Technological and Economic Development of Economy
ISSN: 2029-4913 / eISSN: 2029-4921
2019 Volume 25 Issue 5: 951–978
https://doi.org/10.3846/tede.2019.10291

APPLICATION OF POPULATION EVOLVABILITY IN A HYPER-HEURISTIC FOR DYNAMIC MULTI-OBJECTIVE OPTIMIZATION

Teodoro MACIAS-ESCOBAR[1,2*], Laura CRUZ-REYES[1], Bernabé DORRONSORO[2], Héctor FRAIRE-HUACUJA[1], Nelson RANGEL-VALDEZ[1], Claudia GÓMEZ-SANTILLÁN[1]

[1]*Division of Postgraduate Studies and Research, Technological Institute of Ciudad Madero, Madero City, Mexico*
[2]*Department of Computer Engineering, Cadiz University, Cadiz, Spain*

Received 31 May 2018; accepted 31 March 2019

Abstract. It is important to know the properties of an optimization problem and the difficulty an algorithm faces to solve it. Population evolvability obtains information related to both elements by analysing the probability of an algorithm to improve current solutions and the degree of those improvements. DPEM_HH is a dynamic multi-objective hyper-heuristic that uses low-level heuristic (LLH) selection methods that apply population evolvability. DPEM_HH uses dynamic multi-objective evolutionary algorithms (DMOEAs) as LLHs to solve dynamic multi-objective optimization problems (DMOPs). Population evolvability is incorporated in DPEM_HH by calculating it on each candidate DMOEA for a set of sampled generations after a change is detected, using those values to select which LLH will be applied. DPEM_HH is tested on multiple DMOPs with dynamic properties that provide several challenges. Experimental results show that DPEM_HH with a greedy LLH selection method that uses average population evolvability values can produce solutions closer to the Pareto optimal front with equal to or better diversity than previously proposed heuristics. This shows the effectiveness and feasibility of the application of population evolvability on hyper-heuristics to solve dynamic optimization problems.

Keywords: population evolvability, hyper-heuristic, dynamic multi-objective optimization, dynamic multi-objective evolutionary algorithm, fitness landscape analysis, DNSGA–II.

JEL Classification: A12, C63, C88, M00.

Introduction

Optimization techniques have become a significant area concerning industrial, economics, business, and financial systems. Meta-heuristic optimization algorithms play a relevant role in innovating real–world practical applications.

*Corresponding author. E-mail: *teodoro_macias@hotmail.com*

Optimization problems can be complex; an example of this is the dynamic multi-objective optimization problem (DMOP), which has multiple objectives to solve at the same time while satisfying a set of constraints. This paper does not consider those problems obtained by multi-objectivization adding objectives to single-objective problems, which can be harder or easier than the original (Brockhoff et al., 2007).

It is expected that these problems could also be affected as time progresses. This dynamism could result in a drop in the quality of the solutions obtained by optimization algorithms, as those changes could make these solutions less effective or even unfeasible.

A dynamic multi-objective optimization problem can be defined formally as

$$\begin{aligned} &\min \quad F(\vec{x},t)=\left\{ f_1(\vec{x},t), f_2(\vec{x},t),\ldots, f_m(\vec{x},t)\right\},\ f_i:\Re^n \to R \\ &\text{s.t.} \quad g(\vec{x},t)>0, h(\vec{x},t)=0, \end{aligned} \tag{1}$$

where $\vec{x}$ is a vector of decision variables F is a set of objectives to be minimized at time t. Lastly, g and h represent the set of inequality and equality constraints, respectively.

The changes over time in a problem could affect the difficulty an algorithm faces to solve it. Therefore, fitness landscape analysis (FLA) methods used on dynamic optimization problems must adapt to said changes. Several FLA methods for dynamic problems using time step control and average values were proposed in (Richter, 2013).

However, most FLA methods both for static and dynamic environments only focus on the properties of the problem and pay no attention to the optimization methods used to solve it. Some researchers (Smith, Husbands & O'Shea, 2002; Wang, Li, Zhang & Yao, 2017) have proposed the use of population evolvability as an FLA method capable of combining the properties of optimization problems and algorithms.

The application of dynamic multi-objective evolutionary algorithms (DMOEAs) to solve DMOPs has been gaining interest among authors in the last decade. Several DMOEAs taking different approaches towards dynamism have been proposed, such as the Dynamic Non-dominated Sorting Genetic Algorithm (DNSGA–II) (Deb, Rao & Karthik, 2007), the Dynamic Competitive Cooperative Coevolutionary Algorithm (dCOEA) (Goh & Tan, 2009) and the Dynamic Multi-objective Evolutionary Algorithm with Core Estimation of Distribution (CDDMEA) (Liu, 2010).

There is also a growing interest in the use of hyper-heuristics to solve optimization problems. These are defined as methodologies that can select or generate a set of low-level heuristics (LLHs) to solve problems (Burke et al., 2010). Therefore, hyper-heuristics use the most suitable method (or permutation of methods) for the current state of the problem to obtain good–quality solutions.

The application of hyper-heuristics to solve single and multiple-objective optimization problems have been proposed before with satisfying results (Burke, Silva & Soubeiga, 2005; García, 2010; Maashi, Özcan & Kendall, 2014; Maashi, Kendall & Özcan, 2015; Zamli, Din, Kendall & Ahmed, 2017; Santiago et al., 2019). Hyper-heuristics have also been applied to tackle dynamic single-objective problems (Baykasoğlu & Ozsoydan, 2017; Topcuoglu, Ucar & Altin, 2014). However, the number of proposed hyper-heuristics used to solve DMOP is scarce and, to our knowledge, uses problem–specific heuristics as LLHs. (Nguyen, Zhang, Johnston & Tan, 2014).

This paper proposes a hyper-heuristic that uses DMOEAs as low-level heuristics to solve DMOPs, named Dynamic Population–Evolvability based Multi-objective Hyper-heuristic (DPEM_HH). In this case, a set of DNSGA–II variations, which will be explained in later sections, is used. Population evolvability is applied in the LLH selection method of DPEM_HH to determine which DMOEA is used in each time step. The results obtained by DPEM_HH show a significant difference over several DNSGA–II variations with respect to convergence and diversity.

The main contributions of this paper can be summarized in three elements: i) The application of population evolvability, a fitness landscape analysis method, as an LLH selection method; ii) A hyper-heuristic capable of solving DMOPs and iii) The use of DMOEAs as LLHs.

The remaining sections of this paper follow the next order. Sections 1 and 2 explain DMOEAs and hyper-heuristics, respectively, as well as previous related work. Section 3 describes population evolvability and its application in algorithm selection task. Section 4 presents DPEM_HH, a hyper-heuristic that incorporates population evolvability on its heuristic selection method. Section 5 establishes the experimental settings to evaluate the results obtained by DPEM_HH. These results are shown and analysed in Section 6. Finally, we explain our conclusions of this paper and the possible future work.

1. Evolutionary algorithms in DMOPs

Evolutionary algorithms (EA) have been previously used to solve multi-objective optimization problems. These EAs are defined as multi-objective evolutionary algorithms (MOEA). One of the most widely known examples is the Non-dominated Sorting Genetic Algorithm (NSGA–II) (Deb, Pratap, Agarwal & Meyarivan, 2002), an MOEA that uses a non-dominating sorting approach in combination with an evolutionary process to solve MOPs. One of the main advantages of MOEAs over exact methods is their capability of obtaining good–quality solutions in a reasonable computation time.

However, MOEAs face challenges when solving DMOPs. As the environment changes, previously found solutions could suffer a loss of their quality or become unfeasible. Therefore, two new elements were inserted in MOEAs to adapt to changes (Deb, Rao & Karthik, 2007). First, a change detection method to find an environment change. When a change is detected, the second element, change response, is applied.

Another important issue that must be considered is that diversity within the population must be maintained to avoid the possibility of getting stuck in local optima (Chen & Zeng, 2011). This is crucially important in dynamic optimization, as environmental changes can massively affect the diversity of a current population. To avoid this situation, the change response strategy used by DMOEAs not only focuses on keeping good–quality solutions but also on promoting diversity within the current population. Different change response strategies with different approaches have been previously proposed. In Azzouz, Bechikh, and Said (2017a), these approaches are classified considering what is the response strategy based on: diversity by population replacement, change prediction, use of external memory, the application of parallel EAs or dividing a DMOP into multiple non-dynamic MOPs.

DNSGA–II is a DMOEA proposed in (Deb, Rao & Karthik, 2007) that uses population replacement to maintain diversity. NSGA–II is modified to handle dynamic optimization problems. First, the algorithm selects a subset of random solutions from the parent population and re–evaluate them. If a change is detected in any objective or constraint functions, the algorithm establishes that there has been a change. Then, the parent solutions are re–evaluated before merging with the child population, this process allows both parent and child populations (already evaluated under the new environment) to be evaluated under the new objectives and constraints.

Also, after a change is detected, a subset of the new population is modified. Two versions with different modification methods were proposed in their work. DNSGA–II–A replaces the selected individuals with randomly created solutions. Meanwhile, DNSGA–II–B takes the selected solutions and mutate them. These alterations to the original NSGA–II allow the population to adapt to changes and avoid stagnating at local optima or having unfeasible solutions.

Other DMOEAs have been proposed in the literature such as the Dynamic Multi-objective EA (DMEA), proposed in (Liu & Wang, 2006), this algorithm splits the DMOP into smaller subperiods using an environment feedback changing operator. Each subperiod is seen as a static environment. Therefore, each subperiod is treated as an MOP and solved using a static EA. Each MOP is transformed into a bi–objective optimization problem using static rank variance and static density variance of the population.

dCOEA (Goh & Tan, 2009) is a DMOEA that uses a cooperative–competitive mechanism to solve DMOPs and a memory strategy to maintain diversity. A fixed number of archived solutions are used to detect changes. When a change occurs, dCOEA initiates a competitive process combining individuals from subpopulations and others generated stochastically to increase diversity. Also, a temporal archive containing outdated solutions is used to provide previously obtained information to the current population to allow it to adapt faster to a new environment.

The Multi–strategy Ensemble Multi-objective Evolutionary Algorithm (MS–MOEA) (Wang & Li, 2010) is a memory–based DMOEA. To create offspring, MS–MOEA uses adaptive genetic and differential operators and a Gaussian mutation operator. If a change is detected, solutions are either replaced by new solutions or reinitialized by selecting a random archived solution and adding values to each of its variables according to a Gaussian distribution.

CDDMEA (Liu, 2010) is a change–prediction based DMOEA that incorporates a core estimation of distribution model taking Pareto–optimal solutions obtained in previous environments to predict the position of the optimal solutions in the next environment. The Population Prediction Strategy (PPS), proposed in Zhou, Jin, and Zhang, (2014) is another change–prediction strategy that divides the Pareto set into two parts: a centre point and a manifold. The centre points and manifolds obtained during each previous time step form a sequence and are used to predict the next centre and manifold, respectively. This strategy was incorporated into a dynamic version of the Regularity Model–based Multi-objective Estimation of Distribution Algorithm (RM–MEDA) (Zhang, Zhou & Jin, 2008) and compared to versions of the same algorithm with a random initialization strategy and the feed–forward

prediction strategy (Hatzakis & Wallace, 2006) with RM–MEDA with PPS statistically outperforming the other tested DMOEAs.

In Azzouz, Bechikh, and Said (2017b), an adaptive hybrid population management strategy was proposed. This strategy uses several change response strategies based on memory, local search and randomness to handle dynamic environments. After a change is detected, this strategy evaluates the change severity and determines the size of the memory and number of random solutions that will be used as a change response. A dynamic version of NSGA–II, named Adaptive Population Management–Based Dynamic NSGA–II (A–Dy–NSGA–II), is presented to utilize this strategy as its change response. A–Dy–NSGA–II is compared against a dynamic improved version of NSGA–II (INSGA–II) proposed in Wang and Li (2009), using several different change response strategies with A–Dy–NSGA–II providing better results while keeping adaptability and robustness even when facing substantial changes.

The steady–state and generation EA (SGEA) (Jiang & Yang, 2017) is a recently proposed DMOEA that combines an evolutionary algorithm with a steady–state algorithm, using the latter for offspring creating and the former for population update. This DMOEA incorporates a prediction strategy that reuses well–distributed outdated solutions and new solutions close to the Pareto front based on information obtained from previous environments. SGEA produced satisfying results over renown DMOEAs and strategies such as DNSGA–II, dCOEA, and PPS for DMOPs from the FDA (Farina, Deb & Amato, 2004) and dMOP (Goh & Tan, 2009) test suites.

While there are many DMOEAs proposed in the literature, this paper focuses on the use of DNSGA–II as an element of DPEM_HH because despite being shown to be outperformed by other recent DMOEAs, it holds a high recognition in the dynamic and non-dynamic optimization area for of its simplicity and effectiveness. Also, the NSGA–II source code is available in several MOEA frameworks such as jMetal[1] and both the change detection and response methods of DNSGA–II can be easily implemented into this algorithm.

2. Hyper-heuristics

Meta-heuristics work directly with the solution space, obtaining the most suitable solution or set of solutions according to their defined procedure. However, considering that there are problems with high difficulty and the ample quantity of proposed meta-heuristics, the existence of a methodology capable of selecting from a set of algorithms those that give a better solution for a problem is desirable.

Hyper-heuristics are defined as "an automated methodology for selecting or generating heuristics to solve hard computational search problems" (Burke et al., 2010). This methodology or high–level heuristic (HLH) operates over a heuristic search space, which contains a set of heuristics or meta-heuristics known as low-level heuristics (LLH). The HLH selects the most suitable LLH to apply for solving a problem in its current state until the next selection stage. Hyper-heuristics do not act directly on the solution search space, as they instead select heuristics that will act on that search space, creating new solutions or modifying pre–existing ones.

[1] Framework downloaded from http://jmetal.github.io/jMetal/.

It is possible that a set of LLHs being applied in a certain combination on a problem could cover the flaws of each heuristic, obtaining a better solution compared to what said heuristics could obtain individually. Another advantage is that the variety of the LLHs used could allow a hyper-heuristic to solve a more extensive array of optimization problems in comparison to single DMOEAs.

A classification of hyper-heuristics based on two dimensions is presented in (Burke et al., 2010). The first dimension classifies them according to the nature of the heuristic search space: selection, where LLHs are selected from a set; and generation, where LLHs are generated using available components. Another matter to consider in this dimension is the nature of the low-level heuristics. For this case, two types of LLHs are considered, heuristics that create new solutions and heuristics that alter existing solutions. The second dimension considers if the hyper-heuristic applies a learning mechanism and how is it used. Hyper-heuristics can be classified as non-learning, online (obtain knowledge during execution) or offline (obtain knowledge prior execution) learning.

The process of hyper-heuristics with heuristic–selection methods using perturbative LLHs can be divided into two stages, according to Bilgin, Özcan, and Korkmaz (2006): the heuristic selection mechanism and move acceptance. The heuristic selection method is used to determine which LLH to apply to the problem until the next selection stage. These methods can be adaptive or non-adaptive. Adaptive methods learn from previous selection stages and use any obtained information to update their selection criterion. Several adaptive methods have been previously presented, such as Random Descent, Permutation Descent, Choice Function (Cowling, Kendall & Soubeiga, 2000), Tabu Search (Dowsland, Soubeiga & Burke, 2007) or Roulette Wheel with updated probabilities (Baykasoğlu & Ozsoydan, 2017; Santiago et al., 2019). Meanwhile, non-adaptive methods only consider currently existing data to perform its selection, such as Simple Random or Greedy (Cowling, Kendall & Soubeiga, 2000).

The move acceptance criterion analyses the solutions obtained by the selected LLH and compares them with the current solutions. Then, it determines if the newly obtained solutions replace the current solutions or not. Move acceptance criteria can be deterministic or non-deterministic. Deterministic criteria will always produce the same result, given a specific configuration, such as All Moves, Only Improving or Equal–or–Improving criteria (Cowling, Kendall & Soubeiga, 2000). Non-deterministic criteria will not always bring the same outcome as there are parameters such as time or probabilities that can alter the result. Move acceptance criteria based on Monte Carlo (Ayob & Kendall, 2003), Simulated Annealing (Bai & Kendall, 2005) or Late Acceptance (Burke & Bykov, 2008) are examples of non-deterministic approaches.

The application of hyper-heuristics to solve multi-objective problems is a recent area of research that has been gaining interest among researchers recently. A hyper-heuristic that uses tabu search with roulette wheel selection (TSRoulWheel) is proposed in (Burke, Silva & Soubeiga, 2005) to solve space allocation and timetabling problems. In García (2010), a hyper-heuristic based on genetic algorithm for Social Portfolio Problem (HHGA_SPP) is presented as a methodology capable of solving multi-objective social portfolio problems using a population of permutations of the available low-level heuristics. HHGA_SPP uses a genetic algorithm to select the most adequate permutation. The Markov Chain-based Hyper-heu-

ristic (MCHH) (McClymont & Keedwell, 2011) is an online–learning hyper-heuristic that employs reinforcement learning and Markov chains to update transition weights between a set of low-level heuristics according to their performance. MCHH was applied to the DTLZ test suite (Deb, Thiele, Laumanns & Zitzler, 2002) and compared against a random heuristic selection and TSRoulWheel. The results showed that MCHH has a better convergence with the Pareto optimal front. A Fast Multi-objective Hyper-heuristic Genetic Algorithm (MHypGA), proposed in Kumari, Srinivas, and Gupta (2013), uses reinforced learning with adaptive weights to select from a set of low-level heuristics based on different combinations of selection, crossover and mutation operators. A hyper-heuristic that uses a fuzzy logic engine named Fuzzy Adaptive Multi-objective Evolutionary algorithm (FAME) (Santiago et al., 2019) has been recently proposed. This methodology manages several reproduction operators as low-level heuristics. FAME uses a roulette–wheel mechanism to select the set of crossover and mutation operators to apply on an MOP. The probabilities of each operator are defined by using the Mamdani–type fuzzy inference system (Roy & Chakraborty, 2013). The results obtained by FAME have shown better overall performance than several state–of–the–art algorithms.

While most hyper-heuristics use simple low-level heuristics such as swap or random mutations, some hyper-heuristics using more complex low-level heuristics, such as meta-heuristics, have been proposed before. A new selection method called Fuzzy Inference Selection (FIS) was proposed and used in a hyper-heuristic in (Zamli, Din, Kendall & Ahmed, 2017). The LLH set was composed by the search operator mechanisms from four different meta-heuristics to solve a combinatorial t–way test suite generation problem.

A Choice–function based hyper-heuristic (HH_CF) proposed in Maashi, Özcan, and Kendall, (2014), used three MOEAs, including NSGA–II, as LLHs. A choice function was used to select which MOEA to apply on the MOP being solved for a set of generations. This choice function (*CF*) was applied to each LLH *h*. *CF* was based on two elements, a two–stage ranking scheme (c_1) and a time variable (c_2) and was defined as

$$CF(h) = \alpha c_1(h) + c_2(h). \tag{2}$$

The LLH with the highest *CF*(*h*) was selected. In the ranking scheme, the first stage ranks the solutions obtained by each LLH using multiple performance metrics, making a ranking for each metric. The second stage ranks how many times an LLH had the best rank for all metrics. After this, c_1 for each LLH was calculated by

$$c_1(h) = 2 * (N+1) - \{Freq_{rank}(h) + RNI_{rank}(h)\}, \tag{3}$$

where *N* is the number of LLHs, $Freq_{rank}(h)$ is the ranking of each LLH for the second–stage ranking and $RNI_{rank}(h)$ is the position of a heuristic based on a ranking of a metric that calculates the ratio of non-dominated individuals with respect to the entire population that each heuristic obtained. As it could be noticed, c_1 supports intensification. Meanwhile, c_2 supports diversification as it is defined as the seconds elapsed since the last time the respective LLH was used. Therefore, low-level heuristics that have been not chosen recently will have a higher value with respect to LLHs that were just called. The variable α is a large positive value, set by the authors, that is used to maintain a balance between both functions.

In Maashi, Kendall, and Özcan, (2015), HH_CF was evaluated on the WFG test suite (Huband, Hingston, Barone & While, 2006) and the vehicle crashworthiness design problem (Liao, Q. Li, Yang, Zhang & W. Li, 2008) using three different move acceptance criteria: All Moves, Great Deluge Algorithm (Dueck, 1993) and Late Acceptance (Burke & Bykov, 2008). The results obtained demonstrate the effectiveness of using complex low-level heuristics, such as MOEAs to solve MOPs.

Among the meta-heuristics and MOEAs proposed in the literature, NSGA–II is one of the algorithms most commonly chosen as an element of a hyper-heuristic, usually as a low-level heuristic. Proposals such as HH_CF, which was previously mentioned, and the works of Vrugt and Robinson, (2007), Furtuna, Curteanu, and Leon (2012), Vázquez-Rodríguez and Petrovic (2012), Li, Özcan and John (2017) are examples of the application of NSGA–II within a hyper-heuristic. This is mostly because of the simplicity of this MOEA and its worldwide renown. Despite being outdated and already outperformed by new algorithms, it still is one of the most recognized evolutionary algorithms today and usually accepted as a comparison point when presenting new MOEAs and hyper-heuristics.

Hyper-heuristics have also been used in dynamic optimization context. A memory/search algorithm is merged with a hyper-heuristic on a hybrid technique to solve dynamic single objective problems in Topcuoglu, Ucar, and Altin, (2014). This new framework showed a significant performance difference in comparison to the memory/search algorithm when solving the dynamic generalized assignment problem and the moving peaks benchmark (Branke, 2001). In Baykasoğlu and Ozsoydan (2017), three meta-heuristics, adapted to dynamic changes using auxiliary procedures, were used as LLHs on a hyper-heuristic to solve dynamic multi–dimensional knapsack problems. Each LLH was assigned a selection probability based on the improvement in the quality of the solutions obtained compared to previous solutions. Then, an LLH was selected randomly using the assigned probabilities, promoting the selection of high–improving LLHs. Diversity was also kept by using Triggered Random Immigrants and Adaptive hill–climbing strategies (H. Wang, D. Wang, Yang, 2009). However, most of the current hyper-heuristics focus on single-objective dynamic optimization problems. One of the few works that focus on DMOPs is the Multi-objective Generic Programming based Hyper-heuristic (MO-GPHH), proposed in Nguyen, Zhang, Johnston and Tan (2014), which is focused on the dynamic multi-objective job shop scheduling problem. MO–GPHH is a coevolutionary hyper-heuristic that uses an MOEA, including NSGA–II, as a search strategy to define effective scheduling programs.

While hyper-heuristics have been previously used to solve dynamic optimization problems, to our knowledge there is no existence of a hyper-heuristic that can solve DMOPs using as LLHs a set of dynamic meta-heuristics. In theory, the use of more complex LLHs should allow the hyper-heuristic to solve a wider array of problems effectively. Therefore, this paper proposes a selection–based hyper-heuristic capable of solving DMOPs using DMOEAs as LLHs.

There are few works that use fitness landscape analysis methods as a part of a hyper-heuristic. In Soria-Alcaraz, Ochoa, Sotelo-Figeroa, and Burke (2017), evolvability and landmarking are used by considering a ratio of fitness–improving solutions from a set of neighbours of an initial solution and by using a First–Improvement Hill–Climbing of a solution, re-

spectively. Both methods are used as part of offline–learning strategies to define selection probabilities to each low-level heuristic. Evolvability is used as part of an online–learning strategy in Soria-Alcaraz, Espinal, and Sotelo-Figueroa (2017) by estimating it during each search step using a pre–trained neural network. Both proposals are focused on single-objective optimization, and the FLA methods are applied on a single initial solution.

While FLA methods have been previously used within a hyper-heuristic, to our knowledge, these methods have not been used for dynamic multi-objective optimization. In the specific case of evolvability, its application has only been reported for static single-objective optimization problems and considering the evolvability of single solutions instead of a whole population. Our proposal seeks to utilize a more complex concept of evolvability, named population evolvability, as an online–learning strategy used on a heuristic–selection method. Population evolvability, as the name implies, can evaluate the evolvability of an entire population and will be explained in detail in the next Section.

Also, as previously indicated, there are many different move acceptance criteria in the literature, each one with its degree of complexity and effectiveness. However, we must consider that the focus of this work is to analyse the application of population evolvability as part of a heuristic–selection method, as well of the use of DMOEAs as LLHs. Therefore, it is advisable to begin with the simplest methods and criteria and use, as future work, more complex elements once the usability of this FLA method has been demonstrated. Therefore, the Greedy and Choice Function heuristic–selection methods and the All Moves criterion are used for this paper.

3. Population evolvability

A common issue with FLA methods applied to optimization problems is that they only focus on the properties of a problem, disregarding the algorithm being used to solve it. However, the results obtained by those methods could be misleading, as problems that are defined as hard could be easily solved by a certain group of algorithms while problems considered as easy could be hard to solve (Wang, Li, Zhang & Yao, 2017).

An alternative for EAs to avoid this issue is the use of evolvability as a method to measure the difficulty of a problem. Evolvability is defined as "the ability of a population to produce variants fitter than any yet existing" (Altenberg, 1994). This concept can be used to correlate an optimization problem and algorithm performance.

A method to calculate evolvability of a solution was presented in Smith, Husbands, and O'Shea (2002), by using fitness evolvability portraits. In their work, the evolvability E_a of a candidate solution x is calculated based on a neighbourhood of the solution $N(x)$ and a subset of that neighbourhood containing only solutions with better fitness than the candidate solution $N^*(x) = \{d \mid d \in N(x), f(d) \geq f(x)\}$

$$E_a = \frac{|N^*(x)|}{|N(x)|}. \tag{4}$$

E_a indicates the probability of obtaining a mutation with better fitness than the original solution. Therefore, a high E_a means that the current solution can improve easily. A low E_a

value means that the problem can be considered hard, as it will be difficult for the optimization algorithm to obtain a better solution.

Population evolvability was presented in Wang, Li, Zhang, and Yao (2017), as an extension of the original concept. This work evaluates the evolvability of an entire population instead of a single solution focusing on two factors: a) the probability of the candidate population to evolve into a better solution set, and b) the evolutionary ability of the algorithm used during the optimization process. Population evolvability is calculated using the next equation.

$$evp(P_i) = \begin{cases} \dfrac{\sum\limits_{P_{ij} \in N^+(P_i)} \dfrac{|f^b(P_i) - f^b(P_{ij})|/NP}{\sigma(f(P_i))}}{|N(P_i)|}, & |N^+(P_i)| > 0, \\ 0, & |N^+(P_i)| = 0. \end{cases} \quad (5)$$

where P_i is the candidate population, $f^b(P)$ is the best fitness value obtained in a population P, $\sigma(f(P_i))$ is the standard deviation of the fitness values of the population P_i, NP is the size of P_i, $N(P_i)$ is a set of children populations of P_i defining its neighbourhood, $N^+(P_i)$ is a subset of $N(P_i)$ which contains only neighbours with a better fitness value than P_i. For minimization problems this is denoted as $N^+(P_i) = \{P_{ij} \mid P_{ij} \in N(P_i), f^b(P_{ij}) < f^b(P_i)\}$ in which $j = 1,\ldots, |N^+(P_i)|$.

The range of $evp(P_i)$ is $[0, +\infty)$. This value represents the evolutionary ability of the current state, which is related to the difficulty that the applied algorithm will have to find better solutions for the current problem. In other words, the higher the $evp(P_i)$ value is, the easier for an algorithm to find new solutions with better fitness.

The authors of this concept indicate that population evolvability is feasible for algorithm selection task if it is viewed as a set of obtained characteristics for population–based meta-heuristic algorithms. However, their experiments were applied to non-dynamic single-objective optimization problems and have not been tested yet in a dynamic multi-objective environment. Therefore, this paper takes this FLA method and adapts it to handle dynamism to test its feasibility as a selection method of DMOEAs that solve DMOPs.

One of the main advantages of population evolvability over other FLA methods previously used on hyper-heuristics is the depth of the information obtained by this method. For the FLA methods mentioned in Section 2, evolvability finds the probability of a solution to evolve into a fitter state while landmarking can detect the degree of its evolution. Population evolvability obtains information regarding both topics at the same time. Another point that must be taken into consideration is that evolvability and landmarking focuses on a single solution, while population evolvability works with an entire population, allowing a fitness landscape analysis on a diverse set of solutions. Finally, it must be considered that population evolvability is a novel concept that offers a good breakthrough point on dynamic and multi-objective optimization.

4. Dynamic population–evolvability based multi-objective hyper-heuristic

As the number of proposed DMOEAs to solve DMOPs increase, it becomes more difficult to know which DMOEA (or a subset of DMOEAs) could give the best solutions while solving a certain DMOP. There is also the possibility that the application of a set of DMOEAs on a particular order could obtain better solutions in comparison to what those algorithms could find individually. As mentioned in Section 2, hyper-heuristics have a heuristic selection method that allows them to select from a set of heuristics, which one to use to solve a problem.

Therefore, this work presents the dynamic population–evolvability based multi-objective hyper-heuristic (DPEM_HH). A set of DMOEAs are used as low-level heuristics for DPEM_HH. Also, population evolvability is considered in the heuristic selection process and its application in the proposed hyper-heuristic will be explained in detail later in this Section.

As mentioned in Section 3, population evolvability has yet to be tested as an algorithm selection method on dynamic optimization problems. Considering this, DPEM_HH uses this FLA method on its LLH selection method. The approach allows all DMOEAs from the LLH set to be evaluated after each change, selecting the algorithm that could perform better for that time step according to the LLH selection method.

Figure 1 shows the flowchart of DPEM_HH. First, the hyper-heuristic creates a random initial population. Then, an LLH is randomly selected and applied to the problem. If the stopping criteria have not been reached and a change in the environment has been detected, the heuristic selection process begins. First, each LLH is given a copy of the current population. Then, each heuristic applies its change response method in their respective copy population and is executed for a set of sampled generations. Population evolvability is calculated every generation for all LLHs.

After the sampling process ended, the average population evolvability of each LLH is calculated by dividing the sum of population evolvability values obtained by the number of sampled generations. These values are used in an LLH selection method to define which DMOEA will be used for the next generations until a change is detected or the stopping criteria are reached.

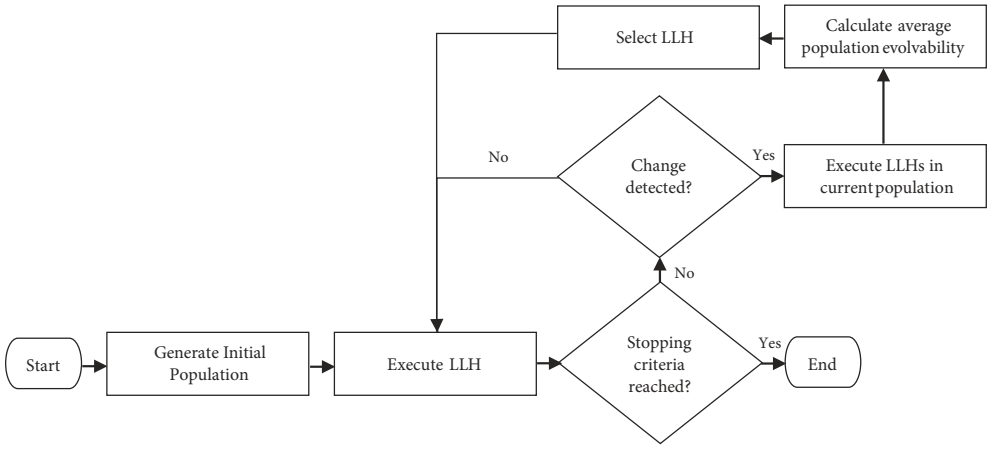

Figure 1. DPEM_HH flowchart

According to Burke et al. (2010), DPEM_HH is classified as an online–learning, heuristic–selecting methodology. In other words, it updates its information about all available low-level heuristics on each selection stage and chooses from that LLH set which heuristic to apply on the problem for a time step during the process.

Population evolvability can be a very costly process because of its exploration process during each generation. Therefore, the number of sampled generations must be carefully defined to avoid a very computationally complex process. The authors of this concept (Wang, Li, Zhang & Yao, 2017) suggest using 20% of the total number of evaluations. Translating this consideration into the dynamic optimization domain, focusing specifically on DMOP, the maximum number of sampled generations per time step for DPEM_HH has been set in 20% of the number of generations passed between changes.

Two LLH selection methods that use population evolvability are proposed and compared in this work: the first method applies a greedy strategy that uses the average population evolvability value obtained by each LLH during the selection process to select an LLH. The second proposal presents a choice function that combines a set of performance metrics used to analyse the solution set obtained by each LLH during the selection process and its respective average population evolvability to perform an LLH selection.

Average population evolvability (EVP). Each sampled generation, a set of neighbour populations is created, and population evolvability of the current generation is calculated for each LLHs. To do so, equation (5) is adapted to handle dynamic environments by adding a variable t denoting the current static period of the problem. However, since said equation focuses on single objective problems, it uses only one value to represent the fitness of a solution. Therefore, an aggregation method must be used to combine all objective function values. DPEM_HH uses a weighted sum method (Li & Deb, 2017), combining the value of m objectives and a weight vector $\lambda = (\lambda_1, ..., \lambda_m)$ to obtain a single fitness value. For this work, all objectives have equal importance. Thus, the vector weight is distributed evenly for DPEM_HH. The aggregation function is defined as

$$\min \quad f(x \mid \lambda) = \sum_{i=1}^{m} \lambda_i f_i(x) \quad \text{s.t.} \quad \lambda_i \geq 0, \sum_{i=1}^{m} \lambda_i = 1, \tag{6}$$

even though it is known that this approach is not suitable to find the nonconvex parts of the Pareto–optimal front, it is only used to compare solutions obtained during the LLH selection process. Meanwhile, the Pareto dominance is used to determine the non-dominated solution set and the solutions that are passed from a generation to the next one.

The modified equation (7) can be used on dynamic optimization problems as the variable t, denoting the current time step, has been added to all relevant parameter. The first child population obtained by each LLH is set as the new population for the next sampled generation. This means that all neighbour populations that were evaluated to obtain $evp_t(P_{i,t})$ have no effect on the quality of the next sampled generation and they are only used to calculate this value. Therefore, it can be compared against non-hyper-heuristic methods, such as DMOEAs.

$$evp_t(P_{i,t}) = \begin{cases} \dfrac{\sum\limits_{P_{ij,t} \in N^+(P_{i,t})} \dfrac{|f_t^b(P_{i,t}) - f_t^b(P_{ij,t})|/NP_t}{\sigma(f_t(P_{i,t}))}}{|N(P_{i,t})|}, & |N^+(P_{i,t})| > 0, \\ 0, & |N^+(P_{i,t})| = 0. \end{cases} \quad (7)$$

For each LLH, the average population evolvability is obtained following equation (8), where *st* indicates the number of sampled generations. This heuristic–selection method follows a greedy strategy. Therefore, the LLH with the highest average evolvability value is selected and used until the next change is detected. Average population evolvability supports exploitation of the solution search space by thoroughly analysing the neighbourhood of each parent population during the sampled generations. Exploration is supported by analysing a solution set in comparison to other FLA methods that only focus on a single solution.

$$\overline{evp_t} = \frac{1}{st}\sum_{i=1}^{st} evp_t(P_i). \quad (8)$$

Choice function (CF): This LLH selection method integrates average population evolvability together with a set of performance metrics applied at the end of the sampled generations, using them in a choice function. The objective is to analyse the possibility of obtaining a diverse set of high–quality solutions using a more complex method based on the properties of the DMOP being solved, the DMOEA used and the quality of the solutions obtained during the sampled generations.

The choice function is based on equation (2) proposed in (Maashi, Özcan & Kendall, 2014), which uses a two–stage ranking scheme (c_1) and a time variable (c_2) for the LLH selection process. The previously presented average population evolvability selection method is a greedy method, choosing the best LLH according to its average evp_t value. Meanwhile, this method supports both intensification and diversification as it was previously explained in Section 2. The c_1 function is modified to insert population evolvability into the said equation by making the ranking of average population evolvability obtained by each LLH the most relevant value. Therefore, the modified c_1 is

$$c_1(h) = 2 * (N+1) - \{Freq_{rank}(h) + EVP_{rank}(h)\}, \quad (9)$$

where $EVP_{rank}(h)$ is the ranking of each LLH according to their average population evolvability. The milliseconds elapsed since the last time an LLH was selected is used to calculate c_2.

This heuristic selection method also takes the exploitation versus exploration dilemma into consideration. As it was mentioned, the c_1 function promotes intensification by searching the LLH with the best–quality solutions for the current period. Also, the addition of average population evolvability means that each parent population from the sampled generations will perform an exhaustive local search looking for fitter neighbour populations. Exploration is supported by the c_2 function, as it is regulated not by the quality of the solution set obtained by an LLH, but the time elapsed since that heuristic was used. Therefore, an LLH that has not been used by DPEM_HH for a long time could be selected even if the quality of its

solutions is not currently the best, allowing the LLHs from DPEM_HH to explore areas from the solution search space that would not have been explored if only c_1 had been considered.

As mentioned in Section 2, the move acceptance criterion of DPEM_HH is *All Moves* (Cowling, Kendall & Soubeiga, 2000). This criterion accepts any new population of solutions obtained by the currently selected LLH. This selection was based on the simplicity of the move acceptance criterion and to test the capability of DPEM_HH to cover the weaknesses of each of its LLHs with their combined strengths.

Algorithm 1 describes the procedure of DPEM_HH. The input is the number of generations per static period, also defined as change frequency (τ_τ), the number of sampled generations per evaluation period (*st*) and the LLH set. The output is a population found by DPEM_HH after the stopping criteria have been reached. First, an initial population is created and a variable to control the number of generations for every execution (τ_{test}) is initialized. Then, the first LLH to be used is chosen randomly and executed for $\tau_\tau - \tau_{test}$ generations.

Algorithm 1 DPEM_HH. Evolvability–based multi-objective hyper-heuristic

BEGIN

1: $pop \leftarrow$ GenerateInitialPopulation()
2: $\tau_{test} \leftarrow 0$
3: $s \leftarrow$ RandomSelectLLH(LLH)
4: **while** stopping criteria not reached
5: $pop \leftarrow$ ExecuteLLH(s, pop, $\tau_\tau - \tau_{test}$)
6: $\tau_{test} \leftarrow 0$
7: **if** change is detected
8: $\tau_{test} \leftarrow st$
9: **for** $i = 1$ to $|$LLH$|$
10: $pop\_evp_i \leftarrow$ Copy(pop)
11: **for** $j = 1$ to st
12: $pop\_evp_i \leftarrow$ ExecuteLLH(llh_i, pop_evp_i, τ_{test})
13: $NP \leftarrow$ GenerateNeighbourhood(llh_i, pop_evp_i)
14: $evp_i \leftarrow evp_i +$ CalculateEvolvabillity(pop_evp_i, NP)
15: **end for**
16: $evp_i \leftarrow evp_i/st$
17: **end for**
18: $MET \leftarrow$ MetricEvaluation(POP_EVP)
19: $s \leftarrow$ SelectLLH(LLH, EVP, MET)
20: $pop \leftarrow$ SetPopulation(s, POP_EVP)
21: **end if**
22: **end while**

END

Since the first selection was random and no population evolvability test was done, τ_{test} is set on 0 for the current LLH execution. When a change is detected, τ_{test} is given the same value than *st*. This is done because the generations used by the selected LLH to compute its respective population evolvability must be accounted for. Then, each LLH is given a copy of the current population (pop_evp_i represents the population copy assigned to the *i-th* LLH) and performs a population evolvability analysis for *st* generations. As equation (7) shows, a population neighbourhood is required to calculate population evolvability. This is done by generating a set of neighbour populations (NP) for each LLH on every sampled generation, which is formed by creating a set of children population (line 13). The average population evolvability of all available LLHs (evp_i denoting the respective evolvability of the *i-th* LLH) and performance metric values (*MET*) of each LLH are calculated (line 16 and 18, respectively). After those values have been obtained, an LLH is chosen according to the selection method being used. Lastly, the current population of the selected LLH obtained during the sampled generations is set as the current population (line 20).

5. Experimental design

Eleven test problems are applied in this work to examine the performance of DPEM_HH in comparison with other algorithms. FDA1, FDA3, FDA4, and FDA5 from the FDA test suite (Farina, Deb & Amato, 2004), the DMZDT test suite (Wang & Li, 2010) and the dMOP test suite (Goh & Tan, 2009). This allows DPEM_HH to be tested in type I (non-changing Pareto optimal front), II (changing Pareto optimal front and optimal solution set) and III (non-changing optimal solution set) DMOPs, according to the classification in Farina, Deb, and Amato, (2004), with convex, nonconvex, discontinuous and multimodal Pareto optimal fronts with uniform and nonuniform optimal solution sets. The time instance t of these DMOPs has been adapted to be set as $t = (1/n_t)[\tau/\tau_\tau]$ (where n_t represents change severity, τ_τ change frequency and τ the current generation). The definition of these problems can be found in the referenced works.

5.1. Algorithms and performance metrics

Two configurations of DPEM_HH are tested in this paper. Each one using one of the LLH heuristic selection methods presented in Section 4 and the All Moves acceptance criteria. DPEM_HH uses three versions of DNSGA–II as LLHs. Each version has a different change response method. The first two versions used, DNSGA–II–A and DNSGA–II–B, were proposed in Deb, Rao, and Karthik, (2007) and explained in Section 1. The third version, DNSGA–II–AB, is proposed in this work. After a change DNSGA–II–AB randomly selects a subset of the solutions from the new population, replacing half of that subset with newly created solutions and mutating the other half.

When analysing the performance of an algorithm regarding the quality of solutions obtained for multi-objective optimization, it is desirable to use a set of performance metrics that can cover as many properties as possible. Such as the number of non-dominated solutions, how close is the solution set to the Pareto–optimal front (POF*) and the uniformity of the distribution among solutions.

In Maashi, Özcan, and Kendall (2014), the set of performance metrics used for the choice function do not consider the POF*. However, it was only applied to static MOPs. DPEM_HH works with DMOPs so there is a possibility that a solution set loses quality and diversity after a problem changes, making an analysis more difficult. Therefore, this paper uses performance metrics that compare the solution set with the current POF*. The POF* of all the selected DMOPs can be mathematically obtained.

The following performance metrics are used for both the choice function and to evaluate the solutions obtained by DPEM_HH and other algorithms used in this experimental study. This selection is based on the information these metrics can provide regarding the size of the non-dominated solution set, convergence to the POF* and diversity of the solution set with respect to itself and the POF*.

Ratio of non-dominated individuals (RNI) (Tan, Lee & Khor, 2002). RNI compares the number of non-dominated individuals obtained in the Pareto–optimal front (POF) with the size of the population P. A higher RNI value means an algorithm found more non-dominated solutions.

$$\mathrm{RNI} = \frac{|\mathrm{POF}|}{|\mathrm{P}|}. \tag{10}$$

Inverted generational distance (IGD) (Sierra & Coello, 2005). IGD measures the convergence and diversity of an obtained POF. This metric is defined as:

$$\mathrm{IGD}(\mathrm{POF},\mathrm{POF}^*) = \frac{\sum_{v \in \mathrm{POF}^*} d(v,\mathrm{POF})}{|\mathrm{POF}^*|}, \tag{11}$$

where POF* is a set of uniformly distributed points in the true POF and $d(v, \mathrm{POF})$ is the shortest Euclidean distance between v and every point of POF. A smaller value means a better convergence.

Maximum spread (MS) (Goh & Tan, 2007). MS measures how well the obtained POF covers the true POF*

$$\mathrm{MS} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left[\frac{\min\left[\overline{\mathrm{POF}_i}, \overline{\mathrm{POF}_i^*}\right] - \max\left[\underline{\mathrm{POF}_i}, \underline{\mathrm{POF}_i^*}\right]}{\overline{\mathrm{POF}_i^*} - \underline{\mathrm{POF}_i^*}}\right]^2}, \tag{12}$$

where $(\overline{\mathrm{POF}_i}, \overline{\mathrm{POF}^*_i})$ are the highest and $(\underline{\mathrm{POF}_i}, \underline{\mathrm{POF}^*_i})$ are the lowest values for POF and POF* for objective i, respectively. A high MS value equals a better spread.

Hyper–volume ratio (HVR) (Van Veldhuizen, 1999). Given the hypervolume of a POF using a reference point, denoted as $HV(\mathrm{POF})$. HVR obtains the ratio of this value with the hypervolume of POF* to measure the convergence and diversity of POF with respect to POF*. Therefore, the best possible ratio value is 1.

$$\mathrm{HVR} = \frac{HV(\mathrm{POF})}{HV(\mathrm{POF}^*)}. \tag{13}$$

In our consideration, this set of metrics provides enough data about the performance of each LLH to make a well–based heuristic selection. Also, it must be considered that all performance metrics previously mentioned have been already used for dynamic multi-objective optimization problems to analyse the performance of EAs. Therefore, this same metric set is also used to evaluate both DPEM_HH configurations when compared against each other and IGD, MS and HVR when compared to the three versions of DNSGA–II tested in this work.

5.2. Parameter setting and implementation

Population size for all test problems is set to 100 for DPEM_HH and DNSGA–II. Every DNSGA–II version, both when running individually or used as LLHs, uses simulated binary crossover as crossover operator and polynomial distribution as mutation operator (Agrawal & Deb, 1995). The crossover and mutation probabilities are set to 0.9 and $1/n$, where n is the number of variables of the DMOP, respectively. Distribution indexes are set to 10 for crossover and 20 for mutation. These values are defined following the original setting of DNSGA– II (Deb, Rao & Karthik, 2007).

Change detection is set according to the current generation being evaluated. When the problem reaches a generation where an alteration on t is detected all algorithms start their respective change response methods. Each DNSGA–II versions apply its change response method to 20% of the new population for every change. In DPEM_HH, the total sampled generations on each time step is set as the 20% of τ_τ. For the choice function, α is set to 2000.

For the FDA and dMOP test suite problems the change severity n_t is set to 10, the number of generations for each time step τ_τ is set to 25. Meanwhile for the DMZDT test suite $n_t = 100$, and $\tau_\tau = 25$. These values were defined according to recommendations from both authors and the guideline defined in Helbig and Engelbrecht (Helbig & Engelbrecht, 2014). For all DMOPs, the total number of generations is set as $n_t{}^*\tau_\tau$.

DPEM_HH was implemented using the framework jMetal 5.1[1], modifying the NSGA–II algorithm provided by this framework to develop the three DNSGA–II versions used as LLHs in this paper. All the algorithms were run 30 times for each DMOP using a laptop with an Intel Celeron 2.13 GHz processor and 4GB RAM.

6. Experimental results and discussion

Both configurations of DPEM_HH are executed for each DMOP. The execution time of the instances from the FDA and dMOP test suites is in a range of 10 to 30 seconds. Meanwhile, DMZDT problems take 1–2 minutes per execution. The two versions of DPEM_HH are compared by evaluating the offline (average of all-time steps) mean and standard deviation of RNI, IGD, MS and HVR of the solution set obtained by each algorithm. Table 1 shows the configuration that obtained the best value for each DMOP per metric and the total number of times a configuration is superior when compared against the other version of DPEM_HH for each metric.

DPEM_HH using CF as the LLH selection method provide the best results for RNI and MS in a higher number of instances than the other version. Meanwhile, DPEM_HH using

EVP obtained the best results regarding IGD and HVR. The difference between both versions with respect to IGD is relevant, with a significant advantage towards average population evolvability. These results lead to an implication that DPEM_HH using this LLH selection method can provide solutions closer to the POF* but slightly less spread and with a smaller non-dominated solution set.

Focusing on each DMOP, EVP is superior in all metrics for dMOP1 and DMZDT4, while CF dominated in all metrics for dMOP2 and DMZDT1. For the FDA instances, EVP provides the best results in terms of closeness to the POF* while CF obtained solution sets with better diversity. Analysing the DMZDT instances, EVP has a clear dominance over CF with respect to convergence and diversity taking into consideration its dominance for IGD and MS, with CF providing better results for only DMZDT1. Lastly, the results obtained for the dMOP test suite slightly favour CF, especially for dMOP2 and all metrics but RNI in dMOP3. However, the results obtained by EVP in dMOP1 dominated those obtained by CF for all metrics.

As every metric provides relevant information regarding the solution set obtained by each DPEM_HH version, all are considered equally important. Therefore, to make a fair comparison, the number of times a configuration is superior in a metric on each of the tested DMOPs are added to its respective total. Analysing this table, EVP shows a slight dominance over CF with a total of 23 to 21. This dominance is emphasized when the comparing IGD, which as previously mentioned, denotes the capability of DPEM_HH using EVP to find solutions with good convergence.

Taking the results obtained and the information provided by Table 1, the configuration of DPEM_HH that uses EVP as the LLH selection method is defined as a canonical version of DPEM_HH that will be compared against all three DNSGA–II versions used in this work. For the remainder of this section, DPEM_HH will refer to that setting.

Table 1. Best configuration of DPEM_HH for each DMOP for all metrics and sum of best ranking

Problem	RNI	IGD	MS	HVR	
FDA1	CF	EVP	EVP	CF	
FDA3	EVP	CF	CF	EVP	
FDA4	EVP	EVP	CF	EVP	
FDA5	CF	EVP	CF	EVP	
DMZDT1	CF	CF	CF	CF	
DMZDT2	CF	EVP	EVP	EVP	
DMZDT3	CF	EVP	EVP	CF	
DMZDT4	EVP	EVP	EVP	EVP	
dMOP1	EVP	EVP	EVP	EVP	
dMOP2	CF	CF	CF	CF	
dMOP3	EVP	CF	CF	CF	
Total	EVP: 5	EVP: 7	EVP: 5	EVP: 6	TOTAL: 23
	CF: 6	CF: 4	CF: 6	CF: 5	TOTAL: 21

Note: EVP and CF indicate average population evolvability and choice function used as the LLH selection method, respectively.

The offline mean and standard deviation of IGD, MS and HVR obtained from all tested algorithms are presented in Tables 2–4, respectively. The result obtained by DPEM_HH is compared against all tested DNSGA–II versions run individually for all DMOPs. The best results are highlighted in italic face. The Friedman aligned ranks test (Hodges & Lehmann, 1962) with a 0.05 significance level is applied to check if there exists any significant difference among a group of algorithms for the selected DMOP set. Then, Holm post–hoc procedure (Holm, 1979) is used for pairwise comparisons between DPEM_HH and all tested DNSGA–II versions.

Table 2 presents the results obtained for the IGD metric by each algorithm. DPEM_HH outperform all tested versions of DNSGA–II in several instances by having a smaller IGD value. For all cases, except for FDA3, DPEM_HH presents a better performance than at least one version of DNSGA–II. These results imply that DPEM_HH can find solutions closer to the POF* in comparison to the other algorithms for most cases. In many cases, statistical significance supports these statements. The performance of DPEM_HH in FDA1, DMZDT1, and dMOP2 shows a statistically significant improvement over all the DNSGA–II versions. This table also shows that DPEM_HH is particularly dominant when compared against DNSGA–II–B.

Although cases like FDA3, DMZDT3, and dMOP3 appeared, where DPEM_HH shows a lower performance against a particular version of DNSGA–II. The concept of hyper-heuristics is still maintained as it produces a better result than another DNSGA–II version. These results demonstrate that DPEM_HH can take the strengths of its LLHs to cover their respective weaknesses to produce solution sets with better convergence than the LLHs applied alone. Therefore, DPEM_HH shows its effectiveness regarding this characteristic.

Table 2. Offline mean and standard deviation of IGD metric for DPEM_HH and all DNSGA–II versions

Problem	DPEM_HH	DNSGA–II–A	DNSGA–II–B	DNSGA–II–AB
FDA1	*1.959E–3(4.710E–4)*	5.525E–3(2.120E–3)*	5.151E–3(1.919E–3)*	5.996E–3(2.804E–3)*
FDA3	1.331E–2(1.342E–3)	1.248E–2(9.520E–4)▪	1.269E–2(1.384E–3)▪	*1.208E–2(1.084E–3)*▪
FDA4	1.181E–2(7.560E–4)	*1.179E–2(4.690E–4)*	1.310E–2(8.470E–4)*	1.240E–2(1.011E–3)*
FDA5	*1.091E–2(6.310E–4)*	1.094E–2(7.570E–4)	1.184E–2(6.990E–4)*	1.134E–2(8.310E–4)
DMZDT1	*4.741E–3(8.800E–5)*	4.898E–3(1.420E–4)*	4.879E–3(1.830E–4)*	4.911E–3(1.610E–4)*
DMZDT2	*9.334E–3(1.144E–3)*	9.800E–3(1.261E–3)	9.693E–3(1.339E–3)	9.687E–3(1.104E–3)
DMZDT3	2.285E–2(2.600E–5)	2.285E–2(1.090E–4)	2.285E–2(5.500E–5)	*2.284E–2(1.050E–4)*▪
DMZDT4	3.579E–2(1.964E–2)	*3.294E–2(1.270E–2)*	3.873E–2(2.151E–2)	3.783E–2(1.854E–2)
dMOP1	*6.616E–3(4.683E–3)*	8.353E–3(4.305E–3)	8.669E–3(4.501E–3)	7.603E–3(4.832E–3)
dMOP2	*1.221E–3(1.020E–4)*	1.378E–3(1.540E–4)*	1.341E–3(2.450E–4)*	1.335E–3(1.300E–4)*
dMOP3	6.703E–3(1.984E–3)	*1.667E–3(9.640E–4)*▪	1.024E–2(2.131E–3)*	2.788E–3(1.483E–3)▪

Note: * and ▪ indicate that DPEM_HH is significantly better or worse for that specific DMOEA, respectively.

Table 3 displays the results obtained for the MS metric. DPEM_HH provides results equal or better than the rest of the algorithms for several cases, apart from FDA3, FDA4, FDA5, DMZDT4 and dMOP3. However, aside from FDA4 and dMOP3 the difference is not significant. The results imply that DPEM_HH can obtain solutions sets with equal or better distribution than DNSGA–II. It must be noted that DPEM_HH offers the best values for all DMZDT instances, except for DMZDT4, and offers a significant advantage over all DNSGA–II versions for FDA1, all these DMOPs are type I (non-changing POF*, different optimal solution set). This could mean that DPEM_HH performs especially well for DMOPs of this type. However, if the performance of an LLH is poor, the results of a hyper-heuristic might be affected. This situation arises in FDA3 and dMOP3, where the performance of DNSGA–II–B affects DPEM_HH and, while still capable of obtaining better results than DNSGA–II–B, it cannot outperform one or the rest of the DMOEAs.

Lastly, Table 4 presents the results obtained by DPEM_HH and DNSGA–II for the HVR metric. DPEM_HH outperforms the results obtained of at least one of the versions of DNSGA–II in all DMOPs, except FDA3. DPEM_HH can obtain solution sets with a better relationship between convergence and diversity in comparison to DNSGA–II. For FDA1, DMZDT1, DMZDT2, and dMOP2. DPEM_HH significantly outperforms the results obtained by all the other tested algorithms. Meanwhile, for FDA4, FDA5, DMZDT3, and dMOP3, it shows a statistically significant advantage over some of the DNSGA–II versions. It must be noted that DPEM_HH shows a significant superiority against DNSGA–II–B for most of the tested instances.

Following the same situation presented in Table 2, there are DMOPs where the performance of DPEM_HH is affected by the poor results obtained by one of the LLHs. While DPEM_HH provides better results than the poor–performing LLHs, it is unable to obtain a better result than all versions of DNSGA–II. This situation once again shows that DPEM_HH fulfils the concept of a hyper-heuristic by using the strengths of its LLHs to cover their weakness. Although, there are cases where the low performance of LLHs can affect the result provided by the hyper-heuristic.

The experimental results allow the analysis of the relevance of each proposed element in this paper. As shown in Table 1, the version of DPEM_HH that uses a greedy LLH selection method based on average population evolvability provides better results for most cases in comparison to DPEM_HH using a choice–function based LLH selection method. After comparing this configuration of DPEM_HH with all three versions of DNSGA–II, the results show significant positive results in terms of convergence while keeping or improving the diversity of the solution set.

Type I DMOPs offer the challenge of finding a single POF* while having the optimal solution set change over time. DPEM_HH takes advantage of its properties and the greedy nature of its selection method to keep an intensification–focused search for solutions close to the POF* while keeping good diversity as population evolvability allows the exploration of several neighbour populations during the selection process. These characteristics allow DPEM_HH to obtain better values for all metrics in most cases, with a statistically significant improvement for IGD and HVR, meaning that the obtained POF is closer and better distributed to POF* in comparison to the three DNSGA–II versions.

Table 3. Offline mean and standard deviation of MS metric for DPEM_HH and all DNSGA–II versions

Problem	DPEM_HH	DNSGA–II–A	DNSGA–II–B	DNSGA–II–AB
FDA1	*9.796E–1(1.134E–2)*	9.302E–1(3.865E–2)*	9.346E–1(4.387E–2)*	9.205E–1(5.438E–2)*
FDA3	9.938E–1(2.519E–3)	9.940E–1(3.002E–3)	9.918E–1(1.511E–2)	*9.941E–1(4.420E–3)*
FDA4	1.000E–0(3.000E–6)	*1.000E–0(1.000E–6)*	*1.000E–0(1.000E–6)*	*1.000E–0(1.000E–6)*
FDA5	1.000E–0(3.000E–6)	1.000E–0(1.000E–6)	9.999E–1(2.900E–5)	*1.000E–0(0.000E–0)*
DMZDT1	*9.693E–1(2.304E–3)*	9.681E–1(3.040E–3)	9.679E–1(3.825E–3)	9.675E–1(3.693E–3)*
DMZDT2	*9.049E–1(1.327E–2)*	8.996E–1(1.452E–2)	9.001E–1(1.466E–2)	9.008E–1(1.318E–2)
DMZDT3	*9.103E–1(1.168E–3)*	9.051E–1(1.430E–2)	9.083E–1(8.060E–3)	9.078E–1(1.451E–2)
DMZDT4	8.109E–1(5.226E–2)	8.251E–1(4.425E–2)	*8.278E–1(4.865E–2)*	8.112E–1(4.649E–2)
dMOP1	*9.170E–1(6.169E–2)*	8.885E–1(6.101E–2)	8.818E–1(6.243E–2)	9.001E–1(6.643E–2)
dMOP2	*9.855E–1(3.367E–3)*	9.830E–1(5.359E–3)	9.844E–1(6.070E–3)	9.839E–1(5.919E–3)
dMOP3	8.374E–1(4.451E–2)	*9.740E–1(1.698E–2)*▪	7.694E–1(3.972E–2)*	9.520E–1(2.627E–2)▪

Note: * and ▪ indicate that DPEM_HH is significantly better or worse for that specific DMOEA, respectively.

Table 4. Offline mean and standard deviation of HVR metric for DPEM_HH and all DNSGA–II versions

Problem	DPEM_HH	DNSGA–II–A	DNSGA–II–B	DNSGA–II–AB
FDA1	*9.690E–1(4.508E–3)*	9.015E–1(1.127E–2)*	9.063E–1(9.023E–3)*	9.001E–1(1.321E–2)*
FDA3	7.073E–1(2.866E–2)	7.262E–1(2.043E–2)▪	7.217E–1(3.159E–2)▪	*7.339E–1(2.384E–2)*▪
FDA4	6.291E–1(2.742E–2)	*6.355E–1(1.862E–2)*	5.871E–1(2.958E–2)*	6.176E–1(2.595E–2)
FDA5	6.638E–1(2.752E–2)	*6.678E–1(2.965E–2)*	6.298E–1(3.319E–2)*	6.503E–1(3.324E–2)
DMZDT1	*9.016E–1(1.061E–3)*	8.982E–1(1.406E–3)*	8.987E–1(1.508E–3)*	8.985E–1(1.087E–3)*
DMZDT2	*7.191E–1(9.305E–3)*	7.095E–1(1.041E–2)*	7.116E–1(1.063E–2)*	7.120E–1(8.866E–3)*
DMZDT3	9.337E–1(9.030E–4)	9.321E–1(3.382E–3)*	*9.339E–1(2.420E–3)*	9.331E–1(3.537E–3)
DMZDT4	5.222E–1(1.908E–1)	*5.391E–1(1.580E–1)*	5.131E–1(1.885E–1)	4.864E–1(1.938E–1)
dMOP1	*9.133E–1(4.727E–2)*	8.923E–1(4.509E–2)	8.898E–1(4.720E–2)	9.037E–1(4.886E–2)
dMOP2	*9.642E–1(3.075E–3)*	9.599E–1(3.382E–3)*	9.615E–1(2.961E–3)*	9.616E–1(3.384E–3)*
dMOP3	9.428E–1(1.725E–2)	*9.816E–1(4.870E–3)*▪	9.130E–1(1.964E–2)*	9.767E–1(6.899E–3)▪

Note: * and ▪ indicate that DPEM_HH is significantly better or worse for that specific DMOEA, respectively.

Type II DMOPs challenge algorithms to adapt to the environment changes to find the new POF* and optimal solution set. The greedy strategy of DPEM_HH allowed it to quickly adapt to the DMOP changes by focusing on the search for good–quality solutions based on their aggregated fitness. This lets DPEM_HH get a solution set closer to the POF* quicker than DNSGA–II. DPEM_HH shows a significant increase with respect to convergence. Just as in the case of type I DMOPs, the explorative nature of the population evolvability helped DPEM_HH to keep diversification equal or slightly better when compared to DNSGA–II.

In the case dMOP2, DPEM_HH proved better than all DNSGA–II versions. However, a unique situation raised with FDA3. While DPEM_HH can obtain well–spread populations, which is this instance's main challenge, it provided to be very difficult for DPEM_HH to find solutions with good convergence for this instance. A possible explanation for this is that a greedy approach might not be the most suitable approach for FDA3, as the diversity of the POF* variates after each environmental change.

The dMOP1 problem is a type III DMOP, meaning that it will keep a single optimal solution set but multiple POF*. As the optimal solution set does not change, the greedy nature of DPEM_HH allows it to exploit areas close to its current solution set and produce solutions closer to the current POF*. Also, the results imply that DPEM_HH can obtain sets with slightly better converge and diversity than DNSGA–II for this type of DMOP.

DPEM_HH performs particularly well for DMOPs with convex POF*, implying a better performance in terms of convergence and diversity than DNSGA–II for DMOPs with POF* of this shape. The positive effects of DPEM_HH are seen after studying the results of FDA1, DMZDT1, and dMOP1. FDA4, FDA5, and DMZDT2 have non-convex POF*, for these DMOPs, DPEM_HH takes advantage of the intensification and diversification provided by the greedy method and the population evolvability, respectively, producing populations with a POF closer to the POF*.

While MS shows that DPEM_HH provides no significant improvement over DNSGA–II regarding the diversity of the solution set, HVR shows that there is a clear superiority of the hyper-heuristic, proving a better relationship between convergence and diversity for the obtained solutions. DPEM_HH seems capable of handling DMOPs with a POF* shifting from convex to non-convex, as the outcome for dMOP1 and dMOP2 favours the hyper-heuristic over all DNSGA–II versions and in case of dMOP2, these differences are even statistically significant.

The discontinuous POF* of DMZDT3 proved to be challenging for DPEM_HH in terms of convergence as the results obtained for IGD were lower than those obtained by DNSGA–II. Similar to FDA3, a possible explanation might be the constant change in the diversity of the POF* that leads to DPEM_HH to find well–spread populations but not very close to the POF*.

DMZDT4 presents another problem characteristic to be tested as it is defined as a multimodal DMOP. DPEM_HH handled this challenge well in comparison to the tested DMOEAs. As Tables 2 and 4 show, the results obtained by the presented hyper-heuristic are slightly better than DNSGA–II-B and DNSGA–II–AB for IGD and HVR. This performance allows to believe that the ability of DPEM_HH (or any hyper-heuristic) to use multiple heuristics, or in this case DMOEAs, makes it capable of handling multimodal problems well in terms of convergence.

The dMOP3 is a type I problem that randomly changes the variable that controls the spread of the Pareto front. This tests the adaptability of an algorithm for sudden and random changes. While DNSGA–II–A performs well, the performance of DNSGA–II–B is poor. An issue that hyper-heuristics face is there is a possibility that if one or a subset of the LLHs have a significantly low performance, the rest of the LLHs might be unable to make up for those results, even if they can obtain high–quality results on their own. This situation shows as

DPEM_HH obtains better results than DNSGA–II–B. However, it cannot surpass DNSGA–II–A for all performance metrics. These results reflect the importance of using an adequate LLH set or the necessity to introduce some flexibility to the greedy selection method.

Along with the results obtained by DPEM_HH, the performance of DNSGA–II–AB, a DMOEA proposed in this work to be used as an LLH, is another outcome from these experimentations that should be noted. This algorithm provides better results for all metrics for FDA3, DMZDT2 and, dMOP1 in comparison to the other two DNSGA–II versions. In general, for the tested set of DMOPs this version can produce solutions with a better convergence with respect to DNSGA–II–B and diversity with respect to DNSGA–II–A. A possible explanation could be that merging the change adaptation techniques of both previously proposed DNSGA–II versions into one, allows the new version to find a balance between exploration, by generating new solutions; and exploitation, by mutating current ones.

Conclusions

In this paper, the use of population evolvability in a hyper-heuristic has been proposed. DPEM_HH is presented in this work as a dynamic multi-objective hyper-heuristic that uses this FLA method as part of a low-level heuristic selection method. This work has proposed two LLH selection methods, the application of an average of the population evolvability values obtained by each DMOEA and the combination of this value with a set of performance metric on a choice function.

Three different versions of DNSGA–II were used as LLHs for DPEM_HH. Two configurations of DPEM_HH, each using a different LLH selection method (greedy and choice function) were tested on a set of DMOPs from the FDA, DMZDT and dMOP test suites. Then, a canonical version was selected, and its results were compared against those obtained by every DMOEA used as an LLH run individually by using a set of performance metrics. The outcome of the comparison was mostly positive as the solution sets obtained by DPEM_HH had an overall better quality in most of the instances tested than the solutions obtained by all DNSGA–II versions.

Each proposed DPEM_HH configuration showed to be useful in certain situations, as there were cases where each variation significantly improved the quality of the solutions obtained by our proposed hyper-heuristic even further. These positive results denoted their utility and application to solve DMOPs.

Both proposed LLH selection methods showed to be effective in certain situations. The average population evolvability was able to find populations closer to the POF*. It also performed well on DMOPs with type I DMOPs and problems with shapeshifting POF* in terms of both convergence and diversity. This is due to the greedy nature of this method, which selects solutions that provide the best immediate results, allowing to quickly find solutions closer to POF* and an easier adaptation to problems with irregular changes. Meanwhile, the choice function–based selection method has a better adaptation to problems that show a pattern in their changes as it focuses on the convergence and diversity of a solution set with respect to the POF* as well as the population evolvability. This method selects which LLH is the most appropriate but not necessarily the one that brings the best immediate results.

The combination of the components used on DPEM_HH, such as a greedy selection method and the application of population evolvability, allowed the proposed hyper-heuristic to attempt to find a balance between exploitation and exploration. This is a possible explanation of why the configuration of DPEM_HH that uses a greedy strategy to perform a low-level heuristic selection, obtained the best results for most of the tested DMOP set and was considered as the canonical version during the experimental design.

These results indicate that the application of population evolvability on a hyper-heuristic is feasible, and it can effectively obtain high–quality solutions if this FLA method is used correctly, either used as the only heuristic choosing factor or combined with other elements to form a more complex selection method.

One of the main advantages of hyper-heuristics over other optimization algorithms is their generality, as the variety of their LLH set could allow them to be used in a wider array of problem types in comparison to individual meta-heuristics. Taking said advantage in consideration, DPEM_HH has yet to be tested using other DMOEAs as LLHs and solving DMOPs than were not selected for this paper and bring different challenges such as constrained problems, DMOPs with a changing number of objectives or decision variables, many–objective DMOPs or problems with dependencies between decision variables, just to name a few examples. Another unexplored area is the consideration of preferences and how to insert said preferences into the aggregated fitness value. A possible solution to this would be the use of multiple weight vectors. Also, while this work used a weighted sum method to unify all objective function values, there are other aggregation methods yet to be tested that could obtain better quality solutions. As mentioned in Section 2, there are many different LLH selection methods and move acceptance criteria that have been proposed and have not been tested on a multi-objective or dynamic environment. This is another topic that must be considered, as this factor could change the quality of the solutions obtained. Lastly, while this paper proved the effectiveness of population evolvability within LLH selection methods, there are still several selection methods where this FLA method could be inserted, such as roulette or tabu search. All these observations can be considered as future work.

Funding

This work was supported by the project TecNM 6308.17-P and the following CONACyT projects: Consolidation National Lab under project 280712; Cátedras CONACyT under Project 3058; and CONACyT National Grant System under Grant 465554. B. Dorronsoro acknowledges the Spanish Ministerio de Economía, Industria y Competitividad and ERDF for the support provided under contracts TIN2014-60844-R (SAVANT) and RTI2018-100754-B-I00 (iSUN), and the University of Cadiz (contract PR2018-056).

Author contributions

TME conceived the study and the selection of the elements used in the proposed hyper-heuristic. TME and LCR were the main responsible for the design and development of the proposed hyper-heuristic and its approach. LCR and HFH conducted the data collection and analysis. BD, CGGS and NRV performed the interpretation of the obtained data. BD, along with rest of the authors oversaw the development and review of the first draft of the article.

Disclosure statement

The authors declare that they have no conflicts of interest.

References

Agrawal, R. B., & Deb, K. (1994). Simulated binary crossover for continuous search space. *Complex Systems*, *9*(3), 1–15.

Altenberg, L. (1994). The evolution of evolvability in genetic programming. *Advances in genetic programming*, *3*, 47–74.

Azzouz, R., Bechikh, S., & Said, L. B. (2017a). Dynamic multi-objective optimization using evolutionary algorithms: a survey. In *Recent Advances in Evolutionary Multi-objective Optimization* (pp. 31-70). Cham: Springer. https://doi.org/10.1007/978-3-319-42978-6_2

Azzouz, R., Bechikh, S., & Said, L. B. (2017b). A dynamic multi-objective evolutionary algorithm using a change severity-based adaptive population management strategy. *Soft Computing*, *21*(4), 885-906. https://doi.org/10.1007/s00500-015-1820-4

Ayob, M., & Kendall, G. (2003). A monte carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine. In *Proceedings of the international conference on intelligent technologies* (Vol. 3, pp. 132-141). InTech.

Bai, R., & Kendall, G. (2005). An investigation of automated planograms using a simulated annealing based hyper-heuristic. In *Metaheuristics: Progress as real problem solvers* (pp. 87-108). Boston, MA: Springer. https://doi.org/10.1007/0-387-25383-1_4

Baykasoğlu, A., & Ozsoydan, F. B. (2017). Evolutionary and population–based methods versus constructive search strategies in dynamic combinatorial optimization. *Information Sciences*, *420*, 159-183. https://doi.org/10.1016/j.ins.2017.08.058

Bilgin, B., Özcan, E., & Korkmaz, E. E. (2006, August). An experimental study on hyper-heuristics and exam timetabling. In *International Conference on the Practice and Theory of Automated Timetabling* (pp. 394-412). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-540-77345-0_25

Branke, J. (2001). *Evolutionary optimization in dynamic environments*. Norwell, MA, USA: Kluwer Academic Publishers. https://doi.org/10.1007/978-1-4615-0911-0_2

Brockhoff, D., Friedrich, T., Hebbinghaus, N., Klein, C., Neumann, F., & Zitzler, E. (2007, July). Do additional objectives make a problem harder?. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation* (pp. 765-772). ACM. https://doi.org/10.1145/1276958.1277114

Burke, E. K., & Bykov, Y. (2008, August). A late acceptance strategy in hill-climbing for exam timetabling problems. In *Practice and Theory of Automated Timetabling* (PATAT 2008). Conference, Montreal, Canada. Retrieved from https://www.patatconference.org/patat2008/proceedings/BykovHC2a.pdf

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Woodward, J. R. (2010). A classification of hyper-heuristic approaches. In *Handbook of metaheuristics* (pp. 449-468). Springer US. https://doi.org/10.1007/978-1-4419-1665-5_15

Burke, E. K., Silva, J. D. L., & Soubeiga, E. (2005). Multi-objective hyper-heuristic approaches for space allocation and timetabling. In *Metaheuristics: Progress as Real Problem Solvers* (pp. 129-158). Boston, MA: Springer. https://doi.org/10.1007/0-387-25383-1_6

Chen, R., & Zeng, W. (2011, August). Multi-objective optimization in dynamic environment: A review. *In Computer Science & Education (ICCSE), 2011 6th International Conference on* (pp. 78-82). IEEE. https://doi.org/10.1109/ICCSE.2011.6028589

Cowling, P., Kendall, G., & Soubeiga, E. (2000, August). A hyperheuristic approach to scheduling a sales summit. In *International Conference on the Practice and Theory of Automated Timetabling* (pp. 176-190). Berlin, Heidelberg: Springer. https://doi.org/10.1007/3-540-44629-X_11

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182-197. https://doi.org/10.1109/4235.996017

Deb, K., Rao, N. U. B., & Karthik, S. (2007). Dynamic multi-objective optimization and decision–making using modified NSGA–II: a case study on hydro–thermal power scheduling. In *International Conference on Evolutionary Multi–Criterion Optimization* (pp. 803-817). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-540-70928-2_60

Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2002, May). Scalable multi-objective optimization test problems. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on* (Vol. 1, pp. 825-830). IEEE.

Dowsland, K. A., Soubeiga, E., & Burke, E. (2007). A simulated annealing based hyperheuristic for determining shipper sizes for storage and transportation. *European Journal of Operational Research*, *179*(3), 759-774. https://doi.org/10.1016/j.ejor.2005.03.058

Dueck, G. (1993). New optimization heuristics. *Journal of Computational Physics, 104*(1), 86-92. https://doi.org/10.1006/jcph.1993.1010

Farina, M., Deb, K., & Amato, P. (2004). Dynamic multiobjective optimization problems: test cases, approximations, and applications. *IEEE Transactions on Evolutionary Computation*, *8*(5), 425-442. https://doi.org/10.1109/TEVC.2004.831456

Furtuna, R., Curteanu, S., & Leon, F. (2012). Multi-objective optimization of a stacked neural network using an evolutionary hyper-heuristic. *Applied Soft Computing, 12*(1), 133-144. https://doi.org/10.1016/j.asoc.2011.09.001

García, R. (2010). *Hiper–heurístico para Resolver el Problema de Cartera de Proyectos Sociales* (MSc Thesis). Instituto Tecnológico de Ciudad Madero, Ciudad Madero, Mexico.

Goh, C. K., & Tan, K. C. (2007). An investigation on noisy environments in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, *11*(3), 354-381. https://doi.org/10.1109/TEVC.2006.882428

Goh, C. K., & Tan, K. C. (2009). A competitive–cooperative coevolutionary paradigm for dynamic multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, *13*(1), 103-127. https://doi.org/10.1109/TEVC.2008.920671

Hatzakis, I., & Wallace, D. (2006, July). Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation* (pp. 1201-1208). ACM. https://doi.org/10.1145/1143997.1144187

Helbig, M., & Engelbrecht, A. P. (2014). Benchmarks for dynamic multi-objective optimisation algorithms. *ACM Computing Surveys (CSUR)*, *46*(3), 37. https://doi.org/10.1145/2517649

Hodges Jr, J. L., & Lehmann, E. L. (1962). Rank methods for combination of independent experiments in analysis of variance. *The Annals of Mathematical Statistics*, *33*(2), 482-497. https://doi.org/10.1214/aoms/1177704575

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, *6*(2), 65-70.

Huband, S., Hingston, P., Barone, L., & While, L. (2006). A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation, 10*(5), 477-506. https://doi.org/10.1109/TEVC.2005.861417

Jiang, S., & Yang, S. (2017). A steady-state and generational evolutionary algorithm for dynamic multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, *21*(1), 65-82. https://doi.org/10.1109/TEVC.2016.2574621

Kumari, A. C., Srinivas, K., & Gupta, M. P. (2013, February). Software module clustering using a hyper-heuristic based multi-objective genetic algorithm. In *Advance Computing Conference (IACC), 2013 IEEE 3rd International* (pp. 813-818). IEEE. https://doi.org/10.1109/IAdCC.2013.6514331

Li, H., & Deb, K. (2017, June). Challenges for evolutionary multiobjective optimization algorithms in solving variable–length problems. In *Evolutionary Computation (CEC), 2017 IEEE Congress on* (pp. 2217–2224). IEEE. https://doi.org/10.1109/CEC.2017.7969573

Li, W., Özcan, E., & John, R. (2017). Multi-objective evolutionary algorithms and hyper-heuristics for wind farm layout optimisation. *Renewable Energy, 105*, 473-482. https://doi.org/10.1016/j.renene.2016.12.022

Liao, X., Li, Q., Yang, X., Zhang, W., & Li, W. (2008). Multiobjective optimization for crash safety design of vehicles using stepwise regression model. *Structural and multidisciplinary optimization, 35*(6), 561-569. https://doi.org/10.1007/s00158-007-0163-x

Liu, C. A. (2010, June). New dynamic multiobjective evolutionary algorithm with core estimation of distribution. In *2010 International Conference on Electrical and Control Engineering* (pp. 1345-1348). IEEE. https://doi.org/10.1109/icece.2010.334

Liu, C. A., & Wang, Y. (2006, September). New evolutionary algorithm for dynamic multiobjective optimization problems. *In International Conference on Natural Computation* (pp. 889-892). Berlin, Heidelberg: Springer. https://doi.org/10.1007/11881070_117

Maashi, M., Özcan, E., & Kendall, G. (2014). A multi-objective hyper-heuristic based on choice function. *Expert Systems with Applications*, *41*(9), 4475-4493. https://doi.org/10.1016/j.eswa.2013.12.050

Maashi, M., Kendall, G., & Özcan, E. (2015). Choice function based hyper-heuristics for multi-objective optimization. *Applied Soft Computing*, *28*, 312-326. https://doi.org/10.1016/j.asoc.2014.12.012

McClymont, K., & Keedwell, E. C. (2011, July). Markov chain hyper-heuristic (MCHH): an online selective hyper-heuristic for multi-objective continuous problems. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation* (pp. 2003-2010). ACM. https://doi.org/10.1145/2001576.2001845

Nguyen, S., Zhang, M., Johnston, M., & Tan, K. C. (2014). Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. *IEEE Transactions on Evolutionary Computation*, *18*(2), 193–208. https://doi.org/10.1109/TEVC.2013.2248159

Richter, H. (2013). Dynamic fitness landscape analysis. In *Evolutionary computation for dynamic optimization problems* (pp. 269-297). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-38416-5_11

Roy, S., & Chakraborty, U. (2013). *Introduction to soft computing: neuro-fuzzy and genetic algorithms*. Dorling Kindersley.

Santiago, A., Dorronsoro, B., Nebro, A. J., Durillo, J. J., Castillo, O., & Fraire, H. J. (2019). A novel multi-objective evolutionary algorithm with fuzzy logic based adaptive selection of operators: FAME. *Information Sciences*, *471*, 233-251. https://doi.org/10.1016/j.ins.2018.09.005

Sierra, M. R., & Coello, C. A. C. (2005, March). Improving PSO–based multi-objective optimization using crowding, mutation and∈– dominance. In *International Conference on Evolutionary Multi–Criterion Optimization* (pp. 505–519). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-540-31880-4_35

Smith, T., Husbands, P., & O'Shea, M. (2002). Fitness landscapes and evolvability. *Evolutionary computation*, *10*(1), 1-34. https://doi.org/10.1162/106365602317301754

Soria-Alcaraz, J. A., Espinal, A., & Sotelo-Figueroa, M. A. (2017). Evolvability metric estimation by a parallel perceptron for on-line selection hyper-heuristics. *IEEE Access, 5*, 7055-7063. https://doi.org/10.1109/ACCESS.2017.2699426

Soria-Alcaraz, J. A., Ochoa, G., Sotelo-Figeroa, M. A., & Burke, E. K. (2017). A methodology for determining an effective subset of heuristics in selection hyper-heuristics. *European Journal of Operational Research, 260*(3), 972-983. https://doi.org/10.1016/j.ejor.2017.01.042

Tan, K. C., Lee, T. H., & Khor, E. F. (2002). Evolutionary algorithms for multi-objective optimization: Performance assessments and comparisons. *Artificial Intelligence Review*, *17*(4), 251-290. https://doi.org/10.1023/A:1015516501242

Topcuoglu, H. R., Ucar, A., & Altin, L. (2014). A hyper-heuristic based framework for dynamic optimization problems. *Applied Soft Computing*, *19*, 236-251. https://doi.org/10.1016/j.asoc.2014.01.037

Van Veldhuizen, D. A. (1999). *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations* (PhD thesis). Air Force Institute of Technology, Alabama, USA. https://doi.org/10.1145/298151.298382

Vázquez-Rodríguez, J. A., & Petrovic, S. (2012). Calibrating continuous multi-objective heuristics using mixture experiments. *Journal of Heuristics, 18*(5), 699-726. https://doi.org/10.1007/s10732-012-9204-8

Vrugt, J. A., & Robinson, B. A. (2007). Improved evolutionary optimization from genetically adaptive multimethod search. *Proceedings of the National Academy of Sciences, 104*(3), 708-711. https://doi.org/10.1073/pnas.0610471104

Wang, Y., & Li, B. (2009, May). Investigation of memory-based multi-objective optimization evolutionary algorithm in dynamic environment. *In Evolutionary Computation, 2009. CEC'09. IEEE Congress on* (pp. 630-637). IEEE. https://doi.org/10.1109/CEC.2009.4983004

Wang, Y., & Li, B. (2010). Multi–strategy ensemble evolutionary algorithm for dynamic multi-objective optimization. *Memetic Computing*, *2*(1), 3-24. https://doi.org/10.1007/s12293-009-0012-0

Wang, M., Li, B., Zhang, G., & Yao, X. (2017). Population Evolvability: Dynamic Fitness Landscape Analysis for Population–based Metaheuristic Algorithms. *IEEE Transactions on Evolutionary Computation*. https://doi.org/10.1109/TEVC.2017.2744324

Wang, H., Wang, D., & Yang, S. (2009). A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems. *Soft Computing, 13*(8-9), 763-780. https://doi.org/10.1007/s00500-008-0347-3

Zamli, K. Z., Din, F., Kendall, G., & Ahmed, B. S. (2017). An experimental study of hyper-heuristic selection and acceptance mechanism for combinatorial t–way test suite generation. *Information Sciences*, *399*, 121-153. https://doi.org/10.1016/j.ins.2017.03.007

Zhang, Q., Zhou, A., & Jin, Y. (2008). RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm. *IEEE Transactions on Evolutionary Computation*, *12*(1), 41-63. https://doi.org/10.1109/TEVC.2007.894202

Zhou, A., Jin, Y., & Zhang, Q. (2014). A population prediction strategy for evolutionary dynamic multiobjective optimization. *IEEE transactions on cybernetics*, *44*(1), 40-53. https://doi.org/10.1109/TCYB.2013.2245892