Fuzzy Bi-Objective Particle Swarm Optimization for Next Release Problem

Carlos Casanova*, Giovanni Daián Rottoli*, Esteban Schab*, Luciano Bracco*, Fernando Pereyra* and Anabella De Battista*
* Computational Intelligence and Software Engineering Research Group (GIICIS)
Regional Faculty from Concepción del Uruguay.
National Technological University (UTN), Entre Ríos, Argentina.
Email: {casanovac,rottolig,schabe,braccol,pereyraf,debattistaa}@frcu.utn.edu.ar

***Abstract*—In search-based software engineering (SBSE), software engineers usually have to select one among many quasi-optimal solutions with different values for the objectives of interest for a particular problem domain. Because of this, a metaheuristic algorithm is needed to explore a larger extension of the Pareto optimal front to provide a bigger set of possible solutions. In this regard the Fuzzy Multi-Objective Particle Swarm Optimization (FMOPSO), a novel *a posteriori* algorithm, is proposed in this paper and compared with other state-of-the-art algorithms. The results show that FMOPSO is adequate for finding very detailed Pareto Fronts.**

***Index Terms*—Search-Based Software Engineering; Multi-Objective Optimization; Particle Swarm Optimization; Next Release Problem; Fuzzy Logic.**

I. Introduction

Search-Based Software Engineering (SBSE) is a discipline that aims to help software engineers build high quality software through the application of search methods. The main strategy is to change the focus from describing how to develop the software to describing what the software characteristics are. This description has to be codified to be understood by a search algorithm capable of generating new possible products and evaluate their quality using a set of rules provided by the engineer [1].

The problems to be solved using this type of approach are formulated as optimization problems that have, in the majority of the cases, a combinatorial search space and multiple objectives. Because of this, metaheuristics are generaly used, discarding classical methods for optimization such as mathematical programming.

This paper introduces a first version of a novel metaheuristic algorithm named Fuzzy Multi-Objective Particle Swarm Optimization (FMOPSO), designed to deal with this kind of problems by creating a fitness function of multiple objectives using fuzzy weight factors. Different configurations of this fitness function are used to guide the method in the aproximation of the Pareto-optimal front. This new algorithm has been tested on two instances of a well-known Search-Based Software Engineering problem, the Next Release Problem (NRP).

DOI reference number: 10.18293/SEKE2019-082

This problem, first proposed by [2], is aimed at finding a requirement subset to be implemented that satisfy the stakeholders' needs, looking for the maximization of the profit and minimization of the implementation cost [3]. In addition, it may also be restricted by dependencies between requirements such as precedence and simultaneity, among others.

The rest of this paper is organized as follows. In Section II the multi-objective optimization is introduced. Section III describes the Fuzzy Bi-Objective Particle Swarm Optimization algorithm proposed in this paper. Section IV explores the behavior of this proposal and compares it with another well known state-of-the-art algorithms. Finally, Section V contains the conclusions and future work.

II. Multi-Objective Optimization

A commonly used optimization approach consists on selecting as objective function one of the system's attributes and using it to define the (total) order of preferences of the feasible solutions, resulting a *mono-objective* problem. The rest of the attributes modeled as constraints.

On the other hand, the *multi-objective* optimization approach uses several attributes as objective function. These objectives compete against each other defining a partial order on the solution space where there are solutions that are not comparable *a priori*. This partial order is called *Dominance Relation*. The set of all the non-dominated solutions is called *Pareto Front*, and is the result of an optimization method that makes no assumptions about the preferences of the decision-maker.

It is important for the Pareto Front to be as detailed as possible so the decision-maker can select the solution that best fits their needs. Additionally, the Pareto Front provides valuable information about the relation between the competing objectives to use to analyze "What if...?" questions.

III. Fuzzy Multi-Objective Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population-based metaheuristic, which means that in each iteration there is a set (swarm) of possible solutions called particles that move through the search space to find new solutions.

The movement or change in the position of each particle is computed using a movement equation. This equation specifies the way in which the solution is perturbed. Thus, this is a perturbative metaheuristic.

Each particle has access to a fitness function that evaluates the efficiency of the particle's position. This function is the objective function to be optimized. Then, each particle i in the iteration $k+1$ change its position $X_i^{[k+1]}$ according to the particle's velocity $V_i^{[k+1]}$, computed using the movement rule:

$$V_i^{[k+1]} = V_i^{[k]} + w_C \times r_1^{[k+1]} \times \left[b_i^{[k]} - X_i^{[k]}\right] + w_S \times r_2^{[k+1]} \times \left[b_G^{[k]} - X_i^{[k]}\right] \quad (1)$$

where $b_i^{[k]}$ is the best position reached by the particle in previous iterations, $b_G[k]$ is the best position reached by the swarm in previous iterations, r_1 and r_2 are random numbers uniformly distributed in the interval [0,1], and w_C and w_S are prefixed constants. These three terms from the movement equation are, in that order, the inertia term, the memory term, and the cooperation term. Consequently, the position $X_i^{[k+1]}$ is modified as follow:

$$X_i^{[k+1]} = X_i^{[k]} + V_i^{[k+1]} \quad (2)$$

A. *Canonical PSO Implementation*

PSO is an algorithm that was originally designed to work with continuous variables. However, in order to approach combinatorial optimization problems, it can be reformulated to take into account the combinatorial aspects of the problem into the movement equation. It is because of this that the Canonical PSO model comes up [4], specifying the necessary and sufficient conditions to use PSO in any domain. According to it, the representation of the following items has to be defined: (a) the position and velocity of a particle; (b) a scalar-valued or vector-valued fitness function; (c) a total or partial order relation on the fitness function codomain; (d) the binary operations:

- $subtraction(position, position) \xrightarrow{-} velocity$
- $external\_product(real\_number, velocity) \xrightarrow{\cdot} velocity$
- $addition(velocity, velocity) \xrightarrow{\oplus} velocity$
- $displacement(position, velocity) \xrightarrow{+} position$

For the Next Release Problem, binary vectors are used to represent the position and velocity of a particle [5]. Each vector has a lenght equal to the number of requirements. In the position vector, each component x_j represents the decision of including (1) or excluding (0) requirement j. In the velocity vector instead, the value 1 in the position v_j means the value in x_j has to change to its complement.

The binary operations are described in the following sections.

B. *Fuzzy Single-Objectivization*

It is possible to compose a single objective function of multiple objectives using fuzzy sets [6]. A fuzzy decision set $\tilde{D}$ can be built from nO objectives $\tilde{O}_1, \tilde{O}_2, \ldots, \tilde{O}_{nO}$ using the intersection according to the triangular norm t, that represents the objectives *confluence*. Thus, $\tilde{D}$ is defined as follow:

$$\tilde{D} = \tilde{O}_1 \cap_t \tilde{O}_2 \cap_t \cdots \cap_t \tilde{O}_{nO} \quad (3)$$

then, using an in-order notation:

$$\mu_{\tilde{D}}(x) = \mu_{\tilde{O}_1}(x)\, t\, \mu_{\tilde{O}_2}(x)\, t \ldots t\, \mu_{\tilde{O}_{nO}}(x) \quad (4)$$

It is reasonable in the majority of the cases to choose the option x with the maximum membership degree to the fuzzy decision set.

Furthermore, [7] presents a mechanism to give to each objective differentiated preferences by affecting the membership function using an exponential weighting factor ρ in order to contract (increase), with $\rho > 1$, or dilate (decrease), with $\rho < 1$, the relevance of each objective.

Let ρ_1, $\rho_2, \ldots, \rho_{nO}$ be the exponential weighting factors associated with each objective such that $\sum_{i=1}^{nO} \rho_i = nO$. In consequence, $\tilde{D}$ is defined as follows:

$$\mu_{\tilde{D}}(x) = \mu_{\tilde{O}_1}^{\rho_1}(x)\, t\, \mu_{\tilde{O}_2}^{\rho_2}(x)\, t \ldots t\, \mu_{\tilde{O}_{nO}}^{\rho_{nO}}(x) \quad (5)$$

The problem addressed in this paper contemplates two objectives: the implementation cost C, and the profit B given by the stakeholders' satisfaction. The membership function of the fuzzy number associated to the cost is:

$$\mu_C(X, \rho_C) = \begin{cases} 1 & if \quad C(X) \le C_{min} \\ \left(\frac{C(X) - C_{max}}{C_{min} - C_{max}}\right)^{\rho_C} & if \quad C_{min} < C(X) < C_{max} \\ 0 & if \quad C(X) \ge C_{max} \end{cases} \quad (6)$$

where $C(X)$ is the cost of implementing the solution X, C_{min} and C_{max} are the minimum and maximum values of reference for the cost variable, and ρ_C is the prefixed weighting factor.

In the same way, the membership function of the fuzzy number associated to the profit is:

$$\mu_B(X, \rho_B) = \begin{cases} 1 & if \quad B(X) \ge B_{max} \\ \left(\frac{B(X) - B_{min}}{B_{max} - B_{min}}\right)^{\rho_B} & if \quad B_{min} < B(X) < B_{max} \\ 0 & if \quad B(X) \le B_{min} \end{cases} \quad (7)$$

where $B(X)$ is the profit given by the solution X, B_{min} and B_{max} are the minimum and maximum values of reference for the profit variable, and ρ_B is the prefixed weighting factor.

If the stakeholders' preferences are not known and it is required to find the whole Pareto Front, this method is still suitable by using multiple weighting factor values to specialize the search in multiple regions of the search space at the same time.

This strategy may be implemented by assigning different values for the weighting factors to disctinct groups of particles, so each group explores a different region of the space (Fig.

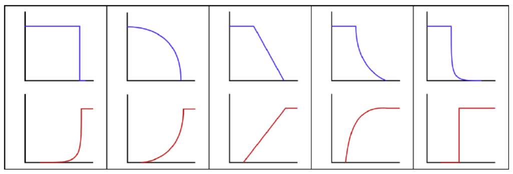
Fig. 1: Configurations of the fuzzy numbers for different groups (by column) with two objectives: Profit (blue): $\bar{\rho} = (0, 0.5, 1, 1.5, 2)$ and Cost (red): $\bar{\rho} = (2, 1.5, 1, 0.5, 0)$.

1). Therefore, the fitness function evaluated for a position X in group j is:

$$\mu_D(X,j) = min\left(\mu_C\left(X, \frac{2(j-1)}{(n-1)}\right), \mu_B\left(X, 2-\frac{2(j-1)}{(n-1)}\right)\right) \quad (8)$$

C. Topology and individual best updates

A key aspect to consider when using PSO is to define how the particles communicate with each other. The component that rules this communication is called *topology*.

Assuming that there are n different weighting factors values or, in other words, n groups into the swarm, the particles assigned to the j^{th} group share a single best group position denoted b_j that is used into the memory term of the movement equation. If each group has the same number of particles, say m, the swarm is thus arranged in a matrix $M^{m\times n}$. Consequently, each particle can be denoted with a double subindex (i, j), with $1 \leq i \leq m$ and $1 \leq j \leq n$.

Additionally, each particle has access to the best position reached by its adjacent groups b_{j-1} and b_{j+1}, if they exist. In order to determine the position to be used in the cooperation term, the one with the highest value is selected.

Furthermore, an additional rule has to be taken into account to update the best group solutions b_j. When a new position $X_{ij}^{[k+1]}$ is computed, it is evaluated according to at most three version of the fitness function: $\mu_D(X_{ij}^{[k+1]}, j-1)$, $\mu_D(X_{ij}^{[k+1]}, j)$ and $\mu_D(X_{ij}^{[k+1]}, j+1)$.

As each particle is affected by its neighbours, this particle can find a good solution to its own fitness function version or to the version of its neighbors. Thus, when it is time to update the best group positions, b_j, the new position to be evaluated are $X_{il}^{[k+1]}$, with $i \in \{1, \ldots, m\}$ and $l \in \{max(1, j-1), \ldots, min(j+1, n)\}$.

D. Binary Operators

The binary operations used in the FMOPSO implementation are Xor, And & Or from Boole's Algebra, denoted by $\oplus$, $\cdot$, $+$, respectively. In this approach, the movement equation leave aside the inertia term:

$$V_{ij}^{[k+1]} = r_1^{[k+1]} \cdot \left(b_j^{[k]} \oplus X_{ij}^{[k]}\right) + r_2^{[k+1]} \cdot \left(b_{Vj}^{[k]} \oplus X_{ij}^{[k]}\right) \quad (9)$$

This rule can compute an unfeasible position, thus, the Xor operation related to the movement is performed in increasing order adding predecessors or removing successors, as appropriate.

Finally, if the particle does not change its position in two consecutive iterations, a mutation is applied on a random component to invert its value, adding predecessors or removing successors too.

IV. Experimental Design

A study on two NRP instances obtained from the *classic* set of instances of [8] were performed. The first one, named *nrp1* has 140 requirements and 100 stakeholders. The second one, named *nrp2*, has 620 requirements and 500 stakeholders. Both instances have precedence relations between requirements. The reference values B_{min} and C_{min} were set to 0 for both instances. The values for B_{max} were 2909 for nrp1 and 14708 for nrp2. The values for C_{max} were 787 for nrp1 and 4758 for nrp2. These values were found through an exact mono-objective optimization using Branch & Bound.

Several tests were conducted on the aforementioned instances using the proposed FMOPSO algorithm. The results were compared with two widely used state-of-the-art algorithms: NSGA-II (Non-Dominated Sorting Genetic Algorithm) [9] and IBEA (Indicator-Based Evolutionary Algorithm) [10].

The metrics used for the comparison were the Hypervolume (HV), to evaluate the quality of the solution, and the Pareto Front Size (PFS), to assess the population diversity [11].

The parameters were tuned ad-hoc by executing each algorithm 5 times with many different configurations to find the best set of values. The best configurations were selected according to the median of the Hypervolume. With this best setting for each algorithm, 10 more executions were performed to obtain a representative value distribution of the selected metrics. All the partial results can be found in [12].

The software used for the experiment was written in C++ using and extending the ParadisEO library [13]. FMOPSO was implemented by the authors. The ParadisEO's versions of NSGA-II and IBEA were used. The crossover operator used to produce two new individuals is equivalent to the boolean operators $+$ and $\cdot$. The mutation operator is the same used in the FMOPSO, as mentioned before. The code can be found in https://github.com/casanovac/FMOPSO.

The running time was the same for all the algorithms: 30 seconds for *nrp1* and 60 seconds for *nrp2*.

A. Results

The results (Table I) show that FMOPSO is a better option regarding diversity, this is, the values for the PFS metric, obtained using this algorithm, are higher than those obtained using the state-of-the-art alternatives. However, FMOPSO is overtaken by IBEA for the Hypervolume, but shows a better performance than NSGA-II, a widely used algorithm in Search-Based Software Engineering (Fig. 2).

Additionally, Figure 3 shows that NSGA-II does not cover the Pareto Front uniformly: there are many regions that were not explored. IBEA, on the other hand, generates the best non-dominated solutions, but with many gaps too. FMOPSO, however, scans the whole front in a uniform way, with a low

TABLE I: Result Descriptors

	Problem	NRP1			NRP2		
Metric	Algorithm	FMOPSO	IBEA	NSGAII	FMOPSO	IBEA	NSGAII
HV	Min	0.5651	0.5586	0.5256	0.4550	0.4967	0.3952
	1^{st} Q.	0.5673	0.5614	0.5323	0.4570	0.4977	0.3996
	Median	0.5683	0.5652	0.5347	0.4592	0.4981	0.4032
	Mean	0.5679	0.5631	0.5351	0.4594	0.4982	0.4027
	3^{rd} Q.	0.5689	0.5646	0.5392	0.4612	0.4990	0.4052
	Max	0.5695	0.5668	0.5422	0.4663	0.4996	0.4106
PFS	Min	227	85	100	338	45	242
	1^{st} Q.	234	85	105.5	347	47.25	251.5
	Median	239	87.5	112	353.5	48	259
	Mean	240	88.22	110.2	353.2	47.9	257.8
	3^{rd} Q.	250	90.5	115	357,5	49	263
	Max	253	95	116	369	50	271

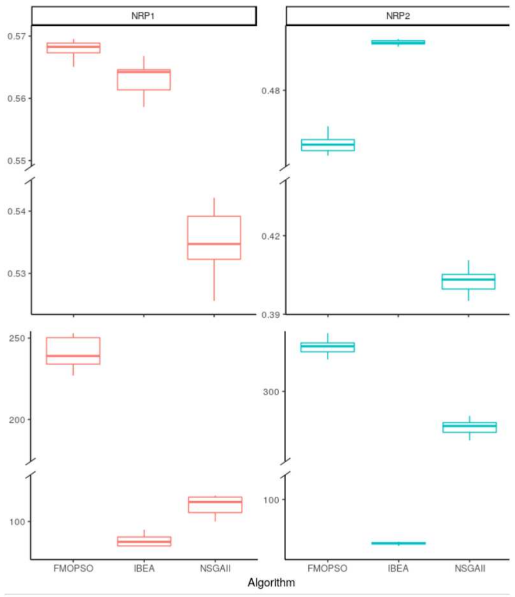

Fig. 2: Box plots with the distribution of the HV (Top) and the PFS (Bottom) on two instances of the Next Release Problem.

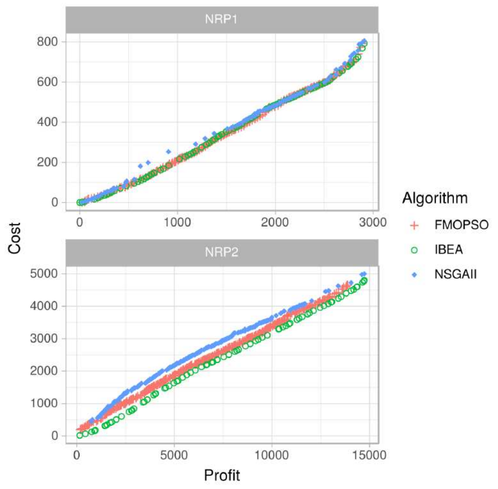

Fig. 3: Pareto Fronts from a random execution of the algorithms

quality loss, as can be seen in the scatterplot related to the second instance of the problem.

According to this study, FMOPSO is an adequate alternative when the purpose is to obtain a very detailed Pareto Front, with a very good level of quality.

V. Conclusions and Future Work

In this paper, the Bi-Objective Next Release Problem has been presented at a conceptual level. A novel *a posteriori* metaheuristic algorithm that aproximates the Pareto Front for the aforementioned bi-objective NRP was also presented: FMOPSO. This algorithm composes a fitness function by using mixed weights for different groups of particles from the swarm.

Technical and theoretical aspects on this algorithm have been presented. A comparative study was performed comparing this proposal with two different state-of-the-art algorithms.. The obtained results are promising and encourage to keep looking forward to the study of many yet unexplored aspects of this new algorithm.

It is necessary to adapt FMOPSO to be used in an interactive way. Extending the FMOPSO to deal with other SBSE problems that include more than two objectives should also be a goal.

Acknowledgements

Thanks to the National Technological University from Argentina (UTN) through the project SIUTICU0005297TC: “Preference-based Multi-objective Optimization Approaches applied to Software Engineering”, and to Kevin-Mark Bozell Poudereux, for proofreading the translation.

References

[1] M. Harman and B. F. Jones, “Search-based software engineering,” *Information and Software Technology*, vol. 43, no. 14, pp. 833–839, 2001.

[2] A. J. Bagnall, V. J. Rayward-Smith, and I. M. Whittley, “The next release problem,” *Information and Software Technology*, 2001.

[3] Y. Zhang, M. Harman, and S. A. Mansouri, “The multi-objective next release problem,” in *Proceedings of the 9th annual conference on Genetic and evolutionary computation - GECCO ’07*. ACM Press, 2007, p. 1129.

[4] M. Clerc, *Particle Swarm Optimization*, 2006.

[5] F. Afshinmanesh, A. Marandi, and A. Rahimi-Kian, “A novel binary particle swarm optimization method using artificial immune system,” in *Computer as a Tool, 2005. EUROCON 2005. The International Conference on*. IEEE, 2005, pp. 217–220.

[6] R. E. Bellman and L. A. Zadeh, “Decision-making in a fuzzy environment,” *Management science*, vol. 17, no. 4, pp. B–141, 1970.

[7] R. R. Yager, “Multiple objective decision-making using fuzzy sets,” *International Journal of Man-Machine Studies*, vol. 9, no. 4, pp. 375–382, 1977.

[8] J. Xuan, H. Jiang, Z. Ren, and Z. Luo, “Solving the Large Scale Next Release Problem with a Backbone-Based Multilevel Algorithm,” *IEEE Transactions on Software Engineering*, vol. 38, no. 5, pp. 1195–1212, sep 2012.

[9] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, “A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii,” in *International conference on parallel problem solving from nature*. Springer, 2000, pp. 849–858.

[10] E. Zitzler and S. Künzli, “Indicator-based selection in multiobjective search,” in *International Conference on Parallel Problem Solving from Nature*. Springer, 2004, pp. 832–842.

[11] C. Grosan, M. Oltean, and D. Dumitrescu, “Performance metrics for multiobjective optimization evolutionary algorithms,” in *Proceedings of Conference on Applied and Industrial Mathematics (CAIM), Oradea*, 2003.

[12] C. Casanova, A. De Battista, E. Schab, G. D. Rottoli, L. Bracco, and F. Pereyra, “Execution Results Bi-Objective NRP with FMOPSO-IBEA-NSGAII,” feb 2019. [Online]. Available: https://dx.doi.org/10.17632/d7y3x97hsx.1

[13] S. Cahon, N. Melab, and E.-G. Talbi, “Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics,” *Journal of heuristics*, vol. 10, no. 3, pp. 357–380, 2004.