Engineering Optimization
Vol. 40, No. 11, November 2008, 1067–1084

Solving large-scale real-world telecommunication problems using a grid-based genetic algorithm

Francisco Luna*, Antonio J. Nebro, Enrique Alba and Juan J. Durillo

Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, Málaga, Spain

(*Received 26 December 2007; final version received 13 June 2008*)

This article analyses the use of a grid-based genetic algorithm (GrEA) to solve a real-world instance of a problem from the telecommunication domain. The problem, known as automatic frequency planning (AFP), is used in a global system for mobile communications (GSM) networks to assign a number of fixed frequencies to a set of GSM transceivers located in the antennae of a cellular phone network. Real data instances of the AFP are very difficult to solve owing to the NP-hard nature of the problem, so combining grid computing and metaheuristics turns out to be a way to provide satisfactory solutions in a reasonable amount of time. GrEA has been deployed on a grid with up to 300 processors to solve an AFP instance of 2612 transceivers. The results not only show that significant running time reductions are achieved, but that the search capability of GrEA clearly outperforms that of the equivalent non-grid algorithm.

Keywords: frequency assignment problem; real-world problem solving; grid computing; genetic algorithms

1. Introduction

Frequency assignment is a well-known problem in Operations Research (Aardal *et al.* 2007) and it is of great importance in real global systems for mobile communications (GSM) networks (Mouly and Paulet 1992). In these networks, the available frequency band is slotted into channels (or frequencies) that have to be allocated to the elementary transceivers (TRXs) installed in the base stations of the network. This problem is known as automatic frequency planning (AFP), the frequency assignment problem (FAP), or even the channel assignment problem (CAP) (Eisenblätter 2001). An optimal frequency assignment allows the capacity of the networks to be increased by avoiding the interferences provoked by channel reuse owing to the limited available radio spectrum, thus improving the quality of service for subscribers and an income for the operators as well.

The AFP problem is an NP-hard problem (Hale 1980) that is even more difficult to address when defined in the context of GSM networks. In such a scenario, solving the problem properly requires realistic and precise interference information from a real-world GSM network in order for an accurate frequency plan to be computed. This information has to consider actual technologies currently used by GSM operators such as frequency hopping (Eisenblätter 2001). This leads to further difficulties for telecommunication companies which, along with the complexity of the

*Corresponding author. Email: flv@lcc.uma.es

ISSN 0305-215X print/ISSN 1029-0273 online

DOI: 10.1080/03052150802294581
http://www.informaworld.com

problem itself and the large size of real-world networks, need accurate physical models of the GSM system (antennae, propagation, etc.). In this context, solving the AFP problem is a task demanding both numerical and computational power in order to overcome the difficulties for finding satisfactory solutions. The approach used in this work to cope with these two practical requirements lies in using metaheuristics and grid computing.

Metaheuristics (Blum and Roli 2003) are a broad family of approximate techniques that can be used to solve optimization problems. Contrary to exact techniques, metaheuristics do not guarantee to find optimal solutions to the problems, but they allow good compromise solutions to be reached in a reasonable amount of time. A metaheuristic can be defined as a high-level strategy that controls a number of subordinated techniques (usually heuristics) in the search for an optimum. These techniques are nowadays widely used. Among them, evolutionary algorithms (EAs) and, in particular, a subfamily of them, genetic algorithms (GAs), have become very popular.

On the other hand, grid computing (Foster and Kesselman 1999, Berman *et al.* 2003) encompasses a number of issues related to the use of large-scale distributed systems as a unique parallel computer. Grid computing systems are a natural evolution of distributed systems in the way that the infrastructure provided by the Internet allows hundreds and thousands of computers to be joined, leading to a computing power that even supercomputers are unable to provide; this way, algorithms that otherwise would be considered as unfeasible can be executed in a reasonable amount of time.

In this article, the performance of a distributed metaheuristic, a genetic algorithm called the grid-based evolutionary algorithm (GrEA) (Nebro *et al.* 2008) is analysed. The GrEA is designed to be executed in a grid computing system based on Condor (Thain *et al.* 2003), a grid computing software. This algorithm has been applied to solve a real-world instance of the AFP problem that corresponds to Denver, a city in Colorado, USA, of more than half a million people. This network is composed of 2612 TRXs that have to be assigned with 18 different available frequencies, so leading to a huge search space of size $18^{2612} \approx 5.11e^{3263}$. A novel formulation of the problem presented in Luna *et al.* (2007) has been used, which is directly imported from real-world GSM frequency planning as currently conducted in the industry.

Even though the instances of any formulation of the problem are potentially very large (*e.g.* the size of current cellular networks is continuously increasing), few works approach this problem with parallel algorithms for addressing the highly increasing computational resources required. In the field of EAs, Crompton *et al.* (1993,1994) presented a distributed GA (Alba and Troya 1999) that uses two different encodings for the individuals and different recombination operators. However, no details on the parallel computing platform are given. A parallel GA for hybrid channel assignment has been proposed by Kwok (2000). The algorithm runs on a cluster composed of twelve Linux workstations. Many other works exist in which the search model is parallel (Alba and Tomassini 2002), but the execution is carried out on sequential machines (*e.g.* Weinberg *et al.* 2001, Alabau *et al.* 2002, Matsui *et al.* 2005). This is also a typical scenario in actual telecommunication companies, where single computers—mostly laptops—are used to perform the optimization. Going one step further here, the AFP problem has been tackled in a grid computing platform with the aim of solving a very large instance of the problem in an affordable wall clock time.

The contributions of the present work can be summarized in the following points.

- A grid-based genetic algorithm has been used to solve a complex instance of the AFP problem. To the best of our knowledge, this is the first time this kind of problem has been addressed with grid technologies (using up to 300 processors).
- An accurate statistical analysis has been carried out to validate the results obtained.
- The study indicates that the search capabilities of GrEA outperform those of its sequential counterpart.
- As a result of the experiments, the best solution known so far to the problem considered has been obtained.

The rest of the article is structured as follows. In the next section, the reader is provided with the details of frequency planning in GSM networks. Section 3 details GrEA, the grid-based GA approach. Some implementation details are given in the following section. In Section 5, the experimental results are presented and analysed. Finally, the conclusions and lines of future work are included in Section 6.

2. Frequency assignment in GSM networks

In the following, a brief description of the GSM architecture is provided first, whereby the basic terminology of the problem is introduced. Next, details of the frequency planning task in GSM networks are given. Finally, a precise mathematical formulation of the AFP model addressed in this article is presented.

2.1. *The GSM system*

An outline of the GSM network architecture is shown in Figure 1. As can be seen, GSM networks consist of many different components. The most relevant ones to frequency planning are base transceiver stations (BTSs) and transceivers (TRXs). Essentially, a BTS is a set of TRXs. In GSM, one TRX is shared by up to eight users in time division multiple access (TDMA) 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 at which a BTS is installed is usually organized in sectors: one to three sectors are typical. The area in which each sector operates defines a cell.

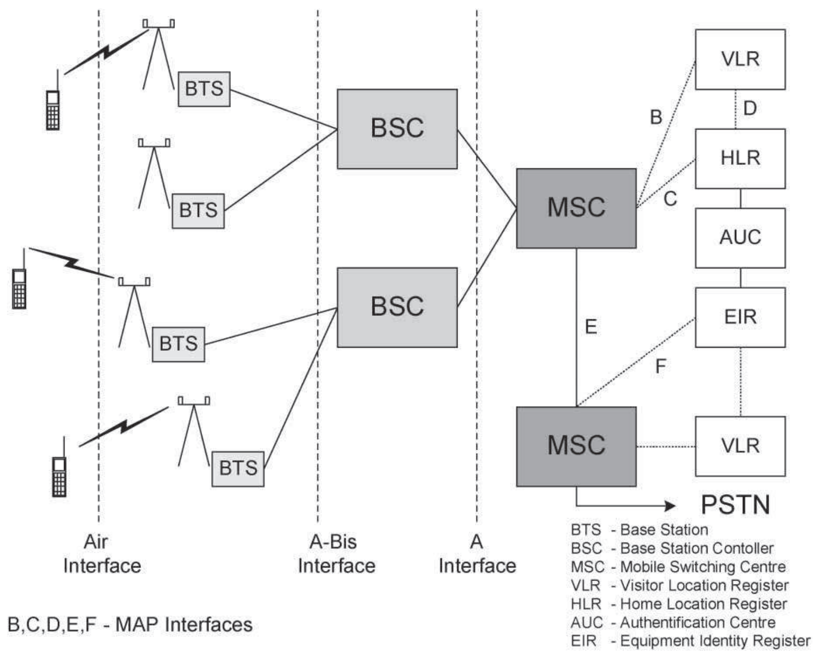

Figure 1. Outline of the GSM network architecture.

The solid lines connecting components in Figure 1 carry both traffic information (voice or data) as well as the 'in-band' signalling information. The dotted lines are signalling lines. The information exchanged over these lines is necessary for supporting user mobility, network features, operation and maintenance, authentication, encryption and many other functions necessary for the network's proper operation.

2.2. *Frequency planning in GSM networks*

Frequency planning is the last step in the layout of a GSM network. Prior to tackling this problem, the network designer has to address some other issues: where to install the BTSs or how to set the configuration parameters of the antennae (tilt, azimuth, etc.), among others (Mishra 2004). Once the sites for the BTSs are selected and the sector layout is decided, the number of TRXs to be installed per sector has to be fixed. This number depends on the traffic demand that the corresponding sector has to hold. Frequency planning lies in the assignment of a channel (a frequency) to every TRX (Eisenblätter 2001). The optimization problem arises because the usable radio spectrum is generally very scarce and, consequently, frequencies have to be reused by many TRXs in the network.

The multiple use of the same frequency may cause interferences that may reduce the quality of service (QoS) down to unsatisfactory levels. Indeed, significant interference may occur if the same or adjacent channels are used in neighbouring, overlapping cells. The point here is that computing this level of interference is a difficult task, which depends not only on the channels, but also on the radio signals and the properties of the 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. Several ways of quantifying this interference exist, ranging from theoretical methods to extensive measurements (Kuurne 2002). They all result in a so-called *interference matrix*, denoted by M. Each element $M(i, j)$ of M indicates the degradation of the network quality if cells i and j operate on the same frequency. This is called *co-channel interference*. Apart from co-channel interference, so-called *adjacent-channel interference* may exist, which occurs when two TRXs operate on adjacent channels (*i.e.* one TRX operates on channel f and the other on channel $f + 1$ or $f - 1$). An accurate interference matrix is therefore an essential requirement for frequency planning because the ultimate goal of any frequency assignment algorithm will be to minimize the sum of the interferences.

In real-life situations, additional complicating factors such as separation constraints among cells, or advanced interference reduction techniques such as frequency hopping or dynamic power control, may be considered. The interested reader is referred to Eisenblätter (2001) for a more detailed description of frequency planning in actual GSM networks.

2.3. *Mathematical formulation*

Let $T = \{t_1, t_2, \ldots, t_n\}$ be a set of n *transceivers*, and let $F_i = \{f_{i1}, \ldots, f_{ik}\} \subset \mathbb{N}$ be the set of valid *frequencies* that can be assigned to a transceiver $t_i \in T$, $i = 1, \ldots, n$. Note that k—the cardinality of F_i—is not necessarily the same for all the transceivers. Furthermore, let $S = \{s_1, s_2, \ldots, s_m\}$ be a set of given *sectors* (or cells) of cardinality m. Each transceiver $t_i \in T$ is installed in exactly one of the m sectors. Henceforth, we denote the sector in which a transceiver t_i is installed by $s(t_i) \in S$. Finally, a matrix $M = \{(\mu_{ij}, \sigma_{ij})\}_{m\times m}$ is given, called the *interference matrix*. The two elements μ_{ij} and σ_{ij} of a matrix entry $M(i, j) = (\mu_{ij}, \sigma_{ij})$ are numerical values greater than or equal to zero. 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) (Walke 2002) when sectors i and j operate on the same frequency. The higher the mean value, the lower the interference and thus the better the communication quality. Note that the interference matrix is defined at sector (cell) level, because the transceivers installed in each sector all serve the same area.

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

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

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

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

$K >> 0$ is a very large constant defined by the network designer so as to make it undesirable to allocate the same or adjacent frequencies to transceivers serving the same area. Furthermore, function $C_{\text{co}}(\mu, \sigma)$ is defined as follows:

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

where

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

is the tail integral of a Gaussian probability distribution function with zero mean and unit variance, and c_{SH} is a minimum quality signalling threshold. Function Q is widely used in digital communication systems because it characterizes the error probability performance of digital signals (Simon and Alouini 2005). This means that $Q[(c_{\text{SH}} - \mu)/\sigma]$ is the probability of the C/I ratio being greater than c_{SH} and, therefore, $C_{\text{co}}(\mu_{s_t s_u}, \sigma_{s_t s_u})$ computes the probability of the C/I ratio in the serving area of sector s_t being below the quality threshold owing to the interferences provoked by sector s_u. That is, if this probability is low, the C/I value in the sector s_t is not likely to be degraded by the interfering signal coming from sector s_u and thus the communication quality yielded is high. (Note that this fits with the definition of a minimization problem.) On the contrary, a high probability—and consequently a high cost—causes the C/I mostly to be below the minimum threshold c_{SH}, thus incurring low-quality communications.

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

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

In Press *et al.* (1992), a numerical method is presented that allows the value of E to be computed with a fractional error smaller than 1.2×10^{-7}. Analogously, the function $C_{\text{adj}}(\mu, \sigma)$ is defined as

$$C_{\text{adj}}(\mu, \sigma) = 100\left[1.0 - Q\left(\frac{c_{\text{SH}} - c_{\text{ACR}} - \mu}{\sigma}\right)\right] = 100\left[1.0 - \frac{1}{2}E\left(\frac{c_{\text{SH}} - c_{\text{ACR}} - \mu}{\sigma\sqrt{2}}\right)\right]. \quad (6)$$

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

Our model ultimately aims at measuring the overall signalling performance of the GSM network. The keystone of this model is to be found in the definition of the interference matrix, which includes the entire probability distribution of the C/I ratio. This definition, 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), allows not only the computation of high-performance frequency plans, but also the prediction of QoS. Indeed, both the definition of the interference matrix and the subsequent computations to obtain the cost values are motivated by real-world GSM networks since they are related to the computation of the bit error rate (BER) performance of Gaussian minimum shift keying (GMSK), the modulation scheme used for GSM (Simon and Alouini 2005).

3. Using GrEA for solving the AFP problem

This section is devoted to presenting the algorithmic approach used for solving the AFP problem described in Section 2. Next, GrEA is introduced along with the representation of the individuals, the genetic operators applied, and a local search algorithm used to improve the solutions.

3.1. *GrEA*

GrEA (Nebro *et al.* 2008) is a steady-state GA (ssGA) following the master/worker parallel model. It has been developed using Condor (Thain *et al.* 2003) and the MW framework (Linderoth *et al.* 2000). GrEA is also a hybrid algorithm (Talbi 2002) since a local search method, which was specially designed for this AFP problem by Luna *et al.* (2007), is applied to the individuals that are generated after the recombination and mutation operators. The basic idea is that a master process executes the main loop of ssGA and the workers perform the function evaluations and the local search step in an asynchronous way. Contrary to a fully sequential version of ssGA, GrEA performs several individual evaluations in parallel; ideally, there should be as many parallel evaluations as available processors in the grid.

For better describing the algorithm, let us call GrEA-master the part of the algorithm corresponding to the master process, as opposed to the worker counterpart, named GrEA-worker. The pseudo-code of GrEA-master is described in Algorithm 1. GrEA-master starts by creating an empty population (line 1) and generating a task list, each task containing a randomly generated individual (line 2). The tasks in the list are sent to the available workers by the underlying Condor system (see Section 4.2). After these two steps, GrEA-master works in a reactive way: when a task

Algorithm 1 Pseudo-code for GrEA-master.

```
1: population ← ∅
2: Initialize taskList
3: while not stoppingCondition do
4:    Receive task
5:    individual ← task.individual
6:    Insert individual into population
7:    while new available workers do
8:       newIndividual ← GA_step()
9:       newTask ←new Task(newIndividual)
10:      taskList.add(newTask)
11:   end while
12: end while
```

is received from a worker (line 4), the individual contained in that task is extracted (line 5), and it is inserted into the population (line 6). Then, for each new available worker detected, the following steps are carried out: first, a GA step (selection, recombination and mutation) is executed (line 8), producing a new individual; secondly, this individual is added to a new task (line 9); finally, this task is inserted into the task list, which is ready to be sent to a worker (by Condor).

Algorithm 2 Pseudo-code for GrEA-worker.

```
while true do
   Receive task
   individual ← task.individual
   newIndividual ← LocalSearch(individual)
   newTask ←new Task(newIndividual)
   Return newTask
end while
```

The mission of GrEA-worker is to receive an individual, evaluate it, and apply the local search. Since the local search may modify the individual, it has to be returned back to GrEA-master. The pseudo-code of GrEA-worker is included in Algorithm 2.

Salient features of GrEA are the awareness of new processors and fault tolerance. These characteristics play a key role in order to make GrEA a grid-enabled algorithm. Thus, whenever a new processor is detected by GrEA, a GA step is performed and a new individual is obtained for evaluation in the worker that will be deployed in the new processor. Concerning fault tolerance, crashes in GrEA-master are automatically managed by Condor by using checkpointing; faults in the processes running GrEA-worker are simply ignored because they do not affect the working principles of the genetic algorithm. For further details, the reader is referred to Nebro *et al.* (2008).

3.2. *Solution encoding and genetic operators*

As defined in Section 2.3, a solution to the problem is obtained by assigning to each transceiver $t_i \in T$ one of the frequencies from F_i, the set of valid frequencies for TRX t_i. A solution is therefore encoded as an array of integer values, p, where $p(t_i) \in F_i$ is the frequency assigned to transceiver t_i. That is, the solutions manipulated by GrEA are tentative frequency plans of the given AFP problem instance. As to the genetic operators, binary tournament has been used as the selection scheme. This operator works by randomly choosing two individuals from the population and the one having the best (lowest) fitness is selected. GrEA applies a uniform crossover (UX) in which every allele of the offspring (*i.e.* the frequency of each TRX) is chosen randomly from one of the two parents with a probability of 0.5. Finally, the mutation operator used is random mutation, in which the frequencies of a set of randomly chosen TRXs of the solution are reassigned with a random valid frequency. Note that the two operators always assign valid frequencies to each TRX and no repair step is required.

It is well known that randomized genetic operators, and especially classical recombination operators, perform very badly in GAs for solving frequency assignment problems (Dorne and Hao 1995, Crisan and Mühlenbein 1998). However, when combined with an accurate local search method (hybridization), they achieve a good intensification/diversification tradeoff. This is the approach followed in this work: the highly randomized UX and random mutation are devoted to explore new regions of the search space, while the local search (see the next section) is designed to seek for accurate solutions located in these regions.

3.3. *Local search*

In order for a GA to perform well on AFP problems, its hybridization with a local search algorithm is almost mandatory. Indeed, the most recent and efficient GAs for solving several flavours of the problem are endowed with some kind of local search: a probabilistic Tabu Search in Alabau *et al.* (2002), an adaptation of Markov Decision Processes in Idoumghar and Schott (2006), the CAP3 method in Kim *et al.* (2007), or others specifically designed for the problem being solved (see Matsui *et al.* 2003, Colombo 2006). The usage of local search turned out also to be essential in ACO algorithms for solving frequency assignment problems (Graham *et al.* 2007), and particularly in the version of the AFP problem used, as shown in Luna *et al.* (2007).

The local search method used in this work is included in Algorithm 3. It first ranks the TRXs with respect to their component cost, CC. Given a frequency plan p and a TRX t, $CC(p, t)$ is defined as

$$CC(p,t) = \sum_{u \in T, u \neq t} C_{\text{sig}}(p,t,u), \tag{7}$$

Algorithm 3 Pseudo-code for the local search.

input: a solution p, a number of steps d
improved $\leftarrow$ **true**
$k \leftarrow 1$
while $k \leq d$ **and** *improved* = **true do**
    improved $\leftarrow$ **false**
    Rank every TRX t_i with $CC(p, t_i)$
    for $i \leftarrow 1$ to n **do**
        Replace frequency $p(t_i)$ with the frequency from F_i that most reduces the objective function value
        if the objective function value was reduced **then** *improved* = **true**
        Update $CC(p, t_i)$
    end for
    $k \leftarrow k + 1$
end while
output: a possibly improved solution p

i.e. $CC(p, t)$ is the value with which TRX t contributes to the total cost of the frequency plan p (Equation 2 defines C_{sig}). This ranking allows the TRXs occurring in the strongest interference to be assigned in the beginning so as to fix low-quality assignments quickly and to lead to further improvements. Then, all the TRXs are traversed and the frequency that most reduces the AFP cost of the entire plan (Equation 1) is chosen (line 8). In the AFP instance solved, this would require $2612 \times 18 = 47{,}016$ evaluations of the AFP cost at each step, which makes the local search unaffordable. Therefore, rather than using Equation (1) for computing the new objective function value, an incremental cost function has been used because the increase of the AFP cost caused by the setting $p(t_i) = f$ can be computed as follows:

$$\Delta(p, p(t_i) = f) = \sum_{t \in \hat{T}} (C_{\text{sig}}(p, t, t_i) + C_{\text{sig}}(p, t_i, t)). \tag{8}$$

Finally, the component cost of each TRX is updated and a new iteration starts (line 10 of Algorithm 3) so as to reassign again first those TRXs that most contribute to the AFP cost. Parameter d indicates for how many steps this algorithm should be maximally executed. After some preliminary tests with the instance solved in this work (see Section 5.2), the local search converges towards a local minimum after six steps and this is the value used for the subsequent experimentation.

4. Implementation details

In this section a brief introduction to Condor is given as well as details about how the MW library is used to implement GrEA.

4.1. *Condor*

Condor is a grid system software package designed to manage distributed collections (pools) of processors spread among a campus or other organizations (Thain *et al.* 2003). Each machine is supposed to have an owner, who can specify the conditions under which jobs are allowed to run; by default, a Condor job stops when a workstation's owner begins using the computer. Hence, Condor jobs use processor cycles that otherwise would be wasted. Compared to other grid computing software, Condor is easy to install and to administrate, and existing programs do not need to be modified or re-compiled to be executed under Condor (they must only be re-linked with the Condor library).

Salient features of Condor include remote system calls, job checkpointing and process migration. Furthermore, Condor pools can be composed of heterogeneous machines, and several pools can be combined using Globus (Foster and Kesselman 1997) and Condor-G (Frey *et al.* 2001).

4.2. *The MW library*

GrEA has been implemented using MW (Linderoth *et al.* 2000), a software library that enables the development of master worker parallel applications using Condor and the C ++ Programming Language.

An MW application consists mainly of subclassing three base classes: MWTask, MWDriver and MWWorker. An MWTask represents the unit of work to be computed by a worker. It includes the inputs and outputs to be marshalled to and from the workers. In this implementation, the input is an integer array (the permutation representing an individual) and the output is a real value containing the fitness value plus another integer array (because the individual can be modified by an improvement method). The MWWorker provides the context for the task to run; in concrete, in GrEA the subclass of MWWorker contains the GrEA-worker code. Finally, the MWDriver subclass manages the whole process: creating tasks, receiving the results of the computations, and deciding when the computation is complete. The GrEA-master code is executed in this subclass.

The MW framework works with Condor to find computing resources for the available tasks, handle communication between the nodes, re-assign tasks if their current machine fails, and globally manage all the parallel computations. That is, MW generates tasks whose computation is subsequently managed by Condor. MW provides hooks to save the state of the driver, so that if the driver, or its machine, crashes, the computation can make progress upon driver restart.

MW can run with one of several RMComm (Resource Management and Communication) implementations. This layer implements communication between the master and the workers, and the management of the worker machines. There are several choices, including communicating via PVM, sockets and shared files. We have chosen the last option because it is the most robust; for example, if the driver (master) crashes, the workers can continue their computation, which is not possible using PVM and sockets. Although process communication using files is rather slow, this application is not intensive in data exchanges, and the communication costs can be acceptable if the computation time is long enough.

4.3. *Grid platform details*

In the experiments the computers of seven laboratories of the Computer Science Department of the University of Málaga have been used. Many of them are equipped with PCs having modern Intel Core 2 Duo, 3 GHz processors. This means that each processor has two cores, so Condor assumes that there are two processors per computer. For the sake of clarity, the term *processor* will be used throughout the article, although the correct term is *core*. The Condor pool also includes slower single-core machines with Pentium IV at 1.66 GHz and AMD Athlon at 1.1 and 2.0 GHz. Up to 300 processors have been used, being all interconnected through a 100 Mbps Fast Ethernet network. The code is written in C ++, and all the machines run different flavours of Linux (Fedora Core, Debian, SuSE, etc.).

5. Results

This section presents the experiments conducted to evaluate GrEA. First, some details of the algorithm settings are given and, secondly, the GSM network instance used is described. Finally, the experiments are presented from the point of view of both parallelism (the parallel efficiency, the workers used, and the tasks computed per minute are analysed in detail) and numerical accuracy.

5.1. *Experimental setup*

The results of running the sequential counterpart of GrEA have been included here. It is a standard ssGA that uses the same operators (selection, crossover, mutation and local search). The aim is to test the performance of GrEA in terms of both the parallelism of the grid-based approach and its numerical accuracy. The parameter settings of these two GAs are detailed in Table 1.

Because of the stochastic nature of GAs, 30 independent runs of the GrEA have been done, but only ten runs of the ssGA were performed owing to time constraints (each execution lasts more than five days on a Pentium IV at 2.8 GHz). Working with real problems and grids is very hard, and thus a small number of independent runs is usually reported (Melab *et al.* 2006, Kuś 2007, Lim *et al.* 2007). The same is done here with ssGA, but reporting on 30 independent runs for GrEA, a value considered as the minimum for statistical significance.

Table 1. Parameter settings of the GAs.

Parameter	Value
Population size	100 individuals
Representation	Integer (array of size 2612)
Crossover operator	Uniform crossover ($p_c = 0.5$)
Mutation operator	Random ($p_m = 0.01$)
Local search steps	6
Selection method	Binary tournament
Replacement strategy	Worst individual
Stopping condition	50,000 iterations

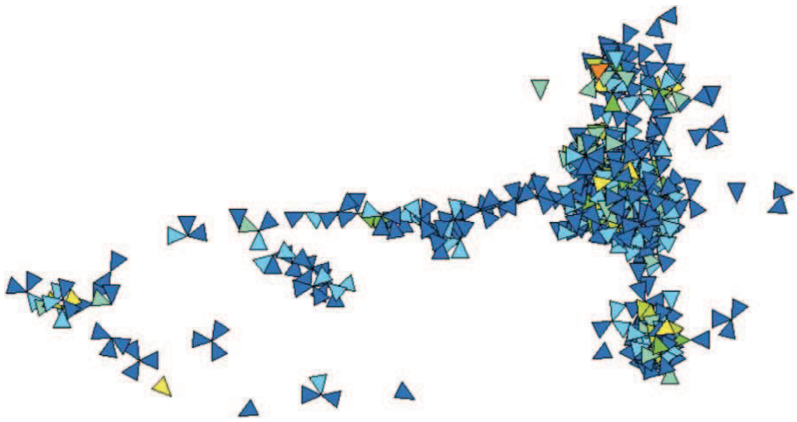

Figure 2. Topology of the GSM network used. Available in colour online.

5.2. *GSM instance used*

The GSM network used here has 711 sectors with 2612 TRXs to be assigned a frequency; the constants in Equations (2), (3) and (6) were set to $K = 100{,}000$, $c_{\text{SH}} = 6$ dB and $c_{\text{ACR}} = 18$ dB, respectively. Each TRX has 18 available channels (from 134 to 151). Figure 2 displays the network topology, every triangle representing a sectorized antenna in which several TRXs operate.

This GSM network is currently operating in Denver (CO, USA), a 400 km^2 city with more than 500,000 people, so its solution is of great practical interest. The data source to build the interference matrix based on the C/I probability distribution uses thousands of Mobile Measurement Reports (MMRs) (Kuurne 2002), rather than propagation prediction models. MMRs are a more accurate data source, as they capture the call location pattern in the network and do not rely on predictions. These properties make this GSM problem more realistic than standard available benchmarks (FAP Web 2008). Indeed, the most similar available instances are the COST 259 benchmark, but the basic traffic load is drawn at random according to an empirically observed distribution, and signals are predicted with several propagation models. The Philadelphia, CELAR and GRAPH instances (FAP Web 2008) are even simpler.

5.3. *Parallel performance*

In this section the performance of GrEA is analysed when solving the AFP network considered. Table 2 summarizes the best value, the mean $\bar{x}$, and the standard deviation σ_n, of several performance measures.

Table 2. Performance measures for GrEA.

Measure	Best value	$\bar{x}$	σ_n
Max number of workers	282	235	24
Average number of active workers	164	140	10
Total CPU time (s)	659,149 (7.62 days)	680,274 (7.87 days)	8476
Wall clock time (s)	4094 (1.14 hours)	4879 (1.36 hours)	429
MW parallel performance	73.44%	71.79%	0.72%
Sequential ssGA time (s)	463,014 (3.36 days)	473,259 (5.48 days)	12 321
Average parallel efficiency		41.28%	

Although the Condor pool is composed of 300 processors at most, a maximum number of 282 have been used, and 235 on average. This is typical behaviour of grid computing systems, where the number of available processors is dynamic, and they have to be shared among different users.

If the number of active workers is considered, it is around 140 on average. This is a consequence of the behaviour of GrEA under Condor/MW. Whenever an MW application has tasks to be computed, it asks the Condor system for available workers. The way MW asks for more workers to Condor is by requesting a predefined number of them. This fixed number is configurable and it has been set up to be 100 workers. This value corresponds to the set of initial tasks created to evaluate the initial population, thus guaranteeing that there will be in the beginning one worker for each task. This effect can be observed in Figure 3. It can be seen that, at the beginning of the computation, the number of worker tasks increases up to 100; after a few minutes, another 100 processors are requested to the grid system. Finally, the algorithm obtains the rest of the available processors. Although the total number of processors is roughly constant (about 220 in the execution depicted in Figure 3), the average number of them is affected by the initial stage of the algorithm.

An advantage of using grid computing systems is to solve, in a reasonable amount of time, problems that otherwise would be considered unfeasible. In Table 2 it can be observed that the average total CPU time reported by Condor (*i.e.* the sum of the computing time of all the

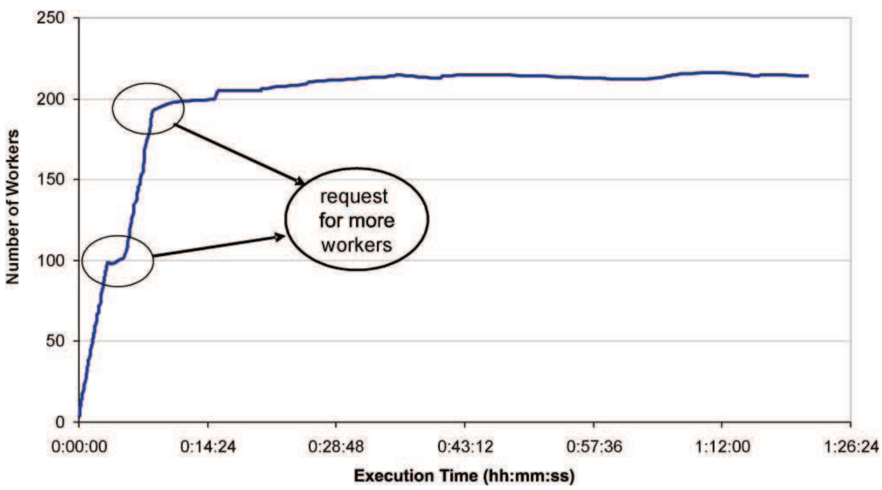

Figure 3. Number of workers during a typical GrEA execution.

processors) is near to 8 days, while the wall clock time is 1.36 hours (more than 139 times faster). These values clearly state the benefits of using the proposed approach.

The parallel performance reported by Condor is about 72% (a parallel efficiency of 0.72), which can be considered as an excellent result in a grid platform. However, the computation/communication could be improved by using a more computing intensive local search. Additional experiments have been done with 10 and 20 local search steps ($d = 10$ and $d = 20$), and the parallel performance grew up to 81% and 90%, respectively. The point here is that the local search strategy is so accurate that it reaches a local optimum after five or six iterations, and therefore the numerical results are similar to the ones reported in this work. It is clear then that there is room for increasing the efficiency of GrEA.

The last two rows of Table 2 include the computing times of the sequential ssGA when solving the AFP problem considered (about 5.4 days), as well as the parallel efficiency against the average wall clock time required by GrEA. When comparing both ssGA and GrEA, the efficiency goes down to 41.28%. The explanation is twofold. On the one hand, the ssGA has been compiled with several optimization options to speedup the computation as much as possible, and some of them are related to the type of processor of the machine where the program runs; these options are disabled in GrEA owing to the heterogeneity of the processors in the grid. On the other hand, there are processors in the grid platform that are much slower (see Section 4.3) than the one used to run ssGA, a Pentium IV at 2.8 GHz, so therefore workers spend longer times to compute the same tasks thus delaying the entire computation. However, in practical terms, ssGA requires more than five days to solve the problem while the grid only needs an hour and a half, so the benefits of this approach still hold.

The throughput (tasks computed per minute) of GrEA when solving the AFP problem is analysed next. In Figure 4 it can be observed that this number remains almost constant at around 630 tasks per minute. This contrasts with the behaviour of the algorithm reported in Nebro *et al.* (2008), where the throughput dropped in time owing to the computation time of the tasks decreasing when the search progressed (from 30 seconds at the beginning down to two seconds at 60% of the computation). This issue does not appear when evaluating the AFP, so it is not necessary to adjust the computation grain dynamically as in Nebro *et al.* (2008).

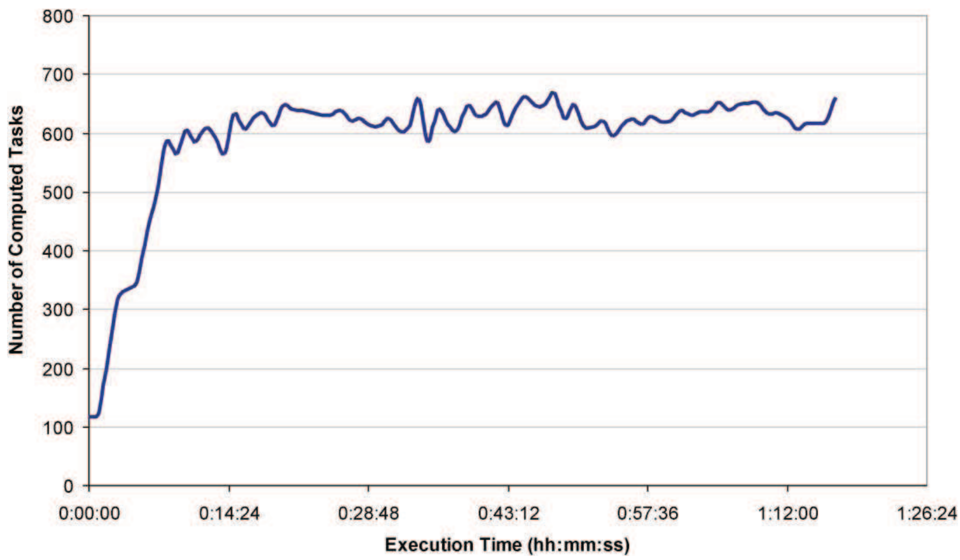

Figure 4. Tasks per minute computed by the GrEA.

Table 3. Numerical efficiency of the algorithms.

	AFP Cost		
Algorithm	Best (min)	$\bar{x}$	σ_n
ACO	88,345.95	90,382.56	935.31
ssGA	85,463.20	86,234.68	523.75
GrEA	83,991.30	84,936.32	375.89

5.4. *Accuracy*

In this section, the numerical efficiency of GrEA is compared to both ssGA and the algorithm for which the best results so far for this problem have been reported, the ACO presented in Luna *et al.* (2007) (its parametrization is detailed in the given reference). Even though the GAs and the ACO algorithm are not directly comparable (the ACO execution time was limited to 30 minutes because other measures were used), the previous results on the problem cannot be ignored and they are included both for completeness and for showing the competitivity of these proposals.

Table 3 includes the best (lowest), the mean $\bar{x}$, and the standard deviation σ_n of the resulting AFP costs reached by the three algorithms. As stated before, 30 independent runs for GrEA and 10 for ssGA have been run. For the ACO algorithm, the results reported in Luna *et al.* (2007) have been used. An accurate statistical analysis has been performed here in order to compare the algorithms numerically with confidence (Sheskin 2003, Demšar 2006). First, a Kolmogorov–Smirnov test is performed in order to check whether the values of the results follow a normal (Gaussian) distribution or not. If the distribution is normal, the Levene test checks for the homogeneity of the variances. If samples have equal variance (positive Levene test), an ANOVA test is done; otherwise a Welch test is performed. For non-Gaussian distributions, the non-parametric Kruskal–Wallis test is used to compare the medians of the algorithms. Figure 5 summarizes the statistical analysis. A confidence level of 95% is always considered (*i.e.* significance level of 5% or p-value under 0.05) in the statistical tests, which means that the differences are unlikely to have occurred by chance with a probability of 95%. The result tables show $\bar{x}$ and σ_n because all the samples follow a Gaussian distribution.

To further analyse the results statistically, a post-hoc testing phase has been included in Table 4 which allows for a multiple comparison of the samples. The `multcompare` function provided by Matlab © has been used. This function chooses the most appropriate type of critical value to be used in the multiple comparison, which ranges from the more conservative *HSD* or *Tukey–Kramer* method to the less conservative *Scheffe's S* procedure (Hochberg and Tamhance 1987). The same confidence level has been kept for this testing phase ($\alpha = 0.05$). The '+' symbols in the table point out that all pairwise comparisons among the algorithms are significant.

The first conclusion that can be drawn from the results is that the two GAs outperform the ACO algorithm. As stated before, this is somehow expected because the computational effort used in

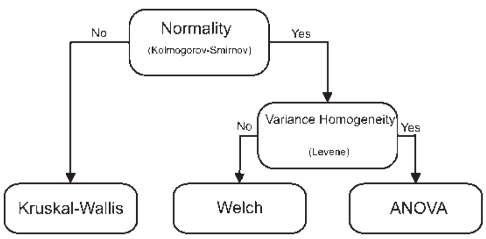

Figure 5. Statistical analysis performed in this work.

Table 4. Post-hoc tests of the results.

ssGA	+	
GrEA	+	+
	ACO	ssGA

the GAs is larger than the ACO one. The improvements are noticeable in both the best solution (88,345.95 down to 83,991.30) and the average solution (90,382.56 down to 84,936.32). If σ_n is considered as a measure of robustness, GrEA and ssGA also outperform the ACO approach.

But the really interesting fact that Table 3 shows is that GrEA has computed more accurate frequency plans (lower AFP costs) than its sequential counterpart, ssGA, and with statistical confidence, as the '+' symbols of Table 4 show (indeed, this is the only actual goal of companies). The relevance of these results arises because these two algorithms share the same operators and parameter settings, and they also use the same computational effort. The only difference is the asynchronism introduced in GrEA, in which individuals evaluated by slower processors are returned back when several iterations (with individuals sent to faster processors) have proceeded. This means a higher diversity in the search is introduced, especially in the early steps, which later

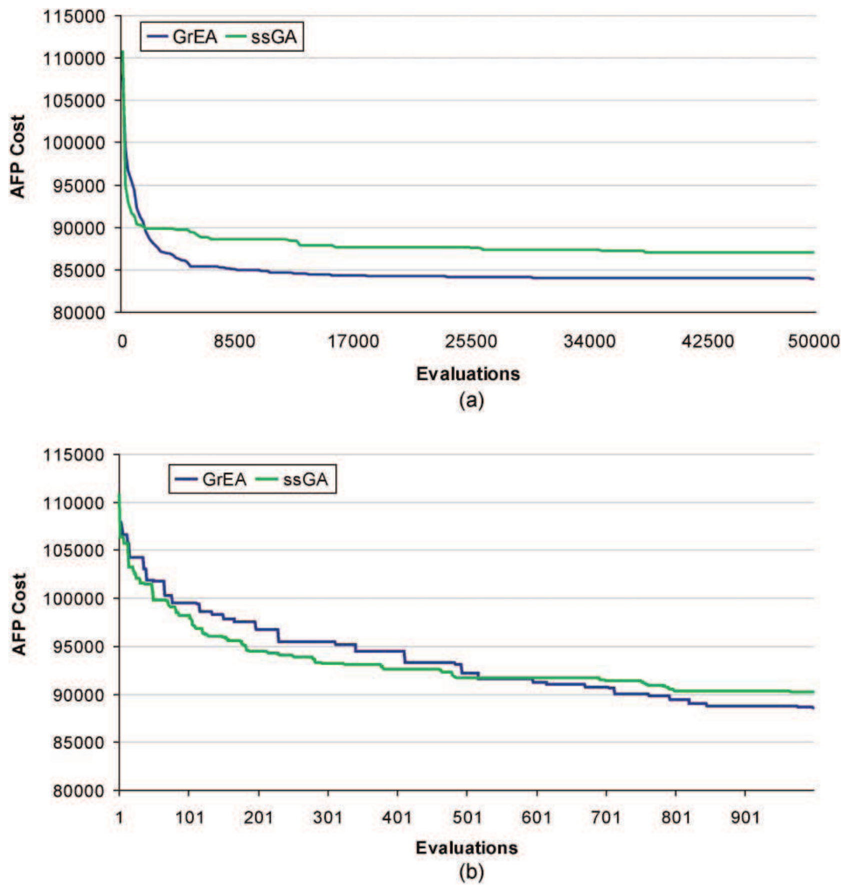

Figure 6. Evolution of the AFP cost in GrEA and ssGA during (a) the whole execution and (b) the first 1000 iterations.

guides the algorithm towards high-quality solutions (Alba and Troya 2001). Figure 6 graphically displays this fact by including the best AFP cost of a typical run of GrEA and ssGA. In the left-hand side of the figure, the evolution of the fitness during the entire computation (50,000 iterations) is shown. It can be seen that GrEA gets stuck later than ssGA and from then they both improve the solution slightly. In Figure 6(b), the first 1000 generations of the same execution have been enlarged in order to show the higher diversity in GrEA at the beginning of the computation. Indeed, ssGA outperform GrEA in the first 500 iterations, but it finds difficulties to continue after that (see the flat regions during iterations 500 and 700). In the meanwhile, GrEA keeps improving the solution. Within the context of this work, it can be concluded that deploying ssGA on the grid is not only a way of reducing the computational time, but also of improving the underlying search models, which allows the algorithm to compute more accurate solutions.

6. Conclusions and future work

This work addresses a real-world instance of the AFP problem. The instance considered has a huge search space, so the approach has been to combine the use of metaheuristics (numerically powerful) and grid computing (computationally powerful). In particular, the GrEA algorithm has been used. It is a grid-enabled GA that runs on a grid platform composed of up to 300 processors.

The problem has been first analysed from the point of view of its solution by a sequential GA; thus, issues such as the problem representation, genetic operators (selection, mutation, crossover) and local search strategy, have been addressed. A very accurate formulation has been used, which is rarely found in the literature of this problem. Secondly, the GrEA algorithm has been applied to solve a complex AFP instance corresponding to an actual city in the USA (Denver, CO).

The experiments carried out reveal that ssGA requires about five days of computation in a modern PC, while executing GrEA in the grid reduces the time to one hour and a half. The overall parallel efficiency obtained is around 72%, which is a quite satisfactory value considering the number of processors used and that shared files in Condor are being used as the message passing mechanism. This value also suggests that there is more room for improvement; for example, more steps in the local search could bring a better computation/communication ratio.

While the reduction of the computing time from several days to less than two hours is an interesting result in practical terms, it is also remarkable that the search capability of the GrEA algorithm outperforms that of the equivalent sequential GA. The fact that better (lower) fitness values can be obtained in the parallel algorithm would allow us to reduce the computing time even more in order to have high-quality solutions in shorter times; this can be useful in real scenarios, where telecommunication companies need to perform many simulations to achieve a satisfactory frequency plan for large cities (like Los Angeles, with more than 40,000 TRXs).

As future work, we are interested to use GrEA to solve even larger instances of the AFP (*e.g.* the aforementioned Los Angeles instance). The use of this algorithm to solve other problems from the telecommunication domain, such as ACP (Automatic Cell Planning), is a matter of future research.

Acknowledgements

This work has been partially funded by the Spanish Ministry of Education and Science and FEDER under contract TIN2005-08818-C04-01 (the OPLINK project). Juan J. Durillo is supported by grant AP-2006-03349 from the Spanish Ministry of Education and Science.

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.

Alabau, M., Idoumghar, L. and 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. and Troya, J., 1999. A survey of parallel distributed genetic algorithms. *Complexity*, 4 (4), 31–52.

Alba, E. and Troya, J.M., 2001. Analyzing synchronous and asynchronous parallel distributed genetic algorithms. *Future Generation Computer Systems*, 17 (4), 451–465.

Berman, F., Fox, G. and Hey, A., 2003. Grid computing. Making the global infrastructure a reality. *In*: *Communications networking and distributed systems*. New York: Wiley.

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

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

Crisan, C. and Mühlenbein, H., 1998. The breeder genetic algorithm for frequency assignment. *In*: *Proceedings of the 5th International Conference on Parallel Problem Solving From Nature*. Lecture notes in computer science Vol. 1498. London: Springer-Verlag, 897–906.

Crompton, W., Hurley, S. and Stephens, N.M., 1993. *Frequency assignment using a parallel genetic algorithm*. *In*: *Proceedings of the 2nd IEE/IEEE Workshop on Natural Algorithms in Signal Processing (NASP'93)*, IEEE Press, 26/1–26/8.

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 Robotices and Neural Networks*, IEEE Press, 81–84.

Demšar, J., 2006. Statistical comparison of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7, 1–30.

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

Eisenblätter, A., 2001. Frequency assignment in GSM networks: Models, heuristics, and lower bounds. Thesis (PhD). Technische Universität Berlin.

FAP Web [online]. http://fap.zib.de/ [Accessed 1 March 2008].

Foster, I. and Kesselman, K., 1997. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 11 (2), 115–128.

Foster, I. and Kesselman, C., 1999. *The grid: Blueprint for a new computing infrastructure*. San Francisco: Morgan-Kaufmann.

Frey, J., Tannenbaum, T., Foster, I., Livny, M. and Tuecke, S., 2001. *Condor-G: A computation management agent for multi-institutional grids*. *In*: *Proceedings of the 10th IEEE Symposium on High Performance Distributed Computing (HPDC)*, IEEE Press, 7–9.

Graham, J.S., Montemani, R., Moon, J.N.J. and Smith, D.H., 2008. Frequency assignment, multiple interference and binary constraints. *Wireless Networks*, 14 (4), 449–464.

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

Hochberg, Y. and Tamhane, A.C., 1987. *Multiple comparison procedures*. New York: Wiley.

Idoumghar, L. and Schott, R., 2006. *A new hybrid GA-MDP algorithm for the frequency assignment problem*. *In*: *Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '06)*, 18–25.

Kim, S.S., Smith, A.E. and Lee, J.H., 2007. A memetic algorithm for channel assignment in wireless FDMA systems. *Computers & Operations Research*, 34 (6), 1842–1856.

Kuś, W., 2007. Grid-enabled evolutionary algorithm application in the mechanical optimization problems. *Engineering Applications of Artificial Intelligence*, 20 (5), 629–636.

Kuurne, A.M.J., 2002. *On GSM mobile measurement based interference matrix generation*. *In*: *Proceedings of the IEEE 55th Vehicular Technology Conference, VTC Spring 2002*, IEEE Press, 1965–1969.

Kwok, Y.K., 2000. *Quasi-static dynamic channel assignment using a Linux PC cluster*. *In*: *Proceedings of the 4th International Conference on High Performance Computing in the Asia-Pacific Region*, IEEE Press, 170–175.

Lim, D., Ong, Y.S., Jin, Y., Sendhoff, B. and Lee, B.S., 2007. Efficient hierarchical parallel genetic algorithms using grid computing. *Future Generation Computer Systems*, 23 (4), 658–670.

Linderoth, J., Kulkarni, S., Goux, J.P. and Yoder, M., 2000. *An enabling framework for master–worker applications on the computational grid*. *In*: *Proceedings of the 9th IEEE Symposium on High Performance Distributed Computing (HPDC)*, IEEE Press, 43–50.

Luna, F., Blum, C., Alba, E. and Nebro, A.J., 2007. *ACO vs EAs for solving a real-world frequency assignment problem in GSM networks*. *In*: *Genetic and Evolutionary Computation Conference (GECCO 2007)*, ACM Press, 94–101.

Matsui, S., Watanabe, I. and Tokoro, K.I., 2003. *An efficient hybrid genetic algorithm for a fixed channel assignment problem with limited bandwidth*. Lecture notes in computer science Vol. 2724. London: Springer-Verlag, 2240–2251.

Matsui, S., Watanabe, I. and Tokoro, K.I., 2005. Application of the parameter-free genetic algorithm to the fixed channel assignment problem. *Systems and Computers in Japan*, 36 (4), 71–81.

Melab, N., Cahon, S. and Talbi, E.G., 2006. Grid computing for parallel bioinspired algorithms. *Journal of Parallel and Distributed Computing*, 66 (8), 1052–1061.

Mishra, A.R., 2004. Radio network planning and optimisation. *In*: *Fundamentals of cellular network planning and optimisation: 2G/2.5G/3G . . . evolution to 4G*. New York: Wiley, 21–54.

Mouly, M. and Paulet, M.B., 1992. *The GSM system for mobile communications*. Palaiseau, France: Cell & Sys.

Nebro, A.J., Luque, G., Luna, F. and Alba, E., 2008. DNA Fragment assembly using a grid based genetic algorithm. *Computers & Operations Research*, 35 (9), 2776–2790.

Press, W.H., Flannery, B.P., Teukolsky, S.A. and Vetterling, W.T., 1992. *Numerical recipes in C: the art of scientific computing*. Cambridge, UK: Cambridge University Press.

Sheskin, D.J., 2003. *Handbook of parametric and nonparametric statistical procedures*. Boca Raton, FL: CRC Press.

Simon, M.K. and Alouini, M.S., 2005. *Digital communication over fading channels: a unified approach to performance analysis*. New York: Wiley.

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

Thain, D., Tannenbaum, T. and Livny, M., 2003. Condor and the grid. *In:* F. Berman, G. Fox and T. Hey, eds. *Grid computing: making the global infrastructure a reality*. New York: Wiley, 299–335.

Walke, B.H., 2002. *Mobile radio networks: Networking, protocols and traffic performance*. New York: Wiley.

Weinberg, B., Bachelet, V. and Talbi, E.G., 2001. *Co-evolutionist meta-heuristic for the assignment of the frequencies in cellular networks*. Lecture notes in computer science Vol. 2037. *In*: Proceedings of the EvoWorkshops on Applications of Evolutionary Computing, London: Springer-Verlag, 140–149.