The Software Project Scheduling Problem: A Scalability Analysis of Multi-objective Metaheuristics

Francisco Luna[a,*], David L. González-Álvarez[b], Francisco Chicano[c], Miguel A. Vega-Rodríguez[b]

[a] *Dept. Computer Science, University Carlos III, Madrid (Spain)*
[b] *Dept. Technologies of Computers and Communications, University of Extremadura, Cáceres (Spain)*
[c] *Dpto. de Lenguajes y Ciencias de la Computación, University of Málaga, Málaga (Spain)*

Abstract

Computer aided techniques for scheduling software projects is a crucial step in the software development process within the highly competitive software industry. The Software Project Scheduling (SPS) problem relates to the decision of who does what during a software project lifetime, thus involving mainly both people-intensive activities and human resources. Two major, conflicting goals arise when scheduling a software project: reducing both its cost and duration. A multi-objective approach is therefore the natural way of facing the SPS problem. As companies are getting involved in larger and larger software projects, there is an actual need of algorithms that are able to deal with the tremendous search spaces imposed. In this paper we analyze the scalability of eight multi-objective algorithms when they are applied to the SPS problem using instances of increasing size. The algorithms are classical algorithms from the literature (NSGA-II, PAES, and SPEA2) and recent proposals (DEPT, MOCell, MOABC, MO-FA, and GDE3). From the experimentation conducted, the results suggest that PAES is the algorithm with the best scalability features.

Keywords: Software project scheduling, scalability analysis, multi-objective optimization

1. Introduction

In the current software industry, the market is becoming highly competitive, so software companies have to elaborate accurate project plans in order to have success. In order to plan a software project, companies need, on the one hand, to estimate the project workload (using for example the COCOMO model [1])

*Corresponding author
Email addresses: fluna@inf.uc3m.es (Francisco Luna), dlga@unex.es (David L. González-Álvarez), chicano@lcc.uma.es (Francisco Chicano), mavega@unex.es (Miguel A. Vega-Rodríguez)

and, on the other hand, to establish the project schedule and to assign resources to tasks. Tasks may be anything from maintaining documents to typing source code, and resources include people, machines, time, etc. Like other similar projects (e.g., civil engineering problems), scheduling and staffing management might be performed with traditional techniques such as Program Evaluation and Review Technique (PERT), Critical Path Method (CPM), and those enclosed within the class known as the Resource-Constrained Project Scheduling Problem (RCPSP) [2]. Although relevant and helpful, these methods are also becoming hardly applicable to the today's software projects, specially PERT/CPM, which are not applicable to the complex networks of activities and their precedence and durations that usually arise [3]. What makes the difference of software projects from those from other areas (production scheduling, RCPSP, etc.) is that only one type of resources are allocated, i.e., employees, having each of them different skills and being able to get involved in several tasks during a working day [4]. Employees' salary is also important as it allows the project to be assigned with a cost (and not only its duration).

When a software company starts scheduling any given project, two main goals are always in conflict: reducing the project cost, that is, completing the entire development at the lowest investment; and, minimizing the project duration so that new projects can be addressed. Of course, these two goals have a direct impact on the company's income. In this paper we focus on the assignment of employees to tasks in a software project so as to minimize these two objectives, that is, it is a multi-objective optimization problem, which has been called the Software Project Scheduling (SPS) problem [5]. Because of their computational complexity and the size of actual projects, this problem cannot be efficiently solved using complete algorithms since enumerating the entire search space would require years of computation [6, 7]. Metaheuristics [8] become the choice here. These are a broad family of approximate techniques that can be considered as general search algorithms, including —but not restricted to— Evolutionary Algorithms, Ant Colony Optimization, Particle Swarm Optimization, Simulated Annealing, Tabu Search, and Iterated Local Search. As opposed to exact techniques, metaheuristics do not guarantee to find optimal solutions to the problems, but they allow to reach good solutions in a reasonable amount of time. A metaheuristic can be defined as a high level strategy which controls a number of subordinated techniques (usually heuristics) in the search for an optimum. Using this kind of algorithms does make sense when no efficient algorithm is known to find the optimal solutions or when the complete enumeration is not viable due to time or memory constraints. Researchers have therefore used these algorithms to solve software engineering optimization problems, as it is clearly shown in [9], where a comprehensive review and analysis of the literature is carried out.

The SPS problem is multi-objective in nature [10] and such a formulation has been used (no aggregation of the objective values into one single value has been considered). 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 [11]. 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 [12].

Previous works in the literature have addressed the multi-objective SPS problem with metaheuristics [10], where 36 instances with up to 30 tasks and 15 employees have been tackled. But actual, relevant, contemporary software projects involve both more people and more tasks. This work evaluates the issue of the scalability of eight multi-objective metaheuristics for the SPS problem, three classical methods —NSGA-II [13], SPEA2 [14], and PAES [15]— plus five novel algorithms —DEPT [16], MO-FA [17], MOABC [18], MOCell [19], and GDE3 [20]— on a set of 36 instances with an exponential increase in both the number of tasks (from 16 to 512) and the number of employees (from 8 to 256). Two quality indicators, the hypervolume (HV) [21] and the attainment surfaces [22], have been used to measure the quality of the resulting Pareto fronts. This piece of research has to be considered as an extension of a previous conference publication [23] and, as a consequence, we want to clearly state the novel contributions included here. They follow two different lines: on the one hand, we have included four new algorithms in the comparison (SPEA2, MOABC, MOCell, and GDE3, i.e., doubling the experimentation conducted) to provide the reader with insights about their scalability capabilities; on the other hand, we have thoroughly analized 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 schedules.

The paper is structured as follows. The next section provides the reader with the formulation of the SPS problem. Section 3 briefly describes the eight multi-objective metaheuristics used. The experimentation performed to assess the performance of these algorithms is detailed in Section 4. In Section 5 we show how a project manager can profit from the results of this paper and Section 6 includes the main conclusions of this work and devises the future lines for further research.

2. The SPS Problem

We follow here the same formulation proposed in [5]. 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 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, which 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) that 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, which are defined in Equations (7) and (8), respectively. The constraints are: first, that each task must be performed by at least one person; second, the set of required skills of a task must be included in the union of the skills of the employees performing the task; and, third, no employee must exceed her/his maximum dedication to the project.

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. In order 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, 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:

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

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

The project duration, p_{dur}, 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:

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

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

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

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.:

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

This total overwork must be zero:

$$p_{over} = 0 \tag{15}$$

3. Algorithms

In this section, we briefly describe the eight metaheuristics used in this study, namely NSGA-II, SPEA2, PAES, DE-PT, MO-FA, MOABC, MOCell, and GDE3. They all are either population-based metaheuristics [8], i.e., they operate on a set of solutions at every iteration, or include an external archive for storing the nondominated solutions found during the search (e.g., PAES is a trajectory-based metaheuristic using such a mechanism), or both (e.g., SPEA2, MOCell are population-based algorithms with external archive). A general template for a multi-objective metaheuristic is displayed in Algorithm 1. The general operation of these algorithms begins by generating the initial solutions, S (usually in a random manner), and updating the set of non-dominated solutions found in this first sampling, A (lines 1 to 3). Then, the search loop starts. It lies in stochasticly varying the solutions included in S and A, and generating a (hopefully improved) new set of solutions (line 7) from which those that are non-dominated are retrieved (line 9). The matching of this general scheme on the eight algorithms used in this work is briefly presented in the following subsections (for a detailed description, interested readers are referred to the references provided for each one).

Algorithm 1 Template of a multi-objective metaheuristic

1: $S(0) \leftarrow$ GenerateInitialSolutions() // S can contain only a solution
2: Evaluation(S)
3: $A(0) \leftarrow$ Update($A(0)$, $S(0)$)
4: $t \leftarrow 0$
5: **while not** StoppingCriterion( ) **do**
6: $t \leftarrow t + 1$
7: $S(t) \leftarrow$ Variation($A(t-1)$, $S(t-1)$)
8: Evaluate($S(t)$)
9: $A(t) \leftarrow$ Update($A(t)$, $S(t)$)
10: **end while**
11: **Output:** A

3.1. NSGA-II

The Non-dominated Sorting Genetic Algorithm II, NSGA-II, was proposed by Deb *et al.* [13]. It is a genetic algorithm based on generating a new population from the original one by applying the typical genetic operators (selection, crossover, and mutation); then, the individuals in the new and old population are sorted according to their rank, and the best solutions are chosen to create a new population. In case of having to select some individuals with the same rank, a density estimation based on measuring the crowding distance to the surrounding individuals belonging to the same rank is used to get the most promising solutions. From Algorithm 1, S and A are considered to be one single set $P = S \cup A$ so that, at each iteration, the non-dominated solutions found are used to generate new solutions within the evolutionary loop.

3.2. SPEA2

The Strength Pareto Evolutionary Algorithm 2, SPEA2, was proposed by Zitler *et al.* in [14]. In this algorithm, each individual has a fitness value that is the sum of its strength raw fitness plus a density estimation. SPEA2 fits perfectly in the general template of Algorithm 1, having a population of solutions plus an external archive. That is, the algorithm applies the selection, crossover, and mutation operators with solutions from S to fill the archive A of individuals; then, the nondominated individuals of both the original population and the archive are copied into a new population. If the number of nondominated individuals is greater than the population size, a truncation operator based on calculating the distances to the k-th nearest neighbor is used. This way, the individuals having the minimum distance to any other individual are chosen.

3.3. PAES

The Pareto Archived Evolution Strategy, PAES, is a metaheuristic proposed by Knowles and Corne [15]. The algorithm is based on a simple (1+1) evolution strategy. To find diverse solutions in the Pareto optimal set, PAES uses an external archive of nondominated solutions, which is also used to decide about the new candidate solutions. That is, from one single solution PAES is able to approximate the entire Pareto front. In this case $|S| = 1$ in Algorithm 1, and the variation operator works with one single solution (a mutation operator) to generate new non-dominated ones. An adaptive grid is used as a density estimator in the archive.

3.4. Differential Evolution with Pareto Tournaments

The Differential Evolution (DE) is an evolutionary algorithm created by Ken Price and Rainer Storn [24]. The fundamental idea behind DE is a scheme for generating new possible solutions (trial individuals) taking advantage of the differences among the population (target individuals), according to its simple formulae of vector-crossover and mutation. We have defined a new multiobjective version that incorporates the Pareto Tournaments concept (DEPT) [16] to choose the best solution between two given ones based (in this case, the target

and the trial individuals). Ties are broken in the tournament (in case on non-dominance) by using the crowding distance of NSGA-II. As it can be seen, this algorithm considers both the solution set S and the archive A from Algorithm 1 to be the same set. Non-dominated solutions are generated by using the DE crossover with solutions in this set.

3.5. Multiobjective Firefly Algorithm

The Firefly Algorithm (FA) is one of the latest nature-inspired optimizers proposed. This algorithm is defined by Xin-She Yang [17] and it is inspired by the flash pattern and characteristics of fireflies. To solve the SPS problem we have developed the Multiobjective Firefly Algorithm (MO-FA) in which the attractiveness of a firefly, determined by its brightness, is associated with the objective functions. So, by generating a set of fireflies, the search loop of the algorithm evolves by moving dominated fireflies towards brighter (non-dominated) ones. As DEPT, MO-FA considers the solution set and the archive to be unified. The parameter values have been established as proposed in [17].

3.6. Multiobjective Artificial Bee Colony

The Artificial Bee Colony (ABC) is a new population-based algorithm proposed by Karaboga [18] and inspired by the collective behaviour of the honey bee swarms. A multi-objective version of ABC (MOABC) has been devised. It keeps the general outline of the ABC algorithm but incorporating the concept of dominance and crowding distance from NSGA-II. In fact, it uses the same idea: the bee colony is split into two sub colonies. As in NSGA-II, the first colony is used to generate the second one and finally those non-dominated solutions from the entire one are retrieved. That is, the matching with Algorithm 1 is the same as NSGA-II: S and A are considered as one single set which is iteratively improved with the ABC search operators.

3.7. MOCell

The Multi-Objective Cellular Genetic Algorithm, MOCell, is a cellular genetic algorithm (cGA) [19]. Like many multi-objective metaheuristics, it includes an external archive to store the nondominated solutions found so far. The matching with Algorithm 1 is therefore straightforward. The archive is bounded and uses the crowding distance of NSGA-II to keep diversity in the Pareto Front. We have used here an asynchronous version of MOCell, called aMOCell4 in [25], in which the cells are explored sequentially (asynchronously). The selection is based on taking an individual from the neighborhood of the current solution (called *cell* in cGAs) and another one randomly chosen from the archive. After applying the genetic crossover and mutation operators, the new offspring is compared with the current one, replacing it if better; if both solutions are nondominated, the worst individual in the neighborhood is replaced by the current one. In these two cases, the new individual is inserted into the archive.

3.8. GDE3

The Generalized Differential Evolution 3, GDE3 [20], is an improved version of the Generalized Differential Evolution (GDE) algorithm [26]. It starts with a population of random solutions, which becomes the current population. At each generation, an offspring population is created using the differential evolution operators; then, the current population for the next generation is updated using the solutions of both, the offspring and the current population. Before proceeding to the next generation, the size of the population is reduced using nondominated sorting and a pruning technique aimed at diversity preservation, in a similar way as NSGA-II, although the pruning used in GDE3 modifies the crowding distance of NSGA-II in order to solve some of its drawbacks when dealing with problems having more than two objectives. Therefore, the matching with Algorithm 1 is the same as NSGA-II: S and A are merged into one single set.

4. Experimentation

This section is aimed at presenting the experiments conducted to evaluate the scalability capabilities of the previously described algorithms on 36 instances of the SPS problem. Recall that we are looking at both the (financial) cost and the duration of the project scheduling given by the aforementioned multi-objective metaheuristics. These are the two conflicting objective functions that guide the search of these algorithms.

4.1. SPS Instances

For the empirical study we have used a total of 36 instances[1]. Each instance represents a different software project. The number of employees and tasks scales up from 8 to 256 and from 16 to 512, respectively. The total number of skills in the project, S, is 10 and the number of skills per employee ranges from 6 to 7. We denote the instances with iT-E, where T and E are the number of tasks and employees, respectively. For example, the instance i128-32 has 128 tasks and 32 employees. The maximum dedication for all the employees is 1 (full working day) in the 36 instances.

4.2. 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 [27, 28]. Two indicators have been used for this purpose in this work: the hypervolume (HV) [21] and the attainment surfaces [22].

The HV is considered as one of the more suitable indicators in the multi-objective community since it provides a measure that takes into account both the convergence and diversity of the obtained approximation set. Higher values

[1] `http://mstar.lcc.uma.es/problems/swscheduling.html`

of the hypervolume metric are desirable. Since this indicator is not free from an arbitrary scaling of the objectives, we have built up a reference Pareto front (RPF) for each problem composed of all the nondominated solutions found for each problem instance by all the algorithms. Then, the RPF is used to normalize each approximation prior to compute the HV value by mapping all the nondominated solutions to $[0, 1]$. This way the reference point to compute the HV values is (1,1), which results from the mapping of the extreme solutions of the RPF.

While the HV allows one to numerically compare different algorithms, from the point of view of a decision maker, knowing about the HV value might not be enough, because it gives no information about the shape of the front. The empirical attainment function (EAF) [22] has been defined to do so. EAF graphically displays the expected performance and its variability over multiple runs of a multi-objective algorithm. 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:

$$\alpha(z) = \frac{1}{r} \sum_{i=1}^{r} I(A^i \preccurlyeq \{z\}) \tag{16}$$

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 $A^i \preccurlyeq \{z\}$ means A^i dominates solution z. Thanks to the attainment function, it is possible to define the concept of k%-attainment surface [22]. 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.

Metaheuristics are stochastic algorithms; therefore the results have to be provided with statistical significance. The following statistical procedure has been used. First, 30 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, as the probability distributions of these values do not follow the normality and homocedasticity conditions [29], the Friedman's test has been used to rank the algorithms attending to their median HV value. Then, the Holm's multicompare test have been used to assess which algorithms are statistically worse than that that ranked the first in the Friedman's test ($1 \times N$ comparison) [30]. All the statistical tests are performed with a confidence level of 95%.

4.3. Parameterization

In order for a fair comparison among all the algorithms to be performed, they all are required to run for 100,000 function evaluations and to obtain 100 nondominated solutions at most. The detailed settings for each of the eight

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

Parameterization used in NSGA-II	
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	
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	
Population Size	1 individual
Mutation	polynomial, $p_m = 1.0/L$
Archive Size	100
Parameterization used in DEPT	
Population Size	32
Crossover Probability	90%
Mutation Factor	50%
Selection Scheme	RandToBest/1/Binomial
Parameterization used in MO-FA	
Population Size	32
Mutation Factor	50%
Alpha (α)	1
Beta (β_0)	1
Gamma (γ)	1
Parameterization used in MOABC	
Population Size	100
Mutation Probability	10%
Mutation Shift	30%
Scout Bees	5
Parameterization used in MOCell	
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	
Population Size	100 individuals
Recombination	Differential Evolution, $CR = 0.1$, $F = 0.5$

algorithms are included in Table 1. A solution to the problem is a vector of floating point numbers in which the component i stores the dedication of employee $\lfloor i/T \rfloor$ to task $i\ MOD\ T$ (where T is the number of tasks). With this encoding, the typical operators from the multi-objective metaheuristic community have been used (see Table 1). Finally, we want to clarify two relevant points. On the one hand, we have not paid attention to the particular parameterization of the algorithms as we have used the standard values given in the seminal works in which they are presented. Such a thorough analysis for each combination algorithm/SPS instance is out of the scope of this work. On the other hand, we want to remark again that the comparison is fair in terms of both

the numerical performance (i.e., the size of the sampling in the search space) and the maximum size of the approximated fronts (i.e., no algorithm is given more chance to cover regions of the Pareto front by using nondominated sets of unbounded size).

4.4. Repair Operator

Because of the size of the SPS instances addressed (see next section), it has been very hard for all the algorithms to compute feasible solutions. We found out that the employees' overwork (Equation (15)) is the most difficult constraint to meet. The reason is that the search operators are not endowed with problem-specific knowledge so it is usual to find assignments in which one or more employees exceed their maximum dedication. In order to deal with such issue, we have used a repair operator that, whenever an overwork in the assignment is detected, it is fixed by dividing the dedication of all the employees to all the tasks by the maximum overwork of the employees. That is, the effect of the operator is:

$$x'_{ij} = \frac{x_{ij}}{\max_{i,\tau}\{e_i^{work}(\tau)\} + \varepsilon} \tag{17}$$

where $\varepsilon = 0.00001$ is used in order to prevent from inaccuracies in the floating-point operations.

This operator increases the project duration of the tentative solution and keeps the cost unchanged. That is: $p'_{dur} = p_{dur} \cdot (\max_{i,\tau}\{e_i^{work}(\tau)\} + \varepsilon)$ and $p'_{cost} = p_{cost}$. In addition, the new solution satisfies the third constraint (the one related to the overwork). From the point of view of the algorithmic complexity, the overhead introduced is the same as the evaluation of the solution, since the coefficient used in the denominator is computed at the same time that the solution is evaluated. If a solution violates the other constraints of the problem, then it is penalized in the selection and replacement operators (it is the last one selected).

4.5. Comparison of the MO algorithms

This first part of the analysis is devoted to compare the multi-objective metaheuristics on the set of 36 SPS instances by using the HV indicator. Table 2 shows the median and interquartile range of the HV values of the algorithms for each instance on 30 independent runs. A special notation appears in the table: a gray coloured background has been used to better show the best (darker gray) and second-best (lighter gray) performing algorithm.

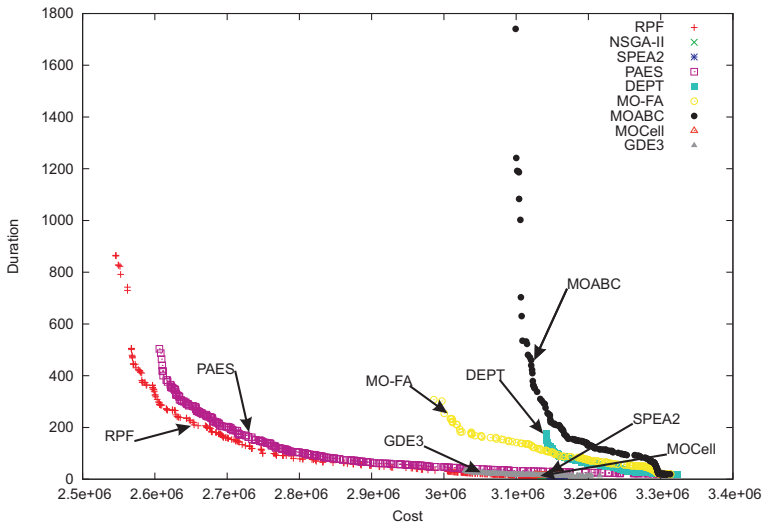
The HV values draw a clear scenario: PAES has been able to approximate the Pareto fronts with the best (highest) indicator values. It has ranked the first in 34 out of the 36 instances studied in this work. Indeed, attending to the output of the Friedman's test (Table 3), PAES has reached an average ranking of 1.0556 and, as a consequence, it has been established as the control method for the $1 \times N$ pos-hoc Holm's test. It can be observed in the last column of Table 3 that all the p-values included are lower than 0.05, so we can state that the differences are statistically significant in all the cases (statistical confidence

Table 2: Median and interquartile range of HV for all the algorithms and SPS instances.

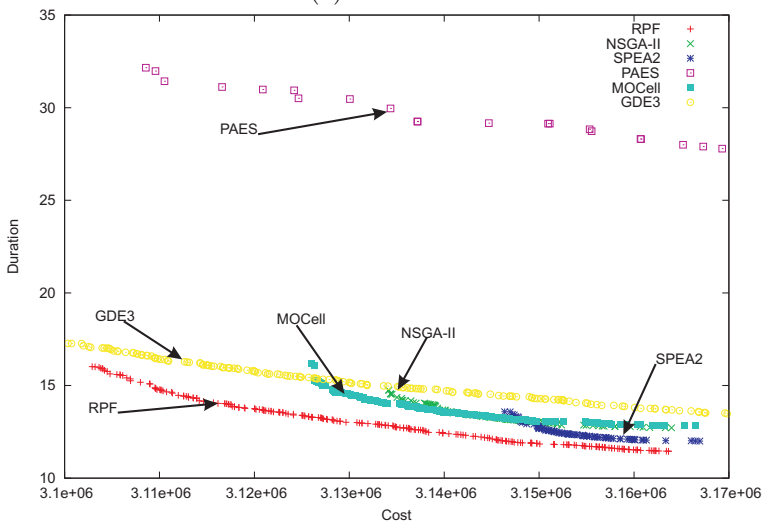
Instance	NSGA-II	SPEA2	PAES	DEPT	MO-FA	MOABC	MOCell	GDE3
i16-8	$0.656_{0.035}$	$0.540_{0.043}$	$0.730_{0.032}$	$0.310_{0.027}$	$0.539_{0.030}$	$0.364_{0.041}$	$0.716_{0.039}$	$0.765_{0.044}$
i16-16	$0.469_{0.036}$	$0.372_{0.040}$	$0.826_{0.015}$	$0.325_{0.062}$	$0.612_{0.034}$	$0.228_{0.042}$	$0.521_{0.053}$	$0.655_{0.039}$
i16-32	$0.147_{0.023}$	$0.086_{0.021}$	$0.811_{0.010}$	$0.217_{0.058}$	$0.386_{0.147}$	$0.137_{0.036}$	$0.177_{0.039}$	$0.421_{0.058}$
i16-64	$0.129_{0.023}$	$0.081_{0.022}$	$0.859_{0.015}$	$0.279_{0.038}$	$0.351_{0.118}$	$0.207_{0.072}$	$0.146_{0.032}$	$0.351_{0.037}$
i16-128	$0.049_{0.021}$	$0.029_{0.021}$	$0.724_{0.013}$	$0.122_{0.038}$	$0.121_{0.105}$	$0.010_{0.014}$	$0.047_{0.022}$	$0.264_{0.033}$
i16-256	$0.019_{0.012}$	$0.010_{0.010}$	$0.683_{0.009}$	$0.074_{0.038}$	$0.063_{0.041}$	$0.018_{0.013}$	$0.031_{0.017}$	$0.225_{0.029}$
i32-8	$0.538_{0.036}$	$0.431_{0.033}$	$0.723_{0.018}$	$0.111_{0.025}$	$0.211_{0.030}$	$0.113_{0.038}$	$0.580_{0.035}$	$0.645_{0.040}$
i32-16	$0.188_{0.038}$	$0.126_{0.014}$	$0.824_{0.019}$	$0.035_{0.022}$	$0.346_{0.078}$	$0.123_{0.031}$	$0.223_{0.029}$	$0.402_{0.067}$
i32-32	$0.125_{0.024}$	$0.079_{0.013}$	$0.743_{0.015}$	$0.105_{0.020}$	$0.276_{0.057}$	$0.120_{0.028}$	$0.143_{0.016}$	$0.333_{0.015}$
i32-64	$0.048_{0.013}$	$0.027_{0.010}$	$0.796_{0.030}$	$0.042_{0.019}$	$0.259_{0.131}$	$0.062_{0.025}$	$0.061_{0.020}$	$0.220_{0.030}$
i32-128	$0.040_{0.011}$	$0.024_{0.009}$	$0.727_{0.015}$	$0.075_{0.019}$	$0.063_{0.012}$	$0.002_{0.006}$	$0.035_{0.016}$	$0.198_{0.020}$
i32-256	$0.009_{0.009}$	$0.000_{0.000}$	$0.619_{0.022}$	$0.007_{0.014}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.030_{0.015}$	$0.194_{0.016}$
i64-8	$0.469_{0.048}$	$0.345_{0.042}$	$0.811_{0.015}$	$0.086_{0.018}$	$0.145_{0.028}$	$0.025_{0.018}$	$0.526_{0.038}$	$0.528_{0.038}$
i64-16	$0.219_{0.019}$	$0.154_{0.015}$	$0.961_{0.008}$	$0.024_{0.008}$	$0.341_{0.041}$	$0.173_{0.074}$	$0.264_{0.030}$	$0.322_{0.044}$
i64-32	$0.065_{0.014}$	$0.031_{0.014}$	$0.798_{0.012}$	$0.000_{0.000}$	$0.312_{0.044}$	$0.011_{0.015}$	$0.059_{0.015}$	$0.239_{0.027}$
i64-64	$0.074_{0.020}$	$0.046_{0.012}$	$0.872_{0.006}$	$0.000_{0.005}$	$0.029_{0.025}$	$0.000_{0.000}$	$0.081_{0.021}$	$0.258_{0.024}$
i64-128	$0.026_{0.009}$	$0.012_{0.007}$	$0.737_{0.014}$	$0.005_{0.006}$	$0.000_{0.002}$	$0.000_{0.000}$	$0.023_{0.011}$	$0.157_{0.025}$
i64-256	$0.000_{0.000}$	$0.000_{0.000}$	$0.619_{0.017}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.007_{0.014}$	$0.150_{0.022}$
i128-8	$0.320_{0.025}$	$0.242_{0.024}$	$0.987_{0.005}$	$0.000_{0.000}$	$0.073_{0.038}$	$0.000_{0.000}$	$0.372_{0.045}$	$0.405_{0.023}$
i128-16	$0.251_{0.028}$	$0.179_{0.026}$	$0.992_{0.010}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.310_{0.030}$	$0.306_{0.033}$
i128-32	$0.211_{0.016}$	$0.148_{0.025}$	$0.983_{0.009}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.267_{0.029}$	$0.270_{0.018}$
i128-64	$0.095_{0.012}$	$0.069_{0.010}$	$0.916_{0.025}$	$0.000_{0.000}$	$0.044_{0.039}$	$0.000_{0.000}$	$0.104_{0.012}$	$0.169_{0.016}$
i128-128	$0.028_{0.011}$	$0.013_{0.012}$	$0.781_{0.014}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.037_{0.014}$	$0.140_{0.015}$
i128-256	$0.008_{0.007}$	$0.000_{0.003}$	$0.877_{0.040}$	$0.015_{0.011}$	$0.723_{0.063}$	$0.254_{0.062}$	$0.027_{0.009}$	$0.148_{0.018}$
i256-8	$0.306_{0.023}$	$0.233_{0.014}$	$0.998_{0.003}$	$0.000_{0.000}$	$0.000_{0.014}$	$0.000_{0.002}$	$0.358_{0.020}$	$0.367_{0.033}$
i256-16	$0.154_{0.015}$	$0.121_{0.008}$	$0.989_{0.011}$	$0.000_{0.000}$	$0.275_{0.033}$	$0.160_{0.109}$	$0.174_{0.011}$	$0.232_{0.031}$
i256-32	$0.086_{0.012}$	$0.068_{0.008}$	$0.927_{0.020}$	$0.000_{0.000}$	$0.002_{0.014}$	$0.000_{0.000}$	$0.104_{0.014}$	$0.155_{0.011}$
i256-64	$0.027_{0.004}$	$0.014_{0.006}$	$0.744_{0.030}$	$0.000_{0.000}$	$0.000_{0.002}$	$0.000_{0.000}$	$0.027_{0.009}$	$0.116_{0.011}$
i256-128	$0.018_{0.006}$	$0.010_{0.004}$	$0.755_{0.015}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.025_{0.008}$	$0.110_{0.010}$
i256-256	$0.003_{0.005}$	$0.000_{0.002}$	$0.868_{0.048}$	$0.000_{0.000}$	$0.880_{0.151}$	$0.236_{0.048}$	$0.023_{0.009}$	$0.106_{0.014}$
i512-8	$0.238_{0.020}$	$0.182_{0.025}$	$0.996_{0.004}$	$0.000_{0.000}$	$0.022_{0.039}$	$0.010_{0.032}$	$0.273_{0.023}$	$0.217_{0.017}$
i512-16	$0.104_{0.013}$	$0.085_{0.006}$	$0.963_{0.016}$	$0.000_{0.000}$	$0.018_{0.034}$	$0.000_{0.000}$	$0.119_{0.011}$	$0.143_{0.013}$
i512-32	$0.065_{0.010}$	$0.048_{0.009}$	$0.963_{0.011}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.081_{0.013}$	$0.117_{0.019}$
i512-64	$0.030_{0.013}$	$0.019_{0.006}$	$0.822_{0.016}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.045_{0.013}$	$0.102_{0.011}$
i512-128	$0.198_{0.005}$	$0.191_{0.003}$	$0.822_{0.013}$	$0.102_{0.004}$	$0.104_{0.006}$	$0.095_{0.003}$	$0.216_{0.013}$	$0.249_{0.009}$
i512-256	$0.000_{0.000}$	$0.000_{0.000}$	$0.561_{0.046}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.000_{0.000}$	$0.020_{0.008}$	$0.072_{0.009}$

Table 3: Average Friedman's rankings with Holm's correction ($\alpha = 0.05$).

Algorithm	Ranking	p_{Holm}
NSGA-II	4.6528	1.3942E-9
SPEA2	5.9861	6.7127E-17
PAES	1.0556	—
DEPT	6.6111	3.8548E-21
MO-FA	5.0000	3.3501E-11
MOABC	6.6944	1.0937E-21
MOCell	3.7083	8.6655E-6
GDE3	2.2917	3.2273E-2



(a) Full view



(b) Zoom at cost 3100000 to 3170000 currency units and duration 10 to 35 weeks.

Figure 1: 50%-Attainment surfaces of the i32-64 instance with full (a) and zoomed (b) views.

at 95%). The second best performing algorithm is GDE3, with an average rank of 2.2917 (it scores the best in one instance and 26 the second best out of 36 instances). The opposite side of the ranking is occupied by MOABC, DEPT, and SPEA2. We want to clarify that the zero HV values obtained by many of the algorithms are due to the normalization process that simply discards the nondominated solutions that are out of the limits of the RPF built up for each instance.

In order to better explain these results, Figure 1 displays the 50%-attainment surface of the i32-64 instance, which has the representative features that mostly appear in all studied instances. We have included both (a) the full plot and (b) a zoom of the region with 3,100,000 to 3,170,000 of the project cost and 10 to 35 weeks of its duration[2]. Figure 1(a) clearly explains the high HV value obtained by PAES. The approximated sets of this algorithm are the ones that cover a larger portion of the objective space, what means that it is the one that can provide the decision maker with a more diverse set of options. This fact becomes more evident as long as the instances are larger. We can therefore conclude that PAES has the desirable property of scalability, which is precisely the aim of this study. Such a behavior is justified by the constrained search space of the SPS problem, the repair operator used, and the disruption provoked by recombination operators. Indeed, once PAES reaches the feasible region of any given instance, its search engine is specially well suited here as it is based on a mutation operator. This mutation changes, on average, one single assignment of an employee to a task ($p_m = 1/L$, where L is the length of the solution) and, as a consequence, it is rather easy to keep the exploration within the feasible region (no need for the repair operator to be applied). On the other hand, it is rather easy for the algorithms that recombine solutions to incur in employees' overwork because of the big changes induced in the resulting project schedules. The repair operator then fixes such infeasibility by decreasing the employees' dedication, which usually leads to a large increment in the project duration but keeping its cost almost the same. However, we also want to show the benefits of solution recombination in Figure 1(b): in this zoomed region, it can be seen that the solutions from NSGA-II, SPEA2, MOCell, and GDE3 clearly dominate those from PAES. That is, these algorithms have sampled this area of the search space better than PAES, so if the decision maker is particularly interested in this region of the Pareto front, PAES is not recommended.

As to the scalability capabilities of the algorithms, let's start by analizing their behavior with an increasing number of employees. The general trend in all the algorithm but PAES is that, the higher the number of employees, the lower (worse) the HV value. In order to support this claim, Figure 2 shows, for each algorithm, the average HV over all the SPS instances addressed with the same number of employees (i.e., i16-8, i32-8, i64-8, ...). This reported tendency is specially relevant (a higher negative slope) in NSGA-II, MOCell, and GDE3. Again, the recombination of solutions on large instances justifies this fact. Indeed, as the instance size gets larger, the approximated fronts get narrower with respect to the RPF (which is mainly determined by the solutions provided by PAES) and the HV values get smaller as well. In the case of PAES, it can be observed that its averaged HV values start decreasing in instances with more than 64 employees.

If we now turn to evaluate the scalability in terms of the increasing number

[2]We have removed from the legend those algorithms that do not have any nondominated solution in this portion of the objective space.

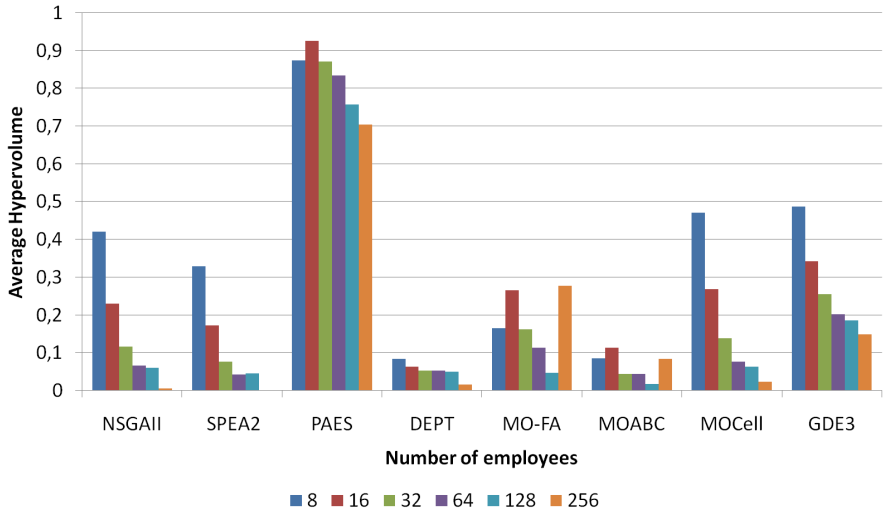


Figure 2: Average HV value over all the SPS instances with the same number of tasks for all the algorithms with an increasing number of employees.

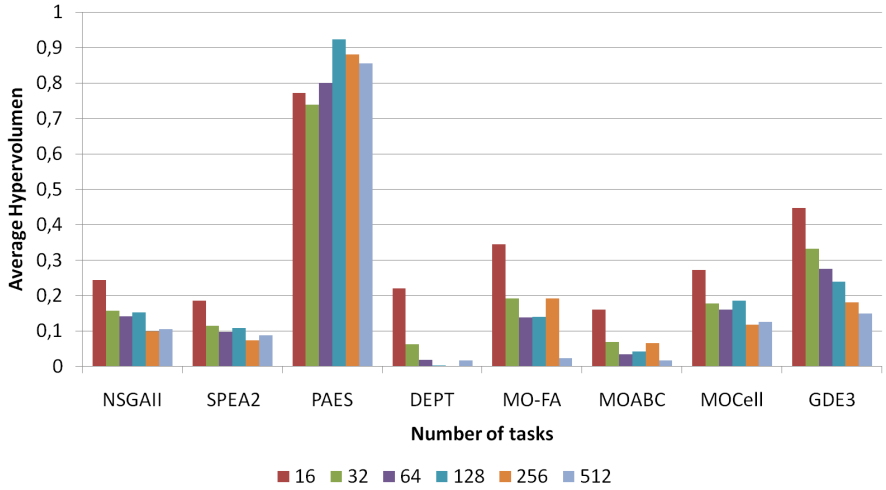


Figure 3: Average HV value over all the SPS instances with the same number of employees for all the algorithms with an increasing number of tasks.

of tasks, similar conclusions can be drawn. On the one hand, more tasks mean increasing difficulty, and this is what Table 2 reflects. In general, the HV values decrease with the number of tasks. This fact is shown in Figure 3, in which we have aggregated, for each algorithm, the average HV over all the SPS instances addressed with the same number of tasks (i.e., i16-8, i16-16, i16-32, ...). The

key issue again is the algorithm with/without solution recombination. All but PAES do not scale properly. However, the aggregated HV values of PAES even increase with the number of tasks. In this work, in which scalability is the feature of the algorithms under investigation, these results are encouraging because this algorithm has shown outstanding results (always bearing in mind the HV indicator). The attainment surfaces previously displayed have pointed out that these results are mainly due to the wide diversity of project plans reached (with low costs and long durations). However, PAES is also outperformed by NSGA-II, SPEA2, MOCell and GDE3 in the region of high-cost short-duration project schedulings.

Finally, regarding wall clock time, the execution of the algorithms requires between a few seconds in the case of the smallest instances to around 5 hours in the case of the largest instances.

4.6. 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, $E + T$ values have to be analyzed. For each instance, all the solutions of the approximated Pareto front obtained in the different independent runs of the algorithms 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 [30] between all the e_i^{par}, t_j^{ahr}, p_{dur}, and p_{cost} are then calculated. This means a large amount of data to process and show. In order to reduce this amount of data without losing interesting information we focus on some relevant correlations to show how are the solutions of the different algorithms and how the solutions in the approximated Pareto front change as the size of the instance increases.

But first, let us illustrate the meaning of the correlations in this context with a first example: the correlations found in the i16-8 instance for PAES. The correlation coefficients are shown in Figure 4. 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 nondominated 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 all the employees except e_6 the correlation is positive. This means that when the cost of the solutions proposed by the algorithm increases, the participation of these employees increases, that is, these employees spend more and more time in the tasks of the project as we move in the objective space

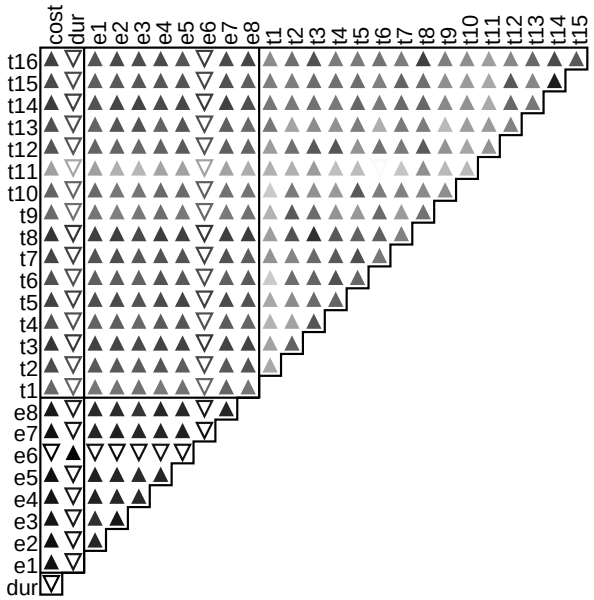


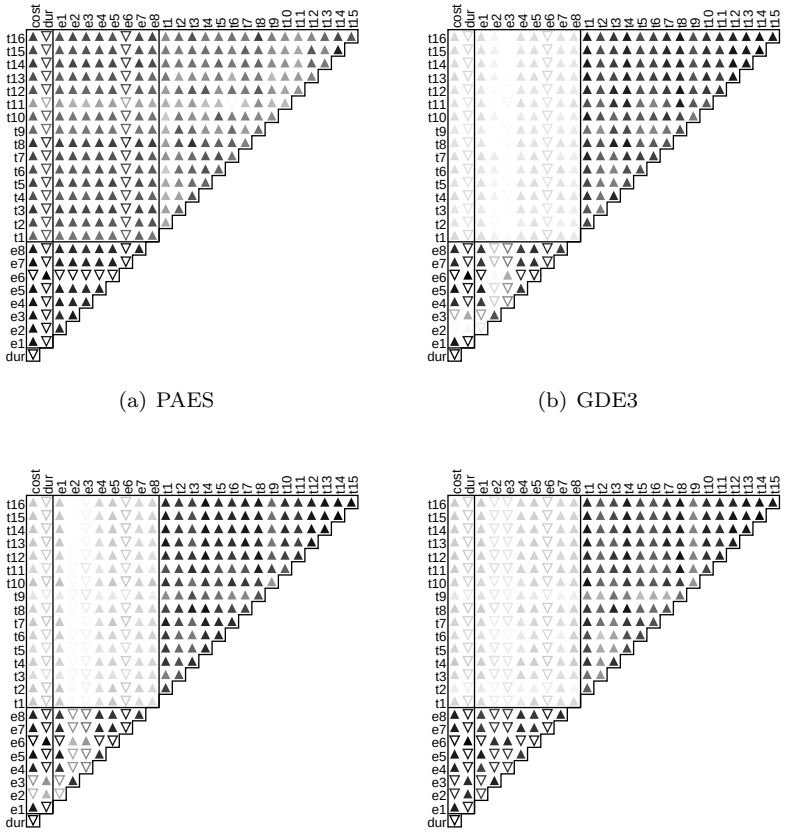
Figure 4: Spearman rank correlation coefficients between p_{cost}, p_{dur}, e_i^{par} and t_j^{ahr} in the i16-8 instance for the solutions obtained with PAES.

to solutions with higher cost and shorter duration (it is also possible to observe the negative correlation of these employees with project duration). However, the participation of e_6 is reduced. This means that the fraction of workload that is performed by e_6 is reduced. The reason is that e_6 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_6 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_6 and the rest of the employees, as expected from the previous discussion.

Let us now focus on 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 rest of the correlations among the tasks and employees can be explained based on the previous observation. In effect, if we move in the Pareto front to the region of minimum project duration (and maximum cost) the value t_j^{ahr} for all the tasks must increase in order to finish the project sooner. This is the reason why all the correlations between the tasks is positive. For the same reason, the correlation between the tasks and the workload of the employees is positive except for e_6. In summary, the information provided by the correlations between t_j^{ahr} and the rest of parameters is not relevant in this

case. This also happens in the rest of the instances.

Now we are ready to compare the solutions in the approximated Pareto front obtained by the different algorithms. For the sake of clarity in the figures we will focus on the i16-8 instance because it is the smallest one, but similar conclusions can be obtained for the other instances. The correlations are shown in Figures 5 and 6.

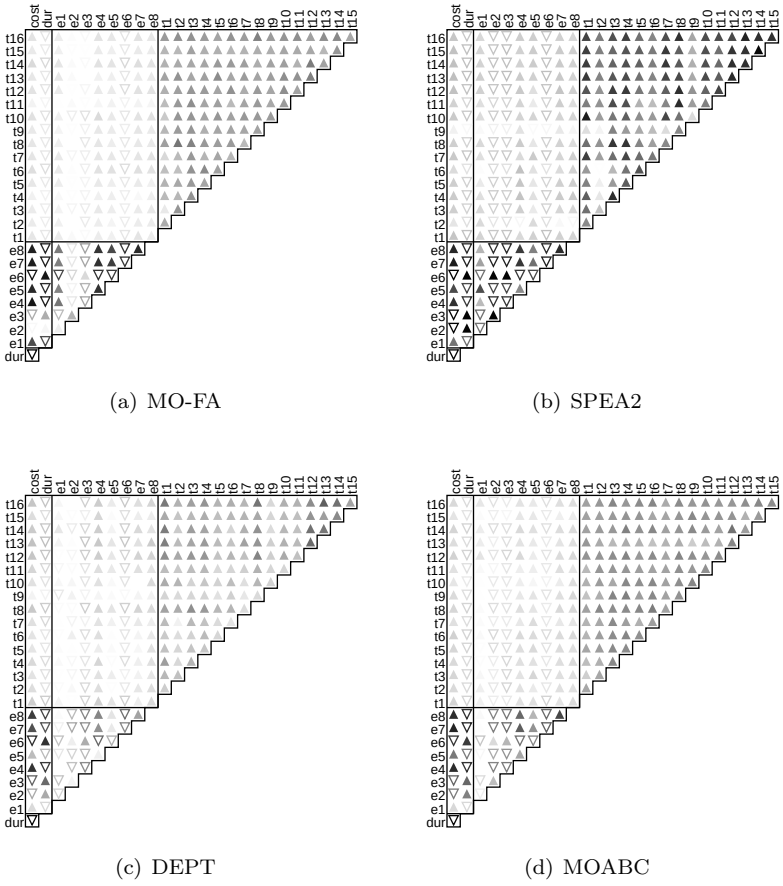


(a) PAES (b) GDE3

(c) MOCell (d) NSGA-II

Figure 5: Spearman rank correlation coefficients between p_{cost}, p_{dur}, e_i^{par} and t_j^{ahr} in the i16-8 instance for the solutions obtained with PAES, GDE3, MOCell, and NSGA-II.

The first conclusion we obtain after a comparison of the correlations in all the algorithms is that the solutions in the approximated Pareto front of the worst algorithms have low correlations between the considered parameters. For example, the correlation between the project cost or the project duration and the amount of human resources in the tasks is low, meaning that it is not always the case that more people is working on the tasks when the project



(a) MO-FA (b) SPEA2

(c) DEPT (d) MOABC

Figure 6: Spearman rank correlation coefficients between p_{cost}, p_{dur}, e_i^{par} and t_j^{ahr} in the i16-8 instance for the solutions obtained with MO-FA, SPEA2, DEPT, and MOABC.

duration is reduced (and the project cost increased). We can also find more inverse correlations between the parallel tasks performed by the employees. A simple explanation for this behaviour is that the solutions obtained by these algorithms are far from the optimal Pareto set and, thus, they are solutions that can be easily improved in many different ways since they are dominated by other solutions. As a consequence, different paths exist in the solution set that provides the same Pareto front and some of these paths have the observed correlations. That is, we can decrease the duration of the project and increase its cost without increasing the workload of the employees and the amount of human resources per task. As the algorithms approach the optimal Pareto set the number of different paths is reduced, limiting the correlations to the ones observed in PAES.

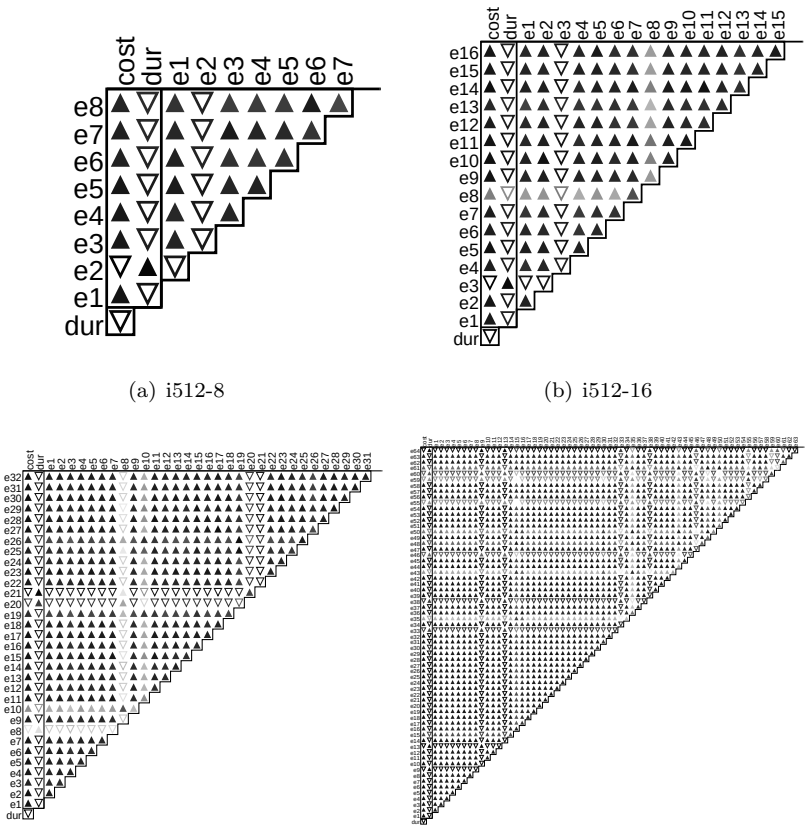
In short, we can conclude that if the correlations observed are high and positive between the amount of human resources and the correlations between the parallel tasks performed by the employees is mainly positive, then the corresponding approximated Pareto optimal set is near the actual Pareto optimal set. If we find low or inverse correlations among the amount of human resources of the tasks and/or a large number of inverse correlations between the number of parallel tasks performed by the employees then we have reasons to think that the approximated Pareto optimal set is far from the actual one.

Now we are interested in analyzing the features of the solutions as the size of the instances increase. For this analysis we focus on only one algorithm, PAES, which was the best algorithm according to the Hypervolume. We also remove the correlations related to the amount of human resources per task because these correlations give no relevant information in these instances, as we previously illustrated. Thus, we only show the correlations of the number of parallel tasks per employee and the duration and the cost of the project for the approximated Pareto set obtained by PAES in the instances with 512 tasks and 8 to 64 employees. The correlations are shown in Figure 7.

We can observe that as the number of employees increases more inverse correlations can be found. We can also see how low correlations appear more frequently in the instances with more employees. The employees having low correlations or inverse correlations with the rest of employees are the ones earning less money. The explanation is similar as the illustrative case of Figure 4. The cheapest employees are used in the low-cost long-duration projects. With a small number of employees the cheapest one is used. As the number of employees increases, more than one employee is used in these low-cost projects. This is the reason why more than one employee have inverse correlations. When this happens we observe that the ones with the highest inverse correlation are those with the lower salary, but as the salary increase, the correlation decreases in absolute value and at some point it changes to a positive correlation. That is, using the correlations between the employees it is possible to classify the employees according to their salary. The employees with low salary are those with a strong inverse correlation with most of the employees. The employees having a low correlation with their partners have a higher salary but still low enough. Finally, most of the employees fall in the third category with higher salary and positive correlations with their partners.

5. A Tool for a Project Manager

We try to describe in this section how can a project manager profit from a tool containing the techniques proposed here. We have already mentioned that the analysis of the solutions based on correlations presented in Section 4.6 can provide some information to the project manager. However, in her/his daily work a project manager needs a more clear, easy to interpret and direct information of the proposed solutions. A plot with the approximated Pareto front of the solutions obtained after the search is not enough for a project manager to decide what is the best solution. The reason is that there are



(a) i512-8

(b) i512-16

(c) i512-32

(d) i512-64

Figure 7: Spearman rank correlation coefficients between p_{cost}, p_{dur} and e_i^{par} in the instances with 512 tasks and 8 to 64 employees for the solutions obtained with PAES.

some constraints and preferences that are not included in the formulation of the problem, they are only in the mind of the manager. For example, perhaps one of the employees gets stressed when s/he is working simultaneously in two tasks requiring programming skills, or one of the tasks has a high probability of being delayed by an external circumstance. These additional preferences and constraints determine the final decision of the project manager and they cannot be easily included in the mathematical model of the problem, since many of them are subjective. Furthermore, even if we could include some of them, others probably couldn't, and perhaps it makes no sense to use a complex mathematical model that, in the end, is not able to satisfy all the requirements.

A reasonable alternative to a more complex model of the problem is to provide the maximum amount of information to the project manager about any

particular solution obtained by the algorithms. In our case, in addition to the project duration and cost we can provide the workload of the employees and the Gantt diagram showing the starting and finishing dates of the tasks. This can be done in a software tool for decision support in which the manager just click in a solution of the approximated Pareto front and all this information is shown. Furthermore, the tool could be a front-end for the algorithms and could allow the user to change any data of the instances and run the algorithm again to obtain a new Pareto front with the new changes. This kind of tool opens the door to the study of "what if" scenarios. The manager could study in this way what happens if the workload of an employee is reduced, what if the cost of a tasks is increased, what if one employee is sick for some weeks, etc. In Figure 8 we show a possible GUI for such kind of tool.

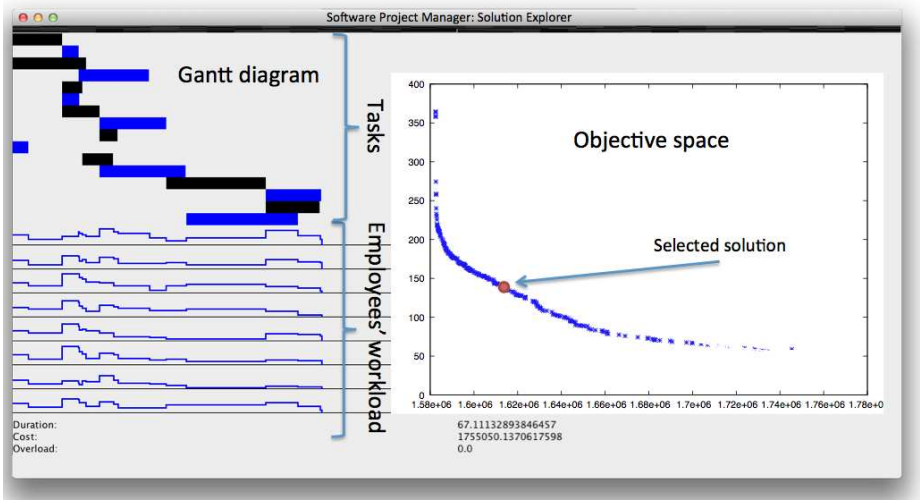
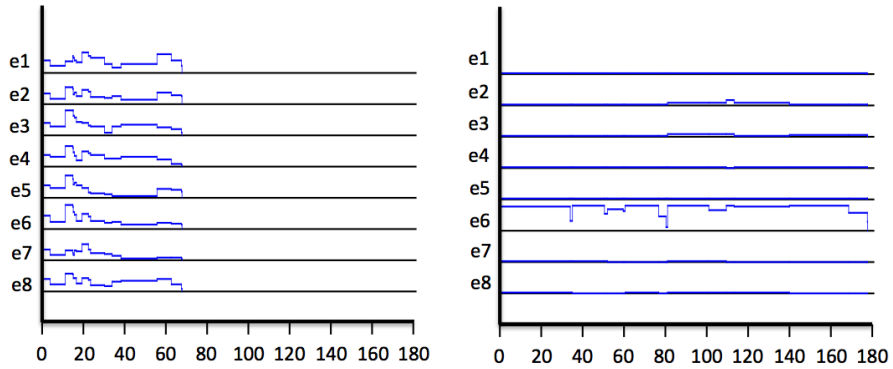


Figure 8: Hypothetical software tool for decision support of a software project manager.

We have partially implemented this kind of tool and, for illustration purposes, we show the workload of the employees along the project for instance i16-8 in Figure 9. On the left we can see the workload of the solution having the minimum duration in the Pareto front and on the right we show the solution with minimum cost. Both solutions were obtained by PAES. We can observe how the solution found by the algorithm to minimize the cost is the one that assigns all the work to the cheapest employee (e_6 in this case). In general, this is not a good solution in a real project. Thanks to the information provided by a tool like the one mentioned here, a project manager can decide to discard this solution.



(a) Minimum duration (b) Minimum cost

Figure 9: Workload of the employees along the project for the instance i16-8. The solutions are the extreme solutions of an approximated Pareto front obtained by PAES.

6. Conclusions and Future Work

In this paper we have analyzed the scalability of eight multi-objective metaheuristic algorithms when they are applied to the Software Project Scheduling problem. The efficient resolution of the problem is important in the context of software companies, which have to deal with large software projects. We have performed an experimental evaluation using a benchmark of 36 automatically generated instances with increasing size and both, the hypervolume indicator and the attainment surfaces, have been used in order to evaluate the quality of the approximated fronts. The results have reported encouraging conclusions taking into account the property of the algorithms being investigated (i.e., scalability). PAES has shown to be not only the algorithm that scales the best but also the one with the best solution quality (in terms of HV). The attainment surfaces have allowed us to refine this outstanding behavior by graphically displaying the reason of these high HV values: PAES is able to reach projects with low cost and long durations, but it is outperformed by other algorithms in a narrow region of the objective space related to high-cost short-duration schedules. We have also gone one step further by analyzing the properties of the obtained solutions by computing the correlations between objectives, the assigned dedication of the employees, and the tasks.

A future line of research would be the design of a new version of the problem including some real-world issues that are not present in the current formulation, like communication overhead in a group of employees or solution robustness. Especially interesting is the design of new hybrid algorithms that combine together the best features of the algorithms analyzed in this work as well as a theoretical analysis of the problem landscape so as to devise advanced operators and algorithms.

Acknowledgment

This work has been partially funded by the Spanish Ministry of Science and Innovation and ERDF (European Regional Development Fund) under contract TIN2008-06491-C04 (M* project), by the Spanish Ministry of Economy and Competitiveness and the ERDF under contracts TIN2012-30685 (BIO project) and TIN2011-28194 (roadME project), and by the Andalusian Government under contract P07-TIC-03044. Thanks also to the Fundación Valhondo, for the economic support offered to David L. González-Álvarez to make this research.

References

[1] B. W. Boehm, C. Abts, A. W. Brown, S. Chulani, B. K. Clark, E. Harrowitz, R. J. Madachy, D. J. Reifer, B. Steece, Software Cost Estimation with Cocomo II, 1st Edition, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.

[2] P. Brucker, A. Drexl, R. Mohring, K. Neumann, E. Pesch, Resource-constrained project scheduling: Notation, classification, models, and methods, European Journal of Operational Research 112 (1) (1999) 3–41.

[3] C. K. Chang, M. Christensen, A net practice for software project management, IEEE Software 16 (6) (1999) 80–88.

[4] A. Barreto, M. Barros, C. M. L. Werner, Staffing a software project: A constraint satisfaction and optimization-based approach, Computer and Opererations Research 35 (10) (2008) 3073–3089.

[5] E. Alba, F. Chicano, Software project management with GAs, Information Sciences 177 (11) (2007) 2380–2401.

[6] G. Antoniol, M. D. Penta, M. Harman, Search-based techniques applied to optimization of project planning for a massive maintenance project, in: 21st IEEE International Conference on Software Maintenance (ICSM 2005), 2005, pp. 240–249.

[7] M. di Penta, M. Harman, G. Antoniol, The use of search-based optimization techniques to schedule and staff software projects: an approach and an empirical study, Software – Practice and Experience 41 (5) (2011) 495 – 519.

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

[9] M. Harman, S. A. Mansouri, Y. Zhang, Search-based software engineering: Trends, techniques and applications, ACM Computing Surveys 45 (1) (2012) 11.

[10] J. F. Chicano, F. Luna, A. J. Nebro, E. Alba, Using multi-objective metaheuristics to solve the software project scheduling problem, in: Proceedings of GECCO, 2011, pp. 1915–1922.

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

[12] C. A. Coello Coello, G. B. Lamont, D. A. Van Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd Edition, Springer, New York, 2007.

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

[14] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the strength Pareto evolutionary algorithms, in: EUROGEN 2001, 2002, pp. 95–100.

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

[16] A. Rubio-Largo, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez, A differential evolution with pareto tournaments for solving the routing and wavelength assignment problem in wdm networks, in: Evolutionary Computation (CEC), 2010 IEEE Congress on, 2010, pp. 129–136.

[17] X.-S. Yang, Firefly algorithms for multimodal optimization, in: 5th International Symposium of Stochastic Algorithms: Foundations and Applications (SAGA'09), Vol. 5792, 2009, pp. 169 – 178.

[18] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Tech. Rep. tr06, Erciyes University, Turkey (2005).

[19] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, E. Alba, Mocell: A cellular genetic algorithm for multiobjective optimization, International Journal of Intelligent Systems 24 (7) (2009) 726 – 746.

[20] S. Kukkonen, J. Lampinen, GDE3: The third evolution step of generalized differential evolution, in: IEEE CEC'05, 2005, pp. 443 – 450.

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

[22] J. Knowles, A summary-attainment-surface plotting method for visualizing the performance of stochastic multiobjective optimizers, in: 5th Int. Conf. on Intelligent Systems Design and Applications (ISDA'05), 2005, pp. 552 – 557.

[23] F. Luna, D. L. González-Álvarez, F. Chicano, M. A. Vega-Rodríguez, On the scalability of multi-objective metaheuristics for the software scheduling problem, in: Proceedings of the 11th International Conference on Intelligent Systems Design and Applications (ISDA'11), IEEE Computer Society, 2011, pp. 1110 – 1115.

[24] K. Price, R. Storn, Differential evolution - a simple evolution strategy for fast optimization, Dr. Dobb's Journal 22 (4) (1997) 18–24 and 78.

[25] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, E. Alba, Design issues in a multiobjective cellular genetic algorithm, in: EMO 2007, Vol. LNCS 4403, 2007, pp. 126–140.

[26] J. Lampinen, DE's selection rule for multiobjective optimization, Tech. rep., Lappeenranta University of Technology, Department of Information Technology (2001).
URL `http://www.it.lut.fi/kurssit/03-04/010778000/MODE.pdf`

[27] A. Jaszkiewicz, A comparative study of multiple-objective metaheuristics on the bi-objective set covering problem and the pareto memetic algorithm, Annals of Operations Research 131 (1-4) (2004) 135 – 158.

[28] A. Jaszkiewicz, On the computational efficiency of multiple objective metaheuristics. the knapsack problem case study, European Journal of Operational Research 158 (2) (2004) 148 – 433.

[29] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: a case study on the CEC2005 special session on real parameter optimization, Journal of Heuristics 15 (6) (2009) 617–644.

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