Memetic Comp. (2011) 3:33–49
DOI 10.1007/s12293-010-0044-5

REGULAR RESEARCH PAPER

Parallel hyperheuristics for the frequency assignment problem

Special issue on nature inspired cooperative strategies for optimization

Carlos Segura · Gara Miranda · Coromoto León

Received: 30 September 2009 / Accepted: 26 May 2010 / Published online: 10 June 2010

Abstract This work presents a set of approaches used to deal with the frequency assignment problem (FAP), which is one of the key issues in the design of GSM networks. The used formulation of FAP is focused on aspects which are relevant for real-world GSM networks. A memetic algorithm, together with the specifically designed local search and variation operators, are presented. The memetic algorithm obtains good quality solutions but it must be adapted for each instance to be solved. A parallel hyperheuristic-based model was used to parallelize the approach and to avoid the requirement of the adaptation step of the memetic algorithm. The model is a hybrid algorithm which combines a parallel island-based scheme with a hyperheuristic approach. The main operation of the island-based model is kept, but the configurations of the memetic algorithms executed on each island are dynamically mapped. The model grants more computational resources to those configurations that show a more promising behavior. For this purpose two different criteria have been used in order to select the configurations. The first one is based on the improvements that each configuration is able to achieve along the executions. The second one tries to detect synergies among the configurations, i.e., detect which configurations obtain better solutions when they are cooperating. Computational results obtained for two different real-world instances of the FAP demonstrate the validity of the proposed model. The new designed schemes have made possible to improve the previously known best frequency plans for a real-world network.

Keywords Frequency assignment problem · Memetic algorithms · Hyperheuristics · Parallel Island-based models · Cooperative strategies

C. Segura (✉) · G. Miranda · C. León
Dpto. Estadística, I. O. y Computación,
Universidad de La Laguna, La Laguna,
38271 Santa Cruz de Tenerife, Spain
e-mail: csegura@ull.es

G. Miranda
e-mail: gmiranda@ull.es

C. León
e-mail: cleon@ull.es

1 Introduction

The frequency assignment problem (FAP) is a well-known NP-complete combinatorial optimization problem of great importance to the radio-communication industry. FAP arises as one of the crucial issues in the design of GSM—global system for mobile communications—networks [51]. This problem is also known as automatic frequency planning (AFP) and channel assignment problem (CAP). The literature of FAP has grown quickly over the past years. This is mainly due to the fast implementation of wireless telephone networks and satellite communications projects [1]. TV broadcasting and military communications problems have also inspired new research in the domain.

The set of applications of FAP leads to many different mathematical and engineering models, but all of them share two common features:

- A set of antennae must be assigned frequencies such that data transmissions between the two end points of each connection is possible.
- Depending on the frequencies assigned to the antennae, they may interfere to one another, resulting in quality loss of signal.

This work is focused in the FAP which arises in the design of GSM networks. In such a case, the available frequency

band is slotted into channels which have to be allocated to the elementary transceivers (TRXs) installed in the base stations of the network. In GSM, FAP is a hard design task because the usable radio spectrum is very scarce and frequencies have to be reused throughout the network, and consequently, some inevitable degree of interference will occur. The goal of the designer is to minimize the interferences of the network, i.e., minimize the quality loss of signal. Tackling the FAP is crucial for today's GSM operators not only at the stage of the initial design, but also in subsequent modifications of the network aimed at solving, for instance, unpredicted interference reports or handling an increase of traffic demand in some areas. Indeed, by mid 2006, GSM services were used by more than 1.8 billion subscribers (http://www.wirelessintelligence.com/) across 210 countries, representing approximately 77% of the world's cellular market. It is widely accepted that the third generation mobile telecommunication system (*Universal Mobile Telecommunication System* or UMTS) [54] will coexist with the enhanced releases of the GSM standard (GPRS [26] and EDGE [23]) at least in the first phases. GSM is then expected to play an important role as a dominating technology for many years. Therefore, frequency planning in these networks will be an important task, at present as well as in the future.

From a mathematical point of view, the FAP is a generalization of the graph coloring problem and therefore, it is NP-hard [29]. From an engineering point of view, the basic FAP formulation is extended in order to tackle real world issues. For instance, initial formulations in the 1970s were very similar to the classical graph coloring problem and assumed that adjacent frequencies do not interfere (only co-channel interferences were considered) [1]. However, real-world models must take into account all possible sources of interference, as well as regulatory concerns, and technological limitations [20]. Nowadays, the large traffic demand and the reduced frequency spectrum make it impossible to find interference-free frequency assignments. Therefore, most efforts in the current literature intend to obtain frequency plans that minimize the overall interference of the network (in other words, that maximize the quality of service). Even so, most of the research deals with benchmarking-like problems. In this work, we use a novel formulation proposed in [44], so as to take full advantage of realistic and accurate interference information from a real-world GSM network.

The FAP was firstly introduced in the early 1970s by Metzger [50]. Since then, multiple approaches to deal with the FAP has been proposed. This large production has been analyzed and organized in several surveys and books [1, 21,32,39]. On one hand, it has been frequently used as a benchmark problem, because of its relation with other combinatorial problems. On other hand, it has been studied as a real-world engineering problem, because of its applications.

Regarding to the approaches designed to deal with FAP formulations, some exact proposals are seen to exist [5,22, 48]. However, they are not feasible when tackling large instances of the problem [1], so several heuristic and metaheuristics methods [3] have also been proposed to deal with FAP. In particular, most of the current research in the FAP is based on using Memetic Algorithms (MAs) [28,31,36, 49,55]. MAs [38,52] are a synergy of a population-based approach with separate individual learning or local improvement procedures for problem search. They are also referred to in the literature as Baldwinian evolutionary algorithms, Lamarckian evolutionary algorithms, cultural algorithms or genetic local search. Usually they make use of problem domain information for implementing the learning process. They are of great value because they perform some orders of magnitude faster than traditional genetic algorithms for some problem domains [24].

According to [1] the here considered version of FAP is classified as a MI-FAP (minimum interference frequency assignment problem) and for this formulation of the problem the publications are more reduced. In [44] an Ant Colony Optimization (ACO) algorithm was adapted to the problem. A comparative study using a large set of metaheuristics, including ACO, was performed in [45]. It included both population-based and trajectory-based metaheuristics. It revealed the good performance of a memetic algorithm with increasing population size. The algorithm is a modified version of a $(1 + 1)$ Evolutionary Algorithm (EA), combined with a local search specifically designed to deal with this version of FAP. It made possible to obtain good quality solutions for different instances. However, it must be adapted for each solved instance. Thus, previously to solving a problem instance, a tuning step of the algorithm must be performed.

Several studies have been performed in order to reduce the execution time and the resource expenditure when using metaheuristics. These studies naturally lead to its parallelization [2,59,61]. Several models of parallel evolutionary algorithms (PEAs) have been designed. PEAs can be classified [12] in four major computational paradigms: master-slave, island-based or coarse-grained, diffusion or cellular, and hybrid paradigm. These evolutionary approaches are proved effectively solving problems, but they are often time and domain knowledge intensive. The heavy dependence on problem specific knowledge affects their reusability. In order to provide a reusable and robust approach, applicable to a wide range of problems and instances, a novel parallel model has been applied. The model combines the operation of an island-based scheme with the hyperheuristic approach to manage the choice of which lower-level metaheuristics should be applied at any given time, depending upon the characteristics of the algorithm, problem, and instance itself.

The proposal here presented lies on the application of the parallel hyperheuristic island-based model using the aforementioned memetic algorithm to the FAP. Our present work has three main aims:

- Design a parallel algorithm for the FAP which is able to avoid the instance tuning step of the memetic algorithm.
- Reduce the required time to achieve good quality solutions.
- Improve the best solutions obtained in previous researches for the tackled instances.

The organization of the paper is as follows. Section 2 describes the mathematical formulation of the frequency assignment problem here analyzed. Section 3 introduces the sequential approach used to solve the FAP. Section 4 is devoted to introduce the concept of hyperheuristics. The designed parallel hyperheuristic-based island model is explained in Sect. 5. The experimental evaluation of the algorithms is presented in Sect. 6. Finally, in Sect. 7 the main conclusions and an outline of future work are offered.

2 Mathematical formulation

In the last years, the basic FAP formulation has been widely extended in order to tackle real world issues [37,44]. Most of the FAP models differs in the way that the interferences are measured. Computing the level of interference is a difficult task which depends on the channels, the radio signals and many other properties of the environment. Several ways of quantifying this interference exist, resulting in the so-called *interference matrix*, usually denoted by M. Some theoretical methods to measure M have been proposed [1]. In [37] extensive measurements in the network are performed in order to calculate M. Theoretical methods have the advantage that new instances can be tackled with less effort. In the case of the extensive measurements methods, the M matrix is constituted by more accurate values, so resulting in more realistic frequency plans. However, applying the method to new networks is an expensive task, because it requires an extensive measurement step for each tackled instance. The FAP formulation here proposed is based on a matrix M calculated by extensive measurements.

Let $T = \{t_1, t_2, \ldots, t_n\}$ be a set of n *transceivers*, and let $F_i = \{f_{i1}, \ldots, f_{ik}\} \subset \mathbb{N}$ be the set of valid *frequencies* that can be assigned to a transceiver $t_i \in T, i = 1, \ldots, n$. Note that k—the cardinality of F_i—is not necessarily the same for all the transceivers. Furthermore, let $S = \{s_1, s_2, \ldots, s_m\}$ be a set of given *sectors* (or cells) of cardinality m. Each transceiver $t_i \in T$ is installed in exactly one of the m sectors. Henceforth we denote the sector in which a transceiver t_i is installed by $s(t_i) \in S$. Finally, the *interference matrix* $M = \{(\mu_{ij}, \sigma_{ij})\}_{m \times m}$, is given. The two elements μ_{ij} and σ_{ij} of a matrix entry $M(i, j) = (\mu_{ij}, \sigma_{ij})$ are numerical values greater than or equal to zero. μ_{ij} represents the mean and σ_{ij} the standard deviation of a Gaussian probability distribution describing the carrier-to-interference ratio (C/I) [64] when sectors i and j operate on a same frequency. The higher the mean value, the lower the interference and thus the better the communication quality. Note that the interference matrix is defined at sector (cell) level, because the transceivers installed in each sector all serve the same area.

A solution to the problem is obtained by assigning to each transceiver $t_i \in T$ one of the frequencies from F_i. A solution (or frequency plan) is henceforth denoted by $p \in F_1 \times F_2 \times \cdots \times F_n$, where $p(t_i) \in F_i$ is the frequency assigned to transceiver t_i. The objective is to find a solution p that minimizes the following cost function:

$$C(p) = \sum_{t \in T} \sum_{u \in T, u \neq t} C_{\text{sig}}(p, t, u) \tag{1}$$

In order to define the function $C_{\text{sig}}(p, t, u)$, let s_t and s_u be the sectors in which the transceivers t and u are installed, that is, $s_t = s(t)$ and $s_u = s(u)$, respectively. Moreover, let $\mu_{s_t s_u}$ and $\sigma_{s_t s_u}$ be the two elements of the corresponding matrix entry $M(s_t, s_u)$ of the interference matrix with respect to sectors s_t and s_u. Then,

$$C_{\text{sig}}(p, t, u) = \begin{cases} K & \text{if } s_t = s_u, |p(t) - p(u)| < 2 \\ C_{\text{co}}(\mu_{s_t s_u}, \sigma_{s_t s_u}) & \text{if } s_t \neq s_u, \mu_{s_t s_u} > 0, |p(t) - p(u)| = 0 \\ C_{\text{adj}}(\mu_{s_t s_u}, \sigma_{s_t s_u}) & \text{if } s_t \neq s_u, \mu_{s_t s_u} > 0, |p(t) - p(u)| = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

In real networks, it is unfeasible to operate with more than one transceiver with the same or adjacent frequencies serving the same area. Thus, K is defined as a very large constant.

Function $C_{\text{co}}(\mu, \sigma)$ is defined as follows:

$$C_{\text{co}}(\mu, \sigma) = 100 \left(1.0 - Q\left(\frac{c_{\text{SH}} - \mu}{\sigma}\right)\right) \tag{3}$$

where

$$Q(z) = \int_z^\infty \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} \, \mathrm{d}x \tag{4}$$

is the tail integral of a Gaussian probability distribution function with zero mean and unit variance, and c_{SH} is a minimum quality signalling threshold. Function Q is widely used in digital communication systems because it characterizes the error probability performance of digital signals [58]. This means that $Q(\frac{c_{\text{SH}} - \mu}{\sigma})$ is the probability of the C/I ratio being greater than c_{SH} and, therefore, $C_{\text{co}}(\mu_{s_t s_u}, \sigma_{s_t s_u})$ computes the probability of the C/I ratio in the serving area of sector

s_t being below the quality threshold due to the interferences provoked by sector s_u. That is, if this probability is low, the C/I value in the sector s_t is not likely to be degraded by the interfering signal coming from sector s_u and thus the communication quality yielded is high. Note that this is compliant as to defining a minimization problem. On the contrary, a high probability—and consequently a high cost—causes the C/I mostly to be below the minimum threshold c_{SH} and thus incurring in low quality communications.

As function Q has no closed form for the integral, it has to be evaluated numerically. For this purpose we use the complementary error function E:

$$Q(z) = \frac{1}{2} E\left(\frac{z}{\sqrt{2}}\right) \tag{5}$$

In [53], a numerical method is presented that allows the value of E to be computed with a fractional error smaller than 1.2×10^{-7}. Analogously, function $C_{\mathrm{adj}}(\mu, \sigma)$ is defined as:

$$\begin{aligned} C_{\mathrm{adj}}(\mu,\sigma) &= 100\left(1.0 - Q\left(\frac{c_{\mathrm{SH}} - c_{\mathrm{ACR}} - \mu}{\sigma}\right)\right) \\ &= 100\left(1.0 - \frac{1}{2} E\left(\frac{c_{\mathrm{SH}} - c_{\mathrm{ACR}} - \mu}{\sigma\sqrt{2}}\right)\right) \end{aligned} \tag{6}$$

The only difference between functions C_{co} and C_{adj} is the additional constant $c_{\mathrm{ACR}} > 0$ (adjacent channel rejection) in the definition of function C_{adj}. This hardware specific constant measures the receiver's ability to receive the wanted signal in the presence of an unwanted signal at an adjacent channel. Note that the effect of constant c_{ACR} is that $C_{\mathrm{adj}}(\mu, \sigma) < C_{\mathrm{co}}(\mu, \sigma)$. This makes sense, since using adjacent frequencies (channels) does not provoke such a strong interference as using the same frequencies.

3 A memetic algorithm with increasing population size

The applied sequential approach (see Algorithm 1) is a memetic algorithm which combines a modified evolutionary algorithm with a $(1 + 1)$ selection operator and a local search specifically designed to face the considered version of the FAP. The algorithm has also been successfully applied to a variation of a combinatorial 2D Packing Problem [42] proposed in the GECCO 2008 contest session, achieving the second best solution in the competition (http://www.sigevo.org/gecco-2008/competitions.html). The algorithm has the ability to perform as a trajectory-based algorithm when no stagnation is detected, however it increases the population size in order to avoid strong local optima when necessary, behaving then as a population-based algorithm.

Individuals are encoded as an array of integer values, p, where $p(x)$ is the slot assigned to the transceiver t_x. $InitPSize$ initial individuals are generated in a completely random way (line 1). For each gene, a random value among the admissible ones is assigned. On each generation, the approach applies a variation operator over the population (line 4). The variation step consists in the application of a mutation operator to each individual in order to produce new offsprings. The $(1+1)$ selection operator is deterministic and selects the best individual between an offspring and its parent. In order to improve the behavior of the approach when dealing with local optima, two improvements were considered. First, if after $SoftBloq$ generations the fitness of the current individual has not been improved, the selection operator used during the generation is a (1, 1), i.e., the offspring is selected independently of its fitness value (lines 6–12). Moreover, if after $HardBloq$ generations the fitness value of none of the individuals has been improved, an extra new individual is introduced in the population (lines 13–17). During the following generations, each individual included in the population is evolved applying the aforementioned rules. In order to avoid an uncontrolled growth of the population, the maximum size of the population is limited to $MaxPopSize$.

Algorithm 1 Pseudocode for the Memetic Approach

```
1: initialise(P)
2: P ← localSearch(P)
3: while not time-limit do
4:    offspring ← variation(P)
5:    offspring ← localSearch(offspring)
6:    for i = 0 to populationsize do
7:       if P(i) is blocked SoftBloq generations then
8:          P(i) ← offspring(i)
9:       else
10:         P(i) ← best(P(i), offSpring(i))
11:      end if
12:   end for
13:   if P is blocked HardBloq generations then
14:      if P.size < MaxPopSize then
15:         increase population size
16:      end if
17:   end if
18: end while
```

3.1 Local search

The application of local search methods allows admissible solutions to be achieved in relatively short times. This is a typical requirement within commercial tools, the context in which the FAP resides. The local search strategy has been specifically designed to deal with our version of FAP. Given its importance, a large effort has been put into making the local search as efficient as possible.

The operation of the designed local search is based on optimizing the assignment of frequencies to TRXs in a given sector, without modifying the remaining network assignments. In [47] a local search method which also substitutes the frequencies assigned to a set of transceivers, leaving intact the remaining network, is proposed. However, in such a case,

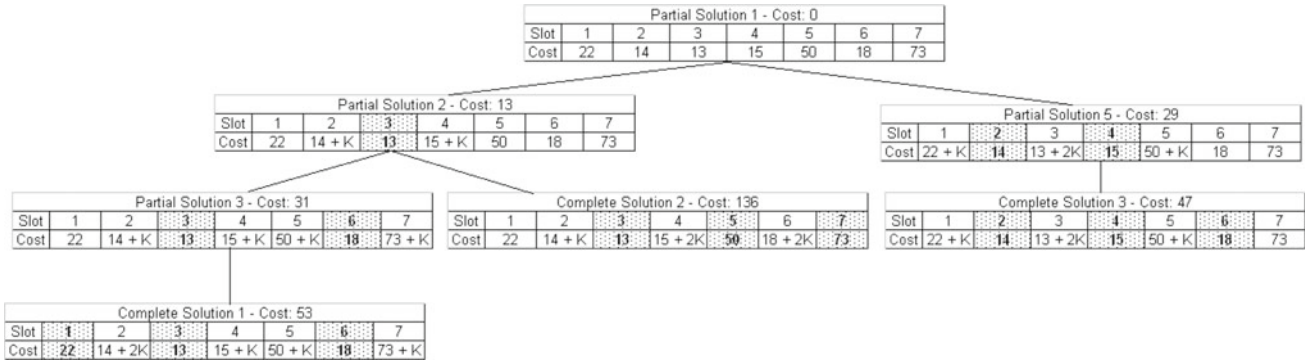

Fig. 1 Generation of a new neighbor by reassigning the frequencies of a sector

the set of considered transceivers are not those inside a sector but the ones satisfying a set of specific properties. In [7,44] the local search methods are simpler. In both cases, the new neighbors are generated by changing the assignment to a only one TRX. Other proposals use tabu search [25] or guided local search [63] in order to implement the local search step.

In our proposal, the neighbors of a candidate solution are obtained by replacing the frequencies in the TRXs of each sector. The reassignment of frequencies within a sector is performed in the following way: first, the available frequencies for the sector are sorted by their involved cost. Then, two possibilities are considered, either assign the frequency with lowest associated cost to a TRX that is allowed to use that frequency, or assign its two adjacent frequencies to two different TRXs (if they are allowed to use these frequencies). For each of the newly generated partial solutions the same process is repeated until all TRXs in the sector have been assigned a frequency. The complete solution with lowest associated cost is considered as the new neighbor, while the other ones are discarded. Figure 1 illustrates the generation of a new neighbor. In this example, it is considered that the sector contains three TRXs, and that each TRX can use any frequency slot. For every node, the cost associated to each slot is shown. The children of a node are generated following the rules previously detailed. The slots assigned to the TRXs are marked in bold. The nodes with three slots assigned are complete solutions, while the other ones are partial solutions. The complete solution identified by the number three is the new neighbor because is the one with lowest cost. The remaining generated solutions are discarded.

The order in which neighbors are analyzed is randomly determined (line 7 of Algorithm 2), but trying to avoid the generation of neighbors that do not improve the current solution. For such a purpose, a set called $currentSectors$ containing the sectors that might improve the current solution is maintained. Initially, all sectors are introduced in $currentSector$ (lines 2 and 4). For the generation of a new neighbor,

Algorithm 2 Pseudocode for the Local Search

```
1: Input: current solution S
2: nextSectors ← {1, ..., numberOfSectors}
3: while (nextSectors != ∅) do
4:    currentSectors ← nextSectors
5:    nextSectors ← ∅
6:    while (currentSectors != ∅) do
7:       sec ← extract a random sector from currentSectors
8:       neighbour ← reassign frequencies of S in sector sec
9:       if (neighbour improves S) then
10:         S ← neighbour
11:         nextSectors += sectors interfered by sec
12:         nextSectors += sectors that interfere sec
13:      end if
14:   end while
15: end while
16: return S
```

a sector sec is randomly extracted from $currentSector$ (line 7) and its frequencies reassigned as aforementioned (line 8). The local search moves to the first new generated neighbor that improves the current solution (lines 9–10), adding all the sectors that interfere or are interfered by sec to the set of the next sectors ($nextSectors$) to consider (lines 11–12). When $currentSectors$ set gets empty (line 6), sectors in $nextSectors$ are transferred to the current set (line 4) and $nextSectors$ set is cleared (line 5). The local search stops when none of the neighbors improves the current solution (line 3).

In cases where the network satisfies a set of properties, the neighbor generation process ensures the achievement of the optimal frequency assignment inside the analyzed sector, considering the remaining network fixed. Such properties are (1) all TRXs in a given sector are allowed to use the same frequency ranges, (2) it is possible to make assignments which do not use the same frequency or adjacent frequencies in any two TRXs serving the same area, and (3) the best assignment does not use the same frequency or adjacent frequencies in any two TRXs withing the same sector. A sketch of the proof is here presented. Let be $Cost(f)$ the cost associated to the assignment of the frequency f to any of the TRXs in the

considered sector. Being f_1 the frequency with minimum associated cost, the best assignment must use f_1, or must simultaneously use $f_1 - 1$ and $f_1 + 1$. In fact, considering an assignment in which $f_1 + 1$ is used, but $f_1 - 1$ is not used, we can substitute the assignment of $f_1 + 1$ by f_1, thus obtaining an assignment with lower cost. In the case of using $f_1 - 1$, but not $f_1 + 1$, the same property holds. In the cases where $f_1 - 1$ and $f_1 + 1$ are not used, since f_1 is the best possible assignment, it must be used. Finally, the simultaneous assignment of both $f_1 + 1$ and $f_1 - 1$, could lead to a better assignment than the ones using f_1 and other frequency f_2. For this reason, in order to ensure that the best assignment is achieved, individuals which use f_1, and individuals which use simultaneously $f_1 - 1$ and $f_1 + 1$ should be analyzed. The way in which neighbors are generated ensure that both possibilities are explored, so the best assignment is achieved under such conditions.

Considering that K is a very large constant, the three considered properties hold for the example illustrated in Fig. 1. Thus, considering the remaining network fixed, the generated solution is the best one.

3.2 Variation operators

A variation step, consisting on applying a mutation operator is performed on each generation (Algorithm 1—line 4). Three different mutation operators were implemented and compared. They include both directed and random operators. The tested mutation operators were the following:

- *Uniform Mutation (UM)*: each gene—or transceiver assignment—is mutated with a probability p_m. In order to perform the new assignment to the gene, a random value among the admissible ones is selected.
- *Mapping Mutation (MM)*: being F the set of accepted frequencies by any of the transceivers, a random bijection $m : F \leftrightarrow F$ is generated. Each transceiver assignment t_x is replace with a probability p_m by the value $m(t_x)$, if $m(t_x)$ is an admissible value for the transceiver t_x.
- *Neighbor-based Mutation (NM)*: first, a random TRX t_x is mutated. Then, its neighbors, i.e., the TRXs which interfere t_x, or are interfered by t_x, are mutated with a probability p_m. The previous steps are repeated N times, but the TRX is selected among those ones which are neighbors of the TRXs that has been mutated in the previous steps. Thus, the mutation operator is focusing on one zone of the network.

4 Hyperheuristics

A reduced version of the designed sequential approach here proposed was used to deal with a combinatorial 2D Packing Problem [42] and with the here analyzed version of FAP [45]. In the case of the FAP, one mutation operator, and one instance of the problem was analyzed. Promising results were achieved, however, the impact of the mutation operators and its parameterization were not studied. Subsequent experiments have revealed the importance of the selected mutation operator as well as its parameterization. Thus, in order to obtain a good performance, it is necessary to adapt the strategy to each solved instance. In order to improve the quality of the achieved solutions, an analysis of the impact of the different parameters with each new instance should be performed. Usually, the users do not have a prior knowledge about the algorithm behavior when applied to a particular instance, so if they have to try many alternatives, the process could take too much user and computational effort. In order to avoid such a step, hyperheuristics can be applied.

A hyperheuristic can be viewed as a heuristic that iteratively chooses between a set of given low-level (meta)-heuristics in order to solve an optimization problem [9]. Hyperheuristics operates at a higher level of abstraction than heuristics, because they have no knowledge about the problem domain. The motivation behind the approach is that, ideally, once a hyperheuristic algorithm has been developed, several problem domains and instances could be tackled by only replacing the low-level (meta)-heuristics. Thus, the aim in using a hyperheuristic is to raise the level of generality at which most current (meta)-heuristic systems operate. Since the main motivation of hyperheuristics is to design problem-independent strategies, a hyperheuristic is not concerned with solving a given problem directly as is the case of most heuristics implementations. In fact, the search is on a (meta)-heuristic search space rather than a search space of potential problem solutions. The hyperheuristic solves the problem indirectly by recommending which solution method to apply at which stage of the solution process. Generally, the goal of raising the level of generality is achieved at the expense of reduced—but still acceptable—solution quality when compared to tailor-made (meta)-heuristic approaches. A diagram of a general hyperheuristic framework [9] is shown in Fig. 2. It shows a problem domain barrier between the low level (meta)-heuristics and the hyperheuristic itself. The data flow obtained by the hyperheuristic could include the quality of achieved solutions (average, improvement, best, worst), the resources (time, processors, memory) invested to achieve such solutions, etc. Based on such a information, the hyperheuristic make its decisions. The data flow coming from the hyperheuristic could include information about which heuristic must be executed, its parameters, stop criteria, etc. The term hyperheuristic also refers to framekwors which are capable of generating new heuristics by combining a set of simpler components [6].

Previously to the appearance of the concept of hyperheuristic, some research was performed analyzing similar ideas.

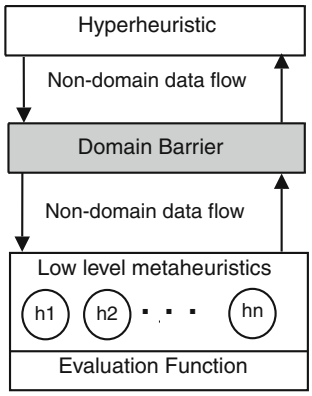

Fig. 2 Hyperheuristic framework

Composer [27] was one of the first proposals which used a search space constituted by heuristics. In such a case, the search was performed by using a hill-climbing strategy. Other proposals consisted in hybridizing genetic algorithms and heuristics [60].

Hyperheuristics can be classified in terms of the characteristics of the low-level metaheuristics into two groups [11], the ones which operate with constructive techniques and the ones which operate with improvement techniques. Constructive techniques are used to build solutions from scratch. At each step, they determine a subpart of the solution. Improvement metaheuristics are iterative approaches which take an initial solution, and modify it with the aim of improving the objective value. Some hyperheuristics have been designed to operate specifically with one kind of low-level metaheuristics, while other ones, can use both, constructive and improvement methods.

Several ways of incorporating the ideas of hyperheuristics into an optimization problem have been proposed. The hyperheuristics which deal with mono-objective optimization problems are much more extensive. In [10] a tabu search based hyperheuristic is presented. It operates with improvement low-level heuristics. The same hyperheuristic was used inside a simulated annealing algorithm [19]. The hyperheuristic was used to combine several neighborhood definitions. Tabu search based hyperheuristics have also been used combined with constructive methods [11]. Other metaheuristics which have inspired the creation of hyperheuristics are genetic algorithms [14] and ant colony optimization [8,13,18]. In the ant-based hyperheuristics, the pheromones represent how well a (meta)-heuristic operates after the application of another (meta)-heuristic. Local search with restart [4] has also been used to implement hyperheuristics. The choice functions has been used multiple times [15,16,33]. In such cases, a scoring function is used to assess the performance of each low-level heuristic. The resources are granted to the heuristic which maximize such a function. In [62] 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 [34,35] the resources are assigned in a random way, and the hyperheuristic decides to accept or reject the changes performed by each heuristic.

5 A parallel dynamic mapped island-based model

Parallel evolutionary algorithms (PEAs) can be classified [12] in four major computational paradigms: master-slave, island-based or coarse-grained, diffusion or cellular, and hybrid paradigm. Such paradigms can be extended to MAs parallelization, just by substituting the EAs by MAs. Island-based models, also known as multi-deme models, have shown good performance and scalability in many areas [2]. In such a model, the population is divided into a number of independent subpopulations or demes. Each subpopulation is associated to an island and an EA or MA configuration is executed over each subpopulation. A *configuration* is constituted by an algorithm together with its parameterization, variation operators and probabilities. Usually, each available processor constitutes an island which evolves in isolation for the majority of the parallel run. Occasionally, some solutions can be transferred among islands following a migration scheme. Several island-based variants are seen to exist:

1. All islands execute identical configurations (homogeneous).
2. All islands execute different configurations (heterogeneous).
3. Each island represents a different region of the genotype domain.
4. Each island represents a different region of the phenotype domain.

Parallel EA-based schemes show the same drawbacks as sequential EA-based approaches. Usually, the dependence on problem or instance specific knowledge hinders the application of them. For instance, comparisons between parallel island-based models [40] show that if there exists an algorithm which clearly outperforms the other ones in solving one type of problem, the homogeneous island-based model using such an algorithm allows to obtain good quality solutions. However, it is difficult to know a priori which configuration is the most appropriate to solve a problem. If the chosen algorithm is not suitable for the problem to solve, poor quality solutions will be achieved. Heterogeneous models allows to execute different configurations on each processor at the same time. By using heterogeneous models, the user avoids the selection of a specific configuration to solve the problem. However, if some of the configurations are not suitable to optimize the problem, a waste of resources is done. In order to provide a more reusable, robust,

and efficient approach, applicable to a wider range of problems and instances, a parallel dynamic mapped island-based model can be applied. The model combines the operation of an island-based scheme with a hyperheuristic approach to manage the choice of which lower-level algorithm configuration is executed on each island at each optimization stage. Similar models have been applied to mono-objective problems [42] and multi-objective problems [56]. The novelty resides in the adaptation of the model to the FAP, and in the incorporation of novel hyperheuristics inside the model.

The underlying principle in adding a hyperheuristic approach to a standard island model is that different configurations have different strengths and weaknesses and it makes sense to combine them in parallel in an intelligent manner. Thus, the proposed parallel model breaks from the standard island scheme adding an adaptive property behavior to it. The adaptive property allows, by applying a hyperheuristic, to perform a dynamic mapping among the configurations and the islands, with the aim of granting more resources to the most promising configurations.

The no-free-lunch Theorem [65] shows that if an algorithm achieves superior results on some problems, it must pay with inferiority on other problems. The aim of the here proposed model is to be able to solve a large set of problems in acceptable times, at the cost of not being able to solve the problems as quick as a tailor made optimization scheme. However, in some cases it could happen that, due to the features of the problem and/or due to the stochastic behavior of the algorithms, the hyperheuristic could fail. Special difficulties arise when dealing with problems in which it is better to apply different algorithms and/or operators at different optimization stages. These difficulties increase when such stages are not long enough to be detected by the hyperheuristic. In order to adapt the resources assignment, the hyperheuristic requires some time to detect the behavior of the algorithms, so, if the changes appear too quickly, the performance of the approach gets worse.

The architecture of the new hybrid model is similar to the island model, i.e., it is constituted by a set of *slave islands* that evolve in isolation applying a certain evolutionary or memetic algorithm to a given population (see Algorithm 3). The number of islands and the set of configurations that could be applied over the local populations are defined by the user. Also, as in the island-based model, a tunable migration scheme allows the exchange of solutions among neighbor islands. Moreover, a new special island is introduced into the scheme. That island, called *master island* (see Algorithm 4), is in charge of applying the hyperheuristic principles, i.e., performing the mapping between the configurations and the slave islands.

In the standard island-based model, a global stop criterion is defined. In the proposed model, besides the global stop criterion, local stop criteria are fixed for the execution

Algorithm 3 Slave Islands Pseudocode

```
1: configureMigration()
2: while (not globalStopCriterion()) do
3:    lastConfig = ∅
4:    newConfig ← receiveConfiguration()
5:    if (newConfig != lastConfig) then
6:       initConfig(newConfig)
7:       lastConfig ← newConfig
8:       checkPopulation(α)
9:    end if
10:   while (not localStopCriterion()) do
11:      runGeneration()
12:      migrate()
13:   end while
14:   sendSolutions()
15: end while
16: sendBestSolution()
```

Algorithm 4 Master Island Pseudocode

```
1: initAdaptiveModel()
2: assignInitConfigsToIslands()
3: while (not globalStopCriterion()) do
4:    [island, config] ← checkForIdleIsland()
5:    if (island != NULL) then
6:       solutions[config] ← receiveSolution(island)
7:       scores ← updateScores(solutions)
8:       nextConf ← selectConfig(scores)
9:       assignConfig(nextConf, island)
10:      resumeExecution(island)
11:   end if
12: end while
13: receiveSolutions()
```

of the configurations on the islands. When a local stop criterion is reached—a quantum of time—the island execution is stopped. Based on the results achieved by the island, a score is assigned to the corresponding configuration. Based on such a score or quality indicator, the selection strategy is applied and the master selects the next configuration that should be executed on the idle island. If the new selected configuration is the same as the island current configuration, the local stop criterion is updated and the execution continues. Otherwise, the island configuration is updated and the changes performed by the algorithm over its subpopulation must be validated. In such a step, the model ensures that the individuals in the subpopulation has not worsen its objective value more than a fixed percentage value (α) along the last configuration run. If an individual does not verify the condition, the original individual is recovered. This step is necessary because unsuitable configurations could excessively degrade the population quality. Finally, when the global stop criterion is reached, every island sends its local solution to the master and the best one is selected as the final solution. Also, as in standard island models, the user must configure the migration scheme: the migration frequency, the number of individuals to migrate at each step, and the migration topology must be specified.

As stated in the algorithm description, the model has been centralized for the incorporation of the hyperheuristic principles. However this centralization does not strongly affects the scalability of the approach. The computational work associated to the tasks performed by the master island—the hyperheuristic selection procedure—is negligible when compared to the effort required by the slave islands—execution of the configurations.

The model has been implemented using the METCO (Metaheuristic-based extensible tool for cooperative optimisation) tool [43]. In order to improve the efficiency of the model, asynchronous communications for the migration scheme have been implemented. All the communications among the processes have been done using the message passing interface tool MPI.

5.1 Mapping scheme

One crucial issue for the correct operation of the model consists in performing a suitable mapping among the islands and configurations. The mapping process is managed by the hyperheuristic. Two different hyperheuristics has been incorporated into the parallel model. Both hyperheuristics are based on the use of choice functions. In parallel island-based models, probability schemes seem more promising than elitist schemes [41]. Therefore, a probability-based selection scheme [62] was incorporated.

The general behavior of both hyperheuristics is similar. First, a set of functions are used to assign a score to each configuration. Then, a probabilistic selection, based on the score of each configuration, is used to decide the next configurations that must be assigned to the idle islands. It is important to note that the behavior of the configurations can change along the different stages of the execution. Moreover, the stochastic behavior of the involved low-level (meta)-heuristics may lead to variations in the results achieved by each configuration. Therefore, it is appropriate to make some selections based on a completely random scheme. Both hyperheuristics can be tuned by means of the parameter β, which represents the minimum selection probability that should be assigned to a configuration. Thus, being n_h the number of involved (meta)-heuristics, a completely random selection is performed in $\beta * n_h$ percentage of the cases.

The first incorporated hyperheuristic (HH_imp) was previously used in [42]. The score assigned to each configuration estimates the improvement that each configuration can achieve, when breaking from the currently achieved solutions. In order to perform such a estimation the quality improvement achieved by each configuration is saved. Improvements are calculated when a configuration is stopped, i.e., when its local stop criterion is reached. The improvement is defined as the difference (in objective value) between the best achieved solution, and the best initial individual. Improvements obtained during the migration stage of the algorithm are discarded, obtaining the improvement imp. Considering a configuration conf, which has been executed j times, the score (s(conf)) is calculated as a weighted average of the last k improvements. The weighted average assigns greater importance to the last executions:

$$s(\text{conf}) = \frac{\sum_{i=1}^{k} i * \text{imp}[\text{conf}][j-i]}{\sum_{i=1}^{k} i}$$

The selection probability of the configuration conf (prob (conf)) is given by:

$$\text{prob}(\text{conf}) = \beta + (1 - \beta * n_h) * \left[\frac{s(\text{conf})}{\sum_{i=0}^{n_h} s(i)} \right]$$

In [56] it was shown that for some problems, given a set of configurations, a dynamic-mapped scheme could lead to better results than a static-mapped heterogeneous scheme and than any of the homogeneous schemes that could be constituted. It suggests that the combination of different strategies working at the same time produces additional benefits. The second incorporated hyperheuristic (HH_Syn) tries to detect synergies between pairs of configurations. It takes some ideas of the ant-based hyperheuristics [8]. In such a case, the hyperheuristic was applied in a sequential scheme, trying to detect how well a (meta)-heuristic operates after the application of another (meta)-heuristic. In our case, since it is incorporated in a parallel scheme, the hyperheuristic tries to detect how well a (meta)-heuristic operates in parallel with another (meta)-heuristic. HH_Syn assigns two different scores to each configuration. The first one, is called the visibility (vis) and represents the independent performance of each configuration. It is calculated as s in HH_imp. The second one, is called the cooperation between pairs (c_p) and represents the performance of a (meta)-heuristic in the presence of other metaheuristics. The improvements achieved along the execution by a metaheuristic m_1, executed in parallel with a metaheuristic $m2$, is saved in the set $imp[m_1][m_2]$. Given two metaheuristics m_1 and m_2, which have been executed in parallel j times, the score $c_p(m_1, m_2)$ is calculated as a weighted average of the last k improvements achieved by m_1, in the presence of m_2.

$$c_p(m_1, m_2) = \frac{\sum_{i=1}^{k} i * \text{imp}[m_1][m_2][j-i]}{\sum_{i=1}^{k} i}$$

Given a metaheuristic m_1 and the set of currently assigned metaheuristics $m\_set = \{h_1, h_2, \ldots, h_n\}$, the score $c_s(m_1)$ is calculated as the maximum c_p of any of its components, i.e., $c_s(m_1) = max\{c_p(m_1, h_1), c_p(m_1, h_2), \ldots, c_p(m_n, h_n)\}$.

When every island is idle, the hyperheuristic must grant the resources among the available configurations. The first assignment is performed as in HH_imp, but substituting s by

vis, i.e., the selection probability of the configuration conf is given by:

$$\text{prob(conf)} = \beta + (1 - \beta * n_h) * \left[\frac{\text{vis(conf)}}{\sum_{i=0}^{n_h} \text{vis}(i)} \right]$$

For the remaining assignments, the global cooperation c_s is also considered. c_s is used with a probability γ. Thus, considering that hh is the set of configurations assigned to any of the islands, the selection probability of the configuration $conf$ is given by:

$$\text{prob(conf)} = \beta + (1 - \beta * n_h - \gamma) * \left[\frac{\text{vis(conf)}}{\sum_{i=0}^{n_h} \text{vis}(i)} \right] + \gamma * \left[\frac{c_s(\text{conf})}{\sum_{i=0}^{n_h} c_s(i)} \right]$$

6 Experimental evaluation

6.1 Description

This section shows the results achieved for two different real-world instances of the FAP when using the sequential and parallel schemes here proposed. Tests have been run on a Debian GNU/Linux cluster of 8 nodes, each one consisting of two Intel(R) Xeon(TM) at 2.66 GHz and 1 Gb RAM. The interconnection network is a Gigabit Ethernet. The C++ compiler and MPI implementation used were *gcc 3.3* and MPICH *1.2.7*. The implementation of the algorithms was performed using METCO [43] (Metaheuristic-based extensible tool for cooperative optimisation).

Comparisons are performed considering two US cities instances: Seattle and Denver. The Seattle instance has 970 TRXs and 15 different frequencies to be assigned. The Denver instance is larger. It is constituted by 2612 TRXs and 18 frequencies. In both cases, the constants used in the mathematical formulation [44] were set to $K = 100{,}000$, $c_{\text{SH}} = 6$ dB, and $c_{\text{ACR}} = 18$ dB. These GSM networks are currently operating so finding their optimal plannings is of great practical interest. It is important to remark that the data source to build the interference matrix based on the C/I probability distribution uses thousands of Mobile Measurement Reports (MMRs) [37] rather than propagation prediction models. The M matrix contains 59,169 elements in the Seattle network, while it contains 20,638 elements for the Denver instance. The analysis is focused in detecting the advantages achieved by the incorporation of the hyperheuristics principles inside the parallel models. Nowadays it is common to have access to computers with several cores. Quad-core computers or even eight-core are accessible for most of the researchers and enterprises. Since the master operates when, at least, one of the slave islands is idle, it can share the computational resources with them. Therefore, parallel executions have been performed using 4 and 8 slave islands.

A set of experiments were executed for each instance in order to test the behavior of the different approaches. For each kind of execution, 30 repetitions were performed and average values considered. Since the tested algorithms are stochastic, in order to provide the results with confidence a suitable statistical analysis must be performed. The statistical comparisons has followed the guidelines presented in [17,57]. First a Kolmogorov–Smirnov test is performed in order to check whether the values of the results follow a normal (gaussian) distribution or not. If so, the Levene test checks for the homogeneity of the variances. If samples have equal variance (positive Levene test), an ANOVA test is done; otherwise a Welch test is performed. For non-gaussian distributions, the non-parametric Kruskal–Wallis test is used to compare the medians of the algorithms. A confidence level of 95% is considered (i.e., significance level of 5% or p value under 0.05), which means that the differences are unlikely to have occurred by chance with a probability of 95%.

6.2 Computational results

First experiment performs a comparison of the costs of the frequency plans obtained by a set of sequential configurations and by the proposed parallel approach with both explained hyperheuristics. The configuration of the memetic algorithm parameters was as follows: $InitPSize = 2$, $SoftBloq = 50$, $HardBloq = 300$, $MaxPopSize = 5$. Many configurations can be constituted by using the set of defined mutation operators and by tuning their internal parameters. A set of 30 sequential configurations were executed and analyzed. They were defined by uniformly dividing the ranges of accepted values. The set of mutation operator configurations was the following:

- UM with $p_m = \{0.1, 0.3, 0.5, 0.7, 0.9\}$
- MM with $p_m = \{0.1, 0.3, 0.5, 0.7, 0.9\}$
- NM with $(p_m, N) = \{(0.1, 1), (0.3, 1), (0.5, 1), (0.7, 1), (0.9, 1), (0.1, 3), (0.3, 3), (0.5, 3), (0.7, 3), (0.9, 3), (0.1, 5), (0.3, 5), (0.5, 5), (0.7, 5), (0.9, 5), (0.1, 7), (0.3, 7), (0.5, 7), (0.7, 7), (0.9, 7)\}$

The presented parallel model has been executed using the hyperheuristics HH_Imp and HH_Syn, and using the 30 described configurations as low-level meta-heuristics. In both hyperheuristics the next parameterization was used: $\alpha = 0.5\%$, $\beta = \frac{0.2}{30}$ and $k = 5$. In HH_Syn γ was fixed to 0.4. Parallel executions were run using 4 and 8 slave islands. They are referred as HH_Imp_4, HH_Imp_8, HH_Syn_4 and HH_Syn_8. Every sequential and parallel execution was performed with a stop criterion of 1 h. For the parallel executions the local stop criteria was fixed to 1 min, when 4 slave

islands were used, and to 2 min for the executions with 8 slave islands. Migration was performed following an asynchronous scheme with a migration probability of 1. The topology consisted in an all to all connected structure. Migrated individuals are selected following an elitist scheme, i.e., the best individual is selected to migrate. Replacements were performed also following an elitist scheme. They only take place when the migrated individual is better than any of the individuals in the new island. Since only one individual is selected a good diversity is maintained among the islands.

Considering the obtained results, sequential algorithms were ordered based on the mean cost achieved at the end of the executions. An index based on such an order is assigned to each configuration. Therefore, for each instance, the best sequential execution will be referred as "seq1", while the worst one will be referred as "seq30". Generally the behavior of a configuration depends on the instance. Table 1 shows the best configurations for the Seattle instance, and its corresponding index for the Denver instance. Most of the best configurations are suitable for both instances. However, some of them are not adequate, so they would produce a waste of resources if applied to the other instance. Thus, the robustness of the scheme can be improved by applying the parallel hyperheuristic-based approach. Moreover, some of the best configurations correspond to high values of p_m, while other ones correspond to low values. Thus, it is very difficult to know, a-priori, which configurations are suitable for a given instance.

Table 1 Robustness of sequential configurations

Configuration	Seattle index	Denver index
NM (0.7, 7)	1	2
NM (0.9, 5)	2	1
NM (0.3, 3)	3	12
NM (0.5, 7)	4	5
NM (0.7, 5)	5	3
UM (0.1)	6	19
NM (0.5, 5)	7	17

Figure 3 displays, for both instances, the evolution of the mean cost achieved by both parallel hyperheuristic-based models, "seq1", "seq15", and "seq30". In both instances the cost achieved by HH_Imp_4 and HH_Syn_4 methods are very similar to the one achieved by the best sequential approach. In the Seattle network, "seq1" is slightly better than the parallel approaches, while in the Denver network, the best results are achieved by the parallel schemes. Therefore, the parallel approaches, even with so few processors, can be used in order to avoid the testing of each one of the low-level configurations. Costs values achieved by "seq15" and "seq30" are clearly worse than the ones achieved by the parallel models. Results achieved by HH_Syn_4 slightly improve the ones obtained by HH_Imp_4, but they are not statistically different. Executions with 8 slave islands produce much betters results. Differences between the models with 4 islands, and

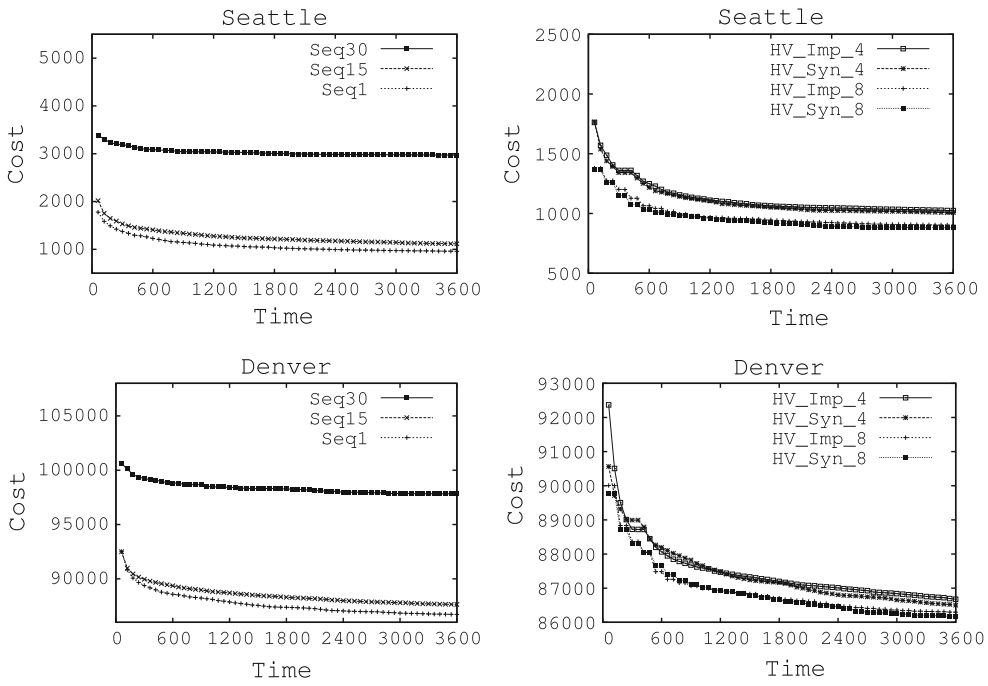

Fig. 3 Evolution of the cost function for Seattle and Denver networks

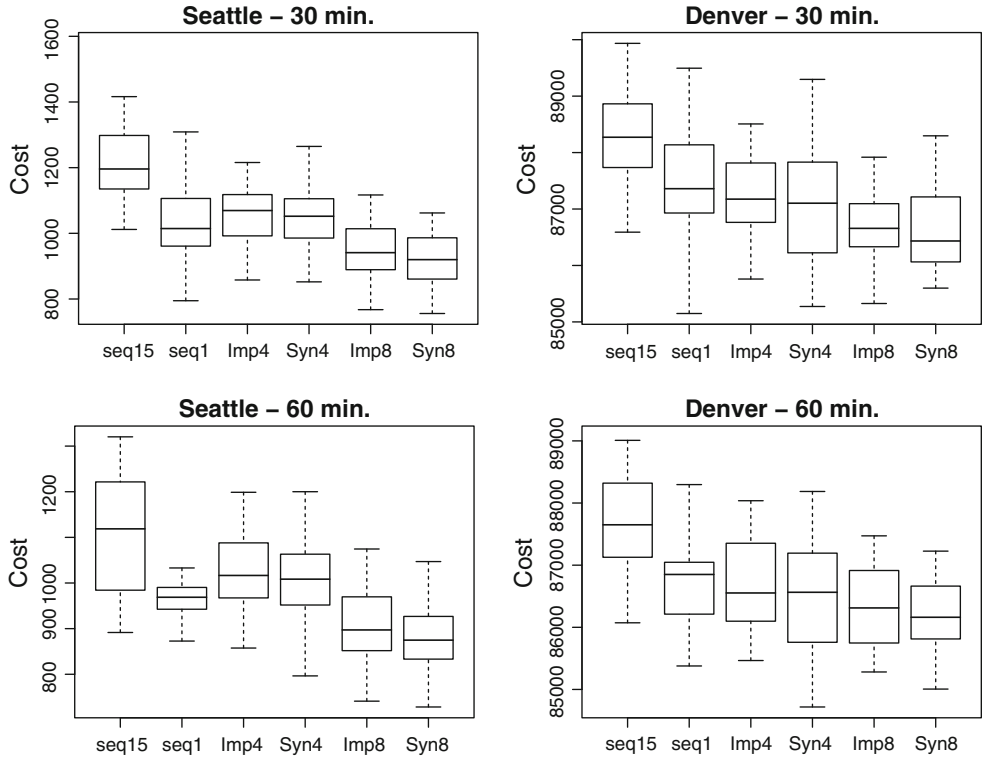

Fig. 4 Box-plots of the achieved costs

Table 2 Statistical analysis fixing the execution time

	Seattle			Denver		
	↑	↔	↓	↑	↔	↓
HH_Imp$_4$	7	6	17	0	3	27
HH_Syn$_4$	7	3	20	0	2	28
HH_Imp$_8$	0	0	30	0	0	30
HH_Syn$_8$	0	0	30	0	0	30

the ones with 8 islands are significant. Therefore, they allow to avoid the testing of each one of the low-level configurations, and to speed up the obtaining of high quality network configurations. In both networks, the mean cost achieved by HH_Syn$_8$ at the end of the executions improves the one achieved by HH_Imp$_8$, but differences are not statistically significant.

Figure 4 offers a more detailed information about the cost achieved by each scheme, using as stop criterion 30 min and 1 h. In both instances, the diagram shows the similarity between "seq1" and the parallel models using 4 slave islands. The parallel models with 8 slave islands clearly improve the results achieved by any other model. Also, it shows the benefits of HH_Syn, when compared to HH_Imp.

Table 2 shows, for both instances, the number of sequential configurations which are better (↑), not different (↔), or worse (↓) than the corresponding row configuration. The comparison is performed in terms of the achieved fitness when considering a stop criterion of 1 h. It shows the better adaptation of the hyperheuristic to the Denver instance, than to the Seattle instance. In both instances, the parallel approach with 4 slave islands is better than most of the sequential configurations. When 8 slave islands are incorporated, the parallel approaches perform better than any sequential approach. Table 3 shows the same information as Table 2, but in this case the amount of resources used are fixed. Thus, the parallel approaches using 4 islands and a stop criterion of 1 h were compared with sequential executions of 4 h, and the parallel ones with 8 slave islands, were compared with sequential approaches of 8 hours. With such a comparison we can detect the number of sequential configurations which make a better usage of the resources than the parallel schemes. In the case of the Seattle instance, about 10 configurations make a worse usage of the resources, i.e., a superlinear speedup is expected when compared with them. In the case of the Denver instance that number is increased to 15. Once again, it shows the better adaptation of the parallel approaches to the Denver network.

The previous experiment has compared the schemes, mainly focused in terms of the achieved quality. However, since the parallel executions use more computational resources than the sequential ones, the improvement achieved by the parallel model must be quantified. In order to measure

Table 3 Statistical analysis fixing the amount of resources

	Seattle			Denver		
	↑	↔	↓	↑	↔	↓
HH_Imp_4	19	3	8	9	6	15
HH_Syn_4	18	2	10	10	5	15
HH_Imp_8	19	4	7	9	6	15
HH_Syn_8	18	3	9	10	5	15

the improvement of the parallel approach, the second experiment analyzes the run-time behavior of the parallel proposed models. The ideas presented in [30] were followed. Each sequential configuration, as well as the parallel models, were executed using as finalization condition the achievement of a certain level of quality. The quality level was established so that HH_Imp_4, i.e., the worst parallel approach, would require 30 min in average, in order to reach it. Since some of the sequential configurations are not able to reach such a quality level, a second stopping criterion, consisting in the execution of a maximum time of 10 h was also considered. Thus, the success ratio is defined as the probability of achieving the required quality level, considering a limitation in the execution time of 10 h. Figure 5 shows, the run length distribution—success ratio versus time—for the parallel models, and for the best behaved sequential configuration when applied to the Seattle network. It shows the similarity among the run length distribution of the hiperheuristic-based methods using 4 slave islands, and the best sequential configuration. Thus, such parallel approaches and the best sequential configuration requires similar times to converge to a plan with the considered quality. The parallel schemes with 8 slave islands are able to obtain such a quality level in less time. Figure 6 shows the same information for the Denver network. In this case, the hyperheuristics-based methods are even able to achieve better results than the best sequential configuration. Therefore, for both instances it is possible to make use of any of the parallel approaches in order to obtain good quality solutions. In fact, by using only 4 processors, the success ratio achieved by the parallel models is similar—or even better—than the ones achieved by the best sequential approach. It shows the good behavior of the parallel approaches, specially considering the large set of low-level metaheuristics incorporated in the model.

For the remaining configurations a summary of the analysis is shown. Table 4 shows, for the Seattle network, the success ratio and the average speedup of the parallel models versus a set of selected sequential configurations. The speedup is marked with a line in the cases in which the sequential configurations were not able to achieve a success ratio greater than 50%. These speedups are calculated based only on the executions which achieved the considered quality

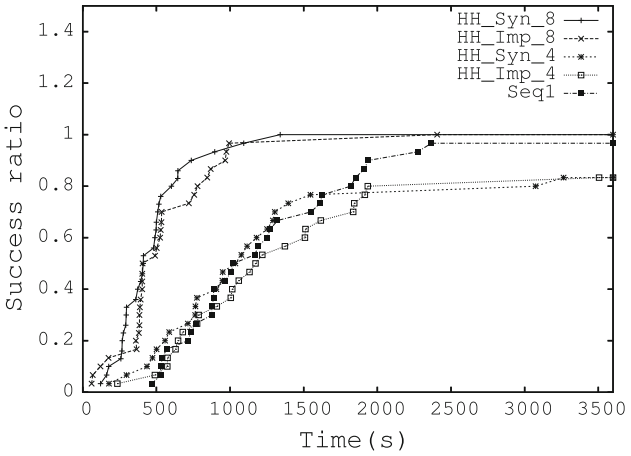

Fig. 5 Run length distribution for the Seattle network

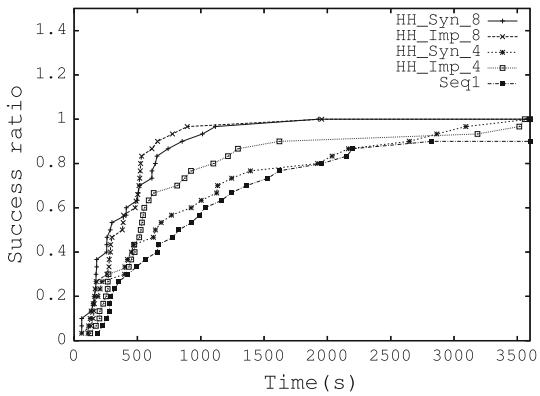

Fig. 6 Run length distribution for the Denver network

level. Therefore, shown speedup values are lower approximations of the real ones. In order to calculate the real values, much longer executions should be performed for the worst-behaved configurations. Although linear speedup is not achieved when comparing with the best configuration, it must be taken into account that when solving a problem, the best configuration is not known a priori, so, the time saving is much greater than the speedup calculated versus the best configuration. In fact, the speedup highly increases when comparing to other configurations. Moreover, the addition of more resources produce faster convergence to high-quality results. The speedup achieved by the models which use 8 slave islands is about the double of the models which use 4 slave island. Thus, with the addition of the first resources the testing of each low-level configuration can be avoided; and with the addition of more resources, the speedup can be improved. Table 5 shows the same information for the Denver network. The behavior is very similar to the one detected in the Seattle instance. Also, it is important to note that for both instances, the parallel model based in HH_Syn has been able to obtain greater speedups than the model using HH_Imp.

Table 4 Speedup of the parallel models in the Seattle network

Config.	HH_Imp_4 speedup	HH_Syn_4 speedup	HH_Imp_8 speedup	HH_Syn_8 speedup	Success ratio (%)
seq1	0.9	1.1	2	2.4	100
seq5	1	1.2	2.2	2.6	100
seq10	2.3	2.6	4.8	5.8	100
seq15	3.8	4.3	8	9.6	100
seq20	6.3	7.1	13.2	15.9	100
seq25	–	–	–	–	0

Table 5 Speedup of the parallel models in the Denver network

Config.	HH_Imp_4 speedup	HH_Syn_4 speedup	HH_Imp_8 speedup	HH_Syn_8 speedup	Success ratio (%)
seq1	1	1.2	2.4	2.4	90
seq5	1.1	1.3	2.6	2.6	90
seq10	2.3	2.8	5.4	5.4	90
seq15	2.4	3	5.8	5.8	90
seq20	11.4	13.9	26.8	26.8	60
seq25	–	–	–	–	0

Although the improvement of the proposed model compared to the sequential model has been demonstrated, in order to further check its validity, it must also be compared with other PEAs. In order to perform such a comparison, some additional parallel executions were carried out. The parameters for such executions were the same as the ones used in the above parallel experiments but in this case, instead of using the hyperheuristic-based model, a homogeneous island-based model was defined for each one of the 30 configurations. Each model used 4 slave islands. The cost values obtained in 30 and 60 min are used in the comparison. Table 6 shows, for the Seattle network, the number of homogeneous schemes which are better, worst or not significantly different from the proposed models. As shown in the table, less than half of the proposed homogeneous models are better than the hyperheuristic-based approaches. Table 7 shows the same information for the Denver network. In this case, the new proposed model is always among the best schemes. It shows again the better adaptation of the proposed method to the Denver instance. Since the adaptation scheme requires some time to detect how the low-level metaheuristics are behaving, the model performs better for longer executions. Results show that the number of homogeneous models which are worse than our model increases with the execution time in every case. Moreover, for both studied instances, the hyperheuristic-based methods have shown their stability, allowing the user to obtain very acceptable results without the requirement of testing a large set of different PEAs.

Also, it is important to check the suitability of the selection scheme. The proposed parallel models were compared

Table 6 Quality comparison of the parallel models for the Seattle network

Model	Better		Worse		Not differ	
	30 m	60 m	30 m	60 m	30 m	60 m
HH_Imp_4	14	14	10	11	6	5
HH_Syn_4	14	14	11	12	5	4

Table 7 Quality comparison of the parallel models for the Denver network

Model	Better		Worse		Not differ	
	30 m	60 m	30 m	60 m	30 m	60 m
HH_Imp	8	8	15	16	7	6
HH_Syn	8	5	13	17	9	8

with a strategy which randomly changes the configurations executed on the islands. Such strategy has been denoted as 4-uniform. The involved configurations, migration scheme and stopping criteria were identical to the ones used in the first experiment. Tables 8 and 9 show the best, worst, average and median of the costs achieved by HH_Imp_4, HH_Syn_4, and 4-uniform approaches when applied to Seattle and Denver instances, respectively. In every case, the average and median costs achieved by the new proposed models are better than the ones achieved by making a random mapping. Moreover, the statistical comparison of 4-uniform with HH_Imp_4 and with HH_Syn_4 shows the superiority of the hyperheuristic-based approaches. As shown, the incorporated hyperheuristic strategies produces an important improvement when compared to random selection schemes.

As stated in the paper, the analyzed GSM networks are currently operating so finding their optimal plannings is of great practical interest. They have been analyzed in several papers [44,45]. The best results have been obtained by using "seq1", with a stop criterion of 10 h. Note that parallel approaches have only been executed with more restricted stopping criteria. For the Denver instance, the best obtained plan has 84,548.9 interference units. Since our knowledge it has only been improved by the plans obtained by using grid systems [46]. In such a case, executions with 300 processors were performed. In the case of the Seattle instance, it was produced a plan of 654.53 units. Since our knowledge it is the best known frequency plan.

7 Conclusions and future work

This paper has presented a set of approaches used to deal with the FAP. Relevant aspects of real-world GSM networks has been considered in the mathematical formulation of the FAP. In previous works, a memetic algorithm with an increasing

Table 8 Quality comparison of the hyperheuristic-based models with a random scheme for the Seattle network

	4-uniform				HH_Syn_4				HH_Imp_4			
	Best	Worst	Mean	Median	Best	Worst	Mean	Median	Best	Worst	Mean	Median
30 min	951	1,316	1,138	1,149	852	1,264	1,051	1,052	857	1,314	1,063	1,069
60 min	949.5	1,278	1,074	1,067	796	1,200	1,007	1,008	857	1,198	1,024	1,016

Table 9 Quality comparison of the hyperheuristic-based models with a random scheme for the Denver network

	4-uniform				HH_Syn_4				HH_Imp_4			
	Best	Worst	Mean	Median	Best	Worst	Mean	Median	Best	Worst	Mean	Median
30 min	86,849	89,174	87,790	87,756	85,273	89,296	87,162	87,103	85,760	88,507	87,198	87,175
60 min	86,390	88,620	87,224	87,201	84,718	88,186	86,502	86,564	85,465	88,037	86,677	86,553

population size had been designed to deal with such a version of the FAP. The algorithm combines a modified evolutionary algorithm with a $(1 + 1)$ selection operator and a specifically designed local search. A theoretical analysis of the local search has been performed, showing the cases in which optimal assignments are performed. In previous researches, promising results were achieved by using such a memetic algorithm. However, it had to be adapted for each solved instance. Particularly, the internal parameters of the variation operators had to be fixed. Thus, previously to solving a problem instance, an algorithm tuning step had to be performed.

A novel parallel approach, based on hybridizing hyperheuristics and the island-based model, has been applied to the FAP. The proposal adds an adaptive property to the well known island-based model by applying the operation principles of the hyperheuristics. The model combines a set of low-level metaheuristics in an intelligent way, granting more computational resources to those configurations that show a more promising behavior. In the considered case, the low-level metaheuristics are configurations of the memetic algorithm, which use different sets of internal parameters and variation operators. Specifically, 30 different configurations have been used. Moreover, new variation operators has been designed and tested. In order to perform the assignment of the resources, two different hyperheuristics have been incorporated inside the model. Both hyperheuristics are based on the use of choice functions and probabilistic selections strategies. The first hyperheuristic (HH_Imp) is based on estimating the improvement that each configuration can achieve, when breaking from the currently achieved solutions. The second hyperheuristic (HH_Syn) tries to detect synergies between pairs of configurations, i.e. it analyzes the behavior of the low-level metaheuristics when they are executed in parallel with other metaheuristics.

Results achieved for two real-world networks demonstrate the validity of the proposed scheme. Both networks are currently operating in US cities, so finding their optimal plannings is of great practical interest. The new designed schemes made possible to improve the previously known best frequency plans for the Seattle instance. In the case of the Denver instance, it has been able to improve the results attained by sequential executions of more than 5 days [46]. However, it was not able to improve the quality of the plans obtained by using large grid systems [46]. The experiments compare the proposed model with the low-level sequential algorithms, and with a set of parallel approaches: a homogeneous island-based model for each considered sequential approach. Results obtained by the designed approach with 4 slave islands are similar to the ones obtained by the best sequential scheme. Therefore, even with so few processors, the hyperheuristic-based approaches can be used in order to avoid the tuning step of the memetic algorithm. Thus, the new model provides high-quality solutions without forcing the user to have a prior knowledge about the behavior of the different configurations when applied to a given instance. Tests with 8 slave islands have allowed to speedup the obtaining of high quality solutions. Thus, the parallel model can be used to improve the quality of the achieved solutions, or to obtain similar solutions in fewer time. Results obtained by both hyperheuristics are similar in terms of the quality achieved at the end of the executions. However, HH_Syn has shown a slightly better behavior. In order to confirm the suitability of the resource assignment performed by the hyperheuristics, they were compared with a model which randomly distributes the resources among the low-level metaheuristics. The frequency plans obtained by the new scheme with any of the incorporated hyperheuristics improve the ones reached by performing a random mapping.

Future work targets the incorporation of some other modern metaheuristics inside the proposed approach to deal with the FAP. Mixing evolutionary and classical optimization approaches, in the same parallel scheme, can also produce many benefits. Since it has been shown the importance of customizing the variation operators, in order to achieve good quality solutions, it would be interesting to incorporate new crossover and mutation operators. Alternative hyperheuristic strategies can be proposed and a deeper analysis with them can be performed. Also, it would be interesting to execute the parallel model in a larger cluster, or in a grid. Thus, an scalability analysis can be performed. In order to confirm the superiority of HH_Syn in relation to HH_Imp, it would be be interesting to apply the parallel model to some other real-world problems, specially to those involving high-cost evaluation functions.

Acknowledgments This work was supported 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 TIN2008- 06491- C04- 02 and by Canary Government project number PI2007/015. The work of Carlos Segura was funded by grant FPU- AP2008- 03213.

References

1. Aardal KI, van Hoesel SPM, Koster AMCA, Mannino C, Sassano A (2007) Models and solution techniques for frequency assignment problems. Ann Oper Res 153(1):79–129
2. Alba E (2005) Parallel metaheuristics: a new class of algorithms. Wiley-Interscience, London
3. Amaldi E, Capone A, Malucelli F, Mannino C (2006) Handbook of optimization in telecommunications, chap optimization problems and models for planning cellular networks. Springer, Berlin, pp 917–939
4. Araya I, Neveu B, Riff MC (2008) An efficient hyperheuristic for strip-packing problems. In: Cotta C, Sörensen K (eds) Adaptive and multilevel metaheuristics,studies in computational intelligence vol 136. Springer, Berlin, pp 61–76
5. Avenali A, Mannino C, Sassano A (2002) Minimizing the span of d-walks to compute optimum frequency assignments. Math Program A 91:357–374
6. Bader-El-Den MB, Poli R, Fatima S (2009) Evolving timetabling heuristics using a grammar-based genetic programming hyperheuristic framework. Memetic Comput 1(3):205–219
7. Bjorklund P, Varbrand P, Yuan D (2005) Optimized planning of frequency hopping in cellular networks. Comput Oper Res 32(1):169–186
8. Burke E, Kendall G, Silva JL, O'Brien R, Soubeiga E (2005) An Ant algorithm hyperheuristic for the project presentation scheduling problem. In: Proceedings of the 2005 IEEE congress on evolutionary computation (CEC 2005), vol 3. Edinburgh, Scotland, pp 2263–2270
9. Burke EK, Kendall G, Newall J, Hart E, Ross P, Schulenburg S (2003) Handbook of Meta-heuristics. Hyper-heuristics: an emerging direction in modern search technology. Kluwer, Dordrecht
10. Burke EK, Kendall G, Soubeiga E (2003) A Tabu-search hyperheuristic for timetabling and rostering. J. Heuristics 9(6): 451–470
11. Burke EK, McCollum B, Meisels A, Petrovic S, Qu R (2007) A graph-based hyper-heuristic for educational timetabling problems. Eur J Oper Res 176(1):177–192
12. Cantú-Paz E (1998) A survey of parallel genetic algorithms. Calc Paralleles 10
13. Chen PC, Kendall G, Vanden Berghe G (2007) An ant based hyperheuristic for the travelling tournament problem. In: Proceedings of IEEE symposium of computational intelligence in scheduling (CISched 2007). Honolulu, Hawaii, pp 19–26
14. Cowling P, Kendall G, Han L (2002) 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). IEEE Computer Society, Honolulu Hawaii, pp 1185–1190
15. Cowling P, Kendall G, Soubeiga E (2001) A parameter-free hyperheuristic for scheduling a sales summit. In: Proceedings of 4th metahuristics international conference (MIC 2001). Porto Portugal, pp 127–131
16. Cowling PI, Kendall G, Soubeiga E (2002) Hyperheuristics: a robust optimisation method applied to nurse scheduling. In: Guervós JJM, Adamidis P, Beyer HG, Martín JLFV, Schwefel HP (eds) PPSN lecture notes in computer science, vol 2439. Springer, Berlin, pp 851–860
17. Demšar J (2006) Statistical comparison of classifiers over multiple data sets. J Machine Learn Res 7:1–30
18. Dorigo M, Maniezzo V, Colorni A (1996) The ant system: Optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern Part B 26(1):29–41
19. Dowsland K, Soubeiga E, Burke E (2007) A simulated annealing hyper-heuristic for determining shipper sizes. Eur J Oper Res 179(3):759–774
20. Eisenblätter A (2001) Frequency assignment in GSM networks: Models, heuristics, and lower bounds. Ph.D. thesis, Technische Universität Berlin
21. Eisenblätter A, Grötschel M, Koster AMCA (2002) Frequency planning and ramifications of coloring. Discuss Math Graph Theory 22(1):51–88
22. Fischetti M, Lepsch C, Minerva G, Romanin-Jacur G, Toto E (2000) Frequency assignment in mobile radio systems using branch-and-cut techniques. Eur J Oper Res 123(2):241–255
23. Furuskar A, Naslund J, Olofsson H (1999) EDGE—enhanced data rates for GSM and TDMA/136 evolution. Ericsson Rev (1)
24. Garg P (2009) A comparison between memetic algorithm and genetic algorithm for the cryptanalysis of simplified data encryption standard algorithm. Int J Netw Secur Appl 1(1):34–42
25. Glover F (1998) A template for scatter search and path relinking. In: AE '97: Selected papers from the third european conference on artificial evolution. Springer, London 3–54
26. Granbohm H, Wiklund J (1999) GPRS—general packet radio service. Ericsson Rev (1)
27. Gratch J, Chien S (1993) Learning search control knowledge for the deep space network scheduling problem. Tech. rep., Champaign, IL, USA
28. Greff JY, Idoumghar L, Schott R (2004) Application of markov decision processes to the frequency assignment problem. Appl Artif Intell 18(8):761–773
29. Hale WK (1980) Frequency assignment: theory and applications. Proc IEEE 68(12):1497–1514
30. Hoos HH (1999) On the run-time behavior of stochastic local search algorithms for SAT. In: Proceedings of AAAI'99. MIT Press, pp 661–666
31. Idoumghar L, Schott R (2006) A new hybrid GA-MDP algorithm for the frequency assignment problem. In: Proceedings of the 18th IEEE international conference on tools with artificial intelligence (ICTAI'06), pp 18–25

32. Jaumard B, Marcotte O, Meyer C (1999) Telecommunications network planning, chap, mathematical models and exact methods for channel assignment in cellular networks. Kluwer, UK, pp 239–256
33. Kendall G, Cowling P, Soubeiga E (2002) Choice function and random hyperheuristics. In: Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning (SEAL 2002). Singapore, pp 667–671
34. Kendall G, Mohamad M (2004) Channel assignment in cellular communication using a great deluge hyper-heuristic. In: Proceedings of the 2004 IEEE international conference on networks (ICON). Singapore, pp 769–773
35. Kendall G, Mohamad M (2004) Channel assignment optimisation using a hyper-heuristic. In: Proceedings of the 2004 IEEE conference on cybernetics and intelligent systems (CIS 2004). Singapore, pp 790–795 (2004)
36. Kim SS, Smith AE, Lee JH (2007) A memetic algorithm for channel assignment in wireless FDMA systems. Comput Oper Res 34:1842–1856
37. Kuurne AMJ (2002) On GSM mobile measurement based interference matrix generation. In: IEEE 55th vehicular technology conference. VTC Spring 2002, pp 1965–1969
38. Le MN, Ong YS, Jin Y, Sendhoff B (2009) Lamarckian memetic algorithms: local optimum and connectivity structure analysis. Memetic Comput 1(3):175–190
39. Leese R, Hurley S (2002) Methods and algorithms for radio channel assignment. Oxford lecture series in mathematics and its applications. Oxford University Press, New York
40. León C, Miranda G, Segredo E, Segura C (2008) Parallel hypervolume-guided hyperheuristic for adapting the multi-objective evolutionary island model. In: International workshop on nature inspired cooperative strategies for optimization, studies in computational intelligence. Springer, Berlin
41. León C, Miranda G, Segura C (2007) Parallel skeleton for multi-objective optimization. In: Genetic and evolutionary computation conference. ACM, London, p 906
42. Leon C, Miranda G, Segura C (2009) A memetic algorithm and a parallel hyperheuristic island-based model for a 2d packing problem. In: GECCO'09: Proceedings of the 11th annual conference on genetic and evolutionary computation. ACM, New York, pp 1371–1378
43. León C, Miranda G, Segura C (2009) METCO: a parallel plugin-based framework for multi-objective optimization. Int J Artif Intell Tools 18(4)
44. Luna F, Blum C, Alba E, Nebro AJ (2007) ACO vs EAs for solving a real-world frequency assignment problem in GSM networks. In: Genetic and evolutionary computation conference (GECCO 2007), pp 94–101
45. Luna F, Estébanez C, León C, Chaves-González JM, Alba E, Aler R, Segura C, Vega-Rodríguez MA, Nebro AJ, Valls JM, Miranda G, Gómez-Pulido JA (2008) Metaheuristics for solving a real-world frequency assignment problem in gsm networks. In: Conference on genetic and evolutionary computation (GECCO 2008), pp 1579–1586
46. Luna F, Nebro AJ, Alba E, Durillo JJ (2008) Solving large-scale real-world telecommunication problems using a grid-based genetic algorithm. Eng Optim 40(11):1067–1084
47. Mannino C, Oriolo G, Ricci F, Chandran S (2007) The stable set problem and the thinness of a graph. Oper Res Lett 35(1):1–9
48. Mannino C, Sassano A (2003) An enumerative algorithm for the frequency assignment problem. Discrete Appl Math 129:155–169
49. Matsui S, Watanabe I, Tokoro KI (2005) Application of the parameter-free genetic algorithm to the fixed channel assignment problem. Syst Comput Jpn 36(4):71–81
50. Metzger BH (1970) Spectrum management technique. In: 38th national ORSA meeting
51. Mouly M, Paulet MB (1992) The GSM system for mobile communications. Mouly et Paulet, Palaiseau
52. Ong YS, Lim MH, Zhu N, Wong KW (2006) Classification of adaptive memetic algorithms: a comparative study. IEEE Trans Syst Man Cybern Part B 36(1):141–152
53. Press WH, Flannery BP, Teukolsky SA, Vetterling WT (1992) Numerical recipes in C: the art of scientific computing. Cambridge University Press, Cambridge
54. Rapeli J (1995) UMTS: targets, system concept, and standardization in a global framework. IEEE Pers Commun 2(1):30–37
55. Salcedo-Sanz S, Bousoño-Calzón C (2005) A portable and scalable algorithm for a class of constrained combinatorial optimization problems. Comput Oper Res 32:2671–2687
56. Segura C, Cervantes A, Nebro AJ, Jaraz-Simn MD, Segredo E, Garca S, Luna F, Gmez-Pulido JA, Miranda G, Luque C, Alba E, Vega-Rodrguez MA, Len C, Galvfn I (2009) Optimizing the DFCN broadcast protocol with a parallel cooperative strategy of multi-objective evolutionary algorithms. In: Springer (ed) 5th international conference devoted to evolutionary multi-criterion optimization, vol 5467. Nantes, France pp 305–319
57. Sheskin DJ (2003) Handbook of parametric and nonparametric statistical procedures. CRC Press, Boca Raton
58. Simon MK, Alouini MS (2005) Digital communication over fading channels: a unified approach to performance analysis. Wiley, London
59. Talbi EG (2006) Parallel Combinatorial Optimization (Wiley Series on Parallel and Distributed Computing). Wiley-Interscience, London
60. Terashima-Marn H, Ross P (1999) Evolution of constraint satisfaction strategies in examination timetabling. In: Proceedings of the genetic and evolutionary computation conference (GECCO99). Morgan Kaufmann, pp 635–642
61. Veldhuizen DAV, Zydallis JB, Lamont GB (2003) Considerations in engineering parallel multiobjective evolutionary algorithms. IEEE Trans Evol Comput 7(2):144–173
62. Vink T, Izzo D (2007) Learning the best combination of solvers in a distributed global optimization environment. In: Proceedings of advances in global optimization: methods and applications (AGO). Mykonos, Greece, pp 13–17
63. Voudouris C, Tsang E (1999) Guided local search. Eur J Oper Res 113(2):449–499
64. Walke BH (2002) Mobile radio networks: networking, protocols and traffic performance. Wiley, London
65. Wolpert D, Macready W (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82