@incollection{FerrucciSBPM,
year={2014},
isbn={978-3-642-55034-8},
booktitle={Software Project Management in a Changing World},
editor={Ruhe, Günther and Wohlin, Claes},
doi={10.1007/978-3-642-55035-5_15},
title={Search-Based Software Project Management},
publisher={Springer Berlin Heidelberg},
author={Ferrucci, Filomena and Harman, Mark and Sarro, Federica},
pages={373-399}
}

Search-Based Software Project Management

Filomena Ferrucci, Mark Harman, Federica Sarro

Abstract. Project management presents the manager with a complex set of related optimisation problems. Decisions made can more profoundly affect the outcome of a project than any other activity. In the chapter, we provide an overview of Search-Based Software Project Management, in which Search-Based Software Engineering (SBSE) is applied to problems in software project management. We show how SBSE has been used to attack the problems of staffing, scheduling, risk, and effort estimation. SBSE can help to solve the optimisation problems the manager faces, but it can also yield insight. SBSE therefore provides both decision making and decision support. We provide a comprehensive survey of Search-Based Software Project Management, and give directions for the development of this subfield of SBSE.

15.1 Introduction

Software Project Management includes several activities critical for the success of a project (e.g., cost estimation, project planning, quality management). These activities often involve finding a suitable balance between competing and potentially conflicting goals. For example, planning a project schedule requires to minimise the project duration and the project cost, and to maximise the product quality. Many of these problems are essentially optimization questions characterised by competing goals/constraints and with a bewilderingly large set of possible choices. So finding good solutions can be hard.

Search-Based Software Engineering seeks to reformulate software engineering problems as search-based optimisation problems and applies a variety of meta-heuristics based on local and global search to solve them (such as Hill Climbing, Tabu Search, and Genetic Algorithms). These meta-heuristics search for a suitable solution in a typically large input space guided by a fitness function that expresses the goals and leads the exploration into potentially promising areas of the search space.

Though the term Search-Based Software Engineering (SBSE) was coined by Harman and Jones in 2001 to cover the application of computational search and optimisation across the wide spectrum of software engineering activities (Harman and Jones 2001), there were already pockets of activity on several specific software engineering problems prior to the introduction of the term SBSE. One such topic was Search-Based Software Project Management, the topic of this chapter. In particular, there was work on search-based project scheduling and staffing by Chang (1994), Chang et al. (1994, 1998) and Chao et al. (1993), and on search-based software development effort estimation by Dolado (2001) and Shukla

(2000). Figure 15.1 shows the number of papers published on the use of search-based approaches for Software Project Management. We can note that the first work aiming at optimising project scheduling and staffing appeared in 1993, while in 2000 the SBSE community started investigating search-based approaches also for software development effort estimation.

This chapter provides a comprehensive review of techniques, results and trends published in relevant papers. We discuss the effectiveness of search-based approaches for supporting project manager in these activities and provide suggestions for future research directions.

The rest of the chapter is organized as follows. Section 15.2 reports on the main features of the most popular search-based techniques used in the context of search-based software project management. Section 15.3 introduces the problems of scheduling and staffing and building predictive models with special focus on software development effort estimation providing a description of search-based approaches proposed in the literature and the empirical studies carried out to assess their effectiveness. Future research directions are instead described in Section 15.4. Section 15.5 concludes the chapter.

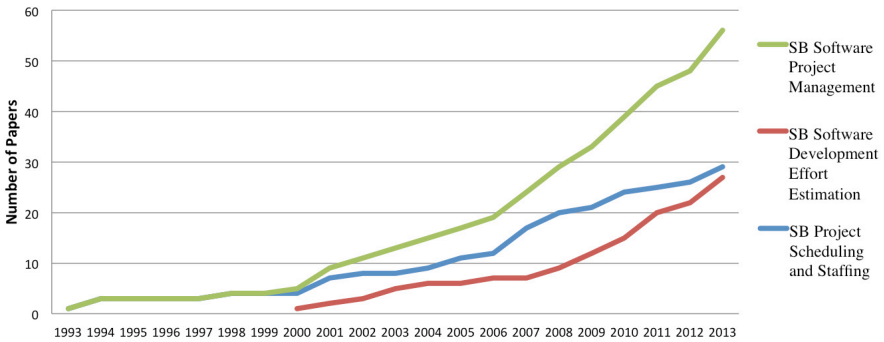

Figure 15.1: Number of Relevant Publications on the Use of Search-Based Approaches for Software Project Management from 1993 to 2013, source: (Zhang 2013).

15.2 Search-Based Software Engineering

Software Engineering, like other engineering disciplines, is concerned with optimization problems: we seek to build systems that are better, faster, cheaper, more reliable, flexible, scalable, responsive, adaptive, maintainable, and testable; the list of objectives for the software engineer is a long and diverse one, reflecting the breadth and diversity of applications to which software is put. The space of possible choices is enormous and the objectives many and varied. Search-Based Soft-

ware Engineering (SBSE) is an approach to software engineering in which search-based optimization algorithms are used to identify optimal or near optimal solutions and to get insight. Thus, in SBSE an SE problem (e.g., test case generation) is treated as a search or optimization problem whose goal is to find the most appropriate solution conforming to some adequacy criteria (e.g., maximizing the code coverage). Rather than constructing test cases, project schedules, requirements sets, designs, and other software engineering artefacts, SBSE simply searches for them.

The search space is the space of all possible candidate solutions. This is typically enormous, making it impossible to enumerate all solutions. Moving from conventional Software Engineering to SBSE basically requires choosing a representation of the problem and defining a suitable fitness function to determine how good a solution is. Typically, a software engineer will have a suitable representation for his/her problem, because one cannot do much engineering without a way to represent the problem in hand. Furthermore, many problems in Software Engineering have a rich and varied set of software metrics associated with them that naturally form good initial candidates for fitness functions (Harman and Clark 2004).

With these two ingredients it becomes possible to implement search-based optimisation algorithms. These algorithms use different approaches to locate optimal or near optimal solutions. However, they are all essentially a search through many possible candidate instances of the representation, guided by the fitness function, which allows the algorithm to compare candidate solutions according to their effectiveness at solving the problem in hand. Many techniques have been used including local search techniques, such as Hill Climbing (HC), and global techniques, such as Genetic Algorithm (GA) and Genetic Programming (GP).

Despite the local maximum problem, HC is a simple technique that is both easy to implement and surprisingly effective (Harman et al. 2002)(Mitchell 2002); these aspects make it a popular first choice among search-based techniques. GA belongs to the larger class of evolutionary algorithms (EAs) (Holland 1975), which loosely model evolutionary searches for fit individuals. GP (Koza 1992) is another form of evolutionary technique that has proved very useful in SBSE for project management.

A comprehensive review of the overall field of SBSE can be found in the work of Harman et al. (2012b). In that overall SBSE survey the reader can find a more detailed explanation of the algorithms used in SBSE. There is also a tutorial on SBSE (Harman et al. 2010), in which the reader can find a gentle introduction to the entire area. The SBSE survey (Harman et al. 2012b) and tutorial (Harman et al. 2010) cover the whole area of SBSE and, as a result, has little time and space available for each sub topic. The present survey focuses on the results, trends, techniques, and achievements in SBSE for project management. Though there have been many surveys on other sub areas of SBSE, including, Testing (McMinn

2004)(Yoo and Harman 2012), Design (Räihä 2010), and Requirements (Zhang et al. 2008), there has been no previous survey on SBSE for Project Management.

15.3 Search-Based Software Project Management

In this section, we provide an overview of the work on search-based Software Project Management, in which SBSE is applied to support project managers for time management (Chapter 1 Section 1.3.3.), cost management (Chapter 1 Section 1.3.4), quality management (Chapter 1 Section 1.3.5), human-resource management (Chapter 1 Section 1.3.6) and risk management (Chapter 1 Section 1.3.8).
The first application of SBSE to Software Project Management has been proposed for project scheduling and resource allocation. Figure 15.2 provides a generic schematic overview of SBSE approaches to project planning. Given in input information about work packages (e.g., cost, duration, dependencies) and staff skills, the search-based approaches search for an optimal work package ordering and staff allocation guided by a single or multi-objectives fitness function. A natural goal for a search-based approach to project management is to find project plans that minimise the completion time of the project. Another goal that has been taken into account is to minimise the risks associated with the development process (e.g., delays in the project completion time, or reduced budgets available).

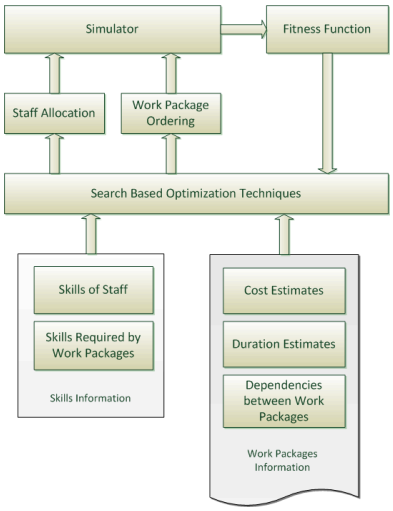

Figure 15.2: A Generic Search-Based Project Management Scheme (Harman et al. 2012b)

Figure 15.2 also highlights one of the important limitations of the approach: it relies on simulation of the likely course of the project, given the guiding project configuration parameters (effectively, the search space). Moreover, the formulation of search-based project management problem does not always model the reality of software projects (e.g., many papers do not have a realistic representation of skill and/or risk). However, there is evidence to suggest that this uptake in making the formulation of the search-based project management problem more realistic (see e.g., (Antoniol et al. 2004)(Antoniol et al. 2005)(Luna et al. 2012)) and oriented towards human aspects (see Chapter 4) is already taking place.

SBSE techniques also have a natural application in predictive modelling (Harman 2010). Software development effort estimation is one of the areas in which this search-based predictive modeling approach has been most widely investigated (Ferrucci et al. 2010c). In particular, the use of search-based approaches in this context has been twofold: they can be exploited to build effort estimation models or to enhance the use of other effort estimation techniques (Sarro, 2011).

In the first case the problem of building an estimation model is reformulated as an optimization problem where the search-based method builds many possible models - exploiting past projects data - and tries to identify the best one, i.e., the one providing the most accurate estimates.

In the second case, search-based methods can be exploited in combination with other estimation techniques to improve critical steps of their application (e.g., features subset selection or the identification of critical parameters) aiming to obtain better estimates.

Many empirical studies were carried out in this field showing that search-based techniques are not only as effective as widely used effort estimation methods (see e.g., (Ferrucci et al. 2010)) but also their use can significantly improve the accuracy of other data-driven effort estimation techniques (see e.g., (Corazza et al. 2013)). Moreover, there is evidence that the use of search-based approaches can help to yield insight into open problems, such as the choice of a reliable measure to compare different estimation models (see e.g., (Ferrucci et al. 2010b)(Lokan 2005)). Furthermore, search-based approaches only have been used to obtain exact prediction (i.e., one point estimate for a project); however, they can be exploited to investigate prediction uncertainty and risk of inaccurate prediction by means of using sensitivity analysis or multi-objective optimisation, as successfully done in other fields of SBSE (see e.g., (Harman et al. 2009)).

Figure 15.3 shows the key ideas developed so far for Search-Based Project Management. In Subsections 15.3.1-15.3.4 we discuss the studies that have been carried out on the use of search-based approaches for project planning and staffing, while in Subsection 15.3.5 we discuss the main studies on search-based effort estimation. Open challenges and future work are reported in Section 15.4.

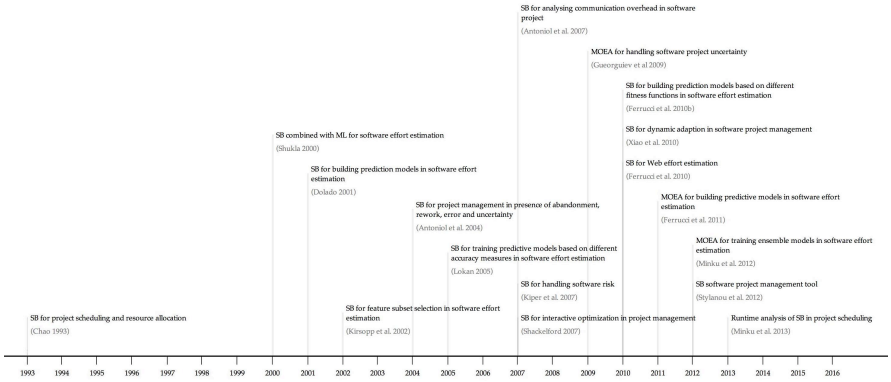

Figure 15.3: Key Ideas Developed for Search-Based Software Project Management

SBSE has also been used to build predictive models to support project managers in other estimation tasks, namely quality prediction and defect prediction. In particular, Azar (2010) considers approaches to improve predictive models of software quality using SBSE. Liu and Khoshgoftaar (2011) apply GP to quality prediction, presenting two case studies of the approach. This GP approach has been extended, refined and further explored in (Khoshgoftaar et al. 2003, 2007)(Liu and Khoshgoftaar 2001, 2003, 2004) (Khoshgoftaar and Liu 2007).

In all these works, GP-evolved predictors are used as the basis for decision support. Bouktif et al. (2002, 2004, 2006) exploit GA and SA for quality prediction for software projects. Other authors have used a combination of GA and GP techniques for estimation as a decision support tool for software managers. Jarillo et al. (2001) and Afzal et al. (2013) apply GA and GP for predicting the number of defects and estimating the reliability of the system. Others exploit GA to search for a suitable configuration of Support Vector Machines to be used for inter-release fault prediction (Di Martino et al. 2011)(Sarro et al. 2013).

15.3.1 Early Work on Search-Based Software Project Planning and Staffing

Chang *et al*. (Chang 1994)(Chang et al. 1994)(Chang et al. 1998)(Chang et al. 2001)(Chao et al. 1993) introduced the Software Project Management Net (SPM-Net) approach for project scheduling (Chapter 1 Section 1.3.3) and resource allocation (Chapter 1 Section 1.3.6). This was the first work on search-based Software Project Management in the literature. The work is evaluated on simulated data, constructed synthetically to mimic the properties of real software projects and to evaluate the properties of the algorithms used. One of the enduring problems re-

searchers concerned with project management face is the lack of available real world project data. It remains a common problem to this day.

At about the same time as the term SBSE was introduced to the mainstream software engineering community, Aguilar-Ruiz et al. (2001, 2002) were also experimenting with computational search as a means of managing and investigating software project management activities. The goal was to provide rules to the manager to help guide the process of project management. As with the work of Chang *et al,* a simulation of the project was used to evaluate the search.

This concept of a software project simulation has remained prevalent throughout the history of work on project management (Chapter 17). To evaluate a fitness for a proposed project plan, it is necessary to run a simulation of the course of the project in order to obtain an assessment of the fitness of the proposed plan. Of course, this raises additional issues as to the validity of the simulation. One of the advantages of SBSE approach, more generally, is that it can operate directly on the engineering material (i.e., the software) in question. This is an aspect of SBSE that is unique to the software engineering domain and not shared by any other application of computational search to other engineering disciplines (Harman 2010a). However, for search-based project management, the familiar issues that arise in computational search for other engineering disciplines arise here also for software engineering (Harman 2010). We have to be aware that our model of reality and our simulation of that model are both important players in the overall computation of fitness and, thereby, impact the results obtained; errors in the model or the simulation may feed through into poor quality solutions found by the search.

15.3.2 Minimising Software Project Completion Time

A natural step for a search-based approach to project management is to focus on techniques that find project plans that minimise the completion time of the project (Chapter 1 Section 1.3.3). Like all project managers, software project managers are concerned with timely product delivery. In highly competitive software engineering application domains, time to market can be a key determinant of the ultimate success of a product.

Antoniol et al. (2004, 2005) applied GAs, HC and SA to the problem of staff allocation to work packages with the aim of reducing project completion time.

At the same time, Alba and Chicano (2005, 2007) also applied search algorithms to software projects. They combined several different objectives for optimisation of project management into a single weighted sum and optimise for this weighted sum. The approach was evaluated on a set of problems generated by an instance generator. Subsequently, this work was extended to handle multiple objectives using a Pareto optimisation approach (Chicano et al. 2011).

The work of Antoniol et al. (2004, 2005) targeted a massive maintenance project, in which work packages were compressible by the allocation of additional staff. This principle of compressible work packages runs contrary to Brooks' famous law (Brooks 1975). That is, adding more staff to a late project simply makes the project even later. Following Brooks' law we may not, in general, assume that a work package of 2 person months will take one month to complete should we choose to allocate 2 people to it. In the most extreme case, the additional communication overheads may mean that the work package takes longer to complete with two people than with one.

However, in some cases it is realistic to assume that the duration of a work package can be derived by dividing the person months needed for the package by the number of engineers working on it. This can only be applied within reason, even where the linearity can be assumed to hold; a 2 person month work package may not be completed in under an hour by allocating one thousand engineers to it! However, linearity may apply for a reasonably useful range of values, where the tasks are highly mechanized or where they are specified in detail and prescriptive. Such work packages may be found in such massive maintenance tasks such as those studied in Antoniol et al. (2004, 2005). They may also be found in situations where software engineering activities are outsourced and therefore more highly specified for this reason.

Many authors have simplified their models of software projects by implicitly or explicitly assuming linearity (effectively denying Brooks' law). Where the linearity assumption cannot be justified and we assume that Brooks' law applies, we can still use SBSE; we simply require a richer model. We can also use SBSE to explore the impact of Brooks' law on the software project planning process. Antoniol et al. (2005) introduced an approach to investigating Brooks' law with different models of communication overhead to explore the influence of nonlinearity on project planning.

More intricate models may be required to adequately capture the true behaviour of the project once it commences. Another example of an important aspect of software engineering projects is the tendency for aspects of the project to be reworked. Software is so flexible, that it is often considered easy to reassign or reimplement a component. This can lead to headaches for the project manager, who would prefer, perhaps, to schedule his or her project on the basis of known completion times.

To make the formulation of the search-based project management problem more realistic, Antoniol et al. (2004) introduced models of the project management problem that account for re-working and abandonment of work packages. In this way we can enrich our models of the eventual software project process to cater for more real world assumptions. This may make the overall model more realistic and the simulation, thereby, more reliable.

Unfortunately, it also makes the model more complex and consequently it becomes harder to explain the outcomes to the users. Care is thus required that the model does not become such a Byzantine work of intricate beauty, that its findings become simultaneously impenetrable to the decision maker; this area of SBSE is primarily concerned with decision support, rather than decision making (Harman et al. 2012b). Insights that accrue to the decision maker rely critically on accessibility of explanations.

Much of the previous work on search-based project management (Aguilar-Ruiz et al. 2001)(Alba and Chicano 2005)(Alba and Chicano 2007)(Chang 1994)(Chang et al. 1994)(Chang et al. 1998)(Chang et al. 2001)(Chao et al. 1993)(Minku et al. 2012)(Minku et al. 2013) has used synthetic data. This can be achieved in a disciplined and controlled manner. For example, Alba and Chicano (2007) used a systematic instance generator to create synthetic software project data concerning work package estimated effort. This approach to the construction of synthetic data allows for experimental control of the evaluation under 'laboratory conditions'. Such experimental control has been argued to be an important aspect of SBSE that complements empirical analysis on real world case studies (Harman et al. 2012). Antoniol et al. (2005) applied their search-based algorithms to real world data from a large Y2K maintenance project, providing empirical evidence about search-based project management that complements (but does not replace the need for) the experimental data from other studies.

Many other approaches and formulations have been introduced for the software project management problem. For example, Alvarez-Valdes et al. (2006) applied Scatter Search to the problem of minimizing project duration. Hericko et al. (2008) used a gradient-based optimization method to optimize project team size while minimising project effort. Chen and Zhang (2013) used an Ant Colony Optimisation (ACO) approach. Kang et al. (2011) optimized the scheduling of human resource allocations by using a variant of SA, taking into account individual and team constraints based on the literature and interviews with experts in the industry, and by employing real data to validate their proposal. Rahman et al. (2010) also reported on an empirical analysis carried out exploiting real data on the use of both GA and a greedy approach that makes the locally optimal choice at each stage to assign developer to tasks and bug fixings activities. Different aspects of the management also focused on the allocation of staff (Barreto et al. 2008) (Kapur et al. 2008) and the provision of decision support (Cortellesa et al. 2008). Antoniol et al. (2011) provided a recent evaluation of search-based techniques for scheduling and staffing for software project management assessed on real world examples in the style of detailed empirical evaluations using non-synthetic data. This paper covers single and multiple objective formulations, catering for conflicting project objectives, schedule fragmentation, and developer expertise. Results are presented for HC, SA and GAs and applied to two real world software projects.

15.3.3 Risk Based Approaches

All software projects suffer from risk (Chapter 1 Section 1.3.8). Risks can be categorised as product risks and process risks. Risks to the product concern the possibility that there may be flaws in the product that make it less attractive to customers, while process risks concern the problems that may cause delays in the project completion time, or reduced budgets available forcing compromise.

Kiper et al. (2007) were concerned with the problem of technical product risks, seeking to find those verification and validation activities that could be deployed to reduce risks subject to budget. This work can be categorised as a product risk; it seeks to reduce the chance that the product will exhibit a risk of faults or other low quality.

Gueorguiev et al. (2009) concerned process risk, focusing on the chances that misestimating the effort required for a work package might lead to overruns which would adversely affect the completion time of the project. The effects of overruns are not immediately obvious, since they can affect the critical path, making previously less important work packages become more important for the overall project completion time.

Jiang et al. (2007) proposed an approach that extracts personnel risk information from historical data and integrates risk analysis into project scheduling performed with GA. A rescheduling mechanism is designed to detect and mitigate potential risks along with the software project development. However, the proposed approach has not been empirically validated.

Xiao et al. (2013) presented a search-based risk mitigation planning method based on GA for project portfolio management. Their results showed that with various risk mitigation actions and project objective settings, different plans can be effectively obtained, thus providing decision support for managers.

15.3.4 Overtime Planning

Effort estimation and planning of projects are hard problems that can be supported by decision support tools. Where these tools are inadequate or the project encounters unexpected ‘mission creep’, the consequences can be highly detrimental for the software engineers working on the project and the products they produce. Typically, the only remaining solution open to the project manager is to fall back on the allocation of overtime. However, unplanned overtime results in bad products, as has been repeatedly demonstrated in the literature (Akula et al. 2008) (Nishikitani et al. 2005). It also has harmful effects on the engineers forced into such punitive working practices (see e.g., Kleppa et al. 2008). The spectre of the ‘death march project’ (Yourdon 1997) hangs over many software engineering activities, largely as a result of the inability to plan for and manage the deployment of over-

time. More thorough overtime planning is not only beneficial to the software engineers who have to undertake the work (Beckers et al. 2008), there is also evidence that it produces better products when used in agile team (Mann and Maurer 2005).

Motivated by these observations Ferrucci et al. (2013) introduced a multi-objective formulation of the project overtime planning for software engineering management. The approach is able to balance trade-offs between project duration, overrun risk, and overtime resources for three different risk assessment models. It is applicable to standard software project plans, such as those constructed using the Critical Path Method, widely adopted by software engineers and implemented in many tools. To analyse the effectiveness of the approach they reported an empirical study on 6 real world software projects, ranging in size from a few person weeks to roughly four person years.

The experiments reveal that the approach was significantly better than standard multi-objective search in 76% of experiments and was significantly better than random search in 100% experiments. Moreover, it always significantly outperforms standard overtime planning strategies reported in the literature. Furthermore, the Pareto fronts obtained by the proposed approach can yield actionable insights into project planning trade-offs between risk, duration, and overtime using different risk assessment models. Software engineers can exploit this information when making decisions about software project overtime planning.

15.3.5 Using SB Approaches for Software Development Effort Estimation

Software development effort estimation concerns with the prediction of the effort needed to develop a software project. Such an effort is usually quantified as person-hours or person-months. Development effort is considered as one the major component of software costs and it is usually the most challenging to predict (Chapter 3.1). In the last few decades, several methods have been proposed to support project managers in estimating software development effort (Briand et al. 2002). In particular, data-driven methods exploit data from past projects to estimate the effort for a new project under development (typical methods are Linear Regression and Case-Based Reasoning). These data consist of information about some relevant factors (named cost drivers) and the effort actually spent to develop the projects. Usually a data-driven method tries to explain the relation between effort and cost drivers building an estimation model (equation) that is used to predict the effort for a new project. Also search-based methods have been used to build effort estimation models by formulating the problem as an optimisation problem that aims to identify the best model, i.e., the one providing the most accurate estimates. In the following, we highlight some key problems in the use of SBSE for effort estimation and how they have been addressed and assessed.

Dolado (2001) was the first to employ GP to automatically derive equations for estimating development effort and he observed a similar or better prediction than regression equations. Based on these encouraging results other investigations have been carried out comparing search-based approaches with other techniques proposed in the literature. Most of the studies are based on GP and only more recently other search-based techniques such as Tabu Search (Ferrucci et al. 2010, 2010a) and multi-objective evolutionary approaches (Ferrucci et al. 2011), (Minku and Yao 2012, 2013). have been employed

As for the setting of these techniques, usually a trial-and-error process has been employed carrying out a validation process with different settings and selecting the one providing the best results (Ferrucci et al. 2010c). This practice is time-consuming and it has to be repeated every time new data is used, thus limiting the adoption of search-based approaches by practitioners. A heuristic approach has been instead exploited in (Ferrucci et al. 2010b) and empirically analysed in (Sarro, 2013). The heuristic approach was originally suggested in (Doval et al. 1998) to set population size and number of generations of a GP for software clustering. In particular, given a project dataset containing V features, they set the number of iterations to $10V$ and stop the search after $1000V$ iterations or if the fitness value of the best solution does not change in the last $100V$ iterations. Thus, such heuristics adapts the search process to the size of the problem under investigation. Sarro (2013) extended the same heuristics to work also with Tabu Search (TS) (setting to V the length of Tabu List) and assessed its effectiveness by comparing it with respect to the use of five different configurations characterized by very small, small, medium, large, and very large number of solutions.

The results obtained by exploiting GP and TS on 7 public datasets highlighted that the considered heuristics is suitable to set both techniques since it provided comparable or superior prediction accuracy with respect to the ones obtained with the other configurations. Moreover, TS and GP configured by using the heuristics are much faster than the configurations obtained using other settings. This allowed saving time and computational resources without affecting the accuracy of the estimation models built with TS and GP, so the use of the heuristics has been revealed a cost-effective way to set these techniques on the considered datasets.

Another crucial design choice for search-based approaches is the definition of the fitness function that indicates how a solution is suitable for the problem under investigation driving the search towards optimal solutions. For the effort estimation problem the fitness function should be able to assess the accuracy of estimation models.

It is worth noting that several different accuracy measures have been proposed in the literature for assessing the effectiveness/accuracy of effort prediction models, such as the Mean of Absolute Error (MAE), the Mean of Squared Error (MSE) the Mean and Median of Magnitude of Relative Error (MMRE and MdMRE, re-

spectively), the Mean and Median of Magnitude of Estimate Relative Error (MERE and MdEMRE, respectively), the Prediction at level k (Pred(k)) (Conte et al. 1986)(Kitchenham et al. 2001). Usually they represent a cumulative measure of the error/residual, i.e., the difference between actual effort and predicted effort (e.g., MAE, MSE), or of the relative error with respect the actual (e.g., MMRE, MdMRE) or the estimated effort (e.g., MEMRE, MdEMRE); or a percentage of the cases where the considered error is less of a chosen threshold (e.g., Pred(25)).

Among them, the MMRE (Conte et al. 1986) represents the most widely used measure for assessing effort estimation proposals, thus it is not surprising that it has been also the most used as fitness function in the study employing search-based techniques. Nevertheless, MMRE reliability has been questioned by several researchers (e.g., (Kitchenham et al. 2001)(Shepperd and MacDonell 2012) and has been shown that it does not select the best model among competing ones. Presently, there does not exist a unique measure universally accepted as the best way to assess the estimation accuracy of effort models. On the other hand, each proposed measure focuses the attention on a specific aspect. As a matter of fact, Pred(25) measures how well an effort model performs, while MMRE measures poor performance; MMRE is more sensitive to overestimates and MEMRE to underestimates. Thus, it could be argued that the choice of the criterion for assessing predictions and establishing the best model can be a managerial issue. So, a project manager could prefer to use Pred(25) as the criterion for judging the quality of a model, while another might prefer to use another criterion, just for example MMRE to better control overestimates, or, to get a more reliable assessment, another could jointly employ several evaluation criteria covering different aspects of model performances (e.g., underestimating or overestimating, success or poor performance).

Based on this consideration search-based methods represent an opportunity due to their flexibility. Indeed, they allow the use as fitness function of any measure able to evaluate some properties of interest, thus allowing a project manager to select his/her preferred accuracy measure so that the search for the model is driven by such a criterion. Moreover, search-based techniques can take into account not only single evaluation criteria but also multiple ones, considering some algebraic expressions of basic measures (e.g., Pred(25)/MMRE) (Ferrucci et al. 2010b) or exploiting more sophisticated approaches based on multi-objective optimization (Ferrucci et al. 2011)(Minku and Yao 2012, 2013)(Sarro 2012a).

Different fitness functions have been employed in the studies carried out so far. They highlighted that such a choice can affect the performance of the obtained models: each fitness function is able to guide towards estimation models with better accuracy in terms of the selected criterion, but some of them can degrade a lot the other summary measures (Burgess and Lefley 2011)(Ferrucci et al.

2010b)(Lokan 2005)(Sarro 2013). Thus, project managers should be aware of this effect and should take care to select the right evaluation criterion as fitness function.

Another aspect that is important for project managers (both for trust on the solution proposed by an estimation technique and for improving the data collection process of current projects) is concerned with the transparency of the proposed solution. Search-based approaches produce transparent solutions because the prediction model is an algebraic expression that makes explicit any information about the contribution of each variable in the model (this is not always the case for other estimation techniques, for example neural networks).

Nevertheless, due to the variable length of the expression tree some proposed GP approaches (e.g., (Dolado 2001)(Burgess and Lefley 2001)(Lefley and Shepperd 2003)) produced not so clear expressions that need to be simplified. To improve the transparency of solutions, an evolutionary computation method, named Grammar Guided Genetic Programming (GGGP), was proposed by Shan et al. (2002) that exploited grammars to impose syntactical constraints and incorporate background knowledge. Another approach to simplify the transparency of solutions provided by GP was exploited in (Ferrucci et al. 2010b) based on the use of trees of fixed depth and crossover and mutation operators that preserved the syntactic structure.

As for the empirical studies all performed a hold-out validation on industrial datasets except for one that employed a dataset which contains academic projects. All the industrial datasets have been widely used in effort estimation studies; they come from a single company (e.g., Desharnais (Menzies et al. 2012)) or from multiple companies (e.g., Tukutuku (Ferrucci et al. 2010)) and are related to both software and Web projects. The criteria used to evaluate the accuracy of the obtained estimates are all based on summary measures; in particular MMRE and Pred(25). To make the comparison more reliable, some of them complemented the analysis with graphical tools (boxplot of residuals) and statistical tests.

As a general result of these studies we can conclude that search-based techniques behave consistently well obtaining estimation accuracy comparable or better than other widely used estimation techniques, such as the ones based on Manual StepWise Regression (MSWR) or Case-Based Reasoning (CBR). Recently, it has also been highlighted that TS outperformed GP since it turns out to be more efficient, while preserving the same accuracy (Sarro, 2013).

search-based methods can be exploited in combination with other estimation techniques to improve critical steps of their application (e.g., features subset selection or the identification of critical parameters) aiming to obtain better estimates.

Search-based methods have also been used in combination with other estimation techniques (see e.g., (Braga et al. 2008)(Faheem et al. 2008)), such as some Machine Learning (ML) techniques, aiming to obtain better estimates. Indeed, as

reported in several studies, ML approaches have the potential as techniques for software development effort estimation; nevertheless, their accuracy strongly depends on an accurate setting of these methods (see e.g., (Song et al. 2013)). As an example, to use CBR we have to choose among many similarity measures, number of analogies, and analogy adaption strategies (Mendes 2009), while to employ Support Vector Regression (SVR) we have to set several parameters depending also on the employed kernel function exploited to deal with non-linear problems (Cortes and Vapnik 1995).

There are no general guidelines on how to best configure these techniques since the appropriate setting often depends on the characteristics of the employed dataset. An examination of all possible values for configuration parameters of each technique is often not computationally affordable, as the search space is too large, also due to the interaction among parameters, which often cannot be separately optimized. Another aspect that can influence the accuracy of estimation techniques is the quality of input features, thus a feature subset selection (FSS) is usually recommended to select a subset of relevant features to be used in the model construction process.

To address both the abovementioned problems the use of search-based approaches has been proposed and investigated. In the following we first discuss four works conceived to select a suitable configuration for some estimation techniques, namely Neural Network, Case-Based Reasoning and Support Vector Regression, and then we discuss four works that exploited GAs to address the FSS problem.

Shukla (2000) was the first to propose a GA to configure Neural Network (NN) predictor in order to improve its estimation capability. In particular, GA had to find suitable weights for NN layer connections guided by a fitness function that minimizes MSE values. The empirical study based on two public datasets, i.e., COCOMO and Kemerer (Menzies et al. 2012), showed that GA+NN provided significantly better prediction than common used AI-oriented methods, such as CARTX and Quick Propagation trained NN. Similarly, Papatheocharous and Andreou (2009) enhanced the use of Artificial Neural Networks by using a Genetic Algorithm. Their results showed that using GA to evolve the network architectures (both input and internal hidden layers) reduced the Mean Relative Error (MRE) produced by the output results of each network.

Chiu and Huang (2007) applied GA to CBR to adjust the reused effort obtained by considering different similarity distances (i.e., Euclidean, Minkowski, and Manhattan distances) between pairs of software projects. The result obtained on two industrial datasets revealed that the proposed GA improved the estimations of CBR.

To automatically select suitable SVR settings Corazza et al. (2010, 2013) proposed and assessed another approach based on the use of TS. A total of 21 datasets were employed and several benchmarks were taken into account. The results re-

vealed that the combination of TS and SVR significantly outperformed all the other techniques showing that the proposed approach represents a suitable technique for software development effort estimation.

The first work that proposed the use of a search-based approach to address the FSS problem in the context of effort estimation was the one of Kirsopp et al. (2002). In particular, they employed Hill Climbing to select the best set of project features to be used with CBR. The combined approach was evaluated on an industrial dataset of 407 observations and the results showed that it performed better than Random Feature Selection and Forward Sequential Selection.

In (Li et al. 2009) a GA was proposed to simultaneously optimise the selection of the feature weights and projects to be used with CBR. The empirical results employing four datasets (two industrial (Menzies et al. 2012) and two artificial (Li et al. 2009)) showed that the use of GA+CBR provided significantly better estimations than CBR.

GA was also used to improve the accuracy of an effort estimation model built by combining social choice (voting rules were used to rank projects determining similar projects) and analogy-based approaches (Koch et al. 2009). In particular, GA was employed to find suitable weights to be associated to the project attributes. The results revealed that the proposed approach provided the best value for Pred(25), but worse MMRE values with respect to other techniques (LR, ANN, CART, COCOMO, and Grey Relational Analysis).

Huang et al. (2008) integrated a GA to Grey Relational Analysis to find the best fit of weights for each software effort driver. The experimental results showed comparable (COCOMO dataset) and better accuracy (Albrecht dataset) with respect to CBR, CART, and ANN.

15.4 Possible Directions for Future Work on Search-Based Project Management

In this section we outline several directions for future work in search-based project management, highlighting promising areas that emerge for the analysis of trends within this subfield of SBSE.

15.4.1 Interactive Optimisation

Several authors have suggested and adopted interactive evolution for design-based (Simons and Parmee 2008, 2013) and comprehension-based (Harman 2007a) software engineering tasks. However, only one attempt has been made to apply this technique, which can incorporate human expert knowledge directly into fit-

ness computation, in project management (Shackelford and Corne 2001). Since a software project management task is inherently human-centric it would be natural to explore the use of interactive evolution as a technique for ensuring that the project manager's expertise is accounted for in software project management (Shackelford 2007). The difficulty, as with all interactive evolution, lies in finding a way in which the manager can influence the computation of fitness without overburdening him or her with request for 'fitness assessment'.

It is also an open challenge as to how this judgment can be best incorporated into fitness. For example, the manager may be aware that certain individuals cannot work together, that certain work packages are more critical or that some dependence can be broken to make project more 'parallelisable'. This information cannot simply be requested from the manager at the outset of the optimisation process; there is too much of it and much of the information is implicit. Rather, we need to make the whole process of using search-based project management tools more interactive, so that the manager is able to 'realise' that they know something of importance at the specific point *i* the optimisation process at which it applies and to introduce this domain knowledge into the overall planning process in a natural and seamless way.

15.4.2 Dynamic Adaptive Optimisation

In order to maximise the value of interactive solutions to project management, we need dynamic adaptive approaches to SBSE (Harman et al. 2012). As an example, effective resource scheduling is complicated by different disruptions, such as requirements changes, bug fixing, or staff turnover, and dynamic resource scheduling can help to address such potentially disruptive events (Xiao et al. 2010)(Xiao et al. 2013). If solutions can be computed in real time and presented to the decision maker in an intuitive form, then the decision maker can ask on-the-fly 'what if' questions to help decide on key project commitments. In an ideal world, the decision maker would interact with the tool, exploring the possible implications of the decisions, with the optimisation continuing to provide updated best-so-far solutions as the decision maker interacts. This may require fundamentally different approaches to the algorithms and formulations that underlie search-based software project management.

15.4.3 Multi-Objective Optimisation

It has been argued that, in order to better match real world scenarios, SBSE should move from a single objective paradigm to a multi objective paradigm (Harman et al. 2007) (Harman et al. 2012b). Indeed, more recent SBSE work has followed a

more multi-objective style of approach, touching many application areas including requirements (Finkelstein et al. 2008, 2009)(Zhang et al. 2007), testing (Everson et al. 2006)(Harman 2011)(Harman et al. 2007b), refactoring (Harman et al. 2007) and also, not least, project management (Ferrucci et al. 2011, 2012, 2013)(Gueorguiev et al. 2009)(Minku and Yao 2012, 2013) (Rodriguez et al. 2011) (Stylanou and Andreou 2013). Most problems in software engineering involve multiple competing objectives and this is an observation most keenly felt in project management. Much of the future work on SBSE for project management is likely to focus on decision support in complex multi-objective problem spaces.

15.4.4 Co-Evolution

In Co-Evolutionary Computation, two or more populations of solutions evolve simultaneously with the fitness of each depending upon the current population of the other. Co-evolution can be either cooperative or competitive. In competitive evolution two (or more) populations of candidate solutions compete with each other for supremacy, the fitness of one depending on the other, such that improvements in one population tend to lead to lower fitness in the other. This is analogue to the well-known 'predator-prey' model of evolutionary biology and it has found application in SBSE work on testing (Adamopoulos et al. 2004) where predators are test cases and programs and their faults are the prey on which the test cases feed.

However, co-evolution need not always follow a predator-prey model; it can also be a co-operative, symbiotic process, just as often occurs in nature. In this co-operative co-evolutionary model, several populations, all of which have distinct fitness functions, nevertheless depend on one another, without necessarily being in conflict. This is a natural model for project management, in which we seek, for example, a mutually supportive allocation of staff to teams and, simultaneously, an allocation of teams to work packages. Ren et al. (2011) explore the co-operative co-evolutionary model of search-based project management. The close fit between project management objectives and co-evolutionary models makes this a natural choice and one that deserves more attention.

15.4.5 Software Project Benchmarking

One of the enduring problems researchers concerned with project management face is the lack of available real world project data. It remains a common problem to this day, especially for project planning and staffing. Indeed, despite there exist both publicly available (Menzies et al. 2012) and private (ISBG 2013) repositories of real project data for software project estimation, there are no available reposito-

ries containing information for project planning. The promising results achieved in search-based project management may lead to the false impression that these techniques are readily available for industrial application, whereas many papers just use synthetic data, do not model skill, or have a realistic representation of skill and/or risk. Previous work on software planning and staffing often relies on simulation of the likely course of the project rather than real data to assess the performance of the proposed approaches. Indeed, data sets of real world projects are scarce; effort data are seldom ever made public not to mention skill and other kind of employee's details (e.g., percentage of overhead, abilities to self-adapt to new project, management style, team leaders or soft skill). Despite researchers are making some effort to employ real data and to deal with a more human-centric problem (Chapter 4), to collect more real data on software projects and to make it publicly accessible still remains an open important challenge.

15.4.6 Confident Estimates

Obtaining exact time and effort estimates in software project management is impossible due not only to the inherently nature of estimates, but also due to incomplete, uncertain and/or noisy input used as the basis of the estimates. Rather than generating exact estimates it would be beneficial to introduce some level of uncertainty and measure its effect on the management process. As an example, sensitivity analysis can be a very useful means of determining the vulnerability of an estimate to particular assumptions and the level of confidence that can be placed in that estimate and the promising results obtained by using search-based approaches to assess software project uncertainty make us confident that these techniques can be a key instrument to support this kind of analysis (see e.g., (Harman et al. 2009)).

Elsewhere, Harman (2007), highlighted four future directions for the use of search-based approaches to obtain more confident predictive modelling that we want herein recall:

1. Incorporation of risk into predictive models;
2. More effective measurement of cost;
3. More reliable models (even at the expense of predictive power; trading median accuracy for reduced variance over iterated predictions);
4. Sensitivity analysis to determine which aspects are more important.

15.4.7 Decision Support Tools

Despite the promising results highlighted in the research papers, the use of search-based approaches for project management is still in the development/research stage. To the best of our knowledge only one tool for project management based on search-based approaches has been recently proposed (Stylanou et al. 2012). We believe that to transfer the most successful methods into practice we need to develop them as freely available decision support tools. Indeed, this will allow an extensive evaluation of the interface between the technical aspects on which the research has been focused and other related socio-technical issues for implementation and exploitation, such as user interface, ease of use, HCI, and decision support. Moreover, this will allow us also to get feedback from practitioners on the usefulness and cost/effectiveness of the proposed approaches.

15.5 Conclusions

SBSE has proved widely applicable across many fields of software engineering activities. In those software engineering activities closely associated with the software product, SBSE has tended to be used as a means of finding good solutions, guided by a fitness function. By contrast, its role in the earlier phases of the software development cycle, more associated with the establishment of plans and processes has tended to be subtler. In particular, for software project management, SBSE has tended to be used to provide decision support rather than to seek a single solution. This is a naturally exploratory and multi-objective scenario. As we have seen, search-based techniques have potential to support software project managers, with predictions, analysis of potential scenarios and the optimised configuration of process parameters. The review of trends in this chapter demonstrates that this is an active and growing field.

References

Adamopoulos K, Harman, M, Hierons RM (2004) How to Overcome the Equivalent Mutant Problem and Achieve Tailored Selective Mutation using Co-Evolution. In: Proceedings of the 6th Conference on Genetic and Evolutionary Computation, pp. 1338-1349

Afzal W, Torkar R, Feldt R, Gorschek T (2013) Prediction of faults-slip-through in large software projects: an empirical evaluation. Software Quality Journal, in press, doi:10.1007/s11219-013-9205-3

Aguilar-Ruiz JS, Ramos I, Riquelme JC, Toro M (2001) An Evolutionary Approach to Estimating Software Development Projects. Information and Software Technology, 43(14):875-882

Aguilar-Ruiz JS, Riquelme JC, Ramos I (2002) Natural Evolutionary Coding: An Application to Estimating Software Development Projects. In: Proceedings of the 4th Conference on Genetic and Evolutionary Computation

Akula B, Cusick J (2008) Impact of Overtime and Stress on Software Quality. In: Proceedings of the 4th International Symposium on Management, Engineering, and Informatics

Alba E, Chicano F (2005) Management of Software Projects with GAs. In: Proceedings of the 6th Metaheuristics International Conference, pp. 1152:1-6

Alba E, Chicano F (2007) Software Project Management with GAs. Information Sciences, 177(11):2380-2401

Alvarez-Valdes R, Crespo E, Tamarit JM, Villa F (2006) A Scatter Search Algorithm for Project Scheduling under Partially Renewable Resources. Journal of Heuristics 12(1-2):95-113

Antoniol G, Di Penta M, Harman M (2004) A Robust Search-based Approach to Project Management in the Presence of Abandonment, Rework, Error and Uncertainty. In: Proceedings of the 10th International Symposium on the Software Metrics, pp. 172-183

Antoniol G, Di Penta M, Harman M (2005) Search-based Techniques Applied to Optimization of Project Planning for a Massive Maintenance Project. In: Proceedings of the 21st IEEE International Conference on Software Maintenance, pp. 240-249

Antoniol G, Di Penta M, Harman M, Qureshi F (2007) The Effect of Communication Overhead on Software Maintenance Project Staffing: a Search–based Approach. In: Proceedings of the 23rd IEEE International Conference on Software Maintenance, pp. 315–324

Antoniol G, Di Penta M, Harman M (2011) The Use of Search-based Optimization Techniques to Schedule and Staff Software Projects: An Approach and an Empirical Study. Software Practice and Experience, 41(5):495-519

Azar D (2010) A Genetic Algorithm for Improving Accuracy of Software Quality Predictive Models: A Search-based Software Engineering Approach. International Journal of Computational Intelligence and Applications, 9(2):125-136

Barreto A, de O. Barros M, Werner CM (2008) Staffing a Software Project: a Constraint Satisfaction and Optimization-based Approach. Computers & Operations Research, 35(10):3073-3089

Beckers DG, van der Linden D, Smulders PG, Kompier MA, Taris TW, Geurts SA (2008) Voluntary or Involuntary? Control over Overtime and Rewards for Overtime in relation to Fatigue and Work Satisfaction. Work and Stress: An International Journal of Work, Health and Organisations, 22(1):33-50

Bouktif S, K´egl B, Sahraoui H (2002) Combining Software Quality Predictive Models: An Evolutionary Approach. In: Proceedings of the International Conference on Software Maintenance, pp. 385-392

Bouktif S, Azar D, Precup D, Sahraoui H, Kégl B (2004) Improving Rule Set Based Software Quality Prediction: A Genetic Algorithm based Approach. Journal of Object Technology, 3(4):227-241

Bouktif S, Sahraoui H, Antoniol G (2006) Simulated Annealing for Improving Software Quality Prediction. In: Proceedings of the 8th Conference on Genetic and Evolutionary Computation, pp. 1893-1900

Braga PL, Oliveira ALI, Meira SRL (2008) A GA-based Feature Selection and Parameters Optimization for Support Vector Regression Applied to Software Effort Estimation. In: Proceedings of the ACM Symposium on Applied Computing, pp. 1788-1792

Briand L, Wieczorek I (2002) Software Resource Estimation, Encyclopedia of Software Engineering, pp. 1160–1196

Brooks Jr FP (1975) The Mythical Man Month: Essays on Software Engineering. Addison-Wesley Publishing Company Reading, MA, USA

Burgess CJ, Lefley M (2001) Can Genetic Programming Improve Software Effort Estimation: a Comparative Evaluation. Information and Software Technology 43(14):863-873

Chang CK (1994) Changing Face of Software Engineering. IEEE Software 11(1):4-5

Chang CK, Chao C, Hsieh S-Y, Alsalqan Y (1994) SPMNet: a Formal Methodology for Software Management. In: Proceedings of the 18th International Computer Software and Applications Conference, pp. 57-57

Chang CK, Chao C, Nguyen TT, Christensen M (1998) Software Project Management Net: a new Methodology on Software Management. In: Proceedings of the 22nd International Computer Software and Applications Conference, pp. 534-539

Chang CK, Christensen MJ, Zhang T (2001) Genetic Algorithms for Project Management. Annals of Software Engineering, 11(1):107-139

Chao C, Komada J, Liu Q, Muteja M, Alsalqan Y, Chang C (1993) An Application of Genetic Algorithms to Software Project Management. In: Proceedings of the 9th International Advanced Science and Technology, pp. 247-252

Chen WN, Zhang J (2013) Ant Colony Optimization for Software Project Scheduling and Staffing with an Event-based Scheduler. IEEE Transactions on Software Engineering, 39(1):1-17

Chicano F, Luna F, Nebro AJ, Alba E (2011) Using Multi Objective Metaheuristics to Solve the Software Project Scheduling Problem. In: Proceedings of the 13th Conference on Genetic and Evolutionary Computation, pp. 1915–1922

Chiu NH, Huang S (2007) The Adjusted Analogy-based Software Effort Estimation based on Similarity Distances. Journal of Systems and Software 80(4):628–640

Conte D, Dunsmore H, Shen V (1986) Software Engineering Metrics and Models. The Benjamin/Cummings Publishing Company, Inc.

Corazza A, Di Martino S, Ferrucci F, Gravino C, Sarro F, Mendes E (2010) How effective is Tabu Search to Configure Support Vector Regression for Effort Estimation?. In: Proceedings of the 6th International Conference on Predictive Models in Software Engineering, pp. 4:1-10.

Corazza A, Di Martino S, Ferrucci F, Gravino C, Sarro F, Mendes E (2013) Using Tabu Search to Configure Support Vector Regression for Effort Estimation. Empirical Software Engineering 18(3):506-546

Cortellessa V, Marinelli F, Potena P (2008) An Optimization Framework for "Build-or-Buy" Decisions in Software Architecture. Computers & Operations Research 35(10):3090-3106

Cortes C, Vapnik V (1995) Support-Vector Networks. Machine Learning 20(3):273-297

Di Martino S, Ferrucci F, Gravino C, Sarro F (2011) A Genetic Algorithm to Configure Support Vector Machines for Predicting Fault-Prone Components. In: PROFES 2011. Lecture notes in computer science, vol. 6759.Springer, Heidelberg, p 247

Dolado JJ (2001) On the problem of the software cost function. Information and Software Technology 43(1): 61-72.

Doval D, Mancordis SB, Mitchell S (1998) Automatic Clustering of Software System using a Genetic Algorithm. In: Proceedings of the 9th International Workshop Software Technology and Engineering Practice, pp. 73-81

Everson RM, Fieldsend JE (2006) Multiobjective Optimization of Safety Related Systems: An Application to Short-Term Conflict Alert. IEEE Transactions on Evolutionary Computation, 10(2):187-198

Faheem A, Bouktif S, Serhani A, Khalil I (2008) Integrating Function Point Project Information for Improving the Accuracy of Effort Estimation. In: Proceedings of the International Conference on Advanced Engineering Computing and Applications in Sciences, pp. 193-19

Ferrucci F, Gravino C, Mendes E, Oliveto R, Sarro F (2010) Investigating Tabu Search for Web Effort Estimation. In: Proceedings of the 36th EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 350-357

Ferrucci F, Gravino C, Oliveto R, Sarro F (2010a) Estimating Software Development Effort Using Tabu Search. In: Proceedings of the 12th International Conference on Enterprise Information Systems, vol 1, pp. 236-24

Ferrucci F, Gravino C, Oliveto R, Sarro F (2010b) Genetic Programming for Effort Estimation: an Analysis of the Impact of Different Fitness Functions. In: Proceedings of the 2nd International Symposium on Search Based Software Engineering, pp. 89-98

Ferrucci F, Gravino C, Oliveto R, Sarro F (2010c) Using Evolutionary Based Approaches to Estimate Software Development Effort. Evolutionary Computation and Optimization Algorithms in Software Engineering: Applications and Techniques, M. Chis, IGI Global, p 13-28

Ferrucci F, Gravino C, Sarro F (2011) How Multi-Objective Genetic Programming is Effective for Software Development Effort Estimation?. In: Proceedings of the 3rd International Symposium on Search Based Software Engineering, Lecture notes in computer science vol. 6956. Springer, Heidelberg, p 274-275

Ferrucci F, Harman M, Ren J, Sarro F (2013) Not Going to Take This Anymore: Multi-objective Overtime Planning for Software Engineering Projects. In: Proceedings of the 35th IEEE International Conference on Software Engineering, pp.462-471

Finkelstein A, Harman M, Mansouri S. A, Ren J, Zhang Y (2008) "Fairness Analysis" in Requirements Assignments. In: Proceedings of the 16th IEEE International Requirements Engineering Conference, pp. 115-124

Finkelstein A, Harman M, Mansouri S. A, Ren J, Zhang Y (2009) A Search Based Approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. Requirements Engineering, 14(4):231-245

Gueorguiev S, Harman M, Antoniol G (2009) Software Project Planning for Robustness and Completion Time in the Presence of Uncertainty using Multi Objective Search-based Software Engineering. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1673-1680

Harman M (2007) The Current State and Future of Search-based Software Engineering. In: Proceedings of the Conference on Future of Software Engineering, pp. 342–357

Harman M (2007a) Search-based Software Engineering for Program Comprehension. In: Proceedings of the 15th IEEE International Conference on Program Comprehension, pp. 3-13

Harman M (2010) The Relationship between Search-based Software Engineering and Predictive Modelling. In: Proceedings of the 6th International Conference on Predictive Models in Software Engineering, pp. 1

Harman M (2010a) Why the Virtual Nature of Software Makes it Ideal for Search-based Optimization. In: Proceedings of the 13th International Conference on Fundamental Approaches to Software Engineering, pp. 1-12

Harman M (2011) Making the Case for MORTO: Multi Objective Regression Test Optimization. In: Proceedings of the 1st International Workshop on Regression Testing, pp. 111-114

Harman M, Clark JA (2004) Metrics Are Fitness Functions Too. In: Proceedings of the 10th International Symposium on Software Metrics, pp. 58-69

Harman M, Jones BF (2001) Search-based Software Engineering. Information and Software Technology, 43(14):833-839

Harman M, Krinke J, Ren J, Yoo S (2009) Search-Based Data Sensitivity Analysis Applied to Requirement Engineering. In: Proceedings of the 11th Conference on Genetic and Evolutionary Computation, pp. 1681-1688

Harman M, McMinn P, Teixeira de Souza J, Yoo S (2010) Search-Based Software Engineering: Techniques, Taxonomy, Tutorial. LASER Summer School 2010 pp. 1-59

Harman M, Tratt L (2007) Pareto Optimal Search-based Refactoring at the Design Level. In: Proceedings of the 9th Conference on Genetic and Evolutionary Computation, pp. 1106–1113

Harman M, Lakhotia K, McMinn P (2007b) A Multi-Objective Approach to Search-based Test Data Generation. In: Proceedings of the 9th Conference on Genetic and Evolutionary Computation, pp. 1098-1105

Harman M, Burke E, Clark J. A, Yao X (2012) Dynamic Adaptive Search-based Software Engineering. In: Proceedings of the 6th IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 1-8

Harman M, Hierons R, Proctor M (2002) A new Representation and Crossover Operator for Search-based Optimization of Software Modularization. In: Proceedings of the 4th Conference on Genetic and Evolutionary Computation, pp. 1351–1358

Harman M, Mansouri A, Zhang Y (2012b) Search-based Software Engineering: Trends, Techniques and Applications. ACM Computing Surveys 45(1):11-75

Hericko M, Zivkovic A, Rozman I (2008) An Approach to Optimizing Software Development Team Size. Information Processing Letters, 108(3):101-106

Holland J (1975) Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI

Huang SJ, Chiu NH, Chen LW (2008) Integration of the Grey Relational Analysis with Genetic Algorithm for Software Effort Estimation. European Journal of Operational Research 188(3):898-909

ISBSG (2013) data repository, available at www.isbsg.org

Jarillo G, Succi G, Pedrycz W, Reformat M (2011) Analysis of Software Engineering Data using Computational Intelligence Techniques. In: Proceedings of the 7th International Conference on Object Oriented Information Systems, pp. 133–142

Jiang H, Chang C.K, Xia J, Cheng S (2007) A History-Based Automatic Scheduling Model for Personnel Risk Management. In: Proceedings of the 31st Computer Software and Application Conference, pp. 361-364

Kang D, Jung J, Bae DH (2011) Constraint-based Human Resource Allocation in Software Projects. Software: Practice and Experience 41(5): 551–577

Kapur P, Ngo-The A, Ruhe G, Smith A (2008) Optimized Staffing for Product Releases and its Application at Chartwell Technology. Journal of Software Maintenance and Evolution: Research and Practice 20(5): 365-386

Khoshgoftaar TM, Liu Y (2007) A Multi-Objective Software Quality Classification Model Using Genetic Programming. IEEE Transactions On Reliability 56(2):237-245

Khoshgoftaar TM, Liu Y, Seliya N (2003) Genetic Programming-based Decision Trees for Software Quality Classification. In: Proceedings of the 15th International Conference on Tools with Artificial Intelligence, pp. 374–383

Kiper JD, Feather MS, Richardson J (2007) Optimizing the V&V Process for Critical Systems. In: Proceedings of the 9th Conference on Genetic and Evolutionary Computation, pp. 1139-1139

Kirsopp C, Shepperd MJ, Hart J (2002) Search Heuristics, Case-based Reasoning And Software Project Effort Prediction. In Proceedings of the Genetic and Evolutionary Computation Conference, pp.1367-1374

Kitchenham B, Pickard LM, MacDonell SG, Shepperd MJ (2001) What Accuracy Statistics Really Measure. IEEE Proceedings Software 148(3):81–85

Kleppa E, Sanne B, Tell GS (2008) Working overtime is associated with anxiety and depression: The Hordaland health study. Journal of Occupational and Environmental Medicine 50(6):658–666

Koch S, Mitlöhner, J (2009) Software Project Effort Estimation with Voting Rules. Decision Support Systems 46(4):895 – 901

Koza JR (1992) Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA

Lefley M, Shepperd MJ (2003) Using Genetic Programming to Improve Software Effort Estimation based on General Data Sets. In: Proceedings of the 5th Genetic and Evolutionary Computation Conference, pp. 2477–2487

Li YF, Xie M, Goh TN (2009) A Study of Project Selection and Feature Weighting for Analogy based Software Cost Estimation. Journal of Systems and Software, 82(2):241-252

Liu Y, Khoshgoftaar TM (2001) Genetic Programming Model for Software Quality Classification. In: Proceedings of the 6th IEEE International Symposium on High-Assurance Systems Engineering: Special Topic: Impact of Networking, pp. 127–136

Liu Y, Khoshgoftaar T (2004) Reducing Overfitting in Genetic Programming Models for Software Quality Classification. In: Proceedings of the 8th IEEE International Symposium on High Assurance Systems Engineering, pp. 56–65

Liu Y, Khoshgoftaar TM (2003) Building Decision Tree Software Quality Classification Models Us ing Genetic Programming. In: Proceedings of the 5th Genetic and Evolutionary Computation Conference, pp. 1808–1809

Lokan C (2005) What Should You Optimize When Building an Estimation Model? In: Proceedings of the 11th IEEE International Symposium on Metrics, pp. 34

Luna F, Chicano JF, Alba E (2012) Robust Solutions for the Software Project Scheduling Problem: a Preliminary Analysis. Int. Journal Metaheuristic 2(1):56-79

Mann C, Maurer F (2005) A Case Study on the Impact of Scrum on Overtime and Customer Satisfaction. In Agile Development Conference, pp. 70–79

McMinn P (2004) Search-based software test data generation: a survey. Softw. Test, Verif. Reliab. 14(2):105-156

Mendes E (2009) Web Cost Estimation and Productivity Benchmarking. Software Engineering, Lecture notes in computer science, vol 5413. Springer, Heidelberg, p194-222

Menzies, T, Caglayan B, Kocaguneli E, Krall J, Peters F, Turhan B (2012) The PROMISE Repository of empirical software engineering data http://promisedata.googlecode.com

Mitchell B. S, Mancoridis S (2002) Using Heuristic Search Techniques to Extract Design Abstractions from Source Code. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1375–1382

Minku LL, Sudholt D, Yao X (2013) Improved Evolutionary Algorithm Design for the Project Scheduling Problem Based on Runtime Analysis IEEE Transactions on Software Engineering. doi 10.1109/TSE.2013.52

Minku LL, Sudholt D, Yao X (2013) Evolutionary Algorithms for the Project Scheduling Problem: Runtime Analysis and Improved Design. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp.1221-1228

Minku LL, Yao X (2013) An Analysis of Multi-objective Evolutionary Algorithms for Training Ensemble Models Based on Different Performance Measures in Software Effort Estimation. In: Proceedings of the 9th International Conference on Predictive Models in Software Engineering, pp. 8:1-8:10

Minku LL, Yao X (2012) Software Effort Estimation as a Multi-objective Learning Problem. ACM Transactions on Software Engineering and Methodology, 22(4):35:1-35:32

Nishikitani M, Nakao M, Karita K, Nomura K, Yano E (2005) Influence of Overtime Work, Sleep Duration, and Perceived Job Characteristics on the Physical and Mental Status of Software Engineers. Industrial Health 43(4):623-629

Papatheocharous E, Andreou SA (2009) Hybrid Computational Models for Software Cost Prediction: An Approach Using Artificial Neural Networks and Genetic Algorithms. Lecture notes in business information processing vol. 19. Springer, Heidelberg, p 87-100

Rahman MM, Sohan SM, Maurer F, Ruhe G (2010) Evaluation of Optimized Staffing for Feature Development and Bug Fixing. In: Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, p 42

Räihä O (2010) A survey on search-based software design. Computer Science Review 4(4):203-249

Ren J, Harman M, Di Penta M (2011) Cooperative Co-evolutionary Optimization on Software Project Staff Assignments and Job Scheduling. In: Proceedings of the 3rd International Symposium on Search Based Software Engineering, pp. 127-141

Rodriguez D, Ruiz M, Riquelme JC, Harrison R (2011) Multiobjective Simulation Optimisation in Software Project Management. In: Proceedings of the 13th Conference on Genetic and Evolutionary Computation, pp. 1883-1890

Sarro F (2011) Search-Based Approaches for Software Development Effort Estimation. In: Proceedings of the 12th International Conference on Product-Focused Software Development and Process Improvement (Doctoral Symposium), pp. 38-43

Sarro F (2013) Search-Based Approaches for Software Development Effort Estimation. PhD Thesis. University of Salerno, Italy

Sarro F, Di Martino S, Ferrucci F, Gravino C (2012) A Further Analysis on the Use of Genetic Algorithm to Configure Support Vector Machines for Inter-release Fault Prediction. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing, pp. 1215-1220

Sarro F, Ferrucci F, Gravino C (2012a) Single and Multi Objective Genetic Programming for Software Development Effort Estimation. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing, pp. 1221-1226

Shackelford MRN, Corne DW (2001) Collaborative Evolutionary Multi-Project Resource Scheduling. In: Proceedings of the Congress on Evolutionary Computation, vol 2, pp. 1131-1138

Shackelford MRN (2007) Implementation Issues for an Interactive Evolutionary Computation System. In: Proceedings of the 2007 Genetic and Evolutionary Computation Conference, pp. 2933-2936

Shan Y, McKay R. I, Lokan CJ, Essam DL (2002) Software Project Effort Estimation using Genetic Programming. In: Proceedings of International Conference on Communications Circuits and Systems, pp. 1108-1112

Shepperd MJ, MacDonell SJ (2012) Evaluating Prediction Systems in Software Project Estimation. Information & Software Technology 54(8):820-827

Shukla KK (2000) Neuro-genetic Prediction of Software Development Effort. Information and Software Technology 42(10):701-713

Simons CL, Parmee IC (2008) User-centered, Evolutionary Search in Conceptual Software Design. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 869–876

Simons CL, Parmee IC (2012) Elegant Object-oriented Software Design via Interactive Evolutionary Computation. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 42(6):1797-1805

Song L, Minku LL, Yao X (2013) The Impact of Parameter Tuning on Software Effort Estimation Using Learning Machines. In: Proceedings of the 9th International Conference on Predictive Models in Software Engineering

Stylianou C, Andreou AS (2013) A multi-objective genetic algorithm for intelligent software project scheduling and team staffing. Intelligent Decision Technologies: An International Journal, 7(1):59-80

Stylianou C, Gerasimou S, Andreou AS (2012) A Novel Prototype Tool for Intelligent Software Project Scheduling and Staffing Enhanced with Personality Factors. In: Proceedings of the 24th International Conference on Tools with Artificial Intelligence pp 277-284

Xiao J, Osterweil LJ, Wang Q, Li M (2010) Dynamic Resource Scheduling in Disruption-Prone Software Development Environments. In: Proceedings of the 13th Conference on Fundamental Approaches to Software Engineering, pp. 107-122

Xiao J, Osterweil LJ, Wang Q, Li M (2010) Disruption-Driven Resource Rescheduling in Software Development Processes, In New Modeling Concepts for Today's Software

Processes. Lecture notes in computer science, vol 6195. Springer, Heidelberg, p 234-247

Xiao J, Osterweil LJ, Chen J, Wang Q, Li M (2013) Search-Based risk mitigation planning in project portfolio management. In: Proceedings of the 2013 International Conference on Software and System Process, pp. 146-155

Yoo S, Harman M (2012) Regression testing minimization, selection and prioritization: a survey. Softw. Test, Verif. Reliab. 22(2):67-120

Yourdon E (1997) Death March The Complete Software Developer's Guide to Surviving 'Mission Impossible' Projects. Prentice-Hall

Zhang Y (2013), SBSE paper repository, crestweb.cs.ucl.ac.uk/resources/sbse_repository/

Zhang Y, Harman M, Mansouri SA (2007) The Multi-Objective Next Release Problem. In: Proceedings of the 9th Conference on Genetic and Evolutionary Computation, pp. 1129-1137

Zhang Y, Finkelstein A, Harman M (2008) Search-Based Requirements Optimisation: Existing Work and Challenges. In Proceedings of the 14th international conference on Requirements Engineering: Foundation for Software Quality, pp. 88-94