J Heuristics (2018) 24:581–616
https://doi.org/10.1007/s10732-018-9369-x

Evaluating selection methods on hyper-heuristic multi-objective particle swarm optimization

Olacir R. Castro Jr.[1] · Gian Mauricio Fritsche[1] · Aurora Pozo[1]

Received: 25 November 2016 / Revised: 15 December 2017 / Accepted: 15 March 2018 / Published online: 22 March 2018

Abstract Multi-objective particle swarm optimization (MOPSO) is a promising meta-heuristic to solve multi-objective problems (MOPs). Previous works have shown that selecting a proper combination of leader and archiving methods, which is a challenging task, improves the search ability of the algorithm. A previous study has employed a simple hyper-heuristic to select these components, obtaining good results. In this research, an analysis is made to verify if using more advanced heuristic selection methods improves the search ability of the algorithm. Empirical studies are conducted to investigate this hypothesis. In these studies, first, four heuristic selection methods are compared: a choice function, a multi-armed bandit, a random one, and the previously proposed roulette wheel. A second study is made to identify if it is best to adapt only the leader method, the archiving method, or both simultaneously. Moreover, the influence of the interval used to replace the low-level heuristic is analyzed. At last, a final study compares the best variant to a hyper-heuristic framework that combines a Multi-Armed Bandit algorithm into the multi-objective optimization based on decomposition with dynamical resource allocation (MOEA/D-DRA) and a state-of-the-art MOPSO. Our results indicate that the resulting algorithm outperforms the hyper-heuristic frame-

This work was supported by CNPq, National Council for Scientific and Technological Development—Brazil (Productivity Grant Nos. 305986/2012-0 and Program Science Without Borders Nos. 200040/2015-4).

✉ Olacir R. Castro Jr.
olacirjr@gmail.com

Gian Mauricio Fritsche
gianfritsche@gmail.com

Aurora Pozo
aurora@inf.ufpr.br

[1] Computer Science's Department, Federal University of Paraná, Curitiba, Paraná, Brazil

work in most of the problems investigated. Moreover, it achieves competitive results compared to a state-of-the-art MOPSO.

Keywords Multi-objective particle swarm optimization · Multi-objective · Hyper-heuristics · Leader selection · Archiving · Fitness-rate-rank-based multi-armed bandit · Adaptive choice function

1 Introduction

Multi-Objective Problem (MOP) is a kind of optimization problem that presents two or more objective functions to be optimized simultaneously. Since these functions are usually in conflict, the optimal solution for a MOP is a set of trade-off solutions called the Pareto optimal set. Multi-objective particle swarm optimization (MOPSO), a multi-objective version of the Particle Swarm Optimization (PSO) (Kennedy and Eberhart 1995), has proven to be well-suited for complex MOPs. Two main modifications are usually made to expand a PSO algorithm into a Pareto-based MOPSO. First, the creation of an external archive to store the non-dominated solutions; second, the use of a leader selection method to select a global guide from the archive (Reyes-Sierra and Coello 2006).

Despite the excellent results presented in the literature, several studies have been carried out to improve the performance of MOPSOs. Among them, this study highlight the ones that show that selecting a proper combination of leader and archiving methods, which is a challenging task because they are problem-dependent, helps the algorithm to improve its search ability (Britto and Pozo 2012) (Castro Jr et al. 2012).

To deal with this challenge, previous works (Castro Jr. and Pozo 2015, 2014) proposed the use of a hyper-heuristic to select the leader and archiving methods for MOPSO dynamically. Additionally, they implemented a framework called Hyper-heuristic Multi-Objective Particle Swarm Optimization (H-MOPSO). The use of hyper-heuristics to dynamically choose the components of MOPSO presents two significant advantages. First, it eliminates the necessity of manually selecting the components, and second, it allows the application of different leader and archiving methods during the search. Alternating these components allows combining techniques that prioritize convergence and others that prioritize diversity, reducing the probability of being stuck in optimal local fronts.

The heuristic selection mechanism used in H-MOPSO was a single roulette wheel that had its probabilities adjusted according to the performance of a given component (a pair of leader selection and archiving methods). Despite the simplicity of the heuristic selection mechanism used, in most cases, the proposed algorithm was able to outperform all the components used separately (Castro Jr. and Pozo 2014) and a state-of-the-art optimizer, MOEA/D-DRA (Zhang et al. 2009; Castro Jr. and Pozo 2015). A previous study compared the H-MOPSO to MOEA/D-DRA. The present study compares H-MOPSO to two other algorithms: MOEA/D-FRRMAB and AgMOPSO. Both MOEA/D-FRRMAB and MOEA/D-DRA use a multi-objective approach based on decomposition and differential evolution (DE). MOEA/D-FRRMAB is state of the art multi-objective evolutionary algorithm that uses the FRRMAB heuristic selection method to enhance MOEA/D (Li et al. 2014a; Gonçalves et al. 2015b; do Nascimento Ferreira et al. 2016; Trivedi et al. 2016; Hitomi and Selva 2017; Gonçalves et al. 2015). The AgMOPSO is a state of the art MOPSO (Zhu et al. 2017). In AgMOPSO, the problem is decomposed into subproblems; then each particle is assigned to optimize each subproblem. The AgMOPSO uses an archive-guided velocity update method. Besides, it uses an immune-based evolutionary strategy to speed up convergence.

Based on those previous results, the hypothesis behind the study presented here is that the heuristic selection mechanism in H-MOPSO strongly affects the performance of the algorithm. This study evaluates two heuristic selection methods that showed outstanding results in the literature: Adaptive Choice Function (ACF) (Gonçalves et al. 2015a; Drake et al. 2012) and Fitness-Rate-Rank-based Multi-Armed Bandit (FRRMAB) (Li et al. 2014a; Gonçalves et al. 2015c).

Choice function methods are extensively used in the hyper-heuristic literature (Gonçalves et al. 2015a; Drake et al. 2012; Cowling et al. 2001; Burke et al. 2013; Maashi et al. 2014; Blazewicz et al. 2013; Guizzo et al. 2015). Furthermore, the ACF employed in this work is self-adaptive. This characteristic avoids the manual configuration of parameters.

Multi-Armed Bandit algorithms (MAB) have shown encouraging results in the Adaptive Operator Selection (AOS) literature (Li et al. 2014a; Gonçalves et al. 2015c; Krempser et al. 2012; Maturana et al. 2012; Li et al. 2011; Fialho et al. 2010; Ferreira et al. 2015; Sabar et al. 2015), especially the FRRMAB (Li et al. 2014a), recently proposed for multi-objective AOS.

This research incorporates these methods into the H-MOPSO framework, along with the original roulette and a random selection mechanism. Then, an experimental study compares the four H-MOPSO variants when optimizing multi-objective problems. The experiments use representative problems from the DTLZ (Deb et al. 2002) and WFG (Huband et al. 2006) families.

Based on the results of this first experimental study, a second empirical study is carried out using the best performing algorithm variant. In this study, we investigate which group of low-level heuristics yields the best results: only archiving, only leader selection or both methods simultaneously. Moreover, we also investigate the influence of the interval used to replace the low-level heuristics. A final study compares the H-MOPSO framework to the MOEA/D-FRRMAB (Li et al. 2014a) hyper-heuristic framework and to a state-of-the-art MOPSO, the AgMOPSO (Zhu et al. 2017).

The remainder of this paper presents the following topics: Sect. 2 presents the background concepts used in this research. Section 3 describes H-MOPSO, and Sect. 4 presents empirical studies evaluating the H-MOPSO variants proposed here. Finally, the Sect. 5 present the conclusions.

2 Background

This section presents some background concepts used in this paper. Section 2.1 describes multi-objective concepts and Sect. 2.2 surveys PSO and MOPSO. The Sects. 2.3 and 2.4 present the archiving and leader selection strategies used in this work.

The metrics employed to assess the performance of the algorithms in the empirical study, Inverted General Distance (IGD) and hypervolume, are presented in Sects. 2.5.1 and 2.5.2, respectively. Section 2.5.3 describes the $R2$ metric used within H-MOPSO to evaluate the performance of the chosen component. Finally, some concepts of hyper-heuristics, as well as the FRRMAB and ACF heuristic selection methods, are presented in Sect. 2.6.

2.1 Multi-objective optimization

Multi-Objective Optimization Problems (MOPs) require the simultaneous optimization (maximization or minimization) of two or more objective functions. These objectives are usually in conflict, which means the problems do not have only one optimal solution (as in single objective optimization problems), but a set of them. A general MOP without constraints can be defined as optimizing $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))$, where $\mathbf{x} \in \Omega$ is an n-dimensional decision variable vector $\mathbf{x} = (x_1, \ldots, x_n)$ from a universe Ω, and m is the number of objective functions. An objective vector $\mathbf{u} = \mathbf{f}(\mathbf{x})$ dominates a vector $\mathbf{v} = \mathbf{f}(\mathbf{y})$, denoted by $\mathbf{u} \preceq \mathbf{v}$ (in case of minimization) if $\mathbf{u}$ is partially less than $\mathbf{v}$ i.e., $\forall i \in \{1, \ldots, m\}, u_i \leq v_i \wedge \exists j \in \{1, \ldots, m\} : u_j < v_j$. A vector $\mathbf{u}$ is non-dominated if there is no $\mathbf{v}$ that dominates $\mathbf{u}$. Given that $\mathbf{u} = \mathbf{f}(\mathbf{x})$, if $\mathbf{u}$ is non-dominated, then $\mathbf{x}$ is Pareto optimal. The set of Pareto optimal solutions is called the Pareto optimal set, and the image of these solutions in the objective space is called the Pareto front (Coello et al. 2006).

2.2 MOPSO

Particle Swarm Optimization (PSO) (Kennedy and Eberhart 1995) is a stochastic optimization technique inspired by the swarming or collective behavior of biological populations developed by James Kennedy and Russel Eberhard in 1995. PSO is similar to the Genetic Algorithm (GA) in the sense that these two evolutionary heuristics are population-based search methods. However, different to GA, PSO has no evolution operators such as crossover and mutation. Instead, PSO maintains a population and tweak its members in response to new discoveries about the space (Eberhart and Shi 2001; Luke 2013).

In PSO simulation, the behavior of an individual (or particle) is affected by either the best local or the best global solutions. The approach then uses the concept of the population (or swarm) and a measure of performance similar to the fitness value used with evolutionary algorithms. Also, the adjustments are analogous to the utilization of a crossover operator. Note that PSO allows individuals to benefit from their past experiences, whereas in an evolutionary algorithm, the current population is usually the only "memory" used. PSO has achieved success for both continuous non-linear and discrete binary optimizations (Coello et al. 2006). To update the position of the particle i ($\mathbf{x}_i$) at the iteration t, PSO employs Eq. (1) (Reyes-Sierra and Coello 2006):

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

where the current velocity $\mathbf{v}_i^t$ is given by Eq. (2):

$$\mathbf{v}_i^t = \underbrace{\omega \mathbf{v}_i^{t-1}}_{\text{inertial}} + \underbrace{C_1 r_1 (\mathbf{x}_{b_i} - \mathbf{x}_i^{t-1})}_{\text{cognitive}} + \underbrace{C_2 r_2 (\mathbf{x}_{g_i} - \mathbf{x}_i^{t-1})}_{\text{social}} \tag{2}$$

In Eq. (2), $\mathbf{x}_{b_i}$ is the best solution found so far by $\mathbf{x}_i$. The best solution that the entire swarm has found is $\mathbf{x}_{g_i}$ (also known as global leader). The inertia weight (ω) of the particle controls the trade-off between global and local experience. The variables 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 personal and global best particles.

In (Shi and Eberhart 1998), it is pointed out that Eq. (2) consists of three parts. The first is the inertial component, which represents the previous velocity of the particle, the second is the cognitive part, which represents the own thinking of the particle, and the third is the social component which represents the collaboration among the particles.

The similarities between PSO and evolutionary algorithms make evident the notion that using a Pareto ranking scheme can be the straightforward way to extend the approach to handling multi-objective problems, hence, creating a Multi-Objective Particle Swarm Optimization (MOPSO). However, to extend PSO into MOPSO some modifications need to be done to deal with the fact that the solution of a multi-objective problem is not a single one, but a set of non-dominated solutions (Durillo et al. 2009; Coello et al. 2006). Extending PSO into MOPSO have three main issues have to be considered (Durillo et al. 2009; Reyes-Sierra and Coello 2006; Parsopoulos and Vrahatis 2008).

1. How to retain the non-dominated solutions found during the search?
2. How to select particles to be used as leaders?
3. How to maintain diversity in the swarm to avoid convergence to a single solution?

For the first issue, the option is to use an external archive or repository, as traditional MOEAs do. However, the number of non-dominated solutions found during the search can increase quickly. This increase in the size of the repository makes its maintenance computationally expensive. Also, the memory usage to store these solutions can largely increase. To avoid such problems, the size of the archive is usually bounded, so it only keeps the N "best" solutions. However, when the archive is full, and a new non-dominated solution is generated, an archiver method is needed to decide whether or not to include the new solution in the archive, and if it does, which existing solutions to remove. There are several archivers available in the literature, and the interested reader is referred to (Britto and Pozo 2012) for further details. Three of these methods are described in Sect. 2.3.

The second issue is particular to MOPSO as it refers to leader selection. The selection of the leader is an important task since the leader is expected to guide the search toward better regions. Also, the leaders need to provide diversity for the search, avoiding the convergence to a single solution. When solving single-objective optimization problems, determining the leader that each particle uses to update its position is a trivial task, since the ranking of solutions according to their fitness value is possible. However, in MOPs, there is usually a set of equally good leader candidates that can be kept in the repository. In the literature, there are different methods for leader selection, and the interested reader is referred to (Castro Jr et al. 2012) for further details. The Sect. 2.4 describes three of these methods. The third issue is left to an appropriate combination of leader and archiver methods.

2.3 Archiving strategies

Researchers are aware of the importance of archiving strategies, and they have proposed different archiving strategies over the past years. In (Britto and Pozo 2012), they conducted an empirical study involving twelve archiving methods. The study concluded that best results could be achieved by archiving methods that do not limit the repository size. Although, it is not always possible to run an algorithm without limiting the repository size since the number of non-dominated solutions exponentially grows with the increased number of the objectives. Therefore, this study focus on archive methods that limit the size of the repository, such as the Ideal and the MGA archive methods (Britto and Pozo 2012). Among the methods that limit the size of the repository, good results regarding convergence were obtained using the Ideal method. When considering diversity, the Multi-level Grid Archiving (MGA) (Laumanns and Zenklusen 2011) achieved the best results.

Since in this work the focus is on archivers that limit the size of the repository, the Ideal and the MGA archivers were selected (Britto and Pozo 2012). Also, the Crowding Distance (CD) (Deb et al. 2000) archiver, one of the most traditional archivers, used by the NSGA-II and SMPSO algorithms, was included. These three archiving methods are described next.

The **Ideal** archiver, proposed in (Britto and Pozo 2012), seeks to achieve convergence in the search. First, it obtains the Ideal point, which is a vector containing the best value for each objective function among the solutions contained in the archive. Then, it calculates the Euclidean distance between each solution in the repository and the Ideal point and removes the farthest one.

The **Multi-level Grid Archiving (MGA)**, proposed in (Laumanns and Zenklusen 2011), divides the objective space into boxes, and then it observes the dominance relation between the boxes. If the new solution to be added is located in a dominated box, it is not included. Otherwise, it is included, and a solution located in a dominated box is randomly removed. If no dominance relation is found between the boxes that contain solutions, the MGA splits the objective space again into smaller boxes until it finds a dominated box.

The **Crowding Distance (CD)** indicator (Deb et al. 2000) is used as archiving criterion in the original SMPSO. CD is a diversity estimator that measures the size of the largest cuboid enclosing a point without including any other point. This metric is used to estimate the density of solutions surrounding a particular individual in the population. In the CD archiver, the new solution is firstly temporarily inserted, surpassing the capacity of the repository. Then, the archiver calculates the CD of all solutions in the repository. Finally, it removes the solution that obtains the smallest value (in a more crowded region).

2.4 Leader selection methods

Castro Jr et al. (2012) conducted an empirical study involving six leader selection methods. They selected four of them from the literature and proposed a new one, and the other one selects randomly. The study concluded that, besides being problem-specific, in general, best results were achieved by using the methods Sigma (Mostaghim and Teich 2003) and NWSum (Padhye et al. 2009), followed by Crowding Distance (CD) (Deb et al. 2000). Next, this section presents these three methods:

In the **NWSum** method, proposed in (Padhye et al. 2009), the weighted sum of the objective vector of a leader candidate is considered as the criterion to select the leader of a particle, where the objective vector values of the current particle are the weight of the sum. In this method, the leader candidate that maximizes the weighted sum of the particle is selected to lead it. By using this approach, a particle chooses as leader a solution that has objective values similar to its own, which introduces convergence in the algorithm.

The **Sigma** method, proposed in (Mostaghim and Teich 2003), is based on the comparison of the Sigma vectors. These vectors are calculated using the objective values of each solution. The Sigma vector can be represented as a vector starting from the origin and crossing the solution under consideration. It is composed of $\binom{m}{2}$ elements, where m is the objective number. At each iteration, the method calculates the Sigma vectors for all the solutions in the repository and all the particles in the population. Then, each particle selects as leader the solution with the closest Sigma vector, considering the Euclidean distance.

In the **Crowding Distance CD** leader selection method, for each particle, two solutions are randomly chosen from the repository, and the method selects the one that has a higher CD value (located in a less crowded region).

2.5 Performance measures

Two performance measures are presented and used in this work to evaluate the results: IGD and hypervolume. Besides, this section also presents the $R2$ indicator used by H-MOPSO. The hypervolume indicator is the only quality measure that is known to be strictly monotonic regarding Pareto dominance. However, the hypervolume has a bias to the knee on concave fronts (Jiang et al. 2014). Then to complement the analysis, we consider the IGD. The IGD can measure at the same time the convergence and diversity of an approximation front and has a low computational cost. Many recent studies have used both, IGD and hypervolume, to compare experimental results of multi-objective algorithms (Elarbi et al. 2017; Yuan et al. 2016; Li et al. 2014b, 2015, 2014a).

2.5.1 Inverted generational distance

Inverted Generational Distance (IGD) (Coello and Cortés 2005) is a widely used metric (Li et al. 2014a; Zhang et al. 2009; Gonçalves et al. 2015a), especially in the many-objective community, due to its ability to measure at the same time the convergence and diversity of a Pareto front approximation, as well as having a low computational cost. IGD computes the smallest distance between each point in discretized true Pareto front (PF_t) and the objective vectors from the solutions in the approximated Pareto front (PF_k). By doing that, IGD allows observing if PF_k is well diversified and converges to PF_t. Its main disadvantage is the need of a well-diversified discretization of PF_t, which is not always available or easy to obtain. Equation (3) presents the mathematical formulation for IGD.

$$IGD \triangleq \frac{\left(\sum_{i=1}^{|PF_t|} d_i^p\right)^{1/p}}{|PF_t|} \tag{3}$$

where $|PF_t|$ is the number of solutions in PF_t, $p = 2$ and d_i is the Euclidean distance between each member of PF_t and the nearest objective vector in PF_k. A result of 0 indicates $PF_k = PF_t$; any other value indicates how far PF_k is from PF_t.

As in (Li et al. 2015), here the PF_k is generated based on a set of well-distributed weight vectors. These are the same weights used in the MOEA/D-FRRMAB algorithm and the calculation of the $R2$ indicator. Each point that composes the discretized PF_k is the intersection between the true Pareto surface of the problem (known *a priori*) and a line that begins in the origin and passes through a weight vector.

2.5.2 Hypervolume

Hypervolume (Zitzler and Thiele 1999) is another extensively used metric (Coello et al. 2006; While et al. 2012; Bader et al. 2010). It expresses, in one scalar, both the convergence and diversity of a Pareto set approximation by measuring the size of the portion of the objective space that is dominated by this set of solutions. Furthermore, this indicator is strictly monotonic, i.e., when a Pareto set approximation dominates another, the indicator value of the former will be greater than the latter. However, hypervolume is very sensitive to the scaling of the objectives and the presence of extreme points (Coello et al. 2006; While et al. 2012; Bader et al. 2010). Hypervolume is defined as follows (Bringmann and Friedrich 2010):

$$HV(PF_k) := VOL\left(\bigcup_{(u_1,\ldots,u_m)\in PF_k} [r_1, u_1] \times \cdots \times [r_m, u_m]\right) \tag{4}$$

where $VOL(.)$ is the Lebesgue measure and the reference point $\mathbf{r}$ is the nadir (anti-optimal or "worst possible") point in space. The greater the hypervolume value of a set, the better the approximation of the Pareto set.

The main disadvantage of the hypervolume indicator is that its calculation is computationally expensive. Even the best-known algorithms for computing hypervolume have running times exponential in the number of objectives, which restricts the use of hypervolume-based methods to problems with up to about ten objectives (Bader et al. 2010; Phan and Suzuki 2013; While et al. 2012).

2.5.3 R2 indicator

Another quality indicator is $R2$ (Hansen and Jaszkiewicz 1998). It was originally proposed to assess the relative quality of two approximation sets. It is an indicator that simultaneously evaluates the convergence and diversity of a Pareto front approximation, and has a low computational cost (Brockhoff et al. 2012). Assuming the standard weighted Tchebytcheff function with a particular reference point $\mathbf{z^*}$, the indicator can be used to evaluate the quality of a single Pareto front PF_k against $\mathbf{z^*}$ (Brockhoff et al. 2012; Phan and Suzuki 2013). The weighted Tchebytcheff function depends on a set of weight vectors $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m) \in \Lambda$. Usually, these weights are uniformly distributed over the objective space. In this case, the $R2$ indicator can be described as:

$$R2(PF_k, \Lambda, \mathbf{z^*}) = \frac{1}{|\Lambda|} \sum_{\boldsymbol{\lambda} \in \Lambda} \min_{\mathbf{u} \in PF_k} \{ \max_{j \in \{1,\ldots,m\}} \{\lambda_j |z_j^* - u_j|\}\} \tag{5}$$

A utopian point is usually used as the reference point $\mathbf{z^*}$. A utopian point is a point that is never dominated by any feasible solution in the objective space, i.e., (0,0) for a bi-dimensional objective space where all the objective values are greater than or equal to 0. The set of weight vectors can be generated using the same approach as in (Li et al. 2015), which assigns points on a normalized hyperplane which is equally inclined to all objective axes and has an interception with each axis (an $m - 1$ dimensional unit simplex).

A lower $R2$ value indicates that an individual set is closer to the reference point, $R2 = 0$ when an individual is positioned on the reference point (Phan and Suzuki 2013). Hypervolume is the only indicator which fulfills the property of strict monotonicity, and $R2$ is only weakly monotonic. It means that the $R2$ value of a Pareto set approximation that dominates another will be better or equal to the $R2$ value of the dominated set. However, $R2$ has a lower computational cost.

2.6 Hyper-heuristics

The task of choosing appropriate parameters or algorithms to solve an optimization problem is often hard. In this context, the hyper-heuristic approach emerges as a high-level methodology which—given a particular problem instance or class of instances, and some low-level heuristics, or components—automatically produces an adequate combination of them to solve the problem efficiently (Burke et al. 2013). In that way, hyper-heuristics operate over the heuristic space, rather than the solution space.

It is possible to classify the hyper-heuristics according to the nature of the heuristic search space: *heuristic selection* are methodologies for choosing or selecting existing heuristics, and *heuristic generation* are the ones for generating new heuristics from the components of existing ones. Furthermore, are considered learning algorithms the hyper-heuristics that use some feedback from the search process. It is possible to classify the hyper-heuristics according to the source of the feedback. One class is the *online* methods when learning takes place while the algorithm is solving an instance of a problem. Another class is the *offline* methods, they gather the knowledge in the form of rules or programs from a set of training instances that hopefully generalize to solving unseen instances (Burke et al. 2013).

This study focuses on online heuristic perturbation selection methods. These techniques aim to improve a candidate solution by automatically selecting and applying a heuristic. The objective is to get the best heuristic to be used depending on the search stage and the search space properties, allowing the use of low-level heuristics that prioritize convergence or diversity along the search, and reducing the chances of getting stuck in local optimal Pareto fronts. These techniques aim to handle the Exploration *vs.* Exploitation dilemma (EvE). According to EvE, it is as important to apply operators (or heuristics) with better performance more often, as it is to employ those that have not been used recently, to evaluate its current performance.

Usually, a hyper-heuristic to select perturbative heuristics combines two separate components: Heuristic selection and move acceptance. The **heuristic selection** component is responsible for selecting and evaluating the low-level heuristics. There are different heuristic selection methods; the simplest are methods which have no learning mechanism, e.g., a random selection. There are also rank-based heuristic selection methods, which use an update rule to rank the low-level heuristics based on their performance, where the best-ranked heuristics are more likely to be selected. Others use a more sophisticated method considering the history of performances obtained by the low-level heuristics. It is important to note that even simple methods are usually better than applying any low-level heuristic individually (Castro Jr. and Pozo 2015; Burke et al. 2013). Among the most known selection methods are Choice Function (Maashi et al. 2014; Cowling et al. 2001); Reinforcement learning; and Adaptive Operator Selection (Burke et al. 2013).

The **move acceptance** component evaluates the solution generated by the heuristic applied using some quality information received from the problem domain, usually the fitness value of the solution. Then, the solution is either accepted or rejected, depending on its performance and on the rules of the move acceptance method. If a solution is better than a previous one, it is accepted; otherwise, an acceptance criterion is used. The use of an appropriate move acceptance method may substantially increase the optimization performance.

According to (Burke et al. 2013), the decision of the move acceptance seems to be more important than the heuristic selection. This strategy can be either deterministic or non-deterministic. Deterministic methods make the same acceptance decision regardless of the stage of the search, using the current and new solution(s). Non-deterministic approaches might generate different decisions for the same input. Most non-deterministic move acceptance methods require additional parameters like the current iteration number. This study uses the Improving and Equal (IE) deterministic move acceptance strategy, due to its simplicity and the results presented in (Bilgin et al. 2007). The IE accepts a new solution if it improves or maintains the previous score value.

In the study presented here, the performance of two heuristic selection methods, which showed good results in the literature, are investigated: Choice Function (Cowling et al. 2001) and Multi-armed Bandit (Fialho et al. 2010). Furthermore, this study compares them to a simple roulette-wheel-based heuristic selection as used in (Castro Jr. and Pozo 2015), and to a random heuristic selection. The following sections describe these methods.

2.6.1 Choice function

The Choice Function method is an online learning heuristic selection based on ranking, proposed by (Cowling et al. 2001). The ranking is made based on a function that has three components (Ozcan et al. 2009):

$$cf(h_i) = \alpha f_1(h_i) + \beta f_2(h_j, h_i) + \delta f_3(h_i) \tag{6}$$

1. The f_1 component is the quality of the low-level heuristic h_i. It is responsible for increasing the exploitation by raising the probability of the heuristics with the best performance.
2. f_2 is the quality of the pair of low-level heuristics (h_i, h_j) when applied together. The objective is to find a cooperative behavior between heuristics that perform well when applied together.
3. The function also has an exploration component (f_3). It computes the elapsed time since the last application of the low-level heuristic h_i. It increases the probability of the low-level heuristics that were not applied recently to evaluate its current performance.

After the low-level heuristic is applied, the quality information is used to update the CF components. There are different ways of updating f_1 and f_2, for instance, using: the last quality information received (Guizzo et al. 2015); or the quality information accumulated since the beginning of the search (Drake et al. 2012); or the mean value of the quality information (Gonçalves et al. 2015a). For each heuristic i, the f_2 component is updated considering the quality of i and the quality of the last applied heuristic j. The f_3 component can be computed using different information as well, for instance: the elapsed time in seconds since the low-level heuristic i was applied, or the number of heuristic selections passed since the last time that i was applied.

The function value $cf(h)$ of the heuristic h_i is the sum of the three components (f_1, f_2 and f_3), weighted by the scale factor parameters (α, β and δ). After updating the ranking, there are different ways of performing the selection, such as: getting the best-ranked heuristic or using some simple roulette rule.

To avoid the parameter configuration of α, β and δ, an Adaptive Choice Function (ACF) is proposed in (Drake et al. 2012) using the following equation:

$$cf(h_i) = \phi f_1(h_i) + \phi f_2(h_j, h_i) + \delta f_3(h_i) \tag{7}$$

where ϕ is an exploitation factor for the best-performing heuristics, and δ is an exploration factor, responsible for increasing the probability of the not recently applied heuristics. The parameter ϕ is set to 0.99 each time the heuristic improves the quality of the solution and is decreased by 0.01 otherwise. The δ parameter is set to $(1 - \phi)$. The objective is to increase exploitation when the quality of the solution is improving and to increase exploration when the current best operators cannot increase the quality of the solution.

In (Gonçalves et al. 2015a), two modifications are proposed: the use of a scale factor (SF) parameter (since the measures used in f_1 and f_2 may be in different scales when compared to f_3); and the use of the mean values of f_1 and f_2, instead of the accumulated values. Among the different choice function versions available, this study uses the ACF, due to its results in previous studies.

2.6.2 Multi-armed Bandit

The Multi-Armed Bandit (MAB) is a problem that considers a set of K independent arms, with an unknown probability of being chosen. The goal is to select the arms that maximize the reward accumulated over time. The Multi-armed Bandit problem fits in the Exploitation vs. Exploration (EvE) dilemma. Many algorithms have been proposed to tackle the MAB problem, one of them being the Upper Confidence Bound (UCB), which provides asymptotic optimality guarantees. In UCB, the selected arm is the one that maximizes the UCB function (Eq. (8)) (Li et al. 2014a; Gonçalves et al. 2015c).

$$\hat{q}_i + C \times \sqrt{\frac{2 \times \ln \sum_{j=1}^{K} n_{j,it}}{n_{i,it}}} \tag{8}$$

Each arm has an empirical quality estimated ($\hat{q}_i$). Where $n_{i,it}$ is the number of times that an operator i has been used before the iteration it. Similarly to the f_1 component from Choice Function, $\hat{q}_i$ is responsible for increasing exploitation, i.e., enhancing the probability of the heuristics with the best performance. The C parameter is a scale factor between exploration (the right term measures how frequently the arm has been tried) and exploitation (the left term, which measures the quality of the arm) (Li et al. 2014a; Gonçalves et al. 2015c).

Some UCB-based algorithms have been used for Adaptive Operator Selection (AOS), as it aims to handle the EvE dilemma. The goal was to select the operators that increase the quality of the optimization solution output. One of these algorithms is the Fitness-Rate-Rank-Based Multi-Armed Bandit (FRRMAB), proposed in (Li et al. 2014a). Usually, in UCB-based algorithms, the raw value of the fitness improvements is used as the reward. However, the range of these values may vary depending on the problem and the search stage. To deal with that, FRRMAB uses a fitness improvement rate (FIR), defined as (Eq. (9)):

$$FIR = \frac{pf - cf}{pf} \tag{9}$$

where the FIR of the operator is the difference between the fitness of the solution before (pf) and after (cf) applying the operator, divided by the old fitness (pf). Then, a sliding time window stores the FIR values of the last W applications. Moreover, the $Reward$ of an operator (i) is the sum of all FIR of that operator in the sliding window. Then, all operators are ranked by reward, and a decaying factor (D) is applied to increase the probability of the best-ranked operators (Eqs. 10 and 11):

$$Reward_i = \sum_{k \leftarrow 0}^{W} \begin{cases} FIR_k^{op} & \text{if } op = i \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Where W is the sliding window size; it means that only the last W fitness improvements are considered.

$$Decay_i = D^{Rank_i} \times Reward_i \tag{11}$$

Finally, the credit value of the operator i is computed as (Eq. (12)):

$$FRR_i = \frac{Decay_i}{\sum_{j=1}^{K} Decay_j} \tag{12}$$

To select the operators, the empirical reward $\hat{q}_i$ is replaced by the FRR_i value in the UCB function. Moreover, n_i indicates the number of times that the operator (i) has been applied in the last W iterations. Among the different algorithms proposed to tackle the MAB problem, this study uses the FRRMAB, due to its results obtained in previous studies.

2.6.3 Roulette-based heuristic selection

The roulette based selection mechanism consists of a roulette wheel which updates the probability of each low-level heuristic according to the difference in the quality value obtained before and after the application of the selected heuristic. The roulette wheel starts with equal probability of choosing any of the low-level heuristics available. The roulette updates the probabilities at each iteration, based on the quality information. In a given iteration, if the selected heuristic improves or maintains its quality, the probability of the selected heuristic is increased in the roulette at the same time it decreases for the other heuristics, and the solution is accepted. However, if quality deteriorates, the heuristic used has its probability decreased, and it is increased the probabilities of the others. This simple hyper-heuristic rewards or punishes the heuristics by increasing or decreasing their participation in the search. A small minimum probability is considered for the heuristics so they are not completely removed from the search since they can show bad results at some stage but perform well later in the search procedure.

In this research, based on (Castro Jr. and Pozo 2015), the credit assignment was fixed: according to the performance of the low-level heuristic, its probability in the roulette is changed by one-tenth of the initial probability. A minimum probability of 0.5% is kept to guarantee that a low-level heuristic will not be removed from the roulette (it means 0% probability).

2.6.4 Random heuristic selection

A random method was also implemented to assess how much the heuristic selection influences the search. This strategy just randomly selects one of the low-level heuristics available without any learning.

3 Hyper-MOPSO

This section describes the Hyper-heuristic Multi-Objective Particle Swarm Optimization (H-MOPSO) framework (Castro Jr. and Pozo 2015, 2014), which dynamically selects the leader and archive methods for a MOPSO. Furthermore, the framework allows the implementation of different hyper-heuristic selection methods. For instance: Roulette, ACF, FRRMAB, and Random. In H-MOPSO, a low-level heuristic is a combination of a leader and an archiving method.

Algorithm 1 illustrates the H-MOPSO. The highlighted steps are the hyper-heuristics steps introduced in a generic MOPSO. The first is the initialization of the hyper-heuristic, this step is specific for each hyper-heuristic. For instance, if the selection method is the roulette, this step sets the initial probabilities for all the low-level heuristics to one divided by the number of low-level heuristics.
The second step is the heuristic selection itself. The hyper-heuristic selection method returns a low-level heuristic according to its strategy.
The third step is an evaluation of the performance of the selected low-level heuristic. In this research, this information is called ($Reward$) and is computed as the normalized difference between the quality of the new repository ($R2_{new}$) and the old ($R2_{old}$) as described by (Eq. (13)):

$$Reward = \frac{R2_{old} - R2_{new}}{R2_{old}} \tag{13}$$

This $Reward$ is then used to update the preferences of the heuristic selection method. For instance, it updates the roulette probabilities; or the Choice Function values; or the UCB values for FRRMAB.

The last step introduced is the move acceptance criterion, where the H-MOPSO decides if the new repository will replace the old one. This work uses the improving and equal (IE) move acceptance method due to its simplicity, and good results obtained in (Bilgin et al. 2007). In IE a new solution is accepted if it improves or maintains the previous score value. On the H-MOPSO, the use of IE means that: the repository generated on the current iteration, using the selected low-level heuristic, is only accepted if its $R2$ value is better than the $R2$ value of the previous repository $R2_{old}$. In other words, if the $R2$ value of the new repository $R2_{new}$ is worse than the previous one, then a copy of the previous repository is restored and used to replace the repository generated on the current iteration.

The introduction of these steps on MOPSO composes the H-MOPSO framework (Algorithm 1): In the first steps, the algorithm initializes the swarm randomly and evaluates the particles. Then, the repository is initialized using the non-dominated solutions from the population. As H-MOPSO uses the number of particles equal to the size of the repository, an archiving strategy is not necessary in the initialization, as even in the worst case (all solutions non-dominated) the size of the repository does not have to be limited. Next, the algorithm initializes the pool of low-level heuristics and the hyper-heuristic strategy according to the heuristic selection method. After initialization steps, the algorithm enters the main loop (steps 5– 18).

The first step of the main loop is to select a low-level heuristic (a combination of archiving and leader methods) using some heuristic selection method (step 6). Next,

Algorithm 1: H-MOPSO

```
1  initialize the swarm;
2  evaluate the particles;
3  initialize the repository;
4  initialize hyper-heuristic;
5  while not reached the stop criteria do
6      heuristic selection;
7      foreach $particle \in swarm$ do
8          leader selection;
9          update velocity;
10         update position;
11         turbulence;
12         fitness evaluation;
13         update $pBest$;
14     end foreach
15     update repository ($swarm$);
16     evaluate heuristic performance;
17     move acceptance;
18 end while
19 return $repository$;
```

for each particle, the MOPSO steps are executed (steps 7–14). The particle selects a global leader from the repository using the leader selection method. Then, the particle velocity and position are updated based on the cognitive and social leaders. After, a turbulence method is applied (mutation). Finally, the algorithm evaluates the fitness of the particle, and the local best ($pBest$) is updated. Next, the repository is updated, using the selected archiving method (step 15). The quality of the low-level heuristic applied is evaluated in step 16, and the probabilities are updated. In step 17, the move acceptance method is used to verify if the new repository is going to be accepted (replace the old one) or discarded (keep using the old one). When the algorithm reaches the stop criterion, the output is the set of solutions contained in the repository (step 19).

An important idea of H-MOPSO is to join the strengths of several already proposed components, i.e., archive and leader methods, in a collaborative framework. The selection of the components that are going to compose these low-level heuristics follows the principle of diversity to promote the collaboration between them. Hence, if it has an archive method that improves the convergence as the Ideal, it has the necessity of including one for diversity as the MGA. Following this principle, this study uses the low-level heuristics presented at Table 1. Besides, the further investigation of the inclusion of other low-level heuristics is an important issue.

Another important design choice is related to the archiving methods. It is possible to note that until the repository size limit exceeds, every archiving method will have the same behavior, i.e., to accept all non-dominated solutions. It means that, at the initial iterations, the hyper-heuristic cannot estimate the quality of different archiving methods. Therefore, it makes sense to begin the Hyper-heuristic selection method only when the repository becomes full.

Table 1 Low-level heuristics available for the heuristic selection methods

Heuristic	Archiver	Leader selection
H1	CD	CD
H2	CD	NWSum
H3	CD	Sigma
H4	MGA	CD
H5	MGA	NWSum
H6	MGA	Sigma
H7	Ideal	CD
H8	Ideal	NWSum
H9	Ideal	Sigma

However, then, the question is which base algorithm should be used until the hyper-heuristic selection method starts. Here, this was decided based on literature review and empirical experiments with MOPSOs. An aspect that has been observed in MOPSO is that the velocity of the particles can become too high in some conditions, generating erratic movements towards the limits of the decision space. The Speed-constrained Multi-objective PSO (SMPSO) (Nebro et al. 2009) algorithm was proposed to avoid such situations. It presents a velocity constriction mechanism based on a factor χ that varies based on the values of the influence factor of the personal (C_1) and social (C_2) leaders, from the velocity equation. Further, the application of a velocity constriction mechanism forces that the range of (C_1) and (C_2) be from [1.5, 2.5]. Besides, in SMPSO, the Crowding Distance metric (Deb et al. 2000) is used for both the leader selection method and the archiving strategy.

H-MOPSO uses the SMPSO algorithm as the baseline, and as a consequence of these design decisions, H-MOPSO uses a velocity constriction mechanism (Durillo et al. 2009; Nebro et al. 2009) and a polynomial mutation as a turbulence factor. The objective of this research is to perform a comparison of four heuristic selection methods, using the SMPSO as the basis. So, only the parameters of the heuristic selection methods were tuned, empirically, in this study.

4 Empirical study

This section contains the experimental study conducted in this work. The parameters used in the experiments are described in Sect. 4.1. Then, Sect. 4.2 presents a comparative study of the four heuristic selection methods: ACF, FRRMAB, Roulette, and Random. Section 4.3 presents a sensitivity analysis of an important parameter of the H-MOPSO framework: the interval used to replace the low-level heuristic. In Sect. 4.4 the H-MOPSO framework using the Roulette heuristic selection, called H-MOPSO-Roulette, is compared to MOEA/D-FRRMAB, a hyper-heuristic framework from the literature, it is also compared to AgMOPSO, a state-of-the-art MOPSO. In (Castro Jr. and Pozo 2014, 2015), the authors compare the performance of the H-MOPSO framework to each combination of the archiving and leader methods used separately.

H-MOPSO obtained the best results in most cases. Therefore, this comparison is not made here.

4.1 Experimental setup

The basic parameters for H-MOPSO used in this study are the same ones proposed for SMPSO (Nebro et al. 2009) since H-MOPSO is based on it. The C_1 and C_2 parameters vary randomly in [1.5, 2.5]. The inertia ω is 0.1, and the polynomial mutation (turbulence) is applied to 15% of the population. For MOEA/D-FRRMAB we followed the same parameters used in (Li et al. 2014a), where the neighborhood size T was set to $0.1 \times N$ and the maximum number of solutions replaced by each child solution n_r was set to $0.01 \times N$, where N is the number of subproblems. The probability of selecting a parent from the neighborhood δ is set to 0.9 and the control parameters for the differential evolution (DE) are $CR = 1$ and $f = 0.5$. The parameters for the polynomial mutation are distribution index $\eta = 20$ and mutation rate $p_m = 1/n$ where n is the number of decision variables. For AgMOPSO we used the same parameter proposed in the paper (Zhu et al. 2017). The MOPSO phase parameters are inertia $w = [0.1, 0.5]$, and $F_2 = 0.5$, and F_1 is dynamically computed. The parameters for the Immune Search phase of AgMOPSO are: crossover and mutation probabilities as $p_c = 0.9$ and $p_m = 1/n$, the distribution index are $\eta_c = 20$ and $\eta_m = 20$. The neighborhood size is $T = 20$.

The parameters of the heuristic selection methods were empirically configured.[1] For the Roulette heuristic selection, the initial probabilities are the same for all the low-level heuristics. The minimum probability of each heuristic was set to 0.1% to prevent them from being removed from the search. Moreover, the increment (or decrement) in the probability of the selected heuristic is set to one-twentieth of the initial probabilities.

For the ACF heuristic selection, the parameters used were: Scaling Factor (SF) of 1.0, and f_1 and f_2 using the accumulated values. For the FRRMAB heuristic selection, the parameters used were: scaling factor (C), set to 0.5, decaying factor for the reward (D), set to 1, and sliding window size (W), set to 50. These parameters were empirically chosen. The move acceptance criterion utilized in all variants was Improving and Equal (IE). The nine low-level heuristics, composed of a leader selection and an archiving method, and available to be selected by each heuristic selection strategy are summarized in Table 1. A representative set of benchmark problems from the DTLZ (Deb et al. 2002) and WFG (Huband et al. 2006) benchmark families were used. They are presented in Table 2, along with their characteristics and number of decision variables. The numbers of objectives (m) used were: 2, 3, 5, 8 and 10. The population and repository maximum size for the H-MOPSO variants, as well as the population for MOEA/D-FRRMAB, were set according to the number of objectives as 91, 91, 210, 156 and 275, respectively (Li et al. 2015). The numbers of iterations were set according to the problem and the objective number. Table 3 summarizes this data (Li et al. 2015). In all experiments, the algorithms execute with the same number

[1] The results of different parameter configurations are available at: `http://www.inf.ufpr.br/gmfritsche/hmopsosupplementary.pdf`.

Table 2 Characteristics and number of decision variables of the problems used

Problem	Characteristics	Decision variables number
DTLZ1	Linear, multi-modal	$m + k - 1, k = 5$
DTLZ2	Concave	$m + k - 1, k = 10$
DTLZ3	Concave, multi-modal	$m + k - 1, k = 10$
DTLZ4	Concave, biased	$m + k - 1, k = 10$
WFG6	Concave, non-separable	$k + l, k = 2 \times (m - 1), l = 20$
WFG7	Concave, biased	$k + l, k = 2 \times (m - 1), l = 20$

Table 3 Number of iterations per problem instance

Problem	Objective number				
	2	3	5	8	10
DTLZ1	400	400	600	750	1000
DTLZ2	250	250	350	500	750
DTLZ3	1000	1000	1000	1000	1500
DTLZ4	600	600	1000	1250	2000
WFG6	400	400	750	1500	2000
WFG7	400	400	750	1500	2000

of iterations (Table 3) and population size. To assess the quality of the Pareto fronts generated by each algorithm, this study uses the IGD and hypervolume as the quality indicators, presented in Sects. 2.5.1 and 2.5.2, respectively. The results measured by the quality indicators in 20 independent runs of the algorithms were submitted to the Kruskal-Wallis test (Kruskal and Wallis 1952) at 5% significance level. Moreover, to summarize the results, the averages of the 20 runs of each algorithm were submitted to the Friedman test (Friedman 1937), also at 5% significance level. This specific number of independent runs was selected because relevant studies in many-objective literature adopt it, such as Deb and Jain (2014); Li et al. (2015), where the NSGA-III and MOEA/DD algorithms are proposed, respectively.

4.2 Heuristic selection methods

The average results considering the hypervolume of each algorithm for each problem and objective number are presented in Table 4. The best results are highlighted in bold font, and if there are statistically significant differences according to the Kruskal-Wallis test, the best algorithm is highlighted in italized. Since non-parametric statistical tests, like the Kruskal-Wallis test, are conservative, we drew conclusions based on the average indicator values when significant differences were not found.

Regarding these results, Roulette had the best average performance in nineteen out of thirty problem instances, two of them with a statistical difference to FRRMAB, ACF, and RANDOM. The FRRMAB achieved the best average performance in six problem instances. In five instances the ACF had the best average. The RANDOM

Table 4 HV: Mean and standard deviation for Random, ACF, FRRMAB, and Roulette

Obj.	Problem	FRRMAB	ACF	ROULETTE	RANDOM
2	DTLZ1	8.69E−1(1.39E−2)	8.54E−1(5.2E−2)	**8.74E−1(4.25E−5)**	8.64E−1(2.07E−2)
	DTLZ2	**3.21E0(3.18E−4)**	3.21E0(2.9E−4)	3.21E0(1.66E−4)	3.21E0(6.97E−4)
	DTLZ3	3.2E0(5.17E−2)	**3.21E0(2.15E−4)**	3.19E0(9.44E−2)	3.13E0(3.44E−1)
	DTLZ4	3.21E0(5.37E−4)	3.21E0(8.3E−4)	**3.21E0(5.92E−5)**	3.21E0(9.42E−4)
	WFG6	8.33E0(3.97E−1)	8.5E0(3.85E−3)	**8.5E0(1.62E−3)**	8.14E0(5.08E−1)
	WFG7	8.38E0(2.08E−1)	**8.43E0(4.4E−2)**	8.41E0(1.36E−1)	8.23E0(2.03E−1)
3	DTLZ1	9.65E−1(8.26E−3)	9.69E−1(3.65E−3)	**9.7E−1(9.39E−4)**	9.65E−1(6.38E−3)
	DTLZ2	7.35E0(5.05E−2)	7.36E0(9.29E−3)	***7.37E0(4.85E−3)***	7.37E0(7.2E−3)
	DTLZ3	7.21E0(3.98E−1)	**7.31E0(2.95E−1)**	7.22E0(4.06E−1)	7.21E0(3.96E−1)
	DTLZ4	7.39E0(5.41E−3)	7.39E0(7.05E−3)	**7.39E0(5.29E−3)**	7.39E0(6.58E−3)
	WFG6	7.01E1(8.87E−1)	7E1(1.02E0)	**7.06E1(1.18E0)**	6.9E1(2.11E0)
	WFG7	6.16E1(1.75E0)	6.22E1(1.11E0)	**6.23E1(2.14E0)**	6.05E1(1.83E0)
5	DTLZ1	9.96E−1(1.2E−3)	9.95E−1(3.2E−3)	**9.97E−1(5E−4)**	9.97E−1(8.2E−4)
	DTLZ2	3.14E1(8.31E−2)	3.15E1(4.37E−2)	***3.15E1(2.82E−2)***	3.15E1(3.42E−2)
	DTLZ3	3.15E1(9.14E−2)	**3.15E1(7.36E−2)**	3.15E1(6.43E−2)	3.15E1(4.89E−2)
	DTLZ4	3.16E1(3.72E−2)	3.16E1(2.22E−2)	**3.16E1(2.04E−2)**	3.16E1(3.08E−2)
	WFG6	7.92E3(9.54E1)	8.04E3(6.74E1)	**8.12E3(4.72E1)**	7.92E3(9.08E1)
	WFG7	6.65E3(1.54E2)	6.74E3(1.25E2)	**6.78E3(8.99E1)**	6.52E3(1.75E2)
8	DTLZ1	9.98E−1(1.33E−3)	9.99E−1(9.85E−4)	**9.99E−1(6.52E−4)**	9.98E−1(1.27E−3)
	DTLZ2	2.47E2(3.23E0)	2.49E2(1.77E0)	**2.49E2(1.79E0)**	2.47E2(2.98E0)
	DTLZ3	**2.46E2(3.75E0)**	2.35E2(2.58E1)	2.45E2(7.16E0)	2.4E2(1.44E1)
	DTLZ4	2.55E2(1.74E−1)	2.56E2(1.83E−1)	**2.56E2(1.15E−1)**	2.56E2(1.34E−1)
	WFG6	**2.64E7(7.26E5)**	2.52E7(1.99E6)	2.64E7(7.6E5)	2.57E7(1.19E6)
	WFG7	**1.77E7(1.8E6)**	1.75E7(1.39E6)	1.7E7(1.04E6)	1.77E7(1.54E6)
10	DTLZ1	9.99E−1(6.45E−4)	9.99E−1(3.94E−4)	**1E0(3.4E−4)**	9.99E−1(5.28E−4)
	DTLZ2	1E3(7.8E0)	**1E3(5.95E0)**	1E3(8.96E0)	1E3(7.11E0)
	DTLZ3	9.98E2(1.03E1)	9.94E2(2.05E1)	**1.01E3(9.78E0)**	9.94E2(3.54E1)
	DTLZ4	1.02E3(3.22E−1)	1.02E3(6.73E−2)	**1.02E3(1.49E−1)**	1.02E3(1.01E−1)
	WFG6	**1.09E10(5.51E8)**	1.07E10(6.5E8)	1.08E10(4.87E8)	1.07E10(4.74E8)
	WFG7	**7.31E9(3.47E8)**	7.31E9(5.25E8)	7.01E9(4.57E8)	7.01E9(4.94E8)

heuristic selection was unable to get the best average result in any problem instance. These results indicate that the use of a heuristic selection method is better than random selection, mainly when the proposed ROULETTE selection is applied.

To ease the visualization of the data presented in Table 4, and to allow general conclusions, we considered the performance per algorithm, ignoring particular problems and objective numbers. To do so, we calculated the average of the 20 independent runs of each algorithm for all problems and objective numbers, and used these averages as samples in the Friedman test; the results are presented in Fig. 1. In this figure, each vertical line represents the average ranking of an algorithm (the smaller, the better)

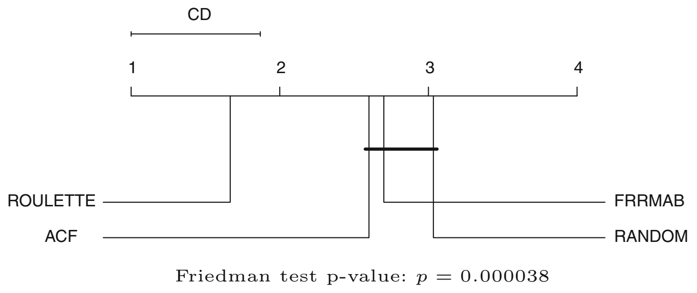


Fig. 1 Critical difference plot for FRRMAB, ACF, Roulette, and Random, HV indicator

and the bold horizontal line represents the statistically significant equivalence among the algorithms.

The best average ranking was achieved by ROULETTE, with a statistical difference to all others according to the Friedman test (p-value: 0.000038). The ACF and FRRMAB achieved the second and third best average ranking, respectively. Thus, the RANDOM selection had the worst average ranking. These results support the conclusion that employing rational heuristic selection method is better than just selecting the low-level heuristics randomly. The results also highlight the good performance achieved by the Roulette approach.

To produce a more in-depth investigation of the results, Fig. 2 presents a Boxplot for each problem instance. From the figure, it is possible to observe that the result varies according to the problem instance. For example, in WFG7 with eight objectives, all methods are very similar, including random. In other instances, such WFG7 with two objectives all three heuristic selection methods are better than a random selection, while being very similar among themselves. In some problem instances, the Roulette selection presented results that are better than the other methods, e.g., DTLZ1, DTLZ2, DTLZ4, and WFG6, with five objectives. But the other methods are very competitive. For example, the FRRMAB presents the best mean in WFG6 with 8 objectives, and the ACF had the best mean in DTLZ3 with 5 objectives.

As done for hypervolume, we also present the results per algorithm for the IGD in Table 5. For this indicator, the results were similar among the heuristic selection methods. For instance, the ROULETTE achieved the best average result in 9 problem instances, the ACF in 7, FRRMAB in 6, even the random selection was able to achieve the best average in 8 problem instances. As similar results were presented by the heuristic selection methods, it is difficult to point a best one. This is supported when the critical difference plot (Fig. 3) is evaluated. All four methods were found statistically equivalent, with a p-value of 0.81252 (much higher than the 0.05 threshold).

We also evaluated the Boxplots for IGD in Fig. 4. The Boxplot allows visualizing that the methods are not always all equivalent. For instance, in DTLZ1 with ten objectives the ROULETTE had best median with results similar to ACF, and better than random and FRRMAB. In DTLZ1 with five objectives the ROULETTE had the best median, but with some results similar to the random selection. The same goes for the other algorithms, for instance, in WFG7 with ten objectives the FRRMAB achieved the best median, but ACF also achieved good results. In DTLZ3 with two objectives

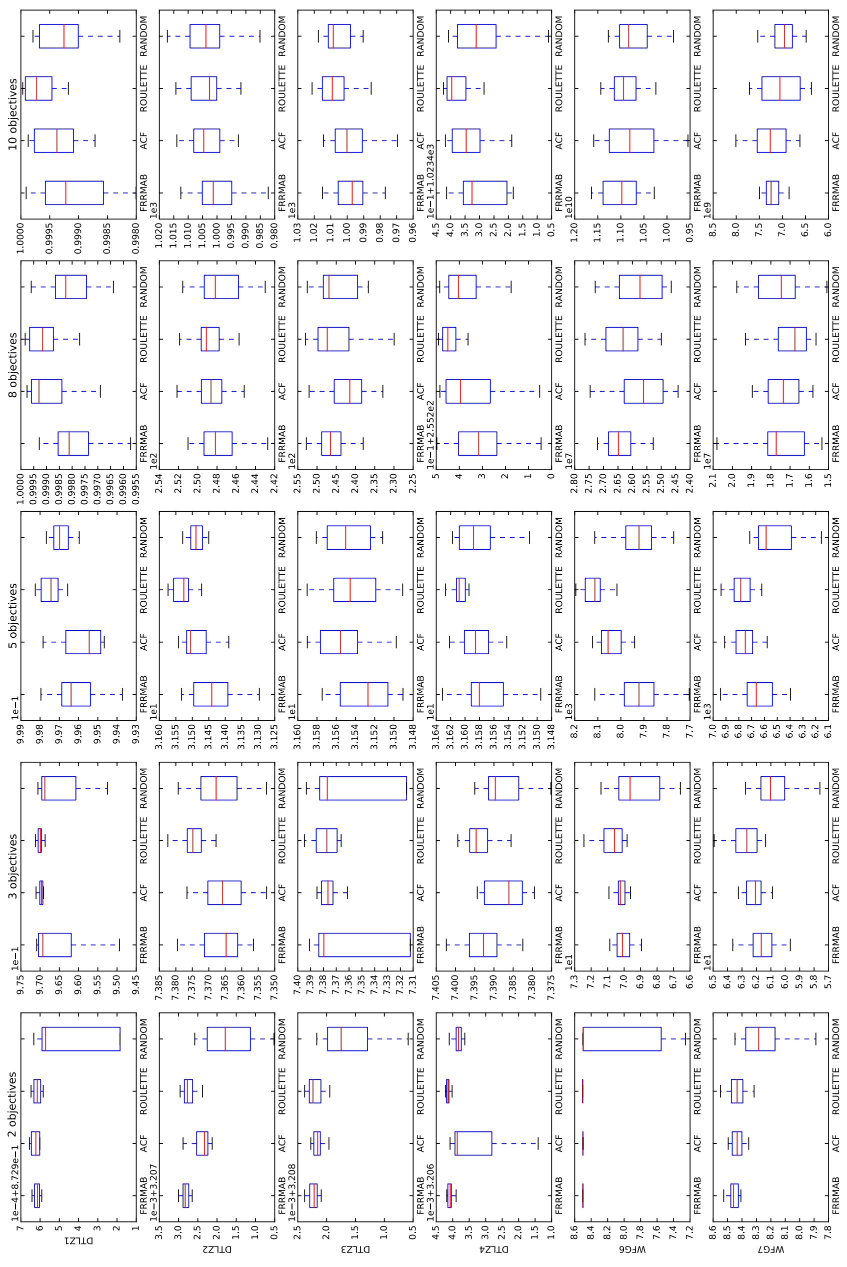


Fig. 2 Boxplot of FRRMAB, ACF, Roulette, and Random, HV indicator

Table 5 IGD: Mean and standard deviation for Random, ACF, FRRMAB, and Roulette

Obj.	Problem	FRRMAB	ACF	ROULETTE	RANDOM
2	DTLZ1	9.17E−5(1.2E−4)	3.16E−4(8.43E−4)	**5.4E−5(9.51E−7)**	1.43E−4(1.93E−4)
	DTLZ2	8.27E−5(7.74E−6)	**8.22E−5(4.82E−6)**	8.25E−5(3.14E−6)	1.02E−4(1.92E−5)
	DTLZ3	9.67E−5(1.35E−4)	**6.19E−5(3.98E−6)**	1.4E−4(3.29E−4)	3.43E−4(1.16E−3)
	DTLZ4	7E−5(1.02E−5)	7.89E−5(1.69E−5)	**6.71E−5(1.3E−6)**	7.41E−5(1.89E−5)
	WFG6	4.07E−4(7.14E−4)	**1.13E−4(4.03E−6)**	1.14E−4(2.17E−6)	7.31E−4(8.95E−4)
	WFG7	5.52E−4(4.13E−4)	**4.32E−4(6.55E−5)**	4.83E−4(2.65E−4)	7.44E−4(3.92E−4)
3	DTLZ1	6.51E−4(1.67E−4)	5.98E−4(8.4E−5)	**5.92E−4(3.13E−5)**	6.73E−4(2.29E−4)
	DTLZ2	8.89E−4(9.02E−5)	8.77E−4(5.58E−5)	**8.66E−4(3.97E−5)**	8.85E−4(7.34E−5)
	DTLZ3	1.04E−3(7.68E−4)	**9.14E−4(5.84E−4)**	1.1E−3(8.25E−4)	1.1E−3(8.34E−4)
	DTLZ4	**9.25E−4(8.59E−5)**	1.03E−3(1.97E−4)	1.1E−3(2.5E−4)	1.01E−3(1.26E−4)
	WFG6	**1.33E−3(1.24E−4)**	1.35E−3(2.01E−4)	1.35E−3(1.71E−4)	1.52E−3(3.24E−4)
	WFG7	3.59E−3(5.14E−4)	**3.38E−3(1.74E−4)**	3.51E−3(6E−4)	3.84E−3(5.95E−4)
5	DTLZ1	1.15E−3(9.24E−5)	1.2E−3(2.08E−4)	**1.06E−3(6.56E−5)**	1.1E−3(3.85E−5)
	DTLZ2	1.27E−3(5.29E−5)	1.24E−3(5.17E−5)	**1.2E−3(4.47E−5)**	1.22E−3(5.69E−5)
	DTLZ3	1.04E−3(5.66E−5)	1.04E−3(7.42E−5)	1.04E−3(8.34E−5)	**1.01E−3(6.6E−5)**
	DTLZ4	**1.3E−3(4.8E−5)**	1.31E−3(9.63E−5)	1.44E−3(1.52E−4)	1.3E−3(7.18E−5)
	WFG6	1.26E−3(4.55E−5)	1.22E−3(4.97E−5)	**1.21E−3(3.58E−5)**	1.26E−3(3.48E−5)
	WFG7	2.07E−3(6.13E−5)	**2.02E−3(5.95E−5)**	2.02E−3(9.1E−5)	2.11E−3(7.07E−5)
8	DTLZ1	1.59E−3(9.87E−5)	1.45E−3(6.72E−5)	**1.45E−3(7.29E−5)**	1.49E−3(9.35E−5)
	DTLZ2	1.14E−3(8.46E−5)	1.12E−3(1.08E−4)	1.17E−3(1.04E−4)	**1.1E−3(6.7E−5)**
	DTLZ3	9.3E−4(2.05E−4)	1.01E−3(3.71E−4)	9.9E−4(2.56E−4)	**8.54E−4(1.9E−4)**
	DTLZ4	1.75E−3(2.6E−4)	1.67E−3(1.2E−4)	1.8E−3(2.02E−4)	**1.65E−3(1.9E−4)**
	WFG6	**2.6E−3(1.45E−4)**	2.7E−3(2.25E−4)	2.69E−3(1.16E−4)	2.67E−3(1.37E−4)
	WFG7	2.48E−3(1.5E−4)	2.57E−3(2.61E−4)	2.71E−3(3.01E−4)	**2.47E−3(2.48E−4)**
10	DTLZ1	1.22E−3(6.12E−5)	1.16E−3(4.83E−5)	**1.14E−3(4.97E−5)**	1.19E−3(4.74E−5)
	DTLZ2	8.18E−4(9.52E−5)	7.56E−4(6.82E−5)	8.25E−4(1E−4)	**7.51E−4(5.15E−5)**
	DTLZ3	2.92E−4(6.86E−5)	2.66E−4(6.71E−5)	3.04E−4(4.72E−5)	**2.1E−4(6.9E−5)**
	DTLZ4	1.14E−3(1.08E−4)	1.14E−3(1.07E−4)	1.19E−3(9.65E−5)	**1.1E−3(8.56E−5)**
	WFG6	**2.02E−3(1.37E−4)**	2.03E−3(1.17E−4)	2.11E−3(1.17E−4)	2.06E−3(1.26E−4)
	WFG7	**1.97E−3(1.4E−4)**	2.12E−3(2.27E−4)	2.25E−3(1.88E−4)	2.11E−3(2.08E−4)

the ACF had the best median, while random performed poorly. In some problems (*e. g.* WFG7 with eight objectives) the random selection had the best median, but the results of the other selection methods were similar. Finally, regarding IGD, it is possible to conclude that the three heuristic selection methods achieved similar results, usually better than, or equivalent to, the random approach.

The results presented indicate that the Roulette selection method is the best for the H-MOPSO algorithm. The FRRMAB and ACF methods achieved similar results in general, and showed superior results than those obtained by Random, although no statistically significant differences were found. This result confirms our hypoth-

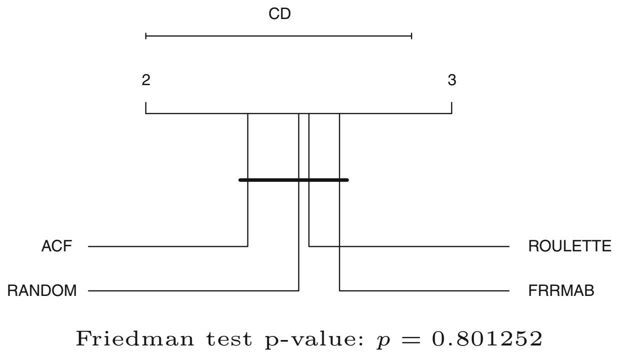


Fig. 3 Critical difference plot for FRRMAB, ACF, Roulette, and Random, IGD indicator

esis that the heuristic selection method has an impact on the performance of the H-MOPSO algorithm; however, this impact was weaker than anticipated. Moreover, we expected that the state-of-the-art heuristic selection methods would outperform the simple roulette wheel. Nevertheless, this was not the case.

According to the IGD indicator, the best average values are distributed among the algorithms. However, if one consider the number of instances (problem/objective number) where each algorithm achieved the best average results, the Roulette performed better than the others.

Following with our experimental study, we present a comparison of the selections made by each hyper-heuristic method: ACF, FRRMAB, Roulette, and Random. The methodology for this comparison is similar to the one from (Li et al. 2014a). Given a problem instance, we divided the search process into consecutive phases. For each phase, we calculate, and plot, the average usage number of each low-level heuristic. The low-level heuristics are presented in Table 1. For this section, we selected the WFG7 with 2 and 5 objectives (problem instances where the heuristic selection methods presented better results than a random selection). Figure 5 displays the average usage number for 10 phases of 20 independent runs, for the WFG7 with 2 objectives. The problem was executed during 400 iterations, so each phase has 40 iterations. It is important to remind that the heuristic selection is only applied after the repository is full (usually from the second epoch onwards for this problem instance). When the random selection is used the low-level heuristics are selected around 4.44 times each epoch (varying from 2.57 to 6.32), which is precisely the behavior expected from a random selection (40 iterations divided by nine low-level heuristics).

From the Fig. 5, it is possible to observe that the heuristic selection methods have preferences for some low-level heuristics, unlike the random selection. The FRRMAB presented a more elitist behavior, selecting H1 (CD/CD) or H4 (MGA/CD) heuristics more frequently than the others. The H1 was more selected up to the eighth epoch, then H4 was selected more times. This ability to change during the search is a characteristic of FRRMAB, due to its sliding window. The ROULETTE and ACF presented similar behavior. Both selected more often a set of low-level heuristics (H1 to H6) and used few times the others (H7, H8, and H9). The first set was selected around 4 times more than the latter. The H7, H8 and H9 heuristics, share the characteristic of having Ideal

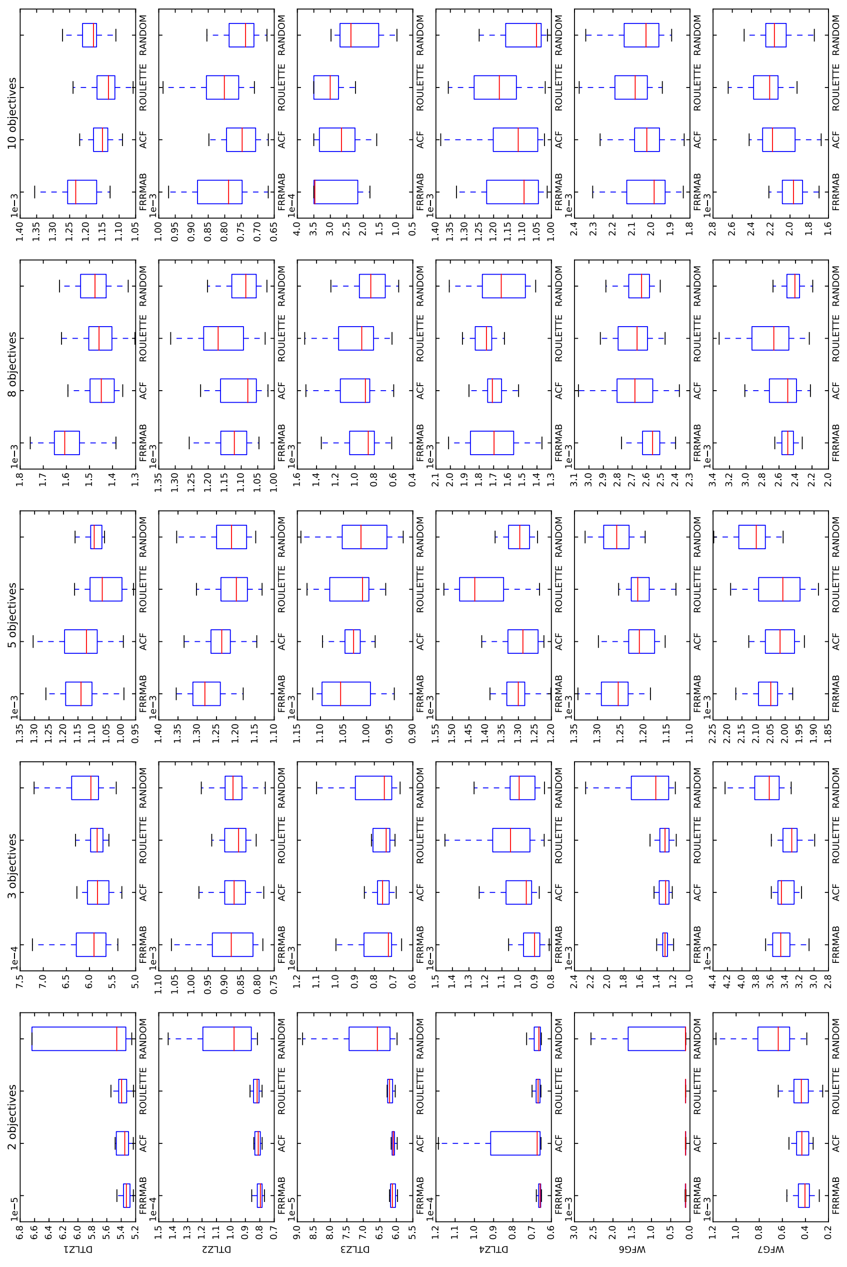


Fig. 4 Boxplot of FRRMAB, ACF, Roulette, and Random, IGD indicator

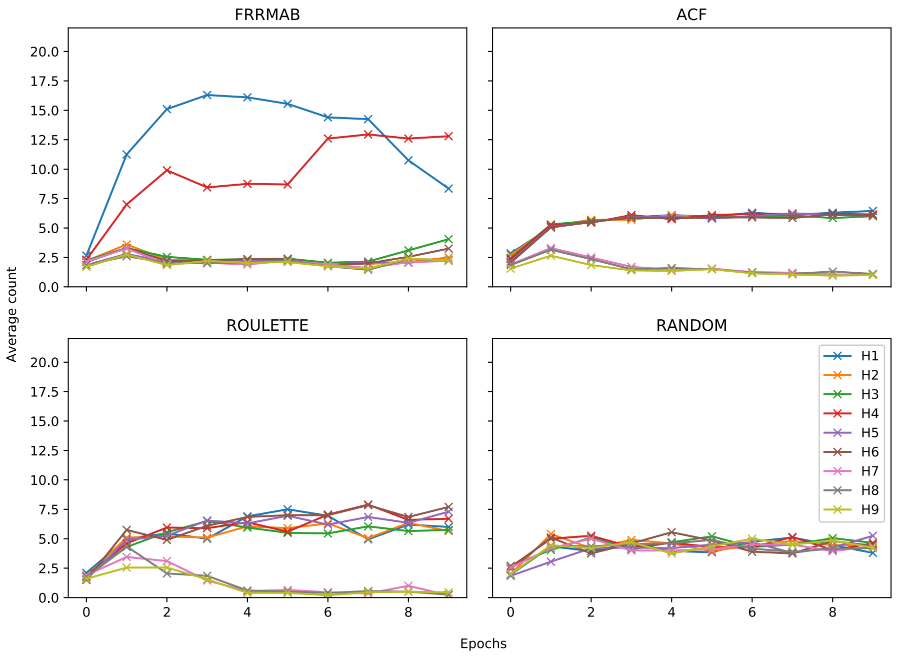


Fig. 5 Average application count of each low-level heuristic for ten epochs: WFG7 with 2 objectives

as the archiver. The difference between the ROULETTE and ACF is that the first select randomly based on probabilities, while the second selects the best, given a score. This difference can be noticed in Fig. 5, where the roulette presents a random selection strategy.

Figure 6 displays the average usage number for 10 phases of 20 independent runs, for the WFG7 with 5 objectives. The problem was executed during 750 iterations, so each phase has 75 iterations. These results support the conclusions from the previous figure. The random approach selects every low-level heuristic a similar amount of times. The FRRMAB preferences changed during the search. The ROULETTE again divided the heuristics into two groups. Initially, all heuristics have similar probabilities, However, during the search the selection of H4, H5, and H6 increases (all three using MGA archiver). The ACF started using H1 more often, then switched to the heuristics that employ the MGA archiver, mainly H6.

The experiments presented in this section allow observing that the heuristic selection methods have different behaviors. Further, given a specific problem instance, one heuristic selection method may be better suited than others. The Roulette selection method has the characteristic of increasing the probabilities of the best performing heuristics during the search. On the other hand, poorly performing heuristics kept a small probability to avoid being removed from the search. This method identifies the difference between the best and the worst performing heuristic but also gives a similar amount of probability to the ones with similar performance. Further, when a set of heuristics has a high probability of being selected, the swap often occurs among the most used heuristics, given the stochastic characteristic of the method.

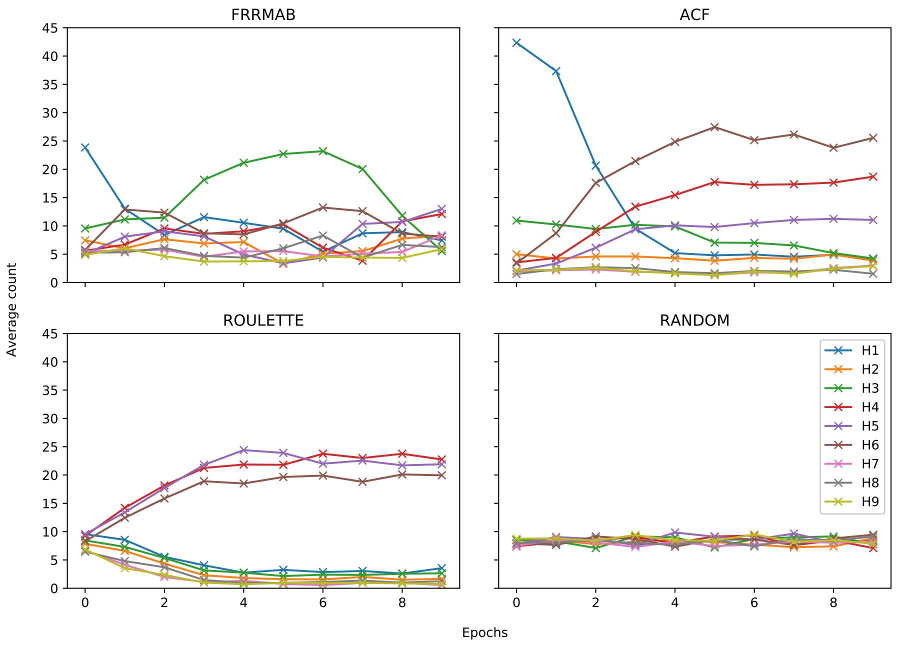


Fig. 6 Average count of the application of each low-level heuristic for ten epochs: WFG7 with 5 objectives

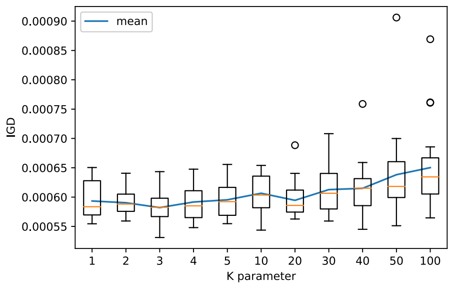
4.3 H-MOPSO-Roulette

In this section, an additional study is presented to understand the influence of the interval (K) used to replace the low-level heuristic. For the sensitivity analysis of the parameter K, this study uses the DTLZ2 problem for three objectives because of the poor results found by H-MOPSO-Roulette on this problem. The results obtained with different K values were evaluated using both the IGD and the Hypervolume indicators. Other approaches to deciding when to change the low-level heuristic could be used (Helbig and Engelbrecht 2014), although this research will be addressed in future works.

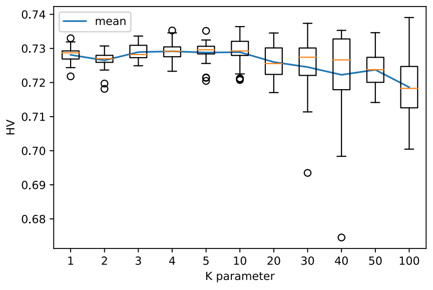
The current study was divided into three phases. First, the parameter K is used only for selecting the archiving method, keeping the leader selection fixed. Then, a similar study is made only for the leader selection method, keeping the archiving method fixed. Finally, the study investigated the parameter K for both approaches simultaneously. At the end of the section, the best results obtained in each of the three studies are compared to identify the best setting.

4.3.1 The archiving selection interval

In this study, K was configured for selecting only the archiving method. Hence, the leader selection method is fixed to use only the SMPSO default (Crowding Distance). The values used in this study were K={1, 2, 3, 4, 5, 10, 20, 30, 40, 50, 100}. For



(a) IGD



(b) HV

Fig. 7 Configuration of parameter K for Archiving

example, if $K = 10$ the selected low-level heuristic (archive method) will be executed 10 times before being replaced.

Figure 7 presents the IGD and HV results of this comparison, through Boxplots for each K value. Among the compared values, the best IGD results were obtained with $K = 3$, while $K = 100$ achieved the worst results in general. For the HV, the best results were obtained with $K = \{3, 4, 5, 10\}$. The value $K = 3$ was selected to be used in the next experiments for the archiving selection interval, because it presented the best IGD results, and is one of the best for HV. In summary, the best results were found when K is smaller or equal to 20. It means that better results were found replacing the low-level heuristic more often (at least once every 20 iterations). This is a significant finding because it means that the rational change of low-level heuristic is beneficial to the algorithm.

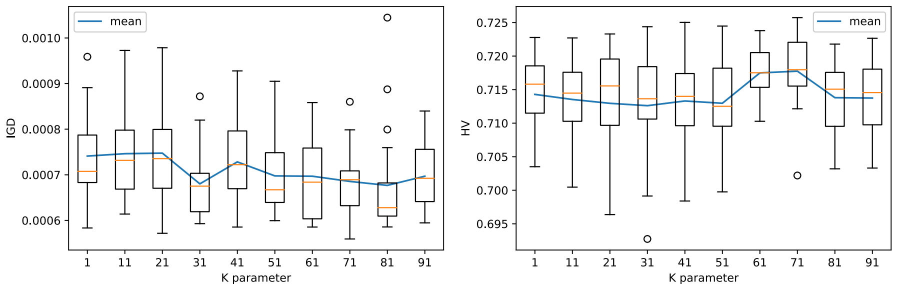
4.3.2 The leader selection interval

A similar study was made, this time for the leader selection method. In this study, the archiving method was fixed to use only the SMPSO default: Crowding Distance. Unlike the archiving method, a different leader selection method can be attributed to each particle, which means that groups of different sizes can use different leader selection methods, from 1 to the whole population. Hence, the K parameter here means the number of particles (group size) in which the same leader selection method was used. In this study the values investigated are K={1, 11, 21, 31, 41, 51, 61, 71, 81, 91}.

The IGD results of this study are presented in Fig. 8 for each K value. The best performing K values for IGD were 81, and for HV it was 71. The value of $K = 71$ was selected to be used in the next experiments for the leader selection interval because it had the best HV results, and was one of the best for IGD.

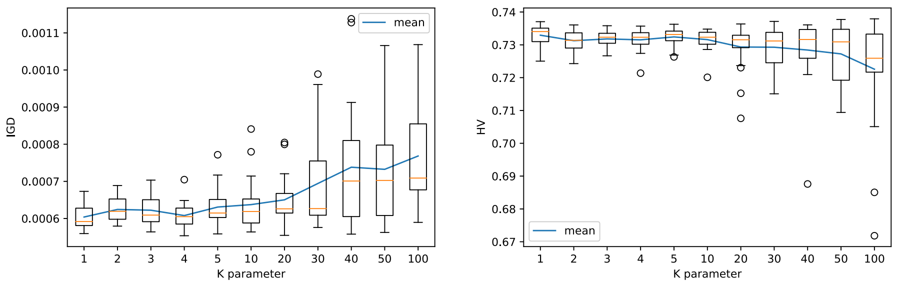
4.3.3 Leader and archiving selection interval

A third experimental study was made, where both methods (archive and leader selection) were replaced simultaneously after K iterations. The values evaluated were



(a) IGD **(b)** HV

Fig. 8 Configuration of parameter K for Leader Selection



(a) IGD **(b)** HV

Fig. 9 Configuration of parameter K for Archive and Leader Selection

K={1, 2, 3, 4, 5, 10, 20, 30, 40, 50, 100}, as in the first study. The IGD and HV results of this combined study can be seen in Fig. 9. It is possible to observe that, in general, smaller values are better than larger ones, and the best value is $K = 1$, for both IGD and HV, as in the original H-MOPSO. In summary, the best results were found when K is smaller or equal to 5. It means that better results were found replacing the low-level heuristic more often (at least once every 5 iterations). Since the best value for K was 1 for both IGD and HV, it is the value selected to be used in the next experiments for the leader and archiving selection interval.

4.3.4 Comparison between selecting leader, archiving or both simultaneously

The previous sections presented three different experimental studies. First, it was investigated the best value for K to select the archiving method and found $K = 3$ as the best value. Next, the studies examined how many particles in the population should use the same leader selection method; the best value obtained was $K = 71$. Finally, it was combined both methods to investigate which is the best interval for the pair leader selection / archiving; this study indicated that changing at each iteration ($K = 1$) is the best approach, as was originally done in the H-MOPSO algorithm.

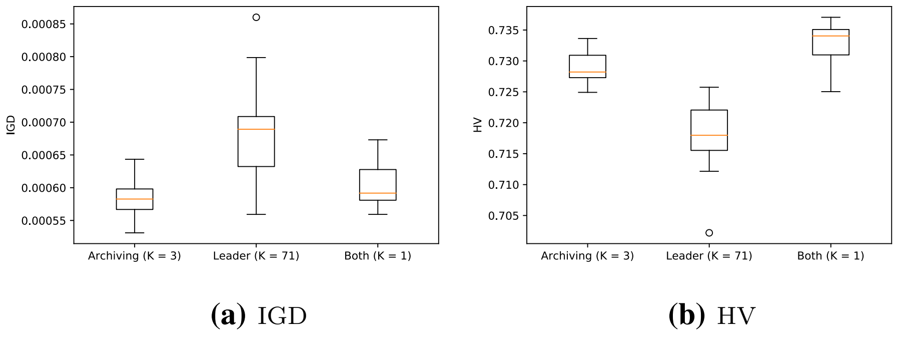


Fig. 10 Comparison between selecting leader, archiving or both simultaneously

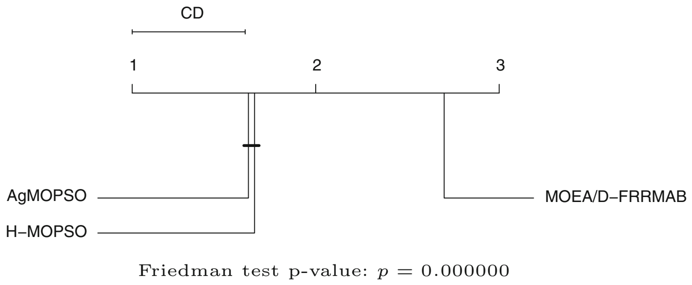


Fig. 11 Critical difference plot for H-MOPSO, AgMOPSO, and MOEA/D-FRRMAB, HV indicator

This section compares the best configurations obtained in each of the three sections, to determine the best configuration for H-MOPSO-Roulette. The results of this comparison are presented in Fig. 10.

For both IGD and HV indicators, selecting only the leader method achieved the worst results. The selection of both (leader and archiving methods) simultaneously achieved the best HV results, compared to selecting archiving only, while they presented similar results for IGD. Based on the results presented, it is possible to conclude that simultaneously selecting both methods is a better option than selecting only one.

4.4 Comparison between H-MOPSO and the state-of-the-art

In this section, our goal is to validate the performance of our framework against two state-of-the-art algorithms. To do so, we selected MOEA/D-FRRMAB (Li et al. 2014a), since similarly to H-MOPSO, it employs a hyper-heuristic as a component to improve its performance. We also compared to AgMOPSO (Zhu et al. 2017), which is a state-of-the-art MOPSO. In the experiments, the H-MOPSO uses the Roulette heuristic selection method. Moreover, the parameter K is set to 1. We took the parameter setting of MOEA/D-FRRMAB and AgMOPSO from (Li et al. 2014a; Zhu et al. 2017) respectively.

When evaluating the results obtained by the HV indicator presented in Table 6, H-MOPSO had the best value with a significant statistical difference in 10 of the

Table 6 HV: Mean (and standard deviation) for H-MOPSO and MOEA/D-FRRMAB and AgMOPSO

Obj.	problem	H-MOPSO	AgMOPSO	MOEA/D-FRRMAB
2	DTLZ1	8.74E−1(4.25E−5)	***8.74E−1(9.74E−6)***	5.35E−1(7.48E−2)
	DTLZ2	3.21E0(1.66E−4)	***3.21E0(5.97E−5)***	1.67E0(1.71E−6)
	DTLZ3	3.19E0(9.44E−2)	***3.21E0(1.52E−4)***	1.67E0(1.11E−3)
	DTLZ4	***3.21E0(5.92E−5)***	3.03E0(4.44E−1)	1.67E0(1.07E−6)
	WFG6	**8.5E0(1.62E−3)**	8.32E0(2.33E−1)	3.87E0(1.09E−1)
	WFG7	8.41E0(1.36E−1)	***8.68E0(9.36E−4)***	3.8E0(6.18E−2)
3	DTLZ1	***9.7E−1(9.39E−4)***	9.63E−1(4.29E−2)	9.54E−1(5.45E−2)
	DTLZ2	7.37E0(4.85E−3)	***7.41E0(1.1E−3)***	7.37E0(3.32E−3)
	DTLZ3	7.22E0(4.06E−1)	***7.41E0(1.61E−3)***	7E0(1.65E0)
	DTLZ4	7.39E0(5.29E−3)	***7.41E0(1.11E−3)***	7.35E0(3.66E−2)
	WFG6	7.06E1(1.18E0)	**7.07E1(1.41E0)**	6.74E1(6.96E−1)
	WFG7	6.23E1(2.14E0)	***7.62E1(8.86E−2)***	6.77E1(9.25E−1)
5	DTLZ1	9.97E−1(5E−4)	***9.99E−1(6.52E−5)***	6.55E−1(7.56E−2)
	DTLZ2	3.15E1(2.82E−2)	***3.16E1(8.54E−3)***	9.03E0(1.26E−3)
	DTLZ3	***3.15E1(6.43E−2)***	1.9E1(1.4E1)	9.53E0(9.56E−1)
	DTLZ4	3.16E1(2.04E−2)	***3.17E1(4.5E−3)***	9.58E0(1.24E0)
	WFG6	8.12E3(4.72E1)	***8.32E3(1.44E1)***	1.62E3(1.06E2)
	WFG7	6.78E3(8.99E1)	***9.03E3(3.56E1)***	1.67E3(2.39E1)
8	DTLZ1	***9.99E−1(6.52E−4)***	3.24E−1(4.28E−1)	6.06E−1(1.81E−2)
	DTLZ2	2.49E2(1.79E0)	***2.55E2(1.66E−1)***	5.4E1(5.58E−3)
	DTLZ3	***2.45E2(7.16E0)***	0E0(0E0)	5.41E1(4.2E−1)
	DTLZ4	***2.56E2(1.15E−1)***	2.55E2(1.45E−1)	7.29E1(1.4E1)
	WFG6	2.64E7(7.6E5)	***2.88E7(6.63E4)***	3.8E6(1.72E5)
	WFG7	1.7E7(1.04E6)	***3.18E7(2.34E5)***	4.08E6(7.11E5)
10	DTLZ1	***1E0(3.4E−4)***	0E0(0E0)	6.46E−1(5.07E−2)
	DTLZ2	1E3(8.96E0)	***1.02E3(6.42E−1)***	1.83E2(2.29E−2)
	DTLZ3	***1.01E3(9.78E0)***	0E0(0E0)	1.84E2(3.34E0)
	DTLZ4	***1.02E3(1.49E−1)***	1.02E3(2.4E0)	3.8E2(1.12E2)
	WFG6	1.08E10(4.87E8)	***1.17E10(2.5E7)***	1.28E9(7.15E7)
	WFG7	7.01E9(4.57E8)	***1.31E10(4.65E7)***	1.58E9(3.74E8)

instances. H-MOPSO was outperformed with a statistical difference by AgMOPSO on 19 instances, while MOEA/D-FRRMAB did not achieve the best results for any problem. In the general analysis, considering all problems and objective numbers, presented in Fig. 11, H-MOPSO-Roulette outperformed MOEA/D-FRRMAB, with a statistical difference according to the Friedman test. Besides, in the general analysis, the H-MOPSO results were statistically equivalent to a state-of-the-art MOPSO, the AgMOPSO.

By looking at the IGD results presented in Table 7, it is possible to observe that H-MOPSO-Roulette can be considered superior to MOEA/D-FRRMAB. The over-

Table 7 IGD: Mean (and standard deviation) for H-MOPSO and MOEA/D-FRRMAB and AgMOPSO

Obj.	problem	H-MOPSO	AgMOPSO	MOEA/D-FRRMAB
2	DTLZ1	1.15E−4(4.06E−6)	***4.78E−5(5.62E−7)***	1.03E−2(5.55E−4)
	DTLZ2	1.52E−4(6.25E−6)	***6.92E−5(7.85E−7)***	1.2E−2(7.34E−8)
	DTLZ3	2.83E−4(6.49E−4)	***5.4E−5(5.18E−7)***	1.09E−2(1.47E−6)
	DTLZ4	**1.29E−4(2.95E−6)**	1.77E−3(4.17E−3)	1.11E−2(1.51E−8)
	WFG6	**1.61E−4(1.97E−5)**	3.33E−4(3.11E−4)	1.71E−2(1.14E−3)
	WFG7	8.01E−4(3.98E−4)	***1.2E−4(2.15E−6)***	1.84E−2(7.72E−4)
3	DTLZ1	1.18E−3(7.35E−5)	***7.59E−4(1.31E−3)***	1.91E−3(2.61E−3)
	DTLZ2	1.6E−3(9.52E−5)	***6.54E−4(5.78E−6)***	1.35E−3(1.27E−5)
	DTLZ3	2.27E−3(1.73E−3)	***5.79E−4(6.74E−6)***	6.33E−3(2.22E−2)
	DTLZ4	1.89E−3(3.67E−4)	***6.35E−4(5.71E−6)***	3.43E−3(2.47E−3)
	WFG6	2.27E−3(2.47E−4)	***1.26E−3(4.07E−5)***	3.18E−3(2.45E−4)
	WFG7	3.25E−3(6.16E−4)	***1.48E−3(2.56E−5)***	3.14E−3(7.91E−5)
5	DTLZ1	2.14E−3(1.21E−4)	***9.15E−4(1.96E−5)***	7.55E−3(6.41E−4)
	DTLZ2	2.35E−3(1.37E−4)	***1.13E−3(2.33E−5)***	6.73E−3(4.24E−7)
	DTLZ3	***1.33E−3(1.86E−4)***	8.45E−3(9.99E−3)	5.48E−3(2.1E−4)
	DTLZ4	3.1E−3(5.53E−4)	***1.03E−3(2.11E−5)***	7.36E−3(4.85E−4)
	WFG6	2.66E−3(1.61E−4)	***1.15E−3(1.8E−5)***	1.01E−2(1.53E−4)
	WFG7	2.67E−3(2.74E−4)	***1.8E−3(5.67E−5)***	8.29E−3(7.97E−5)
8	DTLZ1	***2.71E−3(2.2E−4)***	2.36E−2(2.42E−2)	5.4E−3(4.45E−5)
	DTLZ2	2.32E−3(3.71E−4)	***1.54E−3(4.99E−5)***	4.78E−3(4.45E−7)
	DTLZ3	***1.64E−3(4.22E−4)***	4.99E−3(7.55E−4)	4.8E−3(1.07E−7)
	DTLZ4	4.56E−3(4.59E−4)	***1.74E−3(8.56E−5)***	6.74E−3(4.46E−4)
	WFG6	5.19E−3(3.16E−4)	***2.66E−3(5.75E−5)***	9.02E−3(8.25E−5)
	WFG7	6.04E−3(1.15E−3)	***2.75E−3(7.79E−5)***	9.36E−3(3.61E−4)
10	DTLZ1	***2.17E−3(1.55E−4)***	5.04E−2(2.99E−2)	4.33E−3(1.27E−4)
	DTLZ2	2.06E−3(5.33E−4)	***1.17E−3(4.94E−5)***	4.14E−3(2.39E−7)
	DTLZ3	1.22E−3(3.99E−4)	***4.38E−4(8.76E−5)***	1.62E−3(3.33E−6)
	DTLZ4	2.8E−3(3.16E−4)	***1.79E−3(9.29E−5)***	4.7E−3(4.75E−4)
	WFG6	4.09E−3(4.49E−4)	***2.18E−3(3.99E−5)***	6.43E−3(3.45E−5)
	WFG7	6.69E−3(8.6E−4)	***1.94E−3(4.63E−5)***	9.63E−3(1.07E−3)

all results presented in Fig. 12 confirm this outcome, where H-MOPSO outperforms MOEA/D-FRRMAB with a statistical difference according to the Friedman test. The H-MOPSO performed better than AgMOPSO for DTLZ1, DTLZ3, and DTLZ4, mainly for many-objective problems. Both DTLZ1 and DTLZ3 are multi-modal, besides DTLZ4 and DTLZ3 are concave. Further, DTLZ1 is linear, and DTLZ3 is biased. A possible explanation for the poor results of H-MOPSO for few objectives, compared to AgMOPSO, is that H-MOPSO does not apply the heuristic selection until the repository is full. Then, the algorithm presents different behavior depending on the number of objectives. For problems with few objectives, the repository takes

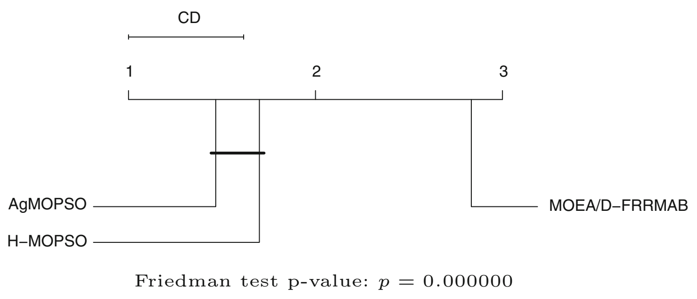


Fig. 12 Critical difference plot for H-MOPSO, AgMOPSO, and MOEA/D-FRRMAB, IGD indicator

longer to fill, so the algorithm executes several iterations as a standard SMPSO. The SMPSO (Nebro et al. 2009) did not achieve good results compared to AgMOPSO (Zhu et al. 2017). Thus, the H-MOPSO also did not achieve good results, for these problem instances. By increasing the number of objectives, the H-MOPSO start using the heuristic selection earlier in the search. Despite the quality deterioration of SMPSO, the online selection of archiving and leader selection methods improved the H-MOPSO results for many-objective optimization against the AgMOPSO.

5 Conclusion

Previous works have proposed the H-MOPSO framework, which uses a hyper-heuristic to select the leader and archiving methods for the MOPSO algorithm. Despite its simplicity, the hyper-heuristic used was able to outperform every combination of the leader and archiving methods used individually in most of the problems. Furthermore, H-MOPSO achieved competitive results when compared to MOEA/D-DRA (Zhang et al. 2009).

In this study, the following question was investigated: can a different hyper-heuristic selection method further improve the search ability of the MOPSO algorithm? To answer this question, this study implemented three alternative heuristic selection methods to use in the H-MOPSO framework. One of them was a simple random selection; the other two were methods that presented good results in the literature: ACF and FRRMAB.

Three experimental studies were conducted. The first of them compared the four heuristic selection methods. The second study was used to identify the best set of low-level heuristics to replace dynamically: only leader, only archiving, or both methods simultaneously; moreover, the studies identified the optimal number of iterations before changing the low-level heuristics inside the H-MOPSO. The third study used the best configuration of our framework to compare it to the state-of-the-art algorithms MOEA/D-FRRMAB (Li et al. 2014a) and AgMOPSO (Zhu et al. 2017).

From the results obtained in the first study, it is possible to observe that although few significant statistical differences were found, in general, Roulette outperformed the other heuristic selection methods. This means that the use of advanced heuristic selection methods was not able to achieve significant improvement in the quality of the generated solutions. The second study, conducted to assess the sensitivity of the algorithm to the interval used to replace the low-level heuristics, shows that in general lower values produce better results than higher ones, and the best value (with no significant differences) is changing at every iteration, like in the previous studies analyzed. In the third empirical study, it was observed that the H-MOPSO framework could achieve good results, and even outperform a hyper-heuristic framework in most of the problems investigated. Compared to a state-of-the-art MOPSO, the H-MOPSO achieved competitive results, mainly for many-objective problems where the hyper-heuristic starts working earlier. Despite our good results obtained so far, other directions of investigation can still be pursued in future studies, like selecting more advanced move acceptance criteria for the hyper-heuristic or investigating alternative methods to assess the quality of the solutions obtained by the low-level heuristics.

Acknowledgements The authors would like to thank the Academic Publishing Advisory Center (Centro de Assessoria de Publicação Acadêmica, CAPA - www.capa.ufpr.br) of the Federal University of Paraná for assistance with English language editing. Also, the authors would like to thank CNPq (National Council for Scientific and Technological Development) and CAPES (Coordination for the Improvement of Higher Education Personnel) for the financial support.

References

Bader, J., Deb, K., Zitzler, E.: Faster hypervolume-based search using Monte Carlo sampling. Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems, Lecture Notes in Economics and Mathematical Systems, vol. 634, pp. 313–326. Springer, Berlin (2010)

Bilgin, B., Özcan, E., Korkmaz, E.: An experimental study on hyper-heuristics and exam timetabling. Practice and Theory of Automated Timetabling VI, Lecture Notes in Computer Science, vol. 3867, pp. 394–412. Springer, Berlin (2007)

Blazewicz, J., Burke, E., Kendall, G., Mruczkiewicz, W., Oguz, C., Swiercz, A.: A hyper-heuristic approach to sequencing by hybridization of DNA sequences. Ann. Op. Res. **207**(1), 27–41 (2013). https://doi.org/10.1007/s10479-011-0927-y

Bringmann, K., Friedrich, T.: An efficient algorithm for computing hypervolume contributions. Evol. Comput. **18**(3), 383–402 (2010). https://doi.org/10.1162/EVCO_a_00012

Britto, A., Pozo, A.: Using archiving methods to control convergence and diversity for many-objective problems in particle swarm optimization. In: IEEE Congress on Evolutionary Computation, pp 1–8, (2012). https://doi.org/10.1109/CEC.2012.6256149

Brockhoff, D., Wagner, T., Trautmann, H.: On the properties of the R2 indicator. In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation, ACM, New York, NY, USA, GECCO ’12, pp 465–472, (2012). https://doi.org/10.1145/2330163.2330230

Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Qu, R.: Hyper-heuristics: a survey of the state of the art. J. Oper. Res. Soc. **64**(12), 1695–1724 (2013)

Castro, Jr. O.R., Pozo, A.: A MOPSO based on hyper-heuristic to optimize many-objective problems. In: IEEE Symposium on Swarm Intelligence (SIS), 2014, pp 1–8, (2014). https://doi.org/10.1109/SIS.2014.7011803

Castro Jr., O.R., Pozo, A.: Using hyper-heuristic to select leader and archiving methods for many-objective problems. In: Gaspar-Cunha, A., Henggeler Antunes, C., Coello, C.C. (eds.) Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science, vol. 9018, pp. 109–123. Springer International Publishing, Berlin (2015). https://doi.org/10.1007/978-3-319-15934-8_8

Castro, Jr. O.R., Britto, A., Pozo, A.: A comparison of methods for leader selection in many-objective problems. In: IEEE Congress on Evolutionary Computation, pp 1–8, (2012). https://doi.org/10.1109/CEC.2012.6256415

Coello, C.A.C., Cortés, N.C.: Solving multiobjective optimization problems using an artificial immune system. Genet. Progr. Evol. Mach. **6**(2), 163–190 (2005). https://doi.org/10.1007/s10710-005-6164-x

Coello, C.A.C., Lamont, G.B., Veldhuizen, D.A.V.: Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation). Springer-Verlag, Secaucus, NJ, USA (2006)

Cowling, P., Kendall, G., Soubeiga, E.: A Hyperheuristic Approach to Scheduling a Sales Summit. In: Burke, E., Erben, W. (eds.) Practice and Theory of Automated Timetabling III, Lecture Notes in Computer Science, vol. 2079, pp. 176–190. Springer, Berlin Heidelberg (2001)

Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. IEEE Trans. Evolut. Comput. **18**(4), 577–601 (2014). https://doi.org/10.1109/TEVC.2013.2281535

Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II. In: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature, Springer-Verlag, London, UK, PPSN VI, pp 849–858 (2000)

Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. IEEE Congr. Evolut. Comput. **1**, 825–830 (2002)

do Nascimento Ferreira, T., Kuk, J.N., Pozo, A., Vergilio, S.R.: Product selection based on upper confidence bound moea/d-dra for testing software product lines. In: 2016 IEEE Congress on Evolutionary Computation (CEC), pp 4135–4142, (2016) https://doi.org/10.1109/CEC.2016.7744315

Drake, J., Ozcan, E., Burke, E.: An improved choice function heuristic selection for cross domain heuristic search. In: Coello, C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) Parallel Problem Solving from Nature—PPSN XII, Lecture Notes in Computer Science, pp. 307–316. Springer, Berlin Heidelberg (2012). https://doi.org/10.1007/978-3-642-32964-7_31

Durillo, J.J., García-Nieto, J., Nebro, A.J., Coello, C.A.C., Luna, F., Alba, E.: Multi-objective particle swarm optimizers: An experimental comparison. In: Proceedings of the 5th International Conference on Evolutionary Multi-Criterion Optimization, Springer-Verlag, Berlin, Heidelberg, EMO '09, pp 495–509 (2009)

Eberhart, R.C., Shi, Y.: Particle swarm optimization: developments, applications and resources. In: Proceedings of the 2001 Congress on Evolutionary Computation, vol 1, pp 81–86, (2001) https://doi.org/10.1109/CEC.2001.934374

Elarbi, M., Bechikh, S., Gupta, A., Said, L.B., Ong, Y.s.: A New Decomposition-Based NSGA-II for Many-Objective Optimization. IEEE Transactions on Systems, Man, and Cybernetics - Systems pp 1–20, (2017). https://doi.org/10.1109/TSMC.2017.2654301

Ferreira, A.S., Gonçalves, R.A., Pozo, A.T.R.: A multi-armed bandit hyper-heuristic. In: 2015 Brazilian Conference on Intelligent Systems (BRACIS), pp 13–18, (2015). https://doi.org/10.1109/BRACIS.2015.31

Fialho, A., Da Costa, L., Schoenauer, M., Sebag, M.: Analyzing bandit-based adaptive operator selection mechanisms. Ann. Math. Artif. Intell. **60**(1–2), 25–64 (2010)

Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J. Am. Stat. Assoc. **32**(200), 675–701 (1937)

Gonçalves, R., Kuk, J., Almeida, C., Venske, S.: MOEA/D-HH: a hyper-heuristic for multi-objective problems. In: Gaspar-Cunha, A., Henggeler Antunes, C., Coello, C.C. (eds.) Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science, vol. 9018, pp. 94–108. Springer International Publishing, Berlin (2015a). https://doi.org/10.1007/978-3-319-15934-8_7

Gonçalves, R.A., Almeida, C.P., Kuk, J.N., Pozo, A.: Moea/d with adaptive operator selection for the environmental/economic dispatch problem. In: 2015 Latin America Congress on Computational Intelligence (LA-CCI), pp 1–6, (2015b) https://doi.org/10.1109/LA-CCI.2015.7435971

Gonçalves, R.A., Almeida, C.P., Pozo, A.: Upper confidence bound (UCB) algorithms for adaptive operator selection in moea/d. In: Gaspar-Cunha, A., Henggeler Antunes, C., Coello, C.C. (eds.) Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science, vol. 9018, pp. 411–425. Springer International Publishing, Berlin (2015c). https://doi.org/10.1007/978-3-319-15934-8_28

Gonçalves, R.A., Almeida, C.P., Pozo, A.: Upper Confidence Bound (UCB) Algorithms for Adaptive Operator Selection in MOEA/D, Springer International Publishing, Cham, pp 411–425. (2015). https://doi.org/10.1007/978-3-319-15934-8_28

Guizzo, G., Fritsche, G.M., Vergilio, S.R., Pozo, A.T.R.: A hyper-heuristic for the multi-objective integration and test order problem. In: Proceedings of the 2015 on Genetic and Evolutionary Computation Conference, ACM, New York, NY, USA, GECCO '15, pp 1343–1350, (2015). https://doi.org/10.1145/2739480.2754725

Hansen, M.P., Jaszkiewicz, A.: Evaluating the quality of approximations to the non-dominated set. Tech. Rep. IMM-REP-1998-7, Technical University of Denmark (1998)

Helbig, M., Engelbrecht, A.P.: Heterogeneous dynamic vector evaluated particle swarm optimisation for dynamic multi-objective optimisation. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp 3151–3159, (2014). https://doi.org/10.1109/CEC.2014.6900303

Hitomi, N., Selva, D.: A classification and comparison of credit assignment strategies in multiobjective adaptive operator selection. IEEE Trans. Evol. Comput. **21**(2), 294–314 (2017). https://doi.org/10.1109/TEVC.2016.2602348

Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Trans. Evol. Comput. **10**(5), 477–506 (2006). https://doi.org/10.1109/TEVC.2005.861417

Jiang, S., Ong, Y.S., Zhang, J., Feng, L.: Consistencies and contradictions of performance metrics in multiobjective optimization. IEEE Trans. Cybern. **44**(12), 2391–2404 (2014). https://doi.org/10.1109/TCYB.2014.2307319

Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp 1942–1948 (1995)

Krempser, E., Fialho, Á., Barbosa, H.: Adaptive operator selection at the hyper-level. Parallel Problem Solving from Nature-PPSN XII pp 378–387 (2012)

Kruskal, W.H., Wallis, W.A.: Use of ranks in one-criterion variance analysis. J. Am. Stat. Assoc. **47**(260), 583–621 (1952)

Laumanns, M., Zenklusen, R.: Stochastic convergence of random search methods to fixed size Pareto front approximations. Eur. J. Oper. Res. **213**(2), 414–421 (2011). https://doi.org/10.1016/j.ejor.2011.03.039

Li, K., Fialho, Á., Kwong, S.: Multi-objective differential evolution with adaptive control of parameters and operators. Learning and Intelligent Optimization, pp. 473–487. Springer, Berlin Heidelberg (2011)

Li, K., Fialho, A., Kwong, S., Zhang, Q.: Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. **18**(1), 114–130 (2014a). https://doi.org/10.1109/tevc.2013.2239648

Li, K., Deb, K., Zhang, Q., Kwong, S.: An evolutionary many-objective optimization algorithm based on dominance and decomposition. IEEE Trans. Evol. Comput. **19**(5), 694–716 (2015). https://doi.org/10.1109/TEVC.2014.2373386

Li, M., Yang, S., Liu, X.: Diversity comparison of pareto front approximations in many-objective optimization. IEEE Trans. Cybern. **44**(12), 2568–2584 (2014b). https://doi.org/10.1109/TCYB.2014.2310651

Luke, S.: Essentials of Metaheuristics, 2nd edn. Lulu, available for free at (2013). http://cs.gmu.edu/~sean/book/metaheuristics/

Maashi, M., Özcan, E., Kendall, G.: A multi-objective hyper-heuristic based on choice function. Expert Syst. Appl. **41**(9), 4475–4493 (2014)

Maturana, J., Fialho, Á., Saubion, F., Schoenauer, M., Lardeux, F., Sebag, M.: Adaptive operator selection and management in evolutionary algorithms. Autonomous Search, pp. 161–189. Springer, Berlin, Heidelberg (2012)

Mostaghim, S., Teich, J.: Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). In: Proceedings of the 2003 IEEE Swarm Intelligence Symposium., pp 26–33 (2003)

Nebro, A.J., Durillo, J.J., Garcia-Nieto, J., Coello, C.A.C., Luna, F., Alba, E.: SMPSO: A new PSO-based metaheuristic for multi-objective optimization. In: Computational intelligence in multi-criteria decision-making., IEEE, pp 66–73 (2009)

Ozcan, E., Bykov, Y., Birben, M., Burke, E.: Examination timetabling using late acceptance hyper-heuristics. In: IEEE Congress on Evolutionary Computation, 2009. CEC '09. pp 997–1004, (2009). https://doi.org/10.1109/CEC.2009.4983054

Padhye, N., Branke, J., Mostaghim, S.: Empirical comparison of MOPSO methods: guide selection and diversity preservation. In: Proceedings of the Eleventh Congress on Evolutionary Computation, IEEE Press, Piscataway, NJ, USA, CEC'09, pp 2516–2523 (2009)

Parsopoulos, K.E., Vrahatis, M.N.: Multi-Objective Particles Swarm Optimization Approaches. In: Multi-Objective Optimization in Computational Intelligence, IGI Global, pp 20–42, 10.4018/978-1-59904-498-9.ch002, (2008). URL http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-59904-498-9.ch002

Phan, D., Suzuki, J.: R2-IBEA: R2 indicator based evolutionary algorithm for multiobjective optimization. In: IEEE Congress on Evolutionary Computation, pp 1836–1845, (2013). https://doi.org/10.1109/CEC.2013.6557783

Reyes-Sierra, M., Coello, C.A.C.: Multi-objective particle swarm optimizers: a survey of the state-of-the-art. Int. J. Comput. Intell. Res. **2**(3), 287–308 (2006)

Sabar, N., Ayob, M., Kendall, G., Qu, R.: A dynamic multiarmed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems. IEEE Trans. Cybern. **45**(2), 217–228 (2015). https://doi.org/10.1109/TCYB.2014.2323936

Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: IEEE International Conference on Evolutionary Computation, pp 69–73, (1998). https://doi.org/10.1109/ICEC.1998.699146

Trivedi, A., Srinivasan, D., Sanyal, K., Ghosh, A.: A survey of multiobjective evolutionary algorithms based on decomposition. IEEE Transactions on Evolutionary Computation PP(99): 1–1, (2016). https://doi.org/10.1109/TEVC.2016.2608507

While, L., Bradstreet, L., Barone, L.: A fast way of calculating exact hypervolumes. IEEE Trans. Evol. Comput. **16**(1), 86–95 (2012)

Yuan, Y., Xu, H., Wang, B., Yao, X.: A new dominance relation-based evolutionary algorithm for many-objective optimization. IEEE Trans. Evol. Comput. **20**(1), 16–37 (2016). https://doi.org/10.1109/TEVC.2015.2420112

Zhang, Q., Liu, W., Li, H.: The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances. In: IEEE Congress on Evolutionary Computation, pp. 203–208, (2009). https://doi.org/10.1109/CEC.2009.4982949

Zhu, Q., Lin, Q., Chen, W., Wong, K.C., Coello, C.A.C., Li, J., Chen, J., Zhang, J.: An external archive-guided multiobjective particle swarm optimization algorithm. IEEE Trans. Cybern. **47**(9), 2794–2808 (2017). https://doi.org/10.1109/TCYB.2017.2710133

Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans. Evol. Comput. **3**(4), 257–271 (1999)