Optimizing a realistic large-scale frequency assignment problem using a new parallel evolutionary approach

Jose Manuel Chaves-González, Miguel A. Vega-Rodriguez, Juan Antonio Gómez-Pulido, Juan Manuel Sánchez-Pérez

Optimizing a realistic large-scale frequency assignment problem using a new parallel evolutionary approach

Journal:	*Engineering Optimization*
Manuscript ID:	GENO-2010-0038.R3
Manuscript Type:	Review
Date Submitted by the Author:	28-Jul-2010
Complete List of Authors:	Chaves-González, Jose; University of Extremadura, Technologies of Computers and Communications Vega-Rodriguez, Miguel A.; University of Extremadura, Technologies of Computers and Communications Gómez-Pulido, Juan; University of Extremadura, Technologies of Computers and Communications Sánchez-Pérez, Juan; University of Extremadura, Technologies of Computers and Communications
Keywords:	parallel hyper-heuristic, frequency assignment problem, parallel heuristic based on metaheuristics, realistic frequency planning

Optimizing a realistic large-scale frequency assignment problem using a new parallel evolutionary approach

José M. Chaves-González*, Miguel A. Vega-Rodríguez, Juan A. Gómez-Pulido, Juan M. Sánchez-Pérez

Department of Technologies of Computers and Communications, University of Extremadura, Escuela Politécnica, Av. Universidad s/n, 10071, Cáceres, Spain

*Corresponding author. Email: jm@unex.es

(Received 21 February 2010; final version received 28 July 2010)

This article analyzes the use of a novel parallel evolutionary strategy to solve complex optimization problems. The work developed here has been focused on a relevant real-world problem from the telecommunication domain to verify the effectiveness of the approach. The problem, known as frequency assignment problem (FAP), basically consists of assigning a very small number of frequencies to a very large set of transceivers used in a cellular phone network. Real data FAP instances are very difficult to solve due to the NP-hard nature of the problem, therefore using an efficient parallel approach which takes the most of different evolutionary strategies can be considered as a good way to obtain high-quality solutions in short periods of time. Specifically, a parallel hyper-heuristic based on several meta-heuristics has been developed. After a complete experimental evaluation, results prove that the proposed approach obtains very high-quality solutions for the FAP and beats any other result published.

Keywords: parallel hyper-heuristic; frequency assignment problem; realistic frequency planning; parallel heuristic based on metaheuristics

1. Introduction

One of the most relevant optimization problems in the telecommunication field is the frequency planning problem. This problem, also known as the frequency assignment problem (FAP) or automatic frequency planning (AFP) is very important nowadays, because cellular phone communications are widely used nowadays, and presumably they will be also relevant in the near and middle future. The problem basically consists of assigning a given and very small number of fixed frequencies (usually no more than 20) to a very large set of transceivers, or TRXs, (there are thousands of them in real networks). The frequencies have the purpose of making communications in the network possible, taking the information from one point to another, whereas the TRXs, which are located in the antennas, give support to all the communications produced. The optimization problem arises because there are too few frequencies to give support to the enormous number of communications produced in a realistic network, therefore, frequencies have to be reused lots of times and this fact causes interferences that have to be minimized to maintain a high quality of service.

The frequency assignment problem is a very well known problem in the literature (FAP web 2007, Aardal *et al*. 2007, Eisenbläter 2001, Kotrotsos *et al.* 2001), however most of the works published are based on benchmark problems (FAP web 2007), and the authors' work is based on real and accurate interference information taken from the telecommunications industry. Therefore, it considers the requirements of realistic GSM (Global System for Mobile communication) networks (Mouly and Paulet 1992, Eisenbläter 2001). Moreover, it is important to remark that the FAP in GSM networks has, even nowadays, a great social impact, because this technology is the most used mobile communication system around the world. In fact, by mid 2009, GSM services were used in

approximately 220 countries and territories serving more than three billion people and providing travellers with access to mobile services wherever they go (GSM World 2010).

In addition to the engineering difficulties of dealing with a real-world problem (it is necessary to consider all possible sources of interference, regulatory concerns, technological limitations, etc. (Eisenbläter 2001)), FAP is also an NP-hard problem (Hale 1980). For this reason, real data instances of the problem are really difficult to solve and although some exact algorithms have been proposed to deal with the FAP (*e.g.* Fischetti *et al.* 2000, Mannino and Sassano 2003), they are not feasible when tackling large instances of the problem (Eisenbläter 2001, Aardal *et al*. 2007). On the other hand, heuristics and metaheuristics methods have proved to be one of the best strategies for solving this kind of optimization problems (Burke and Kendall 2005, Blum and Roli 2003, Glover and Kochenberger 2003), and among these the metaheuristic strategies (Glover and Kochenberger 2003) have shown especially well when addressing real-world FAP instances (Amaldi *et al.* 2006). Contrary to exact techniques, metaheuristics do not guarantee that solutions obtained are optimal, but they provide very good solutions in relatively short periods of time. But the present study goes further, because once all the metaheuristics were properly configured and fixed to obtain high quality results by themselves, the authors designed and developed a parallel cooperative hyper-heuristic (HH) which manages and distributes the work among all the strategies previously developed. This complex heuristic approach was designed with a double aim: first, to improve the results (in both time and quality) obtained by using each individual heuristic, and second, to beat the results obtained by any other approach (which solves the same real-world problem as the present one) published in the bibliography.

Therefore, on the one hand, there are seven very different heuristics which search within the search space of the FAP solutions; whereas on the other hand, the hyper-heuristic designed uses the algorithms previously mentioned as search space. This idea is broadly represented in Figure 1. In addition to this, it is important to remark that the HH designed makes the heuristics search in parallel, as will be explained in Section 5, thus a good performance is achieved in the global search of the optimal solution.

Figure 1. Hyper-heuristic and heuristic search spaces

As will be explained in the following Section (Antecedents and related work), the work presented in this article represents a novel and significant contribution, since it proposes a new parallel approach which has not hitherto been used to solve a very relevant real-world optimization problem. The main contributions of this article are the following three points:

- A new approach (parallel hyper-heuristic based on complex metaheuristics) has been proposed to solve a real-world large-scale optimization problem.
- The novel approach obtains high-quality frequency plans which are better than any other result which has been published so far in the bibliography.
- The hyper-heuristic is based on seven very representative metaheuristics (population and trajectory based, classical and novel) hybridized with a local search method. All the heuristics have been carefully adapted to the requirements of the realistic FAP tackled in the study. Also, some of the metaheuristics included have been never used for solving the frequency assignment problem, so the study and the development of these strategies to solve the FAP is also a contribution of this study.

Finally, the rest of the article is structured as follows: in Section 2, the work carried out on the FAP is examined, and specifically, on the heuristic techniques and the strategies related to the ones used in this study. In Section 3 the background and the mathematical formulation of the frequency assignment problem tackled in this work is presented. Sections 4 and 5 describe the solution approach. In Section 4 the metaheuristics included in the hyper-heuristic are detailed and then, in Section 5 the hyper-heuristic itself is carefully explained. The methodology followed is described in Section 6, while the different experiments

performed, the results obtained and the subsequent analysis are discussed in Section 7. Finally, the conclusions and future work are discussed in the last section.

2. Antecedents and related work

As was said in the Introduction, the frequency assignment problem (FAP web 2007) is a well-known optimization problem in the literature. It was introduced in the seventies by Metzger (1970) and due to its relevance, there are quite a lot of in-depth studies and complete surveys in which different techniques are proposed to solve it (Aardal *et al*. 2007, Borndörfer *et al*. 1998, Eisenblätter 2001, Koster 1999, Leese and Hurley 2002). However, among those works there are only a few studies in which exact algorithms were used in its resolution (Fischetti *et al.* 2000, Mannino and Sassano 2003) because it is proved that the best approaches to deal with this kind of problem are the heuristic methods (Aardal *et al*. 2007).

Furthermore, there is not one single version of the frequency assignment problem (Aardal *et al.* 2007). The FAP goal has changed from minimizing the number of frequencies used in the planning of a telecommunication network (MO-FAP: Minimum Order-FAP, and MS-FAP: Minimum Span-FAP), to try to minimize the interferences (MI-FAP: Minimum Interference FAP) or even the blocks (MB-FAP: Minimum Blocking FAP) produced in a realistic GSM network when a limited number of frequencies are used to support a huge number of communications. There are also some models which even include specific GSM issues, such as frequency hopping (Touhami *et al*. 2010, Björklund *et al.* 2005), or others more theoretical which take into consideration the different stages which occur in the frequency planning of a realistic network (Paunovic *et al.* 2001). Although there is much more to say about the evolution and variants of the FAP, no more details are going to be given here. The interested reader can consult references (Aardal *et al.* 2007 and FAP web 2007) to obtain more information about the different FAP variants and their evolution.

All in all, the FAP version which is usually tackled when working with real-world FAP instances is the MI-FAP. In this particular case, there are only a few frequencies (normally no more than 20) to give support to the hundreds or even the thousands of frequency assignments. Then, frequencies here have to be repeated many times, and this fact causes transmission interferences which obviously have to be minimized. This is the final goal of MI-FAP, which is the variant tackled in this study. Therefore, if the bibliographical search is focused on this version of the problem, it is easy to find out that the best results are usually obtained by metaheuristics methods (Blum and Roli 2003, Glover and Kochenberger 2003), which also are normally hybridized with a local search (LS) heuristic (a greedy method, Talbi 2002) to complement their stochastic behavior. Due to the number of works published, this section focuses on those studies that propose solution approaches similar to the methods used in this work. Seven different metaheuristics have been developed: GA, Genetic Algorithm (Holland 1992), which is the best known type of EA: Evolutionary Algorithm (Bäck *et al.* 1997), SS, Scatter Search (Glover *et al.* 2003, Martí *et al.* 2006), PBIL, Population Based Incremental Learning (Baluja 1994), VNS, Variable Neighborhood Search (Mladenovic and Hansen 1997), ILS, Iterated Local Search (Lourenço *et al.* 2002), GRASP, Greedy Randomized Adaptive Search Procedure (Feo and Resende 1995) and ABC, Artificial Bee Colony (Karaboga and Basturk 2007), which are also hybridized with a LS heuristic (Talbi 2002) to improve the search for high-quality solutions.

But in addition to the metaheuristics mentioned above, this article proposes a new parallel approach to solve our real-world problem. There are a lot of studies in which heuristics and metaheuristics work in parallel to improve their sequential performance when they are used to solve optimization problems (Alba *et al.* 2002, 2004, 2005, Crainic and Toulouse 2003, Cung *et al.* 2003, Talbi 2002). And of course, there are also a lot of works in which heuristics work in parallel to solve the frequency assignment problem (Aardal *et al.* 2007, Alba *et al.* 2005, Chaves-González *et al.* 2008a, 2009a, Eisenblätter 2001, Crompton *et al.* 1994, Funabiki and Takefuji 1992, Kendall and Mohamad 2004, Maple *et al.* 2004).

However, the heuristic technique presented here is very different, since this is the first time that, to the authors' best knowledge, a hyper-heuristic which uses metaheuristics has been proposed to solve a realistic FAP. Therefore, the following paragraphs will discuss on the one hand, the published works in which the seven heuristics used here have been used to solve the MI-FAP variant, and on the other hand, the studies in which hyper-heuristics have been adopted to solve realistic problems.

Starting with the hyper-heuristic backgrounds, it is remarkable that this is a quite recent idea –it was proposed in the late nineties by Hart, Ross and Nelson (1998). Hyper-heuristics can be thought of as "heuristics to choose heuristics" and they basically differ from metaheuristics in which most implementations of metaheuristics search within a search space of problem solutions, while hyper-heuristics always search within a search space of heuristics (Burke *et al.* 2003, Chakhlevitch and Cowling 2005, Ozcan *et al.* 2008, Ross 2005). As a result, when using hyper-heuristics, the goal is to find the right method in a given situation rather than trying to solve directly a problem (Fig. 1). This idea has been applied across many different real-world problems (mainly to solve scheduling and timetabling problems) in diverse research fields. Thus, there are recent works where hyper-heuristics are used for solving: educational timetabling problems (Burke *et al.*2002, 2003, 2004, 2006, 2007), bin packing problems (Ross *et al.* 2002, Cuesta *et al.* 2005), sales scheduling (Cowling *et al.* 2000), space allocation planning (Bai *et al.* 2003, 2008, Burke *et al.* 2005), nurse rostering (Aickelin *et al.* 2006, Burke *et al.* 2003), personnel scheduling (Cowling and Chakhlevitch 2003, Cowling *et al.* 2002, Remde *et al.* 2007) or industrial problems (Dowsland *et al.* 2007, Ayob and Kendall 2003). There are two works from Kendall and Mohamand (2004) which describe the use of hyper-heuristics to solve the FAP, but those studies are very different to the present one because they work with benchmark problems (not real-world problems as in this study) to solve the MS-FAP variant.

Moreover, hyper-heuristics can be developed using a great diversity of techniques (for example, they can be designed using Case-based Reasoning (Burke *et al.* 2002, 2006), Tabu Search or Local Search strategies (Aickelin *et al.* 2006, Burke *et al.* 2003), Graph-based approaches (Burke *et al.* 2007), Genetic Algorithms (Cowling *et al.* 2002), Ant Colony Optimization (Cuesta *et al*. 2005), Simulated Annealing (Dowsland *et al*. 2007, Bai and Kendall 2003), Variable Neighborhood Search techniques (Remde *et al.* 2007), Multi-Objective approaches (Burke *et al*. 2005) or Monte-Carlo strategies (Ayob and Kendall 2003). Therefore, hyper-heuristics are very recent and versatile approaches which can be developed using a high range of different techniques and can be applied over many different optimization problems. However, in spite of the great amount of works consulted, there are no studies in which hyper-heuristics are used to solve the present problem (MI-FAP) or where the hyper-heuristic is developed using the present approach. In all works analyzed, the hyper-heuristics are made up of simple heuristic searches or several variations of a single metaheuristic. Therefore, it can be concluded that the work performed here provides new contributions in both the problem domain (using a new approach to solve a real-world FAP) and in the development of the technique used to solve it (hyper-heuristic developed with several independent metaheuristics).

As specified before, after the hyper-heuristic backgrounds, this article will analyze the works in which the metaheuristics included in the present study (EA, SS, PBIL, VNS, ILS, GRASP and ABC) have been used to solve problem MI-FAP. There are a lot of studies published in which EAs have been used to deal with the FAP. Some of them are classical works (Crisan and Mühlenbein 1998, Crompton *et al*. 1994, Cuppini 1994, Dorne and Hao 1995) and others are very recent (Idoumghar and Schott 2006, 2009, San Jose-Revuelta 2007), however, for that reason (the number of contributions here is rather large), this Section is limited to those works in which the EA is hybridized with another optimization method (in the present case, a local search heuristic) to better tackle the problem (Talbi 2002). This is clearly justified because the metaheuristics are stochastic methods which search diversification and a local search heuristic usually improves their results by promoting search intensification of promising areas. Table 1 summarizes the main features of the most relevant works which deal with MI-FAP. For each contribution, the table shows: the bibliographic

reference, the optimization method used in combination with the EA (which is a kind of LS method in all cases), and the instances used with its size (when this information is available).

Table 1. Works which use hybrid EAs to solve the MI-FAP.

The number of works published in which hybrid EAs have been used to solve the MI-FAP shows the suitability of this kind of technique to deal with the problem. Also, the usage of greedy algorithms (or hill climbers), as LS methods, is the most widely used strategy in combination with the EA because of their ability of easily adding specific problem domain knowledge. Therefore, the called *Low-level Teamwork Hybrids* scheme (Talbi 2002), or LTH (EA (LS)), is especially effective for tackling our problem. However, in addition to hill climbers (or greedy) methods, a wide variety of advanced search heuristics have been used as optimization methods to improve the EAs performance (Table 1). Thus, Markov Decision Processes (MDP) were used in (Idoumghar and Schott 2006, Greff *et al*. 2004), Tabu Search (TS) in (Alabau *et al*. 2002) or (Mabed *et al*. 2002), and a Neural Network in (Salcedo-Sanz and Bousoño-Calzón 2005). With regard to the problem instances used, the instance sizes are similar. In general, there are several thousands of TRXs (transceivers) in the network.

With respect to the other metaheuristics used in this work, their use to solve the FAP compared with EAs is much reduced. Starting with ILS (Lourenço *et al*. 2002) and VNS (Mladenovic and Hansen 1997) techniques, these approaches have been widely used to solve the graph coloring problem (Chiarandini and Stützle 2002, Galinier and Hertz 2006), which is a particularization of the FAP, but to the authors' best knowledge, there are no works published related to these algorithms for solving the present problem. The same occurs with the ABC metaheuristic (Karaboga and Basturk 2007), because despite other approaches based on social insects (such as ACO: Ant Colony Optimization (Dorigo and Stützle 2004)) being used to solve the FAP (Luna *et al*. 2007a, Maniezzo and Carbonaro 1999), there are no works in which the ABC algorithm is used to solve the frequency assignment problem.

SS (Glover *et al*. 2003, Martí *et al*. 2006) was used firstly to solve the graph coloring problem in (Hamiez and Hao 2002), but the algorithm has been also fixed to solve the present problem in (Chaves-González *et al*. 2008b, 2008c, 2009b). In all those cases, the problem tackled is the same as the one used here. Also, SS was also hybridized with an efficient LS method to optimize its results (results will be compared in Section 7).

GRASP (Feo and Resende 1995) has been used in several works to solve the FAP (Chaves-González *et al*. 2009a, Gomes *et al*. 2001, Liu *et al*. 2000, Vieira *et al*. 2008). In all the works found the metaheuristics were hybridized with other heuristic to improve the results. In (Chaves-González *et al*. 2009a) the algorithm is used within a master-slave approach using grid computing to solve the same problem version as the one tackled here (results will be contrasted in Section 7), however, in the works (Gomes *et al*. 2001, Liu *et al*. 2000, Vieira *et al*. 2008), the problem is not tackled as in the present approach, because Liu *et al*. and Gomes *et al*. run their experiments using instances automatically generated, and Vieira *et al*. use the Philadelphia instances (FAP web 2007) in their study. Thus, all those cases consider benchmarking instances, and this work is focused on a real-world problem (and real-world instances).

With regard to PBIL (Baluja 1994), there are quite a lot of works by Chaves-González *et al.* (2009c, 2008a, 2008b, 2008d, 2008e) in which the algorithm has been studied thoroughly when it is used to solve the FAP. Some of those works use the same problem version and instances as are used in this study (Chaves-González *et al*. 2009c, 2008a, 2008b, 2008e), although the results obtained in all those studies are poorer than the results obtained here. In any case, the results and conclusions obtained there helped to improve the algorithm developed here. For example, in (Chaves-González *et al.* 2008e), results showed that the standard version of PBIL hybridized with a local search heuristic was the best approach to tackle the FAP, and this idea was taken to developed the best version of PBIL here. On the other hand, other studies work with benchmarking-like instances of the problem (Chaves-González *et al.* 2008d), or make comparisons between PBIL and other metaheuristics (Chaves-González *et al.* 2008b). Anyway, a complete comparative study will be performed

taking into account those works which use the same problem version and instances as ours in Section 7.

To finish this background section, some of the parallel approaches that have been published to solve the FAP are analyzed. The number of contributions here is also rather large. In fact there are very complete works in which several parallel approaches which can be applied to solve the FAP are thoroughly explained (Eisenblätter 2001, Alba *et al*. 2005), therefore this paragraph is focused on those studies which work with our specific version of the problem (the MI-FAP variant). It is important to point out that all the parallel approaches found combine a parallel strategy with a metaheuristic for solving the problem. Thus, in (Chaves-González *et al*. 2008a), FAP is solved using cluster computing and PBIL. In that work an island model was developed to improve the results obtained with the sequential version of the algorithm. In any case, the model proposed in this article beats the results obtained in those studies. Grid computing has been also proposed to solve FAP. In (Chaves-González *et al*. 2009a), the GRASP metaheuristic is parallelized using a master-slave model to improve both the performance and the results which were obtained with the sequential version of the algorithm. Finally, the solution proposed by Luna *et al*. (2008a) obtains very good results using a grid-based genetic algorithm when solving the same problem, but as will be shown in Section 7, the present approach also beats those results.

In conclusion, after analyzing a considerable number of works related to the study carried out, it can be affirmed that, to the authors' best knowledge, this article proposes a new parallel and evolutionary approach to solve a FAP realistic version which provides the best results published so far in the literature (as will be discussed in Section 7).

3. Optimization problem tackled: the frequency planning problem

The frequency planning is one of the most relevant optimization problems in the telecommunications domain. In fact, the FAP is considered a very important task for current, real-world GSM operators (Mouly and Paulet 1992) because only with an optimum frequency plan which makes the most of the scarce range of available frequencies is it possible to perform a communication of quality between the cell phones of a realistic network. Therefore, in the following subsections there will be, on the one hand, a very brief description of the GSM architecture and some details about the frequency planning applied to this kind of networks; and on the other hand, an explanation of the mathematical formulation of the FAP which is tackled here (Luna *et al*. in (2007a)).

3.1. The frequency assignment problem and the GSM communication system

As stated in the Introduction, the most used communication system used nowadays is the GSM system (Mouly and Paulet 1992). This is an open, digital cellular technology used for transmitting mobile voice and data services. The two most relevant components of the GSM system which refer to frequency planning are the antennas or, as they are more known, base transceiver stations (BTSs) and the transceivers (or TRXs). Essentially, a BTS is a set of TRXs (grouped in sectors). In GSM, one TRX is shared by up to eight users in TDMA (Time Division Multiple Access) mode. The main role of a TRX is to provide conversion between the digital traffic data on the network side and radio communication between the mobile terminal and the GSM network. The site where a BTS is installed is usually organized in sectors (of several TRXs) and the area where each sector operates defines a cell. No more details about the GSM system are going to be given here. To obtain a more detailed explanation about this issue, please consult reference (Mouly and Paulet 1992).

In any case, the frequency planning is the last stage in the design of a GSM network. FAP lies in the assignment of a channel (or a frequency) to every TRX in the network (Eisenblätter 2001). The optimization problem comes up because the usable radio spectrum is

usually very scarce and, consequently, frequencies have to be reused for many TRXs in the network. However, the multiple use of a same frequency usually causes interferences that can reduce the quality of service (*QoS*) down to unsatisfactory levels. In fact, significant interferences will occur if the same or adjacent channels are used in near overlapping cells. The problem is that computing this level of interference is a difficult task which depends on various factors (channels, radio signals, environment…). The more accurate the measure of the interference in a given GSM network, the higher the quality of the frequency plan that can be computed for this network. There are several ways of quantifying this interference. These methods range from theoretical methods to extensive measurements (Kuurne 2002), however the most extended way of quantifying the interferences produced in a GSM network is using the called *interference matrix*. Each element of this matrix contains a pair *(i, j)* which holds two types of interferences: the *co-channel interference*, which represents the degradation of the network quality if the cells *i* and *j* operate on the same frequency; and the *adjacent-channel interference*, which occurs when two TRXs operate on adjacent channels (*e.g.* one TRX operates on channel f and the other on channel $f + 1$ or $f - 1$). Therefore, an accurate interference matrix is an essential requirement for solving the frequency planning problem, because the final goal of any frequency assignment algorithm will be to minimize the sum of all the interferences held in this matrix. Furthermore, in real-world situations, FAP is involved with other additional complicating factors which may be considered. The interested reader is referred to (Eisenblätter 2001) to get more detailed explanations about frequency planning applied over GSM networks.

3.2. Mathematical formulation of the frequency assignment problem tackled

Let $T = \{t_1, t_2, \ldots, t_n\}$ be a set of n transceivers (TRXs) in a telecommunication network, and let $F_i = \{f_{i1}, \ldots, f_{ik}\} \subset N$ be the set of valid frequencies that can be assigned to a transceiver $t_i \in T$, $i = 1, \ldots, n$. Note that k, which is 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. Also, the sector in which a transceiver t_i is installed is denoted by $s(t_i) \in S$. Finally, let a matrix $M = \{(\mu_{ij}, \sigma_{ij})\}_{mxm}$, called the interference matrix where the two elements μ_{ij} and σ_{ij} of a matrix entry $M(i,j) = (\mu_{ij}, \sigma_{ij})$ are numerical values greater or equal than zero. In fact, μ_{ij} represents the mean and σ_{ij} the standard deviation of a Gaussian probability distribution describing the carrier-to-interference ratio (C/I) when sectors i and j operate on a same frequency (Walke 2002). The higher the mean value is, the lower the interference will be, and thus the better the communication quality. Note that the interference matrix is defined at sector (cell) level. 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) will be denoted by $p \in F_1 \times F_2 \times \ldots \times F_n$, where $p(t_i) \in F_i$ is the frequency assigned to the transceiver t_i. The objective, or the plan solution, will be to find a solution p that minimizes the following cost function (Luna *et al.* 2007a):

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

In order to define the function $C_{sig}(p, t, u)$ from equation 1, let s_t and s_u be the sectors in which the transceivers t and u are installed, which are $s_t = s(t)$ 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, $C_{sig}(p, t, u)$ is equal to the expression described in equation 2.

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

Equation 2 represents three types of cost associated to three interferences which can be found in a network frequency planning. $K >> 0$ is a very large constant defined by the network designer just to make it undesirable allocating the same or adjacent frequencies to transceivers serving the same area (*e.g.* installed in the same sector). $C_{co}(\mu, \sigma)$ represents the cost due to *co-channel interferences* (equation 3), whereas $C_{adj}(\mu, \sigma)$ is the cost in the case of *adjacent-channel interferences* (equation 6). Therefore, $C_{co}(\mu, \sigma)$ is defined as shown in the following expression:

$$C_{co}(\mu, \sigma) = 100(1.0 - Q((C_{SH} - \mu)/\sigma)) \quad (3)$$

where the term $Q(z)$ is the tail integral of a Gaussian probability distribution function with zero mean and unit variance (equation 4), while C_{SH} is a minimum quality signalling threshold.

$$Q(z) = \int_z^{\infty} 1/\sqrt{2\pi} \cdot \exp(-x^2/2)\, dx \quad (4)$$

Function Q is widely used in digital communication systems because it characterizes the error probability performance of digital signals (Simon and Alouini 2005). This means that $Q((C_{SH} - \mu)/\sigma)$ is the probability of the C/I ratio being greater than C_{SH} and, therefore, $C_{co}(\mu_{s_t s_u}, \sigma_{s_t s_u})$ 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 caused 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 other hand, 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. However, as function Q has no closed form for the integral, it has to be evaluated numerically. The complementary error function E is used for this purpose.

$$Q(z) = 1/2 \cdot E(z/\sqrt{2}) \quad (5)$$

Press *et al.* (2002) presents a numerical method that allows the value of E to be computed with a fractional error smaller than 1.2×10^{-7}. In a similar way, function $C_{adj}(\mu_{s_t s_u}, \sigma_{s_t s_u})$ is defined as expressed in equation 6.

$$\begin{aligned} C_{adj}(\mu, \sigma) &= 100(1.0 - Q((C_{SH} - C_{ACR} - \mu)/\sigma)) = \\ &= 100(1.0 - 1/2 \cdot E((C_{SH} - C_{ACR} - \mu)/(\sigma\sqrt{2}))) \end{aligned} \quad (6)$$

The only difference between functions C_{co} and C_{adj} is the additional constant $C_{ACR} > 0$ (adjacent channel rejection) in the definition of function C_{adj}. This hardware specific constant measures the ability of the receiver 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 $C_{adj}(\mu, \sigma) < C_{co}(\mu, \sigma)$. This makes sense, since using adjacent frequencies (channels) does not cause such a strong interference as using the same frequencies.

Finally, it should be noted that the mathematical model explained in this section (originally proposed by Luna *et al.* (2007a)) incorporates the definition of a precise interference matrix which is directly imported from real-world GSM frequency planning as currently conducted in the industry (and not generated in a computer by sampling random

variables). This definition allows not only the computation of high performance frequency plans, but also the prediction of *QoS*, because both the definition of the interference matrix and the subsequent computations carried out to obtain the cost values are based on the real-world GSM network operation.

4. Metaheuristics included in the hyper-heuristic

In this section the heuristics which are part of the authors' hyper-heuristic are described. Each single metaheuristic was carefully fixed for solving the proposed FAP, so the generic outline of each algorithm is not explained here (they are widely explained in the bibliography), but the adapted versions which were developed to solve the present problem are explained. Both trajectory (ILS, VNS and GRASP) and population (GA, PBIL, SS and ABC) based metaheuristics have been included, and all of them have been hybridized with the same local search (LS) algorithm, which was specially adapted to solve our problem. Therefore, in the following subsections brief explanations of each metaheuristic are included along with a very short description of the LS which is shared by all of them.

4.1. Local Search

The Local Search (LS) strategy used in this work was proposed by Luna *et al*. (2008b) with the aim to design a method which improves the stochastic behaviour of any metaheuristic approach developed to solve the present version of the frequency assignment problem. All the experiments performed in the study show that all metaheuristics improve their results when they use this local search heuristic, therefore it is a common element included in all the metaheuristics developed here. The aim of the LS designed is to optimize the assignment of frequencies to the TRXs in a given sector, but without changing the remaining network assignments. Candidate solutions (or frequency plans) are encoded for all heuristics developed as arrays of integer values p, where $p(t)$ is the frequency assigned to TRX t. The neighbours of a candidate solution are obtained by replacing the frequencies in the TRXs of each sector by other frequencies which are randomly chosen, but trying to avoid the generation of neighbours that are not going to improve the current solution. It is important to point out here that this local search will never assign the same frequency or adjacent frequencies to any pair of TRXs within the same sector (these co-channel and adjacent-channel interferences are the highest-cost kind of interference, as shown in equation 2, and they have to be avoided). To obtain detailed information about this local search heuristic, please consult reference (Luna *et al*. 2008b).

4.2. Iterated Local Search

Iterated Local Search (ILS, Lourenço *et al*. 2002) algorithm is a very fast trajectory-based metaheuristic used for solving optimization problems. It is based on the idea of applying a great number of times an efficient local search over a single individual to make it evolve. The individual (or solution to the FAP) is encoded as an array of integer values, p, where $p(t_i) \in F_i$ is a valid frequency assigned to the transceiver t_i. The algorithm is summarized in Algorithm 1.

Algorithm 1. Pseudo-code for ILS

```
1: generateInitialSolution (solution)
2: solution ← localSearch (solution)
3: while (not time-limit) do
4:     newSolution ← randomMutation (solution)
```

```
5:      newSolution ← localSearch (newSolution)
6:      solution ← updateSolution (newSolution, solution)
7:  endwhile
8:  return solution
```

As observed, the heuristic is based on the evolution of a unique solution. This solution is firstly randomly generated (line 1) and then improved using the LS heuristic described in Section 4.1 (line 2). After that, the solution is randomly mutated (line 4) to introduce perturbations which avoid local minima and then, the LS (line 5) is applied again to improve the frequency plan obtained after the mutation. The mutation operator used is the random mutation, in which the frequencies of a set of randomly chosen TRXs are reassigned with a random valid frequency. Finally, the solution is updated by comparing the interference cost (the fitness) of the new solution with the interference cost of the original solution (line 6). The solution with the highest cost (which is the worst individual) will be discarded and the algorithm will start a new iteration with the updated solution. This solution will be returned (line 8) when the algorithm reaches its stop condition (line 3).

4.3. Variable Neighborhood Search

The Variable Neighborhood Search (VNS, Mladenovic and Hansen 1997) algorithm is a trajectory-based heuristic used for solving optimization problems. VNS basically consists of systematic changes of neighborhood within a local search method which is being applied over a solution. The solution here is represented as in the previous algorithms explained (array of integer values which represent a valid frequency plan and where each position of the array represents one specific frequency assignment). There are several VNS variants based on the general outline of the algorithm (Mladenovic and Hansen 1997). Among these variants the sVNS (skewed Variable Neighborhood Search) version (Algorithm 2) was finally chosen, because it obtained the best results solving the FAP.

Algorithm 2. Pseudo-code for sVNS

```
1:  generateInitialSolution (solution)
2:  solution ← localSearch (solution)
3:  while (not time-limit) do
4:      k ← 1
5:      while (k <= k_max) do
6:          newSolution ← randomMutation (solution, k)
7:          newSolution ← localSearch (newSolution)
8:          solution ← updateSolution (newSolution, solution)
9:          k ← updateK (k, α, solution, newSolution)
10:     endwhile
11: endwhile
12: return solution
```

The sVNS approach tries to avoid local minima by considering solution searches far from the solution which is being explored at that moment (as trajectory-based heuristic, it only works with one solution). The heuristic starts just as ILS algorithm started: with the generation of a random solution (line 1) and its improvement using the local search described in Section 4.1 (line 2). However, the iteration in this occasion depends on a new parameter, k, which is used in the mutation method to control the mutation probability of the solution (more exactly, the mutation probability is equal to k * 10%). Thus, k parameter is initialized to 1 (the minimum mutation probability, or minimum neighbourhood) at the beginning of the iteration (line 4). This mutation probability (neighbourhood) will change in each step of the algorithm until the value of the k_{max} parameter is reached (9, in our case, as is specified in Section 6). The mutation operator used is the random mutation, but on this occasion, the k

parameter is used to check if a specific TRX has to be mutated or not (the algorithm generates a random value, and if it is lower than the mutation probability, the solution will be changed). In case it has, the new frequency to be assigned to that TRX is randomly chosen from a set of valid frequencies (line 6). These valid frequencies do not introduce new interferences to the sector in which the TRX is installed. Next, the same LS applied over the initial solution in line 2 is used to improve the fitness of the new solution generated (line 7). After that, the individual which will be taken for the next generation is updated (line 8) by choosing the solution with the lowest fitness. Finally, the *k* parameter is updated (line 9, Mladenovic and Hansen 1997) taking into account the distance between the new solution and the current solution (measured by counting the number of different frequencies used within the same sector) and the value of a new parameter, α. The value for this α parameter was established to 0.4 after a rigorous experimental study (Section 6). The new value of the *k* parameter indicates if the search is going to be performed far from or nearby the current solution in the next iteration of the algorithm. At the end of the process, the final frequency plan calculated is returned (line 12).

4.4. Greedy Randomized Adaptive Search Procedure

The Greedy Randomized Adaptive Search Procedure (GRASP, Feo and Resende 1995) is a multi-start trajectory-based metaheuristic commonly applied to combinatorial optimization problems. The algorithm consists of iterations which include successive constructions of a greedy randomized solution (there are several types of constructors with diverse parameters) and its subsequent iterative improvement through a local search (as previously, the LS described in Section 4.1 is used). The greedy randomized solutions are generated by adding elements (in our case, frequencies) to the problem solution set from a list of elements ranked by a greedy function according to the quality of the solution they will achieve. The variability in the candidate set of greedy solutions is obtained through a Restricted Candidate List (RCL), by randomly choosing the elements of the specific solution which is being built. Algorithm 3 shows the general outline of the metaheuristic.

Algorithm 3. Pseudo-code for GRASP

```
1: generateInitialSolution (solution)
2: solution ← localSearch (solution)
3: while (not time-limit) do
4:     newSolution ← greedyRandomizedConstruction (type, k, bias)
5:     newSolution ← localSearch (newSolution)
6:     solution ← updateSolution (newSolution, solution)
7: endwhile
8: return solution
```

As described in previous trajectory-based algorithms, GRASP starts with the generation of a random frequency plan (line 1) followed by the improvement of this first solution with the LS method described in Section 4.1 (line 2). After that, the main part of the algorithm starts with the generation of a greedy randomized solution (line 4), then the application of the LS method over the new solution generated (line 5) and finally the update of the best solution obtained so far by comparing the interference costs of the new generated solution and the previous version saved (line 6). There are several constructors for the generation of the greedy randomized solution (the *type* parameter indicates the one used in line 4). According to our experiments, the best results solving the FAP were obtained using the RG-variant (first random, later greedy; with $k = 1$ and *exponential bias*), so this is the configuration used to solve the present problem. Therefore, the RG-variant is the only version which is going to be briefly described in this section. To obtain detailed explanations about the GRASP constructor variants, please, consult reference (Feo and Resende 1995). The RG-variant modifies in the following terms the way in which the RCL is created: the restricted

candidate list is firstly filled in with *k* frequencies chosen randomly, but the rest of elements until complete the list are included greedily (choosing the best frequencies which incorporate less interference cost to the solution). After that, the element (or frequency) to be incorporated into the partial solution is randomly selected (considering also the exponential bias) from those in the RCL (this is the probabilistic aspect of the heuristic). Once the chosen element is incorporated to the partial solution, the candidate list is updated and the incremental costs are reevaluated (this is the adaptive aspect of the heuristic). This process is repeated until all the frequencies of the tentative frequency plan are assigned. Although the frequency plans obtained using this GRASP constructor are quite good, they are always improved by using the local search method described in Section 4.1 (line 5). This process of construction (line 4), improvement (line 5) and update (line 6), is performed until the stop condition of the algorithm is reached (line 3). Finally, the best solution found so far is returned at the end of the process (line 8).

4.5. Genetic Algorithm

The Genetic Algorithm (GA, Holland 1992) is the best known and used type of Evolutionary Algorithm (EA, Bäck *et al*. 1997). The tentative solutions managed by the present GA are preliminary frequency plans of the given FAP instance encoded as in the previous metaheuristics. A brief description of the algorithm is shown in Algorithm 4. The heuristic starts with the random generation of the population so that all the TRXs of each individual are randomly assigned with one of their valid frequencies. After that, the LS described in Section 4.1 is applied over the whole population.

Algorithm 4. Pseudo-code for GA

```
1:  initialize (population)
2:  population ← localSearch (population)
3:  while (not time-limit) do
4:      parents ← binaryTournament (population)
5:      offspring ← uniformCrossover (parents)
6:      offspring ← randomMutation (offspring)
7:      offspring ← localSearch (offspring)
8:      population ← updatePopulation (offspring)
9:  endwhile
10: return bestIndividual (population)
```

The GA includes binary tournament as the selection scheme (line 4) to choose the parents from which the offspring will be generated. These new frequency plans will replace the worst solutions within the current population if they reduce their interference cost. The binary tournament operator works by randomly choosing pairs of individuals from the population and choosing from each pair the individual having the best (lowest) fitness. The algorithm applies then uniform crossover to each pair of parents in which every gene of the offspring (*i.e.* the frequency of each TRX) is chosen randomly from one of the two parents (line 5). The mutation operator used is the random mutation in which the frequencies of a set of randomly chosen TRXs of the solution are reassigned with a random valid frequency. After the mutation, the same local search applied over the population in line 2 is used to improve the offspring fitness (line 7). Finally, the new offspring is used to update the population (line 8) by replacing the worst individuals by those newly generated which are better than the former ones (they have a lower interference cost). This process will be repeated generation after generation until the stop condition (a time limit) is reached (line 3). Then, the best individual in the population will be returned (line 10).

4.6. Population Based Incremental Learning

The Population Based Incremental Learning (PBIL, Baluja 1994) is a method that combines some properties of the genetic algorithms with the competitive learning for function optimization. The method is based on the evolution of a probability distribution, P, which is represented by a vector which holds the probability of each valid frequency to be assigned to each TRX of a tentative solution. The number of frequencies available for each TRX is not constant, so P is encoded as an array of arrays instead of as a regular matrix. Solutions are represented as in the previous algorithms, by arrays of integer values, p, where $p(t_i) \in F_i$ is the frequency assigned to transceiver t_i. The PBIL algorithm outline appears in Algorithm 5.

Algorithm 5. Pseudo-code for PBIL

```
1:  initialize (probVector)
2:  population ← generatePopulation (probVector)
3:  while (not time-limit) do
4:      population ← localSearch (population)
5:      bestIndiv ← bestIndividual (population)
6:      probVector ← updateprobVector (probVector, LR, bestIndividual)
7:      probVector ← mutateprobVector (probVector, MutP, MutA)
8:      population ← regeneratePopulation (probVector, population)
9:  endwhile
10: return bestIndividual (population)
```

The first thing to do is just initializing the probability vector, *probVector*, with the same probability for all frequencies (at the beginning, all frequencies have the same probability of being assigned to a specific TRX). Then, the first population is generated (line 2) by using the *probVector* previously initialized (line 1). After that, the local search method (Section 4.1) is applied over each individual in the population (line 4). The individual with the lowest fitness is then taken from the population (line 5) to update the probability vector. The probability vector is updated according to the information provided by the best individual of the population and the value of the learning rate (*LR*) parameter (line 6). This update is performed following the expression: $probVector_i \leftarrow probVector_i \times (1.0 - LR) + bestIndiv_i \times LR$. After that, the probability vector is mutated according to the *MutP* (mutation probability) and *MutA* (mutation amount) parameters following the expression: *if* (random (0, 1) < *MutP*) *then* $probVector_i \leftarrow probVector_i \times (1.0 - MutA) + \text{random } [0, 1] \times MutA$. Finally, a new population (which includes the best individual of the previous population) is regenerated by using the new probability vector (line 8). At the end of the process, when the time limit expires, the best solution is returned (line 10).

4.7. Scatter Search

Scatter Search (SS, Glover *et al*. 2003, Martí *et al*. 2006) is a metaheuristic which works with a representative set of solutions which is called *RefSet* (from the words *reference set*). Solutions here are encoded by using the same representation as in the previous algorithms, that is, arrays of integer values. The *RefSet* is composed of the most representative solutions from the population. It is divided into quality solutions (the best frequency plans which solve our problem) and diverse solutions (the most different ones). The number of individuals for each subset has been specially configured to solve the present version of FAP.

Algorithm 6. Pseudo-code for SS

```
1:  initialize (population)
2:  population ← localSearch (population)
3:  RefSet ← generateFrom (population)
```

```
4:  while (not time-limit) do
5:      SubSet ← subSetGenerator (RefSet)
6:      SubSet ← combinationMethod (SubSet)
7:      RefSet ← localSearch (SubSet)
8:      RefSet ← generateFrom (RefSet, population)
9:  endwhile
10: return bestIndividual (RefSet)
```

An outline of the algorithm is shown in Algorithm 6. The algorithm starts with the random generation of the population through the assignment of a valid frequency to each single TRX in each solution (line 1). Then, the LS described in Section 4.1 is applied over the population to improve the quality of solutions (line 2). After generating the *RefSet* (line 3), a Subset Generation Method (line 5) is used to create all possible subsets from the *RefSet*. The next step consists of applying the Solution Combination Method (line 6) to the solutions in each subset. The solutions are combined in a pair-wise way. Finally, the LS is used again (line 7) to try improving the frequency planning obtained as the result of the combination method. Frequency plans are replaced in the *RefSet* so that the best solutions keep in there. When all combinations have been evaluated, the best solution is saved for the next *RefSet* and a new random population is generated to select the (*RefSetSize* – 1) most diverse solutions from there (line 8). The distance used to measure the diversity between two frequency plans is the total amount of different frequencies assigned to every sector in both plans. With this new reference set, the algorithm restarts a new iteration until the time limit for the experiment expires (line 4). When this occurs, the best individual held in the reference set will be returned as the final solution to the problem (line 10).

4.8. Artificial Bee Colony

Artificial Bee Colony (ABC, Karaboga and Basturk 2007) is one of the most recently defined population-based algorithms used to solve optimization problems. It was created in 2005 by Karaboga (2005) and the heuristic is motivated by the intelligent behavior of honey bees. The bee colony consists of three groups of bees: employed, onlookers and scouts bees. In ABC model, the position of a food source represents a possible solution to the optimization problem (in our case, a valid frequency plan) and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. As explained in previous algorithms, solutions are represented by arrays of integer values. The general outline of the algorithm adaptation is shown in Algorithm 7.

Algorithm 7. Pseudo-code for ABC

```
1:  initialize (population)
2:  population ← localSearch (population)
3:  while (not time-limit) do
4:      population ← mutationMethod (employedBees, population)
5:      probVector ← generateSolutionProbability (population)
6:      population ← generateSolutions (onlookerBees, probVector, population)
7:      population ← replacePoorerSolutions (scoutBees, population)
8:  endwhile
9:  return bestIndividual (population)
```

The algorithm starts with the random generation of the population, or bee colony (line 1), and the improvement of the solutions (or frequency plans) within the population using the local search method described in Section 4.1 (line 2). Then, each generation of the algorithm will be divided into the following 4 stages: firstly, the employed bees, which represent up to half of the colony size, perform a random mutation in which the frequencies of a set of randomly chosen TRXs of the solution are reassigned with a random valid frequency. After

doing this operation, the same LS applied in line 2 will be used to improve the mutated solutions. Secondly, a probability vector is generated according to the fitness of each solution. This vector contains the probability that each frequency plan in the population is explored by the onlooker bees. The best solutions will have more probability than the poorer of being chosen by this kind of bees. After that, the onlooker bees generate new individuals by taking solutions from the population according to the probability vector previously generated. A mutation method similar to the one applied in line 4 will be used to create new frequency plans (but in this case the solutions are not randomly chosen, but taken according to the probability vector created in line 5). Finally, the scout bees replace the worst solutions in the population by others which are randomly generated (line 7). This operation expands the search space of the algorithm, avoiding that the heuristic gets stuck in a local optima. At the end of the process (line 9) the best solution held in the population will be returned (line 9).

5. Parallel hyper-heuristic

This article proposes a new approach to solve optimization problems based on heuristics searches: a parallel hyper-heuristic (PHH). The present PHH has been developed using several metaheuristics (described in the previous section), which are very significant trajectory (ILS, VNS, GRASP) and population (GA, SS, PBIL, ABC) based heuristics. As stated in the Introduction section, hyper-heuristics can be thought of as "heuristics to choose heuristics" and they basically differ from metaheuristics in which these search within a search space of problem solutions, while hyper-heuristics always search within a search space of heuristics. A diagram with the general outline of the proposed approach is shown in Figure 2. The present system works as follows: the problem specification and the instances of the problem are taken by the metaheuristics through an adapted evaluation function to evaluate the quality of the generated solutions. The problem itself is solved by these metaheuristics. The task carried out by the hyper-heuristic (HH) consists of controlling the metaheuristics output to distribute the workload according to the results obtained by each heuristic along the whole execution time of the system. The communication between the HH and the "lower-level" metaheuristics does not depend on the problem specifications, because the HH only receives, in each synchronization, information about the costs and the solutions (this is only numerical data) which are obtained by the metaheuristics until that moment. According to that information, the hyper-heuristic chooses the number of instances of each metaheuristic which will work with the solutions until the next synchronization of the system. Finally, at the end of the process, the best solution obtained so far (by one of the metaheuristics) will be returned as the final solution.

Figure 2. General outline of the proposed approach.

The hyper-heuristic achieves a good performance in the global search for the optimum solution, because it makes the metaheuristics search in parallel. An island model with all the metaheuristics has been developed. This model is managed by a master process (our hyper-heuristic) which rules all the metaheuristics. Also, to optimize the communications between the processes, an efficient communication protocol which minimizes the communications has been designed. Therefore, the PHH not only chooses which heuristic is the most appropriate in a given situation, but also distributes efficiently the workload among the different metaheuristics depending on the quality of their results. As will be explained in the next section, a cluster computer with 128 cores has been used to perform all the experiments. Then, one of the cores of our cluster will be used to run the hyper-heuristic process, which is going to work as the master process of the system, and the rest of the cores will be taken by the heuristics to solve the problem itself. The outline of the PHH master process is described in Algorithm 8.

Algorithm 8. Pseudo-code for the master of the PHH

```
1:  initialize the system probability vector, probVectorHH
2:  assign a metaheuristic to each core according to probVectorHH
3:  while (stop condition not satisfied) do
4:      wait until the sync. (while each core runs the assigned metaheuristic)
5:      when sync. → each metaheuristic sends the best solution to the master
6:      receive the solutions sent by the heuristics and sort them by their fitness
7:      update probVectorHH according to the best solutions received
8:      assign a metaheuristic to each core according to the new probVectorHH
9:      send the best solution to each metaheuristic for the restart
10: endwhile
11: return the best solution received at the end of the process
```

The parallel hyper-heuristic (PHH) is a synchronous system which manages and synchronizes the metaheuristics which solve the FAP. The seven metaheuristics are distributed between the available cores at the beginning of the process (line 2). PHH performs this important task by using a probability vector which indicates the approximate occupation of the metaheuristics in our cluster. This probability vector is initialized uniformly before the distribution of the metaheuristics (line 1). Therefore, the heuristics are expected to be distributed homogeneously in our system at the beginning of the process. Moreover, a minimum number of cores per metaheuristic is established (in the present specific case, 5 cores per metaheuristic were configured), just to avoid that in future iterations any algorithm could disappear from the system. Then, once all metaheuristic instances have been assigned to a core of our cluster, the heuristics will start their execution (as described in Section 4) until their stop condition (a time limit) is satisfied. When they finish, each metaheuristic will send the best solution found so far to the master process (line 5). The hyper-heuristic only waits idly until the solutions sent by the metaheuristics start to arrive (line 4). When the reception process is finished, the master process sorts all the solutions according to their quality (line 6). It is important to remark here that each solution received in the hyper-heuristic includes the frequency plan which represents that solution (this is an array of integers, as in the metaheuristics described in the previous section), its fitness value (which shows the quality of the solution) and a code which identifies the heuristic which obtained that solution. Furthermore, the PHH does not accumulate any solution from one synchronization point to the next because each individual heuristic includes elitism strategies. Therefore the system always works with the best frequency plans generated.

After the reception process has finished, the probability vector is updated with the best solutions following a process of selective pressure (line 7). This process only considers a certain number of the best solutions obtained, therefore, for each metaheuristic occurrence in the "top list", the probability (taken from 0 to 1) of that metaheuristic is increased proportionally. Then, as the system is using 127 cores to run the metaheuristics, when there is synchronization point, the top list in that very moment includes, for example, the best 50 solutions provided by the metaheuristics of the system. Thus, if these 50 solutions are distributed as follows: 17 by SS, 9 by GA, 7 by ABC, 6 by VNS, 5 by PBIL, 4 by ILS and 2 by GRASP; then, the updated probability vector will include the following values: P(SS) = 0.34, P(GA) = 0.18, P(ABC) = 0.14, P(VNS) = 0.12, P(PBIL) = 0.1, P(ILS) = 0.08, and P(GRASP) = 0.04. The number of solutions taken into consideration in the synchronization points depends on the number of cores used to run the heuristics. The master process of the PHH only keeps the best solution provided by each core, and the selective pressure grows proportional to the number of cores used to run the metaheuristics. For example, with 127 cores the 50 best solutions out of 127 (selective pressure) are only considered in order to update the probability vector. After this update, the hyper-heuristic reassigns the metaheuristics to each free core (line 8). The new distribution is performed according to the new probability vector but considering besides that each metaheuristic will be assigned to at least 5 cores of the system. Moreover, the hyper-heuristic includes elitism strategies, because the best solution obtained after every synchronization is also sent to each metaheuristic. This

solution will become the initial individual for trajectory-based metaheuristics (ILS, VNS and GRASP); meanwhile the population-based heuristics (GA, SS, ABC and PBIL) will include this solution to the first population generated. Finally, when the stop condition is satisfied (in the present case, a certain number of synchronizations, as will be explained in Section 6), the hyper-heuristic returns the best solution obtained at the end of the process, or in other words, the best frequency plan which solves the optimization problem.

6. Methodology

All the experiments performed have been carried out in a cluster computer which includes 16 identical nodes, each one with 2 processors Intel Quad Xeon CPU 2.33Ghz, 6MB L2 Cache and 8 GB DDR2. Also, all nodes are connected through a full-duplex Gigabyte Switch (1000Mbps). Therefore, the system includes 128 cores (16 nodes × 2 processors per node × 4 cores per processor) fully connected. With regard to the software environment, all nodes include Scientific Linux 5.3 64 bits OS (SL home page 2010), MPICH2 1.0.8p library to use MPI (Message Passing Interface, MPI forum home page 2010), and the GCC 4.1.2 compiler (GCC home page 2010). In the following subsections some more details are given about the experimental conditions of all experiments.

6.1. GSM problem instances

The real-world GSM networks used in the present study correspond to two quite large US cities: Seattle and Denver (both with more than 550,000 inhabitants). Figure 3 shows the topology of both instances. In that figure, each triangle represents a sectorized antenna, where several TRXs (between 2 and 6) operate to give support to the communications produced in a certain area of the network. The number of TRX per sector varies depending on the traffic demand and the specific requirements produced in each zone of the city, but for both instances, 3 or 4 TRX installed per sector is the usual case. As stated in the Introduction, frequency planning is an important and very complex task for current real-world GSM operators, because the number of available frequencies to cover the entire network communications is always very small. In the present case, Denver GSM instance includes 2612 TRXs (or transceivers) distributed in 334 BTSs (base transceiver stations or antennas), and it only has 18 available frequencies to be assigned to each TRX. On the other hand, the Seattle instance includes 970 TRXs installed in 503 BTSs and only 15 different frequencies are allowed to perform the network planning. It is important to highlight here that the first stages of the present research were started using benchmark instances of the FAP library to configure and test the algorithms. For example, (Chaves-González *et al.* 2008d) studied the resolution of several FAP benchmark instances with the PBIL algorithm, however, even this heuristic, which supplies the worst results in the current study (Figure 6), provided results which were optimal or near optimal when tackling benchmark instances. For that reason, real-world instances were used in the present experiments. These instances are much more complex and include realistic demands and requirements taken from the industry, being more interesting and challenging for this research.

Figure 3. Topologies of the GSM instances used in the study

The mathematical formulation used to represent the problem was explained in Section 3. Some constants were defined there, in equations 2, 3 and 6, and their values were set as follows: K = 100,000, C_{SH} = 6 dB and C_{ACR} = 18 dB, respectively (for both instances). Finally, it is also remarkable that the data source considered to build the interference matrix based on the C/I probability distribution (see Section 3) uses thousands of Mobile Measurement Reports (MMRs, Kuurne 2002) rather than propagation prediction models. MMRs are a more

accurate data source, because they capture the cell location pattern in the network and do not rely on predictions, as do the benchmark instances (FAP web 2007).

6.2. Experimental conditions and parameter configuration of the algorithms

Due to the stochastic nature of metaheuristics, each experiment performed in the study includes *30 independent executions*. In order to provide the results with statistical confidence, make comparisons and detect differences between the algorithms within short and long time ranges, three different time limits (120, 600 and 1800 seconds) have been considered. These limits have been chosen for a double reason: on the one hand, to bear in mind the practical interest of telecommunication companies, for which the time requirements are usually very hard; and on the other hand, because there are several studies published where the present problem has been tackled using other approaches (as will be analyzed in the following section) and it was desired to obtain comparable results.

All the single metaheuristics included in the present system (Section 4) have been carefully fixed with complete sets of experiments. Therefore, before doing the experiments with the final approach, an in-depth rigorous study was performed (each single experiment was also repeated 30 times independently) in order to find the best parameter configuration for each heuristic included in the present system. These previous experiments are also very relevant because it is well-know that the success of a hyper-heuristic is based on the good performance of its low-level heuristics. The authors point out here that the goal of this preliminary experimentation was to find the parameter settings that meet the several requirements imposed by the industrial context of the work. First of all, the authors wanted to provide the two real-world FAP instances with very accurate solutions, but using the same parametrization in the two cases. Indeed, it is known that the algorithms can be fine tuned for each instance but, when presenting the optimization tool to a telecommunications engineer who is not an expert in meta-heuristics, he/she can expect a good solution as soon as possible without tuning the solver. Therefore, once each individual heuristic was configured, the parameter setting is not modified at all in the final system. The following items summarize the parameter setting obtained for each of the individual metaheuristics (see Section 4 for the meaning of each parameter) when they are configured to solve the present optimization problem.

- **ILS**: mutation probability (*MutProb*) = 0.2.
- **VNS**: $k_{max} = 9$, $\alpha = 0.4$.
- **GRASP**: type = RG-variant, $k = 1$, *bias* = exponential bias.
- **GA**: population size = 30, offspring size = 6, mutation probability (*MutProb*) = 0.02, parents selection = binary tournament.
- **PBIL**: population size = 20, learning rate (*LR*) = 0.1, mutation probability (*MutP*) = 0.02, mutation amount (*MutA*) = 0.1.
- **SS**: population size = 40, *RefSet* size = 9 (best solutions number = 1, diverse solutions number = 8), solution combination method = uniform crossover, subset generation method = pairs of individuals.
- **ABC**: colony size = 30, employed bees size = 6, onlooker bees size = 24, scout bees size = 1, scout bee is used each iteration of the algorithm, *MutProb* = 0.2.

7. Empirical results

In this section the results obtained in the different experiments with the parallel hyper-heuristic approach are analyzed. The quality of solutions has been also compared with other studies published in the literature. Before doing this, it is necessary to put properly in context how the quality of those results (or frequency plans) is measured. The quality of a solution provided by a metaheuristic depends on the value of its associated fitness. This fitness value

represents in the present case the interference cost associated with a specific frequency plan, which in real-world networks is very high and is the parameter to be minimized by any algorithm which works with this complex optimization problem. Furthermore, the two instances used here have important differences in their requirements and features (different interference matrices, different number of TRXs, number of BTSs, available frequencies, etc), because they cover significantly different cities which have different sizes and topologies, as seen in Figure 3. In this way, with these two instances an important range of GSM network features is covered. Therefore, according to the present study, if another instance (or several other instances) is given to the PPH system, it should be expected that the PHH should work well using the current setting, and no further calibration should be required to solve it.

7.1. Synchronization number study

As described in Section 6, all experiments have been performed in a 128-core cluster. The master process of the PHH distributes the metaheuristics included in the proposed approach and, after a certain period of time, collects the best result generated by each heuristic. Then, the hyper-heuristic will use all the solutions provided to synchronize the system according to the best solutions obtained until that moment (Algorithm 8). This synchronization process is very important, since it determines how many cores will run each heuristic and it also updates the initial solution which is used by the metaheuristics after each synchronization. Therefore, the good evolution of the metaheuristics included in the system will significantly depend on the good configuration in the number of synchronizations which the PHH will perform in each independent execution. Moreover, despite the communications between the master process and the metaheuristics has been optimized by minimizing the number and size of messages passed, it is important that the system is configured with an appropriate number of synchronizations. If the number of synchronizations is too high, the system will spend too much time performing this task, and the heuristics will not have enough time of execution to provide good solutions. On the contrary, if the number of synchronizations is too low, the heuristics will work with an inadequate configuration, and the results will not be as good as they should. For all those reasons, a complete set of experiments was performed using both instances of the problem –Denver and Seattle.

Figure 4. Average result evolution for the different synchronization experiments performed using the Denver and the Seattle FAP instances

Figure 4 summarizes the results obtained with these experiments. As can be observed, the heuristics are synchronized every minute, every 2 minutes, 5 minutes, 10 minutes and 15 minutes. Since the time limit for every experiment is 30 minutes (see Section 6.2), this means that the experiments are performed making 29, 14, 5, 2 and 1 synchronizations respectively. If both graphics (Figure 4) are carefully examined, it can be seen that the best results are obtained when the PHH synchronizes the metaheuristics every 5 minutes. This is clearly observed in Figure 4b, where Seattle instance results are plotted. The row labeled as “Seattle” in Table 2 expresses the best and average results of 30 independent executions for three different time limits (2, 10 and 30 minutes). If the system is configured with a higher number of synchronizations, the results are also very good in the first minutes of the algorithm execution (as shown in Figure 4b), but they become poorer from that moment, because too much time is wasted in operations of synchronization, and the heuristics do not have enough time to make their solutions evolve properly. Therefore, synchronizing every 5 minutes is the best option to obtain optimal frequency plans when the PHH is used to solve the Seattle instance.

With regard to the Denver instance (Figure 4.a), it can be seen that very good results are obtained by using different configurations, but results here depend on the time that the algorithm is running. Thus, for short periods of time (up to 10 minutes) the best frequency

plans are obtained when the system synchronizes the heuristics very frequently (every minute, 2 minutes or 5 minutes). On the contrary, when the time limit gets bigger, very good results are obtained by synchronizing every 5, 10 and 15 minutes. In fact, if only the global results (obtained after the 30 minutes of execution) are considered, the best frequency plans are obtained by synchronizing every 10 minutes (with an average frequency cost of 84484.53 cost units). However, after analyzing the overall results, it can be concluded that the best frequency plans are obtained when the system is synchronized every 5 minutes, because very good results are obtained with this configuration in any period of time (the reduction in the interference cost is optimum if the whole execution time is considered, Figure 4a).

Table 2. Empirical results of the PHH approach (with sync time = 5 min.) for 3 different time limits on both, the Seattle and the Denver instances. The best, mean ($\bar{x}$) and the standard deviation (σ) values of 30 independent executions are presented

All in all, after the result analysis, the best frequency plans are obtained for both instances, Denver and Seattle, when the parallel hyper-heuristic synchronizes the metaheuristics every 5 minutes. More synchronizations make the results poorer, because the metaheuristics do not have enough time to make the solutions improve (too much time is dedicated to the synchronization task), and with less synchronizations the metaheuristics work as separated islands for too long, obtaining worse results because of the lack of control from the hyper-heuristic. The final empirical results obtained with the best configuration found for both instances are shown in Table 2.

7.2. Contribution of each algorithm study

In this subsection the contribution performed for each single metaheuristic to the overall evolution of the system (Figure 5) is discussed. This is important because it proves that all the metaheuristics chosen contribute in the search of the optimal solution of the problem. Only the winner configuration analysis (synchronizations every 5 minutes) has been included. Moreover, the final results obtained by the PHH system and the results provided by each single metaheuristic included in the system (Figure 6) have been compared, and as can be seen, the PHH clearly improves the results of any of the metaheuristics. Moreover, studying those results it is possible to think that some metaheuristics are redundant in the system, but this is false because, as observed in Figure 5, all heuristics contribute to the global system evolution.

Figure 5. Average metaheuristic contributions obtained when the best sync configuration is established to solve both problem instances

Figure 6. Comparison between the results obtained by the different metaheuristics separately and the final results obtained by the PHH system

However, the best frequency plans are usually provided by the population-based heuristics. In case of the Denver instance, the ABC algorithm is the method which makes the highest contribution to the system (in fact, ABC, SS and GA metaheuristics provides approximately the 60% of the best individuals). This is explained since the population based algorithms are the most stochastic ones, and each synchronization supplies these algorithms with a relevant greedy contribution which improves their operation. On the other hand, trajectory-based algorithms are greedier by themselves, so synchronizations have less impact in their evolution. Something similar occurs in case of the Seattle instance, in which the GA algorithm provides more than the 40% of best solutions (Figure 5), while the rest of heuristics share their contribution. In any case, the heuristics which are weak in some situations are very strong in others. Thus, although the trajectory-based heuristics obtain poorer contribution

results than the population-based, they work better than population-based algorithms during the first minutes of execution because they evolve faster, so their contribution is also very relevant to obtain high-quality frequency plans at the end of the process. In conclusion, the hyper-heuristic system uses a well-balanced set of heuristics, and all of them contribute to the search for FAP solutions with greatly reduced interference costs.

7.3. Number of cores study

This subsection studies the variation in the results when different numbers of cores are used. Experiments from 8 cores (the PHH includes 7 metaheuristic in addition to a master process) up to all the 128 cores available in the cluster have been performed. In this respect, it is important to clarify that when the system works with the minimum number of cores (8), each heuristic is assigned constantly to a different core. Therefore, in this particular case the hyper-heuristic only performs a constant delivery of the 7 heuristics designed. Figure 7 summarizes the results produced in these experiments.

Figure 7. Results obtained when the proposed approach is supported using different numbers of cores

All the experiments have been performed using the best synchronization configuration and with the same experimental conditions explained in the previous sections. The results obtained using our two real-world instances are shown: Denver (Figure 7a) and Seattle (Figure 7b). As can be observed, the system obtains very good frequency plans even with a small number of cores. Thus, even with only 8 cores the results are quite good, although the result quality is clearly improved when the number of cores increases (especially if we focus on the Denver instance, in which the improvement is remarkable if the number of cores is high). Therefore, on the one hand it can be concluded that outhe system obtains very good results with a wide range of cores, and on the other hand, a gradual increase in the number of cores is translated into a steady improvement in the quality of the results, which is a remarkable indicator of the good design of the parallel approach.

7.4. Comparative study with other authors

Finally, in this subsection the results obtained in the present study are compared to the results obtained in other works published in the bibliography. Table 3 summarizes this comparison (in three slices of time). Our results are shown in the first row, while the following rows contain the results of other research studies. All the studies included in Table 3 solve the same problem instances with different heuristic methods. Related to this point, it should be stated that there are no comparisons for the Seattle instance since, to the authors' best knowledge, this work represents the first study in which this instance has been tackled. For each study referred to in the table the best and average results (with the standard deviation) are given of 30 independent executions in three different time limits (120, 600 and 1800 seconds). Only the last work in Table 3, (Grid EA by Luna *et al*. 2008a), uses a different time limit, because the algorithm developed in that study was executed in a grid environment taking into consideration an iteration limit instead of a time limit. Despite this fact, that study is relevant because it showed the best results published so far in the resolution of the Denver instance. As is shown, even those results, which were obtained after 1.5 hours of execution are improved upon by the present approach, but in only 30 minutes of execution (one third of the time). The rest of the approaches are compared by using the same time limits and conditions as in this study.

Table 3. Best, average and standard deviation results (for the Denver instance) obtained after 30 independent executions using different approaches.

As may be seen in Table 3, very different approaches have been used to solve the present FAP version. Some of them are heuristics which were included in the PHH system (Chaves-González *et al*. 2009a, Chaves-González *et al*. 2009b), and others are also parallel approaches (Chaves-González *et al*. 2009a, Luna *et al*. 2008a), however the present results improve all of them, in every time limit. Only the Grid GRASP approach (Chaves-González *et al*. 2009a) obtains a better solution than the PHH in 120 seconds, but its average result is worse than ours (87256.9 cost units versus 87100.2), and the standard deviation is quite high (381.2 versus 2309.2), so the best individual obtained in that time slice can only be considered as a peak result. In fact, the PHH system obtains better results than the Grid GRASP approach in the 600 and 1800 time limits.

8. Conclusions

This article describes a new approach to tackle complex optimization problems. The present study was based on the resolution of a realistic frequency assignment problem. Both, the instances used to perform the experiments and the mathematical model adopted to deal with the problem were provided by the industry.

The development of the proposed approach was divided into two main stages. The first stage studied, designed and adapted different and representative metaheuristics (Section 4) to solve the optimization problem. Some relevant population-based heuristics (GA, SS, ABC and PBIL) and some very efficient trajectory-based heuristics (ILS, VNS and GRASP) were included. Those algorithms were also hybridized with a local search method (Section 4.1) to improve their results. Then, after the rigorous parameter adjustment of the algorithms to solve the present problem, a parallel cooperative hyper-heuristic was designed to control all the lower level metaheuristics. It is important to remark here that, to the authors' best knowledge, this is the first time that this approach (parallel metaheuristics within a hyper-heuristic) has been used to solve a real-world problem (FAP). In fact, some of the metaheuristics included in the PHH have not been ever used to solve this problem, so the study and the development of those novel techniques to solve the real-world FAP can be considered also an important contribution of the present work. Furthermore, in order to obtain reliable data from the tests, a complete set of experiments was performed, taking information in different periods of time. The system has been evaluated every 2, 10 and 30 minutes to compare its results with other studies published in the literature. 30 independent executions have been also run for each experiment in order to obtain statistical results. Thus, after analyzing these data, summarized in Tables 2 and 3, it can be stated that the results here are quite reliable, because for all the experiments performed, very competitive frequency plans with very low standard deviations were obtained.

Moreover, if the results are more carefully analyzed, some interesting conclusions can be reached. On the one hand, all heuristics contribute to the global system evolution (Figure 5). Also, the proposed approach obtains very good results when it is run with different number of cores (Figure 7), improving the quality of the frequency plans when the number of cores is increased. Furthermore, the PHH approach obtains the best results when the metaheuristics are synchronized every 5 minutes (Figure 4). On the other hand, the latest related works have been examined to compare the results obtained in the present study with other recent and relevant studies (Table 3).

Therefore, the most important contributions of the study can be summarized as follows: first, a new heuristic approach has been developed to solve a complex optimization problem: a realistic FAP with constraints and requirements taken from the industry; second, the results obtained with this approach represent very high quality frequency plans with much reduced interference costs; and finally, after a complete comparative study it can be stated

that the results obtained with this approach beat all the results published so far in the literature.

As future lines of work, the authors are interested to use the approach to solve even larger FAP instances and to solve other important optimization problems. For example, other relevant problems from the telecommunication domain which will be considered are the ACP (Automatic Cell Planning) or the RND (Radio Network Design) problems.

Acknowledgements

The work presented here was partially funded by the Spanish Ministry of Science and Innovation and the ERDF (European Regional Development Fund) under the contract TIN2008-06491-C04-04 (the M* project).

Table 1. Works which use hybrid EAs to solve the MI-FAP

Reference	Optimization method	Problem instance (size)
(Alabau *et al.* 2002)	Greedy, Tabu Search	Owner (5700 TRXs)
(Mabed *et al.* 2002)	Tabu Search	Owner (639 TRXs)
(Matsui *et al.* 2003)	Greedy	Owner
(Greff *et al.* 2004) (Idoumghar and Schott 2006)	Markov Decision Proc.	Owner (5700 TRXs)
(Salcedo and Bousoño 2005)	Neural Network	Philadelphia
(Colombo 2006)	Greedy	Generated (1667 TRXs)
(Luna, Alba *et al.* 2007a)	Greedy	Owner (2612 TRXs)
(Luna, Blum *et al.* 2007b)	Greedy	Owner (2612 TRXs)
(Luna, Estébanez *et al.* 2008b)	Greedy	Owner (2612 TRXs)

Table 2. Empirical results of the PHH approach (with sync time = 5 min.) for 3 different time limits on both, the Seattle and the Denver instances. The best, mean ($\bar{x}$) and the standard deviation (σ) values of 30 independent executions are presented

	120 seconds		600 seconds		1800 seconds	
	Best	$\bar{x} \pm \sigma$	Best	$\bar{x} \pm \sigma$	Best	$\bar{x} \pm \sigma$
Seattle	925.44	1075.01±59.86	546.76	635.61±43.01	456.64	550.07±46.50
Denver	86340.26	87100.20±381.16	84246.64	85167.94±382.86	83815.78	84527.43±404.72

Table 3. Best, average and standard deviation results (for the Denver instance) obtained after 30 independent executions using different approaches.

	120 seconds		600 seconds		1800 seconds	
	Best	$\overline{x} \pm \sigma$	Best	$\overline{x} \pm \sigma$	Best	$\overline{x} \pm \sigma$
PHH	**86640.3**	**87100.2±381.2**	**84246.6**	**85167.9±382.9**	**83815.8**	**84527.4±404.7**
ACO (Luna, Alba *et al.* 2007a)	91140.0	93978.2±1165.9	89703.4	91726.4±1002.9	88345.9	90382.5±935.3
EA (Luna, Alba *et al.* 2007a)	104247.7	108071.9±1723.4	100701.2	103535.9±1939.7	96490.5	99862.3±1553.1
GRASP (Chaves, Hernando *et al.* 2009a)	88857.4	91225.7±1197.2	87368.4	89369.6±1185.1	86908.4	88850.6±1075.2
Grid GRASP (Chaves, Hernando *et al.* 2009a)	85313.0	87256.9±2309.2	85313.0	86772.1±1701.0	85259.4	85855.3±686.9
ACO (Luna, Estébanez *et al.* 2008b)	90736.3	93439.5±1318.9	89946.3	92325.4±1092.8	89305.9	90649.9±727.5
ssGA (Luna, Estébanez *et al.* 2008b)	87477.3	89540.4±991.1	86755.7	87850.8±573.6	85720.3	86908.9±379.8
SS (Luna, Estébanez *et al.* 2008b)	91216.7	94199.6±1172.3	91069.8	93953.9±1178.6	91069.8	93820.4±1192.3
(1+2)EA (Luna, Estébanez *et al.* 2008b)	87763.9	92294.0±1407.6	86064.8	89669.8±1164.8	85607.3	88574.3±1100.3
LSHR (Luna, Estébanez *et al.* 2008b)	88543.0	92061.7±585.3	88031.0	89430.9±704.2	87743.0	88550.3±497.0
SS (Chaves *et al.* 2008b, 2009b)	86169.4	88692.7±1124.9	84570.6	86843.8±950.5	84234.5	85767.6±686.3
DE (Chaves, Maximiano *et al.* 2008c)	92145.8	95414.2±1080.4	89386.4	90587.2±682.3	87845.9	89116.8±563.8
Grid EA (Luna, Nebro *et al.* 2008a)	Results obtained in 1.5 hours: Best = 83,991.3, $\overline{x} \pm \sigma$ = 84,936.3±375.8					

Figure 1. Hyper-heuristic and heuristic search spaces

Figure 2. General outline of the proposed approach.

Figure 3. Topologies of the GSM instances used in the present study

Figure 4. Average result evolution for the different synchronization experiments performed using the Denver and the Seattle FAP instances

Figure 5. Average metaheuristic contributions obtained when the best sync configuration is established to solve both problem instances

Figure 6. Comparison between the results obtained by the different metaheuristics separately and the final results obtained by the PHH system

Figure 7. Results obtained when the proposed approach is supported using different numbers of cores

References

Aardal, K. I., van Hoesel, S. P. M., Koster, A. M. C. A., Mannino, C. and Sassano, A., 2007. Models and solution techniques for frequency assignment problems. *Annals of Operations Research*, 153 (1), 79–129.

Aickelin, U., Burke, E.K. and Li, J., 2006. An Estimation of Distribution Algorithm with Intelligent Local Search for Rule-based Nurse Rostering. *Journal of the Operational Research Society*, Palgrave Macmillan.

Alabau, M., Idoumghar, L., Schott, R., 2002. New hybrid genetic algorithms for the frequency assignment problem. *IEEE Transactions on Broadcasting*, 48 (1), 27–34.

Alba, E., and Tomassini, M., 2002. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5), 443–462.

Alba, E., Luna, F., and Nebro, A.J., 2004. Parallel heterogeneous genetic algorithms for continuous optimization. *Parallel Computing*, 30 (5-6), 669–719.

Alba, E., editor, 2005. *Parallel Metaheuristics: A New Class of Algorithms*. Hoboken, New Jersey: Wiley.

Alba, E., Chicano, F., Nesmachnow, S., Cancela, H., 2005. Parallel Metaheuristics in Telecommunications, E. Alba (editor), *Parallel Metaheuristics. A New Class of Algorithms*, chapter 20, Wiley, 495–516.

Amaldi, E., Capone, A., Malucelli, F., Mannino, C., 2006. Optimization Problems and Models for Planning Cellular Networks. *Handbook of Optimization in Telecommunications*. Springer, 917–939.

Ayob, M., Kendall, G., 2003. A Monte Carlo Hyper-Heuristic To Optimise Component Placement Sequencing For Multi Head Placement Machine. *International Conference on Intelligent Technologies (InTech'03)*, Thailand, 132–141.

Bäck, T., Fogel, D.B. and Michalewicz, Z, editors, 1997. *Handbook of Evolutionary Computation*. Oxford University Press.

Bai, R., Kendall, G., 2003. An Investigation of Automated Planograms Using a Simulated Annealing Based Hyper-heuristics, in T. Ibaraki, K. Nonobe, M. Yagiura (eds.), *Meta-heuristics: Progress as Real Problem Solvers, Selected Articles from the 5th Metaheuristics International Conference* (MIC 2003), Springer, 87–108.

Bai, R., Burke, E.K., and Kendall, G., 2008. Heuristic, meta-heuristic and hyper-heuristic approaches for fresh produce inventory control and shelf space allocation. *Journal of the Operational Research Society*, 59 (10), 1387–1397.

Baluja, S., 1994. *Population-Based Incremental Learning: A method for integrating genetic search based function optimization and competitive learning.* Technical Report CS-94–163, Carnegie Mellon University.

Björklund, P., Värbrand, P., and Yuan, D., 2005. Optimized Planning of Frequency Hopping in Cellular Networks, *Computers and Operations Research*, 32 (1), 169–186.

Burke, E.K., and Kendall, G., 2005. *Search methodologies: introductory tutorials in optimization and decision support techniques*. Springer.

Burke, E.K., Kendall, G., and Soubeiga, E., 2003. A Tabu-Search Hyper-Heuristic for Timetabling and Rostering. *Journal of Heuristics*, 9 (6), 451–470.

Burke, E.K., Kendall, G., Newall, J., Hart, E., Ross, P., and Schulenburg, S., 2003. Hyper-Heuristics: An Emerging Direction in Modern Search Technology, *Handbook of Metaheuristics* (eds. F.Glover and G.Kochenberger), Kluwer, 457–474.

Burke, E.K., Landa Silva, J.D., Soubeiga, E., 2005. Multi-objective Hyper-heuristic Approaches for Space Allocation and Timetabling. In: T. Ibaraki, K. Nonobe, M. Yagiura (eds.), *Meta-heuristics: Progress as Real Problem Solvers*, Springer.

Burke, E.K., MacCarthy, B., Petrovic, S., Qu. R., 2002. Knowledge Discovery in a Hyper-Heuristic Using Case-Based Reasoning on Course Timetabling. *4th International Conference on the Practice and Theory of Automated Timetabling*, Gent, Belgium, Springer LNCS 2740, 276–286.

Burke, E.K., Meisels, A., Petrovic, S., Qu., R., 2007. A Graph-Based Hyper Heuristic for Timetabling Problems. *European Journal of Operational Research*, 176, 177–192.

Burke E.K. and Newall J.P., 2004. Solving Examination Timetabling Problems through Adaptation of Heuristic Orderings, *Annals of Operations Research*, 129, 107–134.

Burke, E.K., Petrovic, S., and Qu., R., 2006. Case Based Heuristic Selection for Timetabling Problems, *Journal of Scheduling*, 9 (2), 1094–1136.

Blum, C., and Roli, A., 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35 (3), 268–308.

Borndörfer, R., Eisenblätter, A., Grötschel, M., Martin, A., 1998. Frequency assignment in cellular phone networks. *Annals of Operations Research*, 76, 73–93.

Chakhlevitch, K., and Cowling, P.I., 2005. Choosing the Fittest Subset of Low Level Heuristics in a Hyperheuristic Framework. In *European Conference on Evolutionary Computation in Combinatorial Optimization* (EvoCOP2005), 3448, 23–33.

Chaves-González, J.M., Hernando-Carnicero, R., Vega-Rodríguez, M.A., Gómez-Pulido J.A. and Sánchez-Pérez, J.M., 2009a. Solving a Realistic FAP Using GRASP and Grid Computing. In *Advances in Grid and Pervasive Computing*, Springer, 79–90.

Chaves-González, J.M., Vega-Rodríguez, M.A., Gómez-Pulido, J.A., and Sánchez-Pérez, J.M., 2009b. Improving Scatter Search to Solve a Real-World FAP. *Computer Aided Systems Theory - Extended Abstracts. Ed.: A. Quesada, J. Rodriguez, R. Moreno*. IUCTC. Universidad de Las Palmas de Gran Canaria, Spain, 271–273.

Chaves-González, J.M., Vega-Rodríguez M.A., Domínguez-González, D., Gómez-Pulido J.A. and Sánchez-Pérez, J.M, 2009c. Tuning the PBIL algorithm to solve a real-world FAP problem. *International Journal of Reasoning-based Intelligent System (IJRIS)*, Inderscience Publishers, 43–52.

Chaves-González, J.M., Domínguez-González, D., Vega-Rodríguez, M.A., Gómez-Pulido, J.A., Sánchez-Pérez J.M., 2008a. Parallelizing PBIL for Solving a Real-World Frequency Assignment Problem in GSM Networks. In *16th Euromicro Conference on Parallel, Distributed and Network-Based Processing* (PDP 2008), pp. 391–398.

Chaves-González, J.M., Vega-Rodríguez, M.A., Domínguez-González, D., Gómez-Pulido, J.A. and Sánchez-Pérez, J.M., 2008b. SS vs PBIL to Solve a Real-World Frequency Assignment Problem in GSM Networks. In *Applications of Evolutionary Computing, (EvoWorkshops 2008, LNCS 4974)*, 21–30.

Chaves-González, J.M. da Silva-Maximiano, M., Vega-Rodríguez, M.A., Gómez-Pulido J.A. and Sánchez-Pérez, J.M., 2008c. Comparing Hybrid Versions of SS and DE to Solve a Realistic FAP Problem. In *Hybrid Artificial Intelligence System (HAIS 2008)*, LNAI 5271, 257–264.

Chaves-González, J.M., Vega-Rodríguez, M.A., Domínguez-González, D., Gómez-Pulido J.A., and Sánchez-Pérez, J.M., 2008d. Population-Based Incremental Learning to

solve the FAP problem. In *The Second International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2008)*, 123–128.

Chaves-González, J.M., Vega-Rodríguez, M.A., Domínguez-González, D., Gómez-Pulido J.A., and Sánchez-Pérez, J.M., 2008e. Studying different variants of PBIL to solve a real-world FAP problem in GSM networks. In *The Second International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2008)*, 83–88.

Chiarandini, M., and Stützle, T., 2002. An application of Iterated Local Search to Graph Coloring Problem. In *Proceedings of the Computational Symposium on Graph Coloring and its Generalizations*, 112–125.

Colombo, G., 2006. A genetic algorithm for frequency assignment with problem decomposition. *International Journal of Mobile Network Design and Innovation*, 1 (2), 102–112.

Cowling, P., Chakhlevitch, K., 2003. Hyperheuristics for managing a large collection of low level heuristics to schedule personnel, in *Congress on Evolutionary Computation*, 2, 1214–1221.

Cowling, P., Kendall, G., and Han, L., 2002. An Adaptive Length Chromosome Hyperheuristic Genetic Algorithm for a Trainer Scheduling Problem, in *the 4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL'02)*, Singapore, 267–271.

Cowling, P., Kendall, G., and Soubeiga, E., 2000. A Hyper-heuristic Approach to Scheduling a Sales Summit, *PATAT III, LNCS 2079*, Konstanz, Germany, selected articles (eds E.K. Burke and W. Erben), 176–190.

Crainic, T.G., and Toulouse, M., 2003. Parallel strategies for metaheuristics. In F. W. Glover and G. A. Kochenberger, editors, *Handbook of Metaheuristics*, Norwell, MA, USA, Kluwer Academic Publishers.

Crisan, C., and Mühlenbein, H., 1998. The breeder genetic algorithm for frequency assignment. In *Parallel Problem Solving from Nature (PPSN V)*, 897–906.

Crompton, W., Hurley, S., and Stephens, N.M., 1994. A parallel genetic algorithm for frequency assignment problems. In *Proceedings of the IMACS/IEEE Conference on Signal Processing, Robotics and Neural Networks*, 81–84.

Cuesta, A., Garrido, L., and Terashima, H., 2005. Building Hyper-heuristics through Ant Colony Optimization for the 2D Bin Packing Problem, *KES'05 in LNCS 3684*, 654–660.

Cung, V.D., Martins, S.L., Ribeiro, C.C., and Roucairol, C, 2003. Strategies for the Parallel Implementation of Metaheuristics. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, Norwell, MA, USA, Kluwer Academic Publishers, pp. 263–308.

Cuppini, M., 1994. A genetic algorithm for channel assignment problems, *European Transactions on telecommunications and related technologies*, 5, 285–294.

Dorigo, M., Stützle, T., 2004. Ant Colony Optimization. MIT Press, Cambridge, MA.

Dorne, R., and Hao J.K., 1995. An evolutionary approach for frequency assignment in cellular radio networks. In *IEEE International Conference on Evolutionary Computation*, 539–544.

Dowsland, K.A., Soubeiga, E., and Burke, E.K., 2007. A simulated annealing hyper-heuristic for determining shipper sizes. *European Journal of Operational Research*, 179 (3), 759–774.

Eisenblätter A., 2001. *Frequency Assignment in GSM Networks: Models, Heuristics, and Lower Bounds*. PhD Thesis. Technische Universität Berlin.

FAP Web, 2007. http://fap.zip.de/.

Feo T.A. and Resende, M.G.C., 1995. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 6, 109–133.

Fischetti, M., Lepsch, C., Minerva, G., Romanin-Jacur. G., Toto, E, 2000. Frequency assignment in mobile radio systems using branch-and-cut techniques. *European Journal of Operational Research*, 123 (2) 241–255.

Funabiki, N., Takefuji, Y., 1992. A neural network parallel algorithm for channel assignment problems in cellular radio networks. *IEEE Transactions on Vehicular Technology*, 41 (4), 430–437.

Galinier, P., Hertz, A., 2006. A survey of local search methods for graph coloring. *Computers and Operations Research*, 33 (9), 2547–2562.

GCC Compiler Home Page, 2010. http://gcc.gnu.org/.

Glover, F.W., and Kochenberger, G.A., 2003. *Handbook of Metaheuristics (International Series in Operations Research & Management Science)*. Springer.

Glover, F., Laguna, M., and Martí, R., 2003. Scatter search. In *Advances in Evolutionary Computing: Theory and Applications*, Springer, 519–537.

Gomes, F.C., Pardalos, P., Oliveira, C.S. Resende, M.G.C., 2001. Reactive GRASP with path relinking for channel assignment in mobile phone networks. In *Proceedings of the 5th international workshop on Discrete algorithms and methods for mobile computing and communications (DIALM'01)*, 60–67.

Greff, J.Y., Idoumghar, L., Schott, R., 2004. Application of markov decision processes to the frequency assignment problem. *Applied Artificial Intelligence*, 18 (8), 761–773.

GSM World statistics, 2010. http://www.gsmworld.com/newsroom/market-data/market_data_summary.htm.

Hale, W.K., 1980. Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68 (12), 1497–1514.

Hamiez, H.P., Hao, J.K., 2002. Scatter search for graph coloring. In *Artificial Evolution*, 2310, 195–213.

Hart, E., Ross, P., Nelson, J., 1998. Solving a Real-World Problem Using an Evolving Heuristically Driven Schedule Builder, Evolutionary Computation 6 (1), 61–80.

Holland, J.H, 1992. *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, MA, USA, (first edition published in 1975).

Idoumghar, L., and Schott, R., 2009. Two Distributed Algorithms for the Frequency Assignment Problem in the Field of Radio Broadcasting. *IEEE Transactions on Broadcasting*, 55 (2), 223–229.

Idoumghar, L., Schott, R., 2006. A New Hybrid GA-MDP Algorithm For The Frequency Assignment Problem. In *8th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*, 18–25.

Karaboga, D., and Basturk, B, 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39, 459–471.

Karaboga, D., 2005. *An idea based on honey bee swarm for numerical optimization*. Technical Report- TR06, Erciyes University, Engineering Faculty, Computer Engineering Department.

Kendall, G., Mohamad, M., 2004. Channel assignment optimisation using a hyper-heuristic. In *IEEE Conference on Cybernetics and Intelligent Systems*, 2, 791–796.

Kendall, G., Mohamad, M., 2004. Channel Assignment in Cellular Communication Using a Great Deluge Hyper-Heuristic. In *IEEE International Conference on Network (ICON2004)*, 769–773.

Koster, A.M.C.A., 1999. *Frequency assignment - models and algorithms*, Ph.D. thesis, Universiteit Maastricht.

Kotrotsos, S., Kotsakis, G., Demestichas, P., Tzifa, E., Demesticha, V., and Anagnostou, V, 2001. Formulation and computationally efficient algorithms for an interference-oriented version of the frequency assignment problem. *Wireless Personal Communications*, 18, 289–317.

Kuurne, A.M.J., 2002. On GSM mobile measurement based interference matrix generation. In *IEEE 55th Vehicular Technology Conference*, 1965–1969.

Leese, R., Hurley, S., 2002. Methods and Algorithms for Radio Channel Assignment. *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press.

Liu, X., Pardalos, P.M., Rajasekaran, S., and Resende, M.G.C., 2000. A GRASP for frequency assignment in mobile radio networks. *Dimacs series on discrete mathematics and theoretical computer science*, 52, 195–201.

Lourenço, H.R., Martin, O. and Stützle, T, 2002. *Handbook of Metaheuristics, chapter Iterated local search*, Kluwer Academic Publishers, 321–353.

Luna, F., Blum, C., Alba, E., Nebro, A.J., 2007b. ACO vs EAs for Solving a Real-World Frequency Assignment Problem in GSM Networks. *GECCO'07*, London, UK, 94–101.

Luna, F., Alba, E., Nebro, A.J., Pedraza, S., 2007a. Evolutionary algorithms for real-world instances of the automatic frequency planning problem in GSM networks. In *Seventh European Conference on Evolutionary Computation in Combinatorial Optimization (EVOCOP 2007)*, LNCS 4446, 108–120.

Luna, F., Nebro, A.J., Alba, E., Durillo, J.J., 2008a. Solving large-scale real-world telecommunication problems using a grid-based genetic algorithm. *Engineering Optimization*, 40 (11), 1067–1084.

Luna, F., Estébanez, C., León, C., Chaves-González, J.M., Alba, E., Aler, R., Segura, C., Vega-Rodríguez, M.A., Nebro, A.J., Valls, J.M., Miranda, G., Gómez-Pulido, J.A., 2008b. Metaheuristics for Solving a Real-World Frequency Assignment Problem in GSM Networks. In *Genetic and Evolutionary Computation Conference (GECCO 2008)*, 1579–1586.

Mabed, H., Caminada, A., Hao, J.K., Renaud, D., 2002. A dynamic traffic model for frequency assignment. *Parallel Problem Solving from Nature (PPSN VII)*, LNCS 2439, 779–788.

Maniezzo, V., Carbonaro, A., 1999. An ANT Heuristic for the Frequency Assignment Problem. *Future Generation Computer Systems*, 16, 927–935.

Mannino, C., Sassano, A., 2003. An enumerative algorithm for the frequency assignment problem. *Discrete Applied Mathematics*, 129, 155–169.

Maple, C., Guo, L., and Zhang, J., 2004. Parallel genetic algorithms for third generation mobile network planning. In *International Conference on Parallel Computing in Electrical Engineering*, 229–236.

Martí, R., Laguna, M., Glover, F., 2006. Principles of scatter search. *European Journal of Operational Research*, 169 (2), 359–372.

Matsui, S., Watanabe, I., Tokoro, K.I., 2003. An efficient hybrid genetic algorithm for a fixed channel assignment problem with limited bandwidth. In *Genetic and Evolutionary Computation Conference (GECCO 2003)*, 2240–2251.

Metzger, B.H., 1970. *Spectrum management technique*. 38th National ORSA Meeting.

Mladenovic, N., and Hansen, P., 1997. Variable neighborhood search. *Computers and Operations Research*, 24 (11), 1097–1100.

Mouly, M., and Paulet, M.B., 1992. The GSM System for Mobile Communications. *Mouly and Paulet*, Palaiseau.

MPI Forum home page, 2010. http://www.mpi-forum.org/index.html.

Ozcan, E., Bilgin, B., Korkmaz, E.E., 2008. A Comprehensive Analysis of Hyper-heuristics, *Intelligent Data Analysis*, 12 (1), 3–23.

Paunovic, D., Neskovic, N., Neskovic, A., 2001. Automatic frequency planning algorithm in a real land mobile radio system design. *5th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Service (TELSIKS 2001)*, 511–520.

Remde, S.M., Cowling, P.I., Dahal, K.P., and Colledge, N., 2007. Exact/Heuristic Hybrids using rVNS and Hyperheuristics for Workforce Scheduling. *European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP2007)*, LNCS 4446, 188–197.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P, 2002. *Numerical Recipes in C++: The Art of Scientific Computing*. Second edition, Cambridge University Press.

Ross, P., 2005. Hyper-heuristics, Search Methodologies: Introductory Tutorials. In *Optimization and Decision Support Techniques* (eds. E.K. Burke & G. Kendall), Springer, 529–556.

Ross, P., Schulenburg, S., Marín-Blázquez J.G., and Hart, E., 2002. Hyper-heuristics: learning to combine simple heuristics in bin-packing problems, *Genetic and Evolutionary Computation Conference (GECCO 2002)*, New York, 942–948.

Salcedo-Sanz, S., Bousoño-Calzón, C., 2005. A portable and scalable algorithm for a class of constrained combinatorial optimization problems. *Computers & Operations Research*, 32, 2671–2687.

San Jose-Revuelta, L.M., 2007. A new adaptive genetic algorithm for fixed channel assignment. *Information Sciences*, 177 (13), 2655–2678.

Scientific Linux Home Page, 2010. https://www.scientificlinux.org/

Simon M.K. and Alouini M-S., 2005. *Digital Communication over Fading Channels: A Unified Approach to Performance Analysis*, Wiley.

Talbi, E-G., 2002. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(2) 807–819.

Touhami, S., Bourjolly, J.-M., and Laporte, G., 2010. Optimizing Hopping Sequences for Reducing Interference in Frequency Hopping. Cellular Networks, *Engineering Optimization*, 42(1), 33–44.

Vieira, C.E.C., Gondim, P.R.L., Rodrigues, C.A., Bordim, J.L., 2008. A new technique to the channel assignment problem in mobile communication networks. In *IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2008)*, 1–5.

Walke, B.H., 2002. *Mobile Radio Networks: Networking, Protocols and Traffic Performance*. Wiley.

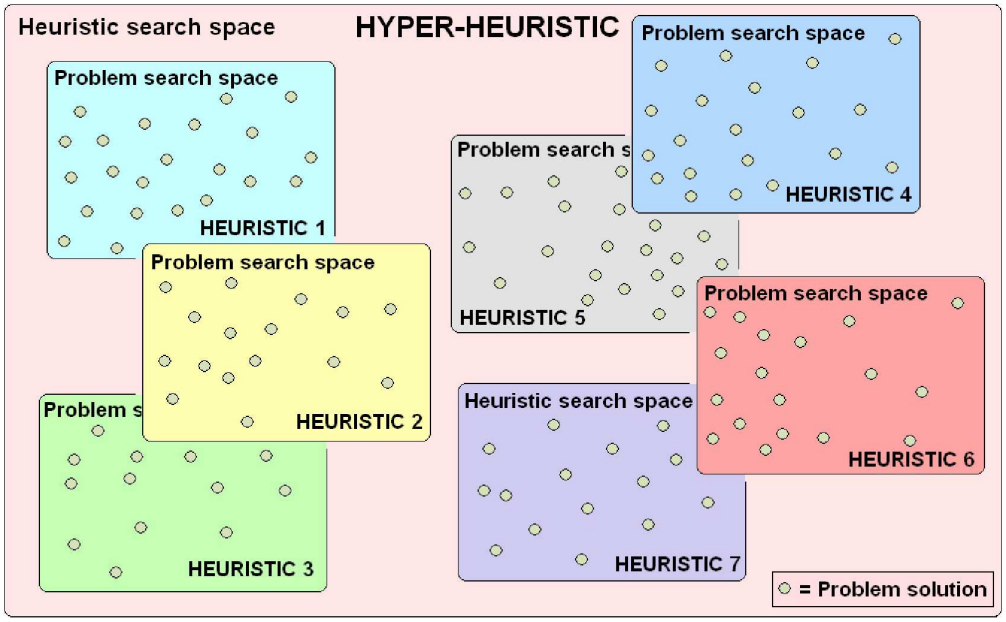

Figure 1
354x217mm (600 x 600 DPI)

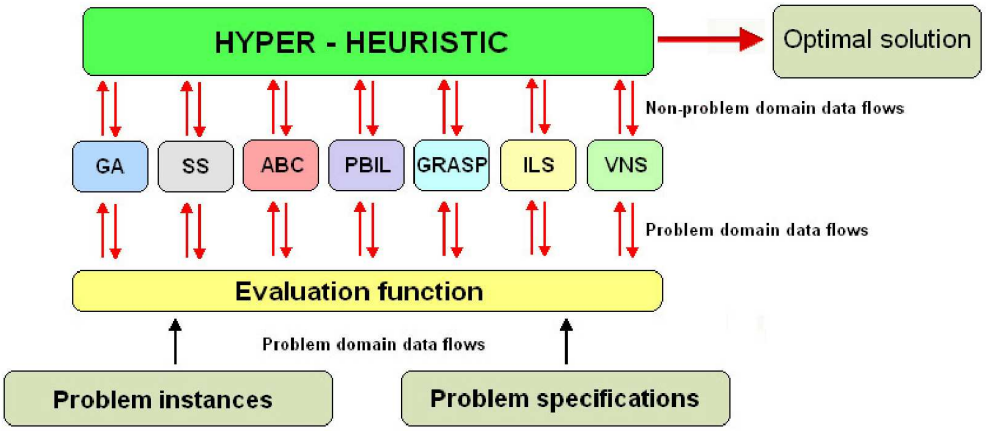

Figure 2
277x137mm (600 x 600 DPI)

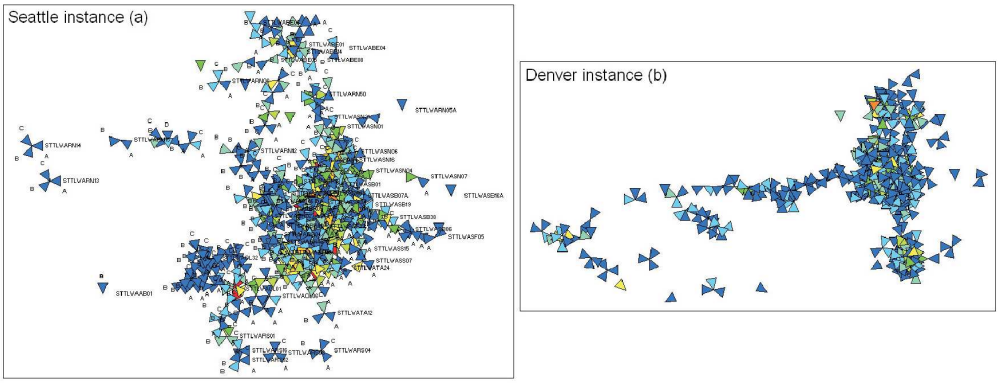

Figure 3
592x226mm (600 x 600 DPI)

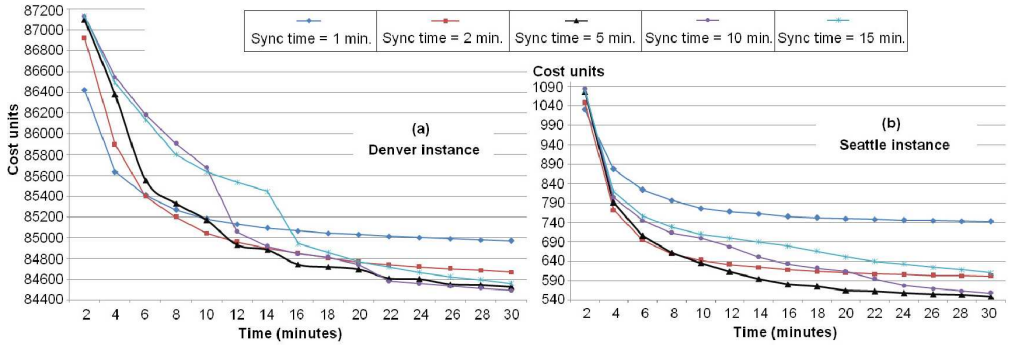

Figure 4
581x213mm (600 x 600 DPI)

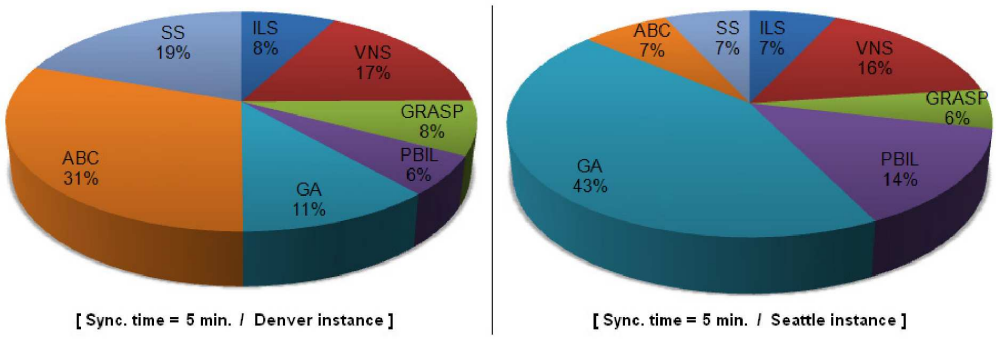

Figure 5
317x108mm (600 x 600 DPI)

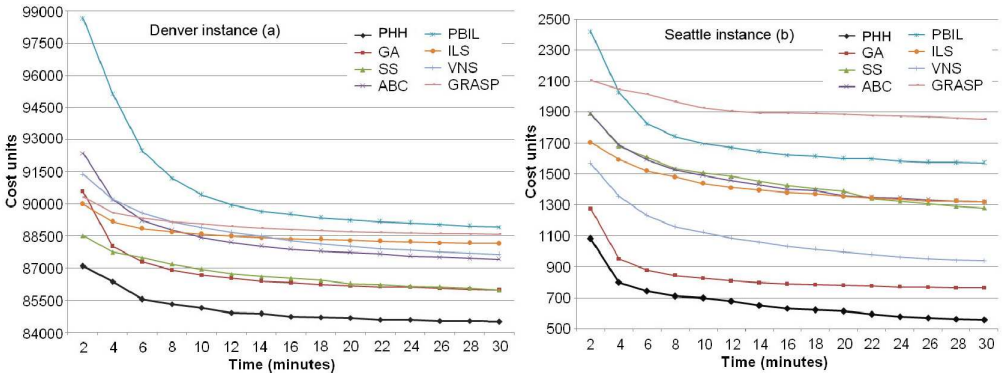

Figure 6
565x208mm (600 x 600 DPI)

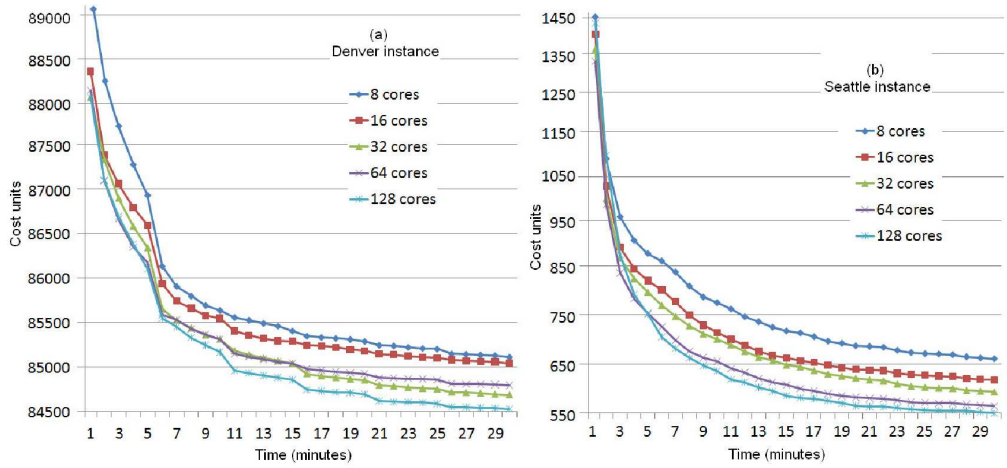

Figure 7
499x234mm (600 x 600 DPI)