Received: 28 February 2022 | Revised: 22 June 2022 | Accepted: 18 July 2022

IET Software

DOI: 10.1049/sfw2.12070

The Institution of Engineering and Technology WILEY

ORIGINAL RESEARCH

Software requirement selection using a combined multi-objective optimisation technique

Wathiq H. Dukhan[1] | Marghny H. Mohamed[2] | Ali A. Amer[3] | Elnomery Allam Zanaty[4] | Omar Reyad[5]

[1]Faculty of Science, Sana'a University, Sana'a, Yemen

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

[3]Faculty of Applied Science, Computer Science Department, Taiz University, Taiz, Yemen

[4]Department of Computer Science, Faculty of Computers and Artificial Intelligence, Sohag University, Sohag, Egypt

[5]Faculty of Science, Sohag University, Sohag, Egypt

Correspondence

Wathiq H. Dukhan, Faculty of Science, Sana'a University, Sana'a, Yemen.
Email: dralwathiq2012@gmail.com

Ali A. Amer, Faculty of Applied Science, Computer Science Department, Taiz University, Taiz, Yemen.
Email: aliaaa2004@yahoo.com

Abstract

The optimal requirements selection set aims primarily at careful search for the best requirements set of the next release of software during development process. This procedure is widely defined as the next release problem (NRP), which is also classified as NP-hard dilemma. Several techniques, in literature, have been proposed to tackle NRP. However, in real examples, the earlier studies still immature as NRP still suffers interactions and restrictions that makes the problem more complicated. Although few interesting works have been presented, yet NRP, based on our study, could be further investigated and effectively tackled. In this research, therefore, NRP is devised as a multi-objective optimisation problem. Two clashing objectives (satisfaction and cost) and two constraints (interactions forms) are formulated. To tackle NRP effectively, a new hybrid genetic and artificial bee colony algorithm (HGABC) is introduced. HGABC combines features of genetic and artificial bee colony algorithms. Experimental study, using case studies and three criteria, have been conducted to show HGABC's power of generating non-dominated effective Pareto solutions versus the state-of-the-art algorithms. Results indicate that HGABC does not just outperform its rivals, yet also gives better Pareto solutions in terms of diversity and quality for almost all the instances of this problem.

KEYWORDS

artificial bee colony, genetic algorithm, next release problem, search-based software engineering

1 | INTRODUCTION

Elicitation and determination of the requirement set is a crucial phase of software engineering process to develop the final software package. The assembling, thoroughly examining, and recording all requirements of customers is well known as requirement engineering (RE) [1]. Following elicitation, requirements list for software development is created. Due to the insufficient communication, the efficient search results for requirements of each customer is being serious matter for the success of requirements engineering, leading to a lower satisfaction rate and high costs. As a result, the requirement assembling phase—of the software development life cycle (SDLC)—is being challenge [2]. So, an incremental development method is a popular approach introduced for the software development. It is an appropriate option for the big-scale software projects. This method is utilised to evolve software in multiple phases. Every phase expresses requirements set which should be produced exclusively by the software company. It is difficult to effectively develop all requirements due to several factors such as, technical constraints, project budget constraints, time-table constraints, project delivery, and inherent requirement conflicts [3]. To make it worse, selecting an appropriate requirements subset in large projects is maximally challenging and error-prone.

Consequently, a process to recognise the best-fit requirements subset which would help the responsible team in taking an action, is in high demand [4]. Bagnall et al. [5] presented single-objective optimisation next release problem. The process was then performed to shrink costs while increasing

customer satisfaction. Zhang et al. [6] presented a multi-objective optimisation for NRP by drawing the well-suited requirements subset whose features must be contained in the next phase of software to preserve customers' contentment while maintaining evolution costs as much minimal. More than often, customer demands requirements that contradicts another customer's demand [7]. So, the NRP is treated as a multi-objective optimisation problem (MOP) due to the conflict between objectives, signalling that the traditional optimisation techniques cannot solve NRP effectively. Therefore, multi-objective optimisation algorithms is needed [8], which is the proposal to optimise many objective functions at once. For such problems, solutions group recognised as the non-dominated solutions or Pareto optimal solutions' set has been emerged [6].

Lately, there has been substantial increase in the popularity of the Search-Based Software Engineering (SBSE) techniques, which addresses problems using search-based algorithms. As a result, these problems are often handled using meta-heuristic techniques. The heuristic search employs random numbers to draw suitable solutions for problems of optimisation without the need to evaluate the entire search space [9]. Penalty-based boundary intersection, weighted sum, and Tchebycheff are examples of aggregation functions or decomposition approaches [10] that are commonly utilised for dealing with MOPs. They divide MOPs into a series of single-objective sub-problems utilising aggregation techniques. While these techniques provide a single solution, the SBSE techniques generate a set of Pareto front solutions, providing decision-makers with a more complete picture of what they truly want to understand.

In this work, therefore, to tackle the multi-objective next release problem (MNRP), we seek to develop a multi-objective method. We approach a new hybrid algorithm, denoted as HGABC, to tackle MNRP problem. HGABC combines features of genetic and artificial bee colony algorithms. The hybridisation process achieves the exploitation, exploration, and convergence capabilities of the optimisation process, as well as the decision-maker's ability to consider a large number of alternatives/optimal solutions. Thus, the combination of both genetic algorithm (GA) and artificial bee colony algorithm (ABC) is done to improve the latter's search capability by using the former's strong global search capability. The aim is to show the power of hybrid method in achieving the desired solutions for MNRP while maintaining the solutions diversity on the Pareto front. The following factors are considered: minimum cost, maximum customer satisfaction, interaction constraint, and cost threshold constraint. Comparing with some state-of-the-art rivals, our proposed method draws promising findings. Furthermore, MNRP is measured using quality indicator of the multi-objective HGABC method, and the results are compared using four multi-objective optimisation quality indicators and four cost bounds.

The main contributions of the research: (1) Presenting a new hybrid algorithm HGABC (through combining ABC and GA). HGABC is extensively examined using two objective functions over several real-world datasets, and this hybridised algorithm is used for the first time to tackle the MNRP in SE field; (2) Formulate the GA's operators (Crossover and Mutations) mathematically to express clearly their key role in enhancing the produced solutions of HGABC. Concisely, a novel hybrid approach HGABC is therefore presented for multi-objective optimisation problems by combining the ABC and GA algorithms as shown in pseudo-code of Algorithm 1. When these two algorithms are combined, ABC is used as the basic algorithm that makes use of GA operators (mutation, and crossover). The HGABC method has the same structure of ABC algorithm; however, the new solution generation process generates more solutions each time due to the use of GA operators in phases of the ABC algorithm. This approach selects optimal solution vectors from the current population-based on dominance values for each generation of ABC to apply GA operations such as mutation and crossover, which allow effective scanning of the search space. As a result, the improved ABC algorithm may keep ABC's global search capabilities while absorbing GA's local optimal solutions to solve the next release problem with greater accuracy. (3) Drawing an extensive empirical study, on two most widely-used and publicly-available datasets, to demonstrate the outstanding performance of the HGABC comparing with the relevant "state-of-the-art" meta-heuristic algorithms.

These contributions are thus summarised as follows: First, by combining the features of ABC and GA, a new hybrid algorithm HGABC is presented and tested on two conflicting objective functions and a real-world problem. Second, the superiority of the created hybrid HGABC is demonstrated by comparing simulation results with those of existing meta-heuristic algorithms over the most commonly-used datasets. Finally, it is worth indicating that utilising the proposed techniques could reduce human errors as the HGABC algorithm was used to selection of optimal subsets or generate the best NRP solutions. This work is planned as follows: Section 2 explores the earlier approaches that used to address NRP and MNRP problems. The optimisation methods and the mathematical formulation of these problems are drawn in Section 3. Thorough explanations of the ABC and GA algorithms and various operators for these algorithms are provided in Section 4. In Section 5, the proposed HGABC optimisation algorithm and its numerous operators are deeply explained. Section 6 offers a numerical cases study and data comparison to illustrate the HGABC method's efficiency and competence. Section 7 summarises paper's conclusions and makes suggestions for future work.

2 | RELATED WORK

The NRP problem was rudimentary devised in SBSE as a problem of optimisation [5]. Several widely-known techniques, such as the linear programing, simulated annealing, and hill-climbing [5, 11], have been proposed to resolve NRP. Utilising aggregation functions for objectives, these techniques however consider NRP with monocular-objective problem. MNRP was presented and devised as a MOP [6]. MONRP was tackled through multi-objective NSGA-II, GA, SPEA2, and

PESA [7, 11, 12], WOA, GWO, SPEA2 and NSGA-II, [13, 14]. The NRP [5, 12, 15] and MNRP [7, 12, 16] have been widely tackled using genetic algorithms. The proposal and utilisation of evolutionary approaches, like multi-objective ABC (MABC) and teaching-learning-based optimisation (TLBO) [17, 18], ant colony optimisation (ACO) [7, 19, 20], have been played a key role in solving real instances for MONRP [21]. For example, Xuan et al. [22] presented an algorithm for resolving large-scale MNRP problems through a multi-level backbone search. Chavez et al. [17] proposed a bi-objective algorithm (TLBO). A multi-objective differential evolution (DE) technique was developed considering the Pareto tournaments (MODE) [23].

In the meanwhile, the Multi-Objective TLBO with ABC (MOTLABC) was suggested in [18]. Zhang et al. [24] proposed a new formularisation for NRP in which customers had a unique goal to maximise their satisfaction. NRP was solved, from three different datasets, using random search, NSGA-II, and two-Archive. Many features of the software are qualitative in nature, and thus quantifying them is difficult; even estimating the software's cost is challenging. Harman et al. [25] introduced a method for the sensitivity analysis depending on exact techniques for measuring the robustness of outputs in the existence of the unpredictability.

Comparing with previous exact algorithms, which were incapable of dealing with spread solutions, Domínguez-Ríos et al. [26] suggested exact techniques for recognising a group of fully-spread solutions through the investigation process. Hamdy and Mohamed [27] sought to tackle MNRP with a greedy-random swarm initialisation strategy. This strategy was used to improve binary PSO, considering the order of the requirements interaction and defining two additional velocity vectors for each particle. The objective functions were created using aggregation functions of overall satisfaction and costs of development. The best solution was then discovered instead of Pareto optimal solutions.

Alrezaamiri et al. [8], used the master-slave model to present the first method which used parallelism to solve MNRP. On the master and slave base, numerous MABC algorithms were run simultaneously. Each MABC algorithm begun with a separate random population and randomly determined operators. Thus, the opportunities of obtaining dissimilar solutions in each run were bigger. As a matter of fact, parallel algorithm runs can produce better results over the sequential runs. Then, utilising the technique of artificial chemical reaction optimisation, the MNRP was tackled [28]. Moreover, a fuzzy inference system-based method was built to decide the fitness of every requirement for the next-release development. The developed algorithm was in charge of selecting the most suitable sub-set of requirements for the next release [29]. Marghny et al. [14] presented a method to tackle the MNRP which combined the Pareto tournament and NSGA-II. This algorithm considered the cost and satisfaction objectives during the software requirements optimisation with the multi-objective formulation. In addition, the differential evolution was integrated into an artificial bee colony and used to tackle the multi-objective NRP [30]. The results showed that this algorithm outperformed its competitors. Utilising the genetic K-means algorithm [31], the clustering of stakeholder, for requirements, was addressed. NRP was deemed an suitable domain for the constrained goal model [32]. The requirements were organised hierarchically and interrelated. Further to this, a goal-oriented devise for the NRP was given, and inference models were utilised to decide Pareto optimal solutions. Kumari et al. [33] integrated the quantum computing into the evolutionary algorithm, differential evolution, and multi-objective quantum-inspired hybrid differential evolution techniques to run more effective investigation in treating the problem of requirements selection. Araújo et al. [34] investigated both uncertainty and fairness in NRP by fusing machine learning and interactive optimisation.

On the other hand, the uncertainty and distance have been considered as objective functions by some recent works. Finkelstein proposed several fairness concepts, which were addressed as objective functions in the MNRP [16]. Geng et al. [35] proposed simultaneous optimisation to tackle MNRP and devised MNRP with five objective functions. Given the negative implications of swelling the number of objective functions on the quality of the solutions, the uncertainty and fairness are considered quality indicators instead of objective functions. Mohsen et al., sought to tackle the MNRP using five multi-objective evolutionary algorithms including grey wolf and whale optimisation [13].

The results illustrated that multi-objective grey wolf algorithm performed better than its competitors. A meta-heuristic approach, named the Binary Artificial Algae Algorithm, was presented to select the optimal requirements subset [36]. The results demonstrated that the presented algorithm produced subsets with no human mistakes. To facilitate work on requirement selection, the Nautilus tool was declared in [37]. It has been utilised to showcase the functionality of Nautilus' primary, and how it is adjusted to tackle software engineering problems. Renzo et al. [38] showed the utility of Virtual Savant (VS) to learn tackling NRP automatically, and in an accurate and swift procedure leveraging an exact method. It was a generic problem-solving method contingent on machine learning models which imitate the solution generation to problem instances using the reference programme. While Table 1 presents a brief summary of the most relevant works that have been surveyed, a classification of the meta-heuristic algorithms is provided in Figure 1.

3 | MULTI-OBJECTIVE NRP

This section holds the description of software requirements selection. However, to comprehend the research purpose, we start by analysing some related-multi-objective concepts from theory of dominance.

3.1 | Review of multi-objective optimisations

In reality, optimisation problems involve appointing the values of decision variables which achieve one or several objectives via maximisation (or minimisation). For example, the single-

TABLE 1 Meta-heuristic algorithms and objectives from selected studies

Reference	Algorithm	Objective
Durillo et al. [39]	NSGA-II, MOCell	Min cost and max satisfaction
Sagrado et al. [19]	ACO, SA, GA	Min effort and max satisfaction
Jiang et al. [20]	HACO, GRASP, FHC	Max Profit and min cost
Souza et al. [40]	ACO, SA, GA	Max Profit and min cost
Cai et al. [41]	GA	Min cost and max satisfaction
Cai and Wei [42]	MOEA/D, NSGA, SPEA2	Min cost and max satisfaction
Araujo and Paixao [43]	IGA	Budget and cost
Silva et al. [44]	MOGAs	Max Profit and min cost
Chaves Gonzalez et al. [21]	MOABC	Min cost and max satisfaction
Chaves Gonzalez et al. [17]	MO-TLBO	Min cost and max satisfaction
Sagrado et al. [7]	ACO	Min cost and max satisfaction
Chaves-Gonzalez and Perez-Toledano [23]	DEPT	Min cost and max satisfaction
Ranjith and Marimuthu [18]	MOTLABC	Max satisfaction, min time, min cost, and max reliability
Araujo et al. [34]	IGA, ML	Max satisfaction and min effort
Raphael Saraiva et al. [45]	MOCell, IBEA, SPEA-II	Max satisfaction min value and risk
Marghny et al. [14]	NSGA-IIPT	Min cost and max satisfaction
A. Hudaib et al. [46]	WGW, WO	Min Req. number and max satisfaction
Geng et al. [35]	NSGA-III, DBEA, eMOEA	Five objective functions.
Hamdy and A. Mohamed [27]	IBPSO, OBPSO	Min cost and max customer satisfaction.
C. Casanova, et al. [47]	FMOPSO	Max Profit and cost, min satisfaction
H Alrezaamiri et al. [8]	PMOABC 2, 4	Min cost and max satisfaction
Marghny et al. [30]	HMABC-DE	Min cost and max satisfaction

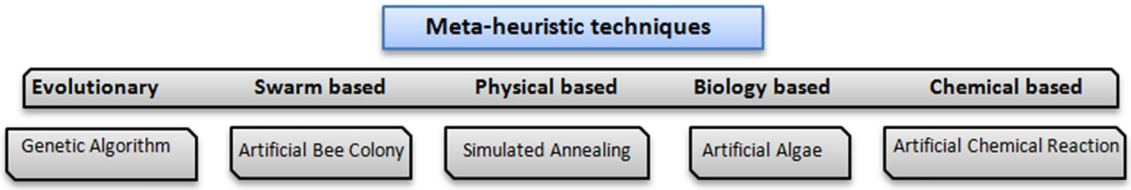

FIGURE 1 Classification of meta-heuristic algorithms

objective problems can get a sole objective or, alternatively, utilise the techniques of weighted aggregation to pool several objectives into one sole objective [48]. If the problem has multiple objectives, the aggregation of those objectives forms the optimisation problem's basic objective, and hence this problem would have exactly one solution. In contrast, problems of multi-objective optimisation (MOPs) have multi-solution [9]. In its turn, MOPs solutions are non-dominated solutions (NDS), called Pareto front. NDS results from the MOP's optimisation for many objective functions simultaneously, while including many conflicting objectives. MOPs can be represented mathematically as a vector holding restrictions (constraints), decision variables, and objective functions. Then, MOP's purpose is to identify the vectors of the best values regarding each objective function that satisfy the relevant restrictions. For example, we have two solutions and k objectives to this problem; the solutions are $X = [x_1, x_2, \cdots, x_k]$ and $Y = [y_1, y_2, \cdots, y_k]$. Just if X is equal or better than Y over all considered objectives $i = 1, 2, \cdots, k$, and X is better than Y in at least one objective, solution X controls solution Y; otherwise, neither solution is controller 'dominant'. Thus, the Pareto front comes to contain the non-dominated solutions set [49].

Figure 2 shows 10 solutions (different and possible), each of which has two minimum objectives (F1 and F2). Solution x_1 is non-dominated, and takes control the solution x_3 because of that $F_1(x_1) < F_1(x_3)$, *and* $F_2(x_1) < F_2(x_3)$, for both objectives this means that solution x_3 is of lower quality than

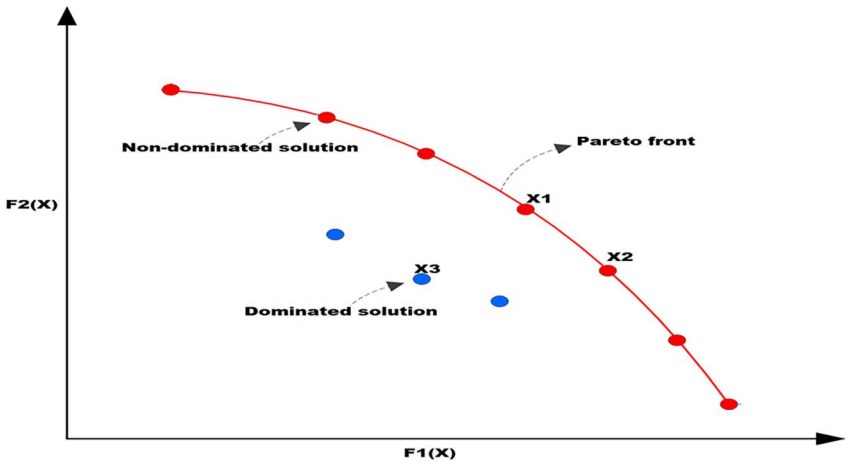

FIGURE 2 Pareto-front solution with dominated and non-dominated

solution x_1. While both solutions x_1 and x_2 are inferiors because of that $F_1(x_1) < F_1(x_2)$ and $F_2(x_2) < F_2(x_1)$, this indicates that every one predominates others in single objective.

3.2 | Formularisation of MO-NRP

The NRP seeks to select a group of customers with different requirements for the next phase (or even a new) of the target software system. NRP's mission is to select the requirements subset that either minimises expense or maximises benefits without exceeding the budget. Zhang et al. [12] In this case, the difficulty is to choose a requirements subset through the release of software system with maximising the satisfaction of customer, and minimising the expected costs. This problem, thus, is multi-objective, and due to the contradicting objectives it is classified as NP-hard.

Let $R = \{r_1, r_2, \ldots, r_n\}$ be the customers' requirements for the software next release. Let $C = \{c_1, c_2, \ldots, c_m\}$ signifies the customer set whose importance to the corporation differs depending on some characteristics such as terms of payment, experience, consistency order, trustfulness, Etc. Consequently, every customer $c_i \in C$ is allocated a weight W_i that measures its significance. The weights set is $W = \{w_1, w_2, \ldots, w_m\}$. Each requirement $r_j \in R$, needs to be implemented using resources like hardware, software tools, and manpower that could be expressed as the costs of development. The cost set is, $\text{cost} = \{\text{cost}_1, \text{cost}_2, \ldots, \text{cost}_n\}$, the development costs. Moreover, different customers might suggest similar requirement. On the other hand, the customer will provide value of priority v_{ij} for every requirement $r_j \in R$, which decide the priority of customer c_i to requirement r_j to be contained in the software's next release. If $v_{ij} = 0$, then r_j was not asked by c_i. Otherwise, $v_{ij} > 0$, c_i asked r_j. All these values v_{ij} are assembled into the priority matrix $V_{m \times n}$ as drawn below:

$$V = \begin{pmatrix} v_{11} & v_{12} & \ldots & v_{1n} \\ v_{21} & v_{22} & \ldots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \ldots & v_{mn} \end{pmatrix}$$

The whole satisfaction of each requirement r_j for a corporation is decided by the priority values' weighted sum of all customers:

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

$S = \{s_1, s_2, \ldots, s_n\}$, describes the satisfaction set for requirements. The issue here is identifying the selection of the optimal requirement subset $X \subseteq$ R out of the list of requirement set R, given that customers; satisfaction is kept high, and costs of development costs is kept as much low within the drawn capacity B. This capacity is drawn for the next release while dependency interactions are maintained. The vector $X = \{x_1, x_2, \ldots, x_n\} : \overrightarrow{X} \subseteq \overrightarrow{R}$, $x_i \in \{0, 1\}$ is a solution vector referring to the target requirements which shall be contained in the next phase. For example, if the $r_j \in R$ is applied, $x_i = 1$, else $x_i = 0$. Requirements interaction, on the other hand, is a factor that employs the relationships among requirements which impact their satisfaction [7]. In our research, two distinct interactions are used: implication interactions $(r_i \Rightarrow r_j)$ if $(r_i \in X \Rightarrow r_j \in X)$; and combination interactions $(r_i \oplus r_j)$ if $(r_i \in X \Leftrightarrow r_j \in X)$. Besides these factors that mark the multi-objective problem, there has been available budget at the beginning of the project. This budget is widely symbolised as percentage β out of the overall costs drawn to finish the next release successfully as follow:

$$B = \beta \sum_{i=1}^{n} \text{cost}_i, \text{ where} (0 < \beta \leq 1) .$$

Hence, the objectives can be presented as following:

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

$$\text{Minimize } E(x) = \sum_{i=1}^{n} \text{Cost}_i * x_i , \tag{3}$$

$$\text{Subject to } \begin{cases} E(x) \leq B : B \text{ is budget} \\ \text{interaction constraints} \end{cases} \tag{4}$$

where $S(x)$ signals the overall satisfaction of customer, and $E(x)$ refers to the net costs of development. After that, the optimisation problem can be reformulated in the following:

$$\text{Minimize } S(x) = \frac{\sum_{i=1}^{n} S_i - \sum_{i=1}^{n} S_i * x_i}{\sum_{i=1}^{n} S_i}, \; S(x) \in [0,1], \tag{5}$$

$$\text{Minimize } E(x) = \frac{\sum_{i=1}^{n} \text{Cost}_i * x_i}{\sum_{i=1}^{n} \text{Cost}_i}, \quad E(x) \in [0,1]. \tag{6}$$

It is worth indicating that the objective functions values are normalised.

4 | GENETIC AND ARTIFICIAL BEE COLONY ALGORITHMS

4.1 | ABC

The Basic Artificial Bee Colony (ABC) algorithm was evolved for optimisation purposes in [50]. ABC is composed of three stages as follows; (1) the employed bees: Every employed bee searches for new solution or food source, and when they discover one, they return to the beehive and dance to signalise their presence; (2) the onlooker bees: Onlooker bees then observe the dance of employed bees to do the source selections depending on the dances to carry out a more thorough investigation in the target area of research; and, (3) scout bees: scout bees look for potential new sources of food in new areas of the search zone, showing new food sources or solutions for the problem [21]. Then, repeats while the best solution is remembered.

4.2 | GA

The Genetic Algorithm (GA) is a population-based algorithm that inspired by Darwinian evolutionary theory [51], which simulated the survival of fitter creatures and their genes. Each solution represents chromosome, and every gene is expressed by parameter. Using a fitness (objective) function, GA evaluates the fitness of every individual in the population. The best solutions are then selected randomly using a selection mechanism (e.g., roulette wheel) to enhance the poor solutions. However, due to the fact that the probability is proportional to fitness (objective value), this operator is highly likely to select the best solutions. The probability of selecting poor solutions also increases the chance of avoiding local optima. This is if good solutions being held captive in a local pool, they can be extracted with the help of other solutions. Individual crossover leads to the exploitation of the 'area' between the two parent solutions. The mutation operator changes genes of chromosomes randomly, maintaining the population's diversity while also increasing GA's exploratory behaviour. The mutation operator, like nature, may result in a significantly better solution, and on the whole would lead other solutions towards the global optimum.

The genetic algorithm generates the initial population $= \left\{x_{i,1}^G, x_{i,2}^G, \cdots, x_{i,D}^G\right\}$, $i = 1, 2, \cdots, N$ for each generation G. The initial population ought to be as much larger to cover the whole parameter space and be generated randomly. By applying crossover and mutation (recombination) operators to previous generation (parents) X_i^G a genetic algorithm creates a new generation (offspring) X_i^{G+1}. Following that, the offspring is placed in the selection step, so the decision over its inclusion in the next generation's population is to be given [14].

4.2.1 | Crossover operator

Single point crossover selects one point randomly, then parents are split at this crossover point, resulting in offspring (children) by gene exchange. The following equation is used to draw the offspring (solution) X_{child}^{G+1}, created by combining two parents $X_i^G \textit{and } X_j^G$ using the crossover operator.

$$\begin{aligned} X_{\text{child1}}^{G+1} &= X_{i=1 \text{ to } k}^G \cup X_{j=k+1 to\ n}^G \text{ and } X_{\text{child2}}^{G+1} \\ &= X_{i=k+1\ to\ n}^G \cup X_{j=1 \text{ to } k}^G. \end{aligned} \tag{7}$$

Where G is the iteration step, the k, as the position of crossover, is selected randomly, $i, j \in \{1, 2, \ldots, N\}$ and $i \neq j$.

4.2.2 | Mutation operator

This operator seeks to change the genes of the offspring, and increase the population's diversity. To avoid premature convergence, and jump out of local or suboptimal solutions, GAs must use this operator. For any solution (parent) X_i^G, generates offspring by alters randomly selected allele, a mutant solution (offspring) X_i^{G+1} is drawn using the following equation.

$$X_i^{G+1} = X_i^G \big| \, x_k^{G+1} = \text{not}\left(x_k^G\right) \tag{8}$$

where k, is the chosen allele or mutation position.

4.2.3 | Selection operator

The greedy criterion is utilised to decide whether the offspring X_{child}^{G+1} should be included in the next generation's population by comparing it to the corresponding parents X_i^G. The selection operator is indicated in the following way:

$$X_i^{G+1} = \begin{cases} X_{\text{child}}^{G+1} \text{ if } f\left(X_{\text{child}}^{G+1}\right) < f\left(X_i^G\right) \\ X_i^G \quad \text{otherwise} \end{cases}. \tag{9}$$

5 | THE PROPOSED HYBRID ALGORITHM

Most population-based optimisation algorithms suffer from the prerequisite of a large population size to avoid premature convergence, this resulting in long processing times that prevent effective exploration and exploitation of the search space. The traditional ABC when compared to common evolutionary algorithms, ABC outperforms in solving various optimisation problems in continuous space. However, it has several obvious disadvantages, such as the fact that the search method of classical ABC can only update one food source at a time, resulting in ABC discarding more of the optimal solutions and being effective at global exploration but poor at local exploitation. To address such limitation, and to maximise the benefits of nature-inspired heuristic approaches while avoiding their drawbacks, hybridisation of multi meta-heuristics is used.

A novel hybrid approach HGABC is presented for multi-objective optimisation problems by combining the ABC and GA algorithms as shown in pseudo-code in Algorithm 1. When these two algorithms are combined, ABC is used as the basic algorithm that makes use of GA operators (mutation, and crossover). The HGABC method has the same structure as the ABC algorithm; however, the new solution generation process generates more solutions each time since the ABC algorithm uses GA operators in all phases. This approach selects optimal solution vectors from the current population-based on dominance values for each generation of ABC to apply GA operations such as mutation and crossover, which allow effective scanning of the search space. As a result, the improved ABC algorithm may keep ABC's global search capabilities while absorbing GA's local optimal solutions to solve the next release problem with greater accuracy.

This section, via the drawn-below Figure 3 & Algorithm 1, illustrates the usage of proposed hybrid algorithm to tackle the MNRP. As earlier stated, combining the ABC method with other viable optimisation methods like the GA can promote its performance. The suggested hybrid method is to utilise the GA algorithm operators for the ABC algorithm. The purpose is to overcome the poor rate of convergence of the ABC algorithm. To enhance the effectiveness of basic ABC, hybridised techniques are leveraged. The suggested approach is a combination that utilises the advantages of both algorithms, namely, ABC and GA. The HGABC algorithm starts by initialising a random population (solutions), and then evaluates them, which will be exploited at the employed bee stage to update the location using information from the random solutions. The present population is then enhanced by utilising the GA's crossover-mutation operators. Following that, in the onlooker bee phase, a probability vector is built based on solutions qualities to decide which solutions will be moved into the next generation of population. These bees search the region surrounding the supplied solution and select the best one.

The algorithm then identifies poor solutions to request the scout bees to find alternatives for those who are unacceptable. Future generations of bees will preserve and use the best solutions retained by the employed bee phase. In addition, to deal with the problem's multi-objective nature, our algorithm includes several strategies from several multi-objective evolutionary algorithms (MOEAs). For instance, from the fast non-dominated sorting genetic algorithm (NSGA-II) [52], the technique of non-dominated solution sorting was utilised. This method divides population into clusters or fronts depending on their dominating relationship. Besides that, from the Pareto archived evolutionary strategy (PAES) [53], the technique of a non-dominated solution archive was used, which archives the best solutions discovered through the algorithm's execution. Algorithm 1 presents the pseudo-code for the multi-objective HGABC to solve the MNRP.

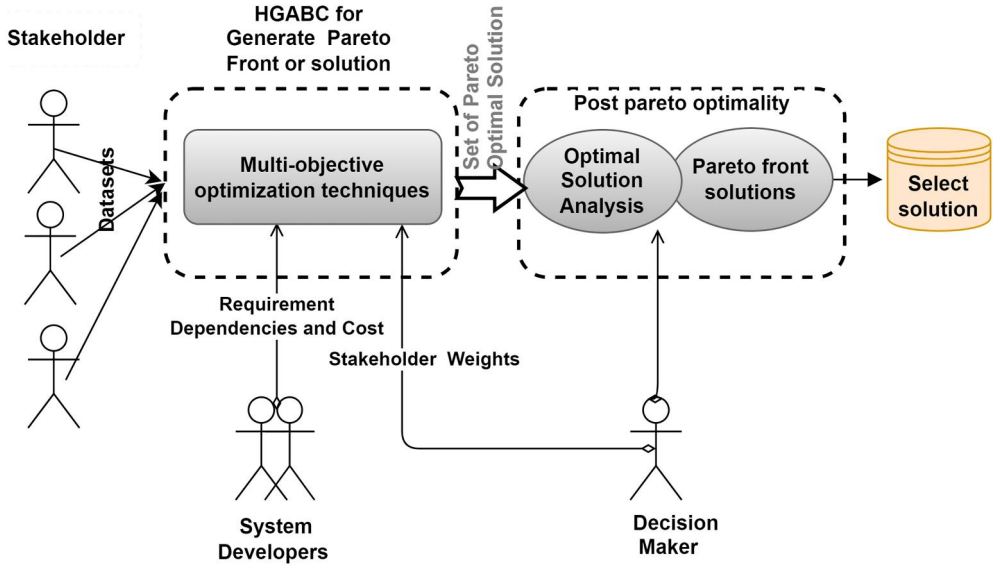

FIGURE 3 Flow diagram of proposed technique (HGABC)

5.1 | Food source representation

First, the data structure, which is shown in Table 2, is a starting point to conduct the HGABC algorithm evaluation. It is comprised of n requirements, with the first part representing each requirement, $r_1, r_2, \ldots, r_n$. The objective function of the problem, which includes total development cost $E(X)$ and total satisfaction $S(X)$, is the second component. The third and final component is the restriction of requirements, and percent of the total budget. Further to this, Table 2 displays examples for encoding of solutions as $X_1, X_2, \ldots, X_n$, indicating the food sources produced by the HGABC algorithm. The solution's coding approach must include the needed information to identify and evaluate actual solutions.

Algorithm 1 The HGABC pseudo code to solve the NRP

```
Output: The values of quality indicators, Figures of pareto front;
//*Initialisation phase
Inputs: Datasets, limit, max-itration, N, B, Pc, Pm
NonDominatedSolutionsarchive ⇐∅
//*Generate initial population and evaluate them *//
X⇐randomGenerationpopulation (NP)
X⇐fastNonDominantedSort (X, NP)
X⇐crowdingDistanceCalculation (X, NP)
while (not stop condition satisfied) do
//* Employed bees phase*//
for i = 1 to NP do
employBee⇐exploreSolution (Xi) by Equation (12);
Mutation and Crossover Operation of GA
Xi⇐greedy selection //* updateEmployBee
endfor
//* Calculate the probability Pi of each solution by Equation (13) *//
probVector⇐generateProbabilityVector (Pi, NP)
// *Onlooker bees phase */
for i = 1 to NP do
Vchosen⇐choose-Employed-Bee (probVector, X, NP)
Mutation and Crossover Operation of GA
Xi⇐greedy selection // *updateOnlookerBee
end for
// *Scout bees phase */
for i = 1 to NP do
if Xi.iterations > limit then
 Xi⇐replace-with-ScoutBee (random)
 end if
end for
//*Sort the colony by quality and update the population *//
X⇐fastNonDominantedSort (X)
X⇐crowdingDistanceCalculation (X)
NP⇐update NonDominatedSolutionsarchive(X)
End while.
```

TABLE 2 Data structure and initial solutions example after encoding

Label	Requirement					Objectives		Constraints	
$\boldsymbol{X}$	r_1	r_2	r_3	…	r_n	Cost	Sat.	IM	Budget
X_1	0	1	0	…	1	$E_1(x)$	$S_1(x)$	$\boldsymbol{r_i \Rightarrow r_j}$	30%
X_2	1	0	0		1	$E_2(x)$	$S_2(x)$		50%
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮		70%
X_n	1	0	1	…	0	$E_n(x)$	$S_n(x)$		100%

As a result, the food source was encoded as a Boolean list $(X = 0/1)$. The X_i is shown whether or not the requirement i was selected for the next release. If the potential solution contains the requirement, then r_i $(i \in [1, n])$ is 1, else it equals 0. The structure of data is used to store all information for the food source and maintains all the requirements in X. Moreover, HGABC algorithm's overarching aim in resolving MNRP is to find the best food source while also having the best value of fitness. Lastly, the two objectives' values (cost and satisfaction) are utilised to validate the food source quality. The equations in Section 3.2 are used to compute these objectives.

5.2 | Initialisation phase

Due to the fact that the ABC approach was established for continuous decision variables, it cannot tackle binary optimisation problems without modification. As a result, as a solution to this problem, binary ABC variants have been produced. The researchers enhanced variants by arranging them differently utilising a Bernoulli technique. The initial solutions or food source $X_i^G = \left\{x_{i,1}^G, x_{i,2}^G, \cdots, x_{i,D}^G\right\}$ in HGABC is a Bernoulli process realisation as:

$$X_{ij}^G = \begin{cases} 1 \text{ if } \alpha_{ij} > 0.5, \\ 0 \text{ if } \alpha_{ij} \leq 0.5, \end{cases} ; \text{ Where } \alpha_{ij} \text{ is a random } \in [0,\ 1]. \tag{10}$$

5.3 | Selection of population

The process of optimisation selects the solutions' population from analogues set of current and advanced populations. In sole-objective instances, the desired solution is that whose fitness value is the highest; but, the dominance strategy is applied in multi-objective scenarios. In our research, we used the swift sorting [52] with the entropy crowding [53] to choose the solutions. The selection was performed first from the F_1,

which is the best non-dominated set and is known as the Pareto front. The remaining population members are then chosen in descending order from the dominated fronts (F_2, F_3, …, F_k). This process is kept repeated until all sets examined. The F_k is therefore regarded as the last dominated set that may be approved. To utilise crowding-entropy sorting to select the population members from which the best solutions are chosen, all members or solutions must first be ranked in decreasing order based on the distance between them.

As mentioned earlier, the HGABC algorithm is a cooperative combination of both GA and ABC algorithms. The goal is to evade the ABC algorithm's low convergence rate. The population-based ABC generates an initial population randomly using Equation (10), which are not the optimal solution, constraints, objectives problem and may require many iterations to get close to it. To handle these constraints, the initial solutions in this phase are subjected to evaluation by a non-dominated, crowding distance, and constraints. The HGABC algorithm produces a set of new solutions after being initialised. During the employed bees' phase, the search for new food sources (solutions) X_i^{G+1} is based on Equation (12). The greedy selection mechanism alternates between new and old source foods to diversify the present population. Because of that an appropriate choice of possible solutions promotes the production of the optimum Pareto front, the most important aim of multi-objective optimisation is to modify the selection process. So, we employ the following selection operator:

$$X_i^{G+1} = \begin{cases} X_i^{G+1} \text{ if } f\left(X_i^{G+1}\right) \prec f\left(X_i^{G}\right) \\ X_i^{G} \quad \text{if } f\left(X_i^{G}\right) \succ f\left(X_i^{G+1}\right) \\ \Phi \quad \text{otherwise} \end{cases} . \tag{11}$$

According to the principle of dominance, a '$\prec$' indicates 'dominate'. If X_i^{G+1} dominates X_i^G this operator replaces the sought-after solution in the existing population pool, and X_i^{G+1} becomes the target solution for the next generation. Moreover, if X_i^G dominates, X_i^{G+1} is kept to reserve elitist solutions; otherwise, both are retained and used to update the solution archive as a new solution. During the evolution process, the archive is used to store non-dominated solutions, and if both are saved, the solution with the shortest crowded distance is picked. The trial solutions that make it through the selection process are regarded as new solutions to upgrade the archive as evolution advances. Iteratively, the update of archive is handled until it is extremely nearer to the Pareto solutions.

5.4 | Phase of employed bee

During the phase of the HGABC's employed bee, every bee travels to construct a new solution or a distinct food source using Equation (12) and evaluate it. It is transformed to a binary searching operator as follows:

$$X_i^{G+1} = \mathrm{mod}\left(\left(X_{i-1,j}^G + \left|X_{ij}^G - X_{i+1,j}^G\right|\right), 2\right) \tag{12}$$

where X_{ij}^G a solution is chosen from the population of the swarm, and mod(·, 2) produces the residual after dividing by 2 for the first variable. To ensure a diverse search across the search space, these solutions must be randomly generated. We utilised Equation (12) for the employed bee seeking phase since it integrates the memory and neighbourhood information considerations. If the equivalent values of two picked solutions are the same as x_{ij}, then the value of x_{ij} remains unaffected. If they are not the same, for example, $x_{ij} = 0$, $x_{i+1,j} = 0$, but $x_{i-1,j} = 1$, then x_{ij} is switched to 1. This increases the diversity of the employed bee phase and can successfully improve exploring abilities. The genetic algorithm's crossover and mutation operators, as well as the selection, are then used.

5.5 | Phase of onlooker bee

The previous phase's information is used in the second stage of HGABC, following the return of employee bees to their hives, they swap information about sources of food with the onlookers. Then, onlookers pick a food source based on the selection likelihood function P_i; thus, the P_i influences the whole search process's exploitability. The likelihood/probability is expressed as follows:

$$P_i = 0.9 * \mathrm{fit}\left(X_i^G\right) / \max\left(\mathrm{fit}\left(X_i^G\right)\right) + 1 \tag{13}$$

where $\mathrm{fit}\left(X_i^G\right)$ indicates the solution X_i^G fitness value. To calculate the fitness value of the solution X_i^G in MOP [54], we employed a distinct metric fitness computation as follows:

$$\mathrm{fit}\left(X_i^G\right) = \left(2^{X_i^G.\mathrm{rank}} + \frac{1}{1 + X_i^G.\mathrm{cd}}\right)^{-1}.$$

According to dominance theory, X_i^G.rank symbolises the rank of the solution (for example, if $X_i^G.\mathrm{rank} = 1$, the solution belongs to the pareto front) and the crowding distance $\left(X_i^G.\mathrm{cd}\right)$ between solutions is denoted by X_i^G.cd. It is crucial to decide which Pareto front the solution belongs on. Then we calculate X_i^G.cd as solutions' crowding distance (solutions belonging to the same front when they are Pareto front the rank = 1). The values of crowding distance and front category for each solution are defined using the non-dominated sorting and crowding distance techniques, respectively. Following the selection of a food source, this algorithm would use the crossover and GA's mutation operators to look for sources of neighbour.

5.6 | Phase of scout bee

The bees, in this phase, are required to improve the capabilities of the proposed algorithm to explore new food sources. If the

food source or solution linked with a worker bee has not been optimised following a specific iterations at this point, it is be eliminated and superseded with a newly produced scout bee, depending on the limit parameter. If the HGABC algorithm's solution X_i^G cannot be enhanced after numerous trials (limit), it will construct a new randomly-picked solution or food source to supersede the one that the bees left in the previous phase in order to jump from local optimal to global optimum solutions. The phase of scout bees is intended to locate new solutions out of previously-unsearched space locations. When this final phase is completed, the algorithm is tested to see if it has reached the stopping condition. If the halting condition is not met, all stages are repeated in their original order. When the halting condition is fulfiled, the algorithm is completed and the obtained Pareto front is returned as its solution.

6 | EXPERIMENTAL RESULTS

In this section, we describe the experimental design, datasets, and HGABC's performance which is evaluated versus the state-of-the-art meta-heuristic methods. The non-dominated solutions (NDS), spread indicator, and hyper-volume (HV) indicator in the Pareto front are utilised to assess performance. Matlab R2015a is used to execute the tests using a computer (Core i7 and 2.6-GHz processor – RAM of 4 GB). Due to the fact that all the algorithms considered for experimental study, in our work, are random algorithms, each trial is repeated 30 times. After 30 cycles, the mean of quality indicators, as well as the standard deviation of results, are presented. The effectiveness of the HGABC algorithm is evaluated on two different real-world datasets, with four constraints for budget analysed on the basis production costs of multi NRP's requirements (100%, 70%, 50%, and 30%). The optimisation process is stopped when the halt conditions are satisfied.

6.1 | Experimental setup

The first dataset [15] composed of 20 requirements along with their costs of development, five customers with their preferences, and 10 interactions constraints. During the elicitation phase, customers may be questioned via questionnaires regarding the priority allocated to each requirement in the next release. These priorities signify the satisfaction of customers for including any requirement in the next release, and each requirement is assigned a rating between 1 and 5 based on its relevance. In fact, a low score implies that the customer is less interested in creating this requirement, whereas a high score shows that the customer is more inclined to have this requirement for the next release. Furthermore, in this dataset, each requirement is connected with a development cost that is evaluated by experts of software system and varies from 1 to 10.

The second dataset, provided by Sagrado et al. [7], contains five customers, their satisfaction ratings, as well as 100 requirements, their development costs, and 42 interaction restrictions between requirements. Additionally, compared to the previous dataset, this one has a substantially more complicated set of constraints, and the range of development costs for each requirement, in this case, is from 1 to 20, and the range of preference numbers is from 1 to 3. Table 3 depicts the interaction constraints or relationships between requirements in both datasets.

Customers are of varying importance levels to any company, thus, in Table 4, the weights given to customers are summarised in the both datasets. So, the numbers range from 1 (the minimally-valued customers) to 5 (the maximally-valued customers). As illustrated in Table 5, the user must also define HGABC variables like limits of budget, size of population (NP), crossover probability Pc, the mutant probability value Pm, and max iteration.

6.2 | Multi-objective optimisation quality indicators

The quality of HGABC algorithm solutions was validated using the following metrics. The hyper-volume measure (HV) is used to assess the convergence and variety of the Pareto front, the number of non-dominated Pareto front solutions identified by an algorithm (NDS), as well as Spread to quantify the diversity of the Pareto front solutions. The area or space coated by the Pareto front Q members using Equation (14) is measured using the HV measure [55]. Higher HV values are preferred for algorithms because the higher the measured value, the higher the quality of the solutions. There are two possibilities when one of the Pareto fronts has a higher HV than the other: Either certain solutions in the superior front dominate those in the lower front, or the superior front solutions are more broadly distributed.

Because the problem had two objectives, two points of reference were required: the minimum and maximum values of the objective functions $R_{\text{mim}}\left(S(x)_{\min}, E(x)_{\min}\right)$ and $R_{\max}\left(S(x)_{\max}, E(x)_{\max}\right)$. Because the HV metric has arbitrary scaling in certain circumstances, it must be normalised before computing the HV for each value of the objective functions. Table 6 lists the reference points that were utilised for each dataset.

$$\text{HV} = \text{volume}\left(\bigcup_{i=1}^{|Q|} v_i\right). \tag{14}$$

The spread metric (Δ-Spread) [55] is the second metric, which estimate the spread or distribution achieved as well as the extent of solutions in the Pareto fronts. In general, Pareto fronts with the smaller spread are desired. Equation (15) calculates the Δ-Spread.

$$\Delta - \text{Spread} = \frac{d_f + d_l + \sum_{i=1}^{N-1}\left|d_i - \overline{d}\right|}{d_f + d_l + (N-1)\overline{d}}, \tag{15}$$

$$\text{Where, } d_i = \sqrt{\left(S(x_{i+1}) - S(x_i)\right)^2 + \left(E(x_{i+1}) - E(x_i)\right)^2}$$

TABLE 3 Interaction constraints in both datasets

Interaction constraints in dataset1							
$r3 \oplus r12$	$r4 \Rightarrow r8$	$r4 \Rightarrow r17$	$r8 \Rightarrow r17$	$r9 \Rightarrow r3$	$r9 \Rightarrow r6$	$r9 \Rightarrow r12$	$r9 \Rightarrow r19$
			$r11 \oplus r13$	$r11 \Rightarrow r19$			
Interaction constraints in dataset2							
$r2 \Rightarrow r24$	$r4 \Rightarrow r5$	$r14 \Rightarrow r32$	$r16 \Rightarrow r40$	$r30 \Rightarrow r52$	$r33 \Rightarrow r58$	$r46 \Rightarrow r68$	$r62 \Rightarrow r83$
$r3 \Rightarrow r26$	$r6 \Rightarrow r7$	$r14 \Rightarrow r34$	$r17 \Rightarrow r43$	$r30 \Rightarrow r53$	$r36 \Rightarrow r61$	$r47 \Rightarrow r70$	$r62 \Rightarrow r84$
$r3 \Rightarrow r27$	$r7 \Rightarrow r30$	$r14 \Rightarrow r37$	$r29 \Rightarrow r49$	$r31 \Rightarrow r55$	$r39 \Rightarrow r63$	$r55 \Rightarrow r79$	$r64 \Rightarrow r87$
$r3 \Rightarrow r28$	$r10 \Rightarrow r32$	$r14 \Rightarrow r38$	$r29 \Rightarrow r50$	$r32 \Rightarrow r56$	$r40 \Rightarrow r64$	$r56 \Rightarrow r80$	$r21 \oplus r22$
$r3 \Rightarrow r29$	$r10 \Rightarrow r33$	$r16 \Rightarrow r39$	$r29 \Rightarrow r51$	$r32 \Rightarrow r57$	$r43 \Rightarrow r65$	$r57 \Rightarrow r80$	$r32 \oplus r33$
			$r46 \oplus r47$	$r46 \oplus r47$			

TABLE 4 Customers weights/importance for company

Customers' weights	cu_1	cu_2	cu_3	cu_4	cu_5
Dataset-1 weights	1	4	2	3	4
Dataset-2 weights	1	5	3	3	1

TABLE 5 Parameters tuning

Parameters	NP	Limit	Pc	Pm	MaxItr.	B% budget
Value	$2n$	0.1 × NP	0.80	0.20	100	30,50,70,100

TABLE 6 The max and min values of the objective functions

Datasets	Dataset2		Dataset1	
Parameters	Cost	Satisfaction	Cost	Satisfaction
R_{max} 100% (cost, satisfaction)	1037	2656	85	893
R_{max} 70% (cost, satisfaction)	720	2194	57	795
R_{max} 50% (cost, satisfaction)	517	1778	41	702
R_{max} 30% (cost, satisfaction)	309	1312	25	555
R_{min} (cost, satisfaction)	0	0	0	0

where $\overline{d}$ is the distances mean between each pair of successive solutions, N is the NDS number of non-dominated solutions, d_i indicates the Euclidean distance between two solutions are adjoining and consecutive, N is the NDS in the PF, and as shown in Figure 4, d_f and d_l are the Euclidean distances between the first and extreme solutions of the optimal Pareto front, as well as the final to the extreme solutions of the optimal PF. lastly, the proposed method was utilised to calculate the quality indicator, NDS. When it comes to capacity, using PFs with a larger NDS is preferable.

6.3 | Experiments and performance analysis

The performance of the HGABC is compared with the relevant state-of-the-art approaches. Tables 7–12 give the findings of the HGABC and other algorithms for assessing quality indicators, including the mean and standard deviation for each trial. Each experiment was run 30 times on each dataset individually. Then,

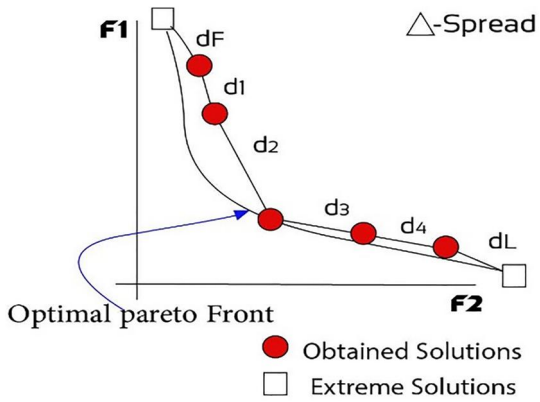

FIGURE 4 Distances between solutions

the results were compared to those obtained by state-of-the-art algorithms (GRASP, DEPT, ACO, and NSGA-II) under four budget limits (100%, 70%, 50%, and 30%) from the total needed development costs. In these tables, the effectiveness of each method is further evaluated by computing the mean values of metrics as well as the standard deviation for eight different instances of the considered problem. It is worth referring that the bolded values in the next tables indicate the best results. The bold font indicates whether or not there are significant differences in the algorithm's results.

6.3.1 | Results: HV indicator

Tables 7 and 8 show results of the HV indicator over both datasets. These tables compare the HGABC algorithm's results to those obtained through other approaches. The findings include the values of mean and standard deviations of the HV and the four budget constraints. The higher the HV indicator value, the higher the findings obtained quality. As a result, the HGABC method regularly outperforms others in the HV metric for each multi NRP instance. Since it combines the ABC and GA properties, it means that HGABC balances exploitation, exploration, and convergence to investigate search space for exploring solutions. It is because it may generate more diversified NDS in the Pareto front, since NDS

TABLE 7 Performance comparison of different algorithms by HV metric—dataset 1

Cost boundary Algorithms	100% Mean% ± Std. dev.	70% Mean% ± Std. dev.	50% Mean% ± Std. dev.	30% Mean% ± Std. dev.
GRASP	-	32.24 ± 0.496**	19.11 ± 0.350**	7.708 ± 0. 37**
ACO	-	38.46 ± 7.08e-2**	23.91 ± 6.75e-2**	10.28 ± 6.57e-2**
NSGA-II	-	32.16 ± 2.30**	20.65 ± 1.60**	9.015 ± 1.12**
DEPT	60.78 ± 1.03e-3**	58.95 ± 2.24e-4**	50.11 ± 1.62e-4	38.88 ± 1.27e-5**
HGABC	**70.76 ± 0.001**	**61.17 ± 0.004**	**50.16 ± 0.0013**	**40.00 ± 0.0013**

TABLE 8 Performance comparison of different algorithms by HV metric—dataset 2

Cost boundary Algorithms	100% Mean% ± Std. dev.	70% Mean% ± Std. dev.	50% Mean% ± Std. dev.	30% Mean% ± Std. dev.
GRASP	-	27.943 ± 7.50e-2**	15.454 ± 6.88e-2**	4.088 ± 8.55e-3**
ACO	-	32.777 ± 1.14e-1**	19.159 ± 9.94e-2**	8.517 ± 6.21e-2**
NSGA-II	-	31.710 ± 8.92e-1**	18.006 ± 5.20e-1**	7.920 ± 2.49e-1**
DEPT	58.03 ± 4.81e-3**	**52.753 ± 4.25e-3****	46.650 ± 7.36e-3	36.51 ± 6.01e-3**
HGABC	**63.716 ± 0.003**	0.52135 ± 0.0039	**49.1 ± 0.0048**	**40.54 ± 0.0054**

offers the best 'desired' solutions. In Table 7, it is noted that HGABC has obtained the best HV results on first dataset over all budget limits, with it is highly superior on 30% and 100%. Just like Table 7, it can be seen that HGABC is still superior over its rivals on the second dataset, in Table 8, with excluding the budget limit of 70%.

Furthermore, the results obtained with HGABC show low dispersion of every budgetary limit investigated, meaning that the total enhancement generated by HGABC is very significant in terms of multi-objective. Due to its greedy nature and local searches, the GRASP algorithm generates the lowest accurate results. It is also restricted in its power to navigate the search space since, unlike the other algorithms; it does not deal with a population.

6.3.2 | Results:Δ−Spread indicator

The values of mean ± standard deviation obtained by using the Δ-Spread indicator are shown in Tables 9 and 10. Smaller Δ-spread indicator readings imply improved performance. As the results in these tables show, the suggested HGABC algorithm delivers better results in most instances, indicating that this algorithm is a good choice to solve this problem and construct an optimum Pareto front from the two data sets. Because this algorithm searches for both global and local solutions and archives them, no high-quality solution is lost when using HGABC algorithm solutions. The algorithm becomes more diversified than previously described approaches by transforming these optimum (or non-dominated) solutions to PFs. Furthermore, since the necessary information for the published techniques was not provided, researchers will not be able to compare the PFs visually using graphs to identify such discrepancies. However, the numerical findings of the HV and Δ-Spread measures show that HGABC can produce higher-quality results.

With excluding the budget limit of 70% on the first dataset, we can clearly include, from both Tables 9 and 10, that HGABC outperforms all its competitors significantly on both datasets.

6.3.3 | Results: NDS indicator

In this case, the primary goal of multi-objective optimisation is to generate a large number of better-distributed solutions on the Pareto front. The greater the number of solutions, the easier it to select the best solution. As a result, rather than optimum solutions for the MNRP are referred to as non-dominated solutions. Tables 11–13 show the values of the mean and standard deviation of NDS obtained for the two datasets using the HGABC method and other techniques described in the literature. Results on both datasets indicate that, from both Tables 11 and 12, that HGABC outperforms all its competitors maximally. These findings demonstrate that the HGABC method delivers the best outcomes, with a greater NDS number.

The HABC-DE technique can explore the solution space more thoroughly, ensuring that all NDS are obtained. As a result, as shown in the tables, this algorithm can generate or detect the greatest NDS in PF. Due to the complexity and size of dataset 2, the non-dominated solutions are greater. Thus, for the two datasets, the disparities between HGABC and the other approaches are more obvious. Figures 5 and 6 depict the HGABC algorithm's Pareto front for datasets 1 and 2, respectively. Technically, HGABC may be more effective than the other algorithms in finding the solution space of this issue. Furthermore, the solutions spread is acceptable, and the

TABLE 9 Performance comparison of different algorithms by Δ-Spread metric—dataset 1

Cost boundary Algorithms	100% Mean% ± Std. dev.	70% Mean% ± Std. dev.	50% Mean% ± Std. dev.	30% Mean% ± Std. dev.
GRASP	-	69 ± 0.060**	73 ± 0.070**	64 ± 0.090**
ACO	-	48 ± 0.020**	52 ± 0.010**	**52 ± 0.030**
NSGA-II	-	80 ± 0.070**	79 ± 0.070**	76 ± 0.090**
DEPT	**40 ± 0.040**	**42 ± 0.030****	48 ± 0.010**	52 ± 0.020**
HGABC	41 ± 1.6938e-16	43 ± 0.002	**46 ± 0.000**	**51 ± 0.000**

TABLE 10 Performance comparison of different algorithms by Δ-Spread metric—dataset 2

Cost boundary Algorithms	100% Mean% ± Std. dev.	70% Mean% ± Std. dev.	50% Mean% ± Std. dev.	30% Mean% ± Std. dev.
GRASP	-	70 ± 0.03**	74 ± 0.04**	60 ± 0.04**
ACO	-	61 ± 0.06**	66 ± 0.06**	68 ± 0.06**
NSGA-II	-	77 ± 0.05**	81 ± 0.06**	80 ± 0.07**
DEPT	44 ± 0.04**	47 ± 0.03**	51 ± 0.03**	56 ± 0.04**
HGABC	**39 ± 0.026**	**46 ± 0.027**	**49 ± 0. 023**	**50 ± 0.020**

TABLE 11 Performance comparison of different algorithms by NDS metric—dataset 1

Cost boundary Algorithms	100% Mean% ± Std. dev.	70% Mean% ± Std. dev.	50% Mean% ± Std. dev.	30% Mean% ± Std. dev.
GRASP	-	20.26 ± 2.18**	17.65 ± 2.22**	11.37 ± 1.47**
ACO	-	20.57 ± 20.57**	17.75 ± 0.61**	13.66 ± 13.66**
NSGA-II	-	11.70 ± 1.90**	11.30 ± 1.82**	9.69 ± 2.09**
DEPT	30.51 ± 2.62**	26.22 ± 2.17**	19.76 ± 0.38**	15 ± 0.00**
HGABC	**46.00 ± 0.00**	**38.00 ± 0.00**	**34.00 ± 0.00**	**23.00 ± 0.00**

TABLE 12 Performance comparison of different algorithms by NDS metric—dataset 2

Cost boundary Algorithms	100% Mean% ± Std. dev.	70% Mean% ± Std. dev.	50% Mean% ± Std. dev.	30% Mean% ± Std. dev.
GRASP	-	120.14 ± 7.27**	75.81 ± 5.81**	57.99 ± 3.66**
ACO	-	70. 98 ± 5.27**	57.68 ± 5.69**	47.12 ± 5.44**
NSGA-II	-	83.32 ± 10.52**	65.54 ± 11.86**	54.34 ± 8.51**
DEPT	144.50 ± 7.16**	139.73 ± 8.32**	123.64 ± 5.20**	110.108 ± 5.45**
HGABC	**340.97 ± 11.4575**	**315.16 ± 15.01**	**237.57 ± 15.67**	**162.43 ± 10.87**

uniform distribution on the Pareto front is less than most other algorithms, and the higher values of NDS and HV indicate that HGABC's superiority.

6.3.4 | Statistical analysis

Tables 13–18 are showing the results of the *t*-test to compare HGABC's quality indicators versus rival algorithms as the following:

In Tables 5–10 the ' *' Indicates to ($p < 0.05$) is statistically significant difference, and '**' Indicates to ($p < 0.01$) is a significant statistical difference. The HGABC algorithm results in Tables 5 and 6 reveal a significant statistical difference in HV between the HGABC and its rivals ($p < 0.01$), as shown in Tables 13 and 14. Exceptionally, there is a non-significant difference between the HGABC and DEPT algorithm in the 50% and 70% cost limitations on dataset1.

Furthermore, when comparing datasets 2, both Tables 9 and 10 reveal a significant difference between the HGABC and

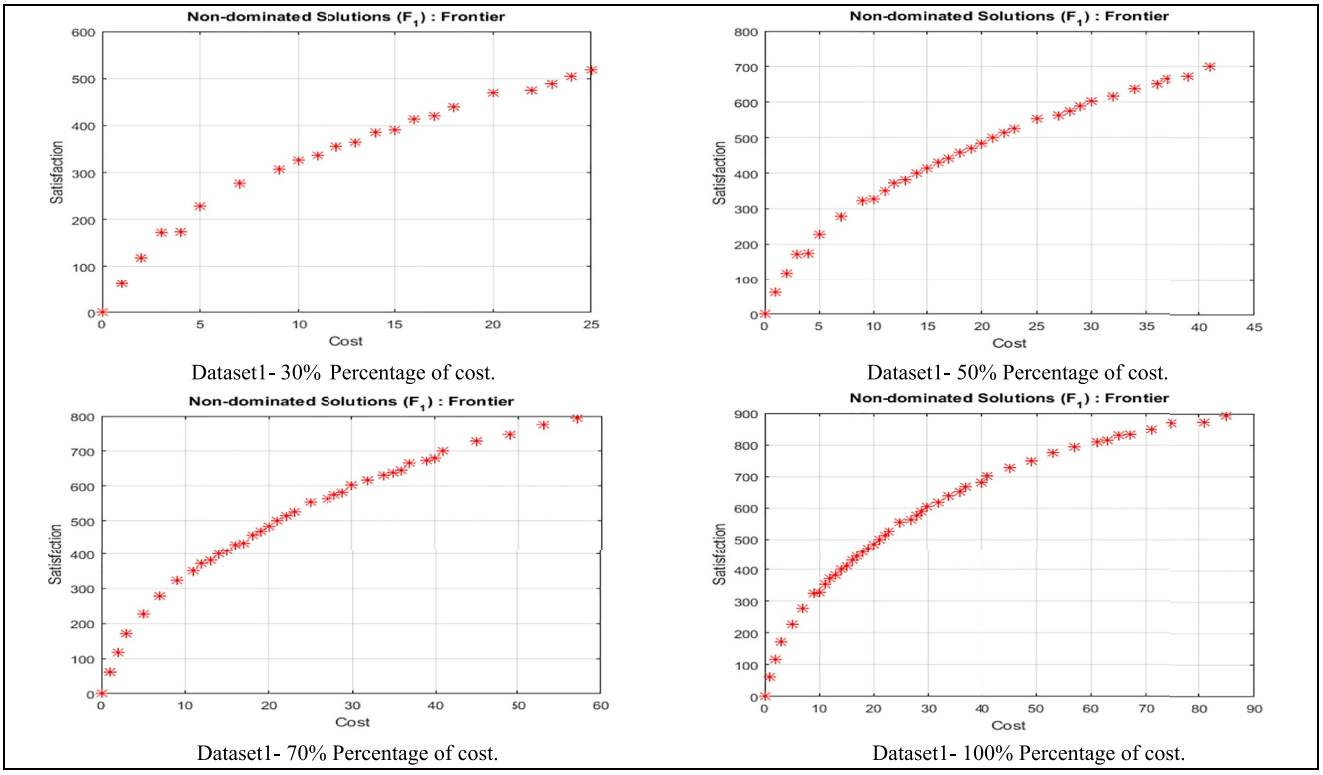

FIGURE 5 Pareto fronts generated by HGABC on dataset1

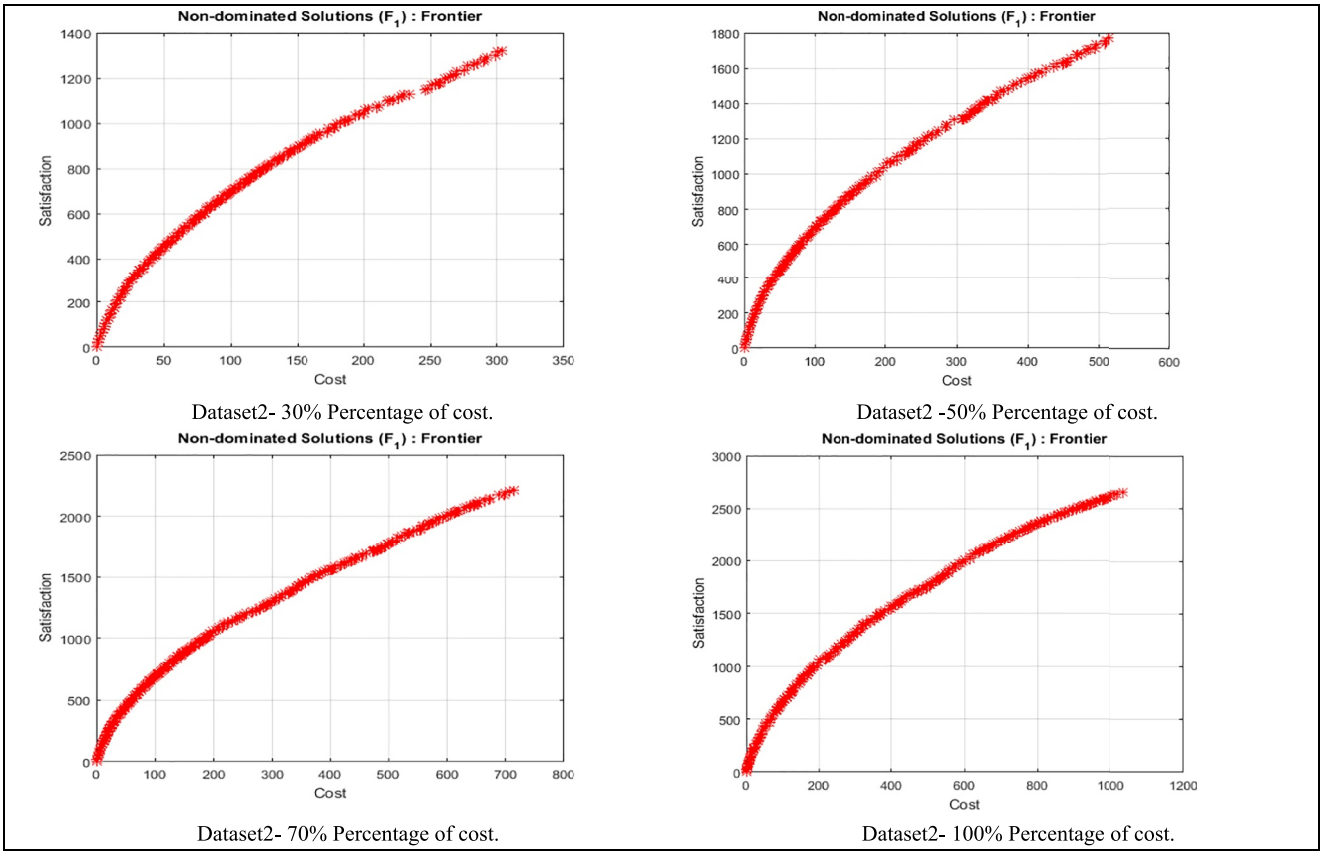

FIGURE 6 Pareto fronts generated by HGABC on dataset2

TABLE 13 Results of statistical significance test of the HV metric—dataset 1

Cost boundary Algorithms	100% *P.* value (*T.* value)	70% *P.* value (*T.* value)	50% *P.* value (*T.* value)	30% *P.* value (*T.* value)
GRASP	-	<0.001** (319.458)	<0.001** (485.905)	<0.001** (478.026)
ACO	-	<0.001** (17.569)	<0.001** (21.300)	<0.001 **(24.777)
NSGA-II	-	<0.001** (69.084)	<0.001** (101.021)	<0.001 **(151.528)
DEPT	<0.001** (53.071)	<0.001** (5.428)	0.866 (0.169)	<0.001** (4.830)
HGABC	**NAN**	**NAN**	**NAN**	**NAN**

TABLE 14 Results of statistical significance test of the HV metric—dataset 2

Cost boundary Algorithms	100% *P.* value (*T.* value)	70% *P.* value (*T.* value)	50% *P.* value (*T.* value)	30% *P.* value (*T.* value)
GRASP	-	<0.001 (15.451)	<0.001 (26.786)	<0.001 (23.352)
ACO	-	<0.001 (78.425)	<0.001 (16.498)	<0.001 **(28.244)
NSGA-II	-	<0.001 (10.678)	<0.001 (32.752)	<0.001 (71.754)
DEPT	<0.001 (6.475)	<0.001 (−4.708)	0.073 (1.823)	<0.001 (3.673)
HGABC	**NAN**	**NAN**	**NAN**	**NAN**

TABLE 15 Results of statistical significance test of the Δ-Spread metric—dataset 1

Cost boundary Algorithms	100% *P.* value (*T.* value)	70% *P.* value (*T.* value)	50% *P.* value (*T.* value)	30% *P.* value (*T.* value)
GRASP	-	<0.001 (3860.213)	<0.001 (−21.126)	<0.001 (−7.912)
ACO	-	<0.001 (11856.792)	<0.001 (−32.863)	0.073 (−1.826)
NSGA-II	-	<0.001 (3300.638)	<0.001 (−25.821)	<0.001 (−15.215)
DEPT	0.974 (0.0323)	<0.001 (7756.791)	<0.001 (−10.954)	0.008 (−2.739)
HGABC	**NAN**	**NAN**	**NAN**	**NAN**

TABLE 16 Results of statistical significance test of the Δ-Spread metric—dataset 2

Cost boundary Algorithms	100% *P.* value (*T.* value)	70% *P.* value (*T.* value)	50% *P.* value (*T.* value)	30% *P.* value (*T.* value)
GRASP	-	<0.001 (−32.570)	<0.001 (−29.677)	<0.001 (−7.524)
ACO	-	<0.001 (−12.487)	<0.001 (−14.491)	<0.001 (−15.588)
NSGA-II	-	<0.001 (−29.881)	<0.001 (−27.276)	<0.001 (−22.571)
DEPT	<0.001 (−5.740)	<0.001 (−1.357)	<0.001 (−2.898)	<0.001 (−7.348)
HGABC	**NAN**	**NAN**	**NAN**	**NAN**

TABLE 17 Results of statistical significance test of number of NDS metric—dataset 1

Cost boundary Algorithms	100% *P.* value (*T.* value)	70% *P.* value (*T.* value)	50% *P.* value (*T.* value)	30% *P.* value (*T.* value)
GRASP	-	<0.001 (44.572)	<0.001 (40.339)	<0.001 (43.33)
ACO	-	<0.001 (4.641)	<0.001 (145.91)	<0.001 (3.745)
NSGA-II	-	<0.001 (75.816)	<0.001 (68.315)	<0.001 (34.881)
DEPT	<0.001 (32.383)	<0.001 (29.734)	<0.001 (205.252)	<0.001 (43817.78)
HGABC	**NAN**	**NAN**	**NAN**	**NAN**

its rivals in most cases, as shown in Tables 15 and 16. All algorithms are close to each other because of the spread or dispersion of solutions in the PF achieved by all methods.

Finally, as indicated in Tables 17 and 18, Tables 11 and 12 reveal a significant difference between the HGABC and its rivals in all cases for NDS.

TABLE 18 Results of statistical significance test of number of NDS metric—dataset 2

Cost boundary Algorithms	100% *P.* value (*T.* value)	70% *P.* value (*T.* value)	50% *P.* value (*T.* value)	30% *P.* value (*T.* value)
GRASP	-	<0.001 (64.047)	<0.001 (53.019)	<0.001 (49.874)
ACO	-	<0.001 (84.071)	<0.001 (59.108)	<0.001 (51.959)
NSGA-II	-	<0.001 (69.278)	<0.001 (47.949)	<0.001 (42.886)
DEPT	<0.001 (79.649)	<0.001 (55.989)	<0.001 (37.800)	<0.001 (23.568)
HGABC	**NAN**	**NAN**	**NAN**	**NAN**

All results in Tables 7–18 show indisputably that the HGABC algorithm outperforms all other algorithms tested, making the HGABC method a highly competitive algorithm. Finally, the algorithm's results are translated into a set of optimal solutions, as shown in Figures 5 and 6 depict Pareto fronts which are generated by HGABC on dataset1 and dataset2 respectively. These figures are significant because they show the number of solutions and the distribution of solutions in the Pareto front. On the other hand, when there are a large number of solutions, we cannot compare the Pareto fronts graphically to notice this difference, but the numerical results are compared using all quality criteria to show that the proposed approach is capable of finding more accurate and quality results.

6.3.5 | Theoretical analysis

Generally speaking, most search optimisation algorithms suffer from long processing times and the inability to completely scan the solution space, resulting in the loss of many optimal solutions to the problem and the inability to effectively explore and exploit the search space.

Concerning the MNRP problem for which our proposed HGABC algorithm comes to overcome, the ABC algorithm, comparing to all addressed algorithms, is considered the most recent work in three stages of improvement, but only one solution is improved in each cycle, which is one of the obvious weaknesses. The HGABC technique therefore comes to solve the problem through hybridisation by introducing the GA operators, improving many solutions in each cycle of the solution, demonstrating that the exploitation and exploration strength of HGABC is much more effective than the original ABC,DE, NSGA, and GRASP algorithms. This feature clearly demonstrates that HGABC outperforms the parallel operation of three ABC algorithms at the same time. Most importantly, all search-based algorithms are distinguished in global search but poor in local search, or vice versa; however, to avoid defects, multiple meta-heuristic hybridisations are used, as it is the case with HGABC.

On the other extreme, despite the superiority of meta-heuristic algorithms over the traditional algorithms, they also have some deficits. For instance, concerning the handling of the multi-objective problems, the speeds of representative population-based algorithms convergence (such as NSGA or DE) is occasionally faster than the speed of ABC's convergence. This is due to the fact that the ABC is not able to use the information properly to identify the promising search path. Meanwhile, the DE and NSGA algorithms' performance is heavily based on the mutation and crossover operator. In many instances, it may be constrained in the local minimum, or the slow convergence. Furthermore, many research have spotted that a clever combination of meta-heuristics would be significantly profitable to promote performance comparing with the single meta-heuristic algorithm. Consequently, combinations of algorithms provides the potential power to further enhance the algorithms performance. HGABC comes thus to confirm these facts evidently as *g* in the experimental results.

In the drawn-above section, the multi-objective ABC was drawn in details. As seen that this algorithm draw its result in a set of solutions. These solutions are named non-dominated solutions (NDSs). As a matter of fact, in the context of multi-objective problems, the solution set constitutes the Pareto front. The NDSs number is one factor to evaluate the multi-objective algorithms. Therefore, the more the number is secured, the better the algorithm is. On the other hand, it can be noted that GRASP is the only algorithm with the worse performance. This may be attributed to the fact that GRASP is a trajectory-based meta-heuristic. Such algorithm thus does not operate well on a population of individuals, limiting the exploration process of the space search.

7 | CONCLUSIONS AND FUTHURE WORK

This work introduced a new hybridised multi-objective algorithm using both genetic and artificial bee colony algorithms, denoted as HGABC. On two most widely-used and publicly-available datasets, we run HGABC to tackle the constrained MONRP. Such hybridisation has been seen capable of improving HGABC's solution performance as well as the speed of convergence. The steps of ABC were modified to give the best performance of HGABC. Both steps, Employed and Onlooker bee, were enhanced by integrating basic GA operators for convergence betterment. Experimental results, including statistical significance tests, using three widely-used metrics showed that the HGABC has been superior over its rivals, with its being maximally superior concerning the NDS metric. In consequence, the HGABC has been seen a promising algorithm, outperforming its rivals concerning both effectiveness and efficiency.

Furthermore, the scout bees' quantity was ameliorated for an effective navigation. Experimental results demonstrated that HGABC algorithm is highly efficient and significantly effective comparing with its rivals. HGABC algorithm generated a better 'non-dominated' solutions (of the lowest spread and biggest hyper-volume) on the Pareto front compared to those generated by current approaches. HGABC algorithm provided effective information that would assist software engineering process for decision makers. Furthermore, considering the statistical tests, it can be deduced that the HGABC has been the best algorithm than its rivals.

In future work, HGABC is planned to be integrated with other evolutionary algorithms that could optimally help solving the NRP. Additionally, it would be motivating to implement HGABC on bigger datasets, and check its expansion to contain more objective functions. To do so, HGABC can be extended in many ways, such as finding a generator for the multi NRP datasets, re-advising the problem, and inventing a extra step in ABC utilising operators of GA. Further to this, we will use clustering algorithms to address the incompleteness of requirements and disagreement among stakeholders in software requirements, and the effect of such techniques for requirement prioritisation in a multi-stakeholder context [56]. The HABC-DE will also be used for e-commerce mobile applications development [57] after the model has been applied in the requirements elicitation in different scenarios, often prioritises functional requirements over quality requirements.

CONFLICT OF INTEREST

Authors declare that they have not had a conflict of interest.

DATA AVAILABILITY STATEMENT

Code and detailed results are publicly available on our GitHub repository: https://github.com/wathiqdukhan and https://github.com/aliamer.

ORCID

Wathiq H. Dukhan https://orcid.org/0000-0002-6836-6493
Ali A. Amer https://orcid.org/0000-0002-2002-948X

REFERENCES

1. Brau, G., Hugues, J., Navet, N.: Towards the systematic analysis of non-functional properties in Model-Based Engineering for real-time embedded systems. Sci. Comput. Program. 156, 1–20 (2018). https://doi.org/10.1016/j.scico.2017.12.007
2. Sabri, O., Alfifi, F.: Integrating knowledge life cycle within software development process to produce a quality software product. In: 2017 International Conference on Engineering and Technology (ICET), pp. 1–7 (2017)
3. Jia, J., et al.: Understanding software developers' cognition in agile requirements engineering. Sci. Comput. Program. 178, 1–19 (2019). https://doi.org/10.1016/j.scico.2019.03.005
4. Del Sagrado, J., et al.: Requirements selection: knowledge based optimization techniques for solving the next release problem. CEUR Workshop Proc. 636, 40–51 (2010)
5. Bagnall, A.J., Rayward-Smith, V.J., Whittley, I.M.: The next release problem. Inf. Software Technol. 43(14), 883–890 (2001). https://doi.org/10.1016/s0950-5849(01)00194-x
6. Zhang, Y., Harman, M., Mansouri, S.A.: The multi-objective next release problem. In: 9th Annu. Conf. Genet. Evol. Comput. GECCO, pp. 1129–1136 (2007)
7. del Sagrado, J., del Aguila, I.M., Orellana, F.J.: Multi-objective ant colony optimization for requirements selection. Empir. Software Eng. 20(3), 577–610 (2015). https://doi.org/10.1007/s10664-013-9287-3
8. Alrezaamiri, H., Ebrahimnejad, A., Motameni, H.: Parallel multi-objective artificial bee colony algorithm for software requirement optimization. Requir. Eng. 25(3), 363–380 (2020). https://doi.org/10.1007/s00766-020-00328-y
9. Coello, C.A.C., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems, vol. 5. Springer (2007)
10. Cai, X., et al.: A decomposition-based coevolutionary multiobjective local search for combinatorial multiobjective optimization. Swarm Evol. Comput. 49, 178–193 (2019). https://doi.org/10.1016/j.swevo.2019.05.007
11. Veerapen, N., et al.: An integer linear programming approach to the single and bi-objective next release problem. Inf. Software Technol. 65, 1–13 (2015). https://doi.org/10.1016/j.infsof.2015.03.008
12. Zhang, Y., Harman, M., Mansouri, S.A.: The multi-objective next release problem. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, pp. 1129–1137 (2007)
13. Ghasemi, M., et al.: Multi-objective whale optimization algorithm and multi-objective grey wolf optimizer for solving next release problem with developing fairness and uncertainty quality indicators. Appl. Intell. 51(1–30), 5358–5387 (2021). https://doi.org/10.1007/s10489-020-02018-2
14. Marghny, M.H., El-Hawary, H.M., Dukhan, W.H.: An effective method of systems requirement optimization based on genetic algorithms. Inf. Sci. Lett. 6(1), 15–28 (2017)
15. Greer, D., Ruhe, G.: Software release planning: an evolutionary and iterative approach. Inf. Software Technol. 46(4), 243–253 (2004). https://doi.org/10.1016/j.infsof.2003.07.002
16. Finkelstein, A., et al.: A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. Requir. Eng. 14(4), 231–245 (2009). https://doi.org/10.1007/s00766-009-0075-y
17. Chaves-González, J.M., Pérez-Toledano, M.A., Navasa, A.: Teaching learning based optimization with Pareto tournament for the multi-objective software requirements selection. Eng. Appl. Artif. Intell. 43, 89–101 (2015). https://doi.org/10.1016/j.engappai.2015.04.002
18. Ranjith, N., Marimuthu, A.: A multi objective teacher-learning-artificial bee colony (MOTLABC) optimization for software requirements selection. Indian J. Sci. Technol. 9(34), 1–9 (2016). https://doi.org/10.17485/ijst/2016/v9i34/95638
19. Del Sagrado, J., Del Águilaa, I.M., Orellana, F.J.: Ant colony optimization for the next release problem a comparative study. In: Proc. - 2nd Int. Symp. Search Based Softw. Eng. SSBSE, pp. 67–76 (2010)
20. Jiang, H., et al.: A hybrid ACO algorithm for the next release problem. In: The 2nd International Conference on Software Engineering and Data Mining, pp. 166–171 (2010)
21. Chaves-Gonzalez, J.M., Perez-Toledano, M.A., Navasa, A.: Software requirement optimization using a multiobjective swarm intelligence evolutionary algorithm. Knowl. Base Syst. 83, 105–115 (2015). https://doi.org/10.1016/j.knosys.2015.03.012
22. Xuan, J., et al.: Solving the large scale next release problem with a backbone-based multilevel algorithm. IEEE Trans. Software Eng. 38(5), 1–18 (2012). https://doi.org/10.1109/tse.2011.92
23. Chaves-González, J.M., Pérez-Toledano, M.A.: Differential evolution with Pareto tournament for the multi-objective next release problem. Appl. Math. Comput. 252, 1–13 (2015). https://doi.org/10.1016/j.amc.2014.11.093
24. Zhang, Y., et al.: Comparing the performance of metaheuristics for the analysis of multi-stakeholder tradeoffs in requirements optimisation. Inf. Software Technol. 53(7), 761–773 (2011). https://doi.org/10.1016/j.infsof.2011.02.001
25. Harman, M., et al.: Exact scalable sensitivity analysis for the next release problem. ACM Trans. Software Eng. Methodol. 23(2), 1–31 (2014). https://doi.org/10.1145/2537853

26. Domínguez-Ríos, M.Á., et al.: Efficient anytime algorithms to solve the bi-objective Next Release Problem. J. Syst. Softw. 156, 217–231 (2019). https://doi.org/10.1016/j.jss.2019.06.097
27. Hamdy, A., Mohamed, A.A.: Greedy binary particle swarm optimization for multi-objective constrained next release problem. Int. J. Mach. Learn. Comput. 9(5), 561–568 (2019). https://doi.org/10.18178/ijmlc.2019.9.5.840
28. Alrezaamiri, H., Ebrahimnejad, A., Motameni, H.: Software requirement optimization using a fuzzy artificial chemical reaction optimization algorithm. Soft Comput. 23(20), 9979–9994 (2019). https://doi.org/10.1007/s00500-018-3553-7
29. Alrezaamiri, H., Ebrahimnejad, A., Motameni, H.: Solving the next release problem by means of the fuzzy logic inference system with respect to the competitive market. J. Exp. \& Theor. Artif. Intell. 32(6), 959–976 (2020). https://doi.org/10.1080/0952813x.2019.1704440
30. Marghny, M.H., et al.: A Hybrid Multi-Objective Optimization Algorithm for Software Requirement Problem. Alexandria Eng. J. (2021)
31. Omar, R., et al.: Genetic K-means adaption algorithm for clustering stakeholders in system requirements. Adv. Mach. Learn. Technol. Appl. Proc. AMLTA, 195 (2021)
32. Aydemir, F.B., et al.: The next release problem revisited. In: 2018 IEEE 26th International Requirements Engineering Conference (RE), pp. 5–16. IEEE (2018)
33. Kumari, A.C., Srinivas, K.: Comparing the performance of quantum-inspired evolutionary algorithms for the solution of software requirements selection problem. Inf. Softw. Technol. 76, 31–64 (2016). https://doi.org/10.1016/j.infsof.2016.04.010
34. Araújo, A.A., et al.: An Architecture based on interactive optimization and machine learning applied to the next release problem. Autom. Softw. Eng. 24(3), 623–671 (2016). https://doi.org/10.1007/s10515-016-0200-3
35. Geng, J., et al.: Supporting many-objective software requirements decision: an exploratory study on the next release problem. IEEE Access. 6(c), 60547–60558 (2018). https://doi.org/10.1109/access.2018.2875122
36. Pirozmand, P., et al.: A novel approach for the next software release using a binary artificial algae algorithm. J. Intell. \& Fuzzy Syst., no. Preprint, 1–15
37. Ferreira, T.N., Vergilio, S.R., Kessentini, M.: Implementing search-based software engineering approaches with Nautilus. In: Brazilian Symposium on Software Engineering, pp. 303–308 (2021)
38. Massobrio, R., et al.: Virtual Savant as a generic learning approach applied to the basic independent Next Release Problem. Appl. Soft Comput. 108, 107374 (2021). https://doi.org/10.1016/j.asoc.2021.107374
39. Nebro, A.J., et al.: Mocell: a cellular genetic algorithm for multiobjective optimization. International Journal of Intelligent Systems 24(7), 726–46 (2009). https://doi.org/10.1002/int.20358
40. de Souza, J.T., et al.: An ant colony optimization approach to the software release planning with dependent requirements. In: International Symposium on Search Based Software Engineering, pp. 142–157 (2011)
41. Xinye Cai, Z.H., Ou, W.: Evolutionary approaches for multi-objective next release problem. Comput. Informatics. 31, 847–875 (2012)
42. Cai, X., Wei, O.: A hybrid of decomposition and domination based evolutionary algorithm for multiobjective software next release problem. In: IEEE Int. Conf. Control Autom. ICCA, pp. 412–417 (2013)
43. Araújo, A.A., Paixão, M.: Machine learning for user modeling in an interactive genetic algorithm for the next release problem. In: International Symposium on Search Based Software Engineering, pp. 228–233 (2014)
44. Da Silva, T.G.N., Rocha, L.S., Maia, J.E.B.: An effective method for MOGAs initialization to solve the multi objective next release problem. In: Mexican International Conference on Artificial Intelligence, pp. 25–37 (2014)
45. Saraiva, R., et al.: Incorporating decision makeres preferences in a multi-objective approach for the software release planning. J. Brazilian Comput. Soc. 23(1), 11 (2017). https://doi.org/10.1186/s13173-017-0060-0
46. Hudaib, A., Masadeh, R., Alzaqebah, A.: WGW: a hybrid approach based on whale and grey wolf optimization algorithms for requirements prioritization. Adv. Syst. Sci. Appl. 18(2), 63–83 (2018)
47. Casanova, C., et al.: Fuzzy Bi-objective particle swarm optimization for next release problem. In: International Conference On Software Engineering And Knowledge Engineering, pp. 509–512 (2019)
48. Jin, Y.: Effectiveness of weighted aggregation of objectives for evolutionary multiobjective optimization: methods, analysis and applications. In: Proc. Int. Conf. Evol. Multi Crit. Optim., pp. 1–32 (2002)
49. Verma, S., Pant, M., Snasel, V.: A comprehensive review on NSGA-II for multi-objective combinatorial optimization problems. IEEE Access. 9, 57757–57791 (2021). https://doi.org/10.1109/access.2021.3070634
50. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Glob. Optim. 39(3), 459–471 (2007). https://doi.org/10.1007/s10898-007-9149-x
51. Holland, J.H.: Genetic algorithms. Sci. Am. 267(1), 66–73 (1992). https://doi.org/10.1038/scientificamerican0792-66
52. Deb, K., et al.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6(2), 182–197 (2002). https://doi.org/10.1109/4235.996017
53. Knowles, J., Corne, D.: The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99, vol. 1, pp. 98–105 (1999)
54. González-Álvarez, D.L., Vega-Rodriguez, M.A.: Hybrid multiobjective artificial bee colony with differential evolution applied to motif finding. In: European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics, pp. 68–79 (2013)
55. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans. Evol. Comput. 3(4), 257–271 (1999). https://doi.org/10.1109/4235.797969
56. Ali, S., et al.: Aspect-based requirements mining technique to improve prioritisation process: multi-stakeholder perspective. IET Softw. 14(5), 482–492 (2020). https://doi.org/10.1049/iet-sen.2019.0332
57. Garcia-López, D., Segura-Morales, M., Loza-Aguirre, E.: Improving the quality and quantity of functional and non-functional requirements obtained during requirements elicitation stage for the development of e-commerce mobile applications: an alternative reference process model. IET Softw. 14(2), 148–158 (2020). https://doi.org/10.1049/iet-sen.2018.5443

How to cite this article: Dukhan, W.H., et al.: Software requirement selection using a combined multi-objective optimisation technique. IET Soft. 16(6), 558–575 (2022). https://doi.org/10.1049/sfw2.12070