Information Sciences Letters

Volume 6
Issue 1 *Jan. 2017*

Article 2

2021

An Effective Method of Systems Requirement Optimization Based on Genetic Algorithms

M. H. Marghny
Department of Computer Science, Faculty of Computer and Information, Assiut University, Assiut, Egypt,
dralwathiq2012@gmail.com

H. M. El-Hawary
Department of Mathematics, Faculty of Science, Assiut University, Assiut, Egypt,
dralwathiq2012@gmail.com

Wathiq H. Dukhan
Department of Mathematics, Faculty of Science, Assiut University, Assiut, Egypt,
dralwathiq2012@gmail.com

Information Sciences Letters
An International Journal

http://dx.doi.org/10.12785/isl/060102

An Effective Method of Systems Requirement Optimization Based on Genetic Algorithms

M. H. Marghny [1], *H. M. El-Hawary*[2] *and Wathiq H. Dukhan*[*2,*]

[1] Department of Computer Science, Faculty of Computer and Information, Assiut University, Assiut, Egypt
[2] Department of Mathematics, Faculty of Science, Assiut University, Assiut, Egypt

Received: 2 Nov. 2016, Revised: 19 Dec. 2016, Accepted: 23 Dec. 2016
Published online: 1 Jan. 2017

Abstract: Requirements engineering is the first step of software development process and it is one of the main concerns of software engineers. System requirements selection is the engineering process to select an optimal set of system requirements for implementation in the next system of the software from many requirements proposed by the customers on condition that budget and customer satisfaction are being balanced. This NP-hard problem is an important issue involving several conflicting objectives that have to be processed by software companies when developing new software systems. Software systems have to perform their function within resource constraints, but they also have to cover the largest number of customer requirements. Additionally, in real life problem, the requirements selection process suffers from complication due to interactions and other constrictions.
In this paper, meta-heuristic techniques have been applied along with adapted/modified multi-objective function which has been successfully applied to several real cases of the problem. The system requirements selection problem has been formulated as a multi-objective optimization problem with two objectives that minimizes the total system's development cost and maximizes customer's satisfaction totality. Moreover, GA has been adapted to solve real cases of the problem and tested with case studies on two real datasets that have been carried out to demonstrate and prove the effectiveness of the multi-objective proposed approach and the obtained experimental results show that the updated GA can effectively generate high quality solutions and performs better than other pertinent algorithms previously published in the literature under a set of public datasets.

Keywords: System requirements selection, Software engineering, Requirements engineering, Genetic algorithm.

1 -Introduction

The effectiveness of software system, it is well-known that it was totally measured by how well both the needs of its stakeholders and its environment were being met [1,2]; and those needs were included in the system requirements. In this regard, the requirements engineering (RE) could be defined as the process by which the requirements are determined; thus, successful RE included the following aspects: understanding the different needs of users, customers and other stakeholders; understanding the contexts in which the system would be developed; modeling; analyzing; negotiating; documenting the requirements of the stakeholders; making sure that the documented requirements were consistent with the negotiated ones; and managing the evolution of the requirements. In this context, back in the 1990s, requirements engineering was considered a major element of the software engineering process. In addition, the early stage of the process was very critical, as its resulted decisions were both crucial and difficult; and that is due to the inadequacy, vagueness and dynamic changing of the available information [3]. Furthermore, these decisions had a long-term impact on the software system [4]. Therefore, the prioritization of the requirements was a decision-making process that enabled systems managers to concentrate on the deliverables that added most value to a system's outcome. In addition, the software system management made good use of this process; and that is in order to identify the software system requirements which should be included in a certain release. Taking into consideration the computational complexity of the problems within software engineering, we could formulate them as optimization problems; hence, they could be solved by using the meta-heuristic search techniques.

* Corresponding author e-mail: dralwathiq2012@gmail.com

As for this field in general, many works have focused on the problem of determining which features or requirements should be covered by the software system that was being constructed. Patrik and Svahnberg [5] mentioned that when a software system was described by a large number of requirements in most cases, we could not fulfill all requirements within the resource constraints. Therefore, they should be limited in some way or another. The solution of these limitations was conducted by prioritizing the candidate requirements and the selection of the best subset of requirements as per the available resources [6]. In addition, the multi-objective optimization could also help software developers when deciding which subset of requirements supposedly directed to the next development phases; and that is in case of facing contradictory goals [7]. They mainly aimed to combine the computational intelligence and the knowledge experience of human experts along with the idea of having a better selection of requirements than that obtained through the judgment of expert developer's alone. This objective was accomplished by embedding artificial intelligence techniques into the management of requirements as a new functionality [8]; and that is in order to take advantage of the meta-heuristic techniques during the execution of the development phases with regard to any software system.

This current paper is divided as follows: **Section** 2 draws attention to the related works. **Section** 3 describes the methodology. **Section** 4 displays a detailed explanation of the different materials and methods. **Section** 5 presents the experiments and their results, and it provides an in-depth analysis and a description of the obtained results. **Section** 6 summarizes the conclusion and the suggestions for future works.

2 -Related Work

In light of the above introduction, we could notice that the optimization of requirements was still an NP-hard problem [10], consisted of selecting an optimal set of requirements that would be developed for the software system; such requirements were always selected for the sake of maximizing the customer's satisfaction and minimizing the development costs. There were two conflicting objectives evaluated within the problem, thus, both of them had to be considerably balanced while solutions were still being found. In the previous studies, Karlsson [11] introduced two methods of selecting and prioritizing the software requirements, specifically the Analytical Hierarchy Process (AHP) and the Quality Function Deployment (QFD). In QFD, the priority was given to the requirements according to the ordinal scale; while in AHP, the requirements were classified by a pair cost-value. Nonetheless, the interdependencies of the requirements were not supported in both two methods, which were real current needs; in addition, they also suffered scalability issues, as a large number of comparisons were to be conducted when the system scale wase increased.

Bagnall et al. [6] were first to mention the problem of selecting the system requirements as a Next Release Problem(NRP). Furthermore, due to the inherent nature of the problem, it was formulated into a multi-objective version, and mainly experimented with MoCell and PAES [12]. Moreover, another relevant field contributed to the issue in question, which is the scientific field of Search-based Software Engineering, in which the search-based optimization algorithms are developed in order to tackle problems in software engineering [13]. Nonetheless, most of the published methods were single-objective evolutionary algorithms seeking to unite the objectives through the use of an aggregation function [14,15]. In all cases, no one looked at the interactions produced between the requirements. In addition, the formulation of single-objective had the inconvenience of making a biased search of the solution domain, as the objectives were to be carefully accumulated in some way such as a weighted sum of objectives.

Recently, the selection of system requirements as an NRP was formulated as a multi-objective optimization problem (MOOP) [16]; thus, each objective was treated independently of the others disregarding the aggregation function and the problem constraint such as the interactions among the requirements or the cost limitations. On the other hand, Feather and Menzies [17] applied the Simulated Annealing and an iterative model for the selection of requirements and the optimization problem, which is called the Defect Detection and Prevention (DDP). The success of this model was illustrated in a pilot study of a real-world instance of requirements interaction model. In addition, Feather et al. [18] summarized the techniques of visualization used to express the status of the requirements, including the Pareto Fronts dictated by the Simulated Annealing. Furthermore, Harman et al. [19] formulated the component selection and the prioritization problem as a feature subset selection problem; and to create the optimal solutions, they applied several search-based approaches. Jalali et al. [20] also took into consideration the requirements optimization, and proposed the KEYS technique, in order to identify the solutions and their key factors simultaneously. On the other hand, in order to solve the NRP, Sagrado et al. [21,22] applied three different meta-heuristic search techniques as follows: Genetic Algorithm, Simulated Annealing and Ant Colony Optimization (ACO). Moreover, Tonella et al. [23,24] proposed an interactive requirements prioritization through the use of an Interactive Genetic Algorithm (IGA) that included incremental knowledge acquisition and combined it with the already existing constraints of dependencies and priorities. Their experimentation was conducted on a real case study by comparing IGA with the state-of-art interactive prioritization technique and the Incomplete Analytic Hierarchy Process (IAHP). In terms of effectiveness, efficiency and robustness, the results

showed that IGA outperformed IAHP. Finally, [34] the multi-objective evolutionary algorithm was introduced to solve such problems successfully; therefore, several real instances were considered.

In this current paper, the researcher presents the non-dominated sorting genetic algorithm with Pareto tournament for the multi-objective optimization (NSGA-IIPT) method, thus adapting the algorithm to work with the problem formulation in which different types of requirements' interactions and cost constraints are considered. The proposed approach in this paper searches for high quality sets of solutions within a given development cost bound, balancing the customers' priorities and the cost requirements.

3 -Methodology

3.1 -Selection of Requirements

Requirements gathering is the process of collecting the needs that must be met by the system under development. In this regard, the tasks related to the requirements stage have a very different nature from those related to design or coding, as the requirements tasks are closer to the problem space such as gathering or negotiating the requirements. In addition, the requirements also have a strong connection with customers. Other aspects, such as the requirements specification, are concerned with translating the requirements into specific modeling languages. At the end, the tasks of requirements management could be defined as a number of actions executed by software engineers in relation to decision making, with regard to the quality, traceability, risk or viability of the requirements. The selection of requirements is one of these management tasks; that is to say, it is the process of determining which requirements (from those gathered with customers) should be included in a system in light of the available resources [9].

3.2 -Multi-Objective System Requirements Selection Problem -Mathematical Modeling

When we face the problem of selecting a set to be developed requirements, it is assumed that there is a set of customers, $C = \{c_1, c_2, \ldots, c_m\}$ and a set of possible system requirements, $R = \{r_1, r_2, \ldots, r_n\}$. The set R is the main list of all requirements agreed upon with customers and desired in a software system. Also, there is a weight w_i associated to each customer c_i that indicates his/her importance level for the system. The set $W = \{w_1, w_2, \ldots, w_m\}$ contains all these weights. Each customer will assign a value v_{ij} to each requirement in R , that represents the degree of priority that customer c_i assigns to requirement r_j to be included in the software system. Value $v_{ij} > 0$. A zero value for v_{ij} means that the customer c_i has not suggested the requirement r_j . All these values v_{ij} are collected in priority matrix $V_{m\times n}$.

$$V = \begin{pmatrix} V_{1,1} & V_{1,2} & \ldots & V_{1,n} \\ V_{2,1} & V_{2,2} & \ldots & V_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ V_{m,1} & V_{m,2} & \ldots & V_{m,n} \end{pmatrix}.$$

For a given a requirement r_i its score s_j is defined as the weighted sum of its values as follows:

$$S_j = \sum_{i=1}^{m} w_i * v_{ij} \quad , \tag{1}$$

The set of scores will be referred to as $S = \{s_1, s_2, \ldots, s_n\}$. Besides, each requirement r_j in R has an associated $cost_j$ measuring the system development cost. All these cost define the set of costs, $cost = \{cost_1, cost_2, \ldots, cost_n\}$. The problem consists in selecting a subset of requirements $X \subseteq R$, which maximizes total satisfaction of the customers and minimizes the total cost needed to develop it, within the resources (cost) limit B established for the system and preserving dependency interactions. The requirements interaction considers the interactions among the requirements. In this study, we take into account two interactions of different type:

Implication interactions $r_i \Rightarrow r_j$ (if $r_i \in X$, r_j should also be in X) and combination interactions $r_i \bigoplus r_j$ (if $r_i \in X$, r_j should belong to X, and vice versa). This problem can be stated formally as the optimization problem in order to formulate the fitness function.

Total satisfaction requirement r_i can be calculated from equation (1). The two objectives to system requirements can be formulated as:

Maximize Satisfaction customer:

$$Maximize \quad S(x) = \sum_{i=1}^{n} S_i * x_i \quad , \tag{2}$$

and minimize total cost requirements:

$$Minimize \quad E(x) = \sum_{i=1}^{n} Cost_i * x_i \quad . \tag{3}$$

. The vector X is a solution vector that indicates the requirements that are to be included in the system $X = \{x_1, x_2, \ldots, x_n\} : \overrightarrow{X} \subseteq \overrightarrow{R}, \quad x_i \in \{0,1\}$ in this vector, x_i is 1 if the requirement is selected for inclusion in the system, 0 otherwise.

Subject to

$$\begin{cases} E(x) \leq B : B & \text{is budget;} \\ \text{interaction constraints.} \end{cases}$$

3.3 -Objective Function

In brief taking the objective function of our proposed method can be formulated as follows

$$\text{Maximize} \quad S(x) = \sum_{i=1}^{n} S_i * x_i \quad ,$$
$$\text{Minimize} \quad E(x) = \sum_{i=1}^{n} Cost_i * x_i \quad ,$$

$$Subjectto \begin{cases} E(x) \leq B : B & \text{is budget;} \\ \text{interaction constraints,} \end{cases}$$

where $S_j = \sum_{i=1}^{m} w_i * v_{ij}$.

Notice that both of these two values are normalized between 0 and 1. In the formula below, we adopted this normalization function and it has proven to be more robust than other normalization functions. The optimization problem can then be mathematically restructured as a multi-objective optimization problem as follows:

$$Minimize \quad S(x) = \frac{\sum_{i=1}^{n} S_i - \sum_{i=1}^{n} S_i * x_i}{\sum_{i=1}^{n} S_i}, \quad S(x) \in [0,1], \tag{4}$$

$$Minimize \quad E(x) = \frac{\sum_{i=1}^{n} Cost_i * x_i}{\sum_{i=1}^{n} Cost_i}, \quad E(x) \in [0,1], \tag{5}$$

Subject to

$$\begin{cases} E(x) \leq B : B & \text{percent is budget;} \\ \text{interaction constraints.} \end{cases}$$

Where cost range from 0 to $\sum Cost_i$, satisfaction range from 0 to $\sum S_i$, $weight_j = \frac{W_i}{\sum_{i=1}^{m} w_i}$, $j = 1,2,\ldots,m$, and $S_j = \sum_{i=1}^{m} weight_i * v_{ij}$.

4 -Materials and Methods

In this section, the researcher describes the developed meta-heuristic technique, in order to manage the problem of requirements selection; as well as a brief description of the non-dominated sorting genetic algorithm with Pareto tournament for the multi-objective optimization (NSGA-IIPT), paying extra attention to the different phases involved in the selection of a set of requirements. However, we shall first present the encoding of the solutions of the proposed approach.

4.1 -Solution/ Individuals Representation

The solution is encoded with the purpose of giving all the information that is required to represent and evaluate solutions for the multi-objective system requirement problem (MOSRP). Figure1 shows the representation of the individual used by the multi-objective meta-heuristic planned during this paper. The individuals express solutions which are the objects of the evolutionary algorithms, so it is necessary to have a good design of fast processing as the genetic operators interact with them. The solutions are encoded as a requirement vector of Boolean (X) of n positions. Each position in X indicates whether or not the requirement j is chosen for the system. If so, that particular position is equal to 1, otherwise, it is 0. The overall number of requirements in X is also saved within the data structure that maintains the individual $(X_{RNumber})$. Finally, the overall quality for the solution is

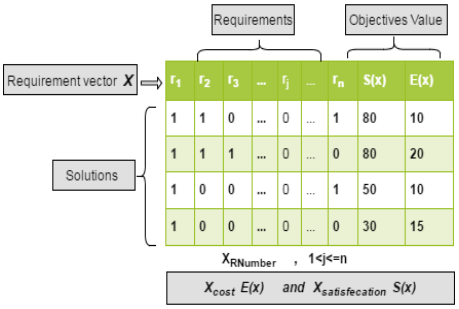


Fig. 1: Individual representation

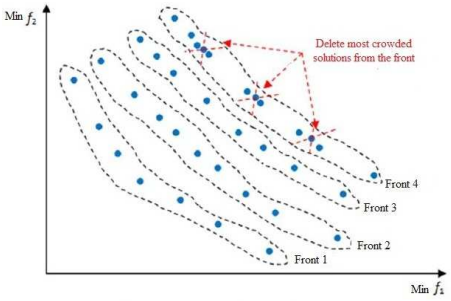


Fig. 2: NSGA-IIPT Front and crowding distance

evaluated through two objectives (overall satisfaction $S(X)$ and cost $E(X)$). These values are calculated as explained in Section 3.2, and are additionally saved within the data structure (see Figure 1)

4.2 -Non-dominated Sorting Genetic Algorithm With Pareto Tournament for Multi-Objective Optimization (NSGA-IIPT)

Genetic algorithms can be useful tool to tackle multi-objective problems by assigning a specific weight to every objective function and changing the multi-objective problem to a single objective problem, e.g., using a scalar objective function [25]. These algorithms are then known as weight-based GAs. NSGA-II varies from simple genetic algorithm which is considered as the most

commonly used multi-objective search algorithm that outputs a set of non-dominated solutions for multiple objectives according to the pareto dominance theory [24, 26, 27, 28]. NSGA-II first sorts the population into several non-dominated fronts by employing a ranking algorithm shown in Figure 2. Then, NSGA-II selects individual solutions from these pareto front or non-dominated solutions and generates new populations by applying selection, crossover and mutation operators. Moreover, NSGA-II defines an indicator known as crowding distance to measure the distance between the individual solutions and the others in the population [26]. If two individual solutions exist in the same pareto front, the solution with a higher crowd distance value is chosen. The aim for the crowding distance indicator is to maximize the diversity of the outputted non-dominated solutions. In this paper, we evolve a multi-objective evolutionary algorithm depending upon the non-dominated sorting genetic algorithm with pareto tournament for multi-objective optimization (NSGA-IIPT) to tackle the system requirement selection problem. The pseudo code is shown in Algorithm 1. The problem was formulated as a MOOP and a restarting process to the evolutionary algorithms in order to store all the non-dominated solution found, to create diversity in the solutions and to improve coverage. The NSGA-IIPT will keep all the non-dominated solutions found and will be updated at the end of each generation with the non-dominated solution of the current population. The redundant and duplicate non-dominated solution will be removed to avoid overlapping solutions and provide a chance for new solutions and more diversity. Which help to obtain a larger number of solutions of the Pareto front and to better control the algorithm's convergence, a pair of parent solutions is selected from the current population by pareto tournament selection. Also, correct initial population to meet objectives and constraints is during the initialization step and normalizing all data duration execution and we address objective function is addressed to remove the negative signal by switching objective satisfaction as shown in equation (4).
Pseudo code for NSGA-IIPT to solve the system requirement selection problem as follows:

Description pseudo code for NSGA-IIPT in algorithm 1:- The first step of NSGA-IIPT is to accept required data, and step 2 contains the initial values of considered variables, step 3 seeks to randomly create the population "P" of encoded individuals using specific representation. Then, correct initial population to meet drawn objective in step4. Rank is non-dominated-sort algorithm used by NSGA-IIPT to classify individual solutions into different dominance levels. Actually, the concept of Pareto dominance compare each solution "X" with all other solutions in the population until it is being dominated by one of them. However, if there is no solution dominates X, the solution X is to be considered non-dominated and consequently selected to be a member of the Pareto front. The whole population that contains N

```
1  Input data and normalize
2  Pc,Pm ← ∅
3  P leftarrow initialize  population
4  Correct initial population(p)
5  Rank=non_dominated sorting(P)
6  CD=crowding distance assignment(P)
7  Sort(P,Rank,CD)
8  While not Termination Condition() Do
9  parents ← selection(p);
10 Pc← Crossover(p_c,parents);
11 Pm← Mutation(p_m,offspring);
12 P← Merge(P,Pc,Pm);
13 Sort(P,Rank,CD)
14 Remove duplicate individul(P)
15 If size(P<pop_size)
16 Generate(R,pop_size-size(P))
17 P← Merge(P,R)
18 Sort(P,Rank,CD)
19 If size(P>pop_size)
20 truncate(P,size(P)-pop_size)
21 end While
22 Pareto_Front=P
```

Algorithm 1: PSEUDO CODE FOR NSGA-IIPT.

individuals (solutions) is sorted using the dominance principle into several fronts (line 5). Solutions on the first Pareto-front "F0" are assigned dominance level 0. Then, as F0 is taken out, the non-dominated-sort calculates the Pareto-front "F1"of the remaining population. Solutions on the second front are being assigned dominance level 1. The process is set to keep progressing in such procedure till all solutions classified. On one hand, for NSGA-IIPT to cut off a front F and select a subset of individual solutions with the same dominance level, the crowding distance is the key measure used to perform selection (line 6). This parameter is used to promote diversity within the population. This front F, on the other hand, needs to be sorted in descending order before it is being split (line 7). The selection process heavily depends on pareto tournament with both Rank and CD becomes the basis of selection of individual solutions for the next generation and selection parents from population p (line 9). Then, a child population pc, pm are generated from the population of parents using genetic operators represented in crossover and mutation (line 10, 11). Both pc, pm populations are combined into a new population P (line 12). Then, population is sorted in descending order by rank and CD (line 13). The duplicate solutions are bound to be removed to diversify solutions as well as leave the chance for new solutions (line 14). After that, new population R is to be generated if size P less than population size, and merged in the original population to be re-sorted in descending order by rank and CD line(15-18). However, if size P more than population size, overflow and weak solutions are truncated to diversify solutions as well as leave the chance for new solutions

(line 20). Finally, a new P population is set to be created using genetic operators of selection, crossover and mutation. As mentioned earlier, this process will be repeated until reaching the pre-determined max iteration according to the stop criteria. Eventually, the optimal solutions are anticipated to be drawn and pareto front is to be called.

5 -Experiments and Results

In this section, the researcher illustrates the proposed methodology, as well as using the datasets to evaluate its performance. Then, the researcher presents the results obtained through the use of different quality indicators. In addition, the researcher provides a comparison in order to compare the obtained results of our work to those of other approaches published in the literature.

All experiments of this current research have been conducted within the same environment. The researcher used an Intel(R) Core(TM) i3 CPU 2.13 GHz processor with 4 GB RAM. We also used M3.30 compiler on a widows 7 kernel 64 bits OS. Taking into consideration that we are dealing with a meta-heuristic algorithm, one hundred independent runs have been performed for each experiment; thus, the results concluded in the next sub-sections represent the average results of these independent executions. In addition, it is worth mentioning that we used the arithmetic mean for being a valid statistical measurement, as the results mainly followed a normal distribution [9]. Furthermore, all very-low-dispersion results will be illustrated in the tables stated in the following section; thus, they could be deemed as statistically reliable.

Moreover, we used two different real datasets in order to test the effectiveness of our approach. Thus, each dataset was restricted with four different cost boundaries applied to the total cost for the development of all requirements of the system (30%, 50%, 70%, and 100% of the total cost); therefore, our technique was tested with a total of eight instances of the system requirements selection problem. The first dataset included 20 requirements and 5 customers, taken from Greer et al. [15]. Table 1 illustrates the cost associated with each requirement, the priority assigned to each requirement for each customer, and the interaction constraints. The priority for each requirement takes values from 1 to 5 according to the importance of each requirement for the customer. Thus, these values could be interpreted as representing a requirement that is 1:5 in the following order: not important, minor, important, very important and extremely important. In addition, each requirement has an associated cost, which is estimated in terms of a score from 1 to 10. Finally, the number of implication and combination interactions between the requirements present in this dataset has been into consideration.

There is a relative level of importance for each customer in the company. The overall priority, regarding the assignment of a particular customer to a specific requirement, will be based on the given level of priority (illustrated in Tables 1 and 2), as well as the weight assigned to that customer (See Equation 1). Table 3 illustrates the customers' weights for the used two datasets. Moreover, the values have been run from 1 to 5, and sorted preliminarily from the least important to the most important customer. We believe that the only two available real datasets used in this work are those included in [29]; hence, we were able of using these datasets in comparing the present results with those of other previous studies referred to in the literature (Section 5.2). The second dataset included 100 requirements, 5 customers and 44 (implication and combination) requirement interactions, as proposed by Sagrado et al. [29]. Table 2 illustrates the cost associated with each requirement, the priority level given to each requirement for each customer, and the interaction constraints. Table 3 illustrates the relative importance of each customer. There is no doubt that the second dataset is more complex than the previous one. Actually, both the number of requirements (100) and the costs (which, in this case, range from 1 to 20) have been precisely extracted from real agile software project developments. Thus, as variables, the greatest cost of development for a requirement is 20 cost units. In agile software engineering methods [31]. this temporal limit is usually defined as a time box. In this case, the priority levels range from 1 to 3, as when the customers have to make an assignment associated with the benefit of being involved in a new requirement, they would rather use a coarse-grained scale. In specific, the requirements could be classified into three categories as follows: (1) inessential, (2) desirable, and (3) mandatory [30,32].

Table 3: Customers' relative importance

Customers weights	c1	c2	c3	c4	c5
For dataset 1	1	4	2	3	4
For dataset 2	1	5	3	3	1

5.1 -Quality Indicators

Since we were working in a multi-objective environment, the parameter configuration of our proposed approach was established ccording to the quality of the Pareto Front produced in each test. In this regard, we used three quality indicators in order to present a comparative work with regard to other relevant published studies.

5.1.1 -Metric for Diversity

The spread metric Δ was the first quality indicator Δ[33]. It measures the diversity among the solutions using the

Table 1: Dataset 1: assignment of the priority level of each requirement, requirements cost and interactions.

	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	r12	r13	r14	r15	r16	r17	r18	r19	r20
c1	4	2	1	2	5	5	2	4	4	4	2	3	4	2	4	4	4	1	3	2
c2	4	4	2	2	4	5	1	4	4	5	2	3	2	4	4	2	3	2	3	1
c3	5	3	3	3	4	5	2	4	4	4	2	4	1	5	4	1	2	3	3	2
c4	4	5	2	3	3	4	2	4	2	3	5	2	3	2	4	3	5	4	3	2
c5	5	4	2	4	5	4	2	4	5	2	4	5	3	4	4	1	1	2	4	1
Cost	1	4	2	3	4	7	10	2	1	3	2	5	8	2	1	4	10	4	8	4

r3 $\oplus$ r12 r11 $\oplus$ r13 r4 $\Rightarrow$ r8 r4 $\Rightarrow$ r17 r8 $\Rightarrow$ r17 r9 $\Rightarrow$ r3 r9 $\Rightarrow$ r6 r9 $\Rightarrow$ r12 r9 $\Rightarrow$ r19 r11 $\Rightarrow$ r19.

Table 2: Dataset 2: assignment of the priority level of each requirement, requirements cost and interactions.

	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	r12	r13	r14	r15	r16	r17	r18	r19	r20
c1	1	2	1	1	2	3	3	1	1	3	1	1	3	2	3	2	2	3	1	3
c2	3	2	1	2	1	2	1	2	2	1	2	3	3	2	1	3	2	3	3	1
c3	1	1	1	2	1	1	1	3	2	2	3	3	3	1	3	1	2	2	3	3
c4	3	2	2	1	3	1	3	2	3	2	3	2	1	3	2	3	2	1	3	3
c5	1	2	3	1	3	1	2	3	1	1	2	2	3	1	2	1	1	1	1	3
Cost	16	19	16	7	19	15	8	10	6	18	15	12	16	20	9	4	16	2	9	3
	r21	r22	r23	r24	r25	r26	r27	r28	r29	r30	r31	r32	r33	r34	r35	r36	r37	r38	r39	r40
c1	2	1	1	1	3	3	3	3	1	2	2	3	2	1	2	2	1	3	3	2
c2	3	3	3	2	3	1	2	2	3	3	1	3	2	2	1	2	3	2	3	3
c3	2	1	2	3	2	3	3	1	3	3	3	2	1	2	2	1	1	3	1	2
c4	1	1	1	2	3	3	2	1	1	1	1	2	2	2	3	2	2	3	1	1
c5	1	1	3	3	3	2	2	3	2	3	1	1	3	3	2	2	1	1	2	1
Cost	2	10	4	2	7	15	8	20	9	11	5	1	17	6	2	16	8	12	18	5
	r41	r42	r43	r44	r45	r46	r47	r48	r49	r50	r51	r52	r53	r54	r55	r56	r57	r58	r59	r60
c1	2	2	3	1	1	1	2	2	3	3	3	3	1	3	2	1	3	1	3	1
c2	3	3	1	1	3	2	2	2	1	3	3	3	1	2	2	3	3	2	1	1
c3	1	3	1	3	3	3	3	1	3	2	3	1	2	3	2	3	2	1	2	3
c4	3	1	1	3	1	2	1	1	3	2	2	1	3	2	1	3	3	1	2	3
c5	3	1	1	2	1	2	3	3	2	2	1	3	3	2	3	1	2	1	3	2
Cost	6	14	15	20	14	9	16	6	6	6	6	2	17	8	1	3	14	16	18	7
	r61	r62	r63	r64	r65	r66	r67	r68	r69	r70	r71	r72	r73	r74	r75	r76	r77	r78	r79	r80
c1	2	2	3	3	1	3	1	3	2	3	1	3	2	3	1	1	2	3	3	1
c2	1	3	2	3	1	2	1	2	3	1	1	3	1	3	2	1	3	3	1	2
c3	1	1	2	3	3	1	3	3	3	1	3	1	3	1	1	2	3	3	1	2
c4	2	2	3	3	3	1	2	1	2	1	2	3	3	2	2	2	1	3	3	1
c5	2	2	1	2	1	3	2	1	2	1	2	2	3	2	1	3	2	3	1	3
Cost	10	7	16	19	17	15	11	8	20	1	5	8	3	15	4	20	10	20	3	20
	r81	r82	r83	r84	r85	r86	r87	r88	r89	r90	r91	r92	r93	r94	r95	r96	r97	r98	r99	r100
c1	2	1	3	1	2	2	2	1	3	2	2	3	1	1	1	2	1	3	1	1
c2	1	2	1	2	2	1	3	2	2	2	3	2	2	3	2	2	1	3	1	1
c3	1	2	3	2	3	1	2	2	3	3	3	3	2	1	1	2	3	3	2	3
c4	3	1	2	2	2	1	1	1	3	1	1	3	3	1	2	1	2	3	1	3
c5	3	2	1	2	2	2	2	1	3	3	3	1	1	3	1	3	3	3	3	3
Cost	10	16	19	3	12	16	15	1	6	7	15	18	4	7	2	7	8	7	7	3

r21$\oplus$ r22 r32 $\oplus$ r33 r46 $\oplus$ r47 r65 $\oplus$ r66
r2 $\Rightarrow$ r24 r3 $\Rightarrow$ r26 r3$\Rightarrow$ r27 r3 $\Rightarrow$ r28 r3 $\Rightarrow$ r29 r4 $\Rightarrow$ r5 r6 $\Rightarrow$ r7 r7 $\Rightarrow$ r30 r10 $\Rightarrow$ r32 r10 $\Rightarrow$ r33
r14 $\Rightarrow$ r32 r14 $\Rightarrow$ r34 r14 $\Rightarrow$ r37 r14 $\Rightarrow$ r38 r16 $\Rightarrow$ r39 r16 $\Rightarrow$ r40 r17 $\Rightarrow$ r43 r29 $\Rightarrow$ r49 r29 $\Rightarrow$ r50
r29 $\Rightarrow$ r51 r30 $\Rightarrow$ r52 r30 $\Rightarrow$ r53 r31 $\Rightarrow$ r55 r32 $\Rightarrow$ r56 r32 $\Rightarrow$ r5 r33 $\Rightarrow$ r58 r36 $\Rightarrow$ r61 r39 $\Rightarrow$ r63 r40 $\Rightarrow$ r64 r43 $\Rightarrow$ r65
r46 $\Rightarrow$ r68 r47 $\Rightarrow$ r70 r55 $\Rightarrow$ r79 r56 $\Rightarrow$ r80 r57 $\Rightarrow$ r80 r62 $\Rightarrow$ r83 r6 $\Rightarrow$ r84 r64 $\Rightarrow$ r87.

spread metric ($\triangle - Spread$), and calculated through the use of the Euclidean distances between solutions one after the other in the Pareto Front, preferably Pareto Fronts of a smaller spread. Equation(6) was used to calculate $\triangle$-Spread, where d_f and d_l are the Euclidean distance's first and last solution in the Pareto Front, to the peripheral solutions of the optimal Pareto Front in the objective space; $d_{i\_}$ is the Euclidean distance between two solutions,d and the mean distance between each pair of solutions; and N is the total number of solutions in the Pareto Front (see Figure 3 for more details). Through the dispersed solutions, the software engineer could understand all possible tradeoffs between cost and customer satisfaction.

$$\triangle - Spread = \frac{(d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N-1)d}, \qquad (6)$$

Where

$$d_i = \sqrt{(S(x_{i+1}) - S(x_i))^2 + (E(x_{i+1}) - E(x_i))^2}.$$

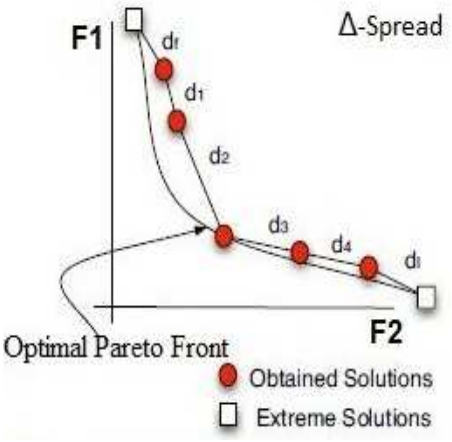


Fig. 3: Distances between solutions

both convergence and diversity of the Pareto fronts obtained. Technically, for Pareto front there is one of two factor used to control HV value to be high: some solutions in the better front takes control of "dominate" solutions in the other, or solutions in the better front are more exceedingly dispersed than the other. However, since both properties are equally good that algorithms of higher values of HV are desirably deemed. In order to calculate this metric, two reference points were needed. Since the problem under consideration has two objectives, these points were $R_{mim}(S(x)_{min}, E(x)_{min})$ and $R_{max}(S(x)_{max}, E(x)_{max})$, i.e., the minimum and maximum values of two objectives tacitly included within points (overall customer satisfaction and development cost). It is worth mentioning that HV is not free from random scaling of the objectives, so that the value of this metric may be distorted as objectives' ranges functions are different. As a result, all the objective function values shall be normalized before the hypervolume being calculated. Table 4, shows the normalization points used for each dataset.

$$HV = volume(\bigcup_{i=1}^{|Q|} v), \qquad (7)$$

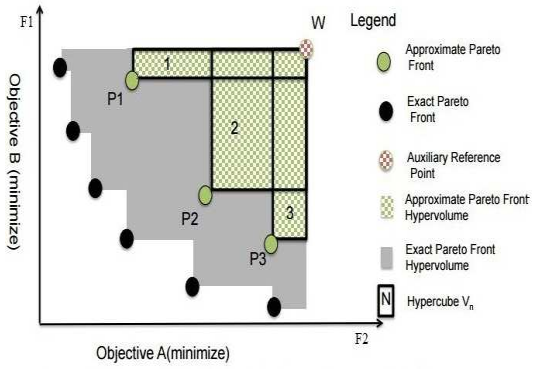


Fig. 4: The hypervolume enclosed by the non-dominated solutions

5.1.2 -The Convergence & Diversity Indicator: hypervolume(HV)

The hypervolume measure (HV), was the second quality indicator [33]. Basically, this measure calculated by equation(7) to compute the volume, drawn in the objective space, contained by members of a non-dominated set of solutions Q (the region enclosed by points into the discontinuous line in Figure(4).
Q = {p1,p2,p3} for problems in which all objectives sought to be minimized. In other words, HV measures

5.1.3 -Pareto Front Size: The Number of the non-dominated solutions (NDS)

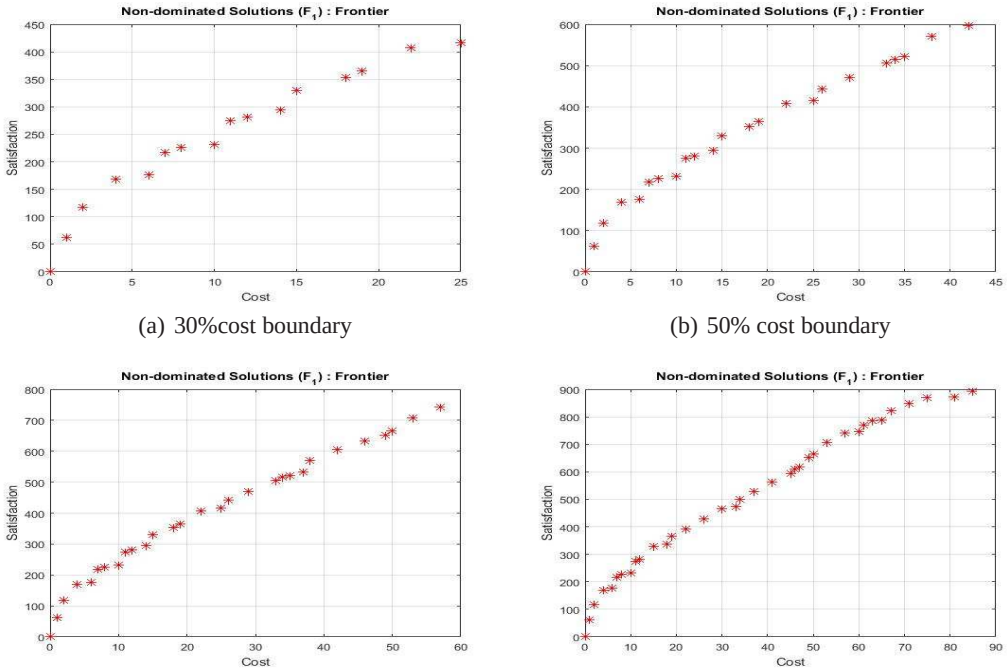
The third quality indicator was the number of non dominated solutions found (NDS). Measures capacity pareto fronts with a higher number solutions are preferred. The measure of number of solutions directly reflects the alternatives provided by the algorithms to the system engineer while selecting the requirements. Any software engineer will be naturally interested in more

Table 4: Datasets main properties and HV reference points.

Datasets	Requirements	Costumers	Interactions constraints	R_mim		R_max	
				Cost	Satisfaction	Cost	Satisfaction
Datasets1	20	5	10	0	0	85	893
Datasets2	100	5	44	0	0	1037	2656

Table 5: NSGA-IIPT configuration for system requirement selection problem.

Initial population size	Crossover probability (Pc)	Mutation factor (Pm)	Parent choice scheme
40	0.8	0.02	Pareto tournament



(a) 30%cost boundary (b) 50% cost boundary

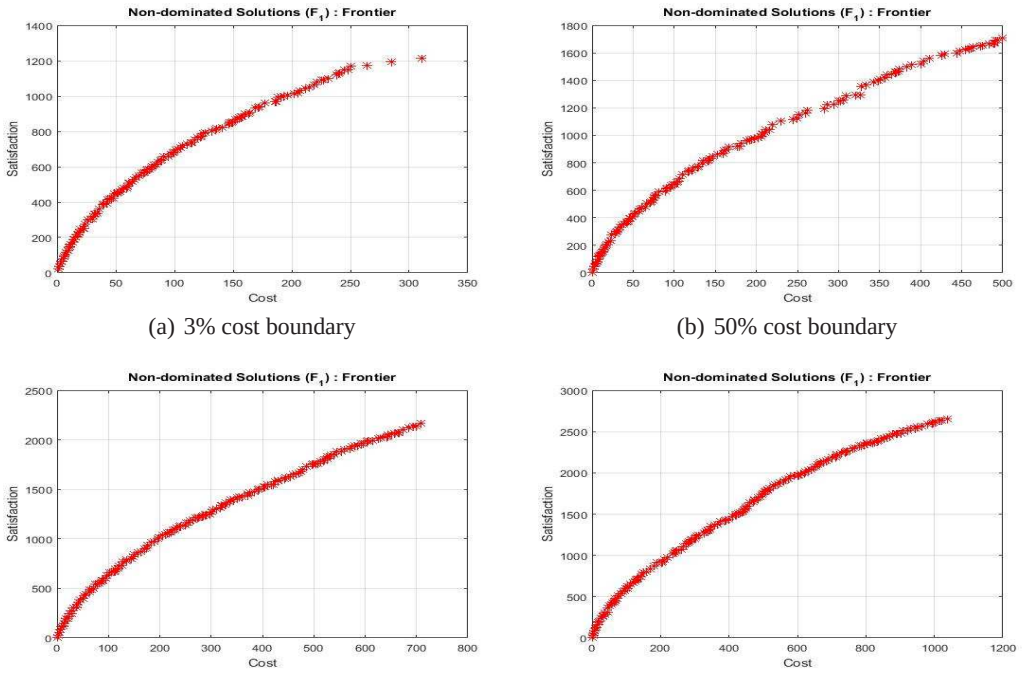
(c) 70% cost boundary (d) 100% not cost boundary

Fig. 5: Pareto fronts for datasets 1

Table 6: Mean HV and standard deviation of the results for the 4 instances of dataset 1.

Dataset 1	NSGA-IIPT	DEPT	ACO	NSGA-II	GRASP
Cost boundary	Mean $\pm$ Std. dev.	Mean $\pm$ Std. dev.	Mean $\pm$ Std. dev.	Mean $\pm$ Std. dev.	Mean $\pm$ Std. dev.
30%	$40.871\% \pm 8.3028\ e^{-3}$	$38.881\% \pm 1.27e^{-5} **$	$10.283\% \pm 6.57e^{-2} **$	$9.015\% \pm 1.12 **$	$7.708\% \pm 0.366 **$
50%	$53.066\% \pm 1.1912e^{-3}$	$50.112\% \pm 1.62e^{-4} *$	$23.912\% \pm 6.75e^{-2} **$	$20.652\% \pm 1.60$**	$19.114\% \pm 0.350$**
70%	$59.637\% \pm 5.3935e^{-4}$	$58.954\% \pm 2.24e^{-4} *$	$38.464\% \pm 7.08e^{-2} **$	$32.157\% \pm 2.30$**	$32.242\% \pm 0.496$**
100%	$62.682\% \pm 3.5896e^{-3}$	$60.776\% \pm 1.03e^{-3} *$			

Independent samples t-test used to compare between NSGA-IIPT algorithm and each other algorithms as follow:
$*$ Statistically significant difference $(p < 0.05)$.
$**$ Statistically significant difference $(p < 0.0)$.
Table 6 shows that there is highly statistically increase significant difference in comparison between NSGA-IIPT algorithm with all other algorithms in all different costs $(p < 0.01)$ except in comparison with DEPT algorithm at 50% cost and more there is statistically increase significant difference $(p < 0.05)$.That means result of our algorithm is better than all other compared algorithms in HV of dataset1.



(a) 3% cost boundary (b) 50% cost boundary

(c) 70% cost boundary (d) 100% not cost boundary

Fig. 6: : Pareto fronts for datasets2

Table 7: Mean HV and standard deviation of the results for the 4 instances of dataset 2.

Dataset 2	NSGA-IIPT	DEPT	ACO	NSGA-II	GRASP
Cost boundary	Mean ± Std. dev.	Mean ± Std. dev.	Mean ± Std. dev.	Mean ± Std. dev.	Mean ± Std. dev.
30%	$41.723\% \pm 9.5854e^{-3}$	$36.508\% \pm 6.01e^{-3} **$	$8.517\% \pm 6.21e^{-2} **$	$7.920\% \pm 2.49e^{-1} **$	$4.088\% \pm 8.55e^{-3} **$
50%	$53.017\% \pm 3.152e^{-3}$	$46.650\% \pm 7.36e^{-3} **$	$19.159\% \pm 9.94e^{-2} **$	$18.006\% \pm 5.20e^{-1} **$	$15.454\% \pm 6.88e^{-2} **$
70%	$60.045\% \pm 5.1526e^{-3}$	$52.753\% \pm 4.25e^{-3} **$	$32.777\% \pm 1.14e^{-1} **$	$31.710\% \pm 8.92e^{-1} **$	$27.943\% \pm 7.50e^{-2} **$
100%	$63.149\% \pm 2.101e^{-3}$	$58.026\% \pm 4.81e^{-3} **$			

Independent samples t-test used to compare between NSGA-IIPT algorithm and each other algorithms as follow:
$**$ Statistically significant difference $(p < 0.0)$.
Table 7 shows that there is highly statistically increase significant difference in comparison between NSGA-IIPT algorithm with all other algorithms in all different costs $(p < 0.01)$. That means NSGA-IIPT result of our algorithm is better than all other compared algorithms in HV of dataset2.

number of solutions. Finally, in regard to the algorithm's configuration, in order for fair comparisons to be made with other work [34], the same stop condition is used for our technique: 10,000 fitness function evaluations. The other algorithm parameters were tuned one by one to obtain the best results for the problem being tackled. Thus, Table 5 summarizes the parameter configuration for our proposal NSGA-IIPT.

5.2 -Results, Discussion, and Comparison With Other Work

In this section, we present the results obtained with our proposed NSGA-IIPT approach, and compare them with those published in other work [34]. We start by investigating the values of the HV and $\Delta - Spread$ quality measures, and finally we perform a comparative study of the number of non-dominated solutions (NDS) obtained by different algorithms.

5.2.1 -The Convergence & Diversity Indicator: Hypervolume (HV) results

The comparative results in related to HV indicator summarized in Tables 6 and 7. The results of previous work (DEPT, ACO, NSGA-II and GRASP) are compared with our proposed approach, NSGA-IIPT. The obtained results demonstrate average hypervolume, and standard deviation, of 100 distinct runs for the two datasets under

Table 8: Mean $\Delta - Spread$ and standard deviation of the results for the 4 instances of dataset 1.

Dataset 1	NSGA-IIPT	DEPT	ACO	NSGA-II	GRASP
Cost boundary	Mean ± Std. dev.	Mean ± Std. dev.	Mean ± Std. dev.	Mean ± Std. dev.	Mean ± Std. dev.
30%	0.48±0.039032	0.52 ±0.02**	0.52±0.03**	0.76 ±0.09**	0.64 ±0.09**
50%	0.43±0.030338	0.48 ±0.01**	0.52 ±0.01**	0.79 ±0.07**	0.73 ±0.07**
70%	0.42±0.036723	0.42± 0.03	0.48 ±0.02**	0.80 ±0.07**	0.69 ±0.06**
100%	0.37±0.017507	0.40 ±0.04**			

Independent samples t-test used to compare between NSGA-IIPT algorithm and each other algorithms as follow:
$**$ Statistically significant difference $(p < 0.0)$.
Table 8 shows that there is highly statistically decrease significant difference in comparison between NSGA-IIPT algorithm with all other algorithms in all different costs $(p < 0.01)$ except in comparison with DEPT algorithm at 70% cost there is no statistically significant difference $(p > 0.05)$.That means result of our algorithm is more better than all other compared algorithms in $\Delta - Spread$ of dataset1.

Table 9: Mean $\Delta - Spread$ and standard deviation of the results for the 4 instances of dataset 2.

Dataset 2	NSGA-IIPT	DEPT	ACO	NSGA-II	GRASP
Cost boundary	Mean ± Std. dev.	Mean ± Std. dev.	Mean ± Std. dev.	Mean ± Std. dev.	Mean ± Std. dev.
30%	0.50±0.041134	0.56 ±0.04**	0.68 ±0.06**	0.80 ±0.07**	0.60 ±0.04**
50%	0.44±0.032874	0.51 ±0.03**	0.66 0.06**	0.81 ±0.06**	0.74 ±0.04**
70%	0.37±0.024859	0.47 ±0.03**	0.61 ±0.06**	0.77 ±0.05**	0.70 ±0.03**
100%	0.37±0.02435	0.44 ±0.04**			

Independent samples t-test used to compare between NSGA-IIPT algorithm and each other algorithms as follow:
$**$ Statistically significant difference $(p < 0.01)$.
The Table 9 shows that there is highly statistically decrease significant difference in comparison between NSGA-IIPT algorithm with all other algorithms in all different costs $(p < 0.01)$.That means result of our algorithm is better than all other compared algorithms in $\Delta - Spread$ of dataset2.

Table 10: Mean number of NDS and standard deviation of the results for the 4 instances of dataset 1.

Dataset 1	NSGA-IIPT	DEPT	ACO	NSGA-II	GRASP
Cost boundary	Mean ± Std. dev.	Mean ± Std. dev.	Mean ± Std. dev.	Mean ± Std. dev.	Mean ± Std. dev.
30%	15.4±0.96609	15 ±0.00	13.66 ±13.66**	9.69 ±2.09**	11.37 ±1.47**
50%	22.2± 1.2293	19.76±0.38**	17.75±0.61**	11.30±1.82**	17.65 ±2.22**
70%	29.5±1.5092	26.22 ±2.17**	20.57 ±20.57**	11.70 ±1.90**	20.26±2.18**
100%	37.3±1.567	30.51 ±2.62**			

Independent samples t-test used to compare between NSGA-IIPT algorithm and each other algorithms as follow:
$**$ Statistically significant difference $(p < 0.01)$.
Table 10 shows that there is highly statistically increase significant difference in comparison between NSGA-IIPT algorithm with all other algorithms in all different costs $(p < 0.01)$ except in comparison with DEPT algorithm at 30% cost there is no statistically significant difference $(p > 0.05)$.That means result of our algorithm is better than all other compared algorithms in NDS of dataset1.

Table 11: Mean number of NDS and standard deviation of the results for the 4 instances of dataset 2.

Dataset 2	NSGA-IIPT	DEPT	ACO	NSGA-II	GRASP
Cost boundary	Mean ± Std. dev.	Mean ± Std. dev.	Mean ± Std. dev.	Mean ± Std. dev.	Mean ± Std. dev.
30%	123.3±6.3078	110.108 ±5.45**	47.12±5.44**	54.34 ±8.51**	57.99 ±3.66**
50%	150.1±4.9092	123.64 ±5.20**	57.68 ±5.69**	65.54 ±11.86**	75.81 ±5.81**
70%	159.8±3.4897	139.73 ±8.32**	70. 98 ±5.27**	83.32 ±10.52**	120.14 ±7.27**
100%	163.8±2.1499	144.50 ±7.16**			

Independent samples t-test used to compare between NSGA-IIPT algorithm and each other algorithms as follow:
$**$ Statistically significant difference $(p < 0.01)$.
The Table 11 shows that there is highly statistically increase significant difference in comparison between NSGA-IIPT algorithm with all other algorithms in all different costs $(p < 0.01)$.That means result of our algorithm is better than all other compared algorithms in NDS of dataset2.

consideration, with the four cost boundaries (30%,50%,70% and 100% -no cost limit). In addition, just 8 instances of the problem have been investigated. As mentioned earlier, the higher the value of HV, the better the quality of the results achieved. It is clearly stated that, from results, NSGA-IIPT algorithm outperforms the others in terms of HV for every problem instance. Moreover, NSGA-IIPT draws very low dispersions for every cost boundary examined. It is worth indicating that for NSGA-IIPT whenever iteration of runs gets increased the dispersions being more decreased. Therefore, it can be concluded that the overall improvement gained by NSGA-IIPT is quite significant when it comes to this measure. To sum up, NSGA-IIPT is proved to be capable to stand above its peers in best exploring the search space, and consequently, the solutions acquired for the requirements selection problem is definitely of better quality. However, GRASP is shown to be the meta-heuristic of the poorest results. Actually, this back to the fact that GRASP is a trajectory-based meta-heuristic and subsequently does not work well with a population of individuals so the exploration of the space search is more bounded.

5.2.2 -Metric for Diversity $\Delta - Spread$ results

In this subsection, the analyzing of the $\Delta - Spread$ quality indicator has been focused on for the same instances of the considered problem. In contrast to HV, as the spread indicator gets lower values, the better results are achieved. Tables 8 and 9 brief the results obtained concerning this indicator as well as taken to be compared with those published in the literature. From the tables it can said that NSGA-IIPT has gained the best results in all cases, with standard deviations being more reduced. Hence, NSGA-IIPT has proved that it computes the fronts with the best distribution of solutions in all instances. Consequently, for NSGA-IIPT, it could be claimed that a set of optimal solutions are produced such that able to capacitate more variety than the results generated by the other approaches published. Finally, graphically comparison could not be performed for the Pareto fronts in purpose of drawing these differences, because there is no information available regarding the approaches published in literature.

5.2.3 -Pareto Front Size: The Number of the non-dominated solutions (NDS) results.

In multi-objective optimization problems; thus, it is preferred for the human expert to find more optimal solutions, when the selection of the final solution has to be fulfilled. Nonetheless, the optimal solutions for the problem tackled in this current paper are unknown, because the set of all final high-quality solutions given by the MOEAs are not called optimal solutions, but non-dominated solutions or Pareto Front. Tables 10 and 11 illustrate the average number of non- dominated solutions obtained through our approach, NSGA-IIPT and other algorithms published in the literature with regard to the different problem instances. The results in the tables illustrate that NSGA-IIPT obtains a higher number of non-dominated solutions in all cases. The number of the found non-dominated solutions (NDS) is bigger for dataset 2 in each case, as the second dataset is more complex. In spite of this fact, we can also observe that the differences between our proposed approach and the other published approaches are more significant due to the more complex dataset. Figures 5 and 6 illustrate the Pareto Front obtained by our proposed approach NSGA-IIPT, for dataset 1 and dataset 2 respectively.

6 -Conclusions and Future Work

In this current paper, we have proposed the use of a novel non-dominated sorting genetic algorithm with Pareto tournament for a multi-objective optimization approach (NSGA-IIPT); and that is in order to tackle real instances of the system requirements selection problem. The paper comprises a constrained multi-objective formulation of the problem in which different types of interactions between the requirements and several cost boundaries have been taken into consideration. Furthermore, we have evaluated the proposed approach in terms of several quality indicators, by comparing the results generated by our proposed approach to several approaches published in the literature (DEPT, ACO, NSGA-II, GRASP). In addition, after analyzing the results, we can easily conclude that NSGA-IIPT is able to obtain the best sets of requirements, thus NSGA-IIPT generates sets of non-dominated solutions with more solutions in the Pareto Fronts (Tables 10 and 11), with a lower spread between the solutions (Tables 8 and 9) and a higher hypervolume (Tables 6 and 7) than the other approaches. Furthermore, the tests showed statistically significant differences for all results, and showed that there is a statistically significant difference increase for NSGA-IIPT higher than the other approaches.
Thus, the results of this paper show that our proposed approach can efficiently produce high quality solutions that allow the system engineers to make decisions regarding the set of requirements that shall be included in the system. The researcher used eight several cases taken from two real-world datasets in order to check the effectiveness of the proposed approach. These datasets included different numbers of requirements, requirement interactions and customer priorities; and both of them had previously been employed by other published works, so that we were able to make comparisons with the results of the current study. Since the results obtained with NSGA-IIPT were good, we believe that in the future works, it might be interesting to work with other multi-objective approaches based on NSGA-IIPT which

can be applied to the problem. A hybrid version of our proposed approach with some other multi-objective approaches could be a good example for these works. It might also be interesting to study the use of this technique, as it could be applied on larger real-world datasets. In pursuit of this objective, it would be of very important to propose a generator of multi-objective optimization system requirements selection problem dataset for the systematic generation of instances. Such enhancement shall be addressed with the assistance of system design experts, as the question of the interactions between the requirements in a complex system is far from trivial. In addition, other formulations of the problem, taking into consideration other and more complex constraints, more dominance relations and more objectives, would also be interesting lines for future works. Finally, we shall mention that the translation of these methods into CASE tools should also be taken into consideration, in order to make this work more useful to the software industry.

References

[1] Bashar Nuseibeh, and Steve Easterbrook, ”Requirements engineering: a roadmap.” Proceedings of the Conference on the Future of Software Engineering. ACM, 2000.

[2] David Lorge Parnas, ”Software engineering programs are not computer science programs.” Software, IEEE 16.6 (1999):19-30.

[3] Guenther Ruhe, Product release planning: methods, tools and applications. CRC Press, 2010.

[4] RL Glass, ”Facts and fallacies of software engineering Boston:Addison-Welsey.” (2003).

[5] Patrik Berander, and Mikael Svahnberg, ”Evaluating two ways of calculating priorities in requirements hierarchies-An experiment on hierarchical cumulative voting.” Journal of Systems and Software 82.5 (2009):836-850.

[6] A.J. Bagnall, V.J. Rayward-Smith,and I.M. Whittley, ”The next release problem.” Information and software technology 43.14 (2001): 883-890.

[7] Peter Schuster, ”Optimization of multiple criteria: Pareto efficiency and fast heuristics should be more popular than they are.” Complexity 18.2 (2012):5-7.

[8] F.J.Orellana, J.Canadas, I.M.del Aguila, and S.Tunez, ”INSCO Requisite-A Web-Based RM-Tool to support Hybrid Software Development.” ICEIS (3-1). 2008.

[9] Alan M. Davis, ”The art of requirements triage.” Computer 36.3 (2003):42-49.

[10] David S Johnson, ”The NP-completeness column: an ongoing guide.” Journal of algorithms 13.3 (1992): 502-524.

[11] Joachim Karlsson, ”Software requirements prioritizing.” Requirements Engineering, 1996., Proceedings of the Second International Conference on. IEEE, 1996.

[12] Juan J. Durillo, Yuanyuan Zhang, Enrique Alba, Mark Harman, and Antonio J. Nebro, ”A study of the bi-objective next release problem.” Empirical Software Engineering 16.1 (2011):29-60.

[13] Mark Harman, S. Afshin Mansouri, and Yuanyuan Zhang, ”Search-based software engineering: Trends, techniques and applications.” ACM Computing Surveys (CSUR) 45.1 (2012):11.

[14] Paul Baker, Mark Harman, Kathleen Steinhofel, and Alexandros Skaliotis, ”Search based approaches to component selection and prioritization for the next release problem.” Software Maintenance, 2006. ICSM’06. 22nd IEEE International Conference on. IEEE, 2006.

[15] Des Greer, and Gunther Ruhe, ”Software release planning: an evolutionary and iterative approach.” Information and Software Technology 46.4 (2004):243-253.

[16] Yuanyuan Zhang, Mark Harman, and S. Afshin Mansouri, ”The multi-objective next release problem.” Proceedings of the 9th annual conference on Genetic and evolutionary computation. ACM, 2007.

[17] Martin S. Feather, and Tim Menzies, ”Converging on the optimal attainment of requirements.” Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on. IEEE, 2002.

[18] Martin S. Feather, Steven L. Cornford, James D. Kiper, and Tim Menzies, ”Experiences using visualization techniques to present requirements, risks to them, and options for risk mitigation.”Requirements Engineering Visualization, 2006. REV’06. First International Workshop on. IEEE, 2006.

[19] Mark Harman, Alexandros Skaliotis, and Kathleen Steinhofel, ”Search–based approaches to the component selection and prioritization problem.” Proceedings of the 8th annual conference on Genetic and evolutionary computation. ACM, 2006.

[20] Omid Jalali, Tim Menzies, and Martin Feather, ”Optimizing requirements decisions with keys.” Proceedings of the 4th international workshop on Predictor models in software engineering. ACM, 2008.

[21] Jose d.Sagrado, Isabel M.d Aguila, and Francisco J.Orellana, ”Ant colony optimization for the next release problem: A comparative study.”Search Based Software Engineering (SSBSE), 2010 Second International Symposium on. IEEE, 2010.

[22] Jose d.Sagrado, Isabel M.d Aguila, and Francisco J.Orellana, ”Requirements interaction in the next release problem.” Proceedings of the 13th annual conference companion on Genetic and evolutionary computation. ACM, 2011.

[23] Paolo Tonella, Angelo Susi, and Francis Palma, ”Interactive requirements prioritization using a genetic algorithm.” Information and software technology55.1 (2013): 173-187.

[24] Paolo Tonella, Angelo Susi, and Francis Palma, ”Using interactive GA for requirements prioritization.” Search Based Software Engineering (SSBSE), 2010 Second International Symposium on. IEEE, 2010.

[25] Abdullah Konaka, David W.Coitb, and Alice E.Smithc, ”Multi-objective optimization using genetic algorithms: A tutorial.” Reliability Engineering & System Safety 91.9 (2006): 992-1007.

[26] Kalyanmoy Deb, Amrit Pratap, and Sameer Agarwal, ”A fast and elitist multiobjective genetic algorithm: NSGA-II.” Evolutionary Computation, IEEE Transactions on 6.2 (2002): 182-197.

[27] Yuanyuan Zhang, Anthony Finkelstein, and Mark Harman, ”Search based requirements optimisation: Existing work

and challenges." Requirements Engineering: Foundation for Software Quality. Springer Berlin Heidelberg, 2008. 88-94.
[28] Shin Yoo, Mark Harman, "Pareto efficient multi-objective test case selection." Proceedings of the 2007 international symposium on Software testing and analysis. ACM, 2007.
[29] Jose d.Sagrado, Isabel M.d Aguila, and Francisco J.Orellana, "Multi-objective ant colony optimization for requirements selection." Empirical Software Engineering 20.3 (2015): 577-610.
[30] Janez Demsar, "Statistical comparisons of classifiers over multiple data sets."The Journal of Machine Learning Research 7 (2006): 1-30.
[31] Ken Schwaber, Mike Beedle, "Agile Software Development with Scrum." (2002).
[32] Karl E. Wilegers, "Software requirements.". 2nd ., Microsoft Press, Redmond, WA, USA, 2003.
[33] Eckart Zitzler, Lothar Thiele, and Johannes Bader, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach." evolutionary computation, IEEE transactions on 3.4 (1999): 257-271.
[34] Jose M.Chaves-Gonzalez, Miguel A.Perez-Toledano, "Differential evolution with Pareto tournament for the multi-objective next release problem." Applied Mathematics and Computation 252 (2015): 1-13.

M. H. Marghny is a Professor of Computer Science, Vice Dean of Faculty of Computers and Information, Assiut University. He received his Ph.D. degree in computer science from the University of Kyushu, Japan, in 2001, his M.Sc. and B.Sc. from Assiut university, Assiut, Egypt, in 1993 and 1988, respectively. He is currently a professor in the Department of Computer Science, and Vice Dean for Education and Student Affairs of the Faculty of Computers and Information, University of Assiut, Egypt. His research interests include data mining, text mining, information retrieval, web mining, machine learning, pattern recognition, neural networks, evolutionary computation, fuzzy systems, and information security. Prof. Marghny is a member of the Egyptian mathematical society and Egyptian syndicate of scientific professions. He is a manager of some advanced research projects in Faculty of Computers and Information, University of Assiut, Egypt.

H. M. El-Hawary is Professor of Mathematics, Dean Faculty of Science, Assiut University. Received the PhD degree in Mathematics Science at numerical analysis from Assiut University, in 1990. His research "Numerical Treatment of Differential Equations by Spectral Methods". Received the MA degree in Mathematics Science at numerical analysis from Assiut University, in 1984. His research "Numerical Solution of a System of First Order Differential Equations by Splines". Received the B.Sc. In Mathematics Science, from Assiut University, in 1980. His research interests include Numerical Analysis, Global Optimization, Partial Differential Equations and Operation Research. Prof. H.M.El-Hawary is a member of the Egyptian mathematical society.

Wathiq H. Dukhan is currently M.Sc. student in the Department of Mathematics of the Faculty of Science, University of Assiut, Egypt. received his B.s. degree in computer mathematics from the Sana'a University, Sana'a, Yemen, He is a Demonstrator in the Department of Mathematics of the Faculty of science, University of Sana'a,Yemen.