2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A Comparison Study of Binary Multi-Objective Particle Swarm Optimization Approaches for Test Case Selection

Luciano S. de Souza, Ricardo B. C. Prudêncio and Flávia de A. Barros

***Abstract*— During the software testing process many test suites can be generated in order to evaluate and assure the quality of the products. In some cases the execution of all suites cannot fit the available resources (time, people, etc). Hence, automatic Test Case (TC) selection could be used to reduce the suites based on some selection criterion. This process can be treated as an optimization problem, aiming to find a subset of TCs which optimizes one or more objective functions (i.e., selection criteria). The majority of search-based works focus on single-objective selection. In this light, we developed mechanisms for functional TC selection which considers two objectives simultaneously: maximize requirements coverage while minimizing cost in terms of TC execution effort. These mechanisms were implemented by deploying multi-objective techniques based on Particle Swarm Optimization (PSO). Due to the drawbacks of original binary version of PSO we implemented five binary PSO algorithms and combined them with a multi-objective versions of PSO in order to create new optimization strategies applied to TC selection. The experiments were performed on two real test suites, revealing the feasibility of the proposed strategies and the differences among them.**

I. Introduction

The importance of software testing in the software development process has grown due to the need of high quality products. Software testing aims to assure quality and reliability of the products. However, it is a very expensive activity, sometimes reaching 40% of the final development cost [1]. In this scenario, automation seems to be the key solution for improving the efficiency and effectiveness of the testing process.

We can identify in the related literature two main testing approaches: White Box (structural) or Black Box (functional) testing. In both approaches, the testing process relies on the (manual or automatic) generation and execution of a Test Suite (TS). The aim is to provide a good coverage of an adopted test adequacy criterion (e.g., code coverage, requirements coverage) in order to satisfy the test goals. Tools for automatic TC generation usually deliver large test suites, trying to cover all possible scenarios (e.g., [2]). Even manually created test suites may be large when a good level of coverage is accomplished. Although it is desirable to fully satisfy the test goals, the execution of large suites is a very expensive task, demanding a great deal of the available resources (time and people) [3].

Large test suites usually contain some redundancies (i.e., two or more TCs covering the same requirement/piece of code). Hence, it is possible to reduce the suite in order to fit the available resources. The task of reducing a test suite based on a given selection criterion is known as *Test Case selection* [4].

TC selection is a hard task, since there may be a large number of TC combinations to consider when searching for an adequate TC subset. A very effective approach to deal with this problem relies on the use of optimization techniques, which is the focus of our research (see [5], [6], [7], [8], [9]). This approach aims to select a TC subset that optimizes a given objective function (i.e., the given selection criterion). Regarding multi-objective selection, we can cite the use of evolutionary approaches [6], [10] and the use of Particle Swarm Optimization (PSO) techniques in our previous work [11].

In [11], we investigated the multi-objective TC selection considering both the requirements coverage (quality) and the execution effort (cost) of the selected subset of TCs as objectives of the selection process. In [11], we developed Binary Multi-Objective PSO (BMOPSO) techniques for TC selection by combining: (1) the binary version of PSO proposed in [12], since the TC selection problem by definition has a binary search space; and (2) the MOPSO [13] and MOPSO-CDR [14], which deals with multi-objective problems. Despite the good results obtained by the BMOPSO techniques on a case study, some limitations can be pointed out, such as the search operators adopted to binary search spaces. Although we adopted a classical binary PSO version (originally proposed in [12]), there are recent alternatives that can be more effective, supported by empirical results obtained in single objective problems.

In this light, in the current work we investigated different single objective PSO algorithms for binary search spaces and combined them with a multi-objective strategy in order to create new binary multi-objective algorithms. More specifically, five different binary PSO techniques were implemented and extended to multi-objective optimization. The developed algorithms were applied to the TC selection problem aiming to provide to the user a set of solutions (test suites) with different values of requirements' coverage *versus* execution effort. The user could then choose the solution that best fits the available resources for executing the tests. It is important to highlight that although the focus of our research is the TC selection problem the proposed techniques can also

Luciano S. de Souza is with the Department of Informatics of Federal Institute of Education Science and Technology of North of Minas Gerais (IFNMG), Pirapora, Minas Gerais, Brazil (email: luciano.souza@ifnmg.edu.br). Ricardo B. C. Prudêncio and Flávia de A. Barros are with the Center of Informatics (CIn), Federal University of Pernambuco (UFPE), Recife, Pernambuco, Brazil (email: {rbcp, fab}@cin.ufpe.br).

This work was partially supported by FAPEMIG, National Institute of Science and Technology for Software Engineering (INES www.ines.org.br), CNPq, CAPES and FACEPE.

be applied to binary multi-objective optimization in other contexts.

The next section explains the Particle Swarm Optimization (PSO) algorithm and its binary versions. After that, section III details the developed multi-objective strategies. Section IV details the experiments performed to evaluate the proposed algorithms and presents the obtained results. Finally, Section V presents some conclusions and future work.

II. Binary Particle Swarm Optimization

Particle swarm optimization (PSO) was originally developed by Eberhart and Kennedy [15]. The PSO is a population based search algorithm based on the simulation of the social behavior of birds and it has been successfully applied in many research and engineering areas [16]. Also, PSO has shown to be a simple and efficient algorithm compared to other search techniques, including for instance the widespread Genetic Algorithms [17].

The conventional continuous PSO algorithm starts its search process with a random population (also called swarm) of *particles*. Each particle represents a candidate solution for the problem being solved and it has four main attributes:

1) the position ($\mathbf{t}$) in the search space (each position represents an individual solution for the optimization problem);
2) the current velocity ($\mathbf{v}$), indicating a direction of movement in the search space;
3) the best position ($\hat{\mathbf{t}}$) found by the particle (the cognitive component);
4) the best position ($\hat{\mathbf{g}}$) found by the particle's neighborhood.

The velocity and position of each particle is updated by using equations 1 and 2

$$\mathbf{v} = \omega\mathbf{v} + c_1 r_1(\hat{\mathbf{t}} - \mathbf{t}) + c_2 r_2(\hat{\mathbf{g}} - \mathbf{t}) \tag{1}$$

where ω represents the inertia factor; r_1 and r_2 are random values in the interval [0,1]; c_1 and c_2 are constants which control the trade-off between the particle's memory and the social guide impact on the particle.

$$\mathbf{t} = \mathbf{t} + \mathbf{v} \tag{2}$$

For a number of iterations, the particles fly through the search space, being influenced by their own experience $\hat{\mathbf{t}}$ (particle memory) and by the experience of their neighbors $\hat{\mathbf{g}}$ (social guide). Particles change position and velocity continuously, aiming to reach better positions and to improve the objective functions considered.

The conventional continuous PSO and most of its variants were originally developed for continuous valued spaces but many problems are, however, defined for discrete valued spaces where the domain of the variables is finite. So, some authors [12], [18], [19], [20] and [16] have proposed binary versions of PSO in order to deal with these kind of problems. But, there is not yet a consensus about which one of their approaches are better than others.

Since the solutions of TC selection problem could be represented as binary vectors this work combined five versions of binary PSO with the MOPSO-CDR approach, creating five new multi-objective strategies aiming to verify which one is better when applied to the TC selection problem.

Following the versions of binary PSO are explained and, after that, the multi-objective approaches are presented.

A. Discrete Binary PSO

Kennedy and Eberhart (see [12]) developed the first discrete binary version of PSO. In the discrete binary PSO, the velocity of a particle is used as a probability to determine whether a bit should be changed to one.

The eq. (2) is modified to:

$$\mathbf{t} = \begin{cases} 1, \text{ if } r_3 \leq sig(\mathbf{v}) \\ 0, \text{ otherwise} \end{cases} \tag{3}$$

where r_3 is a random number sampled in the interval [0,1] and sig($\mathbf{v}$) is defined as:

$$sig(\mathbf{v}) = \frac{1}{1 + e^{-\mathbf{v}}} \tag{4}$$

Equations (3) and (4) were proposed in order to certify that the new positions are binary vectors. The position value $\mathbf{t}$ tends to 1 when the velocity assumes higher values. In turn, $\mathbf{t}$ tends to 0 for lower values of velocity;

B. Novel binary particle swarm optimization

Khanesar *et al* (see [18]) introduces two velocity vectors for each particle v^0 and v^1 where the first is the probability of the bits of the particle change to zero and the second is the probability of change to one.

Two temporary values d^1 and d^0 are also introduced and are updated as:

If $\hat{\mathbf{t}} = 1$ Then $d_1^1 = c_1 r_1$ and $d_1^0 = -c_1 r_1$

If $\hat{\mathbf{t}} = 0$ Then $d_1^0 = c_1 r_1$ and $d_1^1 = -c_1 r_1$

If $\hat{\mathbf{g}} = 1$ Then $d_2^1 = c_2 r_2$ and $d_2^0 = -c_2 r_2$

If $\hat{\mathbf{g}} = 0$ Then $d_2^0 = c_2 r_2$ and $d_2^1 = -c_2 r_2$

These temporary values are used to update the v^0 and v^1 as follows:

$$v^0 = \omega v^0 + d_1^0 + d_2^0 \tag{5}$$

$$v^1 = \omega v^1 + d_1^1 + d_2^1 \tag{6}$$

Hence the eq. (1) is substituted by:

$$\mathbf{v} = \begin{cases} v^1, \text{ if } \mathbf{t} = 0 \\ v^0, \text{ if } \mathbf{t} = 1 \end{cases} \tag{7}$$

And the eq.(3) is replaced by:

$$\mathbf{t} = \begin{cases} \bar{\mathbf{t}}, \text{ if } r_3 \leq sig(\mathbf{v}) \\ \mathbf{t}, \text{ otherwise} \end{cases} \tag{8}$$

Where $\bar{\mathbf{t}}$ is the complement of $\mathbf{t}$. That is, if $\mathbf{t} = 0$ then $\bar{\mathbf{t}} = 1$ and if $\mathbf{t} = 1$ then $\bar{\mathbf{t}} = 0$.

C. Memory Binary PSO

Zhen *et al* (see [19]) propose a memory binary particle swarm optimization based on a new updating strategy where the position or status of a particle is updated based on its previous position or status.

In their method, the velocity $\mathbf{v}$ represents the probability that one bit stays still. When the current position of one particle is consistent with his best position and the best position within its neighborhood its position should remain the same, otherwise it should change with high probability.

The equations (1) and (3) are replaced by:

$$\mathbf{v} = \omega\mathbf{v} + c_1 r_1 Equal(\hat{\mathbf{t}}, \mathbf{t}) + c_2 r_2 Equal(\hat{\mathbf{g}}, \mathbf{t}) \tag{9}$$

$$\mathbf{t} = \begin{cases} \mathbf{t}, \text{ if } r_3 \leq sig(\mathbf{v}) \\ \bar{\mathbf{t}}, \text{ otherwise} \end{cases} \tag{10}$$

Where *Equal()* is a function defined as:

$$Equal(a, b) = \begin{cases} 1, \text{ if } a == b \\ -1, \text{ otherwise} \end{cases} \tag{11}$$

D. New Binary PSO with Velocity Control

Lanzarini *et al* (see [20]) propose two velocity vectors v_1 and v_2 in order to improve the discrete binary version of PSO. Each particle has two vectors (v_1 and v_2) where v_1 is updated as:

$$v_1 = \omega v_1 + c_1 r_1 (2\hat{\mathbf{t}} - 1) + c_2 r_2 (2\hat{\mathbf{g}} - 1) \tag{12}$$

Each element of v_1 is controlled by applying:

$$v_1 = \begin{cases} \delta_1, \text{ if } v_1 > \delta_1 \\ -\delta_1, \text{ if } v_1 \leq -\delta_1 \\ v_1, \text{ otherwise} \end{cases} \tag{13}$$

Where

$$\delta_1 = \frac{limit1_{upper} - limit1_{lower}}{2} \tag{14}$$

The velocity vector v_1 is calculated with eq. (12) and controlled with eq. (13) and its value is used to update the velocity vector v_2 as:

$$v_2 = v_2 + v_1 \tag{15}$$

Vector v_2 is also controlled as v_1 by changing $limit1_{upper}$ and $limit1_{lower}$ by $limit2_{upper}$ and $limit2_{lower}$, yielding δ_2 to control values of v_2 as in eq. (13). $Limit1$ and $limit2$ are predefined algorithm parameters.

Then, $\mathbf{t}$ is calculated as in eq. (3) by using v_2 instead of $\mathbf{v}$.

E. Hybrid Binary PSO

Menhas *et al* (see [16]) propose a hybrid approach that combines concepts of PSO and genetic algorithms (GA). The proposed hybrid strategy comprises a crossover and a mutation operator. The crossover operation is performed to wholly realize the cognitive and social concepts of the PSO and then a mutation operator is added to maintain population diversity.

Their work use a crossover operator after the position update (eq. 3). A random number ($r_4 \in \{0, 1\}$) is generated to determine the actual state of each bit of every candidate solution from the population of current solutions and the position $\mathbf{t}$ is updated again as:

$$\mathbf{t} = \begin{cases} \mathbf{t}, \text{ if } 0 \leq r_4 \leq \alpha \\ \hat{\mathbf{t}}, \text{ if } \alpha < r_4 \leq 2\alpha \\ \hat{\mathbf{g}}, \text{ if } 2\alpha < r_4 \leq 1 \end{cases} \tag{16}$$

The authors proposed to use $\alpha = 0.33$ meaning that a solution has a probability of 33.33% to remain in its current state (previously defined by the eq. 3) and 66.67% chances to be replaced by $\hat{\mathbf{t}}$ or $\hat{\mathbf{g}}$.

The mutation mechanism generates another random number ($r_5 \in \{0, 1\}$) and compares it with a predefined mutation probability parameter ($\mathbf{m}$) to perform mutation as:

$$\mathbf{t} = \begin{cases} \bar{\mathbf{t}}, \text{ if } r_5 \leq \mathbf{m} \\ \mathbf{t}, \text{ otherwise} \end{cases} \tag{17}$$

III. Binary Multi-Objective PSO to Test Case Selection

In this work, we propose some methods that adopt Particle Swarm Optimization (PSO) to solve multi-objective TC selection problems. In contrast to single-objective problems, Multi-Objective Optimization (MOO) aims to optimize more than one objective at the same time.

A MOO problem considers a set of k objective functions $f_1(x), f_2(x), ..., f_k(x)$ in which x is an individual solution for the problem being solved. The output of a MOO algorithm is usually a population of *non-dominated* solutions considering the objective functions. Formally, let x and x' be two different solutions. We say that x dominates x' (denoted by $x \preceq x'$) if x is better than x' in at least one objective function and x is not worse than x' in any objective function. x is said to be *not dominated* if there is no other solution x_i in the current population, such that $x_i \preceq x$. The set of non-dominated solutions in the objective space returned by a MOO algorithm is known as *Pareto front*.

We developed in our work five binary multiobjective PSO approaches by merging: (1) each binary version of PSO presented in Section II; and (2) the MOPSO-CDR algorithm proposed by [14]. We chose the MOPSO-CDR because of its good results in our previous work [9]. Furthermore, the proposed approaches were applied to select functional tests.

The objective functions to be optimized in our work are the functional requirements coverage and the execution effort of the selected TCs, in such a way that we maximize the first

function and minimize the second one. The proposed methods return a set of non-dominated solutions (a Pareto front) considering the aforementioned objectives. By receiving a set of diverse solutions, the user can choose the best one taking into account its current goals and available resources (e.g., the amount of time available at the moment to execute the test cases).

In our previous works (see [8] and [9]), we applied PSO to solve a constrained TC selection problem. However, to the best of our knowledge, besides our other work (see [11]), there is no previous work investigating PSO on a multi-objective functional TC selection problem. We also highlight that no previous work, besides ours, was found in the literature that performed test selection considering both requirements coverage and execution effort in a multi-objective way in the context of functional software testing. Hence, this is an promising study area and this work aim to explore a little further this problem by creating new multi-objective algorithms.

A. Problem Formulation

In this work, the particle's positions were defined as binary vectors representing candidate subsets of TCs to be applied in the software testing process. Let $T = \{T_1, \ldots, T_n\}$ be a test suite with n test cases. A particle's position is defined as $\mathbf{t} = (t_1, \ldots, t_n)$, in which $t_j \in \{0, 1\}$ indicates the presence (1) or absence (0) of the test case T_j within the subset of selected TCs.

As said, two objective functions were adopted. The requirements coverage objective (to be maximized) represents the amount (in percentage) of requirements covered by a solution $\mathbf{t}$ in comparison to the amount of requirements present in T. Formally, let $R = \{R_1, \ldots, R_k\}$ be a given set of k requirements of the original suite T. Let $F(T_j)$ be a function that returns the subset of requirements in R covered by the individual test case T_j. Then, the requirements coverage of a solution $\mathbf{t}$ is given as:

$$R\_Coverage(\mathbf{t}) = 100 * \frac{|\bigcup_{t_j=1}\{F(T_j)\}|}{k} \tag{18}$$

In eq. (18), $\bigcup_{t_j=1}\{F(T_j)\}$ is the union of requirements' subsets covered by the selected test cases (i.e., T_j for which $t_j = 1$).

The other objective function (to be minimized) is the execution effort (the amount of time required to manually execute the selected suite). Formally, each test case $T_j \in T$ has a *cost score* c_j. The total cost (effort) of a solution $\mathbf{t}$ is then defined as:

$$Cost(\mathbf{t}) = \sum_{t_j=1} c_j \tag{19}$$

In this approach, the cost c_j is computed for each test case by using the execution effort estimation model developed by Aranha and Borba [21]. The MOO algorithms will be used to find a good Pareto front regarding the objective functions $R\_Coverage$ and $Cost$.

B. The DBMOPSO-CDR algorithm

The Discrete Binary Multiobjective Particle Swarm Optimization with Crowding Distance and Roulette Wheel (DBMOPSO-CDR) was created by merging the MOPSO-CDR algorithm by [14] and the original discrete binary version of PSO by [12].

The MOPSO-CDR uses an External Archive (EA) to store the non-dominated solutions found by the particles during the search process. The EA can be seen as a secondary swarm, which interacts with the primary swarm in order to define the social guidance ($\hat{\mathbf{g}}$) and the particle memory ($\hat{\mathbf{t}}$) components of the particles at each search iteration.

The following algorithm summarizes the DBMOPSO-CDR:

1) Initialize the swarm by randomly initializing the position $\mathbf{t}$ and the velocity $\mathbf{v}$ of each particle;
2) Evaluate each particle according to the considered objective functions and store in the EA the particles' positions that are non-dominated solutions;
3) Calculate de Crowding Distance value (see [22]) for each non-dominated solution from EA and sort these solutions by this (CD) value;
4) Initialize the memory $\hat{\mathbf{t}}$ of each particle as:
$$\hat{\mathbf{t}} = \mathbf{t} \tag{20}$$
5) WHILE stop criterion is not verified DO
   a) Compute the velocity $\mathbf{v}$ of each particle[1] by using eq. 1.
   The social guide ($\hat{\mathbf{g}}$) in the above eq. 1 is defined as one of the non-dominated solutions stored in the current EA. The following procedure is adopted in order to select the ($\hat{\mathbf{g}}$) component for each particle:
      i) Select one EA's solution as the $\hat{\mathbf{g}}$ value by using Roulette Wheel, in which the probability of choosing a given solution is proportional to it CD value. Hence, solutions with higher CD values (corresponding to not well explored regions in the objectives' space) has a higher probability of being selected;
   b) Compute the new position $\mathbf{t}$ of each particle by using eq. 3;
   c) Use the mutation operator as proposed by [13];
   d) Evaluate each particle of the swarm and update the solutions stored in the EA by inserting all the new non-dominated solutions and by removing all previous EA's solutions that were dominated. Since the size of the EA is limited, whenever it gets full, the solutions in more crowded areas are removed from the EA (see [14]);

[1] As $\mathbf{v}$ is a vector with dimensionality dependent on the problem, it is necessary to compute each dimension.

e) Update the particle's memory $\hat{\mathbf{t}}$ using eq. (20) if the new position $\mathbf{t}$ dominates the previous memory (i.e., if $\mathbf{t} \preceq \hat{\mathbf{t}}$). If neither $\mathbf{t} \preceq \hat{\mathbf{t}}$ nor $\hat{\mathbf{t}} \preceq \mathbf{t}$ is verified (i.e., no solution dominates the other one), then the choice is made using the EA. The algorithm search in the EA for the solutions with minimum Euclidean distance for the $\hat{\mathbf{t}}$ and for $\mathbf{t}$. If the closer solution to $\mathbf{t}$ in the EA is in a less crowded region than the closer solution to the $\hat{\mathbf{t}}$, then the particle's memory is changed to $\mathbf{t}$ using eq. (20). Otherwise, it remains;

6) END WHILE and return the current EA as the Pareto front.

C. The NBMOPSO-CDR algorithm

The Novel Binary Multiobjective Particle Swarm Optimization with Crowding Distance and Roulette Wheel (NBMOPSO-CDR) was created by merging the MOPSO-CDR algorithm by [14] and the novel binary version of PSO by [18].

In the NBMOPSO-CDR the equations (1) and (3) (from DBMOPSO-CDR) are replaced by equations (7) and (8).

D. The MBMOPSO-CDR algorithm

The Memory Binary Multiobjective Particle Swarm Optimization with Crowding Distance and Roulette Wheel (MBMOPSO-CDR) was created by merging the MOPSO-CDR algorithm and the memory binary version of PSO by [19]. The equations (1) and (3) are replaced by equations (9) and (10).

E. The VCBMOPSO-CDR algorithm

The Velocity Control Binary Multiobjective Particle Swarm Optimization with Crowding Distance and Roulette Wheel (VCBMOPSO-CDR) was created by merging the MOPSO-CDR algorithm by [14] and the binary PSO with velocity control by [20].

In the VCBMOPSO-CDR then $\mathbf{t}$ is calculated as in eq. (3) by using the v_2 value (see section II-D) instead of $\mathbf{v}$.

F. The HBMOPSO-CDR algorithm

The Hybrid Binary Multiobjective Particle Swarm Optimization with Crowding Distance and Roulette Wheel (HBMOPSO-CDR) was created by merging the MOPSO-CDR algorithm by [14] and the hybrid binary PSO by [16].

The HBMOPSO-CDR uses a crossover operator (eq. 16) after the position update (eq. 3). The MOPSO-CDR has originally a mutation mechanism (see [14]) but the HBMOPSO-CDR replaces the mutation mechanism of the MOPSO-CDR by its own (eq. 17).

IV. Experiments and Results

This section presents the experiments performed in order to evaluate the search algorithms implemented in this work. The experiments were performed on a case study related to mobile devices. Other case studies in different domains will be performed as future work.

A. Experiments Preparation

Initially, we selected two test suites related to different features in the context of mobile devices[2]: an Integration Suite and a Regression Suite. Both suites have 80 TCs, each one representing an exhaustive test case scenario (see Table I). The Integration Suite (which covers 410 requirements) is focused on testing whether the various features of a mobile device can work together, i.e., whether the integration of the features behaves as expected. The Regression Suite (covering 248 requirements), in turn, is aimed at testing whether updates to a specific main feature (e.g., the message feature) have not introduced faults into the already developed (and previously tested) feature functionalities.

Table I summarizes the characteristics of each suite and a more detailed explanation about them can be seen on [9].

TABLE I
Characteristics of the Test Suites

	Integration Suite	**Regression Suite**
Total Effort to Execute all Test Cases	1053.91 min	699.31 min
# of Requirements	410	248
Redundancy	0.36%	14.09%
# of Test Cases	80	80

B. Metrics

In our experiments, we evaluated the results (i.e., the Pareto fronts) obtained by the algorithms DBMOPSO-CDR, NBMOPSO-CDR, MBMOPSO-CDR, VCBMOPSO-CDR and HBMOPSO-CDR for each test suite according to four different quality metrics usually adopted in the literature of multi-objective optimization. The following metrics were adopted in this paper: Hypervolume [22], Generational Distance [23], Inverted Generational Distance [23] and Coverage [22]. Each metric considers a different aspect of the Pareto front:

1) Hypervolume (HV) [22]: computes the size of the dominated space, which is also called the *area under the curve*. A high value of hypervolume is desired in MOO problems.
2) Generational Distance (GD) [23]: The Generational Distance (GD) reports how far, on average, one Pareto set (called PF_{known}) is from the true Pareto set (called as PF_{true}).
3) Inverted Generational Distance (IGD) [23]: is the inverse of GD by measuring the distance from the PF_{true} to the PF_{known}.
4) Coverage (C) [22]: The Coverage metric indicates the amount of the solutions within the non-dominated set of the first algorithm which dominates the solutions within the non-dominated set of the second algorithm.

Both GD and IGD metrics requires that the PF_{true} be known. Unfortunately, the PF_{true} (for some problems) is

[2]These suites were created by test engineers of the Motorola CIn-BTC (Brazil Test Center) research project.

impossible to know a priori. Instead, a reference Pareto frontier can be constructed and used when comparing different algorithms with respect to the Pareto frontiers they produce [6]. The reference frontier (called here $PF_{reference}$) represents the union of all found Pareto sets, resulting in a set of non-dominated solutions found.

C. Algorithms Settings

In this section, we present the main values of the parameters adopted for the algorithms. We adopted the same values suggested by the authors from whom we derived our algorithms.

- number of particles: 20
- mutation rate: 0.5
- inertia factor ω: linearly decreases from 0.9 to 0.4
- constants C_1 and C_2: 1.49
- Vmax: 4.0
- EA's size: 200 solutions
- maximum fitness evaluations: 200,000

D. Results

For statistical analysis purposes, we executed each search algorithm 60 times on each test suite. In each execution, a Pareto front was produced. The values of coverage and cost observed in the Pareto fronts were normalized since they are measured using different scales. After normalizing the Pareto fronts, all the evaluation metrics were computed.

We initially present the results for metrics Hypervolume, GD and IGD, which are computed for each algorithm individually. Tables II and III show the mean and standard deviation of these metrics for each algorithm respectively on Integration Suite and Regression Suite. For each metric, the superiority of an algorithm over the others is verified using pairwise t-test and when the means are statistically not different we use an asterisk (*) to indicate this.

From tables II, III, IV and V we observe that the NBMOPSO-CDR outperformed the other in terms of GD and Coverage metrics for both Integration and Regression suites. This means that its solution set has better convergence to the optimal Pareto set and its solutions has better dominance among the others. So, the user could choose better combinations between requirements' coverage and execution effort.

For the Hypervolume and IGD metrics we do not have a clearly dominance among the techniques. For Integration suite the HBMOPSO-CDR obtained the best results for this metric, but for Regression suite the NBMOPSO-CDR was the best for Hypervolume and for the IGD metric both NBMOPSO-CDR and MBMOPSO-CDR were obtained the best results. Because of that we cannot point out the best algorithm in terms of coverage of the search space and distribution to the optimal Pareto set.

One important thing to note is that the NBMOPSO-CDR for all metrics outperformed the VCBMOPSO-CDR. So, for our specific TC selection problem and by combination with the MOPSO-CDR algorithm, the binary work of [18] outperformed the work of [20], differently from the results obtained by [20]. And also for some metrics it also outperformed the work of [19] differing from the results obtained by [19]. Furthermore, despite of the works of [18] and [20] represent improvements to the [12] work, for some scenarios of our experiments, the work of [12] outperformed the other two. Nevertheless, further experiments should be performed in order to verify whether these results can be obtained in different TC selection contexts.

TABLE II
MEAN VALUE AND STANDARD DEVIATION - INTEGRATION SUITE

	Hypervolume	GD	IGD
DBMOPSO-CDR	0.6 (5.74E-3)	7.25E-3 (4.32E-4)	3.93E-3 (2.01E-4)
NBMOPSO-CDR	0.63 (1.32E-2)	6.24E-4 (1.62E-4)	4.26E-3 (1.20E-3)
MBMOPSO-CDR	0.65 (5.64E-3)	1.22E-3 (1.89E-4)	2.61E-3 (4.49E-4)
VCBMOPSO-CDR	0.59 (1.22E-2)	1.07E-3 (3.85E-4)	6.31E-3 (9.48E-4)
HBMOPSO-CDR	0.66 (4.55E-3)	2.79E-3 (2.16E-4)	1.80E-3 (1.50E-4)

TABLE III
MEAN VALUE AND STANDARD DEVIATION - REGRESSION SUITE

	Hypervolume	GD	IGD
DBMOPSO-CDR	0.80 (6.87E-3)	1.27E-2 (1.51E-3)	7.36E-3** (7.02E-4)
NBMOPSO-CDR	0.88 (7.85E-3)	6.00E-4 (1.98E-4)	3.00E-3* (1.15E-3)
MBMOPSO-CDR	0.87 (5.86E-3)	1.32E-3 (3.75E-4)	2.87E-3* (9.22E-4)
VCBMOPSO-CDR	0.84 (1.76E-2)	1.14E-3 (3.70E-4)	7.12E-3** (2.60E-3)
HBMOPSO-CDR	0.86 (4.92E-3)	5.51E-3 (8.18E-4)	3.57E-3 (5.56E-4)

V. CONCLUSION

In this work, we propose the use of binary multi-objective PSO for functional TC selection. The main contribution of the current work was to investigate some PSO variations in a multi-objective way for selecting functional test cases considering both requirements coverage and execution effort. The binary multi-objective PSO was barely investigated in the context of TC selection.

We highlight that the developed methods can be adapted to other test selection criteria and it is not limited to two objective functions. Furthermore, we expect that these good results can also be obtained on other application domains.

Knowing that the improvements were made to the original discrete binary version of PSO, we implemented five new binary multi-objective PSO, by combining the MOPSO-CDR approach (see Section III) with the binary PSO versions presented in Section II: the DBMOPSO-CDR, NBMOPSO-CDR, MBMOPSO-CDR, VCBMOPSO-CDR and HBMOPSO-CDR.

In the performed experiments, the NBMOPSO-CDR outperformed the others for the GD and Coverage metrics. For

TABLE IV

COVERAGE - MEAN VALUE AND STANDARD DEVIATION - INTEGRATION SUITE

Algorithm	DBMOPSO-CDR	NBMOPSO-CDR	MBMOPSO-CDR	VCBMOPSO-CDR	HBMOPSO-CDR
C(DBMOPSO-CDR, -)	-	4.07E-4 (1.80E-3)	1.70E-4 (1.32E-3)	5.11E-3 (8.94E-3)	7.25E-4 (3.96E-3)
C(NBMOPSO-CDR, -)	0.92 (6.71E-2)	-	0.72 (0.12)	0.69 (0.24)	0.74 (6.85E-2)
C(MBMOPSO-CDR, -)	0.99 (1.19E-2)	0.10 (7.07E-2)	-	0.22 (0.13)	0.77 (5.38E-2)
C(VCBMOPSO-CDR, -)	0.79 (6.88E-2)	0.19 (0.16)	0.50 (0.11)	-	0.54 (6.63E-2)
C(HBMOPSO-CDR, -)	0.99 (4.06E-3)	1.55E-2 (1.89E-2)	4.37E-2 (3.34E-2)	3.54E-2 (2.93E-2)	-

TABLE V

COVERAGE - MEAN VALUE AND STANDARD DEVIATION - REGRESSION SUITE

Algorithm	DBMOPSO-CDR	NBMOPSO-CDR	MBMOPSO-CDR	VCBMOPSO-CDR	HBMOPSO-CDR
C(DBMOPSO-CDR, -)	-	3.49E-4 (1.90E-3)	0 (0)	7.33E-4 (3.31E-3)	1.95E-3 (7.60E-3)
C(NBMOPSO-CDR, -)	0.96 (4.14E-2)	-	0.82 (0.12)	0.73 (0.21)	0.91 (5.54E-2)
C(MBMOPSO-CDR, -)	0.97 (3.57E-2)	9.92E-2 (9.30E-2)	-	0.36 (0.23)	0.90 (4.87E-2)
C(VCBMOPSO-CDR, -)	0.93 (5.37E-2)	0.15 (0.15)	0.43 (0.19)	-	0.75 (9.64E-2)
C(HBMOPSO-CDR, -)	0.99 (1.79E-2)	5.28E-3 (1.11E-2)	1.46E-2 (2.01E-2)	1.66E-2 (2.15E-2)	-

the others metrics we cannot point out which algorithm dominated the others. Hence, a deeper investigation and further experiments should be done as future work.

Another future work is to perform the same experiments on a higher number of test suites and to verify whether the obtained results are equivalent to the present work and whether these results can be extrapolated to other test's scenarios other than mobile devices. Furthermore, we plan to determine whether the adopted metrics are indeed suitable to measure the quality of test suites and the behavior of them.

Also, we will investigate the impact of changing the PSO's parameters in its performance on the TC selection problem. Besides, we intend to implement theses binary PSO versions with other multi-objective PSO approaches for a more complete comparison between techniques and to determine the distinct advantages of using MOPSO-CDR. Finally, we will investigate strategies to combine search techniques, in order to provide hybrid algorithms for multi-objective TC selection.

REFERENCES

[1] R. Ramler and K. Wolfmaier, "Economic perspectives in test automation - balancing automated and manual testing with opportunity cost," in *Workshop on Automation of Software Test, ICSE 2006*, 2006.

[2] P. Borba, D. Torres, R. Marques, and L. Wetzel, "Target - test and requirements generation tool," in *Motorola's 2007 Innovation Conference (IC'2007)*, 2007.

[3] M. J. Harold, R. Gupta, and M. L. Soffa, "A methodology for controlling the size of a test suite," *ACM Trans. Softw. Eng. Methodol.*, vol. 2, no. 3, pp. 270–285, 1993.

[4] P. Borba, A. Cavalcanti, A. Sampaio, and J. Woodcock, Eds., *Testing Techniques in Software Engineering, Second Pernambuco Summer School on Software Engineering, PSSE 2007, Recife, Brazil, December 3-7, 2007, Revised Lectures*, ser. Lecture Notes in Computer Science, vol. 6153. Springer, 2010.

[5] X.-Y. Ma, B.-K. Sheng, and C.-Q. Ye, "Test-suite reduction using genetic algorithm," *Lecture Notes in Computer Science*, vol. 3756, pp. 253–262, 2005.

[6] S. Yoo and M. Harman, "Pareto efficient multi-objective test case selection," in *Proc. of the Int. Symp. on Software Testing and Analysis*, 2007, pp. 140–150.

[7] K. Barltrop, B. Clement, G. Horvath, and C.-Y. Lee, "Automated test case selection for flight systems using genetic algorithms," in *Proceedings of the AIAA Infotech@Aerospace Conference (I@A 2010). Atlanta, GA*, 2010.

[8] L. S. Souza, R. B. C. Prudêncio, and F. d. A. Barros, "A constrained particle swarm optimization approach for test case selection," in *In Proc. of the 22nd Int. Conf. on Software Engineering and Knowledge Engineering (SEKE 2010)*, Redwood City, CA, USA, 2010.

[9] L. S. de Souza, R. B. Prudêncio, F. de A. Barros, and E. H. da S. Aranha, "Search based constrained test case selection using execution effort," *Expert Systems with Applications*, vol. 40, no. 12, pp. 4887 – 4896, 2013.

[10] S. Yoo and M. Harman, "Using hybrid algorithm for pareto efficient multi-objective test suite minimisation," *J. Syst. Softw.*, vol. 83, pp. 689–701, April 2010.

[11] L. S. Souza, P. B. C. Miranda, R. B. C. Prudêncio, and F. d. A. Barros, "A multi-objective particle swarm optimization for test case selection based on functional requirements coverage and execution effort," in *In Proc. of the 23rd Int. Conf. on Tools with Artificial Intelligence (ICTAI 2011)*, Boca Raton, FL, USA, 2011.

[12] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proc. of the World Multiconf. on Systemics, Cybernetics and Informatics*, 1997, pp. 4104–4109.

[13] C. Coello, G. Pulido, and M. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, 2004.

[14] R. A. Santana, M. R. Pontes, and C. J. A. Bastos-Filho, "A multiple objective particle swarm optimization approach using crowding distance and roulette wheel," in *Proc. of the Ninth Int. Conf. on Intelligent Systems Design and Applications*, Washington, DC, USA, 2009, pp. 237–242.

[15] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, 1995, pp. 1942–1948.

[16] M. I. Menhas, M. Fei, L. Wang, and X. Fu, "A novel hybrid binary pso algorithm," in *Proceedings of the Second international conference on Advances in swarm intelligence - Volume Part I*, ser. ICSI'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 93–100.

[17] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," *LNCS*, vol. 1447, pp. 611–616, 1998.

[18] M. Khanesar, M. Teshnehlab, and M. Shoorehdeli, "A novel binary particle swarm optimization," in *Control Automation, 2007. MED '07. Mediterranean Conference on*, 2007, pp. 1–6.

[19] Z. Ji, T. Tian, S. He, and Z. Zhu, "A memory binary particle swarm optimization," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*, 2012, pp. 1–5.

[20] L. Lanzarini, J. López, J. A. Maulini, and A. De Giusti, "A new binary pso with velocity control," in *Proceedings of the Second international conference on Advances in swarm intelligence - Volume Part I*, ser. ICSI'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 111–119.

[21] E. Aranha and P. Borba, "An estimation model for test execution effort," in *Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement*, 2007, pp. 107–116.

[22] K. Deb and D. Kalyanmoy, *Multi-Objective Optimization Using Evolutionary Algorithms*, 1st ed. Wiley, jun 2001.

[23] C. A. C. Coello, G. B. Lamont, and D. A. van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2007, vol. 5.