Using Multi-objective Metaheuristics to Solve the Software Project Scheduling Problem

Francisco Chicano
University of Málaga, Spain
chicano@lcc.uma.es

Francisco Luna
University of Málaga, Spain
flv@lcc.uma.es

Antonio J. Nebro
University of Málaga, Spain
antonio@lcc.uma.es

Enrique Alba
University of Málaga, Spain
eat@lcc.uma.es

ABSTRACT

The Software Project Scheduling (SPS) problem relates to the decision of who does what during a software project lifetime. This problem has a capital importance for software companies. In the SPS problem, the total budget and human resources involved in software development must be optimally managed in order to end up with a successful project. Companies are mainly concerned with reducing both the duration and the cost of the projects, and these two goals are in conflict with each other. A multi-objective approach is therefore the natural way of facing the SPS problem. In this paper, a number of multi-objective metaheuristics have been used to address this problem. They have been thoroughly compared over a set of 36 publicly available instances that cover a wide range of different scenarios. The resulting project schedulings of the algorithms have been analyzed in order to show their relevant features. The algorithms used in this paper and the analysis performed may assist project managers in the difficult task of deciding who does what in a software project.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; D.2.9 [**Software Engineering**]: Management

General Terms

Experimentation, Algorithms

Keywords

Software project scheduling, multi-objective optimization, metaheuristics

GECCO'11, July 12–16, 2011, Dublin, Ireland.

1. INTRODUCTION

The high complexity of current software projects justifies the research into computer aided tools to properly schedule the project development. In this paper we focus on the assignment of employees to tasks in a software project so as to reduce both the project cost and duration. This problem is known as the Software Project Scheduling (SPS) problem [1].

The SPS problem is multi-objective in nature and it has been formulated as so, rather than aggregating its objectives (minimizing both the project cost and its duration) into one single value. Contrary to single-objective optimization, the solution of a multi-objective problem such as SPS is not one single solution, but a set of nondominated solutions known as the *Pareto optimal set*, which is called *Pareto border* or *Pareto front* when it is plotted in the objective space [5]. Whatever solution of this set is optimal in the sense that no improvement can be reached on an objective without worsening at least another one at the same time. That is, in the context of the SPS problem, it is not possible to reduce the project cost without increasing its duration (or vice versa). The main goal in the resolution of a multi-objective problem is to compute the set of solutions within the Pareto optimal set and, consequently, the Pareto front. Many metaheuristics have been proposed in the literature to deal with multi-objective problems [4]. Indeed, the most well-known algorithms in the multi-objective community fall into this kind of search technique. In this paper, three classical methods —NSGA-II [6], SPEA2 [18] and PAES [12]— plus two recent algorithms —MOCell [16] and GDE3 [14]— have been used to address the SPS problem.

The existing work on this topic usually proposes a metaheuristic algorithm for solving a specific flavour of the problem and presents the results of some experimental evaluation over a set of problem instances. In this work, however, we want to answer some open questions that have not been addressed yet in previous works. These research questions are:

RQ1: How do these five metaheuristics perform when solving the SPS problem? To answer this question, a thorough comparison between NSGA-II, SPEA2, PAES, MOCell, and GDE3 has been performed over a set of 36 SPS instances covering different project scenarios with different levels of difficulty. To the best of our knowledge, PAES, MOCell, and GDE3 are used on the SPS problem for the first time. NSGA-II and SPEA2 are the two most widely used multi-objective algorithms in the literature.

RQ2: Which are the features of the solutions reached by the metaheuristic algorithms? In this case we analyze the solutions of the different algorithms to find correlations between their features and the region in the objective space where these solutions are located in. In other words, we want to know what "metaheuristic algorithms do" to obtain a solution with some concrete values for the objective functions, i.e., the characteristics of the resulting project schedulings.

The paper is organized as follows. The next section presents some related work on the topic of this paper. Section 3 describes the details of the scheduling problem addressed. In Section 4 the algorithms used to solve the problem are explained. After that, we present the results of some experiments in Section 5. Finally, we summarize our conclusions and outline some lines of future work in Section 6.

2. RELATED WORK

Not much work has been devoted to the multi-objective approach of the SPS problem up to now. Even though few related papers exist, they are usually targeted to solving different flavours of the problem.

Guorguiev *et al.* [9] solve the problem of finding the optimal assignment of workpackages to developer teams with the objectives of minimizing project time and maximizing robustness. This problem was previously solved using a single-objective approach in [2]. The robustness of a solution is measured as the variation of the total development time of the project under unexpected events like newly added tasks to perform or changes in the duration of one task. The algorithm used for the experiments is SPEA2.

Duggan *et al.* [7] solve the problem of task allocation in a software project. The effort of the tasks is measured as units of complexity (UOC). The authors define some levels of skill for the staff: novice, average, expert, etc. Each skill level determines the UOC per days that the engineer is able to perform and the number of errors per UOC that s/he introduces. For each pair engineer-task the project manager must assign a skill level. One solution to the problem is a schedule in which each engineer is assigned to a task in each time. The objectives are to minimize the development time and the number of errors. The algorithm used to solve the problem is an elitist version of NSGA.

Hanne and Nickel [10] solve the problem of planning inspections and other activities in a software project. In their approach, a set of developers have to program some source code items. These source code items must also be inspected by the developer team fulfilling some constraints (for example, the author of a source code item must not be one of their inspectors). The multi-objective approach tries to minimize time, cost, and number of defects. This multi-objective problem is solved by using a multi-objective evolutionary algorithm.

3. SOFTWARE PROJECT SCHEDULING

We follow here the same formulation proposed in [1]. Thus, the resources considered are people with a set of skills and a salary. These employees have a maximum degree of dedication to the project. Formally, each person (employee) is denoted with e_i, where i goes from 1 to E (the number of employees). Let SK be the set of skills, and s_i the i-th skill with i varying from 1 to $S = |SK|$. The skills of the employee e_i will be denoted with $e_i^{skills} \subseteq SK$, the monthly salary with e_i^{salary}, and the maximum dedication to the project with e_i^{maxded}. The salary and the maximum dedication are real numbers. The former is expressed in abstract currency units, while the latter is the fraction of time spent on the project.

The tasks are denoted with t_i, where i goes from 1 to T (the number of tasks). Each task t_i has a set of required skills associated with it, that we denote with t_i^{skills}, plus an effort t_i^{effort}, expressed in person-month (PM). The tasks must be performed according to a Task Precedence Graph (TPG). It indicates which tasks must be completed before a new task is started. The TPG is an acyclic directed graph $G(V, A)$ with a vertex set $V = \{t_1, t_2, \ldots, t_T\}$ and an arc set A, where $(t_i, t_j) \in A$ if the task t_i must be completed, with no other intervening tasks, before task t_j can start.

The objectives of the SPS problem are to minimize the cost and the duration of the project. The constraints are (1) that each task must be performed by at least one person, (2) the set of required skills of a task must be included in the union of the skills of the employees performing the task, and (3) no employee must exceed her/his maximum dedication to the project.

Once we know the elements of a problem instance, we can proceed to describe the elements of a solution to the problem. A solution can be represented with a matrix $\mathbf{X} = (x_{ij})$ of size $E \times T$, where $x_{ij} \geq 0$. The element x_{ij} is the degree of dedication of the employee e_i to the task t_j. If the employee e_i performs the task t_j with a 0.5 dedication degree s/he spends half of her/his time in the company on the task. If an employee does not perform a task s/he will have a dedication degree of 0.0 for that task. This information helps to compute the duration of each task and, indeed, the start and the end time of each one, i.e., the time schedule of the tasks (Gantt diagram). From this schedule we can compute the duration of the project (see Figure 1). The cost can be calculated after the duration of the tasks by taking into account the dedication and the salary of the employees. Finally, the overwork of each employee can be calculated using the time schedule of the tasks and the dedication matrix $\mathbf{X}$.

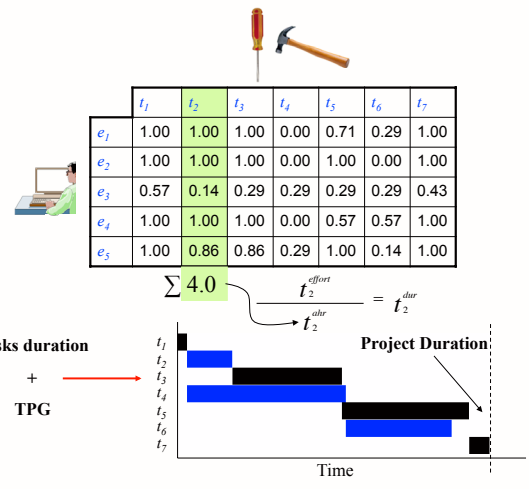

Figure 1: A tentative solution for a sample project. Using the task durations and the TPG, the Gantt diagram of the project can be computed.

In order to evaluate the quality of a given solution, we take into account three issues: project duration, project cost, and solution feasibility. To compute the project duration, denoted with p_{dur}, we need to calculate the duration of each individual task (t_j^{dur}). This is calculated in the following way:

$$t_j^{dur} = \frac{t_j^{effort}}{t_j^{ahr}} \tag{1}$$

where t_j^{ahr} is the amount of human resources (measured in persons) spent on task t_j and is defined as the sum of the dedication degree that the employees have on the task, that is:

$$t_j^{ahr} = \sum_{i=1}^{E} x_{ij} \tag{2}$$

At this point we can also define the participation of employee e_i in the project, e_i^{par}, as the fraction of the total workload of the project that was performed by the employee, that is:

$$e_i^{par} = \frac{\sum_{j=1}^{T} x_{ij} t_j^{dur}}{\sum_{j=1}^{T} t_j^{effort}} = \frac{\sum_{j=1}^{T} \frac{x_{ij}}{\sum_{k=1}^{E} x_{kj}} t_j^{effort}}{\sum_{j=1}^{T} t_j^{effort}} \tag{3}$$

The next step is to compute the starting and ending time for each task (t_j^{start} and t_j^{end}), which are defined according to the following expressions:

$$t_j^{start} = \begin{cases} 0 & \text{if } \nexists t_i, (t_i, t_j) \in A \\ \max\limits_{t_i,(t_i,t_j)\in A} \{t_i^{end}\} & \text{otherwise} \end{cases} \tag{4}$$

$$t_j^{end} = t_j^{start} + t_j^{dur} \tag{5}$$

At the same time, it is possible to compute the project duration (p_{dur}), which is the maximum ending time ever found:

$$p_{dur} = \max_{j=1}^{T} \{t_j^{end}\} \tag{6}$$

The project cost p_{cost} is the sum of the salaries paid to the employees for their dedication to the project. These charges are computed by multiplying the salary of the employee by the time spent on the project. The time spent on the project is the sum of the dedication multiplied by the duration of each task. In summary:

$$p_{cost} = \sum_{i=1}^{E} \sum_{j=1}^{T} e_i^{salary} \cdot x_{ij} \cdot t_j^{dur} \tag{7}$$

In order to check the validity of a solution we must first check that all tasks are performed by somebody, i.e., no task is left undone. That is:

$$t_j^{ahr} > 0 \;\; \forall j \in \{1, 2, \ldots, T\} \tag{8}$$

The second constraint is that the employees performing the task must have the skills required by the task:

$$t_j^{skills} \subseteq \bigcup_{\{i|x_{ij}>0\}} e_i^{skills} \;\; \forall j \in \{1, 2, \ldots, T\} \tag{9}$$

Finally, in order to compute the overwork p_{over} we first need to compute the working function for each employee, defined as:

$$e_i^{work}(\tau) = \sum_{\{j|t_j^{start} \leq \tau \leq t_j^{end}\}} x_{ij} \tag{10}$$

Algorithm 1 Pseudocode for a generic multi-objective EA.

```
1: P(0) ← GenerateInitialPopulation()
2: EvaluateObjectives(P(0))
3: PF ← CreateParetoFront() //Create an empty front
4: t ← 0
5: while not Termination_Condition() do
6:    parents ← Selection(P(t));
7:    offspring←EvolutionaryOperators(parents);
8:    EvaluateObjectives(offspring);
9:    P(t + 1) ←UpdatePopulation(P(t), offspring)
10:   UpdateFront(PF, P(t + 1))
11:   t ← t + 1
12: end while
```

If $e_i^{work}(\tau) > e_i^{maxded}$ the employee e_i exceeds her/his maximum dedication at instant τ. The overwork of the employee e_i^{over} is:

$$e_i^{over} = \int_{\tau=0}^{\tau=p_{dur}} ramp(e_i^{work}(\tau) - e_i^{maxded}) d\tau \tag{11}$$

where $ramp$ is the function defined by:

$$ramp(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \tag{12}$$

The definite integral in (11) always exists and can be easily computed because its integrand is piecewise continuous. The total overwork of the project is then the sum of the overwork for all the employees, i.e.:

$$p_{over} = \sum_{i=1}^{E} e_i^{over} \tag{13}$$

4. MULTI-OBJECTIVE ALGORITHMS

In this section, we briefly describe the five metaheuristics used in this study, namely NSGA-II, SPEA2, PAES, MOCell, and GDE3. They all are evolutionary algorithms (EAs) [3] which are, by far, the most popular metaheuristics for solving MOPs [4, 5] because of their ability of finding a set of trade-off solutions in one single run.

Compared to a single-objective EA, a multi-objective one differs mainly in how the set of non-dominated solutions is managed. The general approach is to keep an external archive, as it can be observed in the pseudocode of a generic multi-objective EA included in Algorithm 1 (the archive is referred as to PF in the algorithm).

Both NSGA-II [6] and SPEA2 [18] use the scheme of Algorithm 1. As evolutionary operators, they have adopted binary tournament selection, simulated binary crossover and polynomial mutation [5]. They differ one each other in the mechanism used to keep a diverse approximated Pareto front. PAES [13] in turn has a population with one single solution that it is iteratively modified by using polynomial mutation only (no crossover operator is required), but it uses an external archive. MOCell [15] is an structured EA that includes an external archive to store the nondominated solutions. It has been engineered with the same evolutionary operators as NSGA-II and SPEA2. Finally, GDE3 [14] also matches the scheme of Algorithm 1, but the typical evolutionary operators are replaced by the differential evolution selection and crossover.

In order to deal with constrained optimization problems such as SPS, all the algorithms have used the constraint

domination principle presented in [5]. The principle states that feasible solutions are better than non-feasible ones and those non-feasible solutions with a smaller overall constraint violation are better (constraints are normalized to be greater than or equal to zero).

5. EXPERIMENTS

In this section we present the results of an empirical study aimed at answering the research questions proposed in Section 1. In the first three sections we describe the methodology, the details of the parameterization of the algorithms, and the instances of the problem used in the experiments. In each of the last two sections we answer the research questions in turn.

For the experiments, we have used jMetal [8], an object-oriented Java-based framework aimed at the development, experimentation, and study of metaheuristics for solving multi-objective optimization problems. jMetal is freely available at `http://jmetal.sourceforge.net/`.

5.1 Methodology

In order to measure the performance of the multi-objective solvers used here, the quality of their resulting nondominated set of solutions has to be considered. Two indicators have been used for this purpose in this article: the hypervolume (HV) indicator [19] and the attainment surfaces [11].

The HV is considered in the field as one of the more accurate indicators. It calculates the volume (in the objective space) covered by members of a nondominated set of solutions for problems where all the objectives are to be minimize, and it provides a measure taking into account the convergence and diversity of the obtained approximation set.

While the HV allows measuring the performance of different algorithms to be compared, from the point of view of a decision maker, knowing about the HV value is not enough, because it gives no information about the shape of the front. Indeed, a project manager wants to know where the front obtained by the different algorithms is located, since the front is what contains the information about the cost *vs.* time trade-off. We need a way of representing the results of a multi-objective algorithm that allows us to observe the expected performance and its variability over multiple runs, in the same way as the average and the standard deviation are used in the single-objective case. To do this we use the concept of empirical attainment function (EAF) [11]. In short, the EAF is a function α from the objective space $\mathbb{R}^n$ to the interval $[0, 1]$ that estimates for each vector in the objective space the probability of being dominated by the approximated Pareto front of one single run of the multi-objective algorithm. Given the r approximated Pareto fronts obtained in the different runs, the EAF is defined as:

$$\alpha(z) = \frac{1}{r} \sum_{i=1}^{r} I(A^i \preceq \{z\}) \tag{14}$$

where A^i is the i-th approximated Pareto front obtained with the multi-objective algorithm and I is an indicator function that takes value 1 when the predicate inside it is true, and 0 otherwise. The predicate $A^i \preceq \{z\}$ means A^i dominates solution z. Thanks to the attainment function, it is possible to define the concept of k%-attainment surface [11]. The attainment function α is a scalar field in $\mathbb{R}^n$ and the k%-attainment surface is the level curve with value $k/100$ for α. Informally, the 50%-attainment surface in the multi-objective domain is analogous to the median in the single-objective one. In a similar way, the 25%- and 75%-attainment surfaces can be used as the first and third "quartile fronts" and the region between them could be considered a kind of "interquartile region". When the number of objectives is one, the 50%-attainment surface is the median and the "interquartile region" is the interquartile range.

Table 1: Parameterization of the algorithms. L is the individual length (number of tasks $\times$ number of employees).

NSGA-II [6]	
Population Size	100 individuals
Selection of Parents	binary tournament + binary tournament
Recombination	simulated binary, $p_c = 0.9$
Mutation	polynomial, $p_m = 1.0/L$
SPEA2 [18]	
Population Size	100 individuals
Selection of Parents	binary tournament + binary tournament
Recombination	simulated binary, $p_c = 0.9$
Mutation	polynomial, $p_m = 1.0/L$
PAES [12]	
Population Size	1 individual
Mutation	polynomial, $p_m = 1.0/L$
Archive Size	100
MOCell [16]	
Population Size	100 individuals (10×10)
Neighborhood	1-hop neighbors (8 surrounding solutions)
Selection of Parents	binary tournament + binary tournament
Recombination	simulated binary, $p_c = 0.9$
Mutation	polynomial, $p_m = 1.0/L$
Archive Size	100 individuals
GDE3 [14]	
Population Size	100 individuals
Recombination	Differential Evolution, $CR = 0.1$, $F = 0.5$

EAs are stochastic algorithms; therefore the results have to be provided with statistical significance. The following statistical procedure has been used. First, 100 independent runs for each algorithm and each problem instance have been performed. The HV indicator and the attainment surfaces are then computed. In the case of HV, the Kruskal-Wallis test has been carried out to check if the differences in the algorithms are statistically significant [17]. Since more than two algorithms are involved in the study, a post-hoc testing phase which allows for a multiple comparison of samples has been performed. All the statistical tests are performed with a confidence level of 95%.

5.2 Parameterization

We have chosen a set of parameter values that guarantees a fair comparison among all the algorithms. All the GAs (NSGA-II, SPEA2, and MOCell), as well as GDE3, use an internal population size equal to 100; the size of the archive is also 100 in PAES, MOCell, and GDE3. The stopping condition is always to compute 100,000 function evaluations for all of the algorithms. Regarding the solution representation, the algorithms adopt a floating point encoding in which gene i represents the dedication of employee $\lfloor i/T \rfloor$ to task i MOD T, where T is the number of tasks of the addressed instance. With this encoding, the typical operators from the multi-objective EAs field have been used. A detailed description of the parameter values adopted for our experiments is provided in Table 1.

5.3 SPS Instances

In order to perform a meaningful study we must analyze a number of instances of the scheduling problem instead of

focusing on only one, which otherwise could bias the conclusions. We have used a total of 36 randomly generated instances that have been previously addressed in [1]. The instance set can be divided into two groups. In the first group we find 18 instances, each one with a different software project. The number of employees can be 5, 10, or 15 and the number of tasks 10, 20, or 30. The total number of skills S can be either 5 or 10 and the number of skills per employee ranges from 2 to 3. We denote the instances of this group with iT-$Eg$$S$. For example, the instance i10-5g5 has 10 tasks, 5 employees and 5 skills. The second group of instances is composed of 18 instances in which the number of skills S is 10. As in the previous group, the number of employees can be 5, 10, or 15 and the number of tasks takes values 10, 20, and 30. However, in this second group two ranges of values are considered separately for the number of skills of the employees: from 4 to 5, and from 6 to 7. The instances in this group are denoted with the name iT-$Ep$$M$, where T and E is the number of tasks and employees, respectively and M is the maximum value in the range for the number of skills per employee. For example, the instance i30-15p7 has 30 tasks, 15 employees and the number of skills per employee varies from 6 to 7. In the 36 instances the maximum dedication for all the employees is 1, which means that all the employees can have complete dedication to the project. All these instances are publicly available at the `http://mstar.lcc.uma.es`.

5.4 Comparison of Algorithms

The first set of experiments is devoted to evaluating the five multi-objective metaheuristics on the set of 36 SPS instances. We have performed a pure multi-objective comparison among the algorithms based on the HV values. Because of room constraints, we just highlight the most interesting findings. The first conclusion that can be drawn from this HV results is that the approximated Pareto fronts reached by PAES have the higher (better) HV values in most of the instances (25 out of 36). MOCell, and GDE3 to a lesser extent, have also addressed the SPS instances properly, since it has performed the best in nine and two instances, respectively, and ranking the second and third for many of the remaining ones. It is worth mentioning that NSGA-II and SPEA2, the most widely used solvers in the literature, have never ranked the first. Taking a look to the average rank, they respectively have scored 3.61 and 4.83 (recall that 1 is the best position and 5 the worst). An in-depth analysis also reveals that the more complex the instance (more tasks and more employees), the better PAES is, i.e., the larger the difference in the HV values with respect to those of the other four algorithms. However, it also occurs that the current settings of PAES lead the algorithm to be outperformed by NSGA-II, MOCell, and GDE3 in the smaller SPS instances (i.e., those with 10 and 20 tasks and 5 employees) and with a small number of skills. The instances matching these requirements are: i10-5g5, i10-5g10, i20-5g5, i10-5p5, i20-5p5, i10-5p7, and i20-5p7. PAES usually assigns a low dedication to the employees for each task in the scheduling, what is beneficial for larger instances with a higher number of tasks and employees, since it avoids constraints violation. However, for instances in which the dedication is expected to be high because few employees with few skills exist, NSGA-II, MOCell or GDE3 have shown to better explore these regions of the search space. The HV values have also shown that the search capabilities of GDE3 diminish as the number of tasks gets increased, that is, when the instances become larger. A careful tracking of the algorithm (not included due to space constraints) reveals that, for these instances, the differential crossover operator hardly computes feasible solutions. Therefore, since the HV indicator is computed only over feasible solutions, the resulting fronts are scarcely populated and lead to small HV values.

Let us now turn to analyze the attainment surfaces of the different algorithms used in the experiments for some instances of the problem. For the sake of clarity, we only show the 50%-attainment surfaces related to the results of the algorithms. We have selected the most representative ones, that is, those with the most interesting features. In addition to the attainment surfaces, in Figures 2 to 6, we also show the best known approximated Pareto front (Reference Pareto Front, RPF).

We observed in the previous analysis that, in general, PAES has performed the best with respect to the HV indicator. If one takes a look to the 50%-attainment functions of the algorithms for the i30-15g5 instance (see Figure 2), it can be easily justified the high HV value obtained by PAES. Indeed, its approximated Pareto fronts are not only close to the Pareto Front but also they cover a larger region in the objective space. In this case, it is clear that the solutions proposed by PAES are better than the ones proposed by the other algorithms. This fact is also observed in the instances i30-10g5, i30-10g10, i30-15g10, i30-10p5, i30-15p5 and i30-15p7; all of the instances with 30 tasks and 10 or 15 employees: the most complex ones. We conclude that PAES has the desirable property of scalability.

In the instance i10-5g5, the 50%-attainment surfaces of all the algorithms are very close to each other (see Figure 3). This fact can also be observed in instances i20-5p5, i30-5p5, i20-10p7. In all the previous instances, SPEA2 is clearly the algorithm with the worst attainment surface. Another interesting related fact appears in the i20-15g10 instance (see Figure 4), in which the attainment surfaces of MOCell, GDE3 and NSGA-II cross that of PAES. This can be also seen in i20-15p5 and i30-10p7. The point is that, according to the HV indicator, PAES has reached the best (highest) HV value in these three instances and, according to the attainment surfaces, the reason is the large extension of the attainment line. However, the other algorithms propose solutions that dominate part of the attainment surface of PAES. Some project managers would prefer the PAES

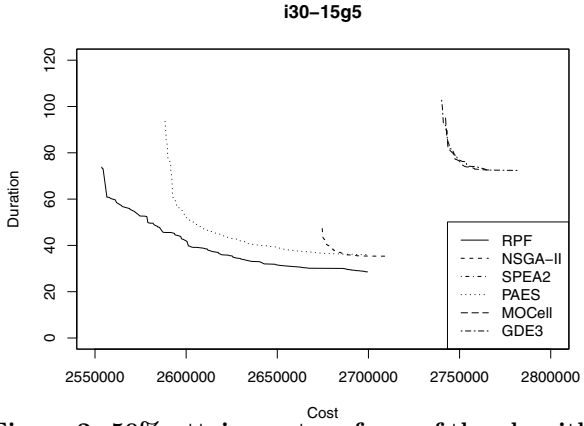

Figure 2: 50%-attainment surfaces of the algorithms in the i30-15g5 instance.

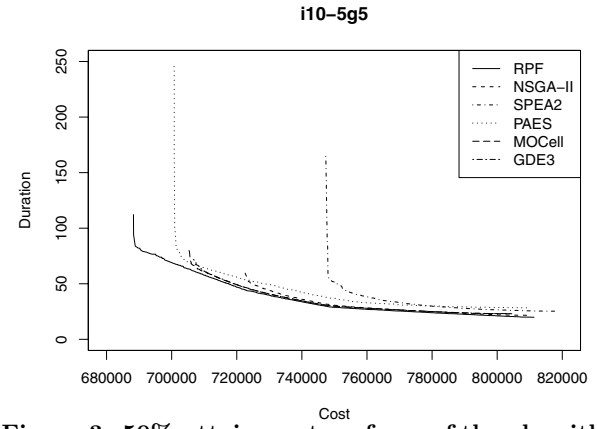

Figure 3: 50%-attainment surfaces of the algorithms in the i10-5g5 instance.

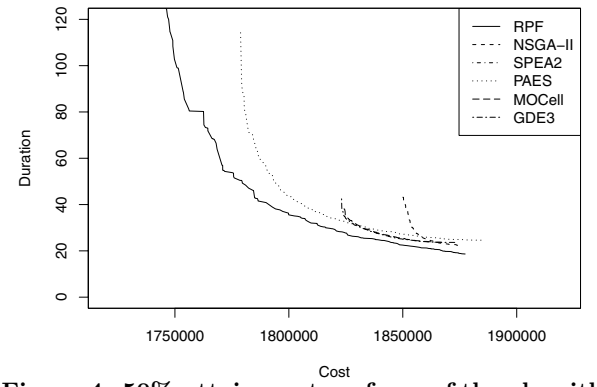

Figure 4: 50%-attainment surfaces of the algorithms in the i20-15g10 instance.

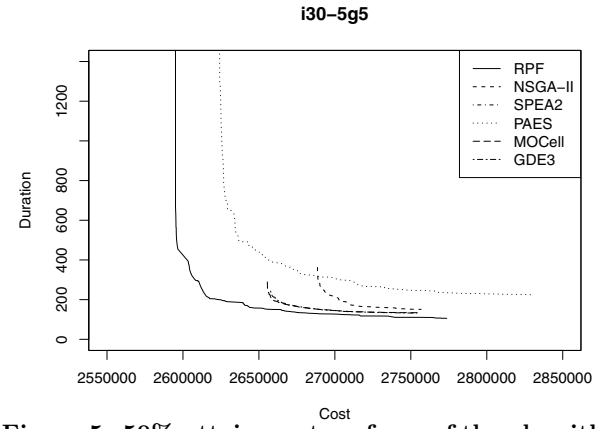

Figure 5: 50%-attainment surfaces of the algorithms in the i30-5g5 instance.

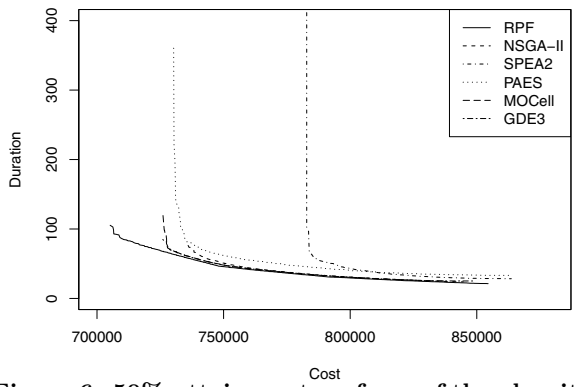

Figure 6: 50%-attainment surfaces of the algorithms in the i10-5p7 instance.

solutions because the range of values for the objectives is wider, but some others would prefer the solutions from MOCell or GDE3 because they have both lower cost and lower duration in their interest region. The previous example illustrates that a scalar indicator like HV is not always the most suitable indicator to make a decision, since scalar values hide a lot of information that could help the project managers in their decisions.

In the i30-5g5 instance, MOCell, GDE3 and even NSGA-II are better than PAES in a specific region of the objective space (see Figure 5). If this region is interesting for the project manager, PAES is not a good algorithm for her/his purposes, even although the hypervolume of PAES is the highest one with statistical significance. The scenario shown in Figure 5 is the most frequent one in the instances solved: 18 out of the 36 instances share this feature. From a visual inspection of the 50%-attainment surfaces we conclude that MOCell and GDE3 dominate the solution of PAES in a specific region, followed by NSGA-II.

Finally, the last observed sort of scenario can be found in the i10-5p7 instance (Figure 6). In this case, the 50%-attainment surface of PAES is clearly dominated by the ones of MOCell, GDE3 and NSGA-II. Other instances with the same behavior are i10-5p5, i10-10p7, and i10-15p7, all of them with 10 tasks. It can be concluded that in the case of simple instances MOCell, GDE3, and to a lesser extent NSGA-II, are the best algorithms for solving the multi-objective problem.

5.5 Analysis of the Problem Solutions

In this section we focus on the solutions obtained using the multi-objective algorithms. We want to analyze the features of these solutions, showing correlations between their features and the region in the objective space they can be found. In particular, we are interested in analyzing the participation of each employee in the project, e_i^{par}, and the amount of human resources spent on each task, t_j^{ahr}. We want to analyze how these values change as the solutions move in the objective space. For each proposed solution by one algorithm for one single instance, the $E + T$ values have to be analyzed. This means a large amount of data to process and show. In order to reduce this amount of data without losing interesting information, the following analysis has been performed. First, we focus on the results of PAES, since it is the algorithm covering, in general, the wider region in the objective space. For each instance, all the solutions of the approximated Pareto front obtained in the different independent runs of the algorithm are considered. The e_i^{par} and t_j^{ahr} values are then computed for each employee and each task in all the previous solutions. The Spearman rank correlation coefficients [17] between all the e_i^{par}, t_j^{ahr}, p_{dur}, and p_{cost} are then calculated. In Figure 7 we how the correlation coefficients for the i20-15g5 instance. An arrow up means positive correlation and an arrow down means negative correlation. The absolute value of the correlation is shown in gray scale (the darker the higher).

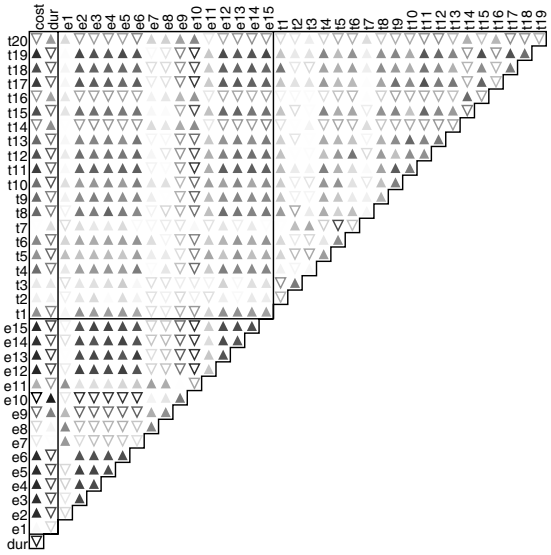

Figure 7: Spearman rank correlation coefficients between p_{cost}, p_{dur}, e_i^{par} and t_j^{ahr} in the i20-15g5 instance for the solutions obtained with PAES.

The first observation we can highlight is the clear inverse correlation between project cost and duration. This is an expected result and it gives no relevant information since we are analyzing solutions belonging to sets of non-dominated solutions where an increase in cost implies a decrease in duration. If we focus on the project cost and the participation of the employees we observe that for employees e_2, e_3, e_5, e_6, e_{11}, e_{12}, e_{13}, e_{14}, and e_{15} the correlation is positive while for e_7, e_8, e_9 and e_{10} is negative. What does this mean? When the cost of the solutions proposed by the algorithm increases, the participation of former set also increases; that is, these employees spend more and more time in the tasks of the project as we move in the objective space to solutions with higher cost and lower duration (observe the negative correlation of these employees with project duration). On the contrary, the participation of e_7, e_8, e_9 and e_{10} is reduced. This does not mean that they spend less time in the project, since we defined participation as a normalized measure; this means that the fraction of workload that is performed is reduced because the other employees increase their participation. But the interesting question is, why does this happen? The answer is that e_7, e_8, e_9 and e_{10} are the cheapest employees, i.e., their salary is actually lower. The algorithm then looks for schedulings that assign most of the work of the project to them because they earn less money. However, when the project duration is reduced (and the cost increased) the other employees have to increase their participation. We can also observe a negative correlation between e_7, e_8, e_9 and e_{10} and the rest of the employees, as expected from the previous discussion.

Let us now turn to the tasks. We observe positive correlations between two initial tasks: t_1 and t_8. If we take a look to the TPG of the instance (Figure 8) we notice that these tasks have no precedence constraints. Indeed, it makes no sense to increase the work in one of them and not in the other, since the saved time does not allow a reduction in the total project duration and the project cost could increase. However, t_2 is also an initial task, and a negative correlation between t_1 and t_2 is identified. In this instance, task t_2 does not require too much effort and can be easily done: it does not belong to the critical path. In this situation the employees are shared between t_2 and t_1, which are parallel tasks. The most interesting observation in the solutions proposed for this instance is perhaps the positive correlation between tasks t_{14}, t_{16}, t_{20} and the project duration. This means that the solutions proposed by the algorithm are shorter when less people work on these tasks. This automatic finding by the algorithm seems counterintuitive and we had to carefully analyze the solutions to understand why this happens. At the end, the conclusion is that the solutions proposed by PAES in the short-duration region of the objective space are not optimal. In a better solution, more human resources would be assigned to tasks t_{14}, t_{16} and t_{20}, thus showing positive correlation with project duration. In fact, the i20-15g5 instance belongs to the class of instances in which PAES is outperformed in some regions; in particular, the short-duration region is outperformed by NSGA-II. After diving into the correlations computed for the approximated Pareto fronts proposed by NSGA-II, we conclude that the positive correlations between the project duration and the mentioned tasks do not appear, therefore supporting the previous explanation.

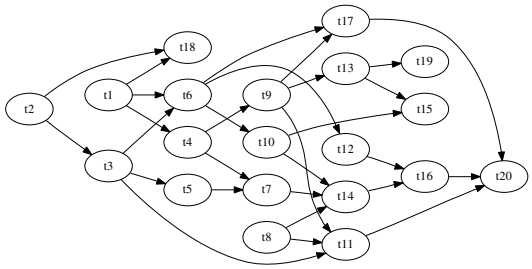

Figure 8: TPG of the i20-15g5 instance.

In summary, the correlation coefficients help us to analyze the kind of solutions proposed by the algorithms. We conclude that the low cost solutions are obtained by assigning most of the project workload to the employee with lowest salary. As the project duration is decreased, however, the other employees are assigned more and more workload. Concerning the tasks, we observe that the amount of human resources for those tasks that must be performed in parallel is increased at the same time when all the tasks are required to finish at the same time. If this requirement is not necessary, the parallel tasks compete and the amount of resources is shared between the tasks. When two tasks are consecutive in the TPG, the human resources can be distributed between the two tasks with a small influence in cost or time, leading to different ways of managing the project.

6. CONCLUSION AND FUTURE WORK

This paper have approached the SPS problem with its natural multi-objective formulation in which both the project cost and its duration have to be minimized. Five multi-objective metaheuristics have been used, namely NSGA-II, SPEA2, PAES, MOCell, and GDE3. They all have been evaluated over 36 SPS instances that cover different scenarios that might appear in software projects. The results of the experimental study allowed us to get valuable conclusions, which are, however, limited by the fact that we only used a finite set of instances.

The results have been analyzed in two complementary ways. First, the HV indicator and the attainment surfaces have been used to evaluate the quality of the five MO algorithms. This has shown that PAES has reached the best approximated Pareto fronts. MOCell has also performed well on the hypervolume indicator. The two most widely used algorithms in the literature, NSGA-II and SPEA2, have reported the lowest (worst) HV values.

The attainment surfaces have allowed us to distinguish the region of the objective space that is better explored by the MO algorithms. PAES has shown to outperform the other algorithms on regions of the objective space with low project cost and long duration, whereas NSGA-II, SPEA2, MOCell, and GDE3 have reached enhanced project schedulings with high project cost but short duration.

Second, an analysis of the resulting project scheduling of the algorithms has been provided. It has been based on computing the Spearman rank correlation coefficients between project cost, project duration, tasks, and dedication of employees. This analysis has not only shown the common sense inverse correlation between the project cost and its duration, but also many other fully informative, positive/negative correlations that emerge from the features of the given instances.

As future work we plan to advance in different lines that are summarized in the following. First, the formulation of the problem could be changed in order to include more realistic situations. Second, one important aspect in project scheduling in general, and in software projects in particular, is the robustness of the solution. Project managers prefer not only good schedulings but also schedulings that can accommodate small changes in the parameters of the problem without a large variation in their cost or makespan. These changes in the parameters of the problem can be a variation in the staff or in the task effort. In the case of software projects it is quite usual that the effort of the tasks is not well estimated, thus a robust solution would be valuable for a project manager. Third, according to the results of the multi-objective algorithms we conclude that some of the algorithms are good for some instances or in some regions of the objective space while others are better in other situations. Perhaps the best algorithm for the software project scheduling problem would be one algorithm that combines some of the features of the best performing algorithms. Thus we plan to design hybrid algorithms combining these features.

Acknowledgements

This work has been partially funded by the Spanish Ministry of Science and Innovation and FEDER under contract TIN2008-06491-C04-01 (the M* project). It has also been partially funded by the Andalusian Government under contract P07-TIC-03044 (DIRICOM project).

7. REFERENCES

[1] E. Alba and F. Chicano. Software project management with GAs. *Information Sciences*, 177(11):2380–2401, June 2007.

[2] G. Antoniol, M. Di Penta, and M. Harman. A robust search-based approach to project management in the presence of abandonment, rework, error and uncertainty. In *10th Int. Symp. on the Software Metrics (METRICS '04)*, pages 172–183, 2004.

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

[4] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2nd edition, 2007.

[5] K. Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001.

[6] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[7] J. Duggan, J. Byrne, and G. Lyons. A task allocation optimizer for software construction. *IEEE software*, Jan 2004.

[8] J. Durillo, A. Nebro, and E. Alba. The jmetal framework for multi-objective optimization: Design and architecture. In *IEEE Congress on Evolutionary Computation, CEC'2010*, pages 4138 – 4325, 2010.

[9] S. Gueorguiev, M. Harman, and G. Antoniol. Software project planning for robustness and completion time in the presence of uncertainty using multi objective search based software engineering. In *GECCO 2009*, pages 1673–1680, 2009.

[10] T. Hanne and S. Nickel. A multiobjective evolutionary algorithm for scheduling and inspection planning in software development projects. *European Journal of Operational Research*, Jan 2005.

[11] J. Knowles. A summary-attainment-surface plotting method for visualizing the performance of stochastic multiobjective optimizers. In *ISDA'05*, pages 552 – 557, 2005.

[12] J. Knowles and D. Corne. The pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization. In *CEC'99*, pages 98–105.

[13] J. Knowles and D. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149 – 172, 2000.

[14] S. Kukkonen and J. Lampinen. GDE3: The third evolution step of generalized differential evolution. In *IEEE Congress on Evolutionary Computation (CEC'2005)*, pages 443 – 450, 2005.

[15] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. A cellular genetic algorithm for multiobjective optimization. In *NICSO 2006*, pages 25–36, 2006.

[16] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. Design issues in a multiobjective cellular genetic algorithm. In *EMO 2007*, LNCS 4403, pages 126–140, 2007.

[17] D. J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC; 4th edition, 2007.

[18] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithms. In *EUROGEN 2001*, pages 95–100, 2002.

[19] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.