A survey on the Software Project Scheduling Problem

Article *in* International Journal of Production Economics · August 2018

DOI: 10.1016/j.ijpe.2018.04.020

3 authors, including:

Abel García Nájera
Metropolitan Autonomous University
39 PUBLICATIONS **450** CITATIONS

SEE PROFILE

Humberto Cervantes
Metropolitan Autonomous University
49 PUBLICATIONS **832** CITATIONS

SEE PROFILE

A Survey on the Software Project Scheduling Problem

***Abstract*— Creating a plan for a software project is a recurring activity in software development organizations that plays a critical role in the project success. When creating a plan for a project, these organizations must deal with the problem of allocating resources to tasks in the project. Because of its importance, there has been considerable research focused on finding ways to solve this problem, which is known as the Software Project Scheduling Problem (SPSP). Solving this problem usually focuses on creating a schedule for a project with minimal duration and cost. As part of our work, we have found only one survey about the SPSP, however it focuses primarily on the methods used to solve it, while the rest of the surveys focus primarily on other scheduling problems such as the Resource-Constrained Project Scheduling Problem. In this paper, we present a survey of the current research focused on solving the SPSP. For this survey, we have analyzed and classified a number of research studies considering a set of criteria that include: the model used to represent the problem, the optimization goals, the optimization techniques used to solve the problem, the methodology used to evaluate the different approaches, and the main results. From our analysis, we produce a set of general observations and provide suggestions that we believe can be useful for future research in this field.**

Keywords: Project scheduling, Software Project Scheduling Problem, Survey, Planning model, Optimization algorithms.

1. INTRODUCTION

Project management is a key activity in software development organizations. Failures in this activity have an extremely high cost [52] and, even though project planning problems are just one possible cause, deciding the allocation of effort to different activities is one of the most important tasks during software project planning [53]. Some of the activities that are performed as part of the project planning phase include defining a calendar with tasks assigned to resources and estimating the termination date and cost associated to the project.

One of the goals that is pursued when creating a plan is the minimization of the project's cost and duration, since most of the time the budget is limited and there is a deadline to conclude the project. Duration and cost, however, are two goals that are typically in conflict, since when one is reduced, the other usually increases.

This problem is known in the literature as the Software Project Scheduling Problem (SPSP) [2]. The goal in solving the SPSP is typically to find the allocation of employees to tasks so that the project's cost and duration are

minimized. Since these objectives are in conflict, that is, the minimization of one of them leads to the deterioration of the other, the result of the optimization process is a set of solutions which represents the trade-offs between both objectives.

SPSP is considered to be an NP-hard problem [13]. There are exact methods that can solve small instances to optimality, however these methods are not practical for larger instances, and their resolution requires other kinds of techniques, such as metaheuristics [50].

In recent years, a considerable amount of research has been conducted around this topic. The studies that have been performed have focused on different aspects, for instance, on proposing models to solve the problem of allocating employees to tasks so that the project's cost and duration are minimized, on introducing or applying more efficient optimization algorithms, and on studying the performance of different metaheuristics to determine which may be more appropriate to solve the problem.

Despite different surveys have been published, we have found only one [30] about the SPSP, yet it focuses primarily on the methods used to solve the problem. The majority of the surveys we found are mostly focused upon a related problem called *Resource Constrained Project Scheduling* (RCPS) [51]. RCPS, as we discuss in Section 2, is sufficiently different from SPSP to grant the need for a survey regarding the SPSP. To address this situation, the goal of our work is to provide an overview of the main research studies that have contributed in solving the SPSP. We have classified these studies along several criteria which include: the model used to address the problem, the optimization objectives, the techniques used to solve the problem, the evaluation method, and the main results.

Our paper provides two main contributions to the field of research regarding SPSP:

- To our knowledge, it is the first extensive survey about the SPSP, and
- It provides a set of general observations and suggestions that we believe can be useful for future research in this field.

The structure of this survey is as follows. In Section 2 we present a summary of the previous work. In Section 3 we describe the methodology we used to perform our literature review. In Section 4 we present a classification of the surveyed studies using the criteria discussed previously. In Section 5 we present a discussion across the dimensions that we considered in our classification of the research papers. Finally, in Section 6 we present our conclusions.

2. PREVIOUS WORK

Research into project scheduling problems has increased in recent years and, as a consequence, several surveys around this topic have been published. We now describe part of the work presented in these surveys.

Firstly, the surveys [24], [25], and [26] focus on describing advances in the solution of the Resource-Constrained Project Scheduling (RCPS) problem which is related to SPSP. Özdamar and Ulusoy present a survey [24] which concentrates on classifying and summarizing different studies that contribute to the resolution of the RCPS problem. They review aspects related to the models used to solve RCPS, including the number of objectives (one or more), the number of projects (single or multiple), the solution strategy (i.e. the methods used to solve the problem) and the constraint specification. According to this survey, 75% of the publications that they reviewed model the problem by considering a single project, while the rest considered multiple projects. In 43% of the publications, exact (optimization) methods were used to solve the problem, while the rest used approximate (heuristic) methods. Regarding optimization objectives, 90% of the publications considered an approach with a single objective function, while only 10% considered an approach with multiple objective functions. Optimization objectives that are related to time included the duration (or makespan), mean lateness, and initial delay, while objectives associated with cost included the net present value and the total costs, including the costs of resource consumption. Regarding constraints, those that were considered include task precedence, renewable resources, non-renewable resources, and mixed (with the three types of constraints). In the survey's conclusions, the authors suggest the need of developing flexible heuristics to solve RCPS.

The work of Kolisch and Padman [25] is focused on presenting the algorithmic and modeling contributions to RCPS. They describe the problem's main model elements, the optimization objectives that are generally considered, the methods used to generate benchmark instance data, and the algorithms used to find solutions to the problem. The main model elements include activities and resources of different types, while the more common optimization objectives were the duration and net present value. To obtain data for the benchmark instances, different generators were used and the most common was ProGen [46]. Regarding the solution methods, the ones used were mainly exact methods like dynamic programming and zero-one programming, but also heuristic methods such as priority

rules, truncated branch-and-bound, integer programming based heuristics, and metaheuristics. The survey concluded that the reviewed models could have taken into account additional aspects such as the costs, resource levelling, and quality, among others, and that it would be desirable to apply more efficient metaheuristic techniques to solve the problem.

The survey in [26] presents an overview of the aspects that have been added or modified in different RCPS models, along with a discussion of other aspects including constraints, alternative optimization objectives that were considered, multi-project approaches, and benchmark instance generators. Some changes in the RCPS models include delays between the moment a task is finished and the moment its successor starts, delivery deadlines, and constraints related to activity interruption, among others. The optimization objectives were associated with aspects such as time, robustness, rescheduling, resources, net present value, and the costs. The evaluation of the optimization objectives was performed in two ways. In the first one, the objectives were evaluated in a single objective function by means of a weighted sum in which each objective has an associated weight, while in the second one, each objective was evaluated using a particular function and, as a consequence, multiple objective functions were used.

Other surveys such as [27], [28], and [29] present an overview of the use of optimization techniques in RCPS and other related problems. The survey in [27] focuses on evaluating the performance of different heuristic methods used to solve RCPS. After performing several experiments, the authors concluded that Simulated Annealing and Genetic Algorithms [42] provided the best performance. The survey in [28] concentrates on presenting a summary of the optimization algorithms that have been used to solve the general Project Scheduling Problem (PSP) along with RCPS. These algorithms included metaheuristic algorithms such as Genetic Algorithms and Particle Swarm Optimization [33]. The target of the survey in [29] was to compare different techniques used to solve RCPS and other related problems, such as Time-Cost Trade-Off, Resource Leveling, and Resource Allocation. The authors of this survey concluded that the metaheuristic methods deliver the best performance and are more promising than exact methods.

As described in [1], there are differences between SPSP and RCPS, including the fact that in SPSP there is a cost associated with the employees and that, besides project duration, project cost must be minimized. Moreover, SPSP

only considers one type of resource with several possible skills (i.e. the employees), as opposed to RCPS which considers several kinds of resources. Furthermore, SPSP resource skills are different from RCPS resource types, since SPSP resource skills are not quantifiable while RCPS requires different quantities of each resource. Alba and Chicano [1] argue that SPSP is more realistic than RCPS, given that it includes the concept of an employee with a salary and skills, and is also capable of performing several tasks during a regular working day.

Regarding specifically to software projects, Ferrucci et al. [63] provide an overview of search-based software project management, in which search-based techniques are applied to problems in software project management, such as staffing, scheduling, risk, and effort estimation. In particular for the SPSP, one of the most recent surveys is presented by Patil et al. [30]. Their work is centered on describing the main methods that have been used to solve the SPSP. First, they describe the main elements of the problem and then they perform a comparison of this problem with RCSP. Finally, they provide an overview of the methods used to solve the SPSP, including Genetic Algorithms and Ant Colony Optimization [33]. They conclude that their survey will allow better solution techniques to the SPSP to be studied and developed.

A related problem to SPSP is the Release Planning Problem (RPP). In the context of iterative development, as in agile software development, software products are developed and released in an incremental fashion, where each release includes a set of new features which is specified by a set of requirements. In this sense, the problem of allocating features to particular releases according to criteria such as business value, dependencies between features, and characteristics of the team has been addressed [54]. Although related to SPSP, RPP is significantly different from it. In SPSP, the solution to the problem results in an allocation of tasks to employees, aspect which is not always considered in RPP. Furthermore, in RPP, features are allocated to releases, while SPSP does not consider features to be a main entity to solve the problem, rather it considers tasks. On the contrary, tasks are usually not considered for solving RPP because individual tasks are only defined once an iteration begins. There have been proposals which combine both approaches such as [54].

After reviewing these surveys, we conclude that they are mainly focused on describing the state of the art around RCPS and the main techniques that have been used to address this and other related problems. We found that few surveys concentrate specifically on the SPSP and they do not present a study classification similar to what has been

done for RCPS, where aspects such as optimization objectives, algorithms used, and evaluation methodologies are considered. For this reason we perform a literature review to produce a survey with these characteristics.

3. METHODOLOGY

The first activity we performed for our survey was to identify the main research papers that present significant contributions to the solution of the SPSP. We selected the papers by following a protocol called a Systematic Mapping Study (SMS) [31]. This protocol considers the following steps:

1. Define research questions derived from research goals.
2. Define search strings.
3. Select search engines and build search expressions to be used on such engines.
4. Select papers according to inclusion and exclusion criteria.
5. Classify results.

The research questions that were defined in step 1 are listed in Table 1, where each question is derived from a particular research goal.

In step 2 of the SMS, we defined a set of search strings and their synonyms, which are shown in Table 2.

Table 1. Research questions for step 1 of the SMS.

Research Question ID	Research Goal	Research Question
RQ-1	Identify the main models that have been proposed to represent the SPSP and their elements.	Which SPSP models have been proposed?
RQ-2	Identify the aspects of software project planning that have been optimized in past studies.	Which are the most common optimization objectives in the SPSP?
RQ-3	Identify which kinds of algorithms or methods have been used in previous studies to solve the SPSP.	Which optimization techniques have been used to solve the SPSP?

Table 2. Search strings associated to the research questions.

Research Question	Search String	Synonyms
RQ-01	"Software project planning model"	-"SPSP model" -"Software management model" -"Project scheduling models" -"Software project planning representation"

	"Elements in modeling of software projects"	-"Software project scheduling components" -"SPSP model variables"
RQ-02	"Software project planning optimization objectives"	-"SPSP objectives" -"Project scheduling optimization" -"Optimization factors in the SPSP"
RQ-03	"SPSP metaheuristic optimization techniques"	-"Algorithms software project scheduling" -"Methods solving the SPSP" -"Optimization software project scheduling"
	"SPSP bio-inspired heuristics"	-"SPSP Soft computing" -"SPSP Computational intelligence -"SPSP Bio-inspired algorithms" -"SPSP Nature-inspired computing

Search strings are formed with groups of keywords that are related to the research questions defined in step 1 of the SMS, and the synonyms are groups of alternative or similar words to the keywords in the search strings.

In step 3, we used different research-oriented search engines including IEEE Xplore, ACM Digital Library, SpringerLink and Google Scholar, along with the Repository of Publications on Search Based Software Engineering (SBSE) [32]. We built queries using the search strings and the synonyms from step 2 and executed these queries on the search engines to obtain a preliminary list of research papers.

In step 4, we filtered the preliminary list of research papers using a set of inclusion and exclusion criteria to obtain a refined list of papers. Inclusion criteria are aspects or guidelines that identified papers must satisfy to be considered in the survey, while the exclusion criteria are elements used to reject papers that do not provide a significant contribution with respect to the research questions previously defined. In Tables 3 and 4 we present the inclusion and exclusion criteria, respectively.

Table 3. Inclusion criteria used in step 4 of our SMS.

Number	Criterion	Rationale
1	The research paper addresses the SPSP.	The focus of our survey is the SPSP.
2	The research paper describes optimization techniques used to solve the SPSP.	One of the goals of this survey is to identify the methods or algorithms used to solve the SPSP.
3	The research paper describes the methodology used to evaluate the proposal.	One of the dimensions of our analysis is the manner in which proposals are evaluated.
4	The research papers were published in a journal or in conference proceedings.	We consider these sources to be the most reliable.

Table 4. Exclusion criteria used in step 4 of our SMS

Number	Criterion	Rationale
1	The research paper does not tackle the SPSP or does not describe the methodology used to solve it.	It does not provide answers to our research questions.
2	The research paper is focused in a different problem, such as RCSP.	Our survey is focused in the SPSP.
3	The research paper is a preliminary study of one published later.	Our survey considers original work.
4	The result is a thesis or a book.	Due to their length, reviewing these documents would require a considerable amount of time.

The preliminary list of papers resulted in a total of 57 entries which, after filtering, was reduced to a total of 30 papers. The rest were discarded as they did not satisfy any of the inclusion criteria or because they satisfied at least one of the exclusion criteria.

From the 30 selected research papers, 16 (53%) were presented in conferences and 14 were published in journals. Sorted by search engine, 8 of the papers were obtained from Google Scholar, 4 from the ACM Digital Library, 6 from IEEE Xplore, 8 from SpringerLink, and 4 from the SBSE Repository. It is worth observing that Google Scholar and SpringerLink provided the largest amount of results for our study.

4. RESEARCH PAPERS OVERVIEW

Similar to what other surveys have done for problems such as RCPS, we define a set of perspectives, or dimensions, from which we will study the papers that were selected from the SMS. The dimensions that we considered for our survey are listed below, which allow the different papers to be compared more easily.

- Project planning model.
- Optimization objectives.
- Algorithms.
- Evaluation methods.
- Main results.

In the following subsections we review the selected papers according to these dimensions.

4.1 Project planning model

All of the research papers that we reviewed describe or present a project planning model which includes a set of elements with attributes and relationships, variables, constraints and assumptions considered for solving the SPSP. The majority of the reviewed papers use the model proposed by Alba and Chicano [1] as a basis, even though this model was originally created to solve the general Project Scheduling Problem (PSP).

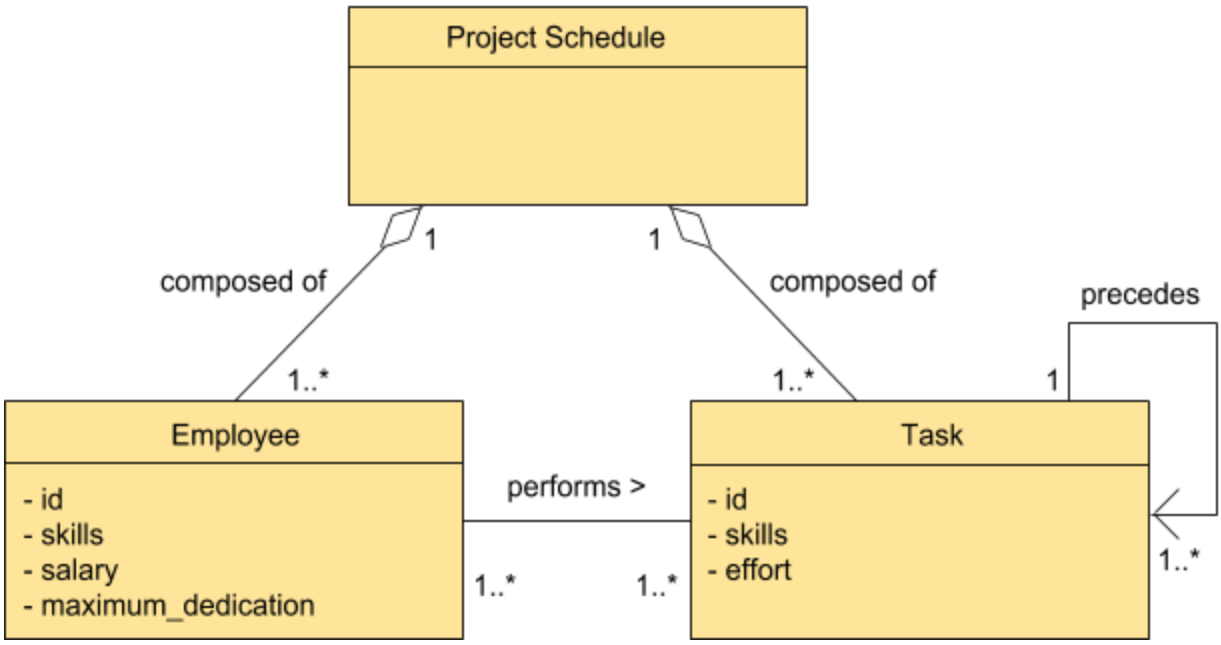

Figure 1. UML class model of the main entities used by Alba and Chicano [1] to solve the SPSP.

The model by Alba and Chicano, which we present in Figure 1 using a UML Class Diagram, represents that a project schedule is composed of a set of tasks and employees and, furthermore, employees are associated with tasks. Tasks are work units that may precede one or more other tasks.

Employees are human resources that are characterized by their skills e_i^{skills}, a monthly salary e_i^{salary}, and a maximum dedication time to the project e_i^{maxded}, where i is the employee number which ranges from 1 to E (the total number of employees). Skills represent capabilities possessed by the employees to perform particular types of tasks. In the context of software projects, it is possible to have skills in particular technologies or in activities such as programming, database management, design or others. Salary is a real number that is expressed in monetary units.

The maximum dedication is a real number that is a ratio between the amount of hours dedicated to a project and the duration of the work day of an employee (an employee that works full-time on a project has a maximum dedication of 1.0). On the other hand, tasks are the units of work which compose the project. Each task t_j is characterized by a set of required skills t_j^{skills} and an effort t_j^{effort} which is expressed in person-months. A task can be associated to one or more employees, and each employee is associated a level of dedication to that task. For example, if a full time employee e_i is assigned a dedication of 0.5 to a task t_j, this means that this particular employee dedicates half of his work day to that task.

Based on the effort associated to a task and the dedications of the employees to that task, it is possible to calculate the duration of a task:

$$t_j^{dur} = t_j^{effort} / \sum_{i=1}^{E} x_{ij}$$

Where E is the number of employees that are associated to task t_j and x_{ij} is the dedication of the employee e_i to the task t_j.

Furthermore, each task can have sequential dependencies to other tasks. When these dependencies exist, a task cannot start until the tasks it depends on (i.e. its predecessors) are finished. Once the duration of the tasks is obtained, the starting and ending times of the task can be calculated by considering its dependencies. The duration of the project is calculated by adding the duration of the longest tasks that must be performed sequentially (this is known as the *critical path* in project management).

The main constraints in this model are the following:

R1. No task can be left unassigned.

R2. The employees assigned to a particular task must together satisfy the skills required by the task.

R3. The employees must not exceed their maximum dedication time to the project.

In the papers we reviewed, [2], [4], [5], [13], [15], [21], [55], [56], [57], [58], and [59] reuse the model of Alba and Chicano to solve the SPSP without any changes. In [9] and [10] only the constraint R1 is reused and the maximum dedication of an employee is a constant (equal to 1). In the rest of the papers, the models exhibit some differences with respect to that of Alba and Chicano. In [8], the employees are classified according to four different skill levels:

beginner, junior, senior, and expert, where an employee with a beginner level also receives the lowest salary, while the employee with an expert level receives the highest salary. Furthermore, tasks have a required skill level, which corresponds to one of the four levels that were previously mentioned. In [12], five different skill levels are considered for the employees: novice, average, good, very good, and expert, while in [19] they are only classified as either novices or experienced. In [11] and [3] a productivity attribute also characterizes the employees, which represents the speed with which the employee completes his or her assigned tasks.

In the paper by Chang et al. [7], a timeline-based model is proposed. In this model, the effort performed by an employee on a particular task can vary across time (tridimensional model), as opposed to the model of Alba and Chicano, in which the dedication of an employee to a particular task is constant across time (bidimensional model). In Chang et al.'s model, employees are characterized by a base salary and by an overtime salary, while their skills are ranked between 0 and 5, where the value of 0 means that the employee does not possess the skill, while 5 means that the employee possesses the maximum capability with respect to that particular skill. This paper also considers a training model, which allows employees to improve their skills as the project progresses. Furthermore, the employees are characterized by a period of availability (with a start and end dates) and tasks are characterized by a deadline which is a date in which the task must be completed.

The models in other studies consider that projects are divided into work packages (WP), which are atomic work units, much like tasks, and which are associated to a development team. In [18], work packages are characterized by an id and an effort required for their completion, and there are no dependency constraints between work packages. In this case, the characteristics of the employees are not described. In [17] and [14], the employees possess a type of skill, and work packages require a particular type of skill to be completed. In both papers, the employees are grouped in development teams according to the skill they possess. In this way, each work package is associated with a team that possesses the required skill for its development. These three papers do not mention anything regarding the salary of the employees.

In [23], the Multistage Human Resource Allocation Problem, a problem related to the SPSP, is solved. This paper proposes a network model which divides a software project in n phases, for example design, coding, testing, and maintenance. A number of employees is distributed across the phases and the employees are characterized by a

monthly salary. In [11], a problem known as Resource Allocation for Software Release Planning is modeled. In this model, projects are divided into releases and the goal is to make an efficient resource assignment so that the planned delivery date for each release can be achieved. This model considers other types of resources besides human, such as material or financial.

In their study, Hanne and Nickel [22] present a model oriented towards optimizing quality (expressed as a number of defects), besides cost and duration in a software development project. Their model considers a set of developers who are responsible for performing development tasks (called items), which are further subjected to inspection and testing to detect and remove defects. As a consequence, in their model, employees only possess development, inspection and test skills.

The research papers that do not reuse Alba and Chicano's model also present variations in the constraints R1, R2 and R3. In [8], R1 and an additional constraint are considered. This additional constraint states that tasks must be associated to employees that possess a particular skill level required by the task. In [3], tasks must be assigned to an employee whose productivity is greater than zero in the corresponding task. In [7] and [11], tasks must not finish after a deadline, while in [23], the duration of a project must not exceed an established deadline and, in the same way, the cost of the project must not exceed an established maximum. In [22], different constraints associated with the development process are described. One of the main constraints is that a developer who is responsible for a particular development task cannot inspect or test the artifact produced by the development task. In [17], developers in one team must share a common set of expertise competences and, furthermore, the development teams are non-overlapping.

More recently, a dynamic version of the SPSP model has been proposed. In [60] and [61], the SPSP is formulated by considering uncertainties and dynamic events that often occur during software project development, and construct a mathematical model for the resulting Multi-objective Dynamic Project Scheduling Problem. The justification is that the changing environments of software companies mean that software project scheduling is a dynamic optimization problem. In this model, tasks effort is uncertain, that is, modifications in task specifications and inaccuracies in the initial estimations may cause changes in the initially estimated task effort. Moreover, new requirements could emerge during the development lifecycle of software, leading to new tasks. Regarding

employees, the model takes into account that employees can leave or return during the project. Additionally to the constraints in the model by Alba and Chicano, this model considers that there should be a maximum number of employees assigned to a task, however, it can be violated with a corresponding penalty.

In all of the research papers (except in [11]), it is assumed that more than one employee can participate in the development of a task or work package, and that the larger the number of employees associated to a task, the shorter its duration (except in [7], where a limit to the number of employees assigned to a task is imposed). Three papers present additional assumptions: in [8], it is assumed that an employee can be competent in tasks that belong to different project phases. In [7], it is possible to reassign employees during the project, as opposed to the model by Alba and Chicano where it is not possible to make changes to the employee's assignments. It is also assumed that the employees can improve their skills during a project and that it is possible to pause and restart tasks. In [14], it is assumed that new tasks can be added during the project's execution, that the real duration of some tasks may exceed their estimated duration and that, for simplicity purposes, work packages do not overlap with each other.

As we mentioned previously, the project planning model that was described in Alba and Chicano's paper is reused in many other research papers. For this reason, we decided to use it as a reference model and compare the characteristics of other models with respect to this one. Table 5 presents the differences and similarities that exist in the project planning models of the papers that were surveyed with respect to the reference model by Alba and Chicano [1].

Table 5. Differences between project planning models of the reviewed papers with respect to the model by Alba and Chicano [1].

Paper	Task attributes		Employee attributes			Constraints	Comments
	Effort	Required Skills	Skills	Salary	Maximum dedication		
[2]	EQ	EQ	EQ	EQ	EQ	R1, R2, R3	Same as reference model
[3]	EQ	NR	NR	EQ	NR	- All of the tasks must be performed by at least one employee who has a productivity > 0 in the corresponding task (this is the only constraint)	- Employees have a productivity attribute associated to tasks
[4]	EQ	EQ	EQ	EQ	EQ	R1, R2, R3	Same as reference model
[5]	EQ	EQ	EQ	EQ	EQ	R1, R2, R3	Same as reference model

[6]	EQ	EQ	EQ	EQ	DIF *	- A task must not be interrupted during its execution - Employees must work on a task from its beginning to its end (that is, without interruption).	* It is called overtime limit overloading limit
[7]	EQ	EQ	DIF*	DIF**	DIF***	- Tasks must not finish after the established deadline - There is a limit on the resources that can be assigned to a task.	* Employees skills are rated in a scale between 0 and 5. ** Employees are assigned a monthly base salary plus an additional payment for overwork *** Maximum dedication corresponds to the amount of hours an employee can work with respect to an established maximum, expressed as a percentage - Employees are also characterized by their availability in the project. -Tasks have an established deadline
[8]	EQ	DIF*	DIF**	EQ	NR	- R1 is reused. - Additional constraint: Each task must be assigned to at least one employee that possesses the skill level required by the task (Similar to R2).	* Employees possess a skill level: beginner, junior, senior and expert ** Tasks are characterized by a required skill level.
[9]	EQ	EQ	EQ	EQ	EQ*	- Only R1	* Maximum dedication of the employees is constant (equal to 1)
[10]	EQ	EQ	EQ	EQ	EQ*	- Only R1	* Maximum dedication of the employees is constant (equal to 1)
[11]	EQ	NR	NR	NR	NR	- Additional constraint: All the tasks that are planned for a particular release must be finished before its due date - Additional constraint: A feature must not require a larger amount of resources than those that are available.	- Employees are characterized by a productivity per type of task
[12]	NR	NR	DIF*	NR	NR	NR	* Employees have a skill level for different types of tasks. Skill levels are the following: novice, average, good, very good and expert.
[13]	EQ	EQ	EQ	EQ	EQ	R1, R2, R3	Same as reference model

[14]	NR	DIF*	DIF**	NR	NR	NR	* Each task requires a particular skill to be performed. ** Each employee has a single skill.
[15]	EQ	EQ	EQ	EQ	EQ	R1, R2, R3	Same as reference model
[16]	NR	EQ	EQ	EQ	NR	NR	
[17]	EQ	DIF*	EQ	NR	NR	- Additional constraint:: Members of the development teams must be in disjoint sets - Additional constraint: Members of a development team must possess a common set of skills - Additional constraint: Each work package must be associated with a team that covers the skill required by the work package.	* Each work package requires a single skill to be developed.
[18]	EQ	NR	NR	NR	NR	NR	
[19]	NR	NR	DIF*	NR	NR	NR	* Employees are of two types: novice or experienced. - Additional variables are considered including *project size* and *scheduled time*.
[20]	DIF*	EQ	DIF**	DIF***	NR	- Additional constraint: No employee must be assigned simultaneously to more than one task at any given moment.	* Tasks are associated an original processing time (initial). ** Employees skills are rated in a scale between 0 and 5. *** The cost of the employees is a daily cost
[21]	EQ	EQ	EQ	EQ	EQ	R1, R2, R3	Same as reference model
[22]	NR	NR	DIF*	NR	NR	- Additional constraint: The author of a coding artifact cannot be the inspector or tester of said artifact. - Additional constraint: A task with a higher priority must be scheduled before a task with a lower priority, if possible.	* The skills of the employees are of coding inspection and testing and are rated in the interval [0-1]. - Tasks are characterized by size, complexity and domain.
[23]	NR	NR	NR	EQ	NR	- Additional constraint: The duration of a project cannot exceed an established maximum duration. - Additional constraint: The total cost of the project cannot	- Additional input variables are considered, including the maximum duration and maximum cost for the project.

						exceed the established maximum cost.	
[55]	EQ	EQ	EQ	EQ	EQ	R1, R2, R3	Same as reference model
[56]	EQ	EQ	EQ	EQ	EQ	R1, R2, R3	Same as reference model
[57]	EQ	EQ	EQ	EQ	EQ	R1, R2, R3	Same as reference model
[58]	EQ	EQ	EQ	EQ	EQ	R1, R2, R3	Same as reference model
[59]	EQ	EQ	EQ	EQ	EQ	R1, R2, R3	Same as reference model
[60]	DIF*	EQ	EQ	EQ	DIF**	R1, R2, R3 - Additional soft constraint: There is a maximum number of employees assigned to each task.	* The initial effort of a task can change. ** The availability of an employee can change during the project. - New tasks can appear during the project.
[61]	DIF*	EQ	EQ	EQ	DIF**	R1, R2, R3 - Additional soft constraint: There is a maximum number of employees assigned to each task.	* The initial effort of a task can change. ** The availability of an employee can change during the project. - New tasks can appear during the project.

Constraints from Alba and Chicano's reference model [1]
R1: No task can remain unassigned.
R2: The employees assigned to a particular task must together satisfy the skills required by the task.
R3: Employees must not exceed their maximum dedication level in the project.

Notation:

EQ: The attribute is the same (equal) as in the reference model.
DIF: There are differences with respect to the reference model (differences are explained in the comments column).
NR: The attribute was not reported or not considered in the model.

4.2 Optimization objectives

In the SPSP, the objectives that are usually targeted for minimization are the duration and the cost of the project. The duration is the required time to complete all of the project's tasks, while the total cost is the sum of the costs of all of the tasks in which the project is divided.

In 18 of the 30 papers we reviewed, [1], [2], [3], [4], [5], [7], [9], [13], [15], [16], [20], [21], [23], [55], [56], [57], [58], and [59], the goal is to minimize only the project cost and duration, while in the rest of the papers there are some differences on the objectives that are considered.

In four papers, an additional objective to cost and duration is considered. In [8] and [6], the level of overtime (or

overload) in the project is considered for minimization. Overtime occurs when the employees exceed their maximum dedication time because they are assigned to tasks that coincide in a particular period, and to be able to finish them, the employees require to work extra hours. In [19], an additional objective is to maximize the average team productivity, while in [22] the third objective is to minimize the expected number of defects during coding activities in the project. In the remaining papers, some additional objectives are considered. In [11], the objective is to maximize the business value for a customer in each release. In [12], the optimization objectives are the duration and the number of defects that are expected to be injected by the developers according to their level of experience. In [10] and [17], the objective is to minimize both the duration and the fragmentation in the project. The fragmentation is the amount of time that a work team can remain inactive as a consequence of the ordering of work packages, based on the application of a queuing model.

In [14], three optimization objectives associated to time and robustness are considered. The first one is to minimize the duration of the project, the second is to minimize the difference in the ending dates when new tasks are added to the project, and the third is to minimize the difference between the initial estimated termination date and the actual termination date when some tasks require additional time to what was originally planned. In [18] the only objective that is considered is to minimize the duration of the project.

Four objectives are taken into account in [60], namely project duration, project cost, robustness performance, which evaluates the sensitivity of a schedule's quality to task effort uncertainties, and stability, which measures the deviation between the new and original schedules. Additionally to these four objectives, [61] considers a fifth objective, the employees satisfaction, which is related to the unwillingness of the employees to engage with the allocated tasks.

It is worth mentioning that even though most papers consider more than one optimization objective, in 14 of the 30 papers the objectives are evaluated in a composite objective function. For example, in [1], [13], [15], and [21], the objectives are evaluated using an inverted weighted sum:

$$f(x) = \{1/q \text{ if the solution is feasible and } 1/(q+p) \text{ otherwise,}$$

where $q = w_{cost} \cdot p_{cost} + w_{dur} \cdot p_{dur}$ and p is the penalty for infeasible solutions, that is, for those solutions that violate some of the problem constraints. In this expression, w_{cost} and w_{dur} are weights associated with the cost and

the duration, respectively. Minku et al. [9], [10], Wu et al. [58], and Xiao et al. [59] use a function that is similar to the previous one:

$$f(x) = w_{cost} \cdot p_{cost} + w_{dur} \cdot p_{dur} .$$

In [6], the objective function is the following:

$$f(x) = Validity \cdot (W_1/OverLoad + W_2/CostMoney + W_3/CostTime),$$

where *Validity* is a penalty that is imposed when the assignments between tasks and employees are incomplete, *OverLoad* is the amount of overtime work performed by the employees in a project, *CostMoney* is the monetary cost of a project, *CostTime* is the time required to finish a project, and W_1, W_2, and W_3 are weights that are associated with the overload, cost, and time objectives, respectively.

In the remaining 14 research papers, the SPSP is formulated as a multi-objective optimization problem:

$$minimize\ f(x) = (f_1(x), \ \ldots, f_k(x)),$$

subject to the corresponding constraints, where x is a solution to the problem and k is the number of objective functions. Table 6 presents a comparison of the research papers that were reviewed according to the optimization objectives.

Table 6. Optimization objectives considered in the research papers that were reviewed

Paper	Objectives						Type of optimization
	Duration	Cost	Overload	Number of defects (Quality)	Fragmentation	Other	
[1]	✓	✓					S
[2]	✓	✓					M
[3]	✓	✓					M
[4]	✓	✓					M
[5]	✓	✓					M
[6]	✓	✓	✓				S
[7]	✓	✓					S
[8]	✓	✓	✓				M
[9]	✓	✓					S
[10]	✓				✓		S

[11]						✓	S
[12]	✓			✓			M
[13]	✓	✓					S
[14]	✓					✓	M
[15]	✓	✓					S
[16]	✓	✓					S
[17]	✓				✓		M
[18]	✓						S
[19]	✓	✓				✓	M
[20]	✓	✓					M
[21]	✓	✓					S
[22]	✓	✓		✓			M
[23]	✓	✓					M
[55]	✓	✓					S
[56]	✓	✓					S
[57]	✓	✓					S
[58]	✓	✓					M
[59]	✓	✓					M
[60]	✓	✓				✓	M
[61]	✓	✓				✓	M

Notation:

S: Single objective problem
M: Multi-objective problem

4.3 Algorithms

In [24], [25], and [26], which were not part of the studies we compare, the authors mention that they mainly used exact techniques and heuristic methods to solve RCPS. This is to be contrasted with the papers we surveyed, which are focused on the SPSP and where metaheuristics were used to solve the problem. Metaheuristics are search methods that find good solutions to optimization problems in a reasonable time, however, there is no guarantee that these solutions are optimal [49]. The main reason why metaheuristics have been used to solve the SPSP is that, when

the number of tasks and employees in a project increases, the number of potential solutions grows exponentially. This exponential increment in the quantity of potential solutions makes it impractical to generate, in a reasonable time, all of the possible solutions in order to find the one with the best duration and cost. A solution to the SPSP can be represented as a matrix of degrees of dedication that associates employees to tasks, which allows the cost and duration of the project to be calculated. In this kind of matrix, there is one column for each task and one row for each employee. Each entry (i,j) in the matrix corresponds to the degree of dedication of employee e_i to task t_j, and it has a value between 0.0 and 1.0.

There are different types of metaheuristics, being one of the most frequently used to solve the SPSP the Evolutionary Algorithms (EAs). From the 30 papers we reviewed, 27 used EAs, either as the proposed technique to solve the problem or as a reference to compare the performance of their proposals.

EAs imitate the evolution process of the species, where the fittest individuals have a higher probability to survive [33]. These algorithms are based on a population, where each individual is represented as a chromosome which contains information about its characteristics. In this case, survival is achieved through reproduction and the descendants are generated following a process called crossover or recombination. Individuals in a population may suffer modifications in some of their characteristics, procedure known as mutation. The probability of survival of an individual is evaluated using a fitness function. Fitness is a score, or value, that is associated with each individual and which reflects the quality of the objective functions evaluations of the problem being solved.

One type of EA that has been used most frequently to solve the SPSP is the Genetic Algorithm (GA). GAs were proposed in the 1970's by John Holland [43]. They consist in a search method that takes a population of individuals which are modified iteratively to find a global optimum. The problem's variables are coded as chromosomes and are manipulated using genetic operators to improve their fitness in each generation. The main steps of a GA are the initialization of the population, the evaluation of that initial population, and then iteratively: selection, recombination, mutation, evaluation, and replacement are applied to the population obtained in a generation until a termination criteria is achieved.

GAs were used in 19 of the reviewed papers. There are some differences, however, with respect to the way the chromosome is encoded and with respect to the selection, mutation and crossover operators. Papers [1] and [21]

report the use of a binary chromosome encoding to represent the degree of dedication of the employees to the tasks in the project, where each degree of dedication is represented using three bits. In this way, if there are E employees and T tasks, the length of each chromosome is $E \times T \times 3$.

García and Gómez [8] used an integer encoding in their Multi-Objective Genetic Algorithm (MOGA) to represent the solutions. Each chromosome has a length of size $m \times n$, where the first n integers correspond to the dedication time of the first employee to the m tasks, the following n integers correspond to the dedication time of the second employee, and so on.

On their papers, Chang et al. [6], [7] used a library called GALib, which includes a variety of encodings and operators that can be adapted to solve different optimization problems. In [6], the GA was applied to obtain employee assignments to tasks in a multi-project environment to minimize three objectives, while in [7] the GA is used to obtain a project calendar with the smallest cost. In [11], GALib was also used, but in this case the GA was applied to obtain optimal assignment of developers to tasks from releases in the project plan. [23] proposes a MOGA based on an encoding method called Improved Fixed-length Encoding and a mechanism of adaptive fitness weight assignment to find solutions close to the optimal.

In the 12 remaining papers, GAs were applied to compare their performance with respect to other algorithms. For example, [13] and [9] compared the performance of the GA used by Alba and Chicano [1] against an Ant Colony Optimization (ACO) algorithms and against an improved version of (1+1) EA, respectively.

In other studies, the problem was addressed as a multi-objective optimization problem, which aim is to find solutions that represent trade-offs between the multiple objectives. When these solutions represent the optimal trade-offs, they are known as Pareto optimal set and their corresponding functions evaluations as Pareto front.

These studies tackled the multi-objective problem by means of multi-objective evolutionary algorithms (MOEAs). One of the main characteristics of these approaches is that, when the objective functions are evaluated, a so-called non-dominated set of solutions is obtained. This non-dominated set is obtained when an inverse correlation is present in the problem's objectives [47], which means that the improvement in one objective corresponds to the deterioration of the others. Generally, in the SPSP, the solution with the shortest duration has the highest cost, while the solution with the smallest cost has the longest duration.

Chicano et al. [2] evaluated the performance of five metaheuristics: three classic ones, including NSGA-II [34], SPEA2 [35], and PAES [36], and two that were recent when their paper was written, which are MOCell [37] and GDE3 [38]. These metaheuristics have the capability of finding a set of non-dominated solutions with respect to the optimization objectives in a single execution. NSGA-II and SPEA2 use binary tournament, simulated binary crossover, and polynomial mutation as evolutionary operators. The difference between these algorithms resides in the mechanism used to obtain a set of solutions, where the values of the objectives have the most uniform distribution along the Pareto front. PAES uses polynomial mutation and does not require a crossover operator, MOCell stores non-dominated solutions in an external archive, and GDE3 uses differential evolution selection and crossover operators.

In [3], the same authors make a comparison of the metaheuristics presented in their previous work [2] (except for GDE3) to evaluate their performance in the solution of a SPSP model with more realistic characteristics, such as the introduction of a delay before a task can be initiated and the fact that there can be hundreds of tasks in a project.

In [5], the performance of eight multiobjective algorithms was evaluated to solve an SPSP model with a greater number of tasks and employees. As in the previous studies, three of the algorithms that are traditionally used to solve multi-objective problems (NSGA-II, SPEA2, and PAES) were applied, along with five algorithms that were considered recent at the time of writing, namely DEPT [39], MO-FA [40], MOABC [41], MOCell [37], and GDE3 [38]. In this paper, the researchers wanted to understand which one of the algorithms offered better scalability and which one explored the objective space more efficiently to obtain higher-quality solutions in terms of cost and duration.

In [19], NSGA-II was used in conjunction with a dynamics systems simulation model for project management to generate a non-dominated set with two or more optimization objectives. In [12], an elitist version of NSGA was applied, which uses a single point crossover and the roulette wheel selection to ensure that good solutions are not lost in the process. Shen et al. [60] proposed a MOEA based on ε-MOEA [62] for solving the dynamic version of SPSP, which generates a robust schedule predictively and adapts the previous schedule in response to critical dynamic events during the project execution.

In [9], a (1+1)-EA algorithm was applied. This is one of the simplest evolutionary algorithms as it does not use a

population of individuals or a crossover operator. In each generation, the selection operator verifies if a mutation has found an improvement. The fitness of an individual is evaluated using a composite objective function which is the same as in [1].

Besides EAs, other types of metaheuristics have been applied for solving the SPSP. In six papers, metaheuristics based on the Swarm Intelligence paradigm were used: Ant Colony Optimization (ACO) was used in four studies, Particle Swarm Optimization (PSO) was used in one, and the Firefly Algorithm was applied in the remaining one. This paradigm is based on the study of the social behavior of organisms (individuals) in swarms or colonies and its goal is to find local optima and a global optimum for a problem [33].

First, Gonsalves and Itoh [20] designed a multi-objective variant of PSO with the goal of minimizing duration and development costs. PSO is a population based search method, in this case the population is a particle swarm. Each particle moves (flies) in the search space to find the best possible position (i.e. the optimum), considering its own position and those of its neighbors. In this way, each particle represents a candidate solution to an optimization problem. The work by Gonsalves and Itoh is one of the first that follows this paradigm to solve the SPSP. On the other hand, in [13], [16], [55], and [59], ACO was used to solve the SPSP. ACO is a search method that is based on the social behavior of ants. Ants communicate indirectly to find the shortest path between the colony and a food source [33]. With ACO, the SPSP becomes a graph-based search problem. Assuming that employees can contribute with a percentage of their dedication time to each task, each task is divided in a network of nodes and arcs (a graph), according to the number of employees and possible dedication time percentages. Ants traverse the graph to find a distribution of dedication times of employees to a task. Once the dedication times are obtained, the solutions are evaluated according to a fitness function, which is the same expression used by Alba and Chicano [1]. This process is repeated until the established number of generations is reached. In [13], the purpose was to demonstrate that ACO presents, on average, a better performance than a GA in areas such as the success rate, the minimization of the project duration and cost, and the final fitness. In [16], an elitist version of ACO is proposed to obtain effective assignments that contribute in minimizing the duration and cost of the software project.

Jin and Yao [15] applied Population-Based Incremental Learning (PBIL) to solve the Team Software Project Management Problem. PBIL is a method that uses an evolving probability vector to obtain candidate solutions, as

opposed to GAs which are methods based on population. Crawford et al. [56] applied the Intelligent Water Drops, which is a constructive metaheuristic inspired on the behaviour of water drops that flow into a river in search of an optimal path to reach their destination.

In [17] and [18], a comparison is made of the performance of search-based techniques including GAs, Hill Climbing, and Simulated Annealing, together with a queue simulation model to optimize the distribution of resources in a software project. In [18], two genome encoding schemes are used: pigeon hole genome and ordering genome. Pigeon hole genome describes a genome as an array of *N* integers, where *N* is the number of work packages in a project and each value in the array indicates the number of the team that is assigned to each package. The ordering genome considers an array of size *N* where each value indicates the position of a work package in the queue that is used. The fitness function is simply the project's deadline. In [17], the GAs that were applied were NSGA-II and a single-objective GA. The main objective of this research was to demonstrate that search-based techniques can be used to solve problems associated with planning software projects. Finally, in [7], besides using a GA, Hill Climbing is used to evaluate and compare the performance of both algorithms. In Hill Climbing, a population of initial solutions is selected and the best of these solutions is determined to be the starting point. This solution suffers a mutation in a random position and its fitness is evaluated. If the fitness is higher that the one from a previous solution then a replacement occurs. This process is repeated until an optimum is found.

Table 7 presents a classification of the reviewed papers according to the optimization algorithms used in the solution of the SPSP or in the evaluation of their performance against other algorithms.

Table 7. Optimization algorithms (metaheuristics) used in the reviewed papers to solve the SPSP

Paper	EVOLUTIONARY ALGORITHMS					SWARM INTELLIGENCE		Other Algorithms
	GA	NSGA-II	SPEA-II	PAES	Other MOEAs	PSO	ACO	
[1]	✓							
[2]		✓	✓	✓	-MOCell -GDE3			
[3]		✓	✓	✓	-MOCell			
[4]		✓		✓	-DEPT -MO-FA			
[5]		✓	✓	✓	-DEPT -MOCell			

					-MOABC -MO-FA -GDE3			
[6]	✓							
[7]	✓							HC
[8	✓	✓						
[9]	✓							(1+1) EA
[10]	✓							(1+1)EA and RS
[11]	✓							
[12]		✓ (NSGA)						
[13]	✓						✓	
[14]			✓					RS
[15]	✓							PBIL
[16]							✓	
[17]	✓	✓						HC and SA
[18]	✓							HC and SA
[19]		✓						
[20]						✓		
[21]	✓							
[22]					-Custom MOEA			
[23]	✓	✓						
[55]							✓	
[56]								IWD
[57]								FA
[58]								EHH
[59]							✓	
[60]					- Based on ε-MOEA			
[61]								MA

Notation: GA: Genetic Algorithm NSGA-II: Non-dominated Sorting Genetic Algorithm II SPEA2: Strength Pareto Evolutionary Algorithm 2 PAES:Pareto Archived Evolution Strategy EHH: Evolutionary Hyper-heuristic MA: Memetic Algorithm	PSO: Particle Swarm Optimization ACO: Ant Colony Optimization HC: Hill Climbing SA: Simulated Annealing RS: Random Search PBIL: Population-based incremental learning IWD: Intelligent Water Drops FA: Firefly Algorithm

4.4 Evaluation methods

In this Section we present a summary of the evaluation of the techniques used for solving the SPSP, including the benchmark sets used and the parameter settings for the algorithms.

4.4.1 Benchmark instances

With the aim of evaluating the performance of the optimization algorithms while solving the Software Project Scheduling Problem, benchmark instance sets were defined in some studies. A benchmark instance consists of a software project configuration which includes the number of tasks, the number of employees and, in some cases, the skills the employees have and the necessary skills to conclude a task.

We observed that 11 studies took into account, totally or partially, the benchmark instances used by Alba and Chicano [1]. They generated software projects with artificial data, which considers 5, 10, 15, and 20 employees, 10, 20, and 30 tasks, 5 and 10 global skills, and 4-5 or 6-7 skills by employee, which add up to 48 benchmark instances. These 48 configurations were used in [13], [9], [10], [55], [56], [57], and [58], while [2], [8], [21], and [59] used 36 of these instances, since the cases with 20 employees were not considered. Luna et al. [4], [5] considered a higher number of employees and tasks with the aim of testing the scalability of the algorithms they used, that is, to measure the performance of the algorithms when the instance size increases. In both studies, 36 benchmark instances were used, where the number of employees and tasks ranged between 8 and 256, and between 16 and 512, respectively. Shen et al. [60] and Shen et al. [61] generated 18 instances for the dynamic SPSP based on those from [1].

Other studies considered configurations with a number of tasks and employees different from those used by Alba and Chicano [1]. For example, in [6], Chang et al. used projects with 18 tasks and 10 employees, while Chang et al. [7] used projects with 3, 15, and 25 tasks, and 3, 10, and 15 employees. Five studies used data from real projects. In [18], data from a project with 84 work packages (WPs) and from 10 to 40 employees was used. In [17], the authors used data from two projects with 84 and 108 WPs, and 20 and 46 employees, respectively. In [14], data from four projects was used, which considered, respectively, 79, 84, 115 and 120 WPs, and 14, 20, 7, and 20 employees. In [60] and [61] three instances were considered, with 12 and 15 tasks, 5 skills, and 10 employees. In five studies [3], [11], [15], [16], and [22] configurations with 100 tasks (or activities) or less are reported, and eight studies [3], [11],

[12], [15], [16], [19], [20], and [22] considered configurations with 20 or less employees. The remaining paper [23] does not provide any information regarding the project configurations that were used.

4.4.2 Parameter settings of the algorithms

In this Section we describe the parameter settings of the algorithms used in the studies for solving the SPSP. Since 24 out of the 30 studies used EAs for tackling the problem, we describe the settings based on the parameters that are usually considered in this kind of algorithms: population size, the evolutionary operators, selection, crossover and mutation, and the maximum number of generations.

In eight of the 30 papers [2], [3], [4], [5], [11], [18], [60], and [61], the algorithms were configured with a population of 100 individuals (except for those that applied PAES, since its evolutionary strategy only requires one individual). The population size in [1], [9], [10], [13], [15], [21], and [58] was 64, while in five studies [6], [17], [19], [22], [23], [56], and [59] the algorithms were configured to use 60 or less individuals. Only in [7], [14], and [55] the population size was larger than 100. Five studies [8], [12], [16], [20], and [57] did not provide information about the size of the the population that was used.

Regarding the evolutionary operators, the main individual selection mechanisms were binary tournament and roulette wheel selection. In the binary tournament selection, two individuals are randomly chosen and, from these, the one with the highest fitness is selected to undergo crossover. The roulette wheel selection simulates the spin of a roulette wheel. Every individual in the population is assigned a pocket of the roulette wheel, which size is proportional to its fitness, that is, fittest individuals are assigned a bigger pocket, hence there is a higher probability that they will be selected. From the studies which provide information about the selection method, ten reported using binary tournament and four opted for the roulette wheel method.

Furthermore, the recombination or crossover operators most frequently used were the single point crossover (SPX) and the simulated binary crossover (SBX). In the former, one crossover point is selected for the two parents. Children are formed by copying the genetic information from the first parent, from the beginning of the chromosome to the crossover point, and from the second parent from the crossover point to the end of the chromosome. In the latter, SBX is an operator used for solving continuous problems, it simulates SPX but with real

numbers. In seven studies SPX is used, SBX is considered in three studies, and the two points crossover operator is used in one study. In [8], two crossover operators were used: flat crossover and simple crossover. Finally, the crossover probability most frequently set is 0.9 (in eight studies).

The mutation operators most commonly used are bit-flip (in six studies) and polynomial mutation (in three studies). In [8], two mutation operators were used, namely gradient mutation and random mutation. The mutation probability was regularly set to 0.3 and lower (in 10 studies) and $1/L$ (in nine studies), where L is the chromosome length.

Finally, the number of generations was variable among the studies. 13 studies reported a maximum number of generations less or equal to 1,000, in eight studies it was between 1,000 and 10,000, in four studies it was between 10,000 and 100,000, and in one study they set a maximum of one million generations.

In order to obtain statistically significant and reliable results, the algorithms were run a predefined number of times to solve each benchmark instance. In the surveyed studies, we observed that the algorithms were run between 10 and 100 times. In [18], [55], and [59] the number of runs was 10, in [15] and [23] they were 20 runs, in [4], [5], [14], [17], [21], [56], [58], [60], and [61] each instance was solved 30 times, in [22] authors did 74 executions of their algorithm, and in [1], [2], [3], [9], [10], [13], and [57] the average number of runs was 100. Eight studies did not provide information about the number of runs that were performed.

Furthermore, some studies used public libraries, which provide implementations of popular algorithms and operators for solving optimization problems. For example, Chang et al. [6] and Chang et al. [7] used a library called GALib [44], while in [2] and [9] a framework called JMetal [45] was used.

The vast majority of the studies used artificial project data using an instance generator developed by Alba and Chicano [1]. This tool generates random project data including the number of employees, tasks, skills, and the associated task precedence graph. Only a few studies used data from real projects, for example, [18], [17], and [14], where data from a project in an European financial organization was considered. Furthermore, the latter study also considered data from an North American corporation.

In Table 8 we provide data from the parameter settings of the algorithms and data regarding the benchmark instances that were used.

Table 8. Parameter settings and benchmark characteristics from the surveyed studies.

Study	Algorithm' parameter settings				Benchmark instances			
	Population size	**Evolutionary operators (only EAs)**	**Maximum number of generations**	**Number of runs per instance**	**Number of tasks**	**Number of employees**	**Number of benchmark instances**	**Source**
[1]	64	Selection: BT Crossover: 2-D SPX Mutation: bit-flip	5,000	100	10, 20, 30	5, 10, 15, 20	48	Instance generator/ public data[1]
[2]	100, except PAES	Selection: BT Crossover: SBX, cp=0.9 Mutation: polynomial, mp=1/L	100,000	100	10, 20, 30	5, 10, 15	36	Instance generator/ (from [1])
[3]	100, except PAES	Selection: BT Crossover: two-point, cp=0.9 Mutation: random, mp=1/L	10^6	100	29	7	2	Data from a project plan[2]
[4]	100, except PAES	NSGA-II: Crossover: SBX, pc=0.9 Mutation: polynomial, mp=1/L PAES: Mutation: polynomial, mp=1/L	100,000	30	16 a 512 (powers of 2)	8 a 256 (powers of 2)	36	Instance generator/ public data[3]
[5]	100, excepto PAES	Selection: BT Crossover: SBX, cp=0.9, Mutation: polynomial, mp=1/L	100,000	30	16 a 512 (powers of 2)	8 a 256 (powers of 2)	36	Instance generator/ (data from [4])
[6]	30 and 60	Crossover: single point crossover, cp=0.65 to 0.9 Mutation: bit-flip, mp=0.0015 to 0.003	500 to 10,000	*	18	10	*	*
[7]	500 to 5,000	Mutation: mp=0.0001 to 0.1	100 to 2,000	*	3, 15, 25	3, 10, 15	*	*
[8]	*	Selection: BT Crossover: flat crossover and simple crossover Mutation: gradient mutation and random mutation, mp= 0.1	*	*	10, 20, 30	5, 10, 15	36	Instance generator (data from [1])
[9]	64	*	5,064	100	10, 20, 30	5, 10, 15, 20	48	Instance generator (data from [1])
[10]	64 (GA) 1 (EA)	Selection: BT Crossover: 2-D SPX	5,064	100	10, 20, 30	5, 10, 15, 20	48	Instance generator

[1] The public data is available from http://tracer.lcc.uma.es/problems/psp/generator.html
[2] The data is available from the MS Project online repository (no link provided)
[3] Available from http://mstar.lcc.uma.es/problems/swscheduling.html

		Mutation: bit-flip and uniform mutation						(data from [1])
[11]	100	Mutation: mp=0.01	500	*	60	6	600	* Random data
[12]	*	Selection:RW Crossover: SPX	*	*	*	6	*	*
[13]	64	Selection: BT Crossover: 2-D SPX Mutation: bit-flip, mp=1/L	5,000	100	10, 20, 30	5, 10, 15, 20	48	Instance generator (data from [1])
[14]	250	Crossover: cp=0.8 Mutation: mp=0.1	180, 250	30	79, 84, 115, 120 (WPs)	7, 14, 20	*	Data from real projects
[15]	64	Selection: BT Crossover: 2-D SPX Mutation: bit-flip, mp=1/L	5,000	20	10	5	4	*
[16]	**	**	*	*	10 (activities)	8	*	*
[17]	50	Crossover: SPX, cp=0.7 Mutation: mp=0.1	100, 250	30	84, 108 (WPs)	20, 46	*	Data from real projects
[18]	100	Crossover: cp=0.6, Mutation: mp=0.1	1,000	10	84 (WPs)	10 to 40 (multiples of 5)	*	Data from a real project
[19]	50	*	200	*	*	2 to 20	*	*
[20]	**	**	*	*	*	7	*	*
[21]	64	Selection: BT and RW Crossover: AG_1X and AG_2DMX; cp=0.3, 0.5, 0.75 and 0.9 Mutation: bit-flip, mp=0.005	20,000	30	10, 20, 30	5, 10, 15	36	Instance generator (data from [1])
[22]	30	Crossover: cp=0.45 and 0.15 Mutation: mp=0.15	1,000	74	100	20	*	* Random data
[23]	20	Selection:RW Crossover: cp=0.7 Mutation: mp=0.3	500	20	*	Maximum 10 per phase	*	*
[55]	200	**	1,000	10	10, 20, 30	5, 10, 15, 20	48	Instance generator (data from [1])
[56]	50	**	1,000	30	10, 20, 30	5, 10, 15, 20	48	Instance generator (data from [1])
[57]	*	**	1,000	100	10, 20, 30	5, 10, 15, 20	48	Instance generator (data from [1])

[58]	64	Crossover: swap-row, swap-column, wap-block, cp=0.75 Mutation: mutate-position, utate-row, mutate-column, mp=0.1	200	30	10, 20, 30	5, 10, 15, 20	48	Instance generator (data from [1])
[59]	24	**	3,000	10	10, 20, 30	5, 10, 15	36	Instance generator (data from [1])
[60]	100	Crossover: 2-D SPX, cp=0.9. Mutation: bit-flip, mp=1/L.	10,000 function evaluations	30	10, 20, 30	5, 10, 15	18	Adapted from instances from [1]
[61]	100	Crossover: 2-D SPX, cp=0.9. Mutation: bit-flip, mp=1/L.	15,000 function evaluations	30	10, 20, 30	5, 10, 15	18	Adapted from instances from [1]
Notación: BT: Binary tournament RW: Roulette wheel cp: crossover probability mp: mutation probability L: individual length (number of task × number of employees)					* Data not provided. ** Does not apply.			

4.5 Main results

Results from the surveyed studies were diverse. In some of them, e. g. [2] [3], [4], [5], [60], and [61], the objective was to evaluate the performance of the algorithms used for solving the SPSP, by measuring the quality of the solutions found in terms of a number of performance indicators: hypervolume, attainment surface, generational distance, spacing, and spread. The hypervolume indicator evaluates convergence and diversity for a set of solutions obtained with an algorithm; the attainment surface distinguishes the regions of the objective space that are better explored by the algorithms; the generational distance measures how far the obtained solutions are from a reference set; the spacing measures the distance variations of neighbouring vectors in the obtained set of solutions; and the spread measures the extent of spreading achieved by the obtained solutions. When these indicators were applied in [3], MoCell was the most appropriate algorithm for solving the proposed SPSP model, which simulated more realistic problem situations. In [4], PAES found better solutions for a scalable version of the SPSP (that is, with a greater number of tasks and employees), while MO-FA and NSGA-II were the second best: the former for instances with a smaller number of tasks and employees, and the latter for instances with a larger number of both. In [2], PAES has reported the higher (better) hypervolume values in 25 out of 36 instances (i.e. PAES has reached the best

approximations to the Pareto fronts), while NSGA-II and SPEA2, have reported the lowest hypervolume values. In [5], PAES was the algorithm which provided the best quality solutions, and also the ones with the best cost and time. Shen et al. [60] compared their proposed ε-MOEA robust approach for the dynamic SPSP against the deterministic version of ε-MOEA. Finally, Shen et al. [61] compared the performance of their memetic algorithms against another seven recent optimizers.

In eight studies the algorithms were evaluated according to the hit rate, that is the percent of runs in which the algorithm found feasible solutions for each problem instance, i. e. solutions which do not violate the problem constraints. Alba and Chicano [1] observed that, when the number of tasks and employees is increased, the problem becomes more difficult to solve, mainly because of the overload constraint. In fact, the hit rate was zero for instances with 30 or more tasks, since in these cases the algorithms were unable to find at least one feasible solution. Given this problem, three studies proposed a mechanism to repair the solutions to achieve a 100% hit rate in all, or nearly all of the benchmark instances. In [8], the proposed MOGA was able to find feasible solutions, that is a 100% hit rate, for the majority of the instances where an employee is allowed to perform 2, 4, and 8 tasks per day. Moreover, the solution sets found by the proposed MOGA were of higher quality than those found by NSGA-II [34], since the hypervolume covered by the former was significantly larger. In [9] and [10], a normalization method was proposed to avoid the maximum dedication constraint from being violated, this resulted in a 100% hit rate for all instances.

Finally, when comparing GAs and other metaheuristics, results were also diverse. In [13], [55], and [59] the proposed ACO obtained a higher hit rate for most of the instances with respect to the corresponding GA. Furthermore, ACO found solutions with better project duration and cost. Some approaches which considered other kind of search-based strategies, obtained better results than those from a GA. In [17] and [18], for example, Simulated Annealing surpassed a GA while minimizing the project completion time. In [7], Hill Climbing overcame a GA while minimizing the optimization objectives for relatively small instances, though the GA was superior for more complex cases involving a higher number of tasks or employees. The authors of this study concluded that the combination of a GA with Hill Climbing is not appropriate for solving this type of complex cases.

5. DISCUSSION

From the literature review, we have observed that the approaches used to solve the SPSP show both differences and similarities across the dimensions that we considered in our classification of the research papers.

Firstly, we identified that the project planning model from Alba and Chicano [1] is used as a basis or reference model in the majority of the papers we have reviewed. Some researchers consider additional aspects in their own project planning models, such as different skill levels and the level of expertise of the employees (junior, senior, etc.), their productivity, interruptions to the tasks, and inclusion of new tasks during project execution, among other features. It should be noted that, in the majority the papers, it is assumed that a task can be performed by more than one employee and that, in general, the duration of the task is inversely proportional to the number of employees that are associated with the task.

We found that the most commonly used optimization objectives in the SPSP are the project's cost and duration, although some researchers consider alternative objectives such as the overload, the number of expected defects, and some objectives associated with time, like project overtime and the task termination time when the duration of the tasks can be modified or when new tasks can be added to a project.

The most frequently used algorithms used to solve the SPSP in these studies are GAs and MOEAs, such as NSGA-II [34], SPEA2 [35], PAES [36], and MOCell [37]. Some less used algorithms included metaheuristics like ACO [33], PBIL [15], and PSO [20], and search-based techniques like Hill Climbing and Simulated Annealing [18]. Regarding the evolutionary operators, the individual selection operators most often used were binary tournament and roulette wheel, the recombination operators most frequently used were SPX and SBX, and the mutation operators most generally used were bit-flip and polynomial mutation.

Regarding the evaluation of the research studies, some similarities were observed. For example, several researchers used the benchmark instances proposed by Alba and Chicano [1], which consist in projects composed of 5, 10, and 15 employees, 10, 20, and 30 tasks, and 5 and 10 skills. In other studies, some configurations with a higher number of tasks and employees were used to measure algorithm scalability.

In three of the six studies that report the hit rate of the algorithms, the results are equal to zero for problems with

30 or more tasks. From our point of view, this restricts the possibilities of using these approaches in real-world projects, as in practice it is highly probable that project plans are composed of a larger number of tasks. We believe this problem results from the constraint that prevents task overload from occurring. To overcome this limitation, [9] and [10] proposed mechanisms to normalize the dedication of employees to tasks so that employees do not exceed their maximum dedication time. These mechanisms allow a 100% hit rate to be obtained when solving most of the instances of the SPSP.

As a result of the literature review, we now discuss some areas of opportunity following the dimensions we used in Section 4. Moreover, the future research we suggest is based on what we have observed in practice.

5.1 Project planning model

It is important to model SPSP as close as possible to real projects in order to apply those models for solving actual problems. For this reason, we propose the following elements that could be considered in the SPSP model.

Firstly, we discovered that in all but four of the project planning models that were identified, there is no distinction with respect to the skill level, technical knowledge, and the level of expertise of the employees. We believe these features need to be taken into account in the SPSP models, since the employees in a software development organization usually possess different capabilities and levels of knowledge, on which their productivity actually depends. Moreover, we observed that only three of the reviewed papers considered the productivity of the employees with respect to their associated tasks. We think this factor should be given a greater importance in SPSP models since productivity has a direct influence on the time required to perform the tasks.

One of the assumptions that is made in the papers (except for [7]) is that, when a higher number of employees are associated with a particular task, its duration is reduced. As Chang et al. [7] state, in actual projects, only a limited number of employees can efficiently work together in the development of a particular task. A higher number of employees working on a task requires more communication, which in turn requires more time to coordinate the employees. As a consequence, there is a negative impact on the duration of the task when more employees are involved. Hence, in order to apply this model in real-life projects, the number of employees working on the same task should be limited.

In practice, it is common that an employee works part-time in more than one project. Thus, when creating the project plan, the availability of that particular employee depends on his assignments with respect to the other projects. This aspect could be modeled by changing the maximum dedication attribute of an employee from a single value to a temporal variable, that is, one that changes over time. This is similar to the timeline-model proposed by Chang et al. [7], however, in this case, the availability would be derived from the other project's plans.

One situation that is rarely considered occurs when a project schedule is modified by adding or removing tasks or by changing their duration. While [14] considers this, it is desirable that the updated plan remains as close as possible to the original allocation of employees to tasks, since many changes could result in overhead costs.

Finally, it is frequent that projects are developed by teams of people who work in different geographic locations, thus, when a task is performed by several employees, it may be desirable to select employees who are geographically close. This aspect could also be considered in the SPSP model by adding location information, to reduce transportation costs and time differences when the team is distributed geographically.

5.2 Optimization objectives

Most research has basically considered two optimization objectives, namely the project cost and the project duration. Although these two objectives are typically the most important in the customer-provider relationship, we believe additional objectives could be considered, which, despite not having a direct influence on the project cost and duration, could lead to a lasting business relationship with both customers and employees.

For instance, customer satisfaction is an important issue for doing continuous business between customers and providers. A customer may want, for example, to work with particular individuals or a particular team. Another objective could be to reduce employee turnover as this typically has a negative impact if it occurs during the execution of a project. Employee turnover may be reduced by assigning employees to tasks where they most excel according to their particular profile.

It is important to mention that only some of the research papers that were reviewed explain the behavior of the optimization objectives with respect to the characteristics of the benchmark instances. We believe it would be worthwhile to analyze the correlation of the results according to the characteristics of the configurations, similar to

what was done by Alba and Chicano in their research [1].

5.3 Algorithms

Many approaches have been proposed for tackling SPSP, most of them being metaheuristic methods, since it is difficult to design exact methods for solving efficiently all instances of the problem. These approaches have been proposed according to the SPSP model that is considered and, of course, if additional models are proposed, there is an opportunity to continue designing appropriate algorithms.

This is specially true if the variables involved in the proposed models change their values, for example, when considering that new tasks can be added during the execution of the project, or when a task takes longer to finish than was originally estimated. These situations would make the search space larger for the remaining problem, which would require more efficient algorithms for a better exploration.

Many of the proposed approaches consider SPSP a single-objective problem and only a small number of papers consider it a multi-objective problem. When combining the two main objectives, project cost and duration, into a single objective function (a composite objective function), the result of the optimization is a single solution. If a single solution is presented to the customer, there will be little opportunity for negotiation. On the other hand, if both objectives are considered to be independent from each other, i. e. a multi-objective problem, the result of the optimization will be a set of solutions, each representing a different project plan scenario. If this set of plan scenarios, or a subset of them, is presented to the customer, there will be a bigger opportunity for negotiation. Thus we believe that another possible research area that could be further exploited is the multi-objective optimization of the SPSP.

5.4 Evaluation methods

The majority of the surveyed studies consider the benchmark set or the instance generator of Alba and Chicano [1] for evaluation. Despite the fact that these studies provide reasonable project and employee data, and that they test the robustness of the proposed approaches, it would be ideal to test future proposals with data from real projects.

Several methods for evaluating the proposed approaches were used, with the hit rate being the most commonly adopted. Hit rate considers the percentage of feasible solutions that were found by the algorithms when solving a

problem instance. Despite giving some idea of their performance, the hit rate does not necessarily reveal the general efficiency of the algorithms nor the practicality of the solutions.

For the multi-objective approaches, the performance of the algorithms was evaluated considering appropriate multi-objective metrics, such as the hypervolume, an indicator that allows convergence and diversity to be measured for a set of solutions. However, there are several other multi-objective indicators that have rarely been used or not used at all, like the generational distance and the inverted generational distance. Such indicators provide additional information regarding performance that could be considered to provide better supported conclusions.

5.5. Main results

Results from the surveyed studies concentrate on giving information regarding the performance of the proposed approaches with respect to the hit rate, hypervolume, and other related metrics. However, they have failed at analyzing and presenting their results from a more realistic perspective, which should consider aspects such as risk and practicality of the solutions.

In other words, with respect to solving the optimization problem, we could have a set of solutions that are feasible and represent the trade-offs between the cost and duration of a project. Some of these solutions, however, could have risks that should be considered. For example, some employee would be uncomfortable with particular tasks to which she/he has not been assigned before. Other example is that an employee maternity or paternity leave could overlap with the project because it had to be taken before it was planned. Another set of those solutions could be impractical because their duration is in conflict with the operational or financial regulations of the company.

6. CONCLUSIONS

In the last decade, several research papers that focus on the solution of the SPSP have been published. Furthermore, several surveys that present the state of the art in the solution of scheduling problems have also been published. These surveys, however, have focused primarily on the solution of RCPS. The lack of surveys focusing on the solution of the SPSP motivated us to propose this survey.

This survey provides two main contributions to the field of research about SPSP: as far as we are concerned, it is the first extensive survey regarding the SPSP and it provides general observations and suggestions that we believe

can be considered for future research.

In our survey, we reviewed 30 research papers that were selected by following the Systematic Mapping Study protocol [31]. The research goals for our study were presented in Table 1. We now describe, for each of these research goals, if they were achieved and how.

Our first research goal was to identify the project planning models that have been proposed to solve the SPSP. We indeed identified these models and our conclusion is that, although a number of models have been proposed, most of them are very similar to the one originally proposed by Alba and Chicano in [1]. Furthermore, we identified some limitations associated with these models, in particular, the fact that tasks are usually associated with multiple employees.

Our second research goal was to identify the aspects of software project planning that can be optimized. This goal was also reached and our conclusion is that the objectives that are considered more frequently are the minimization of the project duration and cost, although some additional objectives are sometimes considered, such as minimizing the number of defects in the project.

Our third research goal was to identify the kinds of algorithms or methods that are appropriate for solving the SPSP. We believe this objective was also reached and our conclusion is that the techniques that are most commonly used to solve the SPSP are the evolutionary algorithms (EAs) and, more specifically, genetic algorithms (GAs) and multi-objective evolutionary algorithms (MOEAs). We also observed that the main interest of these research papers was to analyze the performance of the algorithms and to evaluate the quality of the solutions.

From our point of view, we consider that there has been an important advancement to solve the SPSP and we believe that the future in this field of research looks promising. We also believe that it would be worthwhile to bring these research results to software development organizations.

7. ACKNOWLEDGEMENTS

We wish to acknowledge our university, Universidad Autónoma Metropolitana for supporting this research. Miguel Ángel also wishes to acknowledge the Consejo Nacional de Ciencia y Tecnología (CONACYT) for the scholarship he received to pursue his postgraduate studies.

8. REFERENCES

[1] E. Alba and J. F. Chicano. Software project management with GAs. *Information Sciences*, Volume 177, Issue 11, pages 2380-2401, 2007.

[2] F. Chicano, F. Luna, A. J. Nebro, and E. Alba. Using multi-objective metaheuristics to solve the software project scheduling problem. In *13th Annual Conference on Genetic and Evolutionary Computation*, pages 1915-1922. ACM, 2011.

[3] F. Chicano, A. Cervantes, F. Luna, and G. Recio. A Novel Multiobjective Formulation of the Robust Software Project Scheduling Problem. In *15th European Conference on the Applications of Evolutionary Computation*, pages 497-507. Springer, 2012.

[4] F. Luna, D. L. González-Álvarez, F. Chicano, and M. A. Vega-Rodríguez. On the scalability of multi-objective metaheuristics for the software scheduling problem. In *11th International Conference on Intelligent Systems Design and Applications,* pages 1110-1115. IEEE, 2012.

[5] F. Luna, D. L. González-Álvarez, F. Chicano, and M. A. Vega-Rodríguez. The software project scheduling problem: A scalability analysis of multi-objective metaheuristics. *Applied Soft Computing*, Volume 15, pages 136-148, 2014.

[6] C. K. Chang, M. J. Christensen, and T. Zhang. Genetic algorithms for project management. *Annals of Software Engineering*, Volume 11, Issue 1, pages 107-139, 2001.

[7] C. K. Chang, H. Y. Jiang, Y. Di, D. Zhu, and Y. Ge. Time-line based model for software project scheduling with genetic algorithms. *Information and Software Technology*, Volume 50, Issue 11, pages 1142-1154, 2008.

[8] A. García and M. Gómez. A Multi-objective Genetic Algorithm for the Software Project Scheduling Problem. In *13th Mexican International Conference on Artificial Intelligence,* pages 13-24. Springer, 2014.

[9] L. L. Minku, D. Sudholt, and X. Yao. Evolutionary algorithms for the project scheduling problem: Runtime analysis and improved design. In *14th Annual Conference on Genetic and Evolutionary Computation*, pages 1221-1228. ACM, 2012.

[10] L. L. Minku, D. Sudholt, and X. Yao. Improved evolutionary algorithm design for the project scheduling problem based on runtime analysis. *IEEE Transactions on Software Engineering*, Volume 40, Issue 1, pages 83-102, 2014.

[11] A. Ngo-The and G. Ruhe. Optimized resource allocation for software release planning. *IEEE Transactions on Software Engineering*, Volume 35, Issue 1, pages 109-123, 2009.

[12] J. Duggan, J. Byrne, and G. J. Lyons. A task allocation optimizer for software construction. *IEEE software*, Volume 21, Issue 3, pages 76-82, 2004.

[13] J. Xiao, X. T. Ao, and Y. Tang. Solving software project scheduling problems with ant colony optimization. *Computers & Operations Research*, Issue 40, Volume 1, pages 33-46, 2012.

[14] 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 *11th Annual Conference on Genetic and Evolutionary Computation*, pages

1673-1680. ACM, 2009.

[15] N. Jin and X. Yao. Heuristic optimization for software project management with impacts of team efficiency. In *2014 IEEE Congress on Evolutionary Computation*, pages 3016-3023. IEEE, 2012.

[16] B. Suri and P. Jajoria. Using ant colony optimization in software development project scheduling. In *2013 International Conference on Advances in Computing, Communications and Informatics*, pages 2101-2106. IEEE, 2013.

[17] M. Di Penta, M. Harman, and 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*, Volume 41, Issue 5, pages 495-519, 2011.

[18] G. Antoniol, M. Di Penta, and M. Harman. Search-based techniques for optimizing software project resource allocation. In *Genetic and Evolutionary Computation Conference*, pages 1425-1426. Springer, 2004.

[19] D. Rodríguez, M. Ruiz, J. C. Riquelme, and R. Harrison. Multiobjective simulation optimisation in software project management. In *13th Annual Conference on Genetic and Evolutionary Computation*, pages 1883-1890. ACM, 2011.

[20] T. Gonsalves and K. Itoh. Multi-objective optimization for software development projects. In *International MultiConference of Engineers and Computer Scientists*. 2010.

[21] G. Dupuy, N. Stark, and C. Salto. Algoritmo evolutivo para el problema de planificación en proyectos de desarrollo de software. In *XVIII Congreso Argentino de Ciencias de la Computación*. 2013.

[22] T. Hanne and S. Nickel. A multiobjective evolutionary algorithm for scheduling and inspection planning in software development projects. *European Journal of Operational Research*, Volume 167, Issue 3, pages 663-678, 2005.

[23] F. Wena and C. M. Lin. Multistage human resource allocation for software development by multiobjective genetic algorithm. *The Open Applied Mathematics Journal*, Volume 2, Issue 1, 2008.

[24] L. Özdamar and G. Ulusoy. A survey on the resource-constrained project scheduling problem. *IIE transactions*, Volume 27, Issue 5, pages 574-586, 1995.

[25] R. Kolisch and R. Padman. An integrated survey of deterministic project scheduling. *Omega*, Volume 29, Issue 3, pages 249-272, 2001.

[26] S. Hartmann and D. Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of operational research*, Volume 207, Issue 1, pages 1-14, 2010.

[27] S. Hartmann and R. Kolisch. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, Volume 127, Issue 2, pages 394-407, 2000.

[28] A. Fahmy. Optimization Algorithms in Project Scheduling. In O. Baskan, editor, *Optimization Algorithms - Methods and Applications*. InTech, 2016.

[29] C. Wu, X. Wang, and J. Lin. Optimizations in project scheduling: a state-of-art survey. In *Optimization and Control Methods in Industrial*

Engineering and Construction, pages 161-177. Springer, 2014.

[30] N. Patil, K. Sawant, P. Warade, and Y. Shinde. Survey paper for Software Project Scheduling And Staffing Problem. *International Journal of Advanced Research in Computer and Communication Engineering*, Volume 3, Issue 3, pages 5675 - 5677, 2014.

[31] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. *EBSE Technical Report EBSE-2007-01*, 2007.

[32] Y. Zhang, M. Harman, and A. Mansouri. The SBSE Repository: A repository and analysis of authors and research articles on Search Based Software Engineering, CREST Centre, UCL. Available at: http://crestweb.cs.ucl.ac.uk/resources/sbse_repository/. Last accessed: 04/23/2018.

[33] A. P. Engelbrecht. *Computational Intelligence: An Introduction*. John Wiley & Sons, 2007.

[34] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, Volume 6, Issue 2, pages 182-197, 2002.

[35] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm. Technical report *TIK-Report 103*, Swiss Federal Institute of Technology (ETH) Zurich, Switzerland, 2001.

[36] J. D. Knowles and D. W. Corne. Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary computation*, Volume 8, Issue 2, pages 149-172, 2000.

[37] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. Mocell: A cellular genetic algorithm for multiobjective optimization. *International Journal of Intelligent Systems*, Volume 24, Issue 7, pages 726-746, 2009.

[38] S. Kukkonen and J. Lampinen. GDE3: The third evolution step of generalized differential evolution. In. *2005 IEEE Congress on Evolutionary Computation*, pages 443-450. IEEE, 2005.

[39] K. Price and R Storn. Differential evolution: A simple evolution strategy for fast optimization. *Dr. Dobb's journal*, Volume 22, Issue 4, Pages 18-24, 1997.

[40] X. S. Yang. Firefly algorithms for multimodal optimization. In *International Symposium on Stochastic Algorithms,* pages 169-178. Springer, 2009.

[41] D. Karaboga. An idea based on honey bee swarm for numerical optimization.Technical report *tr06*, Erciyes University, Turkey, 2005.

[42] S. Hartmann. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics*, Volume 45, Issue 7, Pages 733-750, 1998.

[43] J. H. Holland. Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence. MIT Press, 1992.

[44] M. Wall. GALib: A C++ Library of Genetic Algorithm Components. Available at: http://lancet.mit.edu/ga/. Last accessed: 04/23/2018

[45] J. J. Durillo and A. J. Nebro. jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, Volume 42, Issue 10, pages 760-771, 2011.

[46] A. Drexl, R. Nissen, J. H. Patterson, and F. Salewski. ProGen/πx - An instance generator for resource-constrained project scheduling problems with partially renewable resources and further extensions. *European Journal of Operational Research*, Volume 125, Issue 1, pages 59-72, 2000.

[47] M. Ehrgott. *Multicriteria Optimization*. Springer, 2000.

[48] R. Pressman. *Software Engineering: A practitioner's approach*. Mc Graw Hill, 2005.

[49] C. Reeves. *Modern heuristic techniques*. John Wiley & Sons, 1996.

[50] X. S. Yang. *Engineering optimization: an introduction with metaheuristic applications*. John Wiley & Sons, 2010.

[51] C. Artigues, S. Demassey, and E. Neron. *Resource-constrained project scheduling: models, algorithms, extensions and applications*. John Wiley & Sons, 2013.

[52] B. Hardy-Vallee, The Cost of Bad Project Management. *Business Journal*, February, 2012. Available at: http://www.gallup.com/businessjournal/152429/cost-bad-project-management.aspx. Last accessed: 04/23/2018

[53] P. Jalote and B. Vishal. Optimal resource allocation for the quality control process. In *14th International Symposium on Software Reliability Engineering*, pages 26-33. IEEE, 2003.

[54] A. Ngo-The and Günther Ruhe, Optimized Resource Allocation for Software Release Planning, IEEE Transactions on Software Engineering, Volume 35, Issue 1, Pages 109-123, 2009.

[55] B. Crawford, R. Soto, F. Johnson, E. Monfroy, and F. Paredes. A Max–Min Ant System algorithm to solve the Software Project Scheduling Problem. *Expert Systems with Applications*, Volume 41, pages 6634-6645, 2014.

[56] B. Crawford, R. Soto, G. Astorga, and E. Olguín. An Alternative Solution to the Software Project Scheduling Problem. In *Artificial Intelligence Perspectives in Intelligent Systems*, pages. 501-510. Springer, 2016.

[57] B. Crawford, R. Soto, F. Johnson, C. Valencia, and F. Paredes. Firefly Algorithm to Solve a Project Scheduling Problem. In *Artificial Intelligence Perspectives in Intelligent Systems*, pages 449-458. Springer, 2016.

[58] X. Wu, P. Consoli, L. Minku, G. Ochoa, and X. Yao. An Evolutionary Hyper-heuristic for the Software Project Scheduling Problem. In *14th International Conference on Parallel Problem Solving from Nature*, pages. 37-47. Springer, 2016.

[59] J. Xiao, M.-L. Gao, and M.-M. Huang. Empirical Study of Multi-objective Ant Colony Optimization to Software Project Scheduling Problems. In *2015 Annual Conference on Genetic and Evolutionary Computation*, pages 759-766. ACM, 2015.

[60] X. Shen, L. L. Minku, R. Bahsoon, and X. Yao. Dynamic Software Project Scheduling through a Proactive-Rescheduling Method. *IEEE Transactions on Software Engineering*, Volume 42, Issue 7, pages 658-686. IEEE, 2016.

[61] X.-N. Shen, L. L. Minku, N. Marturic, Y.-N. Guod, and Y. Han. A Q-learning-based memetic algorithm for multi-objective dynamic software project scheduling. *Information Sciences*, Volume 428, pages 1-29, 2018.

[62] K. Deb, M. Mohan, and S. Mishra. Evaluating the ε-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal. *Evolutionary Computation*, Volume 13, Issue 4, pages 501-525, 2005.

[63] Ferrucci, F., Harman, M., and Sarro, F. Search-based software project management. In *Software Project Management in a Changing World*, pages 373-399. Springer, 2014.