J Glob Optim (2014) 58:769–794
DOI 10.1007/s10898-013-0088-4

Memetic algorithms and hyperheuristics applied to a multiobjectivised two-dimensional packing problem

Eduardo Segredo · Carlos Segura · Coromoto León

Received: 13 September 2012 / Accepted: 6 July 2013 / Published online: 17 July 2013

Abstract Packing problems are NP-hard problems with several practical applications. A variant of a 2D Packing Problem (2DPP) was proposed in the GECCO 2008 competition session. In this paper, Memetic Algorithms (MAs) and Hyperheuristics are applied to a multiobjectivised version of the 2DPP. Multiobjectivisation is the reformulation of a mono-objective problem into a multi-objective one. The main aim of multiobjectivising the 2DPP is to avoid stagnation in local optima. First generation MAs refers to hybrid algorithms that combine a population-based global search with an individual learning process. A novel first generation MA is proposed, and an original multiobjectivisation method is applied to the 2DPP. In addition, with the aim of facilitating the application of such first generation MAs from the point of view of the parameter setting, and of enabling their usage in parallel environments, a parallel hyperheuristic is also applied. Particularly, the method applied here is a hybrid approach which combines a parallel island-based model and a hyperheuristic. The main objective of this work is twofold. Firstly, to analyse the advantages and drawbacks of a set of first generation MAs. Secondly, to attempt to avoid those drawbacks by applying a parallel hyperheuristic. Moreover, robustness and scalability analyses of the parallel scheme are included. Finally, we should note that our methods improve on the current best-known solutions for the tested instances of the 2DPP.

Keywords Memetic algorithms · Hyperheuristics · Multiobjectivisation · Packing problems · Parameter setting

E. Segredo (✉)· C. Segura · C. León
Dpto. Estadística, I. O. y Computación, Universidad de La Laguna, Avda. Astrofísico Francisco Sánchez, s/n, 38271 La Laguna, Santa Cruz de Tenerife, Spain
e-mail: esegredo@ull.es

C. Segura
e-mail: csegura@ull.es

C. León
e-mail: cleon@ull.es

1 Introduction

Packing problems are a class of optimisation problems which involve packing a set of objects together as densely as possible. They are highly related to cutting problems, whose main goal is to cut large stock sheets into a set of smaller pieces. In many cases, both problems have been analysed together, being referred to as cutting and packing problems. Both problems have been shown to be combinatorial NP-hard problems. Therefore, obtaining high quality solutions is a complex task. However, there is a high interest in solving them because they are related to real life packaging, storage and transportation issues. Therefore, they have many applications and are widely used within more complex systems, e.g. filling containers and trucks, loading pallets, optimising the layout of electrical circuits, etc.

Cutting and packing problems can be classified [29,40] according to several characteristics: the number of dimensions, the number of available patterns, the shape of the patterns—regular or irregular—, the orientation, and the objective to be optimised, among others. Depending on these features, several variants of the problem can be defined. Some of the most popular ones are 2D strip packing, constrained 2D cutting stock, knapsack problems, and packing with cost. Within each category there are also several different formulations. In the GECCO 2008 competition session[1] a variant of a 2D packing problem (2DPP) was proposed. It was a reformulation of a packing problem designed with the aim of hindering the achievement of optimal values and increasing the size of the search space. While it may be difficult to imagine direct practical applications of this particular variant of a 2D packing problem, it is hard and complex enough that it can be used to check the advantages and drawbacks of a given optimisation scheme. This was our main reason for considering this formulation of the problem in our work. Moreover, previous results obtained using different optimisation methods can be used for comparison purposes [27,35].

Several optimisation algorithms have been defined to deal with complex optimisation problems. Among them, several flavours of metaheuristics have been defined. One of the most promising is the family of Memetic Algorithms (MAs) [33]. They are a synergy of Evolutionary Algorithms (EAs) or any population-based approach with a separate individual learning process. Considering the classification exposed in [32], MAs can be categorised as follows. The first generation MAs refers to hybrid algorithms that combine a population-based global search with an individual learning process. The second generation MAs include hyperheuristics among other approaches. In this case, different low-level configurations of a metaheuristic—*memes*—are chosen by the hyperheuristic based on their behaviour to generate local improvements through a reward mechanism. Finally, in third generation MAs the pool of memes is dynamically generated during the optimisation process instead of being specified beforehand. In this paper, we consider first and second generation MAs. In order to test the second generation MAs, a hyperheuristic is considered.

The incorporation of the learning process might lead to stagnation in local optima [34,35]. Several methods for avoiding stagnation have been proposed [18]. Some of the simplest techniques rely on restarting the approach when stagnation is detected. In other cases, a component is used that inserts randomness or noise into the search. Another possibility is the usage of *multiobjectivisation* [24]. The term multiobjectivisation refers to the redefinition of originally mono-objective problems as multi-objective ones. Multiobjectivisation usually decreases the selection pressure of the original approach. Therefore, when used in combination with MAs, some low quality individuals in the population have a higher probability of

[1] http://www.isgec.org/gecco-2008/competitions.html

surviving. However, if properly configured, these individuals could, in the long term, help to avoid stagnation, thus resulting in high quality solutions.

Another drawback of MAs is that the time invested in obtaining high quality solutions could be considerable. With the aim of reducing this time, several paradigms for implementing parallel EAs (pEAs) have been proposed [2]. These paradigms can be extended to MAs parallelisation (pMAs) by substituting EAs by MAs. Island-based models have shown good performance and scalability in many areas [2]. Such models conceptually divide the overall pMA population into a number of independent and separate populations, i.e. there are separate and simultaneously executing MAs—one per processor or island—. Each island evolves in isolation for the majority of the pMA execution, but occasionally, some individuals can be migrated among neighbour islands using a predefined migration stage.

The main objective of this work is twofold. Firstly, to analyse the benefits and disadvantages of a set of first generation MAs. Such a set of first generation MAs is applied to a multiobjectivised version of the 2DPP. Among the proposals, the ones presented in [35] were considered since they yielded the best up-to-date results for the 2DPP. In addition, a novel MA is proposed and an original multiobjectivisation scheme is applied to the 2DPP. We study the effects of the MA applied and of the multiobjectivisation approach utilised on the quality of the solutions obtained. Secondly, in order to avoid the drawbacks of the first generation MAs and to enable their usage in parallel environments, a parallel hyperheuristic is applied. This parallel scheme has the added benefit of facilitating the application of the first generation MAs from the point of view of the parameter setting. It is a hybrid approach which combines a parallel island-based model with a hyperheuristic. In this case, the pool of memes consists of the aforementioned set of first generation MAs. An analysis of the robustness of such a parallel hyperheuristic is performed. The aim is to study the effect caused by different migration stages on the quality of the solutions obtained. In addition, a scalability analysis of this parallel approach is performed. Particularly, we consider the effect that the migration stage has on the scalability of the parallel hyperheuristic. Finally, we should note that our method improves on the current best-known solutions for the tested instances of the 2DPP.

The rest of this paper is organised as follows: In Sect. 2, the background of packing problems, and specifically of the 2DPP, is given. Afterwards, the formal definition of the 2DPP is detailed in Sect. 3. In Sect. 4, the first generation MAs applied to the 2DPP are described. The approaches used to multiobjectivise such a problem are also explained at this point. The parallel hyperheuristic is defined in Sect. 5. Then, the experimental evaluation and the results obtained are presented in Sect. 6. Finally, the conclusions and some lines of future work are given in Sect. 7.

2 Background of packing problems

There are many proposals designed to deal with packing problems. Among them, several exact approaches have been proposed [30]. Usually, the time associated with such algorithms is very large. Therefore, in order to reduce the execution time, some parallel exact approaches have also been designed [4,26]. However, since packing problems usually involve a large search space, exact approaches are practically unaffordable for many real-world instances.

In order to handle large instances, a wide variety of approximation algorithms have been tested. For instance, an approach based on Ant Colony Optimisation (ACO) was used to deal with a multi-objective version of a packing problem in [25], while in [31] a Genetic Algorithm (GA) was used for a mono-objective problem. MAs have also yielded very promising results for packing problems [41].

Regarding the 2DPP defined in the GECCO 2008 competition session, several proposals have also been tested. During the contest, the two best-behaved approaches were based on MAs. Prior to the proposals defined in this paper, the approach which had obtained the best results for the competition session instance was a MA that was able to dynamically change its population size (VARPOP). This approach incorporated an individual learning process specifically designed to deal with the 2DPP. A parallel hybrid model that combines the VARPOP algorithm and hyperheuristics was proposed in [27] in an effort to obtain high quality results faster. Although high quality solutions were obtained for the contest instance, subsequent studies [35] concluded that stagnation in local optima may appear for other ones. So as to avoid such drawbacks, a parallel homogeneous island-based model was applied to a multiobjectivised version of the 2DPP in [35]. This approach was able to find high quality results for those instances in which the approaches based on the VARPOP failed. However, for other instances, the time required by the multiobjectivised parallel model was larger than the time invested by the parallel approach based on the VARPOP algorithm. In addition, it is important to remark that the most suitable multiobjectivisation method depended on the instance to be solved. Therefore, in order to obtain high quality solutions, several multiobjectivisation approaches had to be tested as part of that research.

3 Formal definition of the 2DPP

The problem considered is a variant of a 2D packing problem. It was proposed in the GECCO 2008 competition session. Since the problem has been tackled using many different approaches and its search space is vast, it can be used as a benchmark problem. Problem instances are described by the following data:

- The sizes of a rectangular grid: X, Y.
- The maximum number which can be assigned to a grid position: M. The value assigned to each grid location is an integer in the range $[0, M]$.
- The score or value associated with the appearance of each pair (a, b) where $a, b \in [0, M]$: $v(a, b)$. Note that $v(a, b)$ is not necessarily equal to $v(b, a)$.

A candidate solution is obtained by assigning a number to each grid position. Thus, the search space consists of $(M + 1)^{X \cdot Y}$ candidate solutions. The objective of the problem proposed is to best pack a grid so that the sum of the point scores for every pair of adjacent numbers is maximised. Two positions are considered to be adjacent if they are neighbours in the same row, column, or diagonal of the grid. Once a particular pair is collected, it cannot be collected a second time in the same grid.

Mathematically, the problem objective is to find the grid G which maximises the objective function f:

$$f = \sum_{a=0}^{M} \sum_{b=0}^{M} v_2(a, b) \tag{1}$$

where

$$v_2(a, b) = \begin{cases} 0 & \text{if } (a, b) \text{ are not adjacent in } G \\ v(a, b) & \text{if } (a, b) \text{ are adjacent in } G \end{cases} \tag{2}$$

Figure 1 illustrates the objective function assignment for a candidate solution of a 2×2 grid. Note that although the pairs $(1, 2)$ and $(2, 1)$ are repeated in the grid, they are only considered once when computing the objective value.

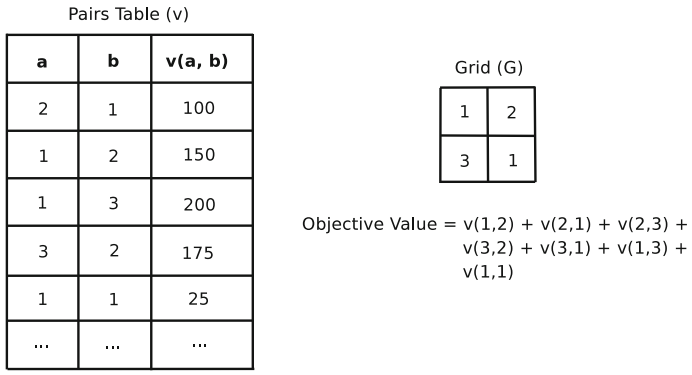

Fig. 1 Assignment of the original objective function for the 2DPP

4 First generation multiobjectivised memetic algorithms for the 2DPP

In this section, the first generation MAs applied in this paper are described. Since the 2DPP has been multiobjectivised using several approaches, the usage of multi-objective algorithms is required; specifically, two different MAs which start from two different Multi-Objective Evolutionary Algorithms (MOEAs) has been used. Finally, we consider a tailor-made learning process for the 2DPP.

4.1 Multi-objective algorithms

In this paper two different MAs are used that are based on two of the most prominent MOEAs: the Non-dominated Sorting Genetic Algorithm II (NSGA- II) [14], and the Strength Pareto Evolutionary Algorithm 2 (SPEA2) [42]. Both MAs incorporate the tailor-made learning process for the 2DPP presented in Sect. 4.2. Although the MA based on the NSGA- II was considered in a previous work [35], our usage of the SPEA2 is an original contribution.

The MA based on the NSGA- II (Algorithm 1) uses a fast non-dominated sorting approach with reduced computational complexity. In addition, it applies a selection operator which combines previous populations with new generated ones, ensuring elitism in the approach. The fast non-dominated approach and the selection operator require the definition of a partial order of the individuals. The *crowded comparison operator* ($\geq_n$) is used to establish such an order. This operator assigns two different attributes to every individual i of the population: the *non-domination rank* (i_{rank}) and the *local crowding distance* ($i_{distance}$).

The non-domination rank makes use of the Pareto Dominance concept. The procedure to calculate it is as follows. First, the set of non-dominated individuals in the population are assigned to the first rank. Then the process is repeated considering only the individuals that do not have a rank assigned. The rank assigned at each step is increased by one. The process ends when every individual in the population has its corresponding rank established.

The local crowding distance is used to estimate the density of solutions surrounding a particular individual. First, the size of the largest cuboid enclosing the individual i without including any other individual that belongs to its rank is calculated. Then, the crowding distance is given by the mean side-length of the cuboid. Finally, the partial order given by the crowded comparison operator $\geq_n$ is the following:

Algorithm 1 MA based on NSGA- II

1: **Initialisation**: Generate an initial population P_0 with N individuals. Assign $t = 0$.
2: **while** (not stopping criterion) **do**
3: **Evaluation**: Evaluate all individuals in the population.
4: **Fitness assignment**: Calculate fitness values of individuals in P_t. Use the non-domination rank in the first generation, and the crowded comparison operator in other generations.
5: **Mating selection**: Perform binary tournament selection on P_t in order to fill the mating pool.
6: **Variation**: Apply genetic operators to the mating pool to create a child population CP.
7: **Learning process**: Perform individual learning processes for every individual in the population.
8: **Survivor selection**: Combine P_t and CP, selecting the best individuals using the crowded comparison operator to constitute P_{t+1}.
9: $t = t + 1$
10: **end while**

Algorithm 2 MA based on SPEA2

1: **Initialisation**: Generate an initial population P_0 with N individuals, and create the empty archive $\overline{P}_0$. Assign $t = 0$.
2: **while** (not stopping criterion) **do**
3: **Evaluation**: Evaluate all individuals in the population.
4: **Fitness assignment**: Calculate the fitness values of individuals in P_t and $\overline{P}_t$. For each individual i, calculate the raw fitness i_{raw} and the density estimate $i_{density}$.
5: **Environmental Selection**: Copy non-dominated individuals in P_t and $\overline{P}_t$ to $\overline{P}_{t+1}$. If $|\overline{P}_{t+1}| > N$ reduce $\overline{P}_{t+1}$. Otherwise, fill $\overline{P}_{t+1}$ with dominated individuals in P_t and P_{t+1}, considering their fitness.
6: **Mating selection**: Perform binary tournament selection on $\overline{P}_{t+1}$ to fill the mating pool.
7: **Variation**: Apply genetic operators to the mating pool and set P_{t+1} to the resulting population.
8: **Learning process**: Perform individual learning processes for every individual in P_{t+1}.
9: $t = t + 1$
10: **end while**

$$i \geq_n j \; if \begin{cases} (i_{rank} < j_{rank}) \\ or \\ ((i_{rank} = j_{rank}) \; and \; (i_{distance} > j_{distance})) \end{cases} \tag{3}$$

The MA based on the SPEA2 (Algorithm 2) establishes an order among the individuals using a fine-grained fitness assignment strategy. The fitness value of each individual, which has to be minimised, is calculated as the sum of the raw fitness of the individual plus a density estimate. The density information ($i_{density}$) is incorporated to discriminate among individuals with identical raw fitness values. In order to calculate the raw fitness, the strength $i_{strength}$ of each individual i is calculated as the number of solutions that it dominates, considering the population and the archive:

$$i_{strength} = |\{j | j \in P_t + \overline{P}_t \wedge i \succ j\}| \tag{4}$$

Then, the raw fitness i_{raw} is calculated as follows:

$$i_{raw} = \sum_{j \in P_t + \overline{P}_t, j \succ i} j_{strength} \tag{5}$$

In order to complete the definition of the MAs considered, other components must be specified. The parent selection operator is the well-known *Binary Tournament* [17]. Finally, individuals are encoded as two-dimensional arrays of integer values G, where $G(x, y)$ is the number assigned to the grid position (x, y).

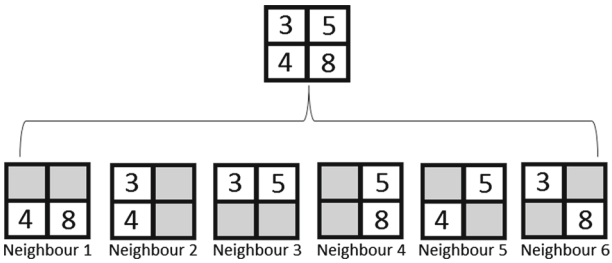

Fig. 2 Generation of neighbours by the learning process

4.2 Learning process for the 2DPP

Multi-objective MAs usually incorporate the usage of a multi-objective learning process [22]. However, since the 2DPP is multiobjectivised in this paper, the learning process only considers the original objective function. The process applied can be classified as Lamarckian learning, i.e. the genotype reflects the learning process improvements. It is based on a mono-objective stochastic hill-climbing local search. The local search strategy applied [27] has the following features. For each pair of adjacent grid positions (i, j) and (k, l), a neighbour is considered. This is illustrated in Fig. 2. Each neighbour is constituted by assigning the best possible values to the positions (i, j) and (k, l)—shading positions in Fig. 2—, leaving intact the assignments in any other grid location. In order to assign the best values to both locations, the trivial solution consists of enumerating all possible pairs so that the best one can subsequently be chosen. This approach is computationally too expensive, so a mechanism is used to prune the values explored. First, all the possible assignments $n \in [0, M]$ to the grid position (i, j) are considered, and the contribution of each assignment $v_{ij}(n)$, assuming position (k, l) is unassigned, is calculated. The same process is performed for position (k, l), assuming position (i, j) is unassigned, which thus calculates $v_{kl}(n)$. The contribution to the objective function obtained by assigning a value a to position (i, j), and a value b to position (k, l), is given by:

$$v_{ij}(a) + v_{kl}(b) + v'(a, b) - v_{rep} \tag{6}$$

where $v'(a, b) = v(a, b) + v(b, a)$ if the pair (a, b) was not already in the grid, or 0 if it was, and v_{rep} is the value associated with pairs that are constituted by both the assignment of the value a to (i, j) and the assignment of the value b to (k, l), which must be considered only once. An upper bound for such a contribution is given by:

$$v_{ij}(a) + v_{kl}(b) + min(bestV(a), bestV(b)) \tag{7}$$

where $bestV(n)$ is the maximum value associated with any pair $(n, m), m \in [0, M]$, i.e.:

$$\max\{v(n, m) + v(m, n)\}. \tag{8}$$

If $bestObj$ is the best objective value currently achieved for an assignment of the positions (i, j) and (k, l), the only values a', b' that must be considered are those that satisfy the following inequality:

$$v_{ij}(a') + v_{kl}(b') + min(bestV(a'), bestV(b')) > bestObj \tag{9}$$

Omitting the values at which the previous inequality is not satisfied reduces the neighbourhood to be considered considerably, resulting in significant time savings.

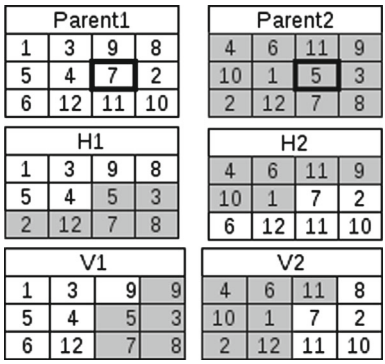

Fig. 3 Operation of the sub-string crossover (SSX)

Since stochastic hill-climbing is used, the order in which neighbours are analysed is determined randomly. The local search moves to the first new generated neighbour that improves the current solution. Finally, the learning process stops when none of the neighbours improves the current solution.

4.3 Genetic operators

A mutation and a crossover operator are applied during the variation stage of the MAs. They are applied with probabilities p_m and p_c, respectively. Several variation operators were tested in [27]. The best-behaved ones were selected for our work. The crossover operator was the *2D Sub-String Crossover* (SSX) [20]. First, a grid cell is selected as the division point. Then, the operator randomly decides to do a vertical or horizontal crossover. SSX is described in Fig. 3. $H1$ and $H2$ are generated by means of a horizontal crossover, while $V1$ and $V2$ are generated by the application of the vertical one. In both cases, the cell $(3, 2)$ is selected as the division point.

The applied mutation operator was the *Uniform Mutation with Domain Information* (UMD). Each gene is mutated with a probability between min_p_m and max_p_m. In order to make new assignments to the gene, a random value is selected from among those that produce a non-zero increase in the objective value.

4.4 Multiobjectivisation approaches

The term *multiobjectivisation* was introduced in [24] to refer to the reformulation of originally mono-objective problems as multi-objective ones. There are two main ways to multiobjectivise a problem. The first one decomposes the original objective into several sub-objectives. The Pareto Front of the new definition should contain a solution with the original optimal value. The second one is based on aggregating an auxiliary function as the second objective. This function is used together with the original objective function. Therefore, the Pareto Front always contains a solution with the original optimal value. The main advantage of the second approach is that it can take into account problem-independent information. Thus, general multiobjectivisation approaches useful for several optimisation problems may be designed. In this paper, we consider problem-dependent and problem-independent multiobjectivisations techniques based on aggregation.

Multiobjectivisation usually decreases the selection pressure. Therefore, when used together with MAs, some low quality individuals in the population have a higher probability

of survival. However, if properly configured, in the long term these individuals could help to avoid stagnation in local optima, thus yielding higher quality solutions.

Several options have been proposed to define the artificial objective function [1,5,37]. Some of these are based on the Euclidean distance in the decision space [37]. These functions are a direct measure of the diversity. The following are considered in this work:

- DCN: The distance to the closest individual has to be maximised.
- ADI: The average distance to all individuals has to be maximised.
- DBI: The distance to the best individual, i.e. the one with the highest original objective value, has to be maximised.

In the case of the NSGA- II, the aforementioned functions are calculated using the individuals in the population. When the SPEA2 is applied, however, the individuals in the population and the archive are taken into account.

Other authors propose the usage of objectives that are able to preserve diversity without using a direct measure for it [1]. Among them, the following are used in this work:

- Random: A random value is assigned as the second objective to be minimised.
- Inversion: In this case, the optimisation direction of the original objective function is inverted and used as the artificial objective.

In addition, two artificial functions based on the DBI and the DCN multiobjectivisations, which try to avoid the survival of individuals with a very low quality, are also applied. They incorporate the use of a threshold ratio ($th \in [0, 1]$) to be specified by the user. These multiobjectivisations are named DBI- THR and DCN- THR, respectively. The DCN- THR approach has never before been used to multiobjectivise the 2DPP. Being *bestObjectiveValue* the original objective value of the best individual considered by the artificial function, the threshold value (v) is defined as:

$$v = bestObjectiveValue \cdot th \tag{10}$$

The alternative objective of individuals whose original objective value is lower than v is assigned to 0. Consequently, individuals that are not able to achieve the fixed threshold are penalised. In the special case where $th = 0$, individuals are never penalised. Thus, DBI- THR and DCN- THR with $th = 0$ behave as the functions DBI and DCN, respectively. In the case where $th = 1$, the MAs behave as if multiobjectivisation had not been applied.

Figure 4 (left-hand side) shows the behaviour of the DBI and the DCN functions when they are integrated with the MAs presented in Sect. 4.1. The maximisation of the original and

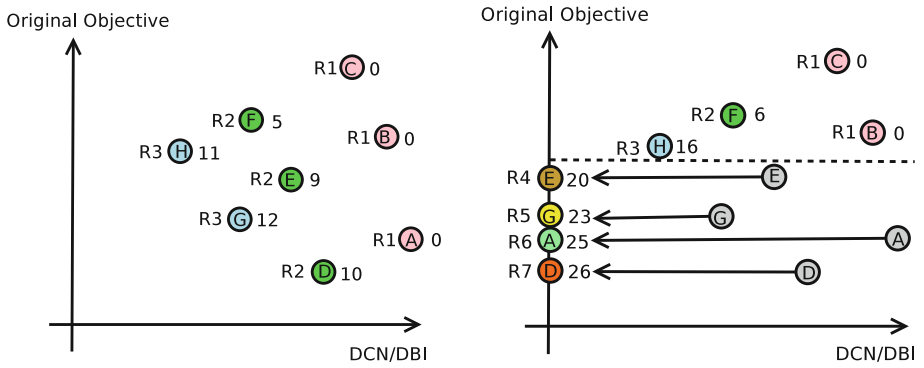

Fig. 4 Behaviour of the MAs without threshold (*left*-hand side) and with threshold (*right*-hand side)

the artificial objective functions are assumed. Note that every candidate solution is tagged with a label that indicates its corresponding ranking assigned by the MA based on the NSGA-II (left side of every solution). Thus, the label Ri means that the corresponding candidate solution belongs to the rank number i. Moreover, the corresponding raw fitness assigned by the MA based on the SPEA2 is also shown on the right-hand side of every candidate solution. The right-hand side of Fig. 4 shows the effect of incorporating the threshold to the DBI and DCN functions. The broken line represents the value of v. We can see that every candidate solution which does not fulfil the minimum quality level established by the threshold ratio is shifted in the objective space. Specifically, a value equal to 0 is assigned to the corresponding alternative objective. The effect of the shift is that the corresponding candidate solution will usually belong to a worse rank in the case of the MA based on the NSGA- II. In the case of the MA based on the SPEA2, a higher raw fitness will be assigned to the corresponding candidate solution. Therefore, the survival probability of the candidate solution will usually decrease.

Finally, a multiobjectivisation which considers problem-dependent information (Dependent) was also tested. In order to calculate the second objective, the original 2DPP objective function (f) is decomposed into two separate functions, f_0 and f_1, so that $f = f_0 + f_1$. The decomposition is performed as follows. First, a table containing all possible pairs whose score is not equal to zero is calculated. Then, this table is sorted based on the score of the appearance of each pair ρ. The resultant position of each ρ, after the sort, is denoted as i_ρ. The value associated with each ρ is taken into account to calculate the function f_{obj}, where $obj = i_\rho \bmod 2$. Finally, f_0 is used as the additional objective. Likewise, f_1 could have been used as the auxiliary objective function.

5 A parallel hyperheuristic for the 2DPP

This section describes the parallel hyperheuristic applied in this paper. It is a hybrid approach that combines a parallel island-based model with a choice-based hyperheuristic.

5.1 Island-based models

In island-based models, the population is divided into a number of independent subpopulations. Each subpopulation is associated with an island and a configuration of a population-based metaheuristic or meme is executed on each island. In this paper, several configurations of the MAs depicted in Sect. 4 are considered as memes. Usually, each available processor constitutes an island which evolves in isolation for the majority of the parallel run. However, collaborative schemes could lead to better behaviour. Therefore, a migration stage that enables the exchange of individuals among islands is generally incorporated.

There are four basic island-based models [10]: all islands execute identical configurations (homogeneous), different configurations are executed on the islands (heterogeneous), each island evaluates different objective function subsets, and each island represents a different region of the genotype or phenotype domains.

Collaboration among islands is handled by means of a migration mechanism. A well designed migration stage can ensure a successful collaboration, meaning the solution search space is better explored. However, if an unsuitable migration stage is introduced in the model, the effect could be similar to, or even worse than, having separate MAs simultaneously executing on several processors with no communication among them. Therefore, the migration stage must be carefully defined. To configure the migration stage, the migration topology—where to migrate the individuals—, and the migration rate—how many individuals are migrated and

how often—must be established. In addition, those individuals that are going to be migrated and those that are going to be replaced must be selected. This selection is performed by the migration and replacement schemes, respectively.

When applying island-based models, landscapes may be completely different from those produced by the corresponding sequential MA. As a result, the island-based model may find better or equivalent solutions in less time. Depending on the migration stage selected, the landscape is affected in different ways [38]. Consequently, in this paper we test several migration stages.

5.2 Hyperheuristics

A hyperheuristic can be viewed as a heuristic that iteratively chooses among a set of given low-level metaheuristics in order to solve an optimisation problem [6]. Hyperheuristics operate at a higher level of abstraction than heuristics because they have no knowledge of the problem domain. Once a hyperheuristic algorithm is developed, several problem domains and instances can be tackled by just replacing the low-level metaheuristics. Thus, the aim in using a hyperheuristic is "raising the level of generality" [6] at which most current metaheuristics operate. Since the main motivation behind hyperheuristics is to design problem-independent strategies, a hyperheuristic is not concerned with solving a given problem directly, as is the case with most heuristics implementations. In fact, the search is performed in a search space of metaheuristics rather than a search space of potential problem solutions. The hyperheuristic solves the problem indirectly by applying the appropriate solving method at each stage of the optimisation process. Generally, "the goal of raising the level of generality" [6] is achieved at the expense of reduced—but still acceptable—solution quality when hyperheuristics are compared to tailor-made metaheuristic approaches.

A diagram of a general hyperheuristic framework [6] is given in Fig. 5. It shows a problem domain barrier between the low-level metaheuristics and the hyperheuristic itself. The data flow obtained by the hyperheuristic can include the quality of the solutions—mean, improvement, best, worst—, the resources—time, processors, memory—invested in obtaining such solutions, etc. The hyperheuristic makes its decisions based on this information. The data flow coming from the hyperheuristic can include information about which metaheuristic is to be executed, its parameters, stopping criteria, etc.

Several ways for incorporating the ideas of hyperheuristics into an optimisation problem have been proposed. Hyperheuristics which deal with mono-objective optimisation problems

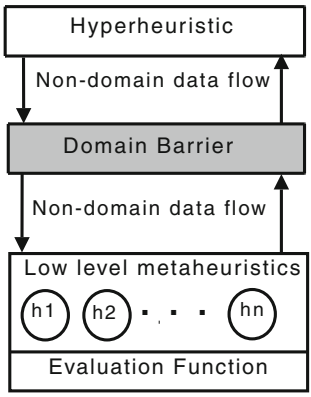

Fig. 5 Levels and dataflow of a hyperheuristic framework

are more widespread. A hyperheuristic based on a tabu search was presented in [8]. The same hyperheuristic was used inside a simulated annealing algorithm [16]. Other metaheuristics which have inspired the creation of hyperheuristics include GAs [11] and ACO [7,9]. Local search with restart [3] has also been used to implement hyperheuristics. Finally, the choice functions have been used multiple times [12,13,23]. In these cases, a scoring function is used to assess the performance of each low-level metaheuristic. All of the resources are allocated to the approach which maximises this function. In [39], a choice function is also used to score each method. However, the resources are assigned using a probability function, which is based on the assigned score. In this paper, we use a hyperheuristic based on the one presented in [39].

5.3 Dynamic-mapped island-based model

In this paper, the dynamic-mapped island-based model (DYN) presented in [28] is used together with different configurations of the MAs exposed in Sect. 4 as low-level approaches. It is a hybrid approach that combines a parallel island-based model and a hyperheuristic based on the one presented in [39]. The architecture of the dynamic-mapped model is similar to the parallel island-based model, i.e. it consists of a set of *worker islands* that evolve in isolation by applying a certain low-level configuration to a given population. In addition, as in the island-based model, a tuneable migration stage allows for the exchange of individuals among neighbouring islands. In this paper we incorporate several migration stages into the dynamic-mapped island-based model. Specifically, four different migration stages, obtained by combining two different migration topologies with two different replacement schemes, are tested.

The migration topologies considered are as follows. The first one is an *all to all connected topology* (ALL). In this topology each island connects with and sends its individuals to all of the remaining ones. The second one is a *unidirectional ring topology* (RING). In such a topology each island connects to exactly two other islands, constituting a logical ring. Considering that there are n_p islands, labelled from 0 to $n_p - 1$, each island γ sends its individuals to island $(\gamma+1) \bmod n_p$, and receives individuals from island $(\gamma+n_p-1) \bmod n_p$. The four migration stages rely on an *elitist migration scheme*. Specifically, a subpopulation individual is migrated when its original objective value is higher than the original objective value of any member of its previous generations. The migration rate is implicitly defined by the aforementioned migration scheme. Finally, two different replacement schemes are employed. The first one is a novel proposal: the *elitist Hamming-based replacement scheme* (HAM). Firstly, it checks whether or not the immigrant has an original objective value higher than all the individuals of the destination island. If so, the immigrant replaces the individual which has the lowest Hamming distance to it, considering the decision space. Otherwise, the immigrant is discarded. In order to validate the results obtained by the proposal, the *elitist ranking scheme* (ELI) [38] is also applied. It ranks all Pareto fronts and replaces an individual from the worst ranked front with the immigrant. This scheme was specifically designed for the multi-objective field and it provides a high selection pressure. The different migration stages are identified by means of the nomenclature *Topology-Replacement_Scheme*. For example, the migration stage which uses the unidirectional ring topology and the elitist Hamming-based replacement scheme is referred to as RING-HAM.

In the standard island-based model there exists a static mapping among the islands and configurations, i.e. each island executes the same configuration over the course of the complete run. In a homogeneous island-based model, there is only one configuration that is executed by every worker island. In a heterogeneous island-based model, the configurations executed by

worker islands are different. However, in the model in question, a dynamic mapping among the islands and configurations is applied. Thus, the configurations executed in each island over the course of the run can vary. This mapping is performed using a hyperheuristic. In order to manage the dynamic mapping, i.e. to apply the hyperheuristic, a new special island, called the *master island*, is introduced into the scheme. In order to implement it, two kinds of stopping criteria are defined. First, a global stopping criterion is established. When this global stopping criterion is reached, every worker island sends its local solution to the master and the run ends. Moreover, a local stopping criterion is fixed for the execution of the configurations on the worker islands. When a local stopping criterion is reached, the island's execution is stopped. The local results are then sent to the master island. At this point, the master island applies the hyperheuristic in order to decide which low-level configuration is going to be applied in the idle island. This configuration is applied by taking as the initial population the final population obtained by the previous configuration.

The hyperheuristic selected (HH_imp) is based on using a scoring strategy and a selection strategy for picking the low-level configuration to be executed. The selection of the low-level configuration is as follows. First, the scoring strategy assigns a score to each low-level configuration. This score estimates the improvement that each low-level metaheuristic or configuration can achieve when it starts from the currently obtained solutions. In order to perform this estimate, the previous improvements on the original objective value achieved by each configuration are used. The improvement (*imp*) is defined as the difference, in terms of the original objective value, between the best achieved individual and the best initial individual. Considering a configuration $conf$, which has been executed j times, the score ($s(conf)$) is calculated as a weighted average of its latest k improvements (Eq. 11). In such an equation, $imp[a][b]$ represents the improvement achieved by configuration a in execution number b. Depending on the value of k, the adaptation level of the hyperheuristic, i.e. the total amount of historical knowledge that the hyperheuristic considers in order to perform its decisions, can be set. The weighted average assigns a greater importance to the latest executions.

$$s(conf) = \frac{\sum_{i=1}^{min(k,j)} (min(k, j) + 1 - i) \cdot imp[conf][j - i]}{\sum_{i=1}^{min(k,j)} i} \tag{11}$$

The stochastic behaviour of the low-level metaheuristics involved may lead to variations in the results they obtain. Therefore, it is appropriate to make some selections based on a random scheme. The hyperheuristic can be tuned by means of the parameter β, which represents the minimum selection probability that should be assigned to a low-level configuration. If n_h is the number of low-level configurations involved, a random selection following a uniform distribution is performed in $\beta \cdot n_h$ percentage of the cases. Therefore, the probability of selecting each configuration $conf$ is given by:

$$prob(conf) = \beta + (1 - \beta \cdot n_h) \cdot \left[\frac{s(conf)}{\sum_{i=1}^{n_h} s(i)} \right] \tag{12}$$

6 Experimental evaluation

In this section, the experiments performed with the first generation MAs described in Sect. 4 and with the parallel hyperheuristic exposed in Sect. 5 are described. The 2DPP is multi-objectivised by the addition of different auxiliary objective functions. However, since the

auxiliary function does not represent any practical information—it is only used to preserve the diversity of the population—showing the data for this second objective is of no use. As a result, only data of the 2DPP original objective function is considered when presenting the computational results.

The sequential and parallel optimisation schemes are implemented using the Metaheuristic-based Extensible Tool for Cooperative Optimisation (METCO) [28]. Tests were run on HECToR [19], the UK's National Supercomputing Service. The Phase 3 (Cray XE6) system of HECToR is contained in 30 cabinets and comprises of a total of 704 compute blades. Each blade contains four compute nodes, each with two 16-core AMD Opteron 2.3 GHZ Interlagos processors. This amounts to a total of 90,112 cores, offering a theoretical peak performance of over 800 Tflops. Each 16-core socket is coupled with a Cray Gemini routing and communications chip. Finally, each 16-core processor shares 16 Gb of memory.

Communications among different islands of the DYN model were implemented asynchronously using the Message Passing Interface (MPI) library [36]. We opted to use MPI for two reasons. Firstly, it works on distributed and shared memory architectures. And secondly, since executions of the DYN model with up to 128 cores were carried out, the HECToR architecture was not suitable for an implementation of the DYN model with OpenMP. The MPI library version was Cray MPICH2 5.6.0, while the compiler was GCC 4.7.2.

Analyses were performed considering two different instances of the 2DPP. The first one is characterised by the following parameters: $X = 10$, $Y = 10$, $M = 99$, and 9032 possible pair scores. The second one is the instance proposed in the competition session. Its parameters are the following: $X = 20$, $Y = 20$, $M = 399$, and 1,5962 possible pair scores.

Since we are dealing with stochastic algorithms, each execution was repeated 24 times. Each experiment was carried out for both instances. So as to be able to present our results with confidence, we ran comparisons applying the following statistical analysis [15]. First, we carried out a *Shapiro-Wilk test* in order to check whether the values of the results follow a normal (Gaussian) distribution or not. If so, the *Levene test* was used to check for the homogeneity of the variances. If samples had equal variance, an ANOVA *test* was done. Otherwise, a *Welch test* was performed. For non-Gaussian distributions, the non-parametric *Kruskal–Wallis* test was used to compare the medians of the algorithms. A significance level of 5 % was considered.

6.1 Analysis of the first generation memetic algorithms

The objective of the first experiment was to analyse the behaviour of the first generation MAs from the point of view of robustness. Particularly, we studied whether the quality of the solutions depends on the MA applied and/or on the multiobjectivisation approach considered. We also conducted additional analyses with the aim of determining whether the most suitable approach depends on the instance of the 2DPP considered. To do so, we defined 16 different configurations of the first generation MAs. They were obtained by combining the two different MAs exposed in Sect. 4, with the 8 multiobjectivisation schemes described in Sect. 4.4. In every configuration, the population and the archive sizes were fixed to 10 individuals. For the multiobjectivisation approaches that incorporate the usage of a threshold value, this value was set to $th = 0.99$. Both the UMD and the SSX operators were applied in every generation, i.e. the probabilities p_m and p_c were set to 1 for every configuration. In addition, the UMD operator used the following parameterisation: $min\_p_m = 0.1$ and $max\_p_m = 0.15$. In the case of the first instance, the configurations were executed considering a stopping criterion of 5 h, while for the second one a stopping criterion of 11.5 h was set.

Table 1 Original objective function for the first generation MAs—first instance

Name	MOEA	Multiobj.	Mean	Median	Max
SEQ1	SPEA2	DBI	5.130×10^8	5.134×10^8	5.152×10^8
SEQ2	SPEA2	DBI-THR	5.129×10^8	5.129×10^8	5.157×10^8
SEQ3	NSGA2	ADI	5.126×10^8	5.125×10^8	5.152×10^8
SEQ4	NSGA2	DCN	5.124×10^8	5.127×10^8	5.142×10^8
SEQ5	SPEA2	DCN	5.120×10^8	5.121×10^8	5.145×10^8
SEQ6	SPEA2	DCN-THR	5.120×10^8	5.118×10^8	5.137×10^8
SEQ7	NSGA2	DBI-THR	5.118×10^8	5.119×10^8	5.143×10^8
SEQ8	NSGA2	DCN-THR	5.118×10^8	5.115×10^8	5.146×10^8
SEQ9	SPEA2	ADI	5.117×10^8	5.112×10^8	5.144×10^8
SEQ10	NSGA2	DBI	5.117×10^8	5.119×10^8	5.139×10^8
SEQ11	NSGA2	Dependent	5.105×10^8	5.102×10^8	5.149×10^8
SEQ12	SPEA2	Dependent	5.104×10^8	5.105×10^8	5.133×10^8
SEQ13	NSGA2	Random	5.103×10^8	5.103×10^8	5.131×10^8
SEQ14	SPEA2	Random	5.099×10^8	5.097×10^8	5.126×10^8
SEQ15	NSGA2	Inversion	5.095×10^8	5.093×10^8	5.127×10^8
SEQ16	SPEA2	Inversion	5.095×10^8	5.097×10^8	5.117×10^8

Table 1 shows, for the first instance and for each MA tested, the mean, the median, and the maximum of the original objective values attained. The configurations of the MAs were sorted in terms of the mean original objective value. An index based on this order was assigned to each configuration. Thus, the first configuration, i.e. the one which achieved the highest mean of the original objective value, is referred to as SEQ1, while the last one is referred to as SEQ16. The differences among the configurations are noticeable and reveal the importance of correctly selecting the appropriate one. In fact, the differences among SEQ1 and the remaining configurations are statistically significant, except for the configurations SEQ2–SEQ4. Statistical tests also confirmed that both the multiobjectivisation approach and the MA used affected the quality of the solutions. For instance, SEQ1 was significantly different from SEQ10. Such configurations are based on the same multiobjectivisation approach, but they consider a different MA. Therefore, properly selecting the MA affects the quality of the solutions. Similarly, SEQ1 and SEQ5, which are both based on the SPEA2, are statistically different. Since they only differ in the multiobjectivisation approach used, the importance of properly selecting this component has also been demonstrated. Finally, the incorporation of a threshold value in the multiobjectivisation approaches tested did not affect the quality of the results. The configurations that used a multiobjectivisation approach with threshold were not statistically different from their non-threshold counterparts.

Table 2 shows the same information for the second instance. In this case, differences among the configurations considered are also noticeable. The results obtained by SEQ1 are statistically different from those obtained by the other configurations, apart from the SEQ2 and SEQ3 configurations. In addition, changing the MA used does not yield significant differences in the results. For example, the differences between SEQ1 and SEQ2 are not statistically significant. In this case, both configurations applied the same multiobjectivisation method but used different MAs. This happened for every pair of configurations in which the multiobjectivisation approach applied was the same and the MA applied was different. However,

Table 2 Original objective function for the first generation MAs—second instance

Name	MOEA	Multiobj.	Mean	Median	Max
SEQ1	NSGA2	DCN-THR	1.008×10^9	1.007×10^9	1.017×10^9
SEQ2	SPEA2	DCN-THR	1.007×10^9	1.006×10^9	1.015×10^9
SEQ3	SPEA2	DBI-THR	1.006×10^9	1.007×10^9	1.015×10^9
SEQ4	NSGA2	DBI-THR	1.005×10^9	1.005×10^9	1.015×10^9
SEQ5	SPEA2	DCN	1.004×10^9	1.005×10^9	1.013×10^9
SEQ6	NSGA2	DCN	1.004×10^9	1.005×10^9	1.015×10^9
SEQ7	NSGA2	ADI	1.004×10^9	1.002×10^9	1.015×10^9
SEQ8	SPEA2	ADI	1.002×10^9	1.002×10^9	1.013×10^9
SEQ9	NSGA2	DBI	1.001×10^9	1.001×10^9	1.011×10^9
SEQ10	SPEA2	DBI	9.997×10^8	9.992×10^8	1.009×10^9
SEQ11	SPEA2	Random	9.961×10^8	9.995×10^8	1.004×10^9
SEQ12	NSGA2	Random	9.950×10^8	9.945×10^8	1.004×10^9
SEQ13	NSGA2	Dependent	9.946×10^8	9.952×10^8	1.001×10^9
SEQ14	SPEA2	Dependent	9.946×10^8	9.956×10^8	1.001×10^9
SEQ15	SPEA2	Inversion	9.864×10^8	9.861×10^8	9.938×10^8
SEQ16	NSGA2	Inversion	9.851×10^8	9.850×10^8	9.923×10^8

the multiobjectivisation approach applied did affect the quality of the solutions. For instance, SEQ1 is statistically different from SEQ4, and they apply the DCN- THR and the DBI- THR multiobjectivisations, respectively. Finally, configurations applying a multiobjectivisation approach with threshold are statistically different from their non-threshold counterparts.

Considering both tested instances, the most suitable configurations of the MAs are different. For example, the configuration SEQ1 for the first instance is the configuration SEQ10 for the second instance. Similarly, the configuration SEQ1 for the second instance is the configuration SEQ8 for the first one. Thus, the most suitable configurations depend on the features of the instance in question, resulting in some robustness problems. Given a new instance, it is difficult to predict which configuration will provide the best results. In addition, if the number of configurations considered is very large, testing each one of them might not be feasible. Therefore, the application of a hyperheuristic seems very promising. Finally, since the sequential models do not converge even after a very long period of time, the usage of parallel models also seems a promising approach.

6.2 Analysis of the parallel hyperheuristic

The aim of the second experiment was to avoid the robustness problems of the first generation MAs by considering a parallel hyperheuristic (DYN). We also analysed the behaviour of the DYN model with respect to the migration stage used. The model was executed with the four migration stages described in Sect. 5.3. A total number of $n_p = 4$ islands was considered. The global stopping criterion was set to 5 h for the first instance and 11.5 h for the second one. For both instances, the local stopping criterion was set to 10 minutes. The HH_imp hyperheuristic of the DYN model was applied with an adaptation level $k = \infty$, and the value of β was set in such a way that a 10 % of the decisions performed by the hyperheuristic followed a uniform distribution, i.e. $\beta \cdot n_h = 0.1$. Finally, the $n_h = 16$ configurations of the first generation MAs

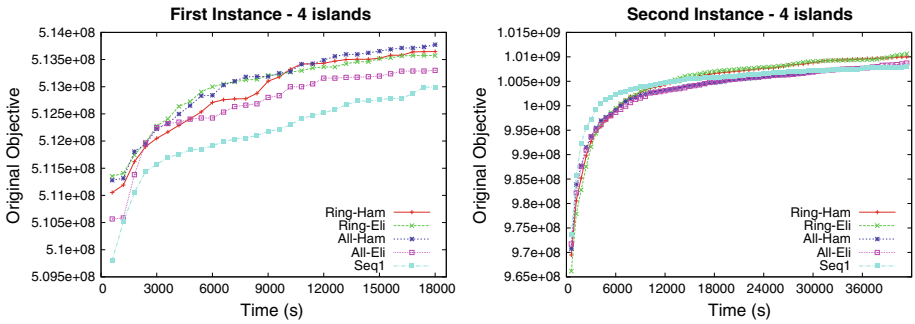

Fig. 6 Evolution of the mean of the original objective function resulting from the DYN model with 4 worker islands for the first and second instances

applied in the previous experiment (Sect. 6.1) were used for the low-level configurations or memes.

Figure 6 shows, for the first and second instances, the evolution of the mean of the original objective value for the DYN model with the four migration stages. In order to compare the results obtained by the parallel models, also shown are the data of the best sequential configuration (SEQ1) for each considered instance. For both instances, the parallel models were able to achieve a higher mean of the original objective value than the corresponding best sequential approach. Moreover, the differences among the parallel model which yielded the highest mean objective value and the best sequential approach for each instance considered were statistically significant. In the case of the first instance, the parallel approach relied on the ALL-HAM migration stage, while the second one it used the RING-ELI migration stage. Consequently, depending on the features of the instance in question, the most suitable migration stage must be properly selected. The box plots of the DYN model with the different migration stages (Fig. 7) confirm the aforementioned conclusions.

The DYN model avoided the need to check for the most suitable configuration of an algorithm for a given instance. The quality of the solutions obtained from the parallel model applying the best migration stage was higher than the quality obtained by the best sequential approach. Thus, high quality solutions can be achieved by a single execution of this parallel model, resulting in lower use of computational resources. This process thus mitigates the robustness problems of the first generation MAs. Lastly, the DYN model facilitates the application of the first generation MAs from the point of view of the parameter setting, and enabled their usage in parallel environments.

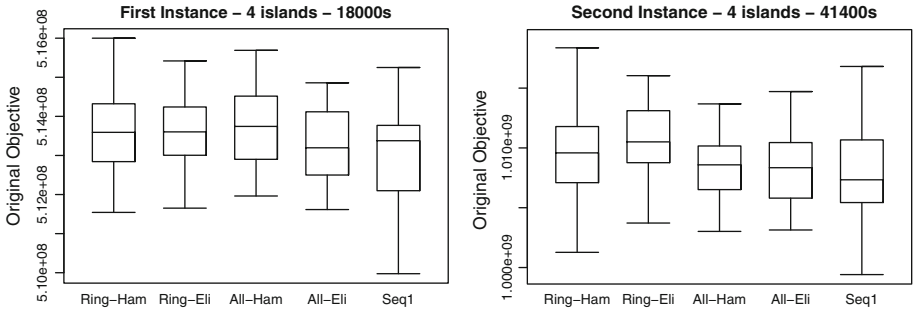

Fig. 7 *Box plots* of the DYN model with 4 islands for the first and second instances

Table 3 Statistical tests of the DYN model—16 islands–5 h–first instance

	Ring-Ham	Ring-Eli	All-Ham	All-Eli
Ring-Ham	↔	↔	↑	↔
Ring-Eli	↔	↔	↑	↔
All-Ham	↓	↓	↔	↔
All-Eli	↔	↔	↔	↔

Table 4 Statistical tests of the DYN model—32 islands–5 h–first instance

	Ring-Ham	Ring-Eli	All-Ham	All-Eli
Ring-Ham	↔	↔	↑	↔
Ring-Eli	↔	↔	↔	↔
All-Ham	↓	↔	↔	↔
All-Eli	↔	↔	↔	↔

The objective of the third experiment was to conduct robustness and scalability analyses of the DYN model. Specifically, we studied the relationship between the effect caused by the change in the migration stage and the number of islands. In this case, the DYN model was executed with the same parameterisation as in the previous experiment, but using a total number of 8, 16, and 32 worker islands (n_p).

Considering the first instance, statistical differences among the different migration stages were not significant when the DYN model was applied with 4 and 8 worker islands. With 16 and 32 islands, however, statistical differences among the migration stages did appear. This means that the importance of properly selecting the migration stage rises with the number of worker islands. Tables 3 and 4 show the statistical significances for the different migration stages considering 16 and 32 worker islands, respectively. Every cell shows whether the row model is statistically better (↑), not different (↔), or worse(↓) than the corresponding column model.

In the case of the second instance, Table 5 shows the results of the statistical tests for the different migration stages considering 4 worker islands. Similarly, Table 6 shows the same information when 8, 16, and 32 islands were used. The amount of significant statistical

Table 5 Statistical tests of the DYN model—4 islands–11.5 h–second instance

	Ring-Ham	Ring-Eli	All-Ham	All-Eli
Ring-Ham	↔	↔	↔	↔
Ring-Eli	↔	↔	↑	↔
All-Ham	↔	↓	↔	↔
All-Eli	↔	↔	↔	↔

Table 6 Statistical tests of the DYN model—8, 16, 32 islands–11.5 h–second instance

	Ring-Ham	Ring-Eli	All-Ham	All-Eli
Ring-Ham	↔	↔	↑	↑
Ring-Eli	↔	↔	↑	↑
All-Ham	↓	↓	↔	↔
All-Eli	↓	↓	↔	↔

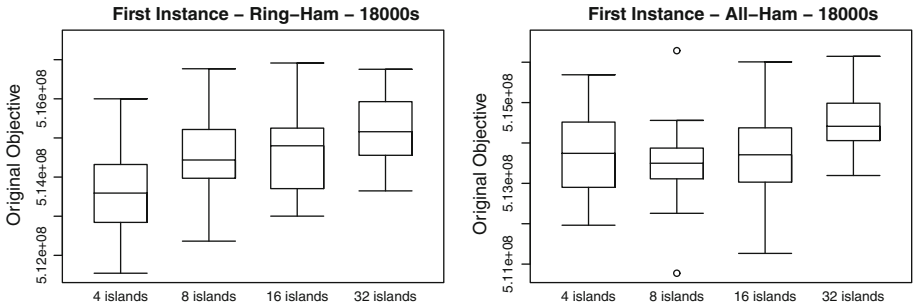

Fig. 8 *Box plots* of the DYN model with the best and worst migration stages for the first instance

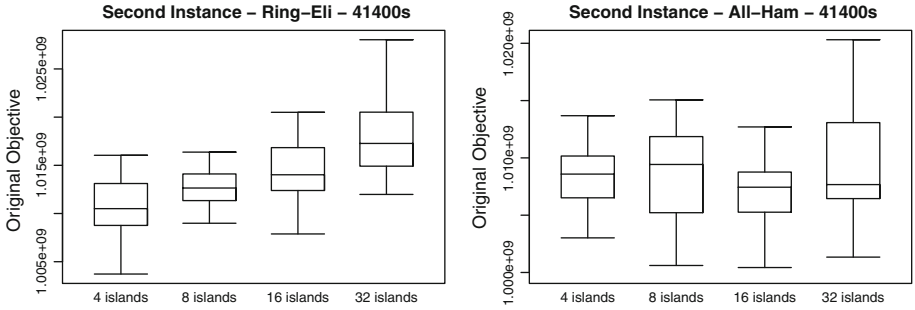

Fig. 9 *Box plots* of the DYN model with the best and worst migration stages for the second instance

differences was larger when 8, 16, and 32 islands were applied. As in the first instance, the importance of selecting the appropriate migration stage increases as a higher number of worker islands is used.

In order to better quantify the importance of selecting the appropriate migration stage for the DYN model, we conducted another analysis. Considering the mean of the original objective value achieved by the parallel models with 32 islands, the best and worst were selected for each instance. Figure 8 shows, for the first instance, the box plots of the best and worst parallel models when they were run with up to 32 islands. The same information is shown in Fig. 9 for the second instance. For both cases, the trend towards obtaining better objective values as the number of islands increases is clear when the best migration stage is considered. However, this did not happen when considering the worst stage.

The above analysis compared different parallel models in terms of the quality achieved at fixed times. However, it is important to quantify the improvement achieved by such parallel approaches in terms of the amount of time saved. To do so, we conducted an additional study that relied on *Run-Length Distributions* (RLD) [21]. RLDs show the relationship between success ratios and time. The success ratio of a particular approach is defined as its probability of achieving a certain quality level.

Figure 10 shows, for the first instance, the RLDs of the best and worst parallel models with up to 32 worker islands. It also includes the RLDs of the sequential configurations SEQ1 and SEQ3 to compare the results obtained by the parallel models. The same information is shown in Fig. 11 for the second instance. In order to calculate the RLDs for both instances, the quality level was set as the median of the original objective value achieved by the configuration SEQ3. In the case of the first instance, the parallel models that used the best migration stage clearly

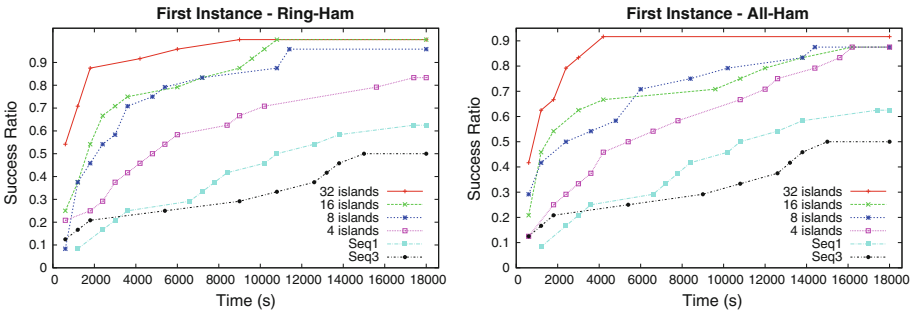

Fig. 10 RLDs of the DYN model with the best and worst migration stages for the first instance

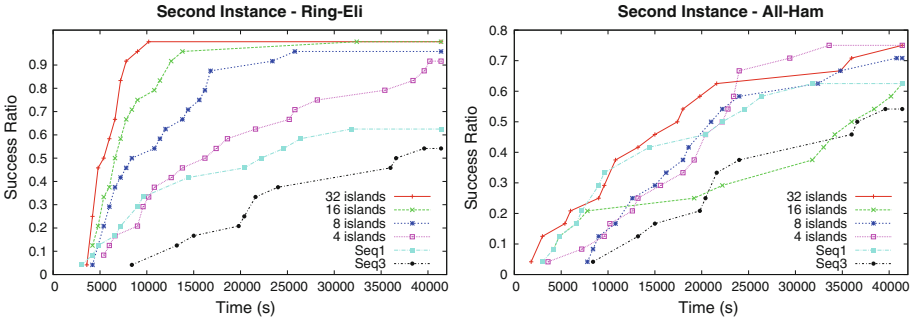

Fig. 11 RLDs of the DYN model with the best and worst migration stages for the second instance

outperformed the sequential configurations, obtaining the same or higher success ratios in less time. In addition, not only were high quality solutions yielded by the best parallel model, but they were obtained in less time when the number of worker islands increased. For example, the best parallel models with 16 and 32 worker islands were able to achieve a 100 % success ratio, i.e. every execution reached the set quality level, although the best parallel model with 32 islands obtained this success ratio in less time. The solutions yielded by the parallel models that used the worst-behaved migration stage were of a lower quality than those ones obtained with the best parallel models. Moreover, none of the worst parallel models was able to reach a 100 % success ratio.

In the case of the second instance, the same conclusions as for the first instance can be extracted for the parallel model using the best migration stage. The behaviour of the parallel models when using the worst migration stage, however, was poor. For example, the parallel model with 4 worker islands obtained a given value for the success ratio in less time than the same model using 32 islands. Summarising, for both instances, selecting the appropriate migration stage does not only affect the quality of the solutions, but also the total amount of time and processors required to achieve said quality level.

In order to quantify the effects that the migration stage has on the scalability of the DYN model, speedup factors were calculated using the data provided by the RLDs. Table 7 shows the resulting speedup factors obtained for SEQ1 by the DYN model applied to the first instance with the best and worst migration stages. In order to calculate these factors the following steps were performed. Firstly, given a model with n_p worker islands, a relative speedup factor ($spr_{[n_p]}$) was calculated for the model with $n_p \div 2$ worker islands. In the case of the parallel models with $n_p = 4$ islands, the best sequential configuration (SEQ1) was used as reference.

Table 7 Speedup factors of the DYN model with the best and worst migration stages—first instance

	Best	Worst
4 islands	1.71	1.61
8 islands	3.98	1.07
16 islands	6.29	1.21
32 islands	15.73	7.26

Table 8 Speedup factors of the DYN model with the best and worst migration stages—second instance

	Best	Worst
4 islands	2.3	1.64
8 islands	4.10	2.57
16 islands	5.70	1.36
32 islands	18.81	2.90

For this instance, and for each relative speedup factor, the quality level was set as the lowest median of the original objective value achieved in 5 h by both of the models considered. The relative speedup is calculated by dividing the time invested by the model using a lower number of processors by the time invested by the model using a higher amount of processors. These times were obtained by considering a 50 % success ratio. Once the relative speedup factors were calculated, the resulting speedup factor for the model with n_p processors ($sp_{[n_p]}$) was calculated as follows:

$$sp_{[n_p]} = \begin{cases} spr_{[n_p]} \cdot sp_{[n_p \div 2]} & if\ n_p \neq 4 \\ spr_{[4]} & if\ n_p = 4 \end{cases} \quad (13)$$

The resulting speedup factors for the second instance are shown in Table 8. The aforementioned procedure was used to calculate the factors by setting the time equal to 11.5 h.

For both instances, the speedup factors increased when the parallel model applied had a larger amount of worker islands. For example, in the case of the first instance, the best parallel model with 16 worker islands obtained a speedup factor equal to 6.29, while the same model considering 32 islands achieved a speedup factor equal to 15.73. In this case, the relative speedup factor calculated for both models was greater than one. This means that the model with 32 islands achieved the set quality level in 50 % of the executions in less time than the model with 16 worker islands. However, this was not the case when the corresponding worst parallel model was applied to each instance. For example, in the case of the second instance, the worst parallel model with 8 worker islands obtained a speedup factor equal to 2.57, while the worst parallel model with 16 worker islands yielded a speedup factor equal to 1.36. In this case, the relative speedup factor calculated for both models was lower than one. This means that the model with a lower number of worker islands attained the defined quality level in 50 % of the executions in less time than the model which considered a higher number of islands. Therefore, incorporating a larger amount of processors to the worst-behaved parallel model for each instance in question did not provide good results.

In the fourth experiment, the DYN model using the best-behaved migration stage for each instance was executed using 64 and 128 worker islands, the goal being to study the scalability of the model with a large number of processors. The parameterisation of the DYN model was the same as in previous experiments. However, the global stopping criterion was set to 2 h due to the availability of the computational resources.

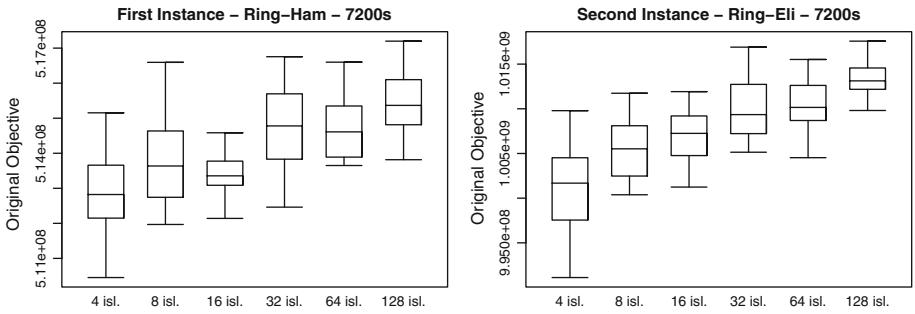

Fig. 12 *Box plots* of the DYN model with the best migration stage for the first and second instances

Table 9 Speedup factors of the DYN model with the best migration stage for both instances

	64 islands	128 islands
First instance	12.58	41.9
Second instance	20.51	25.02

Figure 12 shows the box plots of the DYN model with up to 128 worker islands, applying the corresponding best-behaved migration stage for each instance. Even with a large number of worker islands, the quality of the solutions obtained by the DYN model kept increasing as more resources were considered. In general, the larger the number of worker islands, the higher the quality of the solutions obtained. Table 9 shows the speedup factors for SEQ1 of the parallel model using the best migration stage for each instance, applying 64 and 128 worker islands. Speedup factors were obtained following the same procedure used in the previous experiment. In order to obtain the relative speedup factors for both instances, the quality level was set at the lowest median of the original objective value obtained by the two models in question in 2 h. The calculated speedup factors confirm the benefits of adding a larger amount of processors. The benefits were more noticeable in the case of the first instance. Finally, we should note that using the DYN model avoids the requirement of independently executing each meme. Therefore, the total amount of time that can be saved is greater than the one shown by the calculated speedup factors.

The executions with $n_p = 128$ islands were able to provide better results than the best previously known solution for the first instance [35]. The previous best solution was obtained by a multiobjectivised homogeneous parallel island-based model. It was applied considering 4 islands and 12 h of execution. In order to apply this parallel model, the best low-level configuration or meme for a given instance has to be identified, requiring many computational experiments beforehand. A larger amount of resources was invested in our research to improve on the best previous solution. However, this improvement was achieved in only 2 h. Specifically, the original objective value was raised from 516, 202, 152 to 517, 199, 441. Moreover, the use of a larger number of islands was offset by the reduced time and computational resources needed by the DYN model, since the prior analysis to identify the best meme was not necessary.

Since the second instance is harder than the first one, larger executions were required to improve on the best-known solution for this test case. The previous best solution was obtained by combining a mono-objective parallel island-based model with hyperheuristics [27]. Such a model was applied considering 4 islands and 6 h of execution. For this test case it was demonstrated that multiobjectivisation did not provide benefits in the short term [35].

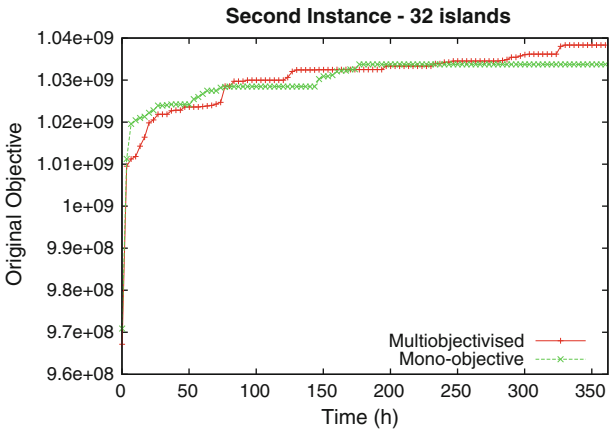

Fig. 13 Evolution of the original objective function with and without multiobjectivisation in the long term

However, it would be interesting to determine whether multiobjectivisation can avoid long-term stagnation. The DYN model was executed with 32 worker islands and considering the same parameterisation as the one applied in the previous experiments. The RING-ELI migration stage was used, while the global stopping criterion was set to 15 days. Due to availability of resources, a larger amount of processors was not considered and only one execution was performed. The parallel approach that obtained the previous best result [27] was also executed considering 15 days of execution. Figure 13 shows the evolution of the original objective value for both schemes. It can be noted that in the short-term the mono-objective approach is superior. Then, for a long period of time both approaches obtains similar values. Finally, in the long-term, the multiobjectivised approach provides the best solution. Since only one execution was performed, the superiority of the multiobjectivised scheme can not be ensured. However, results are quite promising. Moreover, the original objective value was raised from 1, 032, 619, 547 to 1, 038, 329, 890.

7 Conclusions and future work

Packing problems are a class of optimisation problems with many practical applications. They are widely used inside more complex systems, e.g. filling containers, loading pallets, optimising the layout of electrical circuits, and scheduling, among others. Since this kind of problem can be classified according to various features, several variants have been defined in the literature. In this paper, we addressed the 2D packing problem (2DPP) defined in the GECCO 2008 competition session.

Several approaches have been proposed in order to tackle the 2DPP. During the contest, the two best-behaved approaches were based on first generation MAs. Subsequently, a parallel hyperheuristic was proposed in an effort to speed up the process of obtaining high quality results. However, subsequent studies concluded that stagnation in local optima may appear for some instances. So as to avoid this drawback, a parallel island-based model was applied to a multiobjectivised version of the 2DPP. The usage of multiobjectivisation methods avoided stagnation problems, though for some instances, the time required by the parallel model was longer than the time invested by the approaches applied in previous research.

The main contribution of this paper is twofold. Firstly, the advantages and drawbacks of a set of first generation MAs were analysed. The first generation MAs considered was applied to a multiobjectivised version of the 2DPP. A novel MA based on SPEA2 was proposed, and a novel multiobjectivisation approach based on the incorporation of a threshold value (DCN- THR) was applied to the 2DPP. The incorporation of a threshold, which must be specified by the user, avoids the survival of low-quality individuals in the population. Computational results demonstrated that, depending on the instance in question, the quality of the results is affected by the MA considered and/or by the multiobjectivisation approach that is applied. In addition, the incorporation of a threshold value in the multiobjectivisation methods also influences the quality of the solutions. Regarding the benefits of the first generation MAs, they were able to achieve high quality solutions in both of the instances tested. However, some drawbacks in terms of their robustness were also identified. Particularly, the best configuration of the first generation MAs for each of the instances tested was not appropriate for the other one.

Secondly, a parallel scheme whose aim was to avoid the robustness problems of the first generation MAs and to improve their behaviour was applied. This parallel scheme (DYN model) combines a parallel island-based model and a hyperheuristic. The different configurations of the first generation MAs were used as the low-level configurations, or memes, of the parallel approach. The set of memes was the same for both instances. The experimental evaluation demonstrated that high quality solutions were obtained by the DYN model for both instances of the 2DPP. Consequently, the DYN model avoids the requirement of independently testing every low-level configuration being considered. This thus mitigates robustness problems by avoiding the need to first check the most suitable configuration of an algorithm for a given instance. Moreover, the DYN model facilitates the application of the first generation MAs from the point of view of the parameter setting, and enables their use in parallel environments. We also studied the effect that the migration stage has on the quality of the solutions. Differences among the migration stages used were more noticeable when a large number of worker islands was taken into account. Moreover, the addition of extra processors to the DYN model using the appropriate migration stage provided benefits both in terms of saved resources and the quality of the results. Finally, the best-known solutions for the instances considered in this paper were improved. The tests were performed using two instances because no additional data sets were available. We made a considerable effort to carry out a complete analysis using these two instances in order to draw general conclusions. The main strength of using the selected instances is that they were previously analysed with other optimisation schemes. Therefore, they proved quite useful to understanding the advantages of the new proposals.

Several areas of research can be considered for future work. First, it would be desirable to incorporate a larger amount of low-level configurations or memes into the DYN model. Since each low-level configuration might behave differently with different instances, satisfactory results might be achieved if more instances are utilised. However, when a large number of configurations is used, the distribution of the computational resources becomes more difficult, which might affect the performance of the DYN model. It would also be interesting to implement a DYN model suitable for hybrid parallel architectures. Such an implementation would be based on MPI and OpenMP. Thus, the DYN model might profit from the hybrid architecture of machines like HECToR. Finally, another interesting area of research would be the design of a model in which the user can specify certain parameters by assigning the ranges that must be used, instead of specifying particular values for them.

Acknowledgments This work was funded in part by the EC (FEDER) and the Spanish Ministry of Science and Innovation as part of the 'Plan Nacional de I+D+i', with contract number TIN2011- 25448. The work of Carlos Segura was funded by grant FPU- AP2008- 03213. The work of Eduardo Segredo was funded by grant

FPU- AP2009- 0457 The work was also funded by the HPC- EUROPA2 project (project number: 228398) with the support of the European Commission—Capacities Area—Research Infrastructures. Our research made use of the facilities of HECToR, the UK's national high-performance computing service, which is provided by UoE HPCx Ltd at the University of Edinburgh, Cray Inc and NAG Ltd, and funded by the Office of Science and Technology through the EPSRC's High End Computing Programme.

References

1. Abbass, H.A., Deb, K.: Searching under multi-evolutionary pressures. In: Proceedings of the Fourth Conference on Evolutionary Multi-Criterion Optimization, pp. 391–404. Springer (2003)
2. Alba, E.: Parallel Metaheuristics: A New Class of Algorithms. Wiley-Interscience, Hoboken, NJ (2005)
3. Araya, I., Neveu, B., Riff, M.C.: An efficient hyperheuristic for strip-packing problems. In: Cotta, C., Sörensen, K. (eds.) Adaptive and Multilevel Metaheuristics, Studies in Computational Intelligence, vol. 136, pp. 61–76. Springer (2008)
4. Blazewicz, J., Walkowiak, R.: A new parallel approach for multi-dimensional packing problem. In: 4th International Conference on Parallel Processing and Applied Mathematics (PPAM), LNCS, vol. 2328, pp. 194–201. Naleczow, Poland, Springer, Berlin (2002)
5. Bui, L., Abbass, H., Branke, J.: Multiobjective optimization for dynamic environments. In: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, CEC 2005, vol. 3, pp. 2349–2356 (2005)
6. Burke, E.K., Kendall, G., Newall, J., Hart, E., Ross, P., Schulenburg, S.: Handbook of Meta-heuristics. Hyper-heuristics: An Emerging Direction in Modern Search Technology. Kluwer, Berlin (2003)
7. Burke, E.K., Kendall, G., Silva, J.L., O'Brien, R., Soubeiga, E.: An ant algorithm hyperheuristic for the project presentation scheduling problem. In: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, CEC 2005, vol. 3, pp. 2263–2270. Edinburgh, Scotland (2005)
8. Burke, E.K., Kendall, G., Soubeiga, E.: A tabu-search hyperheuristic for timetabling and rostering. J. Heuristics **9**(6), 451–470 (2003)
9. Chen, P.C., Kendall, G., Vanden Berghe, G.: An ant based hyper-heuristic for the travelling tournament problem. In: Proceedings of IEEE Symposium of Computational Intelligence in Scheduling (CISched 2007), pp. 19–26. Honolulu, Hawaii (2007)
10. Coello, C.A., Lamont, G.B., Veldhuizen, D.A.V.: Evolutionary Algorithms for Solving Multi-Objective Problems. Genetic and, Evolutionary Computation (2007)
11. Cowling, P., Kendall, G., Han, L.: An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. In: Proceedings of the 2002 IEEE Congress on Evolutionary Computation, CEC 2002, vol. 2, pp. 1185–1190. IEEE Computer Society, Washington, DC, USA (2002)
12. Cowling, P., Kendall, G., Soubeiga, E.: A parameter-free hyperheuristic for scheduling a sales summit. In: Proceedings of 4th Metahuristics International Conference (MIC 2001), pp. 127–131. Porto Portugal (2001)
13. Cowling, P., Kendall, G., Soubeiga, E.: Hyperheuristics: A robust optimisation method applied to nurse scheduling. In: PPSN, Lecture Notes in Computer Science, vol. 2439, pp. 851–860. Springer (2002)
14. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**, 182–197 (2002)
15. Demšar, J.: Statistical comparison of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
16. Dowsland, K., Soubeiga, E., Burke, E.: A simulated annealing hyper-heuristic for determining shipper sizes. Eur. J. Oper. Res. **179**(3), 759–774 (2007)
17. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing (Natural Computing Series). Springer, Berlin (2008)
18. Glover, F.W., Kochenberger, G.A.: Handbook of Metaheuristics (International Series in Operations Research & Management Science). Springer, Berlin (2003)
19. HECToR: UK National Supercomputing Service. http://www.hector.ac.uk/
20. Hong, T.P., Tsai, M.W., Liu, T.K.: Two-dimentional encoding schema and genetic operators. In: JCIS. Atlantis Press (2006)
21. Hoos, H., Stützle, T.: Stochastic Local Search: Foundations and Applications. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann Publishers, Burlington (2005)
22. Jaszkiewicz, A.: Genetic local search for multi-objective combinatorial optimization. Eur. J. Oper. Res. **137**(1), 50–71 (2002)
23. Kendall, G., Cowling, P., Soubeiga, E.: Choice function and random hyperheuristics. In: Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL 2002), pp. 667–671. Singapore (2002)

24. Knowles, J.D., Watson, R.A., Corne, D.: Reducing local optima in single-objective problems by multi-objectivization. In: Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, EMO '01, pp. 269–283. Springer, London (2001)
25. Lara, O., Labrador, M.: A multiobjective ant colony-based optimization algorithm for the bin packing problem with load balancing. In: Proceedings of the 2010 IEEE Congress on Evolutionary Computation, CEC 2010, pp. 1–8 (2010)
26. León, C., Miranda, G., Rodríguez, C., Segura, C.: 2D Cutting stock problem: a New Parallel algorithm and bounds. In: Proceedings of Euro-Par, LNCS, vol. 4641, pp. 795–804. Springer (2007)
27. León, C., Miranda, G., Segura, C.: A memetic algorithm and a parallel hyperheuristic island-based model for a 2d packing problem. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, GECCO '09, pp. 1371–1378. ACM, New York, NY, USA (2009)
28. León, C., Miranda, G., Segura, C.: METCO: a parallel plugin-based framework for multi-objective optimization. Int. J. Artif. Intell. Tools **18**(4), 569–588 (2009)
29. Lodi, A., Martello, S., Monaci, M.: Two-dimensional packing problems: a survey. Eur. J. Oper. Res. **141**(2), 241–252 (2002)
30. Martello, S., Monaci, M., Vigo, D.: An exact approach to the strip-packing problem. INFORMS J. Comput. **15**(3), 310–319 (2003)
31. Matayoshi, M.: Two dimensional rectilinear polygon packing using genetic algorithm with a hierarchical chromosome. In: Proceedings of the IEEE International Conference on Systems, Man and, Cybernetics, pp. 989–996 (2011)
32. Nguyen, Q.H., Ong, Y.S., Lim, M.H.: Non-genetic transmission of memes by diffusion. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation, GECCO '08, pp. 1017–1024. ACM, New York, NY, USA (2008)
33. Ong, Y.S., Lim, M.H., Zhu, N., Wong, K.W.: Classification of adaptive memetic algorithms: a comparative study. IEEE Trans. Syst. Man Cybern. **36**(1), 141–152 (2006)
34. Segredo, E., Segura, C., León, C.: A multiobjectivised memetic algorithm for the frequency assignment problem. In: Proceedings of the 2011 IEEE Congress on Evolutionary Computation, CEC 2011, pp. 1132–1139 (2011)
35. Segura, C., Segredo, E., León, C.: Parallel island-based multiobjectivised memetic algorithms for a 2D packing problem. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO '11, pp. 1611–1618. ACM, New York, NY, USA (2011)
36. Snir, M., Otto, S., Huss-Lederman, S., Walker, D., Dongarra, J.: MPI: The Complete Reference. The MIT Press, Cambridge (1996)
37. Toffolo, A., Benini, E.: Genetic diversity as an objective in multi-objective evolutionary algorithms. Evol. Comput. **11**, 151–167 (2003)
38. Veldhuizen, D.A.V., Zydallis, J.B., Lamont, G.B.: Considerations in engineering parallel multiobjective evolutionary algorithms. IEEE Trans. Evol. Comput. **7**(2), 144–173 (2003)
39. Vink, T., Izzo, D.: Learning the best combination of solvers in a distributed global optimization environment. In: Proceedings of Advances in Global Optimization: Methods and Applications (AGO), pp. 13–17. Mykonos, Greece (2007)
40. Wäscher, G., Haußner, H., Schumann, H.: An improved typology of cutting and packing problems. Eur. J. Oper. Res. **183**(3), 1109–1130 (2007)
41. Zhou, Y., Rao, Y., Zhang, G., Zhang, C.: An adaptive memetic algorithm for packing problems of irregular shapes. Adv. Mater. Res. **314–316**, 1029–1033 (2011)
42. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In: Giannakoglou, K.C. et al., (eds.) Evolutionary Methods for Design, Optimization and Control with Application to Industrial Problems (EUROGEN 2001), pp. 95–100. International Center for Numerical Methods in Engineering (CIMNE) (2002)