A Multi-objective Approach to the Software Project Scheduling Problem

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 managed optimally 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, several algorithms from the domain of the metaheuristics have been used to address this problem. They have been thorougly 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 analized in order to show their relevant features. The multi-objective approach has been also compared to the single-objective one, in which the objectives are weighted according to their importance for the company. 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

General Terms

Theory, Algorithms

Keywords

Fitness Landscapes, Elementary Landscapes, Quadratic Assignment Problem

1. INTRODUCTION

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

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 [2]. Because of their computational complexity, this problem cannot be efficiently solved using complete algorithms since enumerating the entire search space would require years of computation. Metaheuristics become the choice here.

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 [10]. 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 [9]. 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 [11], SPEA2 [35] and PAES [20]— plus two recent algorithms —MOCell [26] and GDE3 [22]— 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: What 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 in which these solutions are located. 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

No much work has been devoted on 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.* [15] 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 [3]. The robustness of a solution is measured as the variation of the total developing 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.* [13] 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 [16] 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.

The previous papers focus on multi-objective approaches of software project scheduling problems. If we extend the scope we quickly find a related problem that is not specifically formulated for software engineering projects but that can be applied in some cases to these projects. We talk about the Resource Constraint Project Scheduling Problem (RCPSP). In RCPSP, there is a set of activities that must be performed with the help of some resources. There are different kinds of resources that can be classified in renewable (e.g., people) or non-renewable (e.g., paper). The activities are related by precedence constraints and can be performed in different ways, called *modes* in the formulation. Each mode of performance has a different duration and requires a different amount of each kind of resource (mode 1 could require two pieces of paper and three people while mode 2 could require four pieces of paper and two people). RCPSP has been extensively tackled in the literature in which it has been solved with both exact [24] and metaheuristic techniques [27].

We can also find some multi-objective approaches of RCPSP or small variants. Al-Fawzan and Haouari [1] use a multi-objective tabu search (MOTS) to solve the single-mode RCPSP (only one mode of performance is possible for each activity). The two objectives of the multi-objective formulation are to minimize the makespan and to maximize the robustness. In the mentioned paper, the robustness is measured as the sum of the time that the tasks can be extended without changing the due date. Abdelaziz *et al.* [6] use a multi-objective ant system (MOASA) to solve a multi-mode RCPSP with renewable resources. For each pair resource-task, they include in the definition of the problem the probability of success for the task (after performing the task with the resource). The objectives are to minimize the makespan and the cost, as well as to maximize the probability of success. Wang *et al.* [34] use NSGA-II to solve a multi-mode RCPSP with renewable resources and application to an agriculture scheduling problem. Viana and Pinho de Sousa [33] solve also a multi-mode version of the RCPSP with three objectives: minimize makespan, minimize the weighted sum of the activity delays, and minimize a relaxed version of some constraints. The authors use a multi-objective version of Simulated Annealing (MOSA) and MOTS. Belfares *et al.* [5] solve a scheduling problem with military application: the course of action development stage of the military operational planning process. This problem can be formulated as a multi-objective RCPSP. The authors solve this problem using an MOTS.

An interesting scheduling problem proposed by Pathak *et al.* [28] allows a non-linear relationship between the cost and the time required to perform a task. In their approach, a multi-objective genetic algorithm is used to minimize makespan and cost. In [28] the relationship between cost and time for each individual activity is modelled by using fuzzy logic. In a later work [29], they propose the use of an artificial neural network as the model for these relationships.

3. THE SOFTWARE PROJECT SCHEDULING PROBLEM

As stated previously, the problem addressed here lies in assigning employees to tasks. The details of the problem are presented next. The reader is referred to [2] for a detailed justification of the problem formulation, including examples, and preliminary results with a genetic algorithm.

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 ratio between the amount of hours dedicated to the project and the full working day length of the employee.

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 working day 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 $\mathbf{X}$.

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 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,

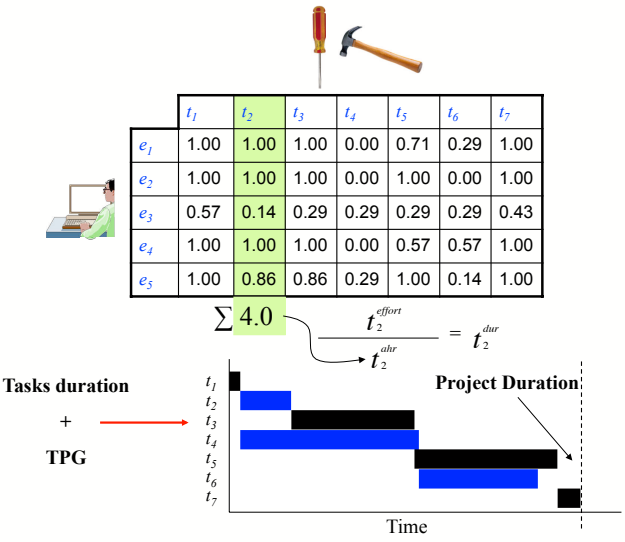

Figure 1: A tentative solution for a sample project. Using the task durations and the TPG, the Gantt diagram of the project can be computed.

that is:

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

From the right-most expression it is clear that:

$$\sum_{i=1}^{E} e_i^{par} = 1 \tag{4}$$

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:

$$t_j^{start} = \begin{cases} 0 & \text{if } \nexists t_i, (t_i, t_j) \in A \\ \max\limits_{t_i, (t_i, t_j) \in A} \{t_i^{end}\} & \text{otherwise} \end{cases} \tag{5}$$

$$t_j^{end} = t_j^{start} + t_j^{dur} \tag{6}$$

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

$$p_{dur} = \max_{j=1}^{T} \{t_j^{end}\} \tag{7}$$

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:

$$p_{cost} = \sum_{i=1}^{E} \sum_{j=1}^{T} e_i^{salary} \cdot x_{ij} \cdot t_j^{dur} \tag{8}$$

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:

$$t_j^{ahr} > 0 \quad \forall j \in \{1, 2, \ldots, T\} \tag{9}$$

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

$$t_j^{skills} \subseteq \bigcup_{\{i|x_{ij}>0\}} e_i^{skills} \quad \forall j \in \{1,2,\ldots,T\} \tag{10}$$

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

$$e_i^{work}(\tau) = \sum_{\{j|t_j^{start} \leq \tau \leq t_j^{end}\}} x_{ij} \tag{11}$$

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:

$$e_i^{over} = \int_{\tau=0}^{\tau=p_{dur}} ramp(e_i^{work}(\tau) - e_i^{maxded}) d\tau \tag{12}$$

where *ramp* is the function defined by:

$$ramp(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \tag{13}$$

The definite integral in (12) 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.:

$$p_{over} = \sum_{i=1}^{E} e_i^{over} \tag{14}$$

4. ALGORITHMS

In this section we provide a general background on multi-objective optimization and describe the details of the multi-objective algorithms used in the experimental section.

4.1 Multi-objective Background

A general multi-objective optimization problem (MOP) can be formally defined as follows (we assume minimization without loss of generality).

DEFINITION 1 (MOP). *Find a vector* $\vec{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ *which satisfies the* m *inequality constraints* $g_i(\vec{x}) \geq 0, i = 1,2,\ldots,m$, *the* p *equality constraints* $h_i(\vec{x}) = 0, i = 1,2,\ldots,p$, *and minimizes the vector function* $\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x}))$, *where* $\vec{x} = (x_1, x_2, \ldots, x_n)$ *is the vector of decision variables.*

The set of all values satisfying the constraints defines the *feasible region* Ω and any point $\vec{x} \in \Omega$ is a *feasible solution.*

As mentioned before, we seek for the *Pareto Optimal Set.* Before its definition some concepts must be introduced.

DEFINITION 2 (PARETO OPTIMALITY). *A point* $\vec{x}^* \in \Omega$ *is Pareto optimal if for every* $\vec{x} \in \Omega$ *and* $I = \{1,2,\ldots,k\}$ *either* $\forall_{i \in I} f_i(\vec{x}) = f_i(\vec{x}^*)$ *or there is at least one* $i \in I$ *such that* $f_i(\vec{x}) > f_i(\vec{x}^*)$.

This definition states that $\vec{x}^*$ is Pareto optimal if no feasible vector $\vec{x}$ exists which would improve some criterion without causing a simultaneous worsening in at least another criterion. Other important definitions associated with Pareto optimality are the following:

DEFINITION 3 (PARETO DOMINANCE). *A vector* $\vec{u} = (u_1, \ldots, u_k)$ *is said to dominate* $\vec{v}=(v_1, \ldots, v_k)$ *(denoted by* $\vec{u} \preccurlyeq \vec{v}$*) if and only if* $\vec{u}$ *is partially smaller than* $\vec{v}$, *i.e.,* $\forall i \in I,\ u_i \leq v_i \ \wedge\ \exists i \in I :\ u_i < v_i$.

DEFINITION 4 (PARETO OPTIMAL SET). *For a given MOP* $\vec{f}(\vec{x})$, *the Pareto optimal set is defined as* $\mathcal{P}^* = \{\vec{x} \in \Omega | \neg \exists \vec{x}' \in \Omega, \vec{f}(\vec{x}') \preccurlyeq \vec{f}(\vec{x})\}$.

DEFINITION 5 (PARETO FRONT). *For a given MOP* $\vec{f}(\vec{x})$ *and its Pareto optimal set* $\mathcal{P}^*$, *the Pareto front is defined as* $\mathcal{PF}^* = \{\vec{f}(\vec{x}) | \vec{x} \in \mathcal{P}^*\}$.

Obtaining the Pareto front of a MOP is the main goal of multi-objective optimization. However, given that a Pareto front can contain a large number of points, a good solution can only include a limited number of them, which should be as close as possible to the true Pareto front (*convergence*), as well as being uniformly spread (*diversity*); otherwise, they would not be very useful to the decision maker. Closeness to the Pareto front, or convervence, ensures that we are dealing with (almost) optimal solutions, while a uniform spread of the solutions, or diversity, means that we have made a good exploration of the search space and no regions are left unexplored.

4.2 MO Solvers

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) [4] and, by far, they are the most popular metaheuristics for solving MOPs [9, 10] because of their ability of finding a set of trade-off solutions in one single run. EAs are based on the concept of survival of the fittest. At each generation (iteration) t (see Algorithm 1), an EA operates on a population of individuals $P(t)$, each one encoding a tentative solution, thus searching in many regions of the problem space at the same time. Each individual is a string of symbols encoding a solution for the problem (*genotype*) and has an associated fitness value which is computed by the objective function. This fitness function is aimed at ranking the quality of the evaluated individual with respect to the rest of the population. The application of stochastic variation operators, such as mixing parts of two strings (*crossover*) or randomly changing their contents (*mutation*), leads this population towards the fittest regions in an iterative manner. The algorithm finishes when a stopping condition is met (e.g., an optimum is found or a number of function evaluations has been carried out). A set of nondominated solutions is maintained during the execution of the algorithm. This set, denoted with PF in Algorithm 1, is usually called *external archive* in the specialized literature. In order to deal with Pareto optimality, EAs are endowed with special mechanisms that allow the algorithms not only to search for optimal solutions, but also to spread them out along the Pareto front. We have used the implementation of these algorithms provided by jMetal [14], an object-oriented Java-based framework aimed at the development, experimentation, and study of metaheuristics for solving multi-objective optimization problems[1].

[1]jMetal is freely available for download at the following URL: `http://jmetal.sourceforge.net/`.

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
```

Starting from the pseudocode of a generic multi-objective EA included in Algorithm 1, the main features of the algorithms used in this work are outlined. For a detailed description, interested readers are referred to the references provided for each solver.

Both NSGA-II [11] and SPEA2 [35] use the scheme of Algorithm 1. As evolutionary operators, they have adopted binary tournament selection, simulated binary crossover, and polynomial mutation [10]. They differ one each other in the mechanism used to keep a diverse approximated Pareto front. PAES [21] 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 [25] 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 [22] also matches the scheme of Algorithm 1, but the typical evolutionary operators are replaced with 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 [10]. It is based on considering feasible solutions as better solutions than non-feasible ones. Among non-feasible solutions, those 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 three sections we answer the research questions in turn.

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 approaches have been used to do so in this article: the hypervolume indicator [36] and the attainment surfaces [19].

Since the exact Pareto fronts of the SPS instances are unknown, a quality indicator which does not require this information beforehand is mandatory. This is why the *hypervolume* indicator, HV, has been chosen in this work. HV is one of the most well-known and widely used indicators in the literature since it allows to measure at once the two desirable features of Pareto fronts, that is, convergence and diversity.

HV calculates the volume (in the objective space) covered by members of a nondominated set of solutions Q for problems where all objectives are to be minimized. Mathematically, for each solution $i \in Q$, a hypercube v_i is constructed with a reference point W plus the solution i as the diagonal corners of the hypercube. The reference point can simply be found by constructing a vector of the worst objective function values. Thereafter, a union of all hypercubes is computed and its hypervolume (HV) is calculated:

$$HV = volume\left(\bigcup_{i=1}^{|Q|} v_i\right). \tag{15}$$

Higher values of the hypervolume metrics are desirable. Since this indicator is not free from an arbitrary scaling of the objectives, the following normalization procedure has been performed in order to guarantee a fair comparison among the algorithms. First, for each problem instance, all the nondominated solutions obtained in all the executions of all the algorithms involved in the experimental study are collected into one single set. Dominated solutions are then removed. This way, we obtain 36 nondominated sets, one for each SPS instance. These resulting sets can be considered the reference Pareto fronts to be used as the basis for the normalization. Let $f_{max} = (f_1^{max}, f_2^{max}, \ldots, f_k^{max})$ and $f_{min} = (f_1^{min}, f_2^{min}, \ldots, f_k^{min})$ be the maximum and minimum values of the k objectives of a given MOP, respectively. Every nondominated solution $f = (f_1, f_2, \ldots, f_k)$ computed by any algorithm for this MOP is normalized, assuming the minimization of all the objectives, by:

$$f_i^{norm} = \frac{f_i - f_i^{min}}{f_i^{max} - f_i^{min}} \text{ if } f_i^{min} \leq f_i \leq f_i^{max}, i = 1, \ldots, k \tag{16}$$

that is, all the objectives of the approximated fronts reached by algorithms are mapped to [0.0,1.0] so that their scales have all the same influence on the HV indicator. It is worth mentioning that, if any of the objective value of f is out of the limits of the reference Pareto front (i.e., either $f_i < f_i^{min}$ or $f_i > f_i^{max}$), then f has no contribution to the HV of the front it is included. This issue makes it possible for a given nondominated set to obtain a HV value of zero when all its solutions are out of the aforementioned limits.

The HV indicator has the advantage of summarizing an entire front into one single scalar value that allows the performance of different algorithms to be compared. However, 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.

In the related literature, the cost-time trade-off is usually presented by showing one of the approximated Pareto fronts obtained in one single run of a given algorithm. However, the optimization algorithms used are stochastic and therefore there is no warranty that the same result is obtained after a new run of the algorithm. Thus, a single run of a stochastic algorithm gives no information about the average

performance of the algorithm. We need a way of representing the results of a multi-objective algorithm that allows us to observe the expected performance and its variability, 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) [19]. 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:

$$\alpha(z) = \frac{1}{r} \sum_{i=1}^{r} I(A^i \preceq \{z\}) \quad (17)$$

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 [19]. 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" (see Fig. 2). When the number of objectives is one, the 50%-attainment surface is the median and the "interquartile region" is the interquartile range.

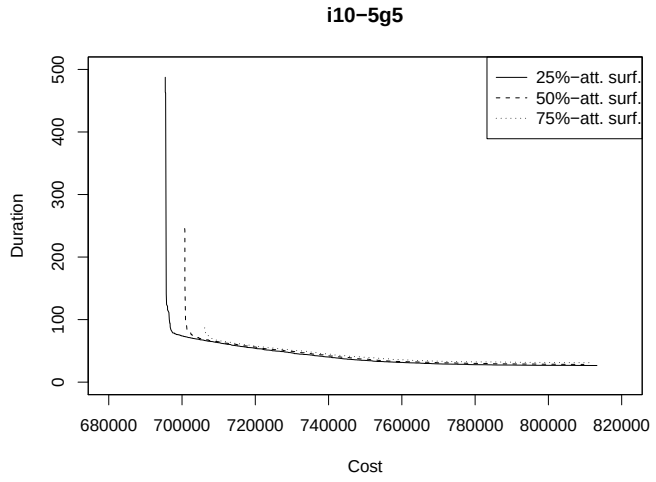

Figure 2: Attainment surfaces for PAES in the i10-5g5 instance. We show the 25%-, 50%-, and 75%-attainment surfaces.

The attainment surfaces provides an engineer with a tool for evaluating the variability of an algorithm for the problem at hand. For example, we can observe in Fig. 2 that the region between the 25%- and the 75%-attainment surfaces is very narrow when the development time is short (and the cost high). This means that the approximated Pareto fronts obtained by the algorithm provide a very accurate value for the development time when the cost is higher than 710,000 currency units. This does not necessarily mean that the time is constant when the cost is high. We can observe in the figure that the time is decreasing. The narrowness is a signal of stability of the algorithm. On the contrary, a wide region means large variability in the approximated Pareto fronts. We can observe a wide interquartile region in Fig. 2 on the left-hand side, before 710,000 currency units. The approximated Pareto fronts proposed by the different independent runs of the algorithm are very different in this region of the objective space. In summary, the variability in the results of one multi-objective algorithm is not reduced to a scalar (as in the single-objective case). This variability depends on the region observed in the objective space. We can find a rich range of possibilities when considering variability in the multi-objective domain. Using attainment surfaces the engineer can analyze and explore this range of possibilities. From a practical point of view, in the SBSE domain, this tool helps the engineer to decide the more suitable multi-objective algorithm for her/his requirements.

As stated before, 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 attaiment surfaces are then computed. In the case of HV, the following statistical analysis has been carried out [32]. First a Kolmogorov-Smirnov test is performed in order to check whether the samples are distributed according to a normal distribution or not. If so, an ANOVA I test is performed; otherwise we perform a Kruskal-Wallis test. 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 such that we allow 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.

As to the solution representation, the algorithms adopt a floating point encoding in which the gene i represent 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. So therefore, the GAs have adopted simulated binary crossover (SBX) and polynomial mutation. The distribution indices for both operators are $\eta_c = 20$ and $\eta_m = 20$, respectively. The crossover probability is $p_c = 0.9$ and the mutation probability is $p_m = 1/L$, where L is the number of decision variables. In PAES we have also used a polynomial mutation operator, with the same distribution index as indicated before. The DE algorithm uses $CR = 0.1$ and $F = 0.5$, as the rates for the DE crossover. A detailed description of the parameter values adopted for our experiments is provided in Table 1. All the probabilities and

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

Parameterization used in NSGA-II [11]	
Population Size	100 individuals
Selection of Parents	binary tournament + binary tournament
Recombination	simulated binary, $p_c = 0.9$
Mutation	polynomial, $p_m = 1.0/L$
Parameterization used in SPEA2 [35]	
Population Size	100 individuals
Selection of Parents	binary tournament + binary tournament
Recombination	simulated binary, $p_c = 0.9$
Mutation	polynomial, $p_m = 1.0/L$
Parameterization used in PAES [20]	
Population Size	1 individual
Mutation	polynomial, $p_m = 1.0/L$
Archive Size	100
Parameterization used in MOCell [26]	
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
Parameterization used in GDE3 [22]	
Population Size	100 individuals
Recombination	Differential Evolution, $CR = 0.1$, $F = 0.5$

distribution indexes are the same as those proposed by the authors of each algorithm.

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 instances that have been previously addressed in [2]. 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-EgS. 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-EpM, 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 employe varies from 6 to 7. In the 36 instances the maximum dedication for all the employees is 1 (full working day). All these instances are publicly available at the URL `http://mstar.lcc.uma.es`.

5.4 MO Solvers Comparison on the SPS Problem

The first set of experiments is devoted to evaluating the five multi-objective metaheuristics on the set of 36 SPS instances. The HV indicator has been used to measure the quality of the resulting approximated Pareto fronts. Instead of including the raw HV median values (which are displayed in Appendix A for reproducibility purposes), Table 2 ranks the algorithms for each instance based on the median of HV over 100 independent runs. Two special notations appear in the table: the gray coloured background has been used to better show the best performing algorithm, whereas the asterisks aside a rank point out that no statistical differences exist between the best solver and the asterisk-marked ones. In the cases in which the HV values are the same for several algorithms, the ranking shown in Table 4 is computed as done in Statistics[2], e.g., this is why SPEA2, MOCEll, and GDE3 have ranked all the fourth in the i30-15g10 instance ($3 + 4 + 5 = 12 \Rightarrow 12/3 = 4$).

The first conclusion that can be drawn from Table 2 is that the approximated Pareto fronts reached by PAES have the higher (better) HV values in most of the instances (25

Table 2: Ranking of the MO solvers based on their HV values.

	Algorithms				
Instances	NSGAII	SPEA2	PAES	MOCell	GDE3
i10-5g5	4	5	3	1	2*
i10-5g10	3	5	4	1	2*
i10-10g5	4	5	1	3*	2*
i10-10g10	4	5	3	1	2*
i10-15g5	4	5	1	3	2
i10-15g10	4	5	1	3*	2*
i20-5g5	4	5	3*	2*	1
i20-5g10	4	5	1	3	2
i20-10g5	4	5	1	2*	3*
i20-10g10	4	5	1	2	3
i20-15g5	4	5	1	2	3
i20-15g10	4	5	1	3	2
i30-5g5	4	5	1	2	3
i30-5g10	2	5	1	3	4
i30-10g5	2	5	1	4	3
i30-10g10	2	4	1	4	4
i30-15g5	2	4	1	4	4
i30-15g10	2	4	1	4	4
i10-5p5	3*	5	4	1	2*
i20-5p5	4	5	3	1	2*
i30-5p5	4	5	1	2	3
i10-10p5	4	5	1	2	3
i20-10p5	4	5	1	2*	3*
i30-10p5	4	5	1	3	2
i10-15p5	4	5	3	1	2*
i20-15p5	4	5	1	3	2
i30-15p5	3.5	3.5	1	3.5	3.5
i10-5p7	3	5	4	1	2*
i20-5p7	4	5	3	1	2*
i30-5p7	4	5	1	2	3
i10-10p7	4	5	3	1	2*
i20-10p7	4	5	1	3	2
i30-10p7	4	5	1	2	3
i10-15p7	4	5	3*	2*	1
i20-15p7	4	5	1	3	2
i30-15p7	3.5	3.5	1	3.5	3.5

[2]Summing up the positions in which the elements should be, and dividing by the number of elements with the same value.

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. Just to shed some light on that, if the rankings over all the problem instances are averaged, PAES scores 1.69 (first), whereas MOCell and GDE3 do 2.33 and 2.53 (second and third). It is worth highlighting that NSGA-II and SPEA2, the most widely used solvers in the literature, have never ranked the first. Having a look at the average rank, they respectively have scored 3.61 and 4.83 (recall that 1 is the best position and 5 the worst). Indeed, based on the HV indicator, SPEA2 has reached the worst approximations to the Pareto fronts in 31 out of the 36 SPS instances.

Table 2 also reveals several interesting facts that help to better characterize the search performed by the best performing algorithms, namely PAES, MOCell, and GDE3. It is clear that PAES have reached the approximated Pareto fronts with the best HV values for most of the addressed instances. If one dives into the raw values of Table 4 in Appendix A, it can be seen that the more complex the instance (more tasks and more employees) the larger the difference in the HV values of the PAES fronts with respect to those of the other four algorithms. As a matter of fact, for these complex instances such as i30-15p5, and i30-15p7, all the algorithms but PAES have reached a median HV value of zero over 100 independent runs, what means that all the nondominated solutions in these approximated sets are out of the limits of the reference Pareto front (see Section 5.1).

However, it seems 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), that also have 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 this regions of the search space. Table 2 also shows that the search capabilities of GDE3 diminishes as the number of tasks gets increased, that is, when the instances become larger. A carefull 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 Figs. 3 to 7, we also show the best known approximated Pareto front (Reference Pareto Front, RPF).

We observed in the previous comparison 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 Fig. 3), it can be easily justified the high HV value obtained by PAES. Indeed, its approximated Pareto fronts are not only near 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.

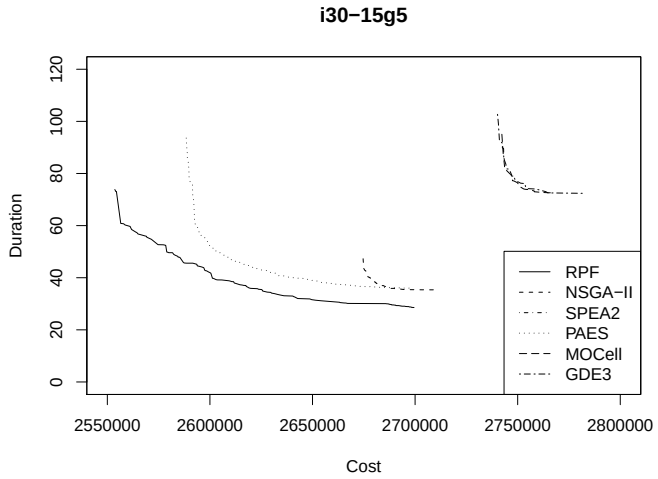

Figure 3: 50%-attainment surfaces of the algorithms in the i30-15g5 instance.

In the instance i10-5g5, the 50%-attainment surfaces of all the algorithms are very close to each other (see Fig. 4). This fact can be also 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 Fig. 5), 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 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 Fig. 6). 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

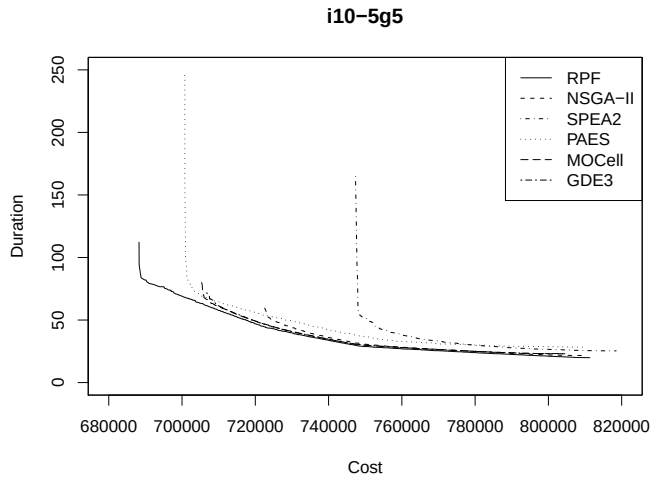

Figure 4: 50%-attainment surfaces of the algorithms in the i10-5g5 instance.

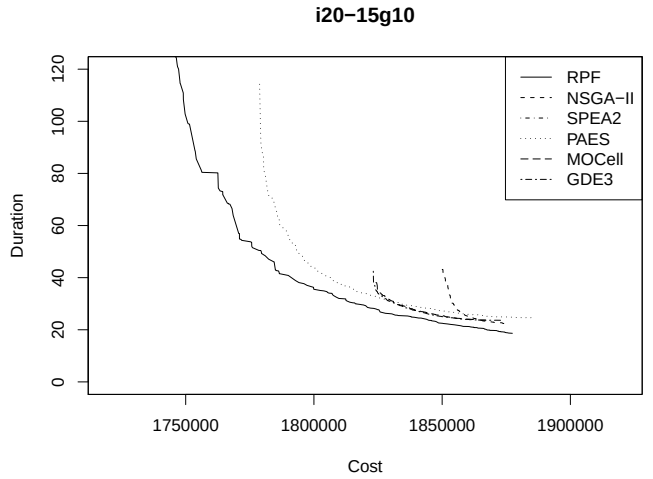

Figure 5: 50%-attainment surfaces of the algorithms in the i20-15g10 instance.

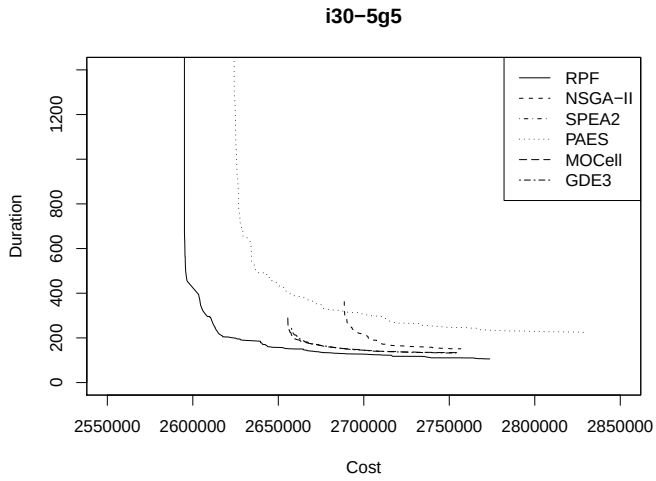

Figure 6: 50%-attainment surfaces of the algorithms in the i30-5g5 instance.

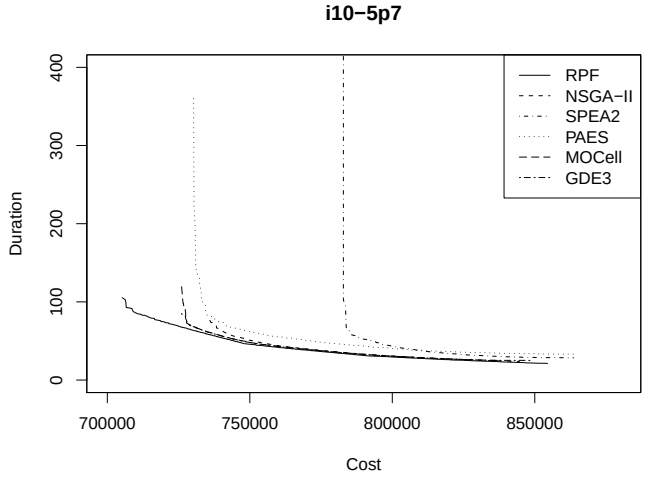

Figure 7: 50%-attainment surfaces of the algorithms in the i10-5p7 instance.

with statistical significance. The scenario shown in Fig. 6 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 (Fig. 7). In this case, the 50%-attainment surface of PAES is clearly dominated by the ones of MOCell, GDE3 and NSGA-II. The HV indicator has reflected this fact by giving a rank of 3 or 4 to PAES (see Table 2). The instances that behave the same 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.

As a summary of this analysis, Table 3 shows the five different scenarios related to the 50%-attainment surface of the algorithms and the instances in which they have emerged.

5.5 In-Depth 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 objetive space. For each proposed solution by one algorithm for one single instance, $E+T$ values have to be analyzed. This means a large amount of data to process and

Table 3: Summary of the five different scenarios found in the instances concerning the 50%-attainment surfaces.

Scenarios	Instances
PAES is clearly the best	i30-10g5, i30-10g10, i30-15g5, i30-15g10, i30-10p5, i30-15p5, i30-15p7
All the best surfaces together	i10-5g5, i20-5p5, i30-5p5, i20-10p7
Crossing surfaces	i20-15g10, i20-15p5, i30-10p7
PAES outperformed in some regions	i10-5g10, i10-10g5, i10-10g10, i10-15g5, i10-15g10, i20-5g5, i20-5g10, i20-10g5, i20-10g10, i20-15g5, i30-5g5, i30-5g10, i10-10p5, i20-10p5, i10-15p5, i20-5p7, i30-5p7, i20-15p7
PAES is dominated	i10-5p7, i10-10p7, i10-15p7, i10-5p5

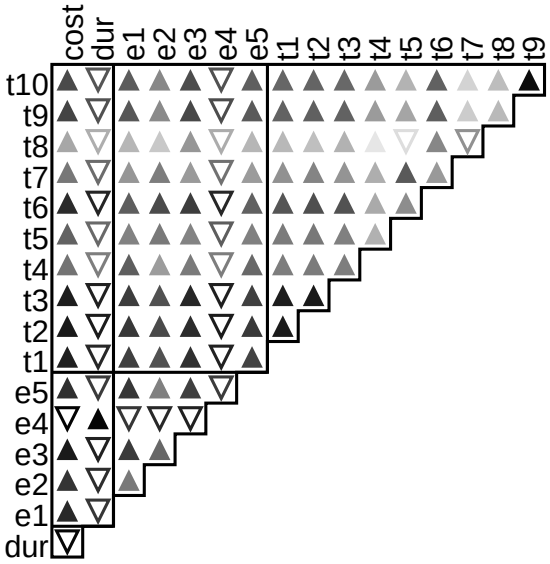

Figure 8: Spearman rank correlation coefficients between p_{cost}, p_{dur}, e_i^{par} and t_j^{ahr} in the i10-5g5 instance for the solutions obtained with PAES.

show. In order to reduce this amount of data without losing interesting information, the following analysis has been performed. First, the focus has been put on the results of just 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 [32] between all the e_i^{par}, t_j^{ahr}, p_{dur}, and p_{cost} are then calculated. Let us start by analyzing the correlations found in the i10-5g5 instance. The correlation coefficients are shown in Fig. 8. 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).

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_1, e_2, e_3 and e_5 the correlation is positive while for e_4 is negative. What does this mean? When the cost of the solutions proposed by the algorithm increases, the participation of e_1, e_2, e_3 and e_5 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_4 is reduced. This does not mean that e_4 spends less time in the project, since we defined participation as a normalized measure; this means that the fraction of workload that is performed by e_4 is reduced because the other employees increase their participation. But the interesting question is, why does this happen? The answer is that e_4 is the cheapest employee, i.e., her/his salary is the lowest one. The algorithm prefers to assign most of the work of the project to e_4 because it earns less money. However, when the project duration is reduced (and the cost increased) the other employees have necessarily to increase their participation. We can also observe a negative correlation between e_4 and the rest of the employees, as expected from the previous discussion.

Let us now turn to the tasks. There is a positive correlation between p_{cost} and the t_j^{ahr}, as we would expect; and a negative correlation between p_{dur} and t_j^{ahr}, also expected. The most interesting correlations regarding the tasks appear when we compare the t_j^{ahr} of different tasks. We observe that t_1, t_2 and t_3 have a high positive correlation. If we take a look to the TPG of the instance (Fig. 9) we notice that these tasks are the first ones to do. It makes no sense to increase the work in one of them and not in the others, since the saved time does not allow a reduction in the total project duration and the project cost could increase. One of such solutions would not be Pareto optimal, and, thus, it would be removed by the algorithm from the population of solutions. The same happens in tasks t_9 and t_{10}; they are the last tasks in the project (no other task depends on them). However, we also observe a negative correlation between t_7^{ahr} and t_8^{ahr}. These tasks are consecutive in the TPG (see Fig. 9) and the negative correlation means that, in general, the human resources move from one task to the other. The consequence of this movement of human resources is that when the t_7^{dur} is decreased t_8^{dur} is increased and vice versa. The net effect is that the sum $t_7^{dur} + t_8^{dur}$ is almost the same, what does not affect p_{dur}. Thus, we conclude that the negative correlation between t_7^{ahr} and t_8^{ahr} must be present in every region of the objective space, that is, this correlation shows that

there exist different ways of obtaining the same time-cost trade-off.

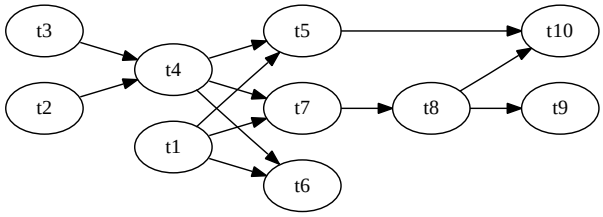

Figure 9: TPG of the i10-5g5 instance.

The trends observed in the i10-5g5 instance are also present in the other instances. However, some other features appear in projects with a higher number of employees and tasks. In the following we analyze the i20-15g5 instance. In Fig. 10 we show the correlation coefficients and in Fig. 11 the corresponding TPG. In this case, e_7, e_8, e_9 and e_{10} are the employees earning less money. We can observe, as in the i10-5g5 instance, that the cheapest employees have a negative correlation with the other employees and with project cost, and a positive correlation with project duration. Furthermore, the cheapest the employee the higher the absolute value of the correlations (darker triangles in the figures).

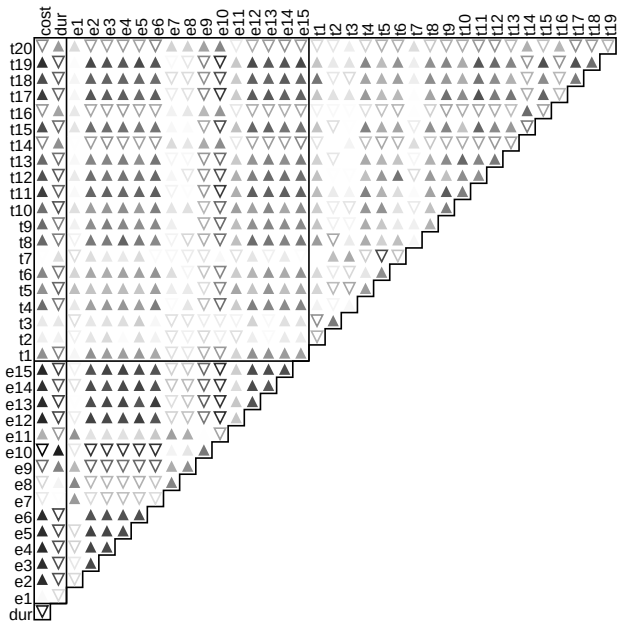

Figure 10: Spearman rank correlation coefficients between p_{cost}, p_{dur}, e_i^{par} and t_j^{ahr} in the i20-15g5 instance for the solutions obtained with PAES.

Concerning the tasks, we observe again positive correlations between two initial tasks: t_1 and t_8. It can be also identified negative correlations between t_1 and t_2, which are also initial tasks. 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 (see Table 3); 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 confirming the previous explanation.

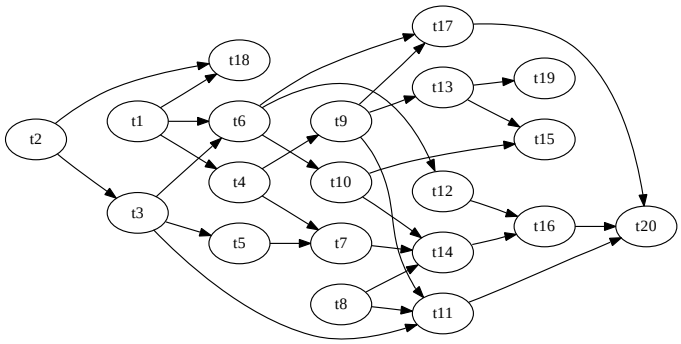

Figure 11: TPG of the i20-15g5 instance.

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 if 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 have been analized in three 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 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 in-depth 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.

Finally, a comparison between the single vs. the multi-objective approach has been developed. The results point out that once the worth of each objective is clear and known, the single-objective algorithms have reported the best results on average over the 36 SPS instances. An interesting finding is that, if one considers only the instances with 30 tasks (the largests ones), the MO algorithms have outperformed the SO ones, even when the weights are known. This fact seems to suggest that the MO approach might be the choice for large, real-world SPS instances. However, more experimentation is required to support this hypothesis.

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. For example, in general, the time required to perform a task is not inversely proportional to the amount of human resources assigned to the task. Thus, we need to consider other relationships between the required time and the manpower. The skill model used in the formulation of this paper can also be improved taking into account levels for each skill. 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 accomodate 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.

7. REFERENCES

[1] M. Al-Fawzan and M. Haouari. A bi-objective model for robust resource-constrained project scheduling. *International Journal of production economics*, Jan 2005.

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

[3] 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 *Proceedings of the 10th International Symposium on the Software Metrics (METRICS '04)*, pages 172–183. IEEE Computer Society, 2004.

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

[5] L. Belfares, W. Klibi, N. Lo, and A. Guitouni. Multi-objectives tabu search based algorithm for progressive resource allocation. *European Journal of Operational Research*, 177(3):1779–1799, Mar 2007.

[6] F. Ben Abdelaziz, S. Krichen, and O. Dridi. A multiobjective resource-constrained project-scheduling problem. In *Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 4724 of *Lecture Notes in Coputer Science*, pages 719–730, Hammamet, Tunisia, 2007. Springer.

[7] C. Blum and A. Roli. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.

[8] C. K. Chang, M. J. Christensen, and T. Zhang. Genetic algorithms for project management. *Annals of Software Engineering*, 11:107–139, 2001.

[9] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, second edition, September 2007. ISBN 978-0-387-33254-3.

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

[11] 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.

[12] T. Dohi, Y. Nishio, and S. Osaki. Optimal software release scheduling based on artificial neural networks. *Annals of Software Engineering*, 8:167–185, 1999.

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

[14] J. J. Durillo, A. J. Nebro, F. Luna, B. Dorronsoro, and E. Alba. jMetal: a Java framework for developing multi-objective optimization metaheuristics. Technical Report ITI-2006-10, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, E.T.S.I. Informática, Campus de Teatinos, 2006.

[15] 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 *Proceedings of the Genetic and Evolutioanry Computation Conference*, pages 1673–1680, Montreal, Canada, July 2009. ACM.

[16] 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.

[17] M. Harman and B. F. Jones. Search based software engineering. *Information and Software Technology*, 43(14):833–839, December 2001.

[18] S. Henninger. Case-based knowledge management tools for software development. *Automated Software Engineering*, 4:319–340, 1997.

[19] J. Knowles. A summary-attainment-surface plotting method for visualizing the performance of stochastic multiobjective optimizers. In *5th International Conference on Intelligent Systems Design and Applications (ISDA'05)*, pages 552 – 557, 2005.

[20] J. Knowles and D. Corne. The pareto archived evolution strategy: A new baseline algorithmfor multiobjective optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 98–105. IEEE Press, 1999.

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

[22] 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.

[23] H.-M. Lee, S.-Y. Lee, T.-Y. Lee, and J.-J. Chen. A new algorithm for applying fuzzy set theory to evaluate the rate of aggregative risk in software development. *Information Sciences*, 153:177–197, July 2003.

[24] A. Mingozzi, V. Maniezzo, S. Ricciardelli, and L. Bianco. An exact algorithm for project scheduling with resource constraints based on a new mathematical formulation. *Management Science*, 44(5):714–729, 1998.

[25] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. A cellular genetic algorithm for multiobjective optimization. In D. A. Pelta and N. Krasnogor, editors, *Proceedings of the Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2006)*, pages 25–36, 2006.

[26] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. Design issues in a multiobjective cellular genetic algorithm. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-Criterion Optimization. 4th International Conference, EMO 2007*, volume 4403 of *Lecture Notes in Computer Science*, pages 126–140. Springer, 2007.

[27] M. Palpant, C. Artigues, and P. Michelon. LSSPER: Solving the resource-constrained project scheduling problem with large neighbourhood search. *Annals of Operations Research*, 131:237–257, 2004.

[28] B. Pathak and S. Srivastava. Moga-based time-cost tradeoffs: Responsiveness for project uncertainties. *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 3085 – 3092, Aug 2007.

[29] B. K. Pathak, S. Srivastava, and K. Srivastava. Neural network embedded multiobjective genetic algorithm to solve non-linear time-cost tradeoff problems of project scheduling. *Journal of Scientific & Industrial Research*, 67:124–131, Mar 2008.

[30] C. L. Ramsey and V. R. Basili. An evaluation of expert systems for software engineering management. *IEEE Trans. on Soft. Eng.*, 15(6):747–759, June 1989.

[31] M. Ronchetti, G. Succi, W. Pedrycz, and B.Russo. Early estimation of software size in object-oriented environments a case study in a CMM level 3 software firm. *Information Sciences*, 176(5):475–489, March 2006.

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

[33] A. Viana and J. P. de Sousa. Using metaheuristics in multiobjective resource constrained project scheduling. *European Journal of Operational Research*, 120(2):359–374, Jan 2000.

[34] H. Wang, D. Lin, and M.-Q. Lid. A competitive genetic algorithm for resource-constrained project scheduling problem. *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, 5:2945 – 2949 Vol. 5, Jul 2005.

[35] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithms. In K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, editors, *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, 2002.

[36] 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.

APPENDIX

A. VALUES OF THE HV INDICATOR

Table 4 includes the raw values of the HV indicator that have been computed after the normalization procedure explaned in Section 5.1.

Table 4: HV of the MO solvers. Median and IQR values.

	NSGAII	SPEA2	PAES	MOCell	GDE3
i10-5g5	$0.649_{0.089}$	$0.452_{0.099}$	$0.692_{0.038}$	$0.736_{0.046}$	$0.726_{0.053}$
i10-5g10	$0.746_{0.069}$	$0.507_{0.081}$	$0.746_{0.072}$	$0.784_{0.030}$	$0.778_{0.039}$
i10-10g5	$0.418_{0.076}$	$0.271_{0.109}$	$0.643_{0.118}$	$0.593_{0.062}$	$0.593_{0.068}$
i10-10g10	$0.457_{0.056}$	$0.167_{0.093}$	$0.507_{0.124}$	$0.651_{0.082}$	$0.647_{0.059}$
i10-15g5	$0.281_{0.070}$	$0.134_{0.211}$	$0.667_{0.161}$	$0.531_{0.082}$	$0.552_{0.076}$
i10-15g10	$0.256_{0.074}$	$0.000_{0.077}$	$0.576_{0.118}$	$0.539_{0.095}$	$0.549_{0.128}$
i20-5g5	$0.423_{0.091}$	$0.000_{0.107}$	$0.592_{0.152}$	$0.620_{0.049}$	$0.621_{0.065}$
i20-5g10	$0.623_{0.079}$	$0.112_{0.158}$	$0.740_{0.104}$	$0.697_{0.044}$	$0.702_{0.040}$
i20-10g5	$0.380_{0.099}$	$0.000_{0.000}$	$0.667_{0.127}$	$0.664_{0.067}$	$0.662_{0.069}$
i20-10g10	$0.277_{0.076}$	$0.000_{0.000}$	$0.690_{0.108}$	$0.442_{0.051}$	$0.442_{0.062}$
i20-15g5	$0.139_{0.075}$	$0.000_{0.000}$	$0.555_{0.209}$	$0.370_{0.073}$	$0.370_{0.070}$
i20-15g10	$0.178_{0.072}$	$0.000_{0.000}$	$0.606_{0.093}$	$0.356_{0.054}$	$0.362_{0.051}$
i30-5g5	$0.475_{0.147}$	$0.000_{0.000}$	$0.829_{0.124}$	$0.666_{0.085}$	$0.655_{0.083}$
i30-5g10	$0.478_{0.100}$	$0.000_{0.000}$	$0.781_{0.129}$	$0.330_{0.091}$	$0.326_{0.066}$
i30-10g5	$0.296_{0.096}$	$0.000_{0.000}$	$0.754_{0.098}$	$0.143_{0.239}$	$0.174_{0.216}$
i30-10g10	$0.290_{0.081}$	$0.000_{0.000}$	$0.669_{0.109}$	$0.000_{0.000}$	$0.000_{0.000}$
i30-15g5	$0.137_{0.072}$	$0.000_{0.000}$	$0.535_{0.095}$	$0.000_{0.000}$	$0.000_{0.000}$
i30-15g10	$0.008_{0.041}$	$0.000_{0.000}$	$0.559_{0.158}$	$0.000_{0.000}$	$0.000_{0.000}$
i10-5p5	$0.663_{0.060}$	$0.538_{0.082}$	$0.601_{0.059}$	$0.680_{0.060}$	$0.680_{0.062}$
i20-5p5	$0.477_{0.092}$	$0.000_{0.021}$	$0.515_{0.258}$	$0.618_{0.069}$	$0.617_{0.066}$
i30-5p5	$0.539_{0.096}$	$0.000_{0.000}$	$0.795_{0.115}$	$0.590_{0.059}$	$0.586_{0.069}$
i10-10p5	$0.497_{0.099}$	$0.202_{0.135}$	$0.753_{0.133}$	$0.667_{0.080}$	$0.661_{0.068}$
i20-10p5	$0.311_{0.058}$	$0.000_{0.000}$	$0.616_{0.114}$	$0.580_{0.043}$	$0.575_{0.040}$
i30-10p5	$0.145_{0.087}$	$0.000_{0.000}$	$0.484_{0.206}$	$0.236_{0.092}$	$0.245_{0.106}$
i10-15p5	$0.330_{0.092}$	$0.066_{0.156}$	$0.527_{0.077}$	$0.583_{0.055}$	$0.581_{0.059}$
i20-15p5	$0.347_{0.089}$	$0.000_{0.000}$	$0.625_{0.090}$	$0.485_{0.045}$	$0.488_{0.045}$
i30-15p5	$0.000_{0.000}$	$0.000_{0.000}$	$0.713_{0.084}$	$0.000_{0.000}$	$0.000_{0.000}$
i10-5p7	$0.660_{0.062}$	$0.387_{0.079}$	$0.552_{0.080}$	$0.695_{0.037}$	$0.693_{0.036}$
i20-5p7	$0.517_{0.151}$	$0.000_{0.097}$	$0.679_{0.123}$	$0.758_{0.090}$	$0.754_{0.080}$
i30-5p7	$0.456_{0.170}$	$0.000_{0.000}$	$0.815_{0.120}$	$0.564_{0.107}$	$0.553_{0.087}$
i10-10p7	$0.401_{0.078}$	$0.373_{0.126}$	$0.572_{0.078}$	$0.666_{0.114}$	$0.660_{0.091}$
i20-10p7	$0.222_{0.073}$	$0.000_{0.000}$	$0.580_{0.092}$	$0.496_{0.060}$	$0.505_{0.061}$
i30-10p7	$0.092_{0.052}$	$0.000_{0.000}$	$0.482_{0.178}$	$0.237_{0.053}$	$0.226_{0.059}$
i10-15p7	$0.415_{0.067}$	$0.022_{0.009}$	$0.643_{0.090}$	$0.654_{0.073}$	$0.669_{0.063}$
i20-15p7	$0.189_{0.088}$	$0.000_{0.000}$	$0.609_{0.081}$	$0.404_{0.053}$	$0.406_{0.070}$
i30-15p7	$0.000_{0.000}$	$0.000_{0.000}$	$0.528_{0.188}$	$0.000_{0.000}$	$0.000_{0.000}$