A multi-objective interactive dynamic particle swarm optimizer

Cristóbal Barba-González[1] · Antonio J. Nebro[1] · José García-Nieto[1] · José F. Aldana-Montes[1]

Abstract

Multi-objective optimization deals with problems having two or more conflicting objectives that have to be optimized simulta-neously. When the objectives change somehow with time, the problems become dynamic, and if the decision maker indicates preferences at runtime, then the algorithms to solve them become interactive. In this paper, we propose the integration of SMPSO/RP, an interactive multi-objective particle swarm optimizer based on SMPSO, with InDM2, an algorithmic template for dynamic interactive optimization with metaheuristics. The result is SMPSO/RPD, an algorithm that provides the search capabilities of SMPSO, incorporates an interactive preference articulation mechanism based on defining one or more reference points, and is able to deal with dynamic problems. We conduct a qualitative study showing the working of SMPSO/RPD on three benchmark problems, remaining a qualitative analysis as an open line of future research.

Keywords Multi-objective optimization · Particle swarm optimization · Interactive decision making · Dynamic optimization problem · Comparative study

1 Introduction

Multi-objective optimization is the discipline dealing with optimizing problems composed of two or more objectives or functions at the same time. These objectives are usually in conflict among them, so improving one implies worsening the others. As a consequence, the optimum on a multi-objective optimization problem is not usually a single solution, but a set of compromise solutions known as the Pareto optimal set; the correspondence of this set in the objective space is called the Pareto front.

Solving these kinds of problems is not a trivial task, particularly when they are also featured with characteristics, such as NP-hard complexity, nonlinearity, epistasis, deceptiveness and constraint handling [19]. For these reasons, finding for the Pareto front is not practical in general, so the goal becomes to get an approximation of Pareto front as accurate as possible by using non-exact algorithms. In this context, the most widely used techniques are metaheuristics [4], a broad family of solvers including evolutionary algorithms, swarm intelligence algorithms, and many others [2].

When dealing with real-world problems, finding an approximation to the Pareto front of a multi-objective problem is only the first step of the optimization process. The second step is MCDM (multi-criteria decision making), and it is related to choosing which solution of that front is most adequate according to the requirements imposed by the decision maker (i.e., the final user, which is an expert in the problem domain). As the optimization can take a significant amount of time, the decision maker may be interested in indicating one or more preference regions of interest of the Pareto front instead of the full front. These regions can be indicated a priori (before running the optimization algorithm) or interactively (during the execution of the algorithm) [9,13].

This work has been partially funded by Spanish Grants TIN2017-86049-R (Spanish Ministry of Education and Science). Cristóbal Barba-González is supported by Grant BES-2015-072209 (Spanish Ministry of Economy and Competitiveness). José García-Nieto is the recipient of a Post-Doctoral fellowship of "Captación de Talento para la Investigación" Plan Propio at Universidad de Málaga.

✉ José García-Nieto
jnieto@lcc.uma.es

Cristóbal Barba-González
cbarba@lcc.uma.es

Antonio J. Nebro
antonio@lcc.uma.es

José F. Aldana-Montes
jfam@lcc.uma.es

[1] Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, Bulevar Louis Pasteur 35, 29071 Málaga, Spain

Another feature of many real-world optimization problems is the fact that they are dynamic, i.e., they can change with time. In the case of multi-objective problems, these change can affect the Pareto set, the Pareto front, or both of them [8]. Solving dynamic problems implies that metaheuristics must be adapted to detect changes in the problems and to react consequently.

In this paper, we present an algorithm which is designed to take into account the preferences of the decision maker using both a priori and interactive schemes and also is able to tackle dynamic problems. This algorithm, called SMPSO/RPD, is the combination of SMPSO/RP [15] and InDM2 [17]. SMPSO/RP is an extension of SMPSO [16], a conventional multi-objective optimizer, empowering it to focus the search in several interests regions by using the concept of reference point [21]. InDM2 is an algorithmic template that facilitates the building of dynamic metaheuristics, enabling to incorporate restarting strategies to be applied when the problem or some of the reference points change.

SMPSO/RPD is implemented in jMetalSP [1], a framework providing support for dynamic problems, including the real-time visualization of the fronts produced when the problem has changed, as well as the integration with the Apache Spark [22] stream processing engine. This way, SMPSO/RPD can solve problems which change as a consequence of the processing of streaming data.

The main contribution of this paper can be summarized as follows:

- We propose a new algorithm, SMPSO/RPD, for solving dynamic problems allowing at the same time to define preference regions a priori and interactively.
- Our technique can adopt strategies for reacting when changes in the problem and/or reference points are detected.
- The fronts are visualized during the execution of the algorithm, showing only those produced when the problem has changed.
- We compare SMPSO/RPD with two other algorithms based on InDM2, showing the fronts obtained when solving benchmark problems.
- The source code of SMPSO/RPD is freely available.[1]

The rest of the paper is structured as follows. Section 2 explains the main background concepts and presents a review of the related work in the specialized literature. In Sect. 3, the SMPSO/RPD algorithm is detailed. Section 4 describes the experimental test carried out in terms of validating the proposal. In Sect. 5, the results obtained are depicted and commented. A further discussion of the algorithm is offered in Sect. 6. Finally, concluding remarks and future lines of research are presented in Sect. 7.

[1] jMetalSP: https://github.com/jMetal/jMetalSP.git.

2 Background

In this section, we provide basic background concepts about SMPSO, SMPSO/RP, InDM2, and jMetalSP, which are the basis of our proposal.

2.1 The speed-constrained multi-objective PSO (SMPSO) algorithm

SMPSO [16] is a multi-objective optimization particle swarm optimization algorithm featured by using a constriction mechanism to compute the particles' velocities and by adopting an external archive to choose the leaders and to store the non-dominated solutions found during the search.

As a conventional PSO [11], SMPSO works by manipulating a set of *particles* (vector solutions), known collectively as the *swarm*. The position of particle $\mathbf{x}_i$ at a given generation t is updated according to Eq. (1):

$$\mathbf{x}_i(t) = \mathbf{x}_i(t-1) + \mathbf{v}_i(t) \tag{1}$$

where the factor $\mathbf{v}_i(t)$ is known as velocity, and it is defined as:

$$\begin{aligned}\mathbf{v}_i(t) = w \cdot \mathbf{v}_i(t-1) + C_1 \cdot r_1 \cdot (\mathbf{x}_{p_i} - \mathbf{x}_i) \\ + C_2 \cdot r_2 \cdot (\mathbf{x}_{g_i} - \mathbf{x}_i)\end{aligned} \tag{2}$$

In Eq. (1), $\mathbf{x}_{p_i}$ is the *local best*, i.e., the best solution that $\mathbf{x}_i$ has viewed; $\mathbf{x}_{g_i}$ is the *global best* or *leader* $\mathbf{x}_{g_i}$, that is, the best particle that the entire swarm has viewed; w is the inertia weight of the particle and controls the trade-off between global and local influence; r_1 and r_2 are two uniformly distributed random numbers in the range [0, 1]; and C_1 and C_2 are specific parameters that control the effect of the personal and global best particles.

To control the velocity of the particles, SMPSO applies a *constriction coefficient* (Eq. 3) obtained from the constriction factor χ originally developed by Clerc and Kennedy (Eq. 2) in [3]

The constriction coefficient is defined as:

$$\chi = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}} \tag{3}$$

where

$$\varphi = \begin{cases} C_1 + C_2 & \text{if } C_1 + C_2 > 4 \\ 0 & \text{if } C_1 + C_2 \le 4 \end{cases} \tag{4}$$

Furthermore, SMPSO bounds the accumulated velocity of each variable j (in each particle) by means of the following *velocity constriction* equation:

$$v_{i,j}(t) = \begin{cases} \delta_j & \text{if } v_{i,j}(t) > \delta_j \\ -\delta_j & \text{if } v_{i,j}(t) \leq -\delta_j \\ v_{i,j}(t) & \text{otherwise} \end{cases} \tag{5}$$

where

$$\delta_j = \frac{(\text{upper\_limit}_j - \text{lower\_limit}_j)}{2} \tag{6}$$

So, in SMPSO the velocity of the particles is computed following Eq. 2, which is then multiplied by the constriction factor defined in Eq. 4 and the result is constrained by applying Eq. 6.

A key component in SMPSO is the external archive, which is updated whenever a new solution is found. It has a maximum capacity, so a density estimator is applied when it becomes full to choose the particle to be removed to foster diversity. The leader of the swarm is selected from the solutions of the archive by applying a binary tournament that considers as winner the particle having the best density estimator value. Finally, the resulting front of SMPSO is the leader archive (not the swarm).

2.2 SMPSO/RP: SMPSO with preference articulation

The SMPSO/RP algorithm (SMPSO with Reference Points) is an extension of SMPSO to empower it with a preference articulation mechanism [15]. Concretely, the adopted approach is to replace the conventional external archive by one or two more reference point archives. Each of these archives has an associated reference point, in such a way that only solutions that are non-dominated by the corresponding reference point are inserted into them, as illustrated in Fig. 1.

As more than one reference point archive can be used (one per preference region), the leader selection strategy of SMPSO is modified to take the leader from a randomly chosen external archive. Another interesting feature of SMPSO/RP is that the reference points of the external archives can be modified during the running of the algorithm. This way, the decision maker can guide interactively the search toward the most preferred regions.

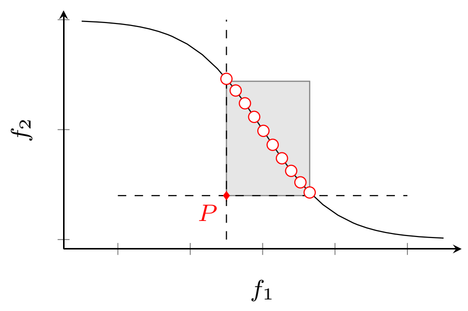

Fig. 1 Example illustrating the preference region demarcated by a reference point P

As SMPSO, SMPSO/RP is implemented in the jMetal framework for multi-objective optimization with metaheuristics [7,14], and its source code is freely available under MIT license in the GitHub repository hosting jMetal.

2.3 InDM2

The underlying idea of InDM2 (interactive dynamic multi-objective decision making) [17] is to provide a software template that, given an interactive multi-objective algorithm based on reference points, to extend it with support for dynamic problems solving in a simple and structured way. The resulting algorithm can define strategies for reacting to changes in the problem and in the reference points. These strategies have two components: one for removing solutions and another for filling the gaps of the removed ones. Examples of removing criteria are random selection, the lowest contribution according to a quality indicator or the solutions in the most crowded regions; examples of filling criteria include the randomly new solutions or solutions created from the remaining ones using some variation operator.

An key feature of InDM2 is the ability to visualize the approximations of the interest regions during the execution of the algorithm. This way, the decision maker can observe whether the optimization process is well focused and, if not, the reference points can be updated to modify the search. The operation of InDM2 was illustrated in [17] by embedding into it two reference point-based solvers, WASF-GA [18] and R-NSGA-II [6]. InDM2 is implemented on top of jMetalSP, which is described next.

2.4 The jMetalSP framework

jMetalSP is a framework written in Java aimed at solving dynamic multi-objective optimization problems with Big Data technologies [1]. It is based on jMetal, which provides most of its core features (including the implementation of SMPSO/RP, as commented previously).

The most salient characteristic of jMetalSP is that it can incorporate one or more components that can receive and analyze data coming in streaming by using the Apache Spark general-purpose cluster computing system [22]. The generated results can lead to the modification of a dynamic problem, which is being solved by a dynamic metaheuristic. The object-oriented architecture of jMetalSP enables the easy incorporation of streaming components. As jMetalSP uses Spark, it can be deployed in Hadoop/Spark clusters [20] and access the HDFS file system.

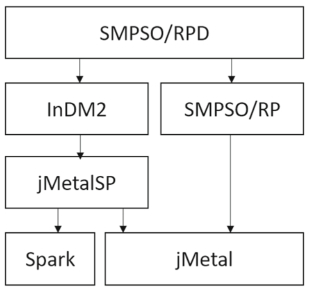

Fig. 2 Layer scheme of the implementation of SMPSO/RPD

2.5 SMPSO/RPD implementation scheme

The implementation of SMPSO/RPD is based on the previously detailed algorithms and software framework, following the structure depicted in Fig. 2. Our proposal combines InDM2 and SMPSO/RP, being the former based on jMetalSP and the latter on jMetal. Finally, jMetalSP relies on jMetal and Spark.

3 Algorithm description

In this section, we provide a high-level description of SMPSO/RPD. A pseudocode is included in Algorithm 1, which is the result of embedding SMPSO/RP into InDM2.

Algorithm 1 Pseudocode of SMPSO/RPD

```
1: $N$ ; // Swarm size
2: $G_{max}$ ; // Maximum number of generations
3: $m$ ; // Mutation (turbulence operator)
4: $S$ ; // Size of the external archives(s)
5: $\varphi_q$ ; // Restart strategy when the reference changes
6: $\varphi_p$ ; // Restart strategy when the problem changes
7: $M \leftarrow \{\text{SMPSO/RP}\}$ ; // Base optimization algorithm
8: $t \leftarrow 0$ ; // Generation counter
9: $PS_t \leftarrow Unchanged$ ; // Problem status
10: $\mathbf{q}_t$ ; // Initial reference point(s)
11: $P_t \leftarrow initializeSwarm(N)$ ;
12: $evaluate(P_t)$ ;
13: $A_t \leftarrow initializeExternalArchives(S, P_t, \mathbf{q}_t)$ ;
14: while true do
15:     while $t < G_{max}$ do
16:         $(P_{t+1}, A_{t+1}) \leftarrow M.step(M, \mathbf{q}_t, m, P_t, A_t)$ ;
17:         if $\mathbf{q}_{t+1} \neq \mathbf{q}_t$ then
18:             $(P'_{t+1}, A_{t+1}) \leftarrow restart(P_{t+1}, \varphi_q, \mathbf{q}_{t+1})$ ;
19:         else if $PS_{t+1} \neq PS_t$ then
20:             $P'_{t+1} \leftarrow restart(P_{t+1}, \varphi_p, A_{t+1})$ ;
21:         end if
22:         $t \leftarrow t + 1$ ;
23:     end while
24:     $R : (A_t)$ ; // Return external archive(s)
25:     $t \leftarrow 0$ ;
26: end while
```

The parameter settings of the algorithm are defined in lines 1–7. They include the swarm size, the maximum number of generations, a mutation operator (turbulence), the size of the external reference-point-based archives (for the sake of simplicity, we assume that all of them have the same size), the restarting strategies when changes in the reference points and/or problem state, and the base algorithm (SMPSO/RP).

After the generation counter is started to 0 (line 8), the following objects are initialized (lines 8–13): the problem status (which can take two values: *changed* or *unchanged*), the initial reference point(s), the swarm and the external archives. Note that the swarm is evaluated after being initialized (line 12).

The main loop of SMPSO/RPD is included in the code between lines 15 and 23, where the following steps take place: A step of SMPSO/RP is executed, resulting in modifications in the swarm and the external archives (line 16); then, if a change in the reference point(s) and/or in the problem status is detected, the corresponding restart strategy is executed (lines 17–21); finally, the generation counter is increased.

By default, it is assumed that the algorithm runs forever (loop between lines 14 and 26) and that every maximum number of generations a Pareto front approximation is returned [(i.e., the external archive(s)] and the algorithm starts again (lines 24 and 25). As SMPSO/RPD runs on jMetalSP, the resulting front can be stored then into files or displayed in a chart.

4 Experimentation

In order to verify the working of SMPSO/RPD, we compare it in this study against two other interactive and dynamic algorithms developed with InDM2 when solving three dynamic benchmark problems. A detailed description of these problems and the experimental methodology are provided next.

4.1 Description of the compared algorithms

In the paper describing InDM2 [17], two-reference-point-based algorithms were adapted to be integrated with it, namely WASF-GA [18] and R-NSGA-II [6].

The *Weighting Achievement Scalarizing Function Genetic Algorithm* (WASF-GA) considers a reference point to define the user preferences. It is based on a scalarizing approach, which takes into account the reference point [21] and a sample of weight vectors.

The *Reference-Point-Based NSGA-II algorithm* or R-NSGA-II gives the chance to the DM of defining one or more reference points. The main change compared to the original NSGA-II [5] algorithm is the replacement of the crowding distance density estimator by a preference distance that equally emphasizes solutions whose objective vectors are close to any of the given reference points with respect to the Euclidean distance. An additional niching operator is needed to control the distribution of the emphasized solutions.

4.2 Benchmarking problems

As benchmark problems, have dealt with those proposed for the CEC 2018 competition on dynamic multi-objective optimization [10]. This test suite, called DF, comprises nine bi-objective and five tri-objective problems; all of them are included in jMetalSP. For simplicity and with the aim of properly illustrating the performance of the proposal, we have selected two bi-objective problems, DF1 and DF6, as well as the tri-objective problem DF10. The number of objectives is represented in the equations as M, and n is the number of decision variables.

The time is defined according to Eq. 7

$$t = \frac{1}{n_t}\left\lfloor \frac{\sigma}{\sigma_T} \right\rfloor \tag{7}$$

where σ is the generator counter, σ_T is the number of generations for which t remains fixed, and n_t is the number of distinct steps in t. In the benchmark specification [10], the authors recommend to set $n = 10$, $\sigma_t = 10$ (fast-changing environment) or $\sigma_t = 30$ (slow-changing environment), and $n_t = 10$.

The first problem is DF1, which is a dynamic bi-objective problem with Pareto fronts changing from convex geometry to concave geometry and Pareto sets also changing over time. The problem is defined in Eq. 8:

$$\min \begin{cases} f_1(\mathbf{x}) = x_1 \\ f_2(\mathbf{x}) = g(\mathbf{x})\left(1 - \left(\frac{x_i}{g(\mathbf{x})}\right)^{H(\mathbf{t})}\right) \\ \text{where:} \\ g(\mathbf{x}) = 1 + \sum_{i=2}^{n}(x_i - G(t))^2 \\ H(t) = 0.75\sin(0.5\pi t) + 1.25 \\ G(t) = |\sin(0.5\pi t)| \\ t = \frac{1}{n_t}\left\lfloor \frac{\sigma}{\sigma_T} \right\rfloor \\ \mathbf{PS}(t) : 0 \le x_1 \le 1, x_i = G(t), i = 2, \dots, n \\ \mathbf{PF}(t) : f_2 = 1 - f_1^{H(t)}, 0 \le f_1 \le 1 \\ \text{and the search space is } [0, 1]^n \end{cases} \tag{8}$$

The Pareto front geometry of problem DF6 is time-changing, and the Pareto front is featured by having knee regions/points and long tails. The problem is defined in Eq. 9.

$$\min \begin{cases} f_1(\mathbf{x}) = g(\mathbf{x})(x_1 + 0.1\sin(3\pi x_1))^{\alpha t} \\ f_2(\mathbf{x}) = g(\mathbf{x})(x_1 + 0.1\sin(3\pi x_1))^{\alpha t} \\ \text{where:} \\ g(\mathbf{x}) = 1 + \sum_{i=2}^{n}\left(|G(t)|\, y_i^2 - 10\cos(2\pi y_i + 10)\right) \\ y_i = x_i - G(t) \\ G(t) = |\sin(0.5\pi t)| \\ \alpha_t = 0.2 + 2.8\,|G(t)| \\ \mathbf{PS}(t) : 0 \le x_1 \le 1, x_i = G(t), i = 2, \dots, n \\ t = \frac{1}{n_t}\left\lfloor \frac{\sigma}{\sigma_T} \right\rfloor \\ \mathbf{PF}(t) : f_1^{\frac{1}{\alpha 1}} + f_2^{\frac{1}{\alpha 1}} = \\ 1 + 0.2\sin\left(3\pi \frac{f_1^{\frac{1}{\alpha 1}} - f_2^{\frac{1}{\alpha 1}} + 1}{2}\right), 0 \le f_1 \le 1 \\ \text{and the search space is } [0, 1]x[-1, 1]^{n-1} \end{cases} \tag{9}$$

The third problem considered, DF10, is a dynamic three-objective optimization problem whose Pareto fronts change over the time from convexity to concavity, and vice versa. Its formulation is included in Eq. 10.

$$\min \begin{cases} f_1(\mathbf{x}) = g(\mathbf{x})\,[\sin(0.5\pi x_1)]^{H(t)} \\ f_2(\mathbf{x}) = g(\mathbf{x})\,[\sin(0.5\pi x_2)\cos(0.5\pi x_1)]^{H(t)} \\ f_3(\mathbf{x}) = g(\mathbf{x})\,[\cos(0.5\pi x_2)\cos(0.5\pi x_1)]^{H(t)} \\ \text{where:} \\ g(\mathbf{x}) = 1 + \sum_{i=3}^{n}\left(x_i - \frac{\sin(2\pi(x_1+x_2))}{1+|G(t)|}\right)^2 \\ t = \frac{1}{n_t}\left\lfloor \frac{\sigma}{\sigma_T} \right\rfloor \\ H(t) = 2.25 + 2\cos(0.5\pi t) \\ G(t) = |\sin(0.5\pi t)| \\ \mathbf{PS}(t) : 0 \le x_{i=1,2} \le 1, x_i = \frac{\sin(2\pi(x_1+x_2))}{1+|G(t)|}, \\ \quad i = 3, \dots, n \\ \mathbf{PF}(t) : \sum_{i=1}^{M} f_i^{\frac{2}{H(t)}} = 1, 0 \le f_{i:1:M} \le 1 \\ \text{and the search space is } [0, 1]^2x[-1, 1]^{n-2} \end{cases} \tag{10}$$

4.3 Experimental methodology and parameter settings

To verify the working of SMPSO/RPD and to compare the results it provides against the other InDMS2-based WASF-GA and R-NSGA-II algorithms, we have conducted the following experiments:

- *Experiment 1* Solving the DF1 and DF6 problems with SMPSO/RPD without (a) indicating any reference point and (b) defining two regions of interest.
- *Experiment 2* Solving the same two problems starting with SMPSO/RPD, WASF-GA, and R-NSGA-II with

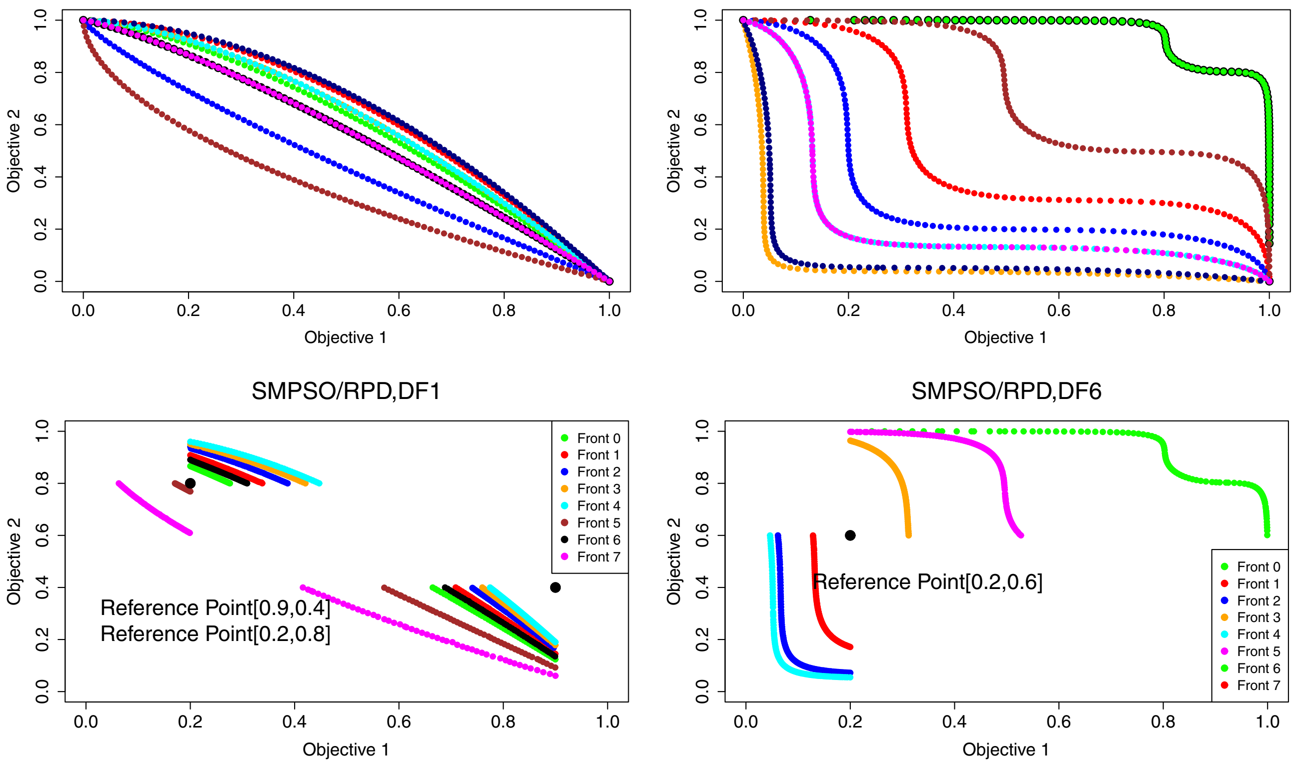

Fig. 3 Pareto fronts approximation obtained by SMPSO/RPD when solving problems DF1 and DF6 without reference points (top) and with reference points (0.2, 0.8) and (0.9, 0.4) for DF1 and reference point (0.2, 0.6) for DF6 (bottom)

reference point (0.0, 0.0) and changing it after a number of fronts have been generated.

- *Experiment 3* Solving problem DF10 with the three algorithms with reference point (0.0, 0.0, 0.0) and changing it after a number of fronts have been generated.

In this study, SMPSO/RPD and InDM2 have been run for solving the aforementioned problems and the preferences (i.e., the reference point) have been manually changed during the execution time. The goal is to assess how SMPSO/RPD is able to adapt the optimization process to the changes (both in the problem configuration and in the preferences) and to show the effect of these changes in the Pareto front approximations obtained, which must approximate only the regions of interest associated with the reference points given.

The configuration of all the algorithms includes common values, when possible, to ensure that the comparative is fair. Thus, all of them use a population/swarm size of 100 solutions, the maximum number of generations is 250, and they use the polynomial mutation operator with probability $p = 1/L$ (L is the number of decision variables of the problem) and the value of the distribution index is 20.0. The restarting strategies when changes in the problem and the reference points are detected consist in removing M solutions selected randomly (where M is the half of the population/swarm size) and generating M new random solutions in both cases. WASF-GA and R-NSGA-II use a SBX crossover operator with probability equal to 0.9 and a distribution index value of 20.0.

5 Results

In this section, we show the fronts obtained by the algorithms compared in accordance with the three experiments defined in the previous section.

5.1 Experiment 1

In Fig. 3, the approximations of the regions of interest found by SMPSO/RPD for the DF1 and DF6 problems are depicted. At the top are shown the Pareto fronts when no reference points are taken into account, whereas we can see at the bottom of this figure how SMPSP/RPD found the Pareto fronts that belong to the regions of interest delimited by the reference points. In case of DF1, the reference points are (0.2, 0.8) and (0.9, 0.4), while for DF6 the new reference point is (0.2, 0.6). For these two problems, it can be observed how the regions of interest are properly delimited by the reference points, so the front approximations generated cover

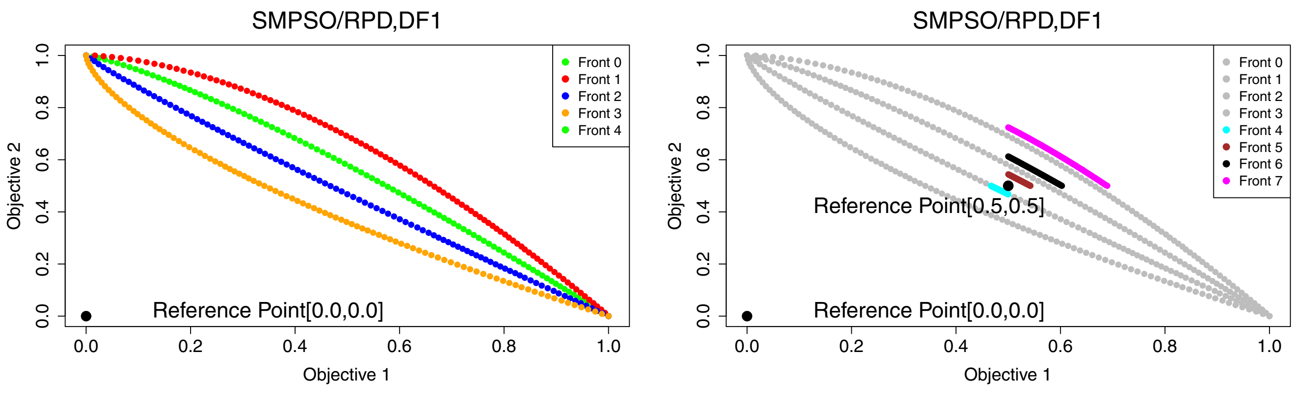

Fig. 4 Approximations of the region of interest found by SMPSO/RPD for the DF1 problem, using first the reference point (0.0, 0.0) (left) and, after producing fix fronts, it is changed to (0.5, 0.5) (right)

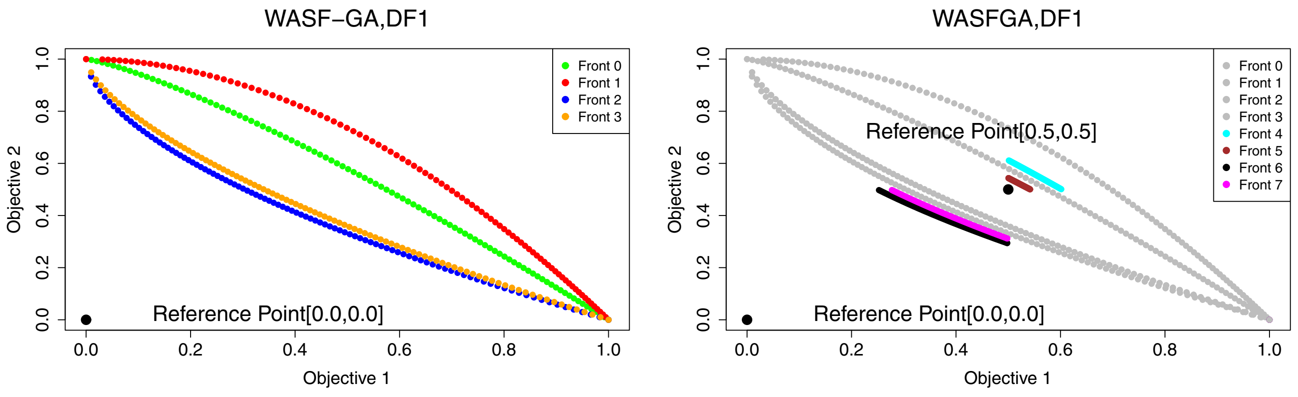

Fig. 5 Approximations of the region of interest found by WASF-GA for the DF1 problem, using reference points (0.0, 0.0) (left) and reference point (0.5, 0.5) (right)

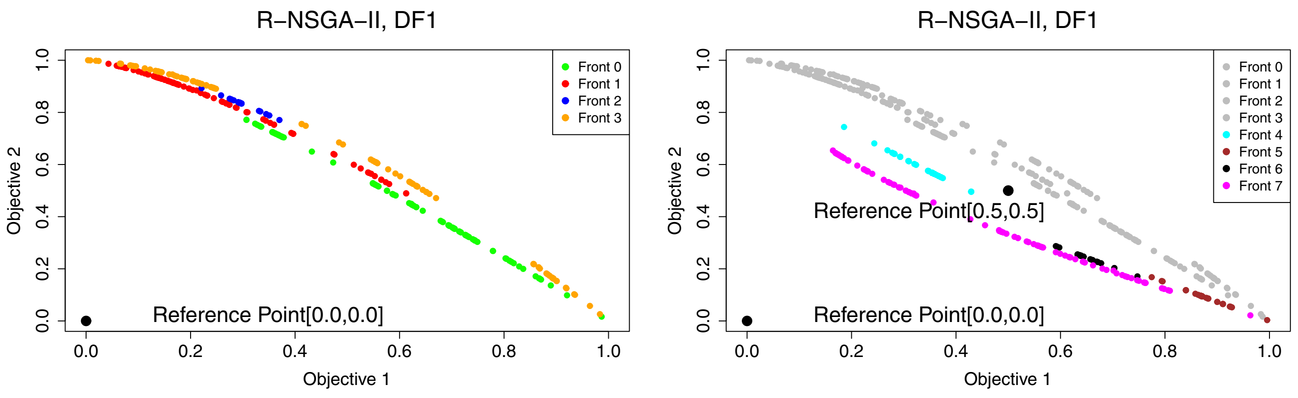

Fig. 6 Approximations of the region of interest found by R-NSGA-II for the DF1 problem, using reference points (0.0, 0.0) (left) and reference point (0.5, 0.5) (right)

the preferred area, independently of the shape in the objective space.

5.2 Experiment 2

This experiment is conducted to highlight the capacity of SMPSO/RPD to dynamically obtain regions of interest delimited by reference points. In order to assess its performance, in visual way, we compare SMPSO/RPD with WASF-GA and R-NSGA-II when solving the bi-objective problems DF1 and DF6. In Figs. 4, 5, and 6, we have simulated a real scenario where the reference points are modified during the optimization process. In the three figures, a chart with the Pareto front approximations obtained when the reference point is (0.0, 0.0) is shown in the left part, whereas the chart in the right part depicts the fronts obtained with the

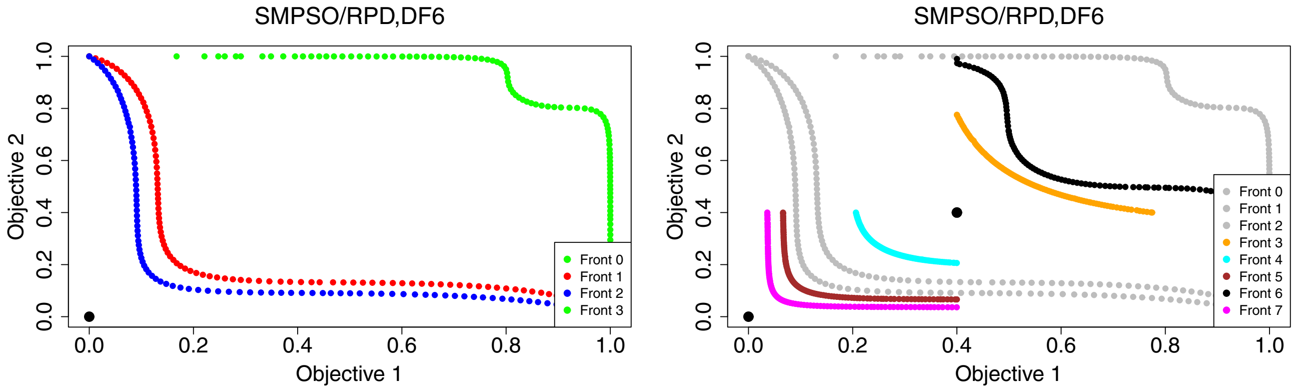

Fig. 7 Approximations of the region of interest found by SMPSO/RPD for the DF6 problem, using no reference points (top left), reference points (0.0, 0.0) (top right) and reference point (0.4, 0.4) (bottom)

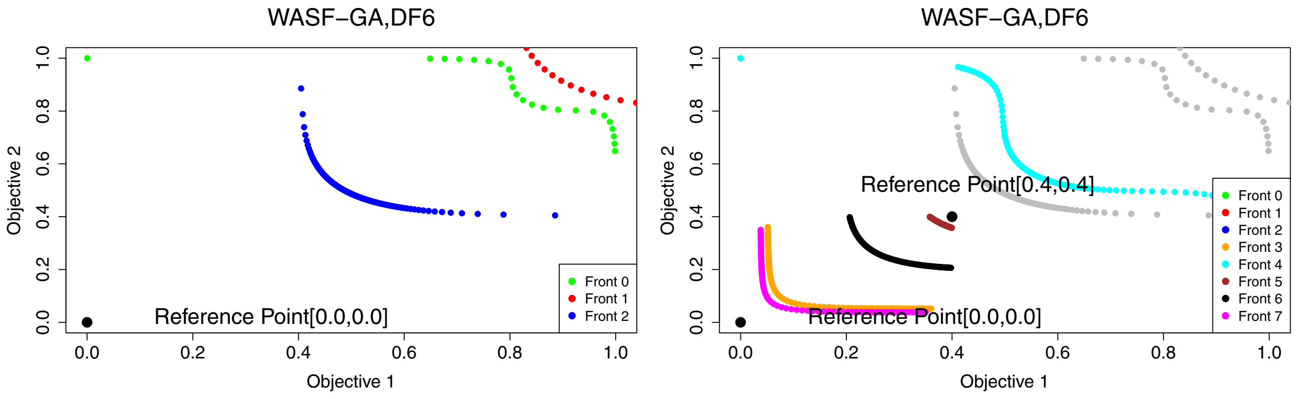

Fig. 8 Approximations of the region of interest found by WASF-GA for the DF6 problem, using reference points (0.0, 0.0) (left) and reference point (0.4, 0.4) (right)

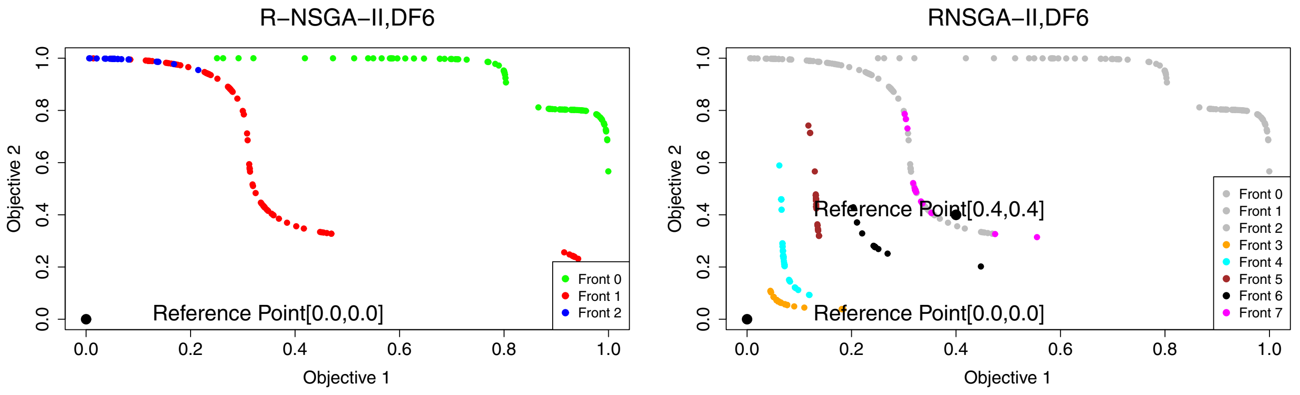

Fig. 9 Approximations of the region of interest found by R-NSGA-II for the DF1 problem, using reference points (0.0, 0.0) (left) and reference point (0.4, 0.4) (right)

reference point having been changed to (0.5, 0.5). (The color of the previous fronts has been changed to gray for the sake of clarity.)

At a glance, we can observe that the approximation of the regions of interest produced by SMPSO/RPD (Fig. 4) and WASF-GA (Fig. 5) is more evenly spread than that generated by R-NSGA-II (Fig. 6). Furthermore, R-NSGA-II approximates areas outside of the preference regions.

Similar behaviors can be inspected in Figs. 7, 8, and 9 but with the DF6 problem. In this case, SMPSO/RPD (Fig. 7) achieves wider fronts than WASF-GA with reference point (0.0, 0.0).

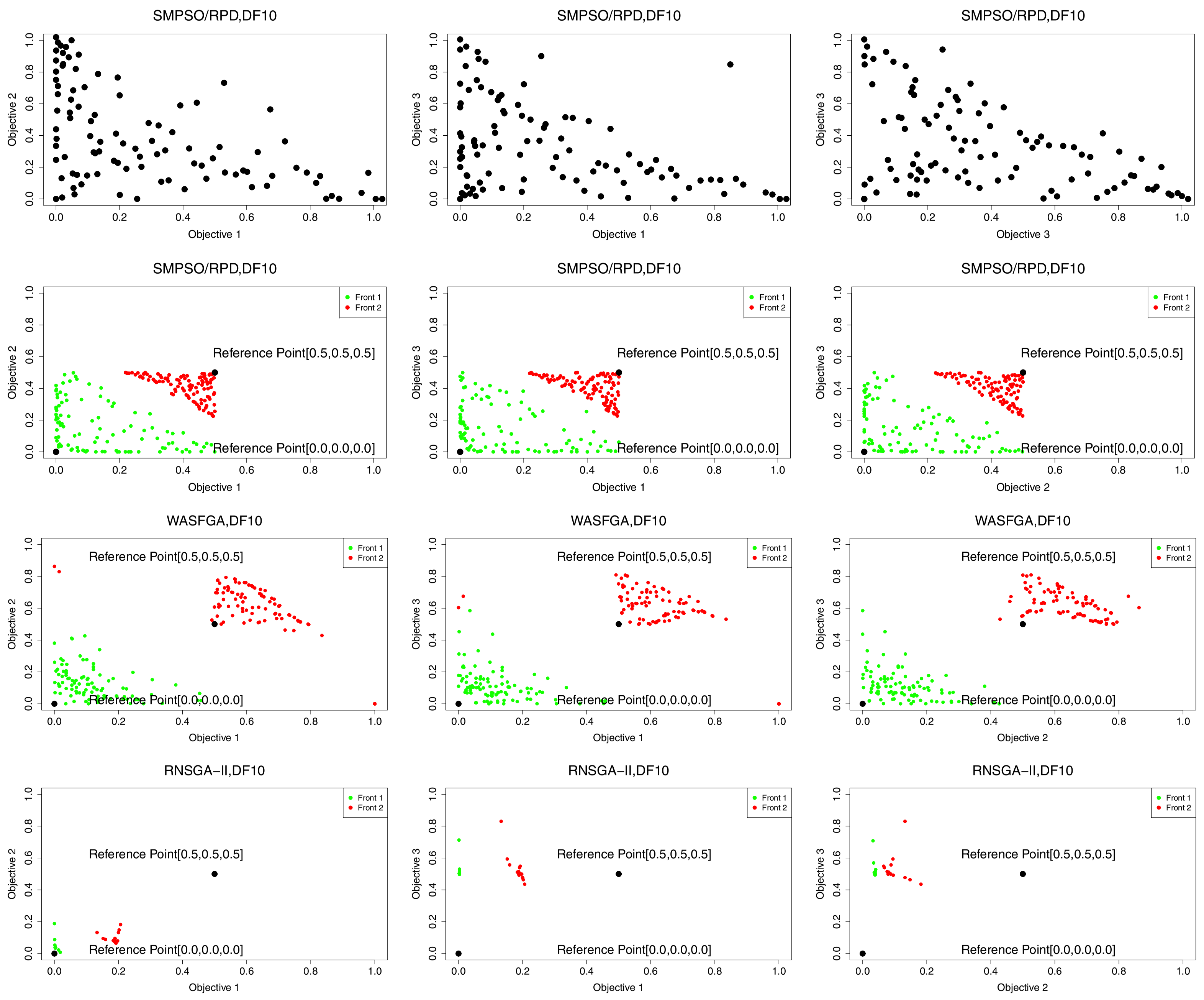

Fig. 10 Approximations of the region of interest found by SMPSO-RPD (first and second rows), WASF-GA (third row) and R-NSGA-II (fourth row) for the DF10 problem, using different reference points. First row without reference point (color figure online)

5.3 Experiment 3

Let us continue with the dynamic three-objective optimization problem DF10. Its Pareto fronts change over time from convexity to concavity, and vice versa, so one of the challenges when facing this problem is how to maintain uniformity of solutions on the badly shaped Pareto front at some time step. To easily visualize the approximations found by the algorithms for DF10, which belong to the three-dimensional objective space, we have used bi-dimensional images, which show the values of each pair of objectives for all the solutions in the approximations (i.e. 2D projections of each pair of objectives). Figure 10 plots the Pareto front approximations generated by SMPSO/RPD without any reference point (first row) and with the reference points (0.0, 0.0, 0.0) (green) or (0.5, 0.5, 0.5) (red). The third row of plots in Fig. 10 illustrates the performance of WASF-GA, and the fourth one corresponds to R-NSGA-II. At the glance, we can observe that the approximation of the regions of interest produced by SMPSO/RPD has a higher degree of convergence and diversity than that generated by R-NSGA-II and WASF-GA.

6 Discussion

After studying the experiments from the previous section, we can observe that SMPSO/RPD has performed as expected, although it is worth discussing a number of issues that have emerged after analyzing the behavior of the algorithm proposed.

The first issue has to do with the way of assessing the performance of the search capabilities of SMPSO/RPD. We have conducted a qualitative experimentation that has shown the behavior of our algorithm when solving a number of benchmark problems. However, the standard methodology in the literature is to perform a quantitative analysis by comparing the proposed technique with other algorithms of the state of the art based on applying some quality indicators about the convergence and diversity degrees of the obtained Pareto front approximations. We can find proposals of quality indicators for dynamic problems, such as the mean IGD and mean hypervolume defined in [10], and there are also metrics, e.g., the R-Metric [12], for reference-point-based algorithms, but to the best of our knowledge, there are not indicators for multi-objective algorithms which are both dynamic and interactive. We can think of an approach to tackle this issue by combining these kinds of quality indicators, taking as the basis a benchmark of dynamic problems, and we would need do define exactly (probably, in terms of number of algorithm iterations) when the problem and the reference points change. We should need besides to decide when the indicators should be applied and the portion (or portions if there is more than one reference point) of the Pareto front that it is expected in that moment in order to apply the indicators. This is an open research line that is left for future work.

From an overall perspective, and in accordance with our study, we can conclude that SMPSO/RPD and InDM2 with WASF-GA have led to approximations with more evenly spread solutions in comparison with those generated when using InDM2 with R-NSGA-II. In many cases, the solutions obtained by SMPSO/RPD and WASF-GA dominated those ones of R-NSGA-II.

Although the performance of SMPSO/RPD and WASF-GA in the selected problems seems similar in the considered bi-objective problems, it must be noted that WASF-GA is limited to one reference point, while the proposed algorithm SMPSO/RPD allows to define any number of them. In the case of the three-objective problem, SMPSO/RPD is clearly the most salient algorithm.

7 Conclusions and future work

We have presented SMPSO/RPD, an interactive multi-objective optimization metaheuristic for solving dynamic multi-objective optimization problems. SMPSO/RPD is an extension of the SMPSO/RP combined with InDM2, which incorporates a preference articulation mechanism based on indicating reference points and interactivity during the optimization process. Hence, it enables the decision maker to interactively change the regions of interest by giving and updating one or more reference points defining them. SMPSO/RPD is implemented within the jMetalSP framework, and its source code is freely available.

SMPSO/RPD has been described in detail, and its working procedure has been analyzed by solving three representative continuous dynamic multi-objective problems. The reported figures have shown how our algorithm behaves when the problem and the reference points change. The practical goal is to give an illustrative idea of the full potential of SMPSO/RPD, as a dynamic algorithm handling preferences interactively, which has been able to generate approximation adjusting to the given preferences (i.e., the region of interest), in real time, while the problem also changes at the same time. We have also discussed a number of open issues related to our proposal.

As a main line of future work, we plan to investigate the mechanisms to compute performance indicators in dynamic optimization scenarios where interactive processes of reference-point selection for preference articulation are involved. To this end, the use of artificial decision makers acting with certain synchronization steps seems to be a promising way to allow fair comparisons among similar approaches. In addition, we also aim to study the performance of SMPSO/RPD when solving complex real-world dynamic problems, for which domain expert knowledge can be included on the fly to influence algorithm behavior and final results. Another open line is to improve the visualization capabilities when dealing with problems having more than two objectives.

References

1. Barba-González, C., García-Nieto, J., Nebro, A.J., Cordero, J.A., Durillo, J.J., Navas-Delgado, I., Aldana-Montes, J.F.: jMetalSP: a framework for dynamic multi-objective big data optimization. Appl. Soft Comput. **69**, 737–748 (2017)
2. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput. Surv. **35**(3), 268–308 (2003)
3. Clerc, M., Kennedy, J.: The particle swarm—explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. Evol. Comput. **6**(1), 58–73 (2002)
4. Coello Coello, C., Lamont, G., van Veldhuizen, D.: Multi-objective Optimization Using Evolutionary Algorithms, 2nd edn. Wiley, New York (2007)
5. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
6. Deb, K., Sundar, J., Ubay, B., Chaudhuri, S.: Reference point based multi-objective optimization using evolutionary algorithm. Int. J. Comput. Intell. Res. **2**(6), 273–286 (2006)
7. Durillo, J.J., Nebro, A.J.: jMetal: a java framework for multi-objective optimization. Adv. Eng. Softw. **42**(10), 760–771 (2011)
8. Farina, M., Deb, K., Amato, P.: Dynamic multiobjective optimization problems: test cases, approximations, and applications. IEEE Trans. Evol. Comput. **8**(5), 425–442 (2004)
9. Jaszkiewicz, A., Branke, J.: Interactive multiobjective evolutionary algorithms. In: Branke, J., Deb, K., Miettinen, K., Słowiński, R. (eds.) Multiobjective Optimization, pp. 179–193. Springer, Berlin (2008)
10. Jiang, S., Yang, S., Yao, X., Tan, K.C., Kaiser, M., Krasnogor, N.: Benchmark problems for CEC2018 competition on dynamic multiobjective optimisation (2018)
11. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
12. Li, K., Deb, K., Yao, X.: R-metric: evaluating the performance of preference-based evolutionary multiobjective optimization using reference points. IEEE Trans. Evol. Comput. **22**(6), 821–835 (2018)
13. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston (1999)
14. Nebro, A., Durillo, J.J., Vergne, M.: Redesigning the jMetal multi-objective optimization framework. In: Proceedings of the Conference on Genetic and Evolutionary Computation, GECCO Companion'15, pp. 1093–1100. ACM (2015)
15. Nebro, A.J., Durillo, J.J., García-Nieto, J., Barba-González, C., Del Ser, J., Coello Coello, C.A., Benítez-Hidalgo, A., Aldana-Montes, J.F.: Extending the speed-constrained multi-objective PSO (SMPSO) with reference point based preference articulation. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) Parallel Problem Solving from Nature—PPSN XV, pp. 298–310. Springer, Cham (2018)
16. Nebro, A.J., Durillo, J.J., García-Nieto, J., Coello Coello, C.A., Luna, F., Alba, E.: SMPSO: a new PSO-based metaheuristic for multi-objective optimization. In: IEEE Symposium on Computational Intelligence in Multi-criteria Decision-Making, pp. 66–73 (2009)
17. Nebro, A.J., Ruiz, A.B., Barba-González, C., García-Nieto, J., Luque, M., Aldana-Montes, J.F.: InDM2: interactive dynamic multi-objective decision making using evolutionary algorithms. Swarm Evol. Comput. **40**, 184–195 (2018)
18. Ruiz, A., Saborido, R., Luque, M.: A preference-based evolutionary algorithm for multiobjective optimization: the weighting achievement scalarizing function genetic algorithm. J. Glob. Optim. **62**(1), 101–129 (2015)
19. Weise, T., Zapf, M., Chiong, R., Nebro, A.J.: Why is Optimization Difficult?, pp. 1–50. Springer, Berlin (2009)
20. White, T.: Hadoop: The Definitive Guide, 1st edn. O'Reilly Media, Inc., Newton (2009)
21. Wierzbicki, A.P.: The use of reference objectives in multiobjective optimization. In: Fandel, G., Gal, T. (eds.) Multiple Criteria Decision Making, Theory and Applications, pp. 468–486. Springer, Berlin (1980)
22. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. In: Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud'10, pp. 10–10. USENIX Association (2010)