Empir Software Eng (2017) 22:175–234
DOI 10.1007/s10664-015-9418-0

Zen-ReqOptimizer: a search-based approach for requirements assignment optimization

Yan Li[1] · Tao Yue[2,3] · Shaukat Ali[2] · Li Zhang[1]

Published online: 5 March 2016

Abstract At early phases of a product development lifecycle of large scale Cyber-Physical Systems (CPSs), a large number of requirements need to be assigned to stakeholders from different organizations or departments of the same organization for review, clarification and checking their conformance to standards and regulations. These requirements have various characteristics such as extents of importance to the organization, complexity, and dependencies between each other, thereby requiring different effort (workload) to review and clarify. While working with our industrial partners in the domain of CPSs, we discovered an optimization problem, where an optimal solution is required for assigning requirements to various stakeholders by maximizing their familiarity to assigned requirements, meanwhile balancing the overall workload of each stakeholder. In this direction, we propose a fitness function that takes into account all the above-mentioned factors to guide a search algorithm to find an optimal solution. As a pilot experiment, we first investigated four commonly applied search algorithms (i.e., GA, (1+1) EA, AVM, RS) together with the proposed fitness function and results show that (1+1) EA performs significantly better than the other algorithms. Since our optimization problem is multi-objective, we further empirically evaluated

Communicated by: Daniel Amyot

✉ Tao Yue
tao@simula.no

Yan Li
YanLL@buaa.edu.cn

Shaukat Ali
shaukat@simula.no

Li Zhang
lily@buaa.edu.cn

[1] Department of Computer Science and Engineering, Beihang University, Beijing, China

[2] Simula Research Laboratory, Fornebu, Norway

[3] Department of Informatics, University of Oslo, Oslo, Norway

the performance of the fitness function with six multi-objective search algorithms (CellDE, MOCell, NSGA-II, PAES, SMPSO, SPEA2) together with (1+1) EA (the best in the pilot study) and RS (as the baseline) in terms of finding an optimal solution using an real-world case study and 120 artificial problems of varying complexity. Results show that both for the real-world case study and the artificial problems (1+1) EA achieved the best performance for each single objective and NSGA-II achieved the best performance for the overall fitness. NSGA-II has the ability to solve a wide range of problems without having their performance degraded significantly and (1+1) EA is not fit for problems with less than 250 requirements Therefore we recommend that, if a project manager is interested in a particular objective then (1+1) EA should be used; otherwise, NSGA-II should be applied to obtain optimal solutions when putting the overall fitness as the first priority.

Keywords Search based software engineering · Requirements assignment · Optimization and empirical evaluation

1 Introduction

Requirements play a key role in the development of large-scale Cyber-Physical Systems (CPSs), which integrate computation, networking, and physical processes. The quality of these requirements has a significant impact on almost every single activity of a product development lifecycle. Due to their inherent complexity, such systems often have a large number of requirements that have to be reviewed, clarified, and checked for their conformance to standards and regulations at a very early stage of the lifecycle. More specifically, these activities include: 1) identifying and clarifying ambiguous and imprecise requirements; 2) identifying and adding missing requirements; 3) identifying and resolving conflict requirements; 4) identifying infeasible or high risk requirements; 5) identifying all relevant and affected documents such as design and test documents; 6) assessing the impact of changes in requirements; and 7) checking the conformance of requirements against various industrial standards and government and other regulations. These activities are often performed by a large number of stakeholders[1] from different organization or different departments of the same organization, having different job functions such as system, software, hydraulics, mechanical and networking engineers. This leads to the situation that they often have various extents of familiarity to requirements; therefore assigning requirements that they are most familiar with is a very important factor to assure the overall quality of a product and the productivity of its development. An overview diagram is provided in Fig. 1 to clarify the relationships among requirements (structured and organized in Requirements Repository), stakeholders and project manager. In the figure, we also present the inputs and output of the Zen-ReqOptimizer tool introduced in this paper.

As shown in Fig. 1, each requirement can be characterized from different aspects such as its complexity, importance, and dependencies with other requirements. Requirements that are more important and complex, and highly dependent on several other requirements

[1] In our context, we define stakeholders as engineers in different organizations who have responsibilities to review and clarify requirements and check their conformance to various standards. Such stakeholders include, for example, domain experts of a specific discipline such as software engineering and requirements engineers who are responsible to manage requirement artifacts.

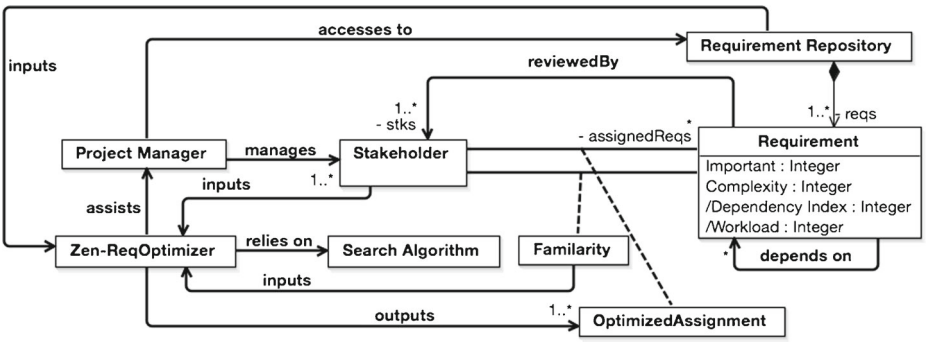

Fig. 1 Overview of the requirements assignment context with Zen-ReqOptimizer (in UML class diagram)

typically require more effort to review, check and clarify. Therefore, the workload for performing these activities on such requirements is higher than that on less important, complex and loosely depended ones. When a project manager manually assigns a set of requirements to a stakeholder, it is important to take their characteristics into account for the purpose of balancing the workload of each stakeholder and best utilizing their experience (measured as Familiarity) of working on specific requirements. Manually doing so does not scale, in a real industrial context, for even an experienced project manager, when assigning a large number of requirements to many stakeholders (implying a large search/solution space), and the manually derived assignment solution might not be optimal as the process is purely based on the experience of the project manager.

For any manual requirement assignment process, a project manager has to incorporate multiple objectives and explore to which extent a solution can be good according to which objective(s). This is a non-trivial multi-objective optimization problem and therefore such a manual process is either biased towards particular preferences of the project manager or more or less random. A scalable and automated requirement assignment solution is expected in this context and it has to reveal the inherent tensions and tradeoffs between the different objectives. In this way, Search Based Software Engineering (SBSE) can be used not only to provide an automated solution, but also to provide insight by exploring the space of possible solutions and their relationships. As reported in the SBSE review (Harman et al. 2009), SBSE has addressed several software engineering problems spanning from requirements, testing to reengineering of a typical software development lifecycle. Particularly for requirements, SBSE has been applied for various optimization problems such as requirements selection (Bagnall et al. 2001), prioritization (Baker et al. 2006), and assignment (Finkelstein et al. 2009), with different objectives such as maximizing customers'/stakeholders' satisfaction, maximizing benefits/value and minimizing cost. None of these work shares the same optimization objective as ours, but promisingly they demonstrated the performance of applying SBSE approaches for solving various optimization problems in requirements engineering.

In this paper, we propose a search-based approach, named as Zen-ReqOptimizer, to address the above described requirements assignment problem in a scalable and automated manner. Zen-ReqOptimizer, as shown in Fig. 1, takes characterized requirements (managed in a repository), stakeholders, and their familiarity to the requirements as input and produces optimized requirements assignment solutions. In this paper, we particularly focus on the selection of search algorithms (via rigorous empirical studies) that perform the best

in terms of solving our optimization problem. First we define three objectives: *ASSIGN* (describing to which extent all requirements are assigned to stakeholders), *FAM* (representing the overall familiarity of stakeholders to requirements allocated to them) and *OWL* (indicating the overall difference of workloads of the stakeholders). We combined these three objectives (with equal weights) into a fitness function and assessed the fitness function for the above-mentioned optimization problem, as a pilot study. We evaluated our fitness function in conjunction with the commonly used search algorithms, i.e., Genetic Algorithms (GAs), (1+1) Evolutionary Algorithm (EA) and Alternating Variable Method (AVM). Random Search (RS) was used as the baseline to evaluate the performance of these algorithms. In a real-world case study with 135 requirements and 10 stakeholders, results show that all the three search algorithms significantly outperformed RS, which indicates the usefulness of applying search algorithms. When comparing the performance of the three search algorithms, results show that (1+1) EA is better than AVM and AVM is better than GA. We carefully designed 120 artificial problems inspired from our industrial application with the number of requirements ranging from 50 to 1000 and the number of stakeholders ranging from 5 to 30 to test the scalability of the algorithms. Results show that the performance of (1+1) EA is not impacted with the increase in the number of requirements and stakeholders and has consistently significantly better performance than the other algorithms.

Since the results of the pilot study indicate that search algorithms are fit for our requirements assignment problem, based on the defined three objectives (i.e., *ASSIGN*, *FAM* and *OWL*),we then conduct an extensive empirical evaluation with a real-world case study (287 requirements and 10 stakeholders) and 120 artificial problems with six multi-objective search algorithms, i.e., Cellular Algorithm hybridized with Differential Evolution (CellDE), Multi-Objective Cellular (MOCell), Fast Nondominated Sorting Genetic Algorithms (NSGA-II), Pareto-Archived Evolution Strategy (PAES), Speed-constrained Multi-objective Particle Swarm Optimization algorithm (SMPSO) and Strength Pareto Evolutionary Algorithms (SPEA2), together with (1+1) EA (the best one in the pilot study) and RS (the baseline). Results of the empirical evaluation show that: (1) for the real-world case study, all the selected search algorithms are cost-effective for solving the requirements assignment problem and (1+1) EA obtains the best performance for each individual objective (i.e., *ASSIGN*, *FAM* or *OWL*) while NSGA-II achieves the best performance for the overall fitness (*FS*); (2) for the 120 artificial problems, the results are consistent with the real-world case study, i.e., all the selected algorithms achieve better performance than RS; (1+1) EA significantly outperforms all the other search algorithms for over 65 problems for *ASSIGN*, 120 problems for *FAM*, 120 problems for *OWL*; NSGA-II significantly outperforms all the other search algorithms for over 98 problems for *FS*. When looking at the overall fitness values for the artificial problems, NSGA-II is able to solve a wide range of problems without having its performance degraded significantly.

The previous work of the proposed approach of this paper was published in Yue and Ali (2014), to compare with which we 1) refined the fitness function, 2) re-conducted a pilot study to assess the performance of three single-objective search algorithms in terms of solving the optimization problem, 3) conducted a full scale empirical study on applying the six multi-objective search algorithms to solve the optimization problem, 4) and discussed the tool support in detail.

The rest of the paper is organized as follows. In Section 2, we present a formal representation of our optimization problem followed by the fitness function used by all the algorithms. Section 3 presents the empirical evaluation, followed by the detailed results of the pilot study and the empirical study in Section 4. We provide the overall discussion of the

Zen-ReqOptimizer methodology in Section 5. Section 6 presents threats to validity. Related work is presented in Sections 7 and 8 concludes the paper.

2 Problem, Objectives and Fitness Function

Recall that our goal is to maximize the assignment of requirements to stakeholders according to their familiarity to the requirements, while balancing the workload of each stakeholder. In Section 2.1 we present a formal representation of the search/optimization problem, followed by the representation of objectives (Section 2.2), and the fitness function (Section 2.3).

2.1 Definitions

A system *SYS* has a set of requirements $Req = \{R_1, R_2, R_3 \dots R_{n_R}\}$, where n_R is the total number of requirements. *SYS* has a set of stakeholders $Stk = \{S_1, S_2, S_3 \dots S_{n_{St}}\}$, where n_{St} is the total number of stakeholders of *SYS*. Each stakeholder S_j from *Stk* has a certain familiarity value FM_{ji} for a requirement R_i from *Req*, which is an integer value ranging from FM_{min} to FM_{max}. A stakeholder S_j may have no familiarity with a requirement R_i and FM_{ji} is assigned a value 0. For *SYS*, we have a set of such values defined as the matrix below:

$$FM = \begin{Bmatrix} FM_{11} & FM_{12} & \cdots & FM_{1i} & \cdots & FM_{1n_R} \\ FM_{21} & FM_{22} & \cdots & FM_{2i} & \cdots & FM_{2n_R} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ FM_{j1} & FM_{j2} & \cdots & FM_{ji} & \cdots & FM_{jn_R} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ FM_{n_{St}1} & FM_{n_{St}2} & \cdots & FM_{n_{St}i} & \cdots & FM_{n_{St}n_R} \end{Bmatrix}$$

In total, we have $n_R * n_{St}$ *FM* values for the problem. According to the manual assignment process in our industrial context (Sections 3.1.4 and 3.1.5), FM_{min} is 0 and FM_{max} is 9. In Section 5, we also discuss how we collected these data for our case study.

Each requirement R_i in *Req* has a complexity, which is an integer value ranging from CM_{min} to CM_{max}. The complexity of each requirement can be calculated based on applications. In Section 5, we discuss how the complexity is calculated in our industrial application. Same as *FM*, in our industrial application there are 10 levels of complexity where CM_{min} is 0 and CM_{max} is 9. For *SYS*, we have $CM = \{CM_1, CM_2, \dots CM_{nR}\}$, and in total we have n_R *CM* values, where each R_i from *Req* has a corresponding CM_i value in *CM*.

Each requirement R_i has a dependency index, which is calculated as the number of requirements in *Req*, on which R_i is dependent. The value of the dependency index for each requirement ranges from DP_{min} to DP_{max}. In fact DP_{min} is 0, which means that the requirement depends on no other requirements. DP_{max} is $n_R - 1$, which represents that the requirement depends on all the other requirements in *SYS*. For *SYS*, we have $DP = \{DP_1, DP_2, \dots, DP_{n_R}\}$, and each R_i has a corresponding value DP_i in *DP*.

In addition, each requirement R_i has an importance value, which is an integer value ranging from IM_{min} to IM_{max}. How we calculated the importance for each requirement of the real-world case study is discussed in Section 5, and similarly to *FM* and *CM*, it ranges from 0 to 9 in our industrial application. For *SYS*, we have $IM = \{IM_1, IM_2, \dots, IM_{nR}\}$, and each requirement R_i *has a corresponding value* IM_i *in IM*.

$S = \{ps_1, ps_2, ps_3 \ldots, ps_{n_s}\}$ is the entire search space of potential solutions, where n_s is the total number of potential solutions. Each solution has n_R elements, each of which represents a particular requirement and the value of each of which represents a specific stakeholder to which the particular requirement is assigned. For example, $ps_k = \{m_{R_1}, m_{R_2}, \ldots, m_{R_{n_R}}\}$ represents a requirements assignment solution. For example, for the i_{th} requirement, $m_{R_i} = j$ denotes that the requirement is assigned to the jth stakeholder; $m_{R_i} = 0$ means that the requirement is not assigned to any stakeholder. This problem representation says that one requirement can be assigned to only one stakeholder and one stakeholder can receive 0, 1 or more requirements to review. Notice that n_s will be exponentially exploded with the growth of n_{St} (number of stakeholders) and n_R (number of requirements), which can be calculated as $ns = {n_{St}}^{n_R}$. Manually finding a best solution out of this large search space is practically impossible and thus justifies the use of search algorithms to find an optimal solution satisfying a set of measures criteria described in the following section.

2.2 Objectives

Recall that assigning requirements to different stakeholders should maximize their familiarity to the assigned requirements and at the same time balance the overall workload for each stakeholder. Before defining a fitness function that is used to assess the quality of a solution, we first provide the definitions of objective: *ASSIGN* (representing the extent of assigning all the requirements to stakeholders), *FAM* (denoting the overall familiarity of the stakeholders to requirements allocated to them) and *OWL* (representing the overall differences of workloads of the stakeholders), as shown in formulas (1–3):

$$ASSIGN = \frac{\sum_{i=1}^{n_{St}} n_{AR_i}}{n_R} \tag{1}$$

$$FAM = \frac{\sum_{i=1}^{n_{St}} \sum_{j=1}^{n_{AR_i}} \frac{FM_{ij} - FM_{min}}{(FM_{max} - FM_{min})}}{\sum_{k=1}^{n_{St}} n_{AR_k}} \tag{2}$$

$$OWL = \frac{\sum_{j=1}^{n_{St}-1} \sum_{k=j+1}^{n_{St}} abs\left(WL_j - WL_k\right)}{n_{St} * (n_{St} - 1)} \tag{3}$$

As shown in Formula (1), $ASSIGN$ calculates a value that tells how far we are from assigning all the requirements to the stakeholders. n_{AR_i} returns the number of requirements assigned to the i_{th} stakeholder in *Stk* and we sum up all the requirements. The total number obtained is then divided by n_R, i.e., the total number of requirements, to represent the ratio of assigned requirements to all the requirements. We aim to maximize *ASSIGN*, meaning that all the requirements ideally have to be assigned to all the stakeholders.

FAM is about computing the overall familiarity of stakeholders to assigned requirements, as shown in Formula (2). Part of the formula is $\sum_{i=1}^{n_{St}} \sum_{j=1}^{n_{AR_i}} \frac{FM_{ij} - FM_{min}}{(FM_{max} - FM_{min})}$, which computes the overall familiarity of each stakeholder to all the requirements assigned to her/him. Dividing it with $\sum_{k=1}^{n_{St}} n_{AR_k}$ gives an average familiarity value for all the stakeholders. A higher value calculated by the formula means higher familiarity and higher fitness. In other words, we want to maximize *FAM*.

As shown in Formula (3),*OWL* computes the overall difference of the workload between each pair of stakeholders. Recall that our objective is to balance the workload across the stakeholders. WL_i computes the workload for all the requirements assigned to the i_{th} stakeholder based on their complexity, dependency index, and importance: $WL_i =$

$\frac{\sum_{j=1}^{n_{AR_i}} \left(\left(\frac{CM_j - CM_{min}}{(CM_{max} - CM_{min})} + \frac{DP_j - DP_{min}}{(DP_{max} - DP_{min})} + \frac{IM_j - IM_{min}}{(IM_{max} - IM_{min})} \right) / 3 \right)}{n_{AR_i}}$. Notice that the requirement characteristics for calculating workload (i.e., complexity CM values, dependency index DP values, and importance IM values) are given equal weights in our formula. *OWL* then uses these computed workload values for each stakeholder and computes how balanced is workload among the stakeholders using the formula $\sum_{j=1}^{nSt-1} \sum_{k=j+1}^{nSt} abs\left(WL_j - WL_k\right)$. For example, if stakeholder A has workload of 0.5 and stakeholder B has workload of 0.4, then the difference of the workloads between stakeholders A and B is 0.1 (i.e., $0.5 - 0.4$). We aim to minimize *OWL*, implying that the overall workload is balanced. Finally we divide the nominator with $n_{St} * (n_{St} - 1)$, as shown in Formula (3), to obtain the average workload of each stakeholder.

2.3 Fitness Function

Based on the measures and objectives defined above we introduce the single-objective fitness function and the multi-objective fitness function. Recall that an optimal solution is required for assigning as many requirements as possible to the stakeholders by maximizing their familiarity to the assigned requirements and at the same time balancing the overall workload of each stakeholder.

For the single-objective search algorithms, we evaluate the fitness of each solution using Formula (4). It always computes a value ranging from 0 to 1, where lower the value better the fitness.

$$FS = ((1 - ASSIGN) + (1 - FAM) + OWL)/3 \tag{4}$$

The optimization problem can then be reformulated as: searching for a solution ps_k from ${n_{St}}^{n_R}$ solutions, which should

$$\text{Minimize } (FS).$$

We can also formulate our optimization problem as a multi-objective optimization problem as: searching for a solution ps_k from ${n_{St}}^{n_R}$ solutions, which should

$$\text{Maximize } (ASSIGN),$$
$$\text{Maximize } (FAM), \text{ and}$$
$$\text{Minimize } (OWL).$$

3 Empirical Evaluation

This section provides an overview of the experiment design, execution and statistical tests used in the analyses.

3.1 Experiment Design

There are two stages in our experiments: the pilot study with three single objective search algorithms (i.e., AVM, GA, (1+1) EA), and the full-scale empirical evaluation with six multi-objective search algorithms (CellDE, MOCell, NSGA-II, PAES, SMPSO and SPEA2). Both studies used RS as the baseline for comparison and additional the full-scale study also included (1+1) EA for comparison, as it achieved the best performance in the pilot study. Both studies were conducted with one real-world case study and 120 artificial problems. The real-world case study was also used to evaluate the integrated solution by working with the tool, Zen-ReqOptimizer we developed (Section 5). The artificial problems

were inspired by the real-world case study for the purpose of evaluating whether the fitness function defined in Section 2.3 can address our problems even with a large number of requirements and stakeholders. Results of experiments are presented in Section 4.

In the rest of the section, we present the research questions (Section 3.1.1) and introduction of the search algorithms (Section 3.1.2). We then introduce the pilot study and empirical evaluation in detail in Sections 3.1.4 and 3.1.5. The defined dependent and independent variable for supporting the statistical analyses are presented in Section 3.1.6.

3.1.1 Research Questions

In these experiments, we address the following research question:

RQ1: Are the selected search algorithms effective to solve our optimization problem, to compare with RS?

RQ2: Among these selected search algorithms, which one fares best in solving our optimization problem?

RQ3: How does the increment of the number of requirements impact the performance of the search algorithms?

RQ4: How does the increment of the number of stakeholders impact the performance of the search algorithms?

RQ5: How does the combined increment of the number of stakeholders and requirements impact the performance of the search algorithms?

RQ6: Among these selected search algorithms, which one has the best performance in terms of time for solving our optimization problem?

3.1.2 Introduction to the Selected Algorithms

In this section, we firstly provide a brief description of the ten algorithms that were selected for our experiments (section Description of the Selected Algorithms) and then introduce selection criteria of search algorithms and parameter settings of these algorithms (Section 3.1.3).

Description of the Selected Algorithms In this section, we provide a brief description of the ten algorithms that were selected for our experiments. These algorithms are classified into four working mechanisms to guide the search (Brownlee 2012), which are: Evolutionary Algorithms (EAs), Swarm Algorithms, Hybrid Algorithms and Stochastic Algorithms (Table 1). Notice that one of our aims is to evaluate a representative set of the existing search algorithms rather than limiting us to the commonly used algorithms (e.g., NSGA-II and SPEA2). To achieve this goal, we chose ten search algorithms systematically based on the categories classified in Brownlee (2012).

Overview of the Search Algorithms Generally speaking, search algorithms mimic natural phenomenon, such as, evolution process or behaviors of birds, e.g., bird flocking to search optimal solutions (Brownlee 2012). Search algorithms rely on a fitness function (objective function) to guide the search and find an optimal solution.

For GAs, individuals (also called as chromosomes) refer to potential solutions of the optimization problem and each one composed of a set of genes representing units for the

Table 1 Classification of the selected search algorithms (Brownlee 2012)

Mechanism			Selected Search Algorithms
Evolutionary Algorithms (EAs)	GAs	Weight-Based GA	WBGA
		Sorting-Based GA	NSGA-II
		Cellular-Based GA	MOCell
	Weight-Based EA		(1+1) EA
	Strength Pareto EA		SPEA2
	Evolution Strategies		PAES
Swarm Algorithm	Particle Swarm Optimization		SMPSO
Hybrid Algorithm	Cellular genetic algorithm + differential evolution		CellDE
Stochastic Algorithm	Local Search		AVM
	Random Search		RS

solution, e.g., assignment sequence for the requirements in our case. The set of individuals handled by GAs is represented as a population. Similarly, for swarm algorithms (Section 3.1.2), the potential solutions are named as particles and the set of the solutions handled by algorithms refers to swarm.

After the definition of fitness function, the search algorithms apply various operators (e.g., selection) to produce the best individuals until the terminating condition is satisfied (e.g., number of generations in GAs). Three operators are usually adapted by various search algorithms, namely selection, crossover (also called recombination) and mutation (Brownlee 2012). More specifically, 1) the selection operator selects solutions with the best fitness value determined with the fitness function. Notice that the selection operator can be applied to all the search algorithms; 2) the crossover operator is mainly applied in various GAs that selects two individuals (parents) and exchange their genes and produce offspring; and 3) the mutation operator is also mostly adapted in GAs that changes the properties of genes, e.g., using mutation rate (i.e., probability of changing the properties of genes).

Evolutionary Algorithms (EAs) EAs are inspired by biological evolution, which was first proposed by Darwin to describe how natural selection and mutation occurs on various genes of species (Brownlee 2012). Since EAs are well known in search-based software engineering (Harman et al. 2009), we choose six of them, which are further classified into GAs, Weight-Based EA, Strength Pareto EA and Evolution Strategies. We describe each of them in detail below.

Genetic Algorithms can be used to solve multi-objective problem by assigning a particular weight to each objective function and converting the multi-objective problem to a single objective problem, e.g., using a scalar objective function (Konak et al. 2006). These algorithms are then called weight-based GAs.

NSGA-II implements the Pareto dominance theory and produces a set of non-dominated solutions for multiple objectives (Deb et al. 2002). First, the population is sorted into several non-dominated fronts based on a ranking algorithm. Second, individual solutions are selected from these non-dominated fronts by calculating the crowd distance to measure distances between the individual solutions and the rest of the solutions in the population (Konak et al. 2006). If two individual solutions are in the same non-dominated front, the solution with a higher value of crowd distance will be selected.

Multi-objective Cellular (MOCell) relies on the cellular model of GAs (cGAs) and assumes that an individual only interacts with its neighbors (Nebro et al. 2007). Moreover, MOCell stores a set of obtained non-dominated individual solutions in an external archive.

After each generation, MOCell replaces a fixed number of solutions chosen randomly from the population by selecting the same number of solutions from the archive until termination conditions are met. Such replacement only takes place when newly generated solutions from the population are worse than ones in the archive.

(1+1) Evolutionary Algorithm (EA) (Droste et al. 2002) is simpler than GA, but it can be more effective in some cases (e.g., Arcuri 2013). The population size is one and the individual is encoded as a string of bits. Then, a bitwise mutation operator is used for exploring the search space. (1+1) EA is used as a representative of weight-based GAs in this paper.

Improved Strength Pareto Evolutionary Algorithm (SPEA2) also implements the Pareto dominance theory (Zitzler et al. 2001). Each individual's fitness value is calculated by adding the strength raw fitness based on objective functions and density estimation. Density estimation computes the distance between a solution and its nearest neighbors with the ultimate aim of maximizing the diversity of solutions. SPEA2 first stores a number of best solutions in an archive by applying various operators such as selection, crossover and mutation. SPEA2 then combines solutions from the archive and non-dominated ones creates a new population. If combined non-dominated solutions are more than the maximum size of the population, the solution with the minimum distance to any other solution is selected by applying the truncation operator to calculate distances to its neighborhood.

Pareto Archived Evolution Strategy (PAES) implements the evolution theory (Knowles and Corne 2000). It uses dynamic crossover, mutation operators, to maximize the suitability of the selected solutions. PAES also stores non-dominated solutions in an archive and newly generated solutions replace the ones in the archive in case they have better fitness.

Swarm Algorithms Particle Swarm Optimization (PSO) is a biological metaheuristic inspired by the social behavior of animals such as bird flocking (Brownlee 2012). The Speed-constrained Multi-objective Particle Swarm Optimization (SMPSO) algorithm implements the PSO theory (Knowles and Corne 2000; Nebro et al. 2009). Similar to NSGA-II, SMPSO selects best solutions by computing crowding distance and also stores selected individual solutions in an archive. SMPSO applies the mutation operator to accelerate the speed of convergence and adapts the *velocity constriction* mechanism to avoid the explosion of swarms. According to Knowles and Corne (2000) and Nebro et al. (2009), the performance of SMPSO is considered as the best among all PSO algorithms.

Hybrid Algorithms Differential Evolution (DE) is a type of EA that produces solutions using the recombination, mutation, and selection operators (Brownlee 2012). DE computes the weighted difference between two randomly selected solutions and integrates obtained parts into a third solution. CellDE is one such example of hybrid algorithms that uses MOCell as a search engine and replaces the typical selection, crossover and mutation operators for GAs with the recombination mechanism of DE (Durillo et al. 2008).

Stochastic Algorithms Random Search (RS) is a stochastic algorithm and is commonly used as a baseline for evaluation (Brownlee 2012). RS randomly samples the search space of solutions to obtain solutions. New solutions are independent of the previous ones.

Alternating Variable Method (AVM) is a local search algorithm proposed by Korel (1990), which is used as a represenetative of local search algorithms.

3.1.3 Selection Criteria of the Search Algorithms and Parameter Settings

In the pilot study, we compared four search algorithms: AVM, GA, (1+1) EA and RS. AVM is used as a representative for local search algorithms. GA is selected since it commonly used search algorithm in SBSE (Ali et al. 2010). For GA, population size was set to 100, crossover rate was set to 0.75, and rank selection was used with 1.5 bias. Standard one-point crossover was implemented and each variable was mutated with equal probability, i.e., *1/number of variables*. Different settings of search algorithms may lead to different sets of solutions, however as reported in the work by Arcuri and Fraser (2011) standards settings of search algorithms still provide good solutions. (1+1) EA was chosen because in several of earlier works it demonstrated better results than GA (Arcuri 2013; Saaty 1987; Yue and Ali 2014). RS is used as the baseline algorithm with which all the algorithms are compared (Ali et al. 2010).

In the full-scale empirical study, we compared other six multi-objective algorithms (CellDE, MOCell, NSGA-II, PAES, SMPSO and SPEA2) with the best algorithm in the pilot study, i.e., (1+1) EA. RS is still used as a baseline for assessing the difficulty of the problems. For the multi-objective algorithms, we also used the default suggested parameters settings from jMetal (Durillo and Nebro 2011).

3.1.4 Pilot Study

In the pilot study, we evaluated three single objective search algorithms with a real-world case study and a set of artificial problems to test their performance in terms of addressing our optimization problem.

Real-world Case Study The real-world case study is a large-scale CPS of the Energy domain, whose software controls and monitors the operation of electrical and mechanical instruments. The system is composed of up to hundreds of control modules and thousands of instruments, connected via communication networks. Requirements review, clarification and checking their conformance to standards and regulations are performed at the Tender and Front-End Engineering Design (FEED) phases. During the Tender phase, various domain experts are involved to prepare tender documents. FEED is a process for the conceptual development of a product.

To evaluate the selected search algorithms, we selected 135 requirements of a high-level system document and identified 10 stakeholders who are responsible to conduct activities during the tender and FEED phases. These ten stakeholders have different responsibilities (job functions) in the organization as shown in Table 2 and have various years of working experience in the organization. Therefore, one can expect that they have various extents of familiarity to the requirements to be assigned. In our context, there is a requirement repository managing all the requirements and a large portion of the requirements in the repository can be reused across projects. Therefore as long as stakeholders' familiarity to these requirements is established, the familiarity values can be reused. In Section 5.3 we will discuss the ways to get the values of familiarity if users using our method have no values of stakeholders' familiarity to requirements.

Requirements are managed using a requirements management tool. Each requirement is assigned a unique ID. Dependencies among themselves and with other documents (e.g., test artifacts, design documents) are maintained in the library. Therefore, it was not difficult to collect the data of requirements dependencies for running the experiment. It is worth

Table 2 Stakeholders' characteristics

Stakeholder responsible for	# of stakeholders
Control module software	3
Control module electronics	1
System hydraulics and mechanics	2
Configuring control module software	1
Overall project responsible	2
Quality control	1
Total	10

noticing that such information can be easily obtained automatically via the requirement repository.

These requirements are classified in different ways. First, they are divided into categories according to their relevance to the key architectural components (e.g., control modules, instruments). Second, they are classified according to commonly seen requirements engineering categories such as functional requirements, non-functional requirements (NFRs) and safety requirements (one type of non-functional requirements). The third category is internal or external. External requirements are more important than internal requirements as any misunderstanding about external requirements will be very expensive to fix. When collecting the *Importance* value of each requirement, safety requirements and external requirements are considered more important than others.

We analyzed a control system requirements document for one project and selected 135 requirements for the experiment in this part. The characteristics of these requirements are presented in Table 3. The complexity of these requirements is measured as the number of natural language sentences contained in each requirement. We understand that we might have simplified the measurement and in the full empirical study we change the measurement method.

Artificial Problems In addition, to evaluate whether the fitness function defined in Section 2.3 really addresses our optimization problem even with large numbers of stakeholders and requirements, we carefully defined 120 artificial problems. We created artificial problems with the increasing number of requirements and stakeholders. For the number of requirements, we used a range of 50 to 1000 with an increment of 50, whereas for the number of stakeholders we used a range from 5 to 30 with an increment of 5. In total we have 20 options for the number of requirements and 6 options for the number of stakeholders

Table 3 Selected requirements' characteristics

Requirement category	# of internal requirements	# of external requirements
Overall system requirements	0	29
Control modules requirements	9	17
Instruments requirements	6	19
Network communication requirements	3	29
Safety NFR requirements	2	6
Other requirements	0	15
Total	20	115

Table 4 Classifications of the selected requirements

Requirement categories	# of subsea requirements	# of topside requirements
Overall system requirements	2	71
Control modules requirements	17	41
Instruments requirements	8	38
Network communication requirements	4	63
Safety NFR requirements	5	17
Other requirements	2	19
Total	38	249

and in this way we defined 20*6 = 120 artificial problems. Note that these artificial problems were inspired from the real-world case study, but with a more realistic scale. For our industrial partner, they often face the challenge to assign hundreds and sometimes a thousand requirements to up to 30 stakeholders for each project. For values for different factors such as familiarity, we generated them based on the actual distribution we observed from the real-world case study, for the purpose of making the artificial problems closers to real cases.

3.1.5 Full-scale Empirical Evaluation

In the full-scale empirical evaluation, we evaluate six multi-objective search algorithms in terms of addressing our optimization problem. Same as the pilot study, we evaluated their performance with a real-world case study and a set of artificial problems.

Real-world Case Study The real-world case study in the full-scale empirical evaluation is a large scale CPS in the Energy domain, whose software controls and monitors the operation of electrical and mechanical instruments. We obtained requirements from a well-known subsea engineering handbook (Bai and Bai 2012) and the subsea production control systems standard.. Both the handbook and standard are often used in industrial practice as reference resources for supporting the design and development of subsea production systems. In the end, we selected 287 requirements and classified them in different ways. First they are divided into categories according to the key architectural components of a subsea control system (e.g. control modules, instruments). Second, they are classified according to commonly seen requirements engineering categories such as functional requirements, NFRs and safety requirements (one type of non-functional requirements). The third classification has two categories: subsea or topside. Subsea requirements are considered more important than topside ones as they often lead to more severe consequences. For example, fixing a subsea component is more expensive than fixing a topside component. The characteristics of these requirements are presented in Table 4. The selected requirements were then inputted to an open source requirements management tool and represented as ReqIF models (ReqIF1.1) for enabling the optimization. Details can be found in Section 5.

Artificial Problems The artificial problems design is the same as the one for the pilot study (Section 3.1.4). We created artificial problems with the increasing number of requirements $\left(\bigcup_{n_R=50,\mathrm{n}_R=\mathrm{n}_R+50}^{1000} n_R\right)$ and stakeholders $\left(\bigcup_{n_{st}=5,n_{st}=n_{st}+5}^{30} n_{st}\right)$.

3.1.6 Dependent and Independent Variables

To assess how suitable these selected algorithms are to solve our requirements assignment problem, we defined different types of dependent variables: direct and indirect dependent variables. The direct dependent variable in our context is the final fitness value (*FFV*) of the solution for a problem obtained after executing an algorithm with the fitness function for a certain number of generations (5000 in our experiments). Indirect dependent variables are derived from the direct dependent variable. In the pilot study, each run of each single objective search algorithm generates a solution with a *FFV*. In the full-scale empirical evaluation, each run of each multi-objective search algorithm generates a Pareto front. To be able to compare the results of single objective search algorithms and multi-objective search algorithms, we selected the solution with the smallest overall fitness value (Formula (4) in Section 2.3) from a Pareto front and defined it as *FFV*.

In our analysis, we used two types of indirect dependent variables: Mean Fitness Value for each problem (MFV_p) and Mean Fitness Value for a specific number of problems (MFV_{sp}). MFV_p is a mean FFV for certain number of runs n_r (100 in our case) of a problem with an algorithm and is defined as:

$$MFV_p = \frac{\sum_{r=1}^{n_r} FFV_r}{n_r} \tag{5}$$

MFV_{sp} is a mean MFV_p for a specific number of problems and is defined as follows:

$$MFV_{sp} = \frac{\sum_{p=1}^{n_{sp}} MFV_p}{n_{sp}}, \text{ where } n_{sp} \text{ is the number of problems and } 1 \leq n_{sp} \leq 120.$$

There are three factors (independent variables) that we manipulated in our experiments: types of search algorithms selected (four in the pilot study and eight in the full-scale empirical study) with the fitness function, number of stakeholders (n_{St}) and number of requirements (n_R).

We further instantiate MFV_{sp} into two concrete metrics: MFV_{sp_req} and MFV_{sp_stk}. MFV_{sp_req} represents mean fitness values for all the problems corresponding to a specific requirements category and a specific number of stakeholder categories (n_{sp_stk}). MFV_{sp_req} is defined as:

$$MFV_{sp\_req} = \frac{\sum_{p=1}^{n_{sp\_stk}} MFV_p}{n_{sp\_stk}} \tag{6}$$

Similarly, MFV_{sp_stk} represents mean fitness values for all the problems with a specific stakeholder category and a specific number of requirement categories (n_{sp_req}). $MFV_{sp\_stk.}$ is defined as:

$$MFV_{sp\_stk} = \frac{\sum_{p=1}^{n_{sp\_req}} MFV_p}{n_{sp\_req}} \tag{7}$$

Moreover, to assess the performance of a multi-objective search algorithm, we select Hypervolume (*HV*) from a number of quality indicators (such as Generational distance, Spread, Epsilon and others), which is frequently used in the literature to measure the convergence and uniform diversity of a Pareto front. *HV* calculates the volume, in the objective space, covered by members of a non-dominated set of solutions Q (e.g., $Q = \{S_1, S_2, S_3\}$) (Durillo and Nebro 2011). Each solution $S_i (S_i \in Q)$ has a hypercube V_i, which is constructed with

a reference point W. W can simply be found by constructing a vector of worst objective function values. HV is calculated as:

$$HV = \text{volume}\left(\bigcup_{i=1}^{|Q|} V_i\right) \tag{8}$$

We used jMetal (Durillo and Nebro 2011) to calculate HV values of obtained Pareto fronts.

3.2 Experiment Execution

Each search algorithm was run 100 times for each problem. We let all the algorithms run up to 5000 generations for each problem. We used a PC with Intel Core i3-3110M CPU 2.4 GHz with 4GB of RAM, running the Windows 7 operating system.

For the pilot study with the three single-objective optimizing algorithms, we collected the fitness value (Formula (4) in Section 2.3) calculated in the 5000_{th} generation at each run, for each algorithm to address each problem. For the full-scale empirical study, we obtained a Pareto front in the 5000^{th} generation of running a multi-objective search algorithm, in which optimal solutions are non-dominated to each other based on the three objectives (Section 2.2). A Pareto front was produced at each run and we selected the solution with the smallest overall fitness value (Formula (4) in Section 2.3) from the Pareto front to facilitate the comparison with single objective search algorithms.

3.3 Statistical Tests

We used the Kruskal-Wallis test, the Wilcoxon signed-rank test and the Vargha and Delaney effect size measure to compare the solutions produced by algorithms based on the guidelines proposed by Arcuri (2013).

An obtained p-value of the Kruskal-Wallis test indicates whether there is a significant difference among the selected algorithms. However this test does not tell us which algorithm is significantly different with which algorithm. Therefore, we further performed the Wilcoxon signed-rank test to calculate a p-value for deciding whether there is a significant difference between a pair of search algorithms. We chose the significance level of 0.05.

As investigated in Arcuri (2013), it is insufficient to interpret results only using p-values; statistical test results must be interpreted in conjunction with an effect size measure, which helps determining practical significance of results. We used the Vargha and Delaney statistics ($\hat{A}_{12}$) to calculate the effect size measure, as recommended in Arcuri (2013). In our context, $\hat{A}_{12}$ is used to compare the probability of yielding the highest value for two algorithms A and B. For *ASSIGN* and *FAM* (as maximizing in Section 2.3), if $\hat{A}_{12}$ is equal to 0.5, the two algorithms are equivalent. If $\hat{A}_{12}$ is greater than 0.5, it means A has higher chances of obtaining a better solution than B. For *OWL* and *FS* (as minimizing in Section 2.3), if $\hat{A}_{12}$ is equal to 0.5, the two algorithms are equivalent. If $\hat{A}_{12}$ is less than 0.5, it means the first algorithm A has higher chances of obtaining a better solution than B.

Moreover, in the artificial problems for the full-scale empirical evaluation, to address RQ3 and RQ4, we choose the Spearman's rank correlation coefficient (ρ) (Sheskin 2007) to measure the relations between the *MFV* of the algorithms and different numbers of requirements and stakeholders, respectively. The value of ρ ranges from -1 to 1, i.e., there is a positive correlation if ρ is equal to 1 and a negative correlation when ρ is -1. A ρ close to 0 shows that there is no correlation between the two sets of data. Moreover, we also report significance of correlation using Prob$>|\rho|$, a value lower than 0.05 means the correlation is statistically significant.

4 Result

Results of the pilot study and the full-scale empirical evaluation are presented in Sections 4.1 and 4.2, respectively.

4.1 Results of the Pilot Study

4.1.1 Results of the Real-world Case Study

To answer RQ1 and RQ2, we compared the four algorithms based on the single-objective fitness function (Formula (4), in Section 2.3). Recall that we repeatedly ran each algorithm for 100 times to account for random variation and the mean fitness values (*FS*) used in this analysis is the average of all the 100 runs: MFV_p.

We first performed the Kruskal–Wallis test and obtained a p-value<0.0001 suggesting that there are significant differences among the four algorithms. The Wilcoxon signed-rank test was performed to test the significance of the results and the Vargha and Delaney statistics was used to calculate $\hat{A}_{12}$, of which results are shown in Table 5. Recall that for *FS*, if $\hat{A}_{12}$ equals to 0.5, two algorithms A and B are equivalent. $\hat{A}_{12}$ is less than 0.5, indicating that A has higher chance of obtaining a better solution than B (Section 3.3).

In Table 5, the second, third and fourth rows show the results of the comparison of the three search algorithms with RS (RQ1). One can see that all the three algorithms significantly outperformed RS since all the $\hat{A}_{12}$ values are less than 0.5 and all the p-values are less than 0.05. Based on the results, we can answer RQ1 as follows: AVM, GA and (1+1) EA are all significantly better than RS in finding an optimal solution for our problem.

To answer RQ2, we compared each pair of the four search algorithms. Results are given in the last three rows of Table 5. For each pair, AVM attains significantly worse performance than (1+1) EA since the $\hat{A}_{12}$ value is greater than 0.5 and the p-value is less than 0.05. Both AVM and (1+1) EA significantly outperformed GA because the $\hat{A}_{12}$ values are 0 and the p-values are less than 0.05. We therefore can answer RQ2: (1+1) EA performed significantly better than AVM and GA.

4.1.2 Results of the Artificial Problems

Table 6 summarizes the results of the Wilcoxon signed-rank test for RQ1. The column $A > B$ means the number of problems out of 120 for which an algorithm A was significantly better than B; $A < B$ means the number of problems out of 120 for which an algorithm A was significantly worse than B; and $A = B$ means the number of problems for which

Table 5 Results of the Vargha and Delaney statistics and the Wilcoxon signed-rank test at significance level of 0.05 – real-world case study

	Pair of algorithms	$\hat{A}_{12}$	p-value
RQ1	AVM vs RS	0	<**0.0001**
	(1+1) EA vs RS	0	<**0.0001**
	GA vs RS	0	<**0.0001**
RQ2	AVM vs (1+1) EA	0.72	<**0.0001**
	AVM vs GA	0	<**0.0001**
	(1+1) EA vs GA	0	<**0.0001**

there were no significant differences between A and B based on p-values calculated by the Wilcoxon signed-rank test. Table 6 also provides results of the Vargha and Delaney statistics, i.e., values before "/" in the table. The column $A > B$ means the number of problems out of 120, for which an algorithm A has higher chances of obtaining higher fitness value than B; $A < B$ means the number of problems out of 120, for which an algorithm A has a higher chance of obtaining a lower fitness value than B; and $A = B$ means the number of problems for which there were no differences between A and B as $\hat{A}_{12}$ equals to 0.5.

Results for RQ1 and RQ2 To answer RQ1, we compared each search algorithm with RS, based on the mean fitness values obtained for each problem (MFV_p, $0 < p <= 120$). Results for RQ1 are shown in the first three rows of Table 6. Based on the above results, we can answer RQ1 as follows: AVM, (1+1) EA and GA are all significantly better than RS for finding an optimal solution for all the 120 artificial problems.

Results to answer RQ2 are presented in the last three rows of Table 6. These results are also based on the mean fitness values obtained for each problem (MFV_p, $0 < p <= 120$) for each algorithm.

AVM vs $(1 + 1)$ *EA*: AVM performed better than (1+1) EA for 40 problems, but for 30 problems AVM was significantly better than (1+1) EA. AVM performed worse than (1+1) EA for 80 problems, 67 out of which (1+1) EA was significantly better than AVM. There were no significant differences between the two algorithms for 23 problems.

AVM vs GA: AVM performed better than GA for 120 problems and out of these 120 problems AVM performed significantly better than GA for 119 problems. GA performed significantly worse than AVM for 1 problem and only for 1 problem there was no significant difference between the two algorithms.

$(1 + 1)$ EA vs GA: For all the 120 problems, (1+1) EA was significantly better than GA.

Based on the above results, we can answer RQ2 as follows: (1+1) EA performed significantly better than AVM and GA for more than half of the 120 problems.

Results for RQ3 To answer RQ3, we plotted MFV_{sp_req} (Formula (6) in Section 3.1.6) for all the 120 problems with increasing numbers of requirements for each algorithm $\left(\bigcup_{n_R=50, n_R=n_R+50}^{1000} n_R\right)$ in Fig. 2. In this case, given a fixed number of requirements, we have six problems, each of which corresponds to exactly one stakeholder category; therefore $n_{sp\_stk} = 6$ in Formula (6).

From Fig. 2, RS performed worst regardless of the numbers of requirements since mean fitness values are always greater than 0.45 when the number of requirements is greater

Table 6 Results of the Vargha and Delaney statistic (without/with the Wilcoxon signed-rank test) – artificial problems

	Pair of algorithms (A vs B)	A > B	A < B	A = B
RQ1	AVM vs RS	120/120	0/0	0/0
	(1+1) EA vs RS	120/120	0/0	0/0
	GA vs RS	120/120	0/0	0/0
RQ2	AVM vs (1+1) EA	40/30	80/67	0/23
	AVM vs GA	120/119	0/0	0/1
	(1+1) EA vs GA	120/120	0/0	0/0

than 550. For GA when the number of requirements increases, MFV_{sp_req} for GA increase noticeably. When the number of requirements is greater than 550, the performance of GA is very close to RS. For AVM, we can see that as the number of requirements increases, its MFV_{sp_req} increase suggesting that the performance of AVM degraded gently with the increase in the number of requirements. (1+1) EA is the best algorithm among the four since the growth of MFV_{sp_req} is the slowest.

Results for RQ4 To answer RQ4, we plotted MFV_{sp_stk} (Formula (7) in Section 3.1.6) for the 120 problems with increasing number of stakeholders for each algorithm $\left(\bigcup_{n_{St}=5, n_{St}=n_{St}+5}^{30} n_{St}\right)$ in Fig. 3. In this case, given a fixed number of stakeholders, we have 20 problems, each of which corresponds to exactly one requirement category; therefore $n_{sp\_req} = 20$ in Formula (7).

We can observe from Fig. 3 that for GA and RS, as the number of stakeholders increases MFV_{sp_stk} decreases in a similar trend. MFV_{sp_stk} for GA is less than that for RS. For AVM, we observed that with the increase in the number of stakeholders, MFV_{sp_stk} decreases. However, MFV_{sp_stk} for AVM is much less than GA and RS. When the number of stakeholders is greater than 20, MFV_{sp_stk} decreases quickly and when the number of stakeholders is greater than 25, MFV_{sp_stk} of AVM are near to that of (1+1) EA. For (1+1) EA, we can observe that when the number of stakeholders is greater than 20, its performance stays stable and is not significantly impacted by the increase in the number of stakeholders.

Results for RQ5 To answer RQ5, we plotted mean fitness values for each algorithm with both increasing number of requirements $\left(\bigcup_{n_R=50, n_R=n_R+50}^{1000} n_R\right)$ and increasing number of stakeholders $\left(\bigcup_{n_{St}=5, n_{St}=n_{St}+5}^{30} n_{St}\right)$ in Fig. 4. In this case, given a fixed number of requirements and a fixed number of stakeholders, we have one problem corresponding to exactly one stakeholder and requirement category combination; therefore y-axis in Fig. 4 are MFV_p (Formula (5) in Section 3.1.6).

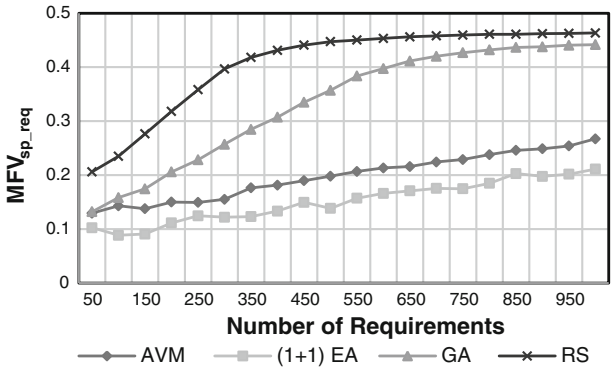

Fig. 2 MFV_{sp_req} with varying numbers of requirements

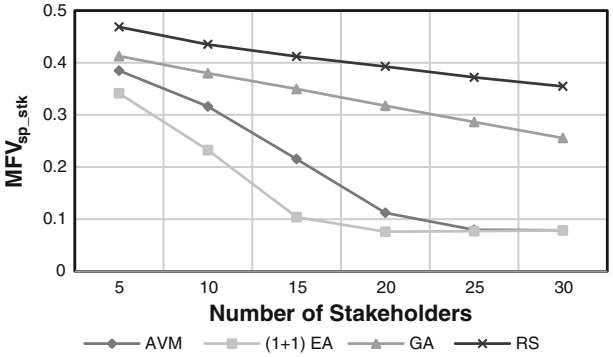

Fig. 3 MFV_{sp_stk} with varying numbers of stakeholders

Results for AVM are shown in Fig. 4, from which one can see that when the number of requirements becomes higher, AVM's performance degraded. When the number of stakeholders is no less than 25, changes in the number of requirements have little impact on the performance of AVM. Results for (1+1) EA are shown in Fig. 4: when the number of stakeholders is greater than 15, even the increase in the numbers of requirements and stakeholders, MFV_p do not significantly increase. When the number of stakeholders is 15, the performance of (1+1) EA maintains stable until the number of requirements is larger than 750. This suggests that (1+1) EA has the ability to solve even complex problems with no much impact on its performance.

For GA and RS shown in Fig. 4, MFV_p is higher for the problems with a smaller number of stakeholders. GA and RS reached worst performance more quickly than the other

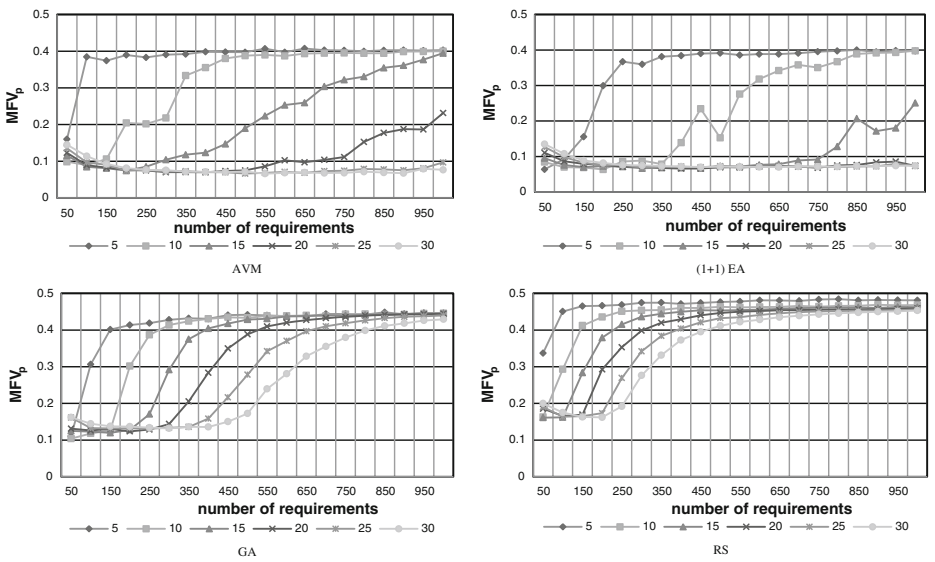

Fig. 4 MFV_p with varying number of requirements and stakeholders

algorithms when the number of requirements is large. For problems with a large number of requirements, the performance of GA and RS remains almost the same.

Based on the above results, we can conclude that (1+1) EA has the ability to solve a wide range of problems without having its performance degraded significantly. We expect that even for more complex problems, (1+1) EA will find optimal solutions.

4.1.3 Summary and Discussion for the Pilot Study

We observed from the real-world case study that AVM, GA and (1+1) EA are all significantly better than RS in finding an optimal solution (RQ1), and (1+1) EA attains the best performance and GA attains the worst performance in terms of finding an optimal solution (RQ2) based on *FS* calculated with Formula (4) in Section 2.3.

For the experiments based on the artificial problems, we observed that AVM, (1+1) EA and GA all significantly outperformed RS for all the artificial problems (RQ1), which is consistent to what we observed from the real-world case study. For the artificial problems, (1+1) EA outperformed AVM and AVM is better than GA (RQ2). When we looked at the impact of varying numbers (50–1000) of requirements on the performance of each algorithm for the artificial problems (RQ3), (1+1) EA was the best among all the four as it achieved better mean fitness values consistently and its performance does not significantly degrade along with the increase of the total number of requirements to assign in the problems, implying that (1+1) EA is the best among the four to solve problems with a large number of requirements to assign. Regarding the impact of varying numbers (5–30) of stakeholders on the performance of the search algorithms (RQ4), both AVM and (1+1) EA did not show significant performance degradation along with the increase of the number of stakeholders. When we looked into the combined impact of both the number of stakeholders and the number of requirements on the performance of the search algorithms (RQ5), we observed that (1+1) EA is the best and has the ability to solve a wide range of problems without having its performance degraded significantly.

(1+1) EA is a global search algorithm and thus managed to find global optimal solutions as compared to AVM, which is a local search algorithm and its performance is good when problems are simpler (Fig. 4). As we can see from Fig. 4 with the increased number of stakeholders and requirements, the performance of AVM decreased. This is due to the fact that with the increased numbers of stakeholders and requirements, there are more local optimal solutions and because of the nature of the working of AVM, it tends to converge to a local optimal and get stuck there, and consequently has to restart from other points in the search space. In contrast, (1+1) EA, due to its use of mutation operators, manages to explore the search space more exhaustively and thus always has a non-zero probability of escaping from local optimal solution (Fig. 4). In the case of GA, even though it is a global search algorithm, its performance is worse as compared to (1+1) EA as shown in Fig. 4. This may be because (1+1) EA uses only mutation for exploring the search space as compared to GA. GA relies on both mutation and crossover for exploration and exploitation of the search space. Therefore, to compare with (1+1) EA, it needs more generations to exploit (via crossover) on a specific area of the search space. Overall, it requires more generations to find a global optimal solution. By increasing the number of generations (5000 in current experiment settings), we expect that the performance of GA might be improved; however this requires more detailed empirical evaluation. Note that increasing the number of generations implies requiring long execution time of a search algorithm. Considering

(1+1) EA already outperformed GA, we therefore do not see the reason to conduct further investigation on the performance of GA, from the perspective of addressing our optimization problem.

4.2 Results of the Full-scale Empirical Evaluation

4.2.1 Results of the Real-world Case Study

Recall that RQ1-RQ2 address questions how the selected algorithms perform for our optimization problems (as compared with RS), while RQ3-RQ4 tackle the question corresponded with the impact on the performance of these algorithms with the growth of the number of requirements and stakeholders respectively. RQ5 tackles the question corresponding to the impact of the growth of the number of requirements and stakeholders on the performance of the algorithms. RQ6 addresses the question about the time performance of these algorithms. For the real-world case study, since we only have one set of requirements to be assigned to one set of stakeholders, it is then irrelevant to answer RQ3, RQ4 and RQ5. In the rest of the section, we report the results for RQ1, RQ2 and RQ6.

Recall from Section 3.2 that for each problem, each algorithm was run 100 times. To compare the algorithms, we selected the solution with the smallest overall fitness value (Formula (4) in Section 2.3) from the Pareto front obtained for each run. We conducted the Wilcoxon signed-rank test at the significance level of 0.05 to compare each pair of the algorithms in terms of the three objectives *ASSIGN*, *FAM*, *OWL* and *FS*. Results are presented in Table 7. Recall that for *ASSIGN* and *FAM*, if $\hat{A}_{12}$ is greater than 0.5, it means that the first algorithm A has higher chances of obtaining a better solution than B, since our aim is to maximize these two objectives. For *OWL* and *FS*, if $\hat{A}_{12}$ is less than 0.5, it means the first algorithm A has higher chances of obtaining a better solution than B since we aim to minimize them.

RQ1 and RQ2 To answer RQ1, as shown in Table 7, in terms of *ASSIGN* only CellDE has no significant difference with RS as the p-value is greater than 0.05. All the other algorithms are significantly better than RS in terms of obtaining a higher value of *ASSIGN* since all the $\hat{A}_{12}$ values are greater than 0.5 and all the p-values are less than 0.05. For *FAM*, *OWL* and *FS*, all the algorithms performed significantly better than RS.

To answer RQ2, as shown in Table 7, for *ASSIGN*, (1+1) EA performed significantly better than NSGA-II since the $\hat{A}_{12}$ value is less than 0.5 and the p-value is less than 0.05. NSGA-II has no significant difference with MOCell and SPEA2. MOCell performed significantly better than PAES, which however performed significantly better than SMPSO and CellDE. SMPSO performed better but not significantly better than CellDE. We therefore can conclude that (1+1) EA achieved the best performance and CellDE is the worst among the all for *ASSIGN*.

For *FAM*, (1+1) EA achieved significantly better performance than NSGA-II, which performed significantly better than MOCell and SPEA2. MOCell and SPEA2 have no significant difference, both of which performed significantly better than PAES. PAES obtained a significantly better performance than CellDE. CellDE performed better but not significantly better than SMPSO. So we can conclude that (1+1) EA achieved the best performance among the all and SMPSO is the worst algorithm for *FAM*.

Table 7 Results of the Vargha and Delaney statistics and the Wilcoxon signed-rank test at the significance level of 0.05– real-world case study

RQ	Pair of algorithms		Objectives							
	(A vs B)		ASSIGN		FAM		OWL		FS	
			$\hat{A}_{12}$	p-value	$\hat{A}_{12}$	p-value	$\hat{A}_{12}$	p-value	$\hat{A}_{12}$	p-value
1	CellDE	RS	0.56	0.9403	0.94	<**0.0001**	0.39	**0.0286**	0.13	<**0.0001**
	MOCell	RS	0.99	<**0.0001**	1	<**0.0001**	0	<**0.0001**	0	<**0.0001**
	NSGA-II	RS	0.97	<**0.0001**	0.99	<**0.0001**	0	<**0.0001**	0	<**0.0001**
	PAES	RS	0.90	<**0.0001**	1	<**0.0001**	0	<**0.0001**	0	<**0.0001**
	SPEA2	RS	0.98	<**0.0001**	0.99	<**0.0001**	0	<**0.0001**	0	<**0.0001**
	SMPSO	RS	0.68	**0.0405**	0.93	<**0.0001**	0.40	**0.0322**	0.12	<**0.0001**
	(1+1) EA	RS	**1**	<**0.0001**	1	<**0.0001**	0	<**0.0001**	0	<**0.0001**
2		MOCell	0.16	<**0.0001**	0	<**0.0001**	1	<**0.0001**	1	<**0.0001**
		NSGA-II	0.14	<**0.0001**	0.05	<**0.0001**	0.99	<**0.0001**	0.96	<**0.0001**
	CellDE	PAES	0.26	<**0.0001**	0	<**0.0001**	0.99	<**0.0001**	1	<**0.0001**
		SPEA2	0.14	<**0.0001**	0.09	<**0.0001**	0.99	<**0.0001**	0.95	<**0.0001**
		SMPSO	0.40	0.1589	0.51	0.7436	0.50	0.9329	0.55	0.3834
		(1+1) EA	0.05	<**0.0001**	0	<**0.0001**	1	<**0.0001**	1	<**0.0001**
		NSGA-II	0.41	0.1034	0.18	<**0.0001**	0.54	**0.4994**	0.82	<**0.0001**
		PAES	0.68	<**0.0001**	0.70	<**0.0001**	0.30	<**0.0001**	0.23	<**0.0001**
	MOCell	SPEA2	0.43	0.2208	0.36	0.1277	0.43	**0.0413**	0.64	<**0.0001**
		SMPSO	0.73	<**0.0001**	1	<**0.0001**	0	<**0.0001**	0	<**0.0001**
		(1 + 1)EA	0.18	<**0.0001**	0	<**0.0001**	1	<**0.0001**	0.67	<**0.0001**
		PAES	0.72	<**0.0001**	0.84	<**0.0001**	0.28	<**0.0001**	0.15	<**0.0001**
	NSGA-II	SPEA2	0.51	0.8123	0.74	<**0.0001**	0.38	**0.0081**	0.26	<**0.0001**
		SMPSO	0.76	<**0.0001**	0.95	<**0.0001**	0	<**0.0001**	0.04	<**0.0001**
		(1 + 1) EA	0.27	<**0.0001**	0	<**0.0001**	1	<**0.0001**	0.20	<**0.0001**
		SPEA2	0.29	<**0.0001**	0.3	**0.0008**	0.63	**0.0010**	0.72	**0.0002**
	PAES	SMPSO	0.60	**0.0002**	1	<**0.0001**	0.01	<**0.0001**	0	<**0.0001**
		(1 + 1) EA	0.12	<**0.0001**	0	<**0.0001**	1	<**0.0001**	0.90	<**0.0001**
	SPEA2	SMPSO	0.76	<**0.0001**	0.91	<**0.0001**	0.01	<**0.0001**	0.06	<**0.0001**
		(1 + 1) EA	0.26	<**0.0001**	0	<**0.0001**	1	<**0.0001**	0.41	0.6959
	SMPSO	(1 + 1) EA	0.11	<**0.0001**	0	<**0.0001**	1	<**0.0001**	1	<**0.0001**

For *OWL*, (1+1) EA significantly outperformed NSGA-II, which performed significantly better than MOCell. MOCell significantly outperformed SPEA2, which performed significantly better than PAES. PAES performed significantly better than CellDE and SMPSO. CellDE revealed no significant difference with SMPSO. Once again (1+1) EA achieved the

best performance among all the algorithms and CellDE and SMPSO are both the worst for *OWL*.

When comparing the overall fitness value considering all the three objectives together, NSGA-II achieved significantly better performance than SPEA2 and (1+1) EA. SPEA2 performed better but not significantly better than (1+1) EA. (1+1) EA performed significantly better than MOCell, which performed significantly better than PAES. PAES outperformed SMPSO and CellDE. SMPSO performed better but not significantly better than CellDE. In general, we can conclude that NSGA-II has the best performance and CellDE has the worst performance.

Based on the above results, we find that all the search algorithms obtained significantly better performance than RS, (1+1) EA achieved the best performance in terms of *ASSIGN, FAM* and*OWL*. NSGA-II achieved the best performance for *FS*. SMPSO achieved the worse performance for *FAM* and*OWL* and CellDE achieved the worse performance for *ASSIGN* and *FS*.

RQ6 In this section, we report the average time taken by all the algorithms. As shown in Table 8 (1+1) EA used on average 0.24 s (the lowest) and SPEA2 spent on average 10.42 s (the highest).

In addition, the Vargha and Delaney test along with the Wilcoxon signed-rank test was performed to evaluate which algorithm achieved the best time performance. Results are reported in Table 9. Recall that a higher run time value means a worse time performance; therefore if $\hat{A}_{12}$ is less than 0.5, it means the first algorithm A has higher chances of obtaining a better time performance than B.

Results show that (1+1) EA achieved significantly better time performance than SMPSO, which achieved better but not significantly better than RS. Except for (1+1) EA and SMPSO, all the other algorithms (i.e., CellDE, MOCell, NSGA-II, PAES and SPEA2) performed worse than RS; CellDE achieved significantly better time performance than NSGA-II, which significantly outperformed PAES. PAES achieved significantly better time performance than MOCell and MOCell significantly outperformed SPEA2. We can therefore conclude that (1+1) EA achieved the best time performance among the all and SPEA2 had the worse time performance.

Table 8 Average running times of the algorithms (in seconds)

Algorithm	Average running time
CellDE	1.13
MOCell	2.53
NSGA-II	2.17
PAES	1.23
SPEA2	10.42
SMPSO	0.80
(1+1) EA	0.24
RS	0.81

Table 9 Results of the Vargha and Delaney and the Wilcoxon signed-rank test at significance level of 0.05 – real-world case study timing analysis

Pair of algorithms		$\hat{A}_{12}$	p-value
CellDE	MOCell	0	<**0.0001**
CellDE	NSGA-II	0	<**0.0001**
CellDE	PAES	0.07	<**0.0001**
CellDE	SPEA2	0	<**0.0001**
CellDE	SMPSO	1	<**0.0001**
CellDE	(1+1) EA	1	<**0.0001**
CellDE	RS	1	<**0.0001**
MOCell	NSGA-II	0.75	<**0.0001**
MOCell	PAES	1	<**0.0001**
MOCell	SPEA2	0	<**0.0001**
MOCell	SMPSO	1	<**0.0001**
MOCell	(1 + 1) EA	1	<**0.0001**
MOCell	RS	1	<**0.0001**
NSGA-II	PAES	0	<**0.0001**
NSGA-II	SPEA2	0	<**0.0001**
NSGA-II	SMPSO	1	<**0.0001**
NSGA-II	(1 + 1) EA	1	<**0.0001**
NSGA-II	RS	1	<**0.0001**
PAES	SPEA2	0	<**0.0001**
PAES	SMPSO	1	<**0.0001**
PAES	(1 + 1) EA	1	<**0.0001**
PAES	RS	1	<**0.0001**
SPEA2	SMPSO	1	<**0.0001**
SPEA2	(1 + 1) EA	1	<**0.0001**
SPEA2	RS	1	<**0.0001**
SMPSO	(1 + 1) EA	1	<**0.0001**
SMPSO	RS	0.4	0.79
(1 + 1) EA	RS	0	<**0.0001**

4.2.2 Results of the Artificial Problems

RQ1 and RQ2 To answer RQ1 and RQ2, we compared the selected search algorithms and then compared each pair of them, based on mean fitness values (MFV_p) achieved after 5000 generations for each algorithm and each of the 120 artificial problems. Recall that each problem was repeated for 100 times to account for random variation. We have 6*20 problems (the number of stakeholders varying from 5 to 30 with an increment of 5, requirements varying from 50 to 1000 with increment of 50). For each of the problems, we conducted the one sample Wilcoxon signed-rank test to compare each pair of the algorithms in terms of *ASSIGN*, *FAM*, *OWL* and *FS*.

Table 10 only shows the problem that is with 50 requirements and 5 stakeholders, for the illustration purpose. As shown in Table 10, all the algorithms achieved the significantly better performance than RS for *ASSIGN*, *FAM*, *OWL* and *FS*. For *ASSIGN* MOCell, SPEA2, NSGA-II and (1+1) EA had no significant difference, each of which performed significantly better than SMPSO. SMPSO performed significantly better than PAES and CellDE, which had no significant difference. For *FAM*, (1+1) EA obtained the best performance while PAES obtained the worst performance. For *OWL* (1+1) EA obtained the best perfor-

Table 10 Partial results for comparing the eight algorithms for each objective and the overall fitness value

RQ	Pair of algorithms		Objectives							
	(A vs B)		ASSIGN		FAM		OWL		FS	
			$\hat{A}_{12}$	p-value	$\hat{A}_{12}$	p-value	$\hat{A}_{12}$	p-value	$\hat{A}_{12}$	p-value
1	CellDE	RS	0.78	<**0.0001**	1	<**0.0001**	0.11	<**0.0001**	0	<**0.0001**
	MOCell	RS	0.88	<**0.0001**	1	<**0.0001**	0.01	<**0.0001**	0	<**0.0001**
	NSGA-II	RS	0.87	<**0.0001**	1	<**0.0001**	0	<**0.0001**	0	<**0.0001**
	PAES	RS	0.78	<**0.0001**	0.78	<**0.0001**	0.05	<**0.0001**	0.08	<**0.0001**
	SPEA2	RS	0.87	<**0.0001**	1	<**0.0001**	0.02	<**0.0001**	0	<**0.0001**
	SMPSO	RS	0.83	<**0.0001**	0.99	<**0.0001**	0.16	<**0.0001**	0	<**0.0001**
	(1 + 1) EA	RS	0.86	<**0.0001**	1	<**0.0001**	0	<**0.0001**	0	<**0.0001**
2		MOCell	0.40	<**0.0001**	0.64	**0.0003**	0.75	<**0.0001**	0.45	0.2171
		NSGA-II	0.40	**0.0001**	0.05	<**0.0001**	0.85	<**0.0001**	0.98	<**0.0001**
	CellDE	PAES	0.51	0.9860	0.97	<**0.0001**	0.62	**0.0046**	0.03	<**0.0001**
		SPEA2	0.40	**0.0001**	0.06	<**0.0001**	0.78	<**0.0001**	0.97	<**0.0001**
		SMPSO	0.44	**0.0145**	0.90	<**0.0001**	0.38	**0.0147**	0.10	<**0.0001**
		(1 + 1)EA	0.41	**0.0003**	0	<**0.0001**	1	<**0.0001**	0.09	<**0.0001**
		NSGA-II	0.51	0.3458	0.01	<**0.0001**	0.65	**0.0007**	0.99	<**0.0001**
		PAES	0.62	<**0.0001**	0.95	<**0.0001**	0.33	**0.0005**	0.04	<**0.0001**
	MOCell	SPEA2	0.51	0.3458	0.02	<**0.0001**	0.55	0.4405	0.98	<**0.0001**
		SMPSO	0.55	**0.0027**	0.83	<**0.0001**	0.13	<**0.0001**	0.12	<**0.0001**
		(1 + 1) EA	0.52	0.1489	0	<**0.0001**	1	<**0.0001**	0.10	<**0.0001**
		PAES	0.61	<**0.0001**	1	<**0.0001**	0.19	<**0.0001**	0	<**0.0001**
	NSGA-II	SPEA2	0.50	1	0.55	0.2620	0.41	0.0032	0.41	0.0718
		SMPSO	0.54	**0.0226**	1	<**0.0001**	0.07	<**0.0001**	0	<**0.0001**
		(1 + 1) EA	0.51	0.7656	0	<**0.0001**	1	<**0.0001**	0	<**0.0001**
		SPEA2	0.39	<**0.0001**	0.00	<**0.0001**	0.71	<**0.0001**	1	<**0.0001**
	PAES	SMPSO	0.43	**0.0140**	0.17	<**0.0001**	0.25	<**0.0001**	0.80	<**0.0001**
		(1 + 1) EA	0.40	**0.0001**	0	<**0.0001**	1	<**0.0001**	0.73	<**0.0001**
	SPEA2	SMPSO	0.54	**0.0226**	1	<**0.0001**	1	<**0.0001**	0	**0.0198**
		(1 + 1) EA	0.51	0.7656	0	<**0.0001**	1	<**0.0001**	0	<**0.0001**
	SMPSO	(1 + 1) EA	0.46	**0.0498**	0	<**0.0001**	1	<**0.0001**	0.40	<**0.0001**

Table 11 Results of the Vargha and Delaney statistical test – 120 artificial problems (without/with the Wilcoxon signed-rank test)

RQ	Pair of algorithms		ASSIGN			FAM			OWL			FS		
	(A vs B)		A>B	A<B	A=B	A>B	A<B	A=B	A>B	A<B	A=B	A>B	A<B	A=B
1	CellDE	RS	117/108	3/0	0/12	120/120	0/0	0/0	73/51	47/17	0/52	120/120	0/0	0/0
	MOCell	RS	120/120	0/0	0/0	120/120	0/0	0/0	120/120	0/0	0/0	120/120	0/0	0/0
	NSGA-II	RS	120/120	0/0	0/0	120/120	0/0	0/0	120/120	0/0	0/0	120/120	0/0	0/0
	PAES	RS	120/120	0/0	0/0	120/120	0/0	0/0	120/120	0/0	0/0	120/120	0/0	0/0
	SPEA2	RS	120/120	0/0	0/0	120/120	0/0	0/0	120/120	0/0	0/0	120/120	0/0	0/0
	SMPSO	RS	120/119	0/0	0/1	119/119	1/0	0/1	84/51	36/21	0/48	120/120	0/0	0/0
	(1+1) EA	RS	120/120	0/0	0/0	120/120	0/0	0/0	120/120	0/0	0/0	120/120	0/0	0/0
2	CellDE	MOCell	3/2	117/117	0/1	1/1	119/118	0/1	0/0	120/120	0/0	1/0	119/119	0/1
		NSGA-II	0/0	120/120	0/0	0/0	120/120	0/0	0/0	120/120	0/0	0/0	120/120	0/0
		PAES	36/28	84/70	0/22	6/4	114/113	0/3	0/0	120/120	0/0	6/4	114/113	0/3
		SPEA2	0/0	120/120	0/0	0/0	120/120	0/0	0/0	120/120	0/0	0/0	120/120	0/0
		SMPSO	0/0	120/118	0/2	62/51	58/40	0/29	51/20	69/31	0/69	45/30	75/65	0/25
		(1+1) EA	0/0	120/120	0/0	0/0	120/120	0/0	0/0	120/120	0/0	6/5	114/114	0/1
	MOCell	NSGA-II	62/11	51/23	7/86	0/0	120/120	0/0	108/97	12/3	0/20	0/0	120/120	0/0
		PAES	120/116	0/0	0/4	40/37	80/71	0/12	111/105	9/3	0/12	52/50	68/64	0/6
		SPEA2	89/62	25/19	6/39	0/0	120/118	0/2	118/117	2/0	0/3	1/0	119/110	0/10
		SMPSO	101/90	17/10	2/20	120/120	0/0	0/0	118/118	2/2	0/0	120/120	0/0	0/0
		(1+1) EA	3/0	114/65	3/55	0/0	120/120	0/0	0/0	120/120	0/0	23/20	97/95	0/5

Table 11 (continued)

RQ	Pair of algorithms		ASSIGN			FAM			OWL			FS		
	(A vs B)		A>B	A<B	A=B	A>B	A<B	A=B	A>B	A<B	A=B	A>B	A<B	A=B
	NSGA-II	PAES	120/115	0/0	0/5	111/104	9/6	0/10	24/16	96/83	0/21	112/108	8/6	0/6
		SPEA2	114/57	4/0	2/63	119/113	1/0	0/7	120/116	0/0	0/4	120/116	0/0	0/4
		SMPSO	108/103	10/3	2/14	120/120	0/0	0/0	118/118	2/2	0/0	120/120	0/0	0/0
		(1+1) EA	2/0	112/82	6/38	1/1	119/118	0/1	0/0	120/120	0/0	107/98	13/9	0/13
	PAES	SPEA2	1/0	119/115	0/5	41/39	79/71	0/10	112/110	8/3	0/7	42/38	78/75	0/7
		SMPSO	4/2	116/105	0/13	117/116	3/2	0/2	118/118	2/2	0/0	109/105	11/8	0/7
		(1+1) EA	0/0	120/117	0/3	0/0	120/120	0/0	0/0	120/120	0/0	39/28	81/74	0/18
	SPEA2	SMPSO	104/97	16/11	0/12	120/120	0/0	0/0	118/118	2/2	0/0	120/120	0/0	0/0
		(1+1) EA	1/0	117/107	2/13	1/0	119/118	0/2	0/0	120/120	0/0	57/52	63/59	0/9
	SMPSO	(1+1) EA	0/0	118/109	2/11	0/0	120/120	0/0	0/0	120/120	0/0	5/5	115/115	0/0

mance while SMPSO obtained the worst performance For *FS*, NSGA-II and SPEA2 had no significant difference, both of which performed significantly better than CellDE CellDE revealed no significant difference with MOCell, both of which performed significantly better than SMPSO, followed by (1+1) EA and PAES.

Table 11 summarizes the results of the Vargha and Delaney statistic test (with or without the Wilcoxon signed-rank test applied) for the 120 artificial problems. Results in Table 11 show that for all the objectives and the overall fitness (*FS*), all the search algorithms significantly outperformed RS for most of the 120 artificial problems. Based on the results presented in Table 11, we can answer RQ2 as follows: 1) For *ASSIGN*, (1+1) EA performed significantly better than NSGA-II, SPEA2, MOCell, SMPSO, PAES and CellDE for 82, 107, 65, 109, 117 and 120 problems respectively; CellDE achieved the worst performance; 2) For *FAM*, (1+1) EA performed significantly better than NSGA-II for 118 problems and the other algorithms for all the 120 problems; CellDE achieved the worst performance; 3) For *OWL*, (1+1) EA performed significantly better than the other algorithms for all the 120 problems; CellDE achieved the worst performance; and 4) For *FS*, NSGA-II performed significantly better than each other algorithm for over 90 problems; CellDE achieved the worst performance.

RQ3 To answer RQ3, we calculated the Spearman's rank correlation of MFV_{sp_req} (Formula (6) in Section 3.1.6) with the increase in the number of requirements $\left(\bigcup_{n_R=50, n_R=n_R+50}^{1000} n_R\right)$. Given a fixed number of requirements, we have six problems, each of which corresponds to exactly one stakeholder category; therefore $n_{sp\_stk} = 6$ in Formula (6).

As shown in Table 12, the values of MFV_{sp_req} of all the algorithms (except for SPEA2 and (1+1) EA) increase significantly along with the increasing number of requirements, since the values of Spearman's ρ are greater than 0 and the p-values are less than 0.05. For SPEA2 and (1+1) EA, the value of MFV_{sp_req} increases but not significantly increases along with the increasing number of requirements, since the Spearman's ρ value is positive and the p-values are greater than 0.05. Based on the results, we can conclude that the performance of SPEA2 and (1+1) EA are not impacted significantly by the number of requirements and for the other algorithms, their performance decreases significantly with the increasing number of requirements.

Table 12 Spearman's correlation analysis of MFV_{sp_req} with the increasing number of requirements (full-scale empirical evaluation)

	Increasing number of requirements	
Algorithm	Spearman's ρ	Prob> |ρ|
CellDE	0.96	< **.0001**
MOCell	0.97	< **.0001**
NSGA-II	0.62	< **0.0034**
PAES	0.89	< **.0001**
SMPSO	0.97	< **.0001**
SPEA2	0.08	0.7479
(1+1) EA	0.17	0.4620
RS	0.95	< **.0001**

As shown in Table 12, the values of MFV_{sp_req} of all the algorithms (except SPEA2 and (1+1) EA) increase significantly along with the increasing number of requirements, since the values of Spearman's ρ are greater than 0 and the p-values are less than 0.05. For SPEA2 and (1+1) EA, the value of MFV_{sp_req} increases but not significantly increases along with the increasing number of requirements, since the Spearman's ρ value is positive and the p-values are greater than 0.05. Based on the results, we can conclude that the performance of SPEA2 and (1+1) EA are not impacted significantly by the number of requirements and for the other algorithms, their performance decreases significantly with the increasing number of requirements.

RQ4 To answer RQ4, we calculated the Spearman's rank correlation between MFV_{sp_stk} (Formula (7) in Section 3.1.6) with the increasing number of stakeholders $\left(\bigcup_{n_{st}=,n_{st}=n_{st}+5}^{30} n_{st}\right)$. Given a fixed number of stakeholders, we have 20 problems, each of which corresponds to exactly one requirement category; therefore $n_{sp\_req} = 20$ in Formula (7). The results are presented in Table 13. We can observe from the table that for RS, the values of MFV_{sp_stk} increase but not significantly along with the increasing of the number of stakeholders since the values of Spearman's ρ are greater than 0 and the p-values are greater than 0.05. For CellDE, NSGA-II, SMPSO, SPEA2, and (1+1) EA, the values of MFV_{sp_stk} increase significantly along with the increase in the number of stakeholders. For MOCell, the values of MFV_{sp_stk} decrease but not significantly. Based on the results, we can conclude that PAES seems more appropriate to handle problems with a large number of stakeholders.

RQ5 To answer RQ5, we plotted MFV_p (Formula (5) in Section 3.1.6) for each algorithm for all the 120 problems with both the increasing number of requirements $\left(\bigcup_{n_R=50,\mathrm{n}_R=\mathrm{n}_R+50}^{1000} n_R\right)$ and the increasing number of stakeholders $\left(\bigcup_{n_{St}=5,\mathrm{n}_{St}=\mathrm{n}_{St}+5}^{30} n_{St}\right)$, as shown in Fig. 5. There are eight line charts for the eight search algorithms in the figure. In each line chart, x-axis is the number of requirements and y-axis is MFV_p. Six lines in each chart respectively represent the six stakeholder categories.

Table 13 Spearman's correlation analysis of MFV_{sp_stk} with the increasing number of stakeholders (full-scale empirical evaluation)

	Increasing number of stakeholders	
Algorithm	Spearman's ρ	Prob> |ρ|
CellDE	1	<**.0001**
MOCell	−0.2	0.7040
NSGA-II	0.8286	**0.0416**
PAES	−1	<**.0001**
SMPSO	1	<**.0001**
SPEA2	0.94	<**.0001**
(1+1) EA	0.94	<**.0001**
RS	0.09	0.8717

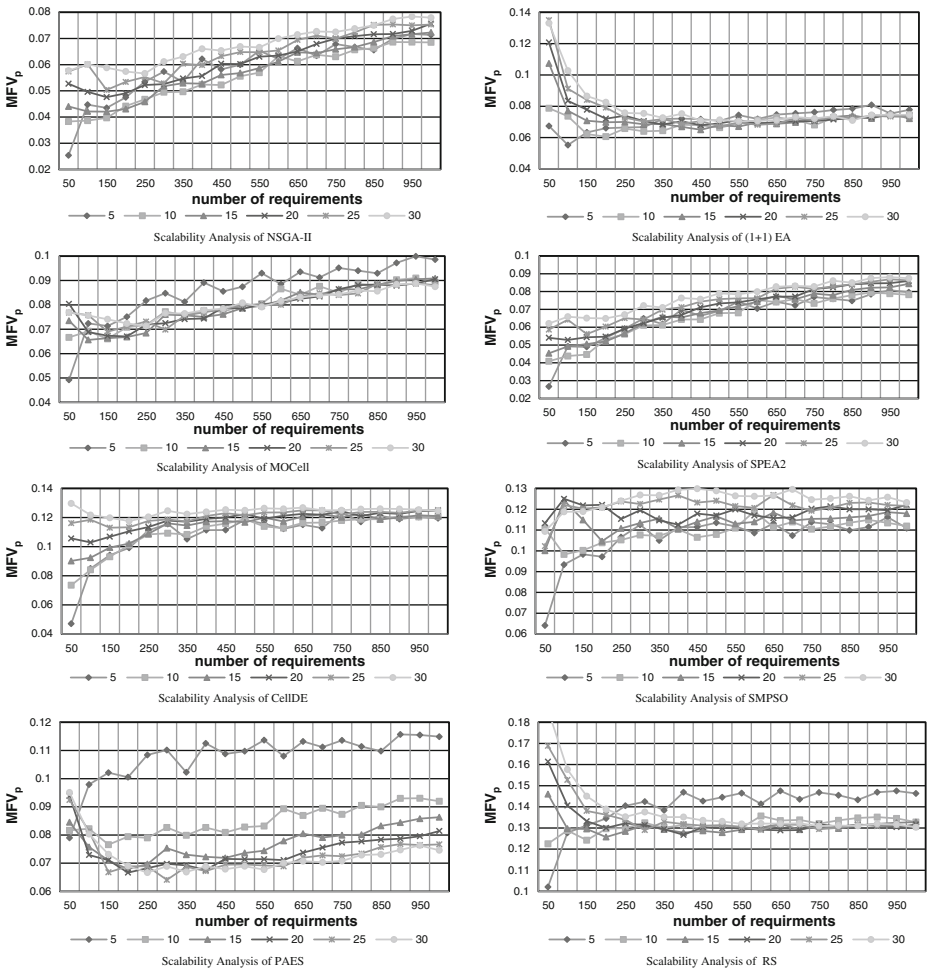

Fig. 5 MFV_p with varying numbers of requirements and stakeholders

One can observe from Fig. 5 that NSGA-II performed the worst for problems with 30 stakeholders. For all the stakeholder categories, MFV_p increases with the increase in the number of requirements suggesting that its performance decreases with the increase in the number of requirements. Regarding (1+1) EA, one can observe from the figure that it is not fit for the problems when the number of requirements is less than 250, as MFV_p values of these problems are much higher than the ones for the problems with a higher number of requirements. The difference of the performance of (1+1) EA for solving problems with different number of stakeholders gets lesser noticeable with the increase in the number of requirements.

As for MOCell, MFV_p increases with the increase in the number of requirements. When the number of requirements is no more than 100, MOCell obtains the least MFV_p value for the problems with 5 stakeholders. When the number of requirements increases, MFV_p for the problems with 5 stakeholders increases quite noticeably. When requirements are

greater than 300, MOCell obtained the worst performance for the problems with 5 stakeholders. Results for SPEA2 are similar to what we observed for NSGA-II, which in general, obtained better performance than SPEA2. The performance of SPEA2 decreases along with the increase of the number of requirements, especially when the number of stakeholders is small.

For CellDE, one can see that the line of 30 stakeholders has the largest MFV_p value. For different stakeholder categories, CellDE' performance for problems with more stakeholders is worse than those with fewer stakeholders. When the number of requirements is less than 300, the difference between different stakeholder categories is large. From Fig. 5, we can observe that SMPSO's performance is the worse for the problems with 30 stakeholders, followed by 25 stakeholders. It is interesting to notice that when the number of requirements gets higher, the difference between different stakeholder categories gets smaller.

For PAES, MFV_p values are higher for the problems with 5 stakeholders, followed by the problems with 10 stakeholders, and so on. When requirements are less than 250, MFV_p values decrease with the increase in the number of requirements for problems with more than 5 stakeholders while MFV_p values increase with the increase in the number of requirements for problems with 5 stakeholders. In general, with the increase in the number of requirements, MFV_p values increase. Results of RS show that MFV_p values are very constantly high.

Based on the above results, we can conclude that NSGA-II has the ability to solve a wide range of problems without having its performance degraded much. (1+1) EA is fit for the problems with more than 250 requirements and more than 5 stakeholders. Except PAES, the performance of the other algorithms degrades with the increase in the number of requirements and improves along with the increase in the number of stakeholders.

RQ6 We report the average time of all the problems taken by the eight algorithms in Table 14. One can notice that SPEA2 took more than 10 s to find optimal solutions, while the other algorithms however took much less time.

We conducted the Vargha and Delaney test for running time (with or without the Wilcoxon signed-rank test applied) for the 120 artificial problems. Results are shown in Table 15. From this table, we can find that for running time (1+1) EA performed significantly better than SMPSO for all the 120 problems. SMPSO performed better than RS for 108 problems, among which for 103 problems results are statistical significant. SMPSO

Table 14 Average running times for the selected algorithms (full-scale empirical evaluation)

Algorithm	Average running time (in seconds)
CellDE	1.80
MOCell	4.23
NSGA-II	3.72
PAES	1.81
SPEA2	10.92
SMPSO	1.33
(1+1) EA	0.95
RS	1.42

performed worse than RS for 12 problems, for 7 problems out of which SMPSO performed significantly worse. There are no significantly differences between SMPSO and RS for 10 problems. RS performed significantly better for all the 120 problems than CellDE, MOCell, NSGA-II, PAES and SPEA2. Among these five search algorithms, PAES had the very close time performance with CellDE, for 61 problems CellDE performed better than PAES and for 60 out of 61 CellDE is significantly better, and for 59 problems PAES performed better than CellDE and for 57 out of 59 PAES is significantly better. For three problems, there are

Table 15 Results of the Vargha and Delaney statistical test for time – 120 artificial problems (without/with the Wilcoxon signed-rank test) (full-scale empirical evaluation)

Pair of algorithms			A>B	A<B	A=B
(A	vs.	B)			
CellDE		MOCell	120/120	0/0	0/0
		NSGA-II	120/120	0/0	0/0
		PAES	61/60	59/57	0/3
		SPEA2	120/120	0/0	0/0
		SMPSO	0/0	120/120	0/0
		(1 + 1) EA	0/0	120/120	0/0
		RS	0/0	120/120	0/0
MOCell		NSGA-II	21/13	99/93	0/14
		PAES	0/0	120/120	0/0
		SPEA2	120/120	0/0	0/0
		SMPSO	0/0	120/120	0/0
		(1 + 1) EA	0/0	120/120	0/0
		RS	0/0	120/120	0/0
NSGA-II		PAES	0/0	120/120	0/0
		SPEA2	120/120	0/0	0/0
		SMPSO	0/0	120/120	0/0
		(1 + 1) EA	0/0	120/120	0/0
		RS	0/0	120/120	0/0
PAES		SPEA2	120/120	0/0	0/0
		SMPSO	0/0	120/120	0/0
		(1 + 1) EA	0/0	120/120	0/0
		RS	0/0	120/120	0/0
SPEA2		SMPSO	0/0	120/120	0/0
		(1 + 1) EA	0/0	120/120	0/0
		RS	0/0	120/120	0/0
SMPSO		(1 + 1) EA	0/0	120/120	0/0
		RS	108/103	12/7	0/10
(1 + 1) EA		RS	120/120	0/0	0/0

no significant differences. CellDE and PAES performed significantly better than NSGA-II for 120 problems, which performed significantly better than MOCell for 93 problems. MOCell outperformed SPEA2 for all the 120 problems.

Based on the above results, we can conclude that (1+1) EA and SMPSO are the two algorithms that achieved significantly better performance than RS. Beyond these two, the others performed worse than RS and SPEA2 gets the worse time performance. Notice that this observation is consistent with the one we made for the real-world case study (Section 4.2.1).

4.2.3 Overall Discussion of the Full-scale Empirical Evaluation

In this section, we provide an overall discussion of the full-scale empirical study. Section 4.2.3 provides insights on the results of RQ1-RQ2, while Section 4.2.3 discusses results of RQ3-RQ4 and Section 4.2.3 discusses results of RQ5. Recall that RQ1-RQ2 address questions related to the performance of the selected algorithms in terms of finding optimal solutions: whether a selected search algorithm is better than RS and which one is the best one among the all. RQ3-RQ5 refers to the impact of the increasing number of requirements or stakeholders or both. RQ6 addresses the question related to the time performance in Section 4.2.3. Before getting deeper into specific discussions, the key results are first summarized below (summarized in Table 16):

- (1+1) EA achieved the best performance in terms of finding optimal solutions for *ASSIGN*, *FAM* and *OWL*; NSGA-II achieved the best performance for *FS*;
- The performance of SPEA2 and (1+1) EA in terms of finding optimal solutions does not significantly change along with the increase in the number of requirements. The performance of the other algorithms decreases along with the increase in the number of requirements;
- PAES's performance in terms of finding optimal solutions increases significantly with the increase in the number of stakeholders. The other algorithms' performance decreases significantly with the increasing the number of stakeholders;
- (1+1) EA and NSGA-II were able to solve a wide range of problems without having their performance (in terms of finding optimal solutions) degraded significantly, but (1+1) EA is not fit for problems with less than 300 requirements;
- Regarding the performance of the algorithms in terms of execution time, (1+1) EA achieved the best performance; SPEA2 however required more time than all the other algorithms.

In Section 4.1.3, we have discussed the results for the pilot study. (1+1) EA is the best among all the search algorithms (i.e., AVM, GA, (1+1) EA and RS). Therefore, in full-scale empirical evaluation, we evaluated (1+1) EA and the other six multi-objective algorithms (i.e., CellDE, MOCell, NSGA-II, PAES, SMPSO and SPEA2). Results show that (1+1) EA still performed well, but not as good as NSGA-II in terms of *FS*. (1+1) EA is significantly better than all the multi-objective algorithms in terms of *ASSIGN*, *FAM* and *OWL* for the real-world case study and more than half of the 120 artificial problems. Details are summarized in Table 16 Therefore we recommend that, if a project manager is interested in a particular objective then (1+1) EA should be used; otherwise, NSGA-II should be applied to obtain optimal solutions when balancing all the objectives.

Discussion Related to RQ1 and RQ2 When comparing the selected algorithms with RS, we observed that all the selected algorithms achieved significantly better performance

Table 16 Summary for the full-scale empirical evaluation

Problem	RQ	Statistical analysis	Key results
Real-world case study	1	Wilcoxon signed-rank test and	All the algorithms (except CellDE) performed significantly better than RS for *ASSIGN*, *FAM*, *OWL* and *FS*
	2	Vargha and Delaney statistical test	For *ASSIGN* (1+1) EA performed the best and CellDE performed the worst
			For *FAM*, (1+1) EA performed the best and SMPSO performed the worst
			For *OWL*, (1+1) EA performed the best and CellDE and SMPSO performed the worst
			For *FS*, NSGA-II performed the best and SMPSO performed the worst
	6		(1+1) EA performed the best and SPEA2 performed the worst
Artificial problems	1	Wilcoxon signed-rank test and	For *ASSIGN* (*FAM, OWL, FS*), all the algorithms significantly outperformed RS for over 107 (118, 50, 120) problems
	2	Vargha and Delaney statistical test	For *ASSIGN* (1+1) EA performed the best and CellDE performed the worst
			For *FAM*, (1+1) EA performed the best and CellDE performed the worst
			For *OWL*, (1+1) EA performed the best and CellDE performed the worst
			For *FS*, NSGA-II performed the best and CellDE performed the worst
	3	Spearman's correlation analysis	The performance of SPEA2 and (1+1) EA is not impacted significantly with an increasing number of requirements
			Others' performance decreases with the increase in the number of requirements
	4		PAES's performance increases significantly with the increase in the number of stakeholders
			The performance of MOCell is not impacted significantly with an increasing number of stakeholders
			The other algorithms' performance decreases significantly with the increase in the number of stakeholders

Table 16 (continued)

Problem	RQ	Statistical analysis	Key results
	5	N/A	NSGA-II is able to solve a wide range of problems without having its performance degraded much. (1+1) EA is fit for the problems with more than 250 requirements and more than 5 stakeholders
	6	Wilcoxon signed-rank test and Vargha and Delaney statistical tests	(1+1) EA performed the best and SPEA2 performed the worst

than RS. We can conclude that search-based techniques are fit for solving our requirements assignment problem. When looking into the performance of each algorithm, (1+1) EA achieved the best performance for objectives *ASSIGN*, *FAM* and *OWL*. However, when combining all the objectives together (measured as *FS*), NSGA-II achieved significantly better performance than all the others. Though (1+1) EA is a global single-objective search algorithm and managed to find global optimal solutions, NSGA-II is a multi-objective search algorithm and for each run, it generates a Pareto front, in which each solution is non-dominated to others. To compare with the other algorithms (e.g., (1+1) EA), we selected the solution with the smallest *FS* from a Pareto front. So NSGA-II always obtained better performance for*FS* than (1+1) EA. For example, for the problem with 50 requirements and 5 stakeholders, NSGA-II produced 7 solutions in the Pareto front (Table 17). In Table 17 each row represents a solution. Each solution has values for *ASSIGN*, *FAM OWL* and *FS*. So we selected the 5th solution as the final result of NSGA-II for the problem.

Discussion Related to RQ3 and RQ4 To further look into the performance of each algorithm as the number of requirements increases (RQ3) and as the number of stakeholders increases (RQ4), we plotted MFV_{sp_req}(Formula (6) in Section 3.1.6, $n_{sp\_stk} = 6$) and MFV_{sp_stk} (Formula (7) in Section 3.1.6, $n_{sp\_req} = 20$) for all the 120 problems for each algorithm as shown in Figs. 6 and 7.

From Figs. 6 and 7, RS is always the algorithm with the worst performance, since RS chooses solutions randomly without any guideline. CellDE and SMPSO have almost equivalent performance and their performance stays at the second best. MOCell, SPEA2 and NSGA-II behaved in a similar pattern, i.e., the performance decreases kbut not significantly with the increase in the number of requirements and stakeholders. For (1+1) EA, its performance increases with the increase in the number of requirements and PAES's performance increases with the increase in the number of stakeholders.

Discussion Related to RQ5 According to Section 4.2.2, we can conclude that for each algorithm, the MFV_p of problems with 5 stakeholders is higher than those with more than 5 stakeholders. To further discuss the results of RQ5, for the 120 artificial problems, we further plotted surfaces of MFV_p (Formula (5) in Section 3.1.6) for each algorithm with both the increasing number of requirements $\left(\bigcup_{n_R=50, n_R=n_R+50}^{1000} n_R\right)$ and the increasing number of stakeholders $\left(\bigcup_{n_{St}=5, n_{St}=n_{St}+5}^{30} n_{St}\right)$. Figures 8, 9, 10, 11, 12, 13, 14, and 15 show the combined effect of the increase in the number of requirements and stakeholders on the

Table 17 A Pareto front generated by NSGA-II for the problem with 50 requirements and 5 stakeholders

Solution	ASSIGN	FAM	OWL	FS
1	1	0.78	0.0007	0.0736
2	0.96	0.8958	0.0089	0.051
3	1	0.8	0.0008	0.0669
4	1	0.76	0.0006	0.0802
5	1	0.88	0.0013	**0.0404**
6	1	0.86	0.0012	0.0471
7	1	0.82	0.0010	0.0603

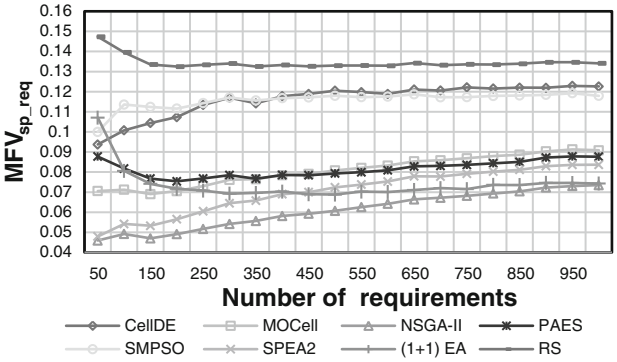

Fig. 6 MFV_{sp_req} with varying number of requirements

performance of the search algorithms. The x-axis and y-axis of the figure are the number of stakeholders and the number of requirements respectively, and z-axis is MFV_p.

For RS, we can see from Fig. 8 that when the number of requirements is less than 200, MFV_p values dramatically increase with the increase in the number of stakeholders. When the number of requirements is above 200 and the number of stakeholders is 15, obtained MFV_p values are the smallest. When the number of requirements is greater than 300, the number of requirements has little impact on the performance of RS.

From Fig. 9, we can see that CellDE obtains the smallest MFV_p when the number of stakeholders and requirements is the minimum (5 and 50 respectively). When stakeholders are less than 20 and requirements are less than 300, MFV_p values increase with the increasing the number of requirements and stakeholders.

From the results for MOCell (Fig. 10), we can observe that variation trend of MFV_p is different in the area of stakeholders less than 10 and requirements less than 200 from the other area, since MFV_p increases quickly with the increasing the number of stakeholders and requirements. In most situations, MFV_p first decreases with the increasing the number of stakeholders and then increase. MFV_p obtains the smallest value when stakeholders are 10. From Fig. 10, we can observe that MFV_p increases with the increasing the number of requirements.

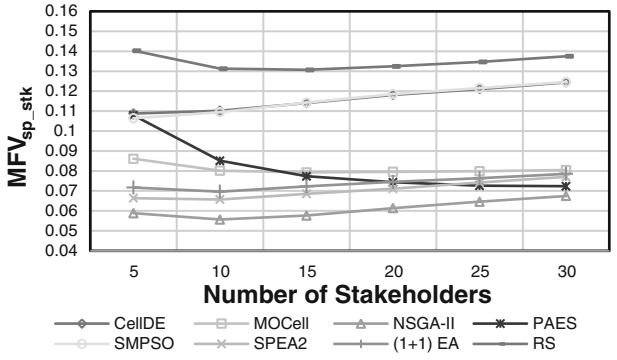

Fig. 7 MFV_{sp_stk} with varying number of stakeholders

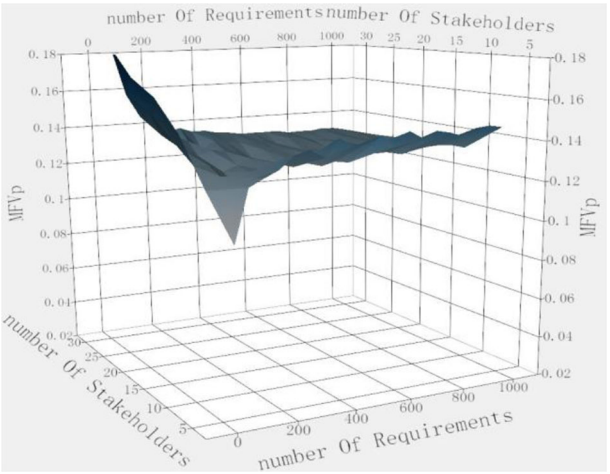

Fig. 8 Surface plot for RS

For PAES, as shown in Fig. 11, one can observe that the values of MFV_p decease with the increase of the number of requirements and stakeholders when requirements are less than 400 and stakeholders are less than 15. When the number of requirements is greater than 400 and stakeholders are more than 15, MFV_p increases slowly with the increasing of the number of requirements and decreases slowly with the increasing of the number of stakeholders.

As shown in Fig. 12, for SMPSO, its performance is the best for the problems with 5 stakeholders and 50 requirements. With the increasing of the number of stakeholders and requirements, its performance decreases quickly. When stakeholders are more than 10 and requirements are more than 200, the performance decreases with the increase of the number of stakeholders and requirements but not so obviously.

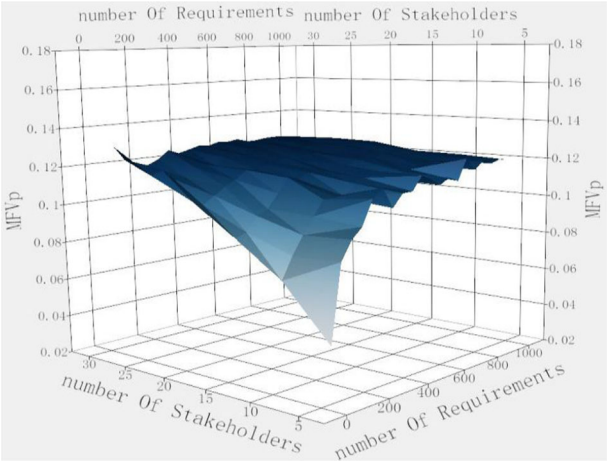

Fig. 9 Surface plot for CellDE

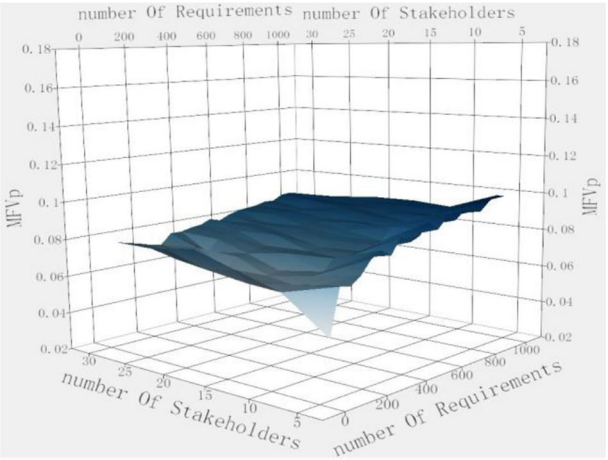

Fig. 10 Surface plot for MOCell

The results for SPEA2 (Fig. 13) show that the MFV_p value reaches the smallest when the number of stakeholders is 10. With the variation in stakeholders from 5 to 10, and requirements from 50 to 200 $MFV_{p.}$ increases quickly. In general, SPEA2' performance decreases with the increasing of the number of requirements and stakeholders.

Results for (1+1) EA (shown in Fig. 14) indicate that, when the number of requirements is less than 200, the performance of (1+1) EA decreases with the increasing number of the stakeholders. When requirements are more than 200, the performance is impacted significantly with varying number of requirements.

As shown in Fig. 15, for NSGA-II, even with the increase in the numbers of requirements and stakeholders, MFV_p values do not significantly increase. This suggests that NSGA-II has the ability to solve complex problems with no much impact on its performance. We can

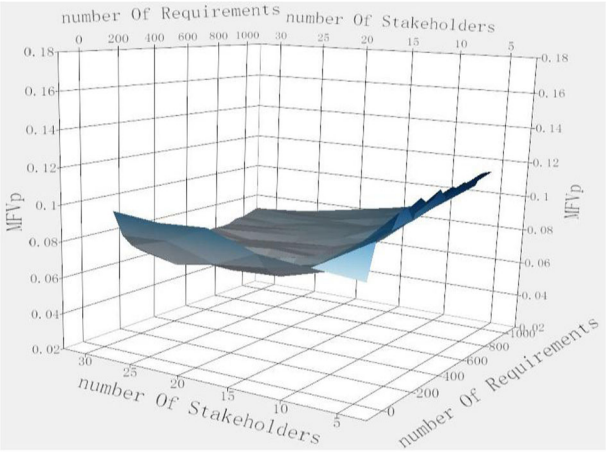

Fig. 11 Surface plot for PAES

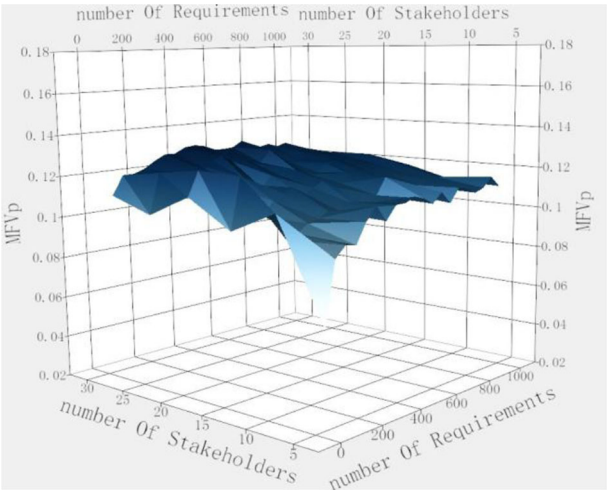

Fig. 12 Surface plot for SMPSO

also observe that MOCell, SPEA2 and NSGA-II have the very similar surface plots and the difference is that NSGA-II has the best performance in general.

Based on the results for the artificial problems, we can conclude that: NSGA-II achieved the best performance and stays stable as the increasing number of requirements and stakeholders. (1+1) EA is a fit for problems with a relatively large number of stakeholders. The performance of MOCell and PAES increases as the number of requirements increases. We also observed from the real-world case study that (1+1) EA and NSGA-II are the algorithms with the best performance to find an optimal solution for the problem. These results are consistent with what we observed from the artificial problems.

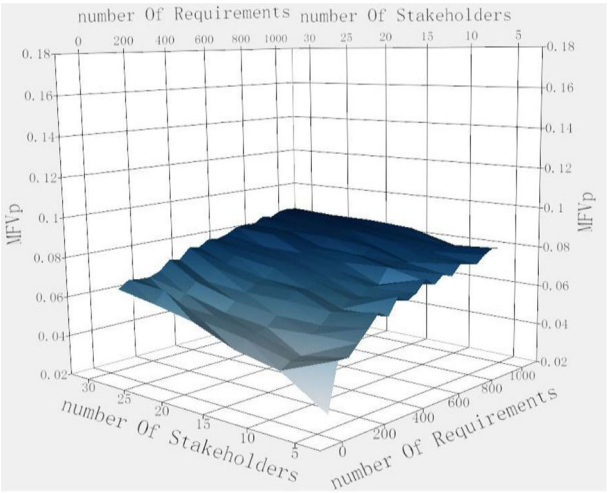

Fig. 13 Surface plot for SPEA2

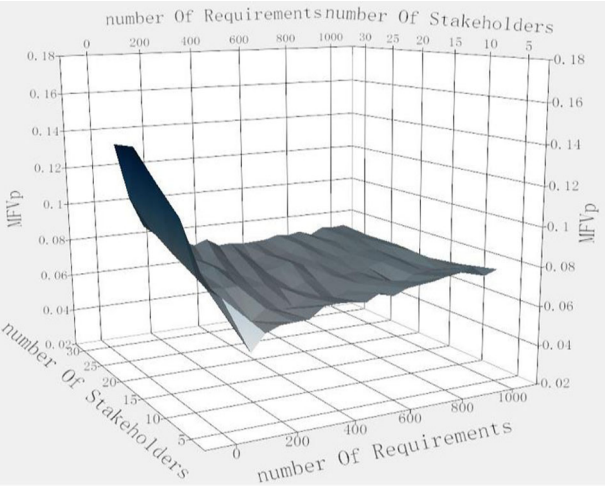

Fig. 14 Surface plot for (1 + 1) EA

When we look at the impact of varying numbers (50–1000) of requirements on the performance of each algorithm for the artificial problems, NSGA-II is the best as it achieved better mean fitness values consistently and its performance does not significantly degrade along with the increase in the number of requirements to assign in the problems, implying that NSGA-II is the best to solve problems with a large number of requirements to assign. Regarding the impact of varying number (5–30) of stakeholders on the performance of the search algorithms, NSGA-II also did not show significant performance degradation along with the increase in the number of stakeholders. When we looked into the combined impact of both the number of stakeholders and the number of requirements on the performance of the search algorithms, we observed that NSGA-II and (1+1) EA are the best two, but (1+1) EA is not a fit for the problems with less than 200 requirements and NSGA-II

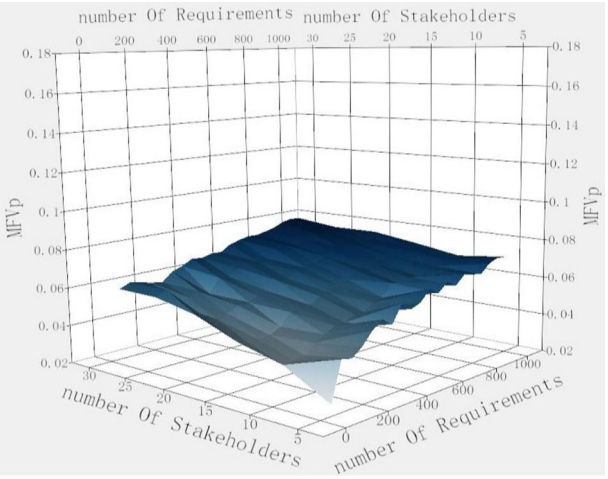

Fig. 15 Surface plot for NSGA-II

has the ability to solve a wide range of problems without having its performance degraded significantly. From the surface plots (Figs. 14 and 15) we can find (1+1) EA and NSGA-II both obtained good performance with large number requirements, since (1+1) EA is a global search algorithm and thus managed to find global optimal solutions and its use of mutation operators leads to explore the search space more exhaustively, while NSGA-II uses a fast non-dominated sorting genetic algorithm and is able to find much spread of solutions.

Discussion Related to RQ6 We measured the execution time of all the selected algorithms. We can conclude from Section 4.2.1 (the real-world case study) and Section 4.2.2 (the artificial problems) that (1+1) EA achieved the best time performance. The other selected search algorithms performed worse than RS in terms of time. SPEA2 is the worst among all. For our requirements assignment problem, with the number of stakeholders varying from 5 to 30, the number of requirements varying from 50 to 1000, the average time is 0.95 s for (1+1) EA (with the best time performance) and 10.92 s for SPEA2 (with the worst time performance). It is important to point it out that in our context, time is not an important factor to determine which algorithm should be used in Zen-Optimizer since the worst time performance is less than 11 s, which is practically acceptable.

Summary of the Results of Evaluating the Algorithms with HV We used jMetal (Durillo and Nebro 2011) to calculate the *HV* value of each Pareto front generated by each multi-objective algorithm at each run in the full-scale empirical study. Based on *HV*, we conducted the Vargha and Delaney statistics and the Wilcoxon signed-rank test at the significance level of 0.05 to compare the performance of each pair of the search algorithms. Results are consistent with what we observed when using mean fitness values. Results are provided in Appendix for reference.

5 The ZenReqOptimizer Methodology

In this paper, we proposed a methodology to assign requirements to different stakeholders by maximizing their familiarity to the assigned requirements and at the same time balancing the overall workload of each stakeholder. We developed the tool support (named Zen-ReqOptimizer) to realize this proposed method. This section presents a discussion about the proposed methodology and tool support. In Section 5.1 we introduce the architecture of the tool Zen-ReqOptimizer, Section 5.2 summarizes the application process of the Zen-ReqOptimizer methodology, and Section 5.3 discusses open issues.

5.1 Architecture of the Zen-ReqOptimizer Tool

The Zen-ReqOptimizer tool was implemented to realize the approach of requirements assignment and its architecture is shown in Fig. 16. For search algorithms, we used the existing jMetal library (Durillo and Nebro 2011), whereas ProR/RMF was used for creating the requirements repository, which contains the information about requirements such as importance and dependency to solve our optimization problem.

As described in Section 2.1, in order to use search-based techniques, we should have: 1) number of requirements, 2) number of available stakeholders, 3) each stakeholder's familiarity to each requirement, 4) information of requirements, including complexity, importance

and dependencies of these requirements. Below, we discuss how the Zen-ReqOptimizer tool automatically obtains these data from the requirements repository.

As shown in Fig. 16, one of the inputs of the Zen-ReqOptimizer tool is a set of requirements to be assigned, which is represented as ReqIF models (RMF 2008). ReqIF (ReqIF1.1 2013) is an OMG standard for exchanging requirements and a data model. The Requirements Modeling Framework (RMF 2008) is an Eclipse-based framework, which manages textual requirements (structured as ReqIF models). ReqIF allows users to exchange requirements with open source and commercial requirements management tools such as IBM Rational DOORS (2009) or PTC Integrity (2012). ProR (RMF 2008) is the user interface in RMF that allows users to work with ReqIF-based requirements. The Zen-ReqOptimizer tool is based on RMF and ProR and the tool supports the ReqIF 1.1 Standard (ReqIF1.1 2013).

In our requirements repository (ReqIF models), each requirement has six ReqIF attributes: *ID*, *Description*, *ReqFeature*, *DeployRequirement*, *IsOptional* and *Link*. It is possible to add new attributes when needed. Each requirement is assigned a unique *ID*. *Description* is used to describe the requirements in detail. Regarding the complexity of each requirement, the Zen-ReqOptimizer tool automatically generates the complexity of each requirement by counting the length of natural language sentences of the attribute *Description* of each requirement. In Section 3.1.4, requirements are classified in different ways and *ReqFeature, DeployRequirement* and *IsOptional* are used to reflect the classification. The importance of requirements can also be automatically obtained from the attributes *ReqFeature*, *DeployRequirements* and *IsOptional*. *ReqFeature* has three values: safety requirements, functional requirements, and other non-functional requirements. In the domain of CPSs, safety requirements are considered more important, than the functional requirements, which are considered more important than non-functional requirements. Notice that different heuristics can be defined for specific contexts. We only implemented one heuristic that is most suitable for our context. The attribute *DeployRequirements* has two values: subsea and topside. For the subsea oil and gas production domain, subsea requirements are considered more important than topside requirements, because low quality requirements related to subsea equipment, subsea control software and instruments often lead to catastrophic consequences such as oil spill. *IsOptional* represents that the requirement is optional or not. When requirements are optional, (e.g., "the master control system of the subsea control system provides an optional capability to remotely communicate with the off-site control center), we considered this requirement less important. *Link* describes requirements dependencies, which in the repository were manually established

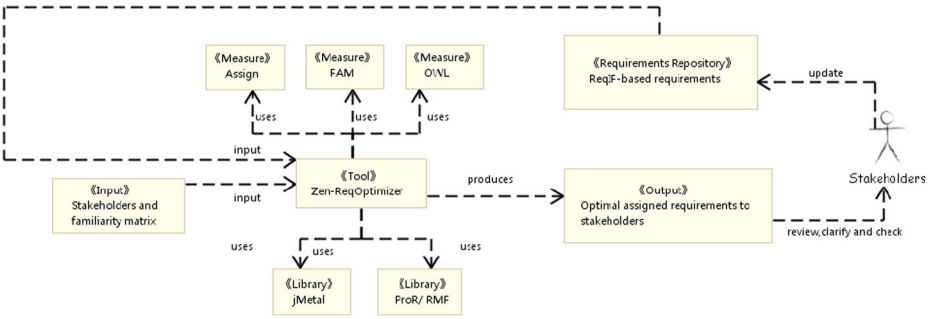

Fig. 16 Architecture for the Zen-ReqOptimizer tool

and maintained. Therefore, the value of a requirements' dependency can then be obtained automatically from the attribute *Link*.

In addition to a set of requirements to be assigned as a ReqIF models, the Zen-ReqOptimizer tool has the other input: a set of available stakeholders and their familiarity to each requirement. In the current implementation of the Zen-ReqOptimizer tool, the stakeholders' familiarity to the requirements is represented as a matrix with n_{st} rows and n_R columns (n_{st} is the number of stakeholders; n_R is the number of requirements), since each stakeholder has different familiarity to different requirements. The project manager is expected to provide familiarity data in our current implementation of the Zen-ReqOptimizer tool.

It is also very important to point out that in any non-trivial industrial context, requirements are often managed using requirements management tools and sometimes Excel spreadsheets. Reusing requirements is becoming an important indicator of the maturity of requirements engineering in an organization. A large portion of requirements can be reused across projects especially in the context of product line engineering. Therefore, it is worthwhile to make some effort to manually classify requirements into safety, functional and non-functional categories, construct requirements dependencies, and establish stakeholders' familiarity to requirements, since such captured information can be reused when the Zen-ReqOptimizer tool performs requirements assignment optimization.

By taking the established requirements repository as input, the Zen-ReqOptimizer tool automatically calculates values for measures (i.e., search objectives): *ASSIGN*, *FAM* and *OWL*, based on the definitions to these objectives (Section 2.2). Then the Zen-ReqOptimizer tool selects an algorithm to run based on the number of stakeholders and requirements of the problem, as we recommended in Section 4.2.3. All the multi-objective algorithms have been realized in jMetal (Durillo and Nebro 2011). jMetal is an open source software providing multi-objective optimization techniques, which is simple and easy to use. In the Zen-ReqOptimizer tool, jMetal is used as a library.

After running a selected search algorithm according to the problem, the Zen-ReqOptimizer tool generates assigned requirements to each stakeholder as output. When each stakeholder gets the requirements assigned to her/him, she/he should review, clarify and check their conformance to industry standards and government or other regulations. If there are any inconsistently with these standards, project managers should manage to make changes in these requirements and the requirements repository is updated accordingly.

5.2 The Zen-ReqOptimizer Methodology Application Process

Based on the empirical study, we can conclude that if a project manager is interested in a particular objective, then the Zen-ReqOptimizer tool uses (1+1) EA to obtain an optimal solution. Otherwise, if the overall fitness value is considered as the most important factor, then the Zen-ReqOptimizer tool relies on NSGA-II to return an optimal solution.

Figure 17 describes the process of our proposed approach Zen-ReqOptimizer to assist the project manager in accomplishing requirements allocation for stakeholders to review, clarify and check requirements. To apply our proposed approach in practice and to solve real problems such as the one arose in our real-world case study (Sections 3.1.4 and 3.1.5), we need to take into account some practical issues, such as how to collect data for running a selected algorithm: 1) number of requirements, 2) number of available stakeholders, 3) each stakeholder's familiarity to each requirement, 4) information of requirements, including complexity, importance and dependencies of these requirements. Zen-Optimizer can automatically collect the number of requirements and the information of these requirements

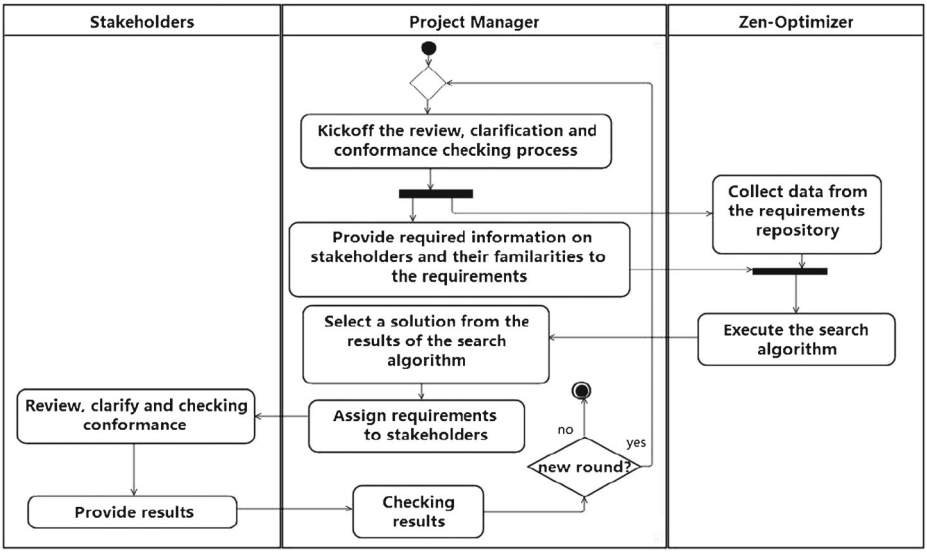

Fig. 17 Application scenario of the proposed approach as a UML activity diagram

from the requirements repository as discussed in Section 5.1. The number of available stakeholders and each stakeholder's familiarity to each requirement should be provided by the project manager. However, the information on the number of available stakeholders can often be obtained from a project management system and the information regarding the stakeholders' familiarity to requirements can be reused. When all required data are available, Zen-Optimizer searches for a set of optimal solutions, of which automated search process is guided by the three objectives (Section 2.2). The project manager then selects one solution from the optimal solutions returned by the search algorithm and assigns the requirements to the stakeholders. The stakeholders perform reviewing, clarification and conformance checking activities and provide feedback to the project manager. Based on the feedback, the project manager will decide whether there is a need to run Zen-Optimizer for another round such that modified requirements should be reviewed again. The process stops when all the requirements are being reviewed, clarified and checked and no more new issues are raised.

5.3 Open Issues

The first open issue in using the Zen-ReqOptimizer methodology is related to the collection of required data. It is easy to obtain the numbers of requirements and stakeholders. The characteristics of the requirements can be obtained from the requirement repository where the requirements to be assigned are managed, as we discussed in Section 5.1. Familiarity data can be obtained: directly from project managers, directly from each individual stakeholder, from historic data (e.g., on whether a particular stakeholder reviewed a specific type of requirement in the past), or a combination of the three. Which method(s) to apply depends on the current requirements engineering practice in a particular organization. For example, an organization might not even have a requirements repository, in which all the requirements are managed and characterized. If that is the case, it will be very difficult to facilitate any kind of automated collection of data. However, in practice, especially in

the context of developing large-scale CPSs, organizations often rely on requirements management tools and therefore it is realistic to make an assumption that an organization that applies the Zen-ReqOptimizer methodology has a requirement repository, from where the effort required to collect data for running search algorithms can be reduced.

Another concern is that if required information for running search algorithms is (partially) missing then the Zen-ReqOptimizer tool still works. For example, if the project manager cannot obtain any familiarity value the Zen-ReqOptimizer tool will make $FAM = 0$ in Formula (2) (Section 2.2) and updating FS in Formula (4), i.e., replacing 3 with 2 in the denominator, implying that FAM is not useful in terms of guiding search algorithms towards optimal solutions. The multi-objective problem then becomes a two-objective problem and can still be solved with the Zen-ReqOptimizer tool. As for complexity, dependency and importance, if we miss one or two of these three types of values, our approach can also take this situation as that all the values of missing information are the same. WL_j in Formula (3) (Section 2.2) will lose one or two corresponding parts and Formula (3) is still taken as one objective. Only when missing all of complexity, dependency and importance, Formula (3) becomes meaningless and the problem also becomes a two-objective problem. If we miss all the information, it is meaningless to solve the requirement assignment problem since there is no constraint and a requirement can be assigned to any stakeholder.

In the full-scale empirical evaluation, we created 120 artificial problems, which have a range of familiarity values consistent with that of our real-world case study. However, it is important to point out that if the artificial problems are with different characteristics in terms of familiarity values, experiment results might be different. However we do not expect that the conclusion (regarding which search algorithm is the best) will be different, unless the objectives and fitness function are changed. However, this entails further experiments in the future using real-world case studies having different characteristics than our current real-world case study.

Another concern is whether the objectives and the fitness value are sensitive to small differences of familiarity values To study this, we designed 24 artificial problems with the number of requirements ranging from 500 to 650 (4 values for the number of requirements) and the number of stakeholders ranging from 5 to 30 (6 values for the number of stakeholders). We simulated 9 situations with FM_{ij} values ranging from (0,1), (0,2), ... (0,9). Based on the results obtained for the 24 problems, we observed that *ASSIGN*, *FAM*, *OWL* and the overall fitness are not sensitive to small FM_{ij} differences. In our work, we used familiarity since this is the practice at our industry partner for the manual assignment. Of course, introducing another measure (e.g., ability of validating requirements) to the Zen-ReqOptimizer methodology is straightforward since one needs to define a new measure and treat it as another objective.

In practice, it is commonly observed that one stakeholder may have more time to spend on the requirements reviewing activity than others. When taking the availability of stakeholders into account, our approach of assigning requirements to stakeholders can be further refined. Moreover, considering that developing large scale CPSs needs to carefully review critical requirements (e.g., safety requirements), it is then required to have one requirement reviewed by more than one stakeholder. To accommodate these practical needs, the requirements assignment problem should be re-defined and therefore new empirical studies should be conducted to draw conclusion about which search algorithm should be applied to obtain optimal solutions. The Zen-ReqOptimizer methodology should therefore be updated to enable the collection of required data for running search algorithms.

Reviewing activities are often constrained by limited time and monetary budget. Therefore, prioritizing requirements to be reviewed based on for example, their importance, within

limited time and monetary budget, should be also supported by the Zen-ReqOptimizer methodology in the future.

6 Threats to Validity

Internal validity threats are related to internal factors that may impact the outcome of results (e.g., parameters of search algorithms, data source to compute fitness value) (Ali et al. 2010; Barros and Dias-Neto 2011). A likely threat to *internal validity* in our context is due to the fact that we only experimented with default parameter settings of all the algorithms. Parameter tuning may improve the performance of algorithms, however default parameters settings have shown to provide reasonable results (Arcuri and Fraser 2011).

A common *conclusion validity* threat in experiments using search algorithms is the effect of random variations inherited in search algorithms on results. To deal with this, we ran experiments 100 times to decrease the likelihood of obtaining the results by accident. Furthermore, we applied the Wilcoxon test to compare the algorithms' mean fitness values of 100 runs to determine the statistical significance of the results. We chose this test since it is appropriate for continuous data (Arcuri and Briand 2011), thus matching our situation. To determine the practical significance of the results obtained, we measured the effect size using the $\hat{A}_{12}$ values, which is recommended to be used in conjunction with the Wilcoxon test to better understand the results (Arcuri 2013).

Construct validity threats are concerned with the validity of the comparison measures for all the selected search algorithms. The most frequently observed threat was using objectives that have severe limitations, as they are not precise. For example, we use familiarity values between stakeholders and requirements to compute *FAM* which are possibly not an ideal way of doing so. Another case is that we use sentence length of requirement to represent its complexity, which is also not very precise. One of our future works will focus on proposing more precise way of defining *FAM* and *Complexity*. Another *construct validity* threat is related to the fairness of comparing the search algorithms. In our experiment, we used the same stopping criterion for all the search algorithms, i.e., number of fitness evaluations. We ran each algorithm for 5,000 evaluations to seek the best solution for requirements assignment. This criterion is a comparable measure across all the algorithms.

A common *external validity* threat in any type of experiments is related to generalization of results. We ran our experiments on just one real-world case study and results may not hold for other case studies. However, all empirical studies face the same external validity threat. In addition to the real-world case study, we also conducted an empirical evaluation of our proposed fitness function using 120 artificial problems of varying complexity and our results are consistent with the results of real-world case study. Our 120 artificial problems were similar to the real-world case and thus the results based on the artificial problems may not be generalized to problems with characteristics different than our real-world case study. In the future, we plan to conduct additional case studies with characteristics different than our real-world case study.

7 Related Work

In this paper, we applied search-based techniques to optimize requirements assignments in the context of developing large scale CPSs. The related work to this research stream includes

studies in requirements prioritization (Section 7.1) and applying search based techniques to address various requirements engineering optimization problems (Section 7.2).

7.1 Studies on Requirements Prioritization

When developing large scale CPSs, requirements engineering activities such as determining core requirements, selecting a subset of requirements, handling contradictory requirements and establishing relative importance of requirements are crucial to ensure the quality of the system to deliver and the productivity of the development process (Wohlin 2005). All these activities can be categorized into the research stream of requirements prioritization. In the literature, requirements are prioritized according to different criteria such as the importance of requirements (Karlsson et al. 1998) penalty for not fulfilling requirements (Herrmann and Daneva 2008), cost in terms of time required to realize requirements (Herrmann and Daneva 2008) and risk of taking requirements into products and releases (Herrmann and Daneva 2008). Various prioritization techniques have been applied to address requirements prioritization problems including the Analytical Hierarchy Process (AHP), the 100-dollar test, numerical assignment and ranking, which are discussed in the rest of the section.

AHP is a systematic decision-making method based on mathematics and psychology, which has been adapted for requirements prioritization through comparing all possible pairs of hierarchically classified requirements (Saaty 1987). The authors of Cortellessa et al. (2008) have studied that AHP is not suitable for large numbers of requirements since the total number of comparisons to perform with AHP are $n^*(n-1)/2$ (n is the total number of the requirements). When combing with numerical assignment methods requirements groups (along with dependencies) and transitivity of results/answers can drastically reduce the number of comparisons but in the end the number of comparisons can be still very high.

The 100-dollar test is a very straightforward prioritization technique, with which stakeholders are given 100 dollars or hours to distribute among requirements according to stakeholders' experience and domain knowledge The amount of money or number of hours assigned to a requirement reflects the stakeholders' preferences. Each stakeholder is free to put the whole amount given to her/him on only one requirement based on her/his preference, which might largely influence results There are two challenges with this technique: a large number of requirements to prioritize (Lehtola et al. 2004), a person performing the activity miscalculates the money or hours (Bradner 1997).

Numerical assignment is a most commonly used requirements prioritization technique suggested both in RFC 2119 (Bradner 1997) and IEEE Std. 830–1998 (Karlsson and Ryan 1997) and has been used to group requirements into different priority groups (Wohlin 2005) Each of such a priority group represents what stakeholders can relate requirements to their certain characteristics (e.g. critical, standard, optional) along with relative terms such as high, medium and low (Zhang et al. 2010) The challenge with this approach is that requirements being grouped into the same priority group have the same level of priority, which might not be preferred in the context expecting more accurate prioritization of requirements.

Ranking is based on an ordinal scale but requirements are ranked without ties in rank. Each requirement has a unique rank by using the bubble sort or binary search tree algorithms, which seem to be more suitable for a single stakeholder. The list of ranked requirements can be obtained in a variety of ways: e.g., using the bubble sort or binary search tree algorithms (Sommerville and Kotonya 1998).

Karlsson et al. made an overall evaluation of six methods (AHP, Hierarchy AHP, numerical assignment and three ranking methods: minimal spanning tree, binary search tree and

bubble sort) for prioritizing requirements with one real-world case study (Karlsson et al. 1998). The evaluation is from three aspects: inherent characteristics, objective measures of the methods and subjective measures by the authors. Inherent characteristics are the attributes of the methods, such as consistency indication indicating whether a prioritizing method is able to indicate consistency in the decision makers' judgment. Objective measures are observed during the evaluation such as the required number of decisions, total time consumption and time consumption per decision. Subjective measures are grades jointly assigned on an ordinal scale after the study by the authors, including the ease of use of the prioritizing method, reliability of results describing how reliable the results are judged to be and fault tolerance describing how insensitive the method is to judgmental errors. The evaluation results show that AHP was the most promising approach in terms of yielding the trust-worthiest results, but it may be problematic to scale up for projects with more than one hundred requirements.

In the context of developing large scale CPSs, hundreds and thousands of requirements have to be prioritized and many aspects above-mentioned, such as the importance of requirements, penalty for not fulfilling requirements (in aFtddition to cost of implementing requirements) should been taken into account, which requires a more scalable optimization solution; therefore search based optimization techniques can be very useful in this context, as we will discuss in Section 7.2.

7.2 Search-based Requirements Engineering

Search-based Software Engineering (SBSE) techniques have already been applied to many problems throughout the software engineering lifecycle Requirements engineering is a very important part of software engineering and SBSE has also been applied to optimize and prioritize requirements We summarize the related work of applying SBSE for requirements engineering in Table 18.

Bagnall et al. (2001) defined the problem of selecting an optimal next release as a search problems (NRP) and used three types of approaches: exact techniques, greedy algorithms and local search (hill climbing and simulated annealing) to find a high quality but possibly suboptimal solution to balance customer requests, resource constraints and requirements interdependencies. The evaluation was performed with 5 problems with different scales, from the smallest problem with 100 customers and 140 tasks to the largest problem with 500 customers and 3250 tasks. For the smallest problem, exact techniques proved to be sufficient. For the larger problems, simulated annealing found the best solutions with modest amount of computing. Feather and Menzies (2002) built an iterative requirements interaction model to select and optimize requirements using simulated annealing, which is the NASA-developed Defect Detection and Prevention (DDP) process and tool used for dealing with requirements, risks and risk mitigations. This novel iterative model for supporting execution, summarization and decision is to converge towards near-optimal attainment of requirements in large-scale requirements engineering contexts. Greer and Ruhe (2004) proposed an approach for allocating requirements to increments based on three means: 1) assessing and optimizing the degree to which the ordering conflicts with stakeholder priorities with technical precedence constraints; 2) balancing requirements and available resource for all the increments; and 3) continuously planning incremental software development using Genetic Algorithm. The proposed approach, based on the evaluation reported in the paper, was able to generate a small set of the most promising candidate solutions to support final decisions and derive potential release plans. Baker et al. (2006) used greedy and simulated annealing algorithms to solve NRP by ranking and selecting candidate software

Table 18 Related work on search-based requirements engineering

Reference	Topic	Objective	Technique	Case study
(Bagnall et al. 2001)	Requirements selection and optimization	Balance customers' requests, resource constraints and requirements interdependencies	Exact techniques, Greedy Algorithms and local search (Hill climbing and Simulated Annealing)	Five randomly generated problems with varying numbers of customers and requirements (customers/requirements: 100/140, 500/620, 500/1500, 500/3250, 1000/1500)
(Feather and Menzies 2002)	Requirements selection and optimization	Maximize reducing risks related to requirements by applying mitigations and minimize the sum total cost of performing mitigations	Simulated Annealing	One real world case study with 32 requirements, 69 risks and 99 risk mitigations
(Greer and Ruhe 2004)	Requirements selection and optimization	Minimize penalty defined as the degree of deviation of the monotonicity property between requirements, and maximize the total benefit	Genetic Algorithm	One real world case study (20 requirements and 5 stakeholders)
(Baker et al. 2006)	Requirements selection and optimization	Maximize the total cost sum of weights and minimize the total cost of the selected components	Simulated Annealing Greedy Algorithm	One real world case study (no details reported in the paper)
(Zhang et al. 2007)	Requirement satisfaction for NRP	Maximize value of selected requirements in terms of their importance, minimize cost to fulfill the selected requirement	Random Search, Single-Objective GA, Pareto GA and NSGA-II	Empirical study 1 with three problems (customer/requirement): 15/40,50/80, and 100/140 Empirical study 2 with two problems (customer/requirement): 100/25, 2/200

Table 18 (continued)

Reference	Topic	Objective	Technique	Case study
(Saliu and Ruhe 2007)	Requirements selection for NRP	Maximize value of the assignments of related features in terms of potential synergies in implementation and business values (i.e., degree of stakeholders satisfaction)	ε-Constraint algorithm	One real world case study (ReleasePlanner with 33 features, 5 types of effort-based resource)
(Finkelstein et al. 2008)	Fairness analysis in requirements assignments	Maximize each stakeholder's possible satisfaction of requirements in the next release	NSGA-II	Three problems: 1) randomly generated with 30 requirements and 5 customers, 2) from Motorola, and 30 from Greer and Ruhe (2004))
(Herrmann and Daneva 2008)	A systematic literature review on requirements prioritization	Benefit of requirements providing to the customer and cost of implementing requirements	/	/
(Zhang et al. 2008)	A literature review on search based requirements optimization	/	/	/
(Harman et al. 2009)	A review and classification of literature on SBSE	/	/	/
(Finkelstein et al. 2009)	Fairness analysis in requirements assignments	Maximize each stakeholder's possible satisfaction	NSGA-II Two-archive Random Search	Three data sets: 1) randomly generated artificial problems with 30 requirements and 5 customers, 2) from Motorola, and 3) from Greer and Ruhe (2004))

Table 18 (continued)

Reference	Topic	Objective	Technique	Case study
(Yuanyuan and Harman 2010)	Requirements interaction management	Minimize the cost of realizing selected requirements, maximize customers' satisfaction	NSGA-II	27 combination levels of random data sets generated using a pseudo random approach, according to distributions of scale range
(Durillo et al. 2011)	Requirements selection for NRP	Minimize the total cost of realizing selected requirements, maximize the total satisfaction	NSGA-II, MOCell and PAES	Six artificial problems (customer/requirement): 15/40, 50/80, 2/200, 100/20, 100/25, and 100/40); A case study from Motorola with 5 customers and 35 requirements
(Zhang et al. 2013)	Requirements interaction management	Minimize the cost of realizing selected requirements, maximize customers' satisfaction	NSGA-II	RALIC data sets: PointP data set with 143 requirements and 77 stakeholders; RankP data set with 143 requirements and 79 stakeholders; 27 data sets generated random as same as the one in Yuanyuan and Harman (2010)

components. The proposed approach was evaluated with a set of software components from the component base of a large telecommunications organization. The evaluation results show that the two algorithms convincingly significantly outperformed domain expert's judgment.

In Saliu and Ruhe (2007) and Zhang et al. (2007), two applications of SBSE to address the NRP problem were presented using multi-objective formulation Saliu and Ruhe presented a decision support approach that formulates the NRP problem as a bi-objective optimization problem and aims to optimize the value of release plans from both the business and implementation perspectives. The proposed technique aimed to assign as many solution-domain-coupled features in the same release as possible. They used the ILOG-CPLEX Optimizer (Hickey and Davis 2003) to implement an ε-Constraint algorithm that solves the biobjective optimization problem. A case study was conducted to evaluate the effectiveness and investigate the applicability of the proposed technique. Results show that this release plans generated Pareto-optimal solutions, which provided better support for decision-making than that method providing just a single solution. Zhang et al. however addressed the multi-objective NRP to select a set of requirements that maximize the total value and minimize the required cost. They conducted an empirical study on Random Search, Single-Objective GA, Pareto GA and NSGA-II. Results show that NSGA-II is well suited to the multi-objective NRP since NSGA-II performed the best and Pareto GA shared part of the front with NSGA-II. Finkelstein et al. (2008, 2009) applied a multi-objective optimization approach to investigate the trade-offs of various fairness between multiple customers with three data sets. In the work published in 2008, NSGA-II was used and results validated the approach. In the later work published in 2009, two-archive and RS were compared with NSGA-II and results show that NSGA-II significantly outperformed RS for all the data sets. For the random data set and Greer data set, NSGA-II and two-archive shared very similar performance. More recently, in Durillo et al. (2011) the NRP problem was addressed to get the highest possible number of customers satisfied while minimizing resource cost. In this empirical study of this work, three multi-objective algorithms (NSGA-II, MOCell and PAES) were evaluated to solve NRP. It was concluded that NSGA-II is the technique computing the highest number of optimal solutions. MOCell provided the product manager with the widest range of different solutions and PAES is the fastest technique (but with the least accurate results).

There are some papers about reviewing the literature in terms of applying SBSE techniques to address requirements optimization problems. In Zhang et al. (2008), Zhang et al. pointed out that SBSE techniques can be applied to address various requirements optimization problems during the requirements analysis phase and presented the challenges for Search Based Requirements Optimization. Herrmann and Daneva (2008) conducted a systematic literature review on addressing requirements prioritization challenges in terms of benefits that it provides to the customer and cost of implementing requirements using search techniques In Harman et al. (2009), Harman et al. presented a comprehensive analysis and review of trends of SBSE, with a section particularly dedicated to requirements engineering.

In Yuanyuan and Harman (2010) and Zhang et al. (2013) the requirements interaction problem was studied by evaluating archive-based multi-objective evolutionary algorithms, in terms of minimizing the cost of realizing selected requirements and maximizing customers' satisfaction. During the empirical study, "27 combination random data sets" (Zhang et al. 2010) was used to evaluate the search techniques. NSGA-II was capable of maintaining solution quality and diversity while respecting constraints imposed by the requirement interaction management.

Our work follows into the category of requirements assignment but with different objectives as compared to the related work, since we aim to maximize requirements engineers/domain experts' familiarity to their assigned requirements, while keeping the balance of their workload. In our empirical study, we used a real-world case study with 287 requirements, which is quite large as compared to what has been reported in the literature (as shown in Table 18), and 120 artificial problems of varying complexity with the number of requirements ranging from 50 to 1000 with an increment of 50 and the number of stakeholders ranging from 5 to 30 with an increment of 5. Moreover, we systematically compare six multi-objective search algorithms (CellDE, MOCell, NSGA-II, PAES, SMPSO, SPEA2) and one single-objective search algorithms ((1+1) EA). As reported in Section 4, the results of our empirical studies show that all the selected search algorithms significantly outperformed RS and NSGA-II obtained the best performance for *FS*. To compare with the related work, though NSGA-II in our context achieved the best performance for *FS*, but (1+1) EA achieved best performance for *ASSIGN*, *FAM* and *OWL*. NSGA-II's and (1+1) EA's performance degraded but not significantly with the increase of the number of requirements and stakeholders.

8 Conclusion and Future Work

Cyber-Physical Systems (CPSs) are large-scale integrated systems of systems, composed of control systems, embedded software, and communication networking. Such systems are commonly seen in the domains of such as Energy, Health Care, and Transportation. Due to the inherent complexity of such systems, a large portion of effort is allocated to requirements engineering as the quality of requirements has a significant impact on almost every single down steam activity of the development lifecycle of such a system.

Based on our experience of working with our industrial partners and a systematic domain analysis we conducted in an organization developing large-scale CPSs in the Energy domain, we observed that requirements engineering for developing such systems often involve multiple stakeholders from different organizations or different departments of the same organization. Such stakeholders need to review and clarify a large number of requirements and check their conformance to various standards and regulations for each single project. Such requirements have various extents of importance to an organization and complexity, complicated dependencies between each other. Reviewing and clarifying these requirements and checking their conformance to other documents require different expertise of stakeholders and various effort/workload required. Therefore, assigning proper requirements to stakeholders based on their familiarity to these requirements, while keeping the overall workload required, is an optimization problem. Without a systematic approach with tool support, it is infeasible to handle the assignment manually even by experienced project managers or domain experts.

To address the above-mentioned problem, first as a pilot experiment, we empirically studied three search algorithms: (1+1) Evolutionary Algorithm (EA), Genetic Algorithm, Alternating Variable Method, and used Random Search (RS) as a comparison baseline, along with a newly proposed fitness function. A quite large-scale real-world case study (with 135 requirements and 10 stakeholders) was conducted and results show that all the three search algorithms significantly outperformed random search. To determine which algorithm can handle more complex problem (with for example 1000 requirements and 30 stakeholders), we created 120 artificial problems with varying complexity. Our results show that

(1+1) EA gives the best results together with our proposed fitness function as compared to the rest of the algorithms.

Since our optimization problem is a multi-objective problem, we further investigated the following multi-objective algorithms: CellDE, MOCell, NSGA-II, PAES, SMPSO, SPEA2, together with (1+1) EA (the best in the pilot study) and RS (as the baseline) with our proposed fitness function. We used a real-world case study (with 287 requirements and 10 stakeholders) and 120 artificial problems of varying complexity. The results show that (1+1) EA obtained the best performance for each single objective and NSGA-II obtained the best performance for the overall fitness. NSGA-II has the ability to solve a wide range of problems without having their performance degraded significantly while (1+1) EA is not fit for problems with less than 250 requirements. Based on the results of the experiments, we suggest using (1+1) EA when a project manager is interested in a particular objective and using NSGA-II when she/he gives priority to the overall fitness value.

Our future work includes: (1) Refining *FAM* by taking into account the fact that a stakeholder has higher familiarity to a requirements if she/he has reviewed a similar kind of requirements in the past; (2) Defining new constraints such as the availability of stakeholders for performing requirements engineering activities; (3) Taking into account the situation that one requirement might need to be reviewed by more than stakeholders, especially when they are critical; and (4) Enhancing Zen-ReqOptimizer by implementing the requirements prioritization functionality.

Acknowledgments This work was supported by the Zen-Configurator project (No. 240024) and the MBT4CPS project (No. 240013) funded by the Research Council of Norway under the category of Young Research Talents of the FRIPO funding scheme. Tao Yue and Shaukat Ali are also supported by the EU Horizon 2020 project U-Test (http://www.u-test.eu/), the MBE-CR (An Innovative Approach for Longstanding Development and Maintenance of the Automated Cancer Registry System, No. 239063) and the Certus SFI (http://certus-sfi.no/). It was also supported in part by a grant from the National Natural Science Foundation of China (No. 61370058, No. 61170087).

Appendix

In this section, we present results of evaluating the multi-objective search algorithms using Hyper Volume (*HV)*—a commonly used quality indicator with multi-objective search algorithms. The results are for the full-scale empirical study. Recall that each search algorithm was run 100 times for each problem and generated a Pareto front at each run. We calculated the *HV* value of each Pareto front.

Real-world Case Study

We obtained 100 *HV* values for the real world case study for each algorithm. We conducted the Wilcoxon signed-rank test at the significance level of 0.05 for the *HV* values and results are presented in Table 19. Algorithms with larger values of *HV* are desirable; if $\hat{A}_{12}$ is greater than 0.5, it means that algorithm *A* has a higher chance of obtaining a higher *HV* than *B*. An $\hat{A}_{12}$ value less than 0.5 means the Algorithm A has lesser chance of obtaining a higher value of *HV* than B. From Table 19, we can conclude that in terms of *HV*, NSGA-II obtains significant better results than SPEA2, followed by MOCell, PAES and SMPSO. CellDE obtains the worst. These results are consistent with what we observed from Section 4.2.1.

Table 19 Results of the Vargha and Delaney statistics and the Wilcoxon signed-rank test at the significance level of 0.05 (in terms of *HV*) – real-world case study

Pair of algorithms (A vs B)		$\hat{A}_{12}$	p-value
CellDE	MOCell	0.0014	<**0.0001**
	NSGA-II	0.0000	<**0.0001**
	PAES	0.1145	<**0.0001**
	SPEA2	0.0000	<**0.0001**
	SMPSO	0.3588	**0.0002**
MOCell	NSGA-II	0.0122	<**0.0001**
	PAES	0.7941	<**0.0001**
	SPEA2	0.0796	<**0.0001**
	SMPSO	0.9992	<**0.0001**
NSGA-II	PAES	0.9981	<**0.0001**
	SPEA2	0.8005	<**0.0001**
	SMPSO	1.0000	<**0.0001**
PAES	SPEA2	0.0205	<**0.0001**
	SMPSO	0.8514	<**0.0001**
SPEA2	SMPSO	1	<**0.0001**

Artificial Problems

For each of the 120 artificial problems, we conducted the one sample Wilcoxon signed-rank test to compare each pair of the algorithms in terms of *HV*. Table 20 summarizes the

Table 20 Results of the Vargha and Delaney statistical test (in terms of *HV*) - 120 artificial problems (without/with the Wilcoxon signed-rank test)

Pair of algorithms (A vs B)		A>B	A<B	A=B
CellDE	MOCell	1/0	119/118	0/2
	NSGA-II	0/0	120/120	0/0
	PAES	6/6	114/109	0/5
	SPEA2	0/0	120/120	0/0
	SMPSO	72/55	48/30	0/35
MOCell	NSGA-II	0/0	120/120	0/0
	PAES	52/48	68/65	0/7
	SPEA2	5/1	115/103	0/16
	SMPSO	120/120	0/0	0/0
NSGA-II	PAES	108/100	12/8	0/12
	SPEA2	120/115	0/0	0/5
	SMPSO	120/120	0/0	0/0
PAES	SPEA2	44/40	76/72	0/8
	SMPSO	114/111	6/3	0/6
SPEA2	SMPSO	120/120	0/0	0/0

results of the Vargha and Delaney statistic test (with or without the Wilcoxon signed-rank test applied). Without the Wilcoxon signed-rank test, $A > B$ means the number of problems (out of 120) that A is better than B for obtaining a better solution; $A < B$ means the number of problems (out of 120) that A is worse than B for obtaining a better solution; and $A = B$ means the number of problems for which there are no differences between A and B.

Results show that for *HV*, NSGA-II performed significantly better than SPEA2 for 115 problems and SPEA2 performed significantly better than PAES for 72 problem. PAES performed significantly better than MOCell for 65 problems. MOCell performed significantly better than CellDE for 118 problems. CellDE performed significantly better than SMPSO for 55 problems and significantly worse than SMPSO for 30 problems. SMPSO and CellDE had no significant difference for 35 problems. Based on the results, we can conclude that in terms of *HV*, NSGA-II achieves the best performance, followed by SPEA2, PAES and MOCell. CellDE and SMPSO share similar performance, where both performed worse than the other search algorithms. In terms of *HV*, for all the 120 artificial problems, results are similar to what have been reported in Section 4.2.2.

References

Ali S, Briand LC, Hemmati H, Panesar-Walawege RK (2010) A systematic review of the application and empirical investigation of search-based test case generation. IEEE Trans Softw Eng 36:742–762. doi:10.1109/TSE.2009.52

Arcuri A (2013) It really does matter how you normalize the branch distance in search – based software testing. Verification and Reliability 23:119–147

Arcuri A, Briand L (2011) A practical guide for using statistical tests to assess randomized algorithms in software engineering. Paper presented at the 33rd international conference on software engineering (ICSE), Hawaii, 21–28 May 2011

Arcuri A, Fraser G (2011) On parameter tuning in search based software engineering. Lecture Notes in: Computer Science, pp 33–47

Bagnall AJ, Rayward-Smith VJ, Whittley I (2001) The next release problem. Inf Softw Technol 43:883–890

Bai Y, Bai Q (2012) Subsea engineering handbook. Gulf Professional Publishing, USA

Baker P, Harman M, Steinhöfel K, Skaliotis A (2006) Search based approaches to component selection and prioritization for the next release problem. Paper presented at the 22nd IEEE international conference on software maintenance, Philadelphia, Pennsylvania, 24–27 September 2006

Barros MO, Dias-Neto AC (2011) Threats to validity in search-based software engineering empirical studies. RelaTe-DIA, 2011, 5(1)

Bradner S (1997) Key words for use in RFCs to indicate requirement levels. doi:10.17487/RFC2119

Brownlee J (2012) Clever algorithms: nature-inspired programming recipes. lulu.com; 1st edn

Cortellessa V, Crnkovic I, Marinelli F, Potena P (2008) Experimenting the automated selection of COTS components based on cost and system requirements. J UCS 14:1228–1255

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II evolutionary computation. IEEE Trans 6:182–197. doi:10.1109/4235.996017

Droste S, Jansen T, Wegener I (2002) On the analysis of the (1+1) evolutionary algorithm. Theor Comput Sci 276:51–81. doi:10.1016/S0304-3975(01)00182-7

Durillo J, Nebro A, Luna F, Alba E (2008) Solving three-objective optimization problems using a new hybrid cellular genetic algorithm. In: Rudolph G, Jansen T, Lucas S, Poloni C, Beume N (eds) Parallel problem solving from nature – PPSN X. Lecture Notes in Computer Science, vol 5199. Springer, Berlin Heidelberg, pp 661–670, DOI doi:10.1007/978-3-540-87700-4_66

Durillo J, Zhang Y, Alba E, Harman M, Nebro A (2011) A study of the Bi-objective next release problem. Empir Softw Eng 16:29–60. doi:10.1007/s10664-010-9147-3

Durillo JJ, Nebro AJ (2011) jMetal: a java framework for multi-objective optimization. Adv Eng Softw 42:760–771. doi:10.1016/j.advengsoft.2011.05.014

Feather MS, Menzies T (2002) Converging on the optimal attainment of requirements. In: International conference on requirements engineering. IEEE Joint, 2002, pp 263–270

Finkelstein A, Harman M, Mansouri SA, Ren J, Zhang Y (2008) Fairness analysis in requirements assignments. In: International requirements engineering, IEEE, pp 115–124

Finkelstein A, Harman M, Mansouri SA, Ren J, Zhang Y (2009) A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. Requir Eng 14(4):231–245

Greer D, Ruhe G (2004) Software release planning: an evolutionary and iterative approach. Inf Softw Technol 46(4):243–253

Harman M, Mansouri SA, Zhang Y (2009) Search based software engineering: a comprehensive analysis and review of trends techniques and applications. Department of Computer Science, King College London, Tech Rep TR-09-03

Herrmann A, Daneva M (2008) Requirements Prioritization Based on Benefit and Cost Prediction: an agenda for future research. In: International requirements engineering, IEEE, pp 125–134

Hickey AM, Davis AM (2003) Elicitation technique selection: How do experts do it? In: 11th IEEE international requirements engineering conference, IEEE, pp 169–178

Karlsson J, Ryan K (1997) A cost-value approach for prioritizing requirements software. IEEE 14(4):67–74

Karlsson J, Wohlin C, Regnell B (1998) An evaluation of methods for prioritizing software requirements. Inf Softw Technol 39(14):939–947

Knowles JD, Corne DW (2000) Approximating the nondominated front using the Pareto archived evolution strategy. Evol Comput 8(2):149–172. doi:10.1162/106365600568167

Konak A, Coit DW, Smith AE (2006) Multi-objective optimization using genetic algorithms: a tutorial. Reliab Eng Syst Saf 91(9):992–1007. doi:10.1016/j.ress.2005.11.018

Korel B (1990) Automated software test data generation. IEEE Trans Softw Eng 16:870–879. doi:10.1109/32.57624

Lehtola L, Kauppinen M, Kujala S (2004) Requirements prioritization challenges in practice. In: Product focused software process improvement. Springer, Berlin, pp 497–508

Nebro A, Durillo J, Luna F, Dorronsoro B, Alba E (2007) Design issues in a multiobjective cellular genetic algorithm. In: Evolutionary multi-criterion optimization. Springer, Berlin Heidelberg, pp 126–140

Nebro AJ, Durillo JJ, Garcia-Nieto J, Coello Coello CA, Luna F, Alba E (2009) SMPSO: a new PSO-based metaheuristic for multi-objective optimization. In: IEEE symposium on computational intelligence in multi-criteria decision-making, March 30 2009–April 2 2009, pp 66–73. doi:10.1109/MCDM.2009.4938830

PTC Integrity (2012) http://www.ptc.com/product/integrity

Rational DOORS (2009) http://www.ibm.com/developerworks/downloads/r/doorswebaccess/

ReqIF1.1 (2013) Document formal/2013-10-01. Technical report, OMG

RMF (2008) http://download.eclipse.org/rmf/documentation/rmf-latex/main.html

Saaty RW (1987) The analytic hierarchy process—what it is and how it is used. Math Model 9(3):161–176

Saliu MO, Ruhe G (2007) Bi-objective release planning for evolving software systems. In: Proceedings of the 16th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering, ACM, pp 105–114

Sheskin DJ (2007) Handbook of parametric and nonparametric statistical procedures. Chapman and Hall/CRC

Sommerville I, Kotonya G (1998) Requirements engineering: processes and techniques. Wiley, New York

Wohlin C (2005) Engineering and managing software requirements. Springer, Berlin

Yuanyuan Z, Harman M (2010) Search based optimization of requirements interaction management. In: 2nd international symposium on search based software engineering, 7–9 September 2010, pp 47–56. doi:10.1109/SSBSE.2010.16

Yue T, Ali S (2014) Applying search algorithms for optimizing stakeholders familiarity and balancing workload in requirements assignment. Paper presented at the proceedings of the 2014 conference on genetic and evolutionary computation, Vancouver, BC

Zhang Y, Alba E, Durillo JJ, Eldh S, Harman M (2010) Today/Future importance analysis. In: Proceedings of the 12th annual conference on genetic and evolutionary computation. ACM, pp 1357–1364

Zhang Y, Finkelstein A, Harman M (2008) Search based requirements optimisation: existing work and challenges. In: Requirements engineering: foundation for software quality. Springer, Berlin Heidelberg, pp 88–94

Zhang Y, Harman M, Lim SL (2013) Empirical evaluation of search based requirements interaction management. Inf Softw Technol 55(1):126–152. doi:10.1016/j.infsof.2012.03.007

Zhang Y, Harman M, Mansouri SA (2007) The multi-objective next release problem. In: Proceedings of the 9th annual conference on genetic and evolutionary computation. ACM, pp 1129–1137

Zitzler E, Laumanns M, Thiele L (2001) SPEA2: improving the strength Pareto evolutionary algorithm. Paper presented at the the EUROGEN 2001-evolutionary methods for design, optimization and control with applications to industrial problems

Yan Li obtained her bachelor for computer science and technology from Center South University, Changsha, China and her master degree for computer application technology from Fuzhou University, Fuzhou, China. She is a PhD candidate at Beihang University. Her main research interests are: product line engineering, search-based software engineering and requirements engineering.

Tao Yue is now a senior research scientist of Simula Research Laboratory, Oslo, Norway, where she is leading the expertise area of Model Based Engineering (MBE). She is also affiliated to University of Oslo as an associate professor. She has received the PhD degree in the Department of Systems and Computer Engineering at Carleton University, Ottawa, Canada in 2010. Before that, she was an aviation engineer and system engineer for seven years. She has around 16 years of experience of conducting industry-oriented research with a focus on MBE in various application domains such as Avionics, Maritime and Energy, and Communications in several countries including Canada, Norway, and China. Her present research area is software engineering, with specific interested in requirements engineering, requirements-based testing, model-based product line engineering, model-based system engineering, modelbased testing, search-based software engineering and empirical software engineering. Dr. Yue has been on the program and organization committees of many international, IEEE and ACM conferences such as ACM/IEEE International Conference on Model Driven Engineering, Languages and Systems, International Conference of Requirements Engineering, and International Systems and Software Product Line Conference. She is also on the editorial board of on the Editorial Board of Empirical Software Engineering Journal. She is PI and CO-PI of several national and international research projects.

Shaukat Ali is currently a senior research scientist in the Software Engineering department, Simula Research Laboratory, Norway. He has been affiliated to Simula Research Lab since 2007. He has been involved in many industrial and research projects related to Model-based Testing (MBT) and Search-Based Software Engineering since 2003. He has experience of working in several industries and academic research groups in many countries including UK, Canada, Norway, and Pakistan. Shaukat has been on the program committees of several international conferences and also served as a reviewer for several software engineering journals.

Li Zhang received her BS, MS, and PhD from the School of Computer Science and Engineering, Beihang University, China in 1989, 1992, and 1996, respectively. She is now a professor in the School of Computer Science and Engineering, Beihang University, China. She is a committee member of Software Engineering in China Computer Federation (CCF), committee member of education in CCF, and a committee member of computer applications in the Chinese Society of Astronautics (CSA). She is interested in software engineering, software architecture modeling, and system reliability analysis.