reached an optimal solution at least once among the 30 runs developed on each instance of these two sets. Based on the above-mentioned, the memetic algorithm reached a near-optimal performance for the last two instance sets.

D. Comparison with the Heuristic Algorithms Reported in the Literature for the Addressed Problem

In this section, the performance of the memetic algorithm is compared with those of the heuristic search and optimization algorithms reported until now in the literature for solving the addressed problem.

To the best of the authors' knowledge, only three heuristic search and optimization algorithms have been reported until now in the literature for solving the addressed problem. In this respect, a traditional evolutionary algorithm was reported in [6], a traditional memetic algorithm was reported in [7] which integrates a hill-climbing algorithm within the framework of an evolutionary algorithm, and a hybrid evolutionary algorithm was reported in [8] which incorporates a simulated annealing algorithm within the framework of an evolutionary algorithm.

ISSN 2395-8618

In [7, 8], the three above-mentioned algorithms have been evaluated on the six instance sets described in Section V.A. The results obtained by each of the three algorithms for the six instance sets are detailed in Table IV, as reported in [7, 8].

Based on the results in Table IV, the performance of the algorithm reported in [8] is better than those of the algorithms reported in [6, 7]. Thus, the algorithm reported in [8] may be considered as the best algorithm reported until now in the literature for solving the addressed problem.

Below, the performance of the algorithm reported in [8] is compared with that of the memetic algorithm proposed here. The algorithm reported in [8] will be referred as algorithm HEA.

TABLE IV
RESULTS OBTAINED BY THE HEURISTIC ALGORITHMS REPORTED IN THE LITERATURE FOR THE ADDRESSED PROBLEM

Instance Set	Evolutionary algorithm [6]	Memetic algorithm [7]	Hybrid evolutionary algorithm [8]
	Av. Dev. (%)	Av. Dev. (%)	Av. Dev. (%)
j30_5	0	0	0
j30_10	0	0	0
j60_5	0.42	0	0
j60_10	0.59	0.1	0
j120_5	1.1	0.75	0.64
j120_10	1.29	0.91	0.8

With respect of the Av. Dev. (%) values obtained by the algorithm HEA and the memetic algorithm proposed here for the first four instance sets, the two algorithms have obtained Av. Dev. (%) values equal to 0%. These results mean that the two algorithms reached an optimal solution in each of the 30 runs developed on each instance of these four instance sets. Thus, for the first four instances sets, the two algorithms have obtained an optimal performance in respect of the Av. Dev. (%) values.

However, with respect of the Av. Dev. (%) values obtained by the algorithm HEA and the memetic algorithm for the last two instance sets, the algorithms have obtained very different Av. Dev. (%) values. In this respect, the algorithm HEA has obtained Av. Dev. (%) values equal to 0.64% and 0.8%, whereas that the memetic algorithm has obtained Av. Dev. (%) values equal to 0.1% and 0.36%. These results mean that the fitness level of the solutions reached by the memetic algorithm for the last two instance sets is significantly higher than that of the solutions reached by the algorithm HEA. Therefore, for the last two instance sets, the performance obtained by the memetic algorithm in respect of the Av. Dev. (%) values is significantly better than that of the algorithm HEA. These results are mainly because of the reasons discussed below.

In the memetic algorithm proposed here, most components into the evolutionary framework are diversity-adaptive. The behavior of these components is adaptive based on the diversity of the evolutionary algorithm population, with the aim of improving the evolutionary-based search. Unlike the memetic algorithm proposed, in the algorithm HEA, most components into the evolutionary framework are non-adaptive. These components totally disregard the diversity of the evolutionary algorithm population and therefore the possibility of improving the evolutionary-based search. Based on the mentioned, the memetic algorithm has significant advantages to develop the evolutionary-based search.

VI. CONCLUSIONS

In this paper, the project scheduling problem introduced in [6] was addressed. This problem supposes really important assumptions concerning the effectiveness of human resources. In this respect, the problem supposes that the effectiveness of a human resource depends on several factors inherent to its work context, and then different effectiveness levels can de defined for each human resource in relation to different work contexts. Besides this assumption, the problem considers a priority optimization objective for project managers. This objective implies optimizing the effectiveness of the sets of human resources defined for the project activities.

To solve the addressed problem, a memetic algorithm was proposed. This memetic algorithm incorporates diversity-adaptive components into the framework of an evolutionary algorithm. The incorporation of these diversity-adaptive components is meant for improving the performance of the evolutionary-based search.

To evaluate the proposed memetic algorithm, exhaustive computational experiments were developed. Specifically, the performance of the memetic algorithm was evaluated on six instance sets with different complexity levels. After that, the performance of the memetic algorithm on the instance sets was compared with those of the heuristic search and optimization algorithms reported until now in the literature for solving the addressed problem.

According to the analysis of the results obtained by the experiments, it may be stated that the memetic algorithm has reached an optimal performance for the first four instance sets and a near-optimal performance for the last two instance sets. Besides, from the comparative analysis developed, it may be stated that the performance of the memetic algorithm is significantly better than those of the heuristic search and optimization algorithms reported until now in the literature for the addressed problem.

In our future work, new diversity-adaptive components will be evaluated into the framework of the memetic algorithm. In addition, new diversity measures will be evaluated.

REFERENCES

[1] G. R. Heerkens, *Project Management*. McGraw-Hill, 2002.

[2] R. K. Wysocki, *Effective Project Management*. Wiley Publishing, 3rd Edition, 2003.

ISSN 2395-8618

[3] O. Bellenguez, E. Néron, "Lower Bounds for the Multi-skill Project Scheduling Problem with Hierarchical Levels of Skills". In: E. Burke and M. Trick (eds.) *PATAT 2004. Lecture Notes in Computer Science*, vol. 3616, Springer, pp. 229–243, 2005.

[4] T. Hanne, S. Nickel, "A multiobjective evolutionary algorithm for scheduling and inspection planning in software development projects", *European Journal of Operational Research* 167, 663–678, 2005.

[5] W. J. Gutjahr, S. Katzensteiner, P. Reiter, Ch. Stummer, M. Denk, "Competence-driven project portfolio selection, scheduling and staff assignment", *Central European Journal of Operations Research*, vol. 16, no. 3, pp. 281–306, 2008.

[6] V. Yannibelli, A. Amandi, "A knowledge-based evolutionary assistant to software development project scheduling", *Expert Systems with Applications*, vol. 38, no. 7, pp. 8403–8413, 2011.

[7] V. Yannibelli, A. Amandi, "A Memetic Approach to Project Scheduling that Maximizes the Effectiveness of the Human Resources Assigned to Project Activities", in: E. Corchado et al. (eds.) *HAIS 2012. Lecture Notes in Computer Science*, vol. 7208, Springer, pp. 159–173, 2012.

[8] V. Yannibelli, A. Amandi, "A Diversity-Adaptive Hybrid Evolutionary Algorithm to Solve a Project Scheduling Problem", in: E. Corchado et al. (eds.) *IDEAL 2014, Lecture Notes in Computer Science*, vol. 8669, Springer, pp. 412–423, 2014.

[9] J. Blazewicz, J. Lenstra, A. Rinnooy Kan, "Scheduling Subject to Resource Constraints: Classification and Complexity", *Discrete Applied Mathematics*, vol. 5, pp. 11–24, 1983.

[10] V. Yannibelli, A. Amandi, "Project scheduling: A multi-objective evolutionary algorithm that optimizes the effectiveness of human resources and the project makespan", *Engineering Optimization*, vol. 45, no. 1, pp. 45–65, 2013.

[11] M. Srinivas, L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24, no. 4, pp. 656–667, 1994.

[12] O. Bellenguez, E. Néron, "A branch-and-bound method for solving multi-skill project scheduling problem", *RAIRO—Operations Research*, vol. 41, no. 2, pp. 155–170, 2007.

[13] L. E. Drezet, J. C. Billaut, "A project scheduling problem with labour constraints and time-dependent activities requirements", *International Journal of Production Economics*, vol. 112, pp. 217–225, 2008.

[14] H. Li, K. Womer, "Scheduling projects with multi-skilled personnel by a hybrid MILP/CP benders decomposition algorithm", *Journal of Scheduling*, vol. 12, pp. 281–298, 2009.

[15] V. Valls, A. Pérez, S. Quintanilla, "Skilled workforce scheduling in service centers", *European Journal of Operational Research*, vol. 193, no. 3, pp. 791–804, 2009.

[16] U. Aickelin, E. Burke, J. Li, "An Evolutionary Squeaky Wheel Optimization Approach to Personnel Scheduling", *IEEE Transactions on evolutionary computation*, vol. 13, no. 2, 433–443, 2009.

[17] C. Heimerl, R. Kolisch, "Scheduling and staffing multiple projects with a multi-skilled workforce", *OR Spectrum*, vol. 32, no. 4, pp. 343–368, 2010.

[18] A. E. Eiben, J. E. Smith, *Introduction to Evolutionary Computing*, Springer, 2nd edition, 2015.

[19] F. J. Rodriguez, C. García-Martínez, M. Lozano, "Hybrid Metaheuristics Based on Evolutionary Algorithms and Simulated Annealing: Taxonomy, Comparison, and Synergy Test", *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 6, pp. 787–800, 2012.

[20] E. Talbi (ed.), *Hybrid Metaheuristics*, Studies in Computational Intelligence, no. 434, Springer, 2013.

[21] R. Kolisch, S. Hartmann, "Experimental Investigation of Heuristics for Resource-Constrained Project Scheduling: An Update", *European Journal of Operational Research*, vol. 174, pp. 23–37, 2006.

[22] P. De Bruecker, J. Van den Bergh, J. Beliën, E. Demeulemeester, "Workforce planning incorporating skills: State of the art", *European Journal of Operational Research*, vol. 243, no. 1, 1–16, 2015.

[23] J. Van den Bergh, J. Beliën, P. De Bruecker, E. Demeulemeester, L. De Boeck, "Personnel scheduling: A literature review", *European Journal of Operational Research*, vol. 226, no. 3, 367–385, 2013.

[24] K. Braekers, R. F. Hartl, S. N. Parragh, F. Tricoire, "A bi-objective home care scheduling problem: Analyzing the trade-off between costs and client inconvenience", *European Journal of Operational Research*, vol. 248, no. 2, pp. 428–443, 2016.

[25] H. Li, N. K. Womer, "Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming", *European Journal of Operational Research*, vol. 246, no. 1, pp. 20–33, 2015.

[26] T. Silva, M. De Souza, R. Saldanha, E. Burke, "Surgical scheduling with simultaneous employment of specialised human resources", *European Journal of Operational Research*, vol. 245, no. 3, pp. 719–730, 2015.