Omni-directional RND Optimisation using Differential Evolution: In-depth Analysis via High Throughput Computing

Sílvio Mendes[1,2], Patricio Domingues[1,3], David Pereira[1], Renato Vale[1], Juan A. Gomez-Pulido[2], Luis Moura Silva[3], Miguel A. Vega-Rodríguez[2] and Juan M. Sánchez-Pérez[2]

[1] School of Technology and Management - Polytechnic Institute of Leiria,
2411-903 Leiria, Portugal
[2] Polytechnic School -University of Extremadura
10071 Cáceres, Spain
[3] CISUC, Department of Informatic Engineering, University of Coimbra
3030 Coimbra, Portugal

{smendes, patricio}@estg.ipleiria.pt, {ei12407,ei12405}@student.estg.ipleiria.pt, jangomez@unex.es, luis@dei.uc.pt, {mavega,sanperez}@unex.es

Abstract. The Radio Network Design (RND) constitutes an important class of problems, particularly in the planning of wireless communication networks. RND problems are challenging to tackle since they fall in the NP-hard class of optimisation problems. In this paper, we assess the viability of adapting the Differential Evolution (DE) algorithm to a wide-scale real world RND problem. To fulfil the high computational demands of the DE approach, we resort to a pool of more than 150 non-dedicated machines, whose CPU cycles are scavenged through a high throughput system. Our results show that DE is a viable approach for RND problems if proper computing power is available.

Keywords: RND, Differential Evolution, high throughput computing, mathematical techniques in telecommunications, Bio-inspired optimisation

1 Introduction

With the rapid growth of communication infrastructures, the radio network design (RND) problem has become a major issue for the efficient placement of radio transmitters, in such a way that coverage area is maximized while the number of antennas is minimized. However, since RND is a NP-Hard problem [1], exhaustive search of the solution space is computationally not tractable, and thus alternative approaches are sought. In this paper, we apply the Differential Evolution (DE) methodology to tackle the RND problem, combining it with the Condor high throughput platform [2] to perform a directed, yet computationally heavy search using non-dedicated computing resources. Previous partial experiments had been conducted [3][4], but this work scales up to a real world sized problem, in order to evaluate the effectiveness of DE on such extent using a discretized representation. Until now, DE had proven to converge with real-valued continuous functions. High throughput computing (HTC) pierces the time required to obtain scientific evidence, in which we could actually consider the RND optimisation itself not less important, but also as a real world scientific research sidekick case study.

This paper is organized as follows. In section 2, we successively introduce the RND problem, the differential evolution methodology, and how we apply DE to RND. In section 3, we briefly review the Condor high throughput computing system, while our main results are presented in section 4. Section 5 discusses related work, while section 6 concludes the paper and presents venues for future work.

2 RND and Differential Evolution

2.1 RND

The RND problem consists in minimizing the number and locations of transmission antennae to cover a maximum area in order to give service to the highest possible amount of terminal paraphernalia. Hence we can consider it as being an intrinsically multi-objective problem. Specifically, a base station transmitter (BS or BTS) is a radio signal transmitting device, with a determined type of coverage. In this work we disregard different types of wave propagation models, since any complex model can be applied.

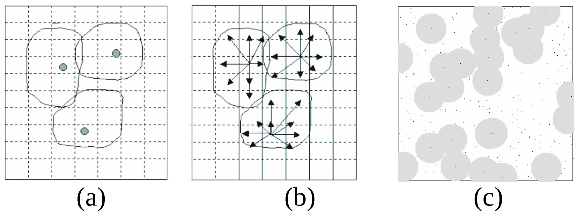


Fig. 1. The terrain is digitalized by means of a grid. (b) Three BS locations are shown, together with the locations under their influence.

We consider a digitalized model of the terrain. The area is divided in sectors or locations (atomic bits of terrain). Each coordinate (x,y) of the grid represents a possible BS transmitter location. Fig. 1 represents this model, showing how three BS transmitters are located in available coordinates of the grid (Fig. 1a), and how some locations are under the influence of these BS with different coverage degrees (Fig. 1b). Fig. 1c depicts several Base Station transmitters with a simplified omni-directional wave propagation model which is used in this and previous work [3].

It is important to note, that if two or more BS transmitters are close to each other, their cover areas are overlapped, and thus the locations inside these areas might have different degrees of coverage (for example, one location can be under the influence of two BS transmitters, while another location can be inside the cover area of only one transmitter; in this case, the second location ends up with a lower level of intensity of the received signal).

In order to mathematically define this problem [5], let us consider the set L of all potentially covered locations and the set M of all potential transmitter locations. Let G be the graph, (M $\cup$ L, E), where E is a set of edges such that each transmitter location is linked to the locations it covers. As the geographical area needs to be discretized, the potentially covered locations are taken from a grid, as previously shown in Figure 1a. In our case, we are focusing (in average) a $\binom{1000}{50}$ echelon problem. The vector **x** will be a solution to the problem where $x_i \in \{0,1\}$, and $i \in [1, 1000]$, indicates whether a transmitter is used (1) or not (0) in the corresponding location. Other complementary information can be modelled at this stage, in order to add further restrictions on RND specific domain problem modelling.

Since the objective of RND is to find the minimum subset [6] of transmitters that covers a maximum surface of an area, we are looking for a subset M' $\subseteq$ M such that |M'| is minimum and such that |Neighbours(M', E)| is maximum, where:

$$\text{Neighbours}(M', E) = \{u \in L \mid \exists v \in M', (u, v) \in E\} \qquad (1)$$

$$M' = \{t \in M \mid x_t = 1\}$$

The main constraint of this problem relates to the set of available locations for the antennas, since there are some places where the antennas can not be placed (public recreation areas, some roofs, water or mountain zones, etc.).

We also need to establish a fitness function (f) to evaluate the efficiency of a BTS set disposed in a determined way in the grid. In our case, f (Eq. 2) depends on the square of the cover rate and on the number of BS transmitters [5][6].

$$f(x) = \frac{CoverRate(x)^2}{NumberTransmittersUsed(x)} \qquad (2)$$

Where

$$CoverRate(x) = 100\,\frac{Neighbours(M', E')}{Neighbours(M, E)} \qquad (3)$$

A textual description for the *modus operandi* of the previously referred fundamentals is presented next. Primarily, we need to build a BS network giving the maximum coverage to an area. So, in cell planning design, we have to determine the set of available locations for the BS. Then, the goal is to obtain high percentage coverage for the considered area, using the lowest amount possible of BTS. This must be done in a resourceful manner. The set of available locations can be modelled through an array of coordinates related to the grid. In this case, the size of the array would be the size of the problem instance (or solution). Every instance problem should be evaluated by the fitness function in order to differentiate a better solution from a worst one.

The problem we consider recalls the *Unicost Set Covering Problem* (USCP), which is a known NP-hard combinatorial optimisation problem [1]. However, the RND problem differs from the USCP in the fact that the target is to select a subset of BTS that ensures a good coverage of a given area, and not to ensure a total coverage.

Due to its NP-hardness classification, we wanted to exhaustively assess a bio-inspired algorithm, because previous implementations with genetic algorithms (in sequential and parallel implementations) and other evolutionary techniques have been employed with success in some other works [5][7][8].

2.2 Differential Evolution

Differential Evolution (DE) is an algorithm created by Ken Price and Rainer Storn [9]. Since 1994, DE has been used for many optimisation problems with satisfactory results [10][11][12]. This is the main motivation to use it in our research, with the future decisive purpose of comparing it with other works [13]. The DE base code is fully available to researchers [14].

DE is a very simple population-based stochastic function minimizer, which can be categorized into a class of *floating-point encoded, evolutionary algorithms*. It is currently used in a wide range of optimisation problems, including multi-objective optimisation [15]. Generally, the function to be optimised, *F*, is of the form:

$$f(X): R^n \rightarrow R \tag{4}$$

The optimisation target is to minimize the value of the objective function f (*X*),

$$\min(f(X)) \tag{5}$$

by optimising the values of its parameters:

$$X = (x_1, \ldots, x_{n_{param}}), x \in R \tag{6}$$

where *X* denotes a vector composed of n_{param} objective function parameters. Usually, the parameters of the objective function are also subject to lower and upper boundary constraints, $x^{(L)}$ and $x^{(U)}$, respectively:

$$x_j^{(L)} \leq x_j \leq x_j^{(U)}, j = 1, \ldots, n_{param} \tag{7}$$

As with all population-based evolutionary optimisation algorithms, DE handles a population of solutions, instead of a single solution for the optimisation of a domain dependant problem. Population *P* of generation *G* contains n_{param} solution vectors, each one usually known as an individual of the population. Consequently, each vector represents a potential solution for the optimisation problem.

$$P^{(G)} = X_i^{(G)}, i = 1, \ldots n_{pop}, G = 1, \ldots, G_{\max} \tag{8}$$

So, the population *P* of generation *G* contains n_{pop} individuals, each one containing n_{param} parameters (usually referred as *chromosomes*):

In order to establish a starting point for optimum seeking, the population $P^{(0)}$ (initial population) must be initialized. This is usually done by seeding $P^{(0)}$ with random values that are within the given boundary constraints:

$$P^{(0)} = x_{i,j}^{(0)} = r_{i,j}(x_j^{(U)} - x_j^{(L)}) + x_j^L \qquad \textbf{(9)}$$

Where

$$i = 1,\dots n_{pop},\ j = 1,\dots,n_{param}$$

where *r* denotes a uniformly distributed random value within range [0.0, 1.0].

The population reproduction scheme of DE is different from other evolutionary algorithms. From the 1st generation forward, the population of the following generation $P^{(G+1)}$ is created in the following way on basis of the current population $P^{(G)}$. First, a temporary individual (usually referred as trial) that can possibly populate the subsequent generation, $P^{'(G+1)}$, is generated as shown in Equation 10.

$$x_{i,j}^{'(G+1)} = \begin{cases} x_{C_i,j}^{(G)} + F \cdot (x_{A_i,j}^{(G)} - x_{B_i,}^{(G} & \text{if} \quad r_{i,j} \le Cr \\ x_{C_i,j}^{(G)} & \end{cases} \qquad \textbf{(10)}$$

Where

$$i = 1,\dots n_{pop},\ j = 1,\dots,n_{param}$$
$$A = 1,\dots n_{pop}, B = 1,\dots,n_{pop}, C = 1,\dots,n_{pop}, A_i \ne B_i \ne C_i \ne i$$
$$Cr \in [0,1], F \in [0,2], r \in [0,1[$$

A, B and C are three randomly chosen indexes referring three individuals of the population. Fig. 2 schematically illustrates how the reproduction of a Nth generation population yields the next generation (N+1).

F, *Cr* and n_{pop} are DE control parameters that remain constant during the search process. n_{pop} represents the population size, *F* is a real-valued factor in range [0.0, 2.0] that controls the amplification of differential variations, and *Cr* is a real-valued crossover factor in range [0.0,1.0] controlling the probability to choose the mutated value for *x* instead of its current value. According to Lampinen et al. [10], both *F* and *Cr* affect the convergence velocity and robustness of the search process. Their optimal values are dependent on the individuality of the objective function f (*X*), and on the population size n_{pop}. Normally, suitable values for *F*, *Cr* and n_{pop} can be found by trial-and-error after some experiments using different values. Practical advice on how to select control parameters n_{pop}, *F* and *Cr* are given in [9][11][12][13].

Usually, higher values of n_{pop} and *F* result in a more robust search but at the expense of a lower convergence velocity. Conversely, excessively small values for n_{pop} and for *F* may cause premature convergence and/or stagnation before finding the global optima. Typically n_{pop} varies from $2{*}n_{param}$ to $20{*}n_{param}$. Small population size is recommended for unimodal and moderately multimodal problems. However, highly multimodal problems may require n_{pop} higher than $50{*}n_{param}$. According to practical experiences, *F* lower than $2/n_{pop}$ or higher than 1.2 are unlikely to work best. Hence, *F* within range $[(2/n_{pop}), 1.2]$ should be used as initial estimation.

Generally a high value of *Cr*, ranging from 0.9 to 1.0 can be recommended as a starting point unless the objective function is *a priori* known to be separable. Small values of *Cr* may result in higher convergence speed, but only in cases where separable functions or functions with a low degree of *epistasis* are used (masking effect of a gene action by another one), or if low variable dependency exists [18][19]. With *Cr*=1.0 the algorithm is rotationally invariant, an important property for solving real life problems, since these can rarely be represented by separable functions, and in fact, are more often subject to non-linear interactions between the variables. The generational scheme of DE also differs from other evolutionary algorithms. Based on the current population $P^{(G)}$ and on the trial vector $X^{(G)}$, the population of the next generation $P^{(G+1)}$ is created as follows:

$$X_i^{(G+1)} = \begin{cases} X_i^{'(G+1)} & \text{If } f_{\cos t}(X_i^{'(G+1)}) \leq f_{\cos t}(X_i^{(G)}) \\ X_i^{(G)} & \text{Otherwise} \end{cases} \quad \textbf{(11)}$$

Accordingly, each computed trial vector (generally known as a *donor* vector) is compared with the target vector. The one with the lower value of cost function $f_{\cos t}(X)$ will remain in the population of the next generation.

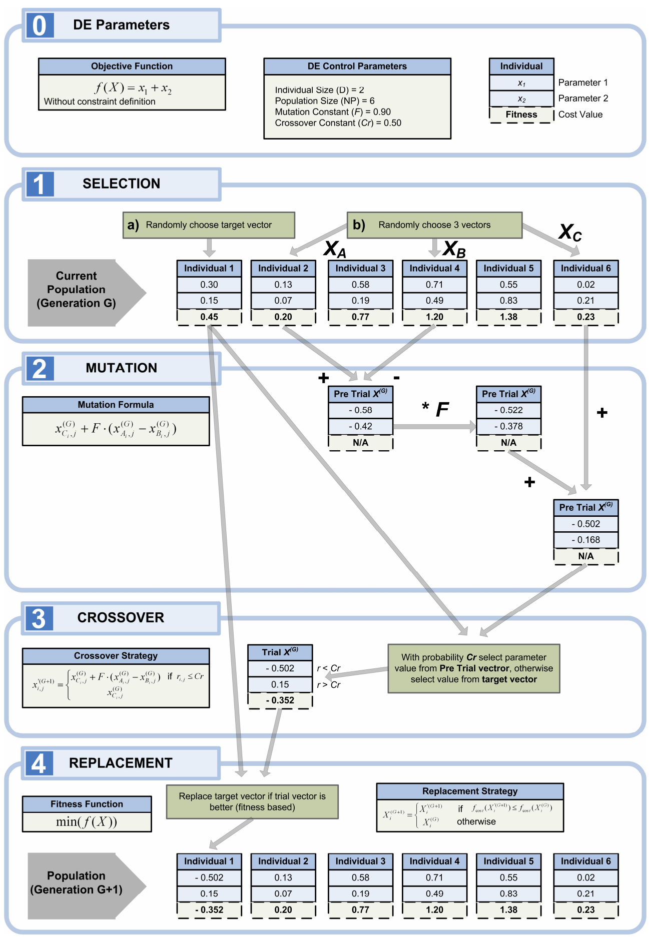
2.3 RND Differential Evolution-based Model

Our generational algorithmic model retains the main characteristics from the original DE, namely the *panmictic* steady-state iterative structure and the donor vector that schemes the generational iterations.

Previous works [19] have stated that usage of general steady-state algorithms overwhelm other generational models. The removal of the donor vector element has not been considered due to the intrinsic nature of the mathematical operations in the variation operators. These operations are known as *differential mutation* operations in DE context.

In our approach, we kept the iterative element (see blocks 1-4 of Fig. 2), where the successive generations try to get an optimal solution, ending when the stopping criterion is met, i.e., when a number of generations is reached or when the fitness of the current solution is better than a predetermined value. Additionally, all design issues took into account the differential evolution fast convergence, proven in previous works [16], and its canonical self adapting differential mutation operator.

The **Nearest Point Differential Mutation Operator (NPDM)** has been created for maximum effectiveness [4] and is described as follows: relies on the DE differential mutation scheme and enforce RND hard constraints. Before fitness computation of the trial vector, each gene is checked to see if the location is an available BTS location (since differential mutation will create non legitimate alleles). If not, it is replaced with the nearest available location (using the Euclidian distance) not yet in the offspring. Further details about this operator and its effectiveness can be found in [4].



0 DE Parameters
Objective Function
Without constraint definition
DE Control Parameters
Individual Size (D) = 2
Population Size (NP) = 6
Mutation Constant (F) = 0.90
Crossover Constant (Cr) = 0.50
Individual
Parameter 1
Parameter 2
Fitness
Cost Value
1 SELECTION
a) Randomly choose target vector
b) Randomly choose 3 vectors
Current Population (Generation G)
Individual 1
Individual 2
Individual 3
Individual 4
Individual 5
Individual 6
2 MUTATION
Mutation Formula
Pre Trial
* F
3 CROSSOVER
Crossover Strategy
Trial
With probability Cr select parameter value from Pre Trial vectror, otherwise select value from target vector
4 REPLACEMENT
Fitness Function
Replace target vector if trial vector is better (fitness based)
Replacement Strategy
Population (Generation G+1)

Fig. 2. Illustrated example of Differential Evolution algorithm depicting the usage of floating-point encoding in its Bio-inspired strategy.

In its canonical form, the DE algorithm is only capable of handling continuous variables. The NPDM operator needs thus to enforce both integer representation and the location constraints, although, the donor vector will use underlying temporary floating point values. According to [20], the handling of integer and discrete variables in DE can be done rather easily, by truncating a real value to an integer, just before the fitness function evaluation (Eq. 12, where *INT(x)* is the truncate function).

$$f_{\cos t}(Y_i), i = 1, \ldots, n_{param} \qquad \textbf{(12)}$$

Where

$$y_i = \begin{cases} x_i & \text{, for continuous variables} \\ INT(x_i) & \text{, for integer variables} \end{cases}$$

$$x_i \in X$$

We also included an enhancement technique [4], called Superimposed Grid Initialization (SIGRI), with two intentions in mind: (a) reduction and orientation of the search space in order to route the global search direction, and (b) creation of alternate initial populations with good diversity that can effectively empower individual collaboration. This approach is based on the fast paced convergence achieved by DE and by the usage of its canonical self adapting operator. This simple heuristic defines boundaries on each of the gene initial values. The same boundaries are to be applied at the gene level for each individual in the population.

$$X = (x_1, \ldots, x_{n_{param}}), x \in R \qquad (13)$$

Where

$$x_j^{(L)} \leq x_j \leq x_j^{(U)}, j = 1, \ldots, n_{param}$$

The lower and upper boundary constraints for each one of the genes composing the individual are $x^{(L)}$ and $x^{(U)}$, respectively. Further details about SIGRI and its effectiveness can be found in [4].

3 Non-dedicated High Throughput Computing

Although the differential evolution approach studied in this paper aims to reduce the computational effort, the search space is still considerable, and thus we resorted high throughput computing. Specifically, the executions of the experiments described in this paper were carried out through the Condor high throughput computing framework [2]. Specifically, the Condor framework permits to harvest computing resources that would otherwise be left idle, allowing users with access to the Condor system to submit batches of independent tasks. These tasks are then scheduled by the Condor master over the available computing resources. If a task does not complete in the assigned machine – for instance, the remote machines is taken back for interactive usage or the machine is simply turned off – the execution lease times out after a given time interval and Condor automatically reschedules the task to another machine. All of this is practically transparent to the application programmer, with application submitters only providing the unchanged application binary (this needs to be a console application), a specially tailored submit file (this is a simple text file holding instructions to the Condor system, such as the required environments, for example, Windows or Linux, or what are the command-line arguments to be passed for the application), and the required input files.

Since Differential Evolution algorithms are computing intensive and can easily be split in independent tasks, an easy to use high throughput computing system like Condor is an important tool for running such class of algorithms, and in fact was instrumental for conducting our experiments.

4 Results

In this section, we present the main results. We first characterize the used computing environment and then present and discuss main results.

4.1 Computing Environment

At the academic institution where all experiments were performed, a Linux machine fulfils the role of the Condor server, while the Condor client is installed over 170 Windows XP machines that are distributed through 10 classrooms (each classroom holds 17 machines). Four additional client machines (the fastest ones) belong to an advanced laboratory. The main characteristics of the pool of client machines are reported in Table 1 grouped by types (from type A to type F). The columns INT and FP refer respectively to NBench benchmark [21] integer and floating-point performance indexes. The NBench's indexes are used to assess relative performance among the monitored machines, since the same benchmark binary was used throughout the machines. Finally, the column INTFP represents the combination of both INT and FP indexes, facilitating the comparison between machines. As can be seen by this last column, the fastest machines – type F – are roughly 70% faster than the slowest ones (type A). Furthermore, the Windows machines are primarily assigned for teaching activities (essentially to support classes), and are also used by students for their practical assignments and other e-activities (e-mail, web, etc.). Therefore, the machines are primarily devoted to interactive usage, with Condor tasks being scheduled to a machine only when no interactive user is logged on. Additionally, whenever an interactive login occurs at a machine that is running a Condor task, the task is suspended, and after 10 minutes, if the interactive usage persists, the task is evicted and rescheduled to another machine. Again, this conservative configuration was adopted to prioritize interactive users over Condor's tasks, although it provokes a high churn rate of tasks, since machines are frequently used for interactive usage, resulting in a vast percentage of interrupted executions. This is aggravated by the fact that the classrooms are heavily used (they are open 20 hours on weekdays, and 13 hours on Saturdays, only closing on Sundays).

Table 1: Main characteristics of the Condor's pool of machines that run the experiments.

Type	Qty	CPU	INT	FP	INTFP
A	34	P4 @2.4 GHz	39.010	36.557	37.784
B	51	P4 @2.6GHz	42.038	40.323	41.181
C	51	P4 @3.0 GHz	43.198	41.715	42.457
D	17	P4 @3.2 GHz	46.705	44.257	45.581
E	17	P4 @3.4 GHz	48.975	46.800	48.888
F	4	Core 2 Duo 6600 @ 2.40 GHz	63.697	63.736	63.717

A subtle issue regarding Condor lies in the way it handles machines with the hyperthreading technology (in our case, machines of type B, C, D and E all have hyperthreading enabled). For Condor, the hyperthreading machines are, by default regarded as having two processors, and thus two virtual clients are allocated per machine. This means that up to two tasks can be simultaneously running in those machines, a situation that substantially lengthens the execution time of the tasks, since the hyperthreading technology has been reported to marginally improve performance by 5% or 10% (this depends on the workload). In our case, the overload of machines with hyperthreading resulted in higher execution times per task, which in turn further exposes the execution of tasks to interruptions caused by machines being claimed for interactive usage. Note that although dual core machines (type F) are also used by Condor to host two virtual clients, the architecture of such machines effectively supports two simultaneous tasks without substantial (if any) performance degradation (each task runs in an individual core).

4.2 Experiments

In our experiments we considered base station transmitters with omni-directional cells, each transmitter having 2810 associated matrix point cells (coverage radius size is 30). Other cell shapes are possible but deferred for future work. We used a 450x300 matrix map depicting the city of Malaga (Spain) that covers 6.4 x 4.25 Km. Each sector in the grid matrix represent approximately a 15 x 15m squared area. Under this scenario, it is impossible to attain a 100% cover rate due to waste spots (mountains and considerable bodies of water); hence the maximum cover rate is 95.522%. A set of 1000 predefined possible BS locations have been defined. Several combinatory scales have been used, as for instance, in the approach of the Pareto front construction; the number of BS varied from 1 to 100. Further details can be found at [22]. Since DE is well known for being a fast optimisation algorithm, two main tests sets where made within a 100,000 and 200,000 fitness evaluation boundary. Most experimental aspects are based on 30 independent test runs each, yielding a total aggregate of several thousandths executions, performed on Windows XP machines under the control of Condor. Each set of experiment was performed having as parameter set the best known configurations. The code was developed in C# and requires the .Net (or Mono) framework to run.

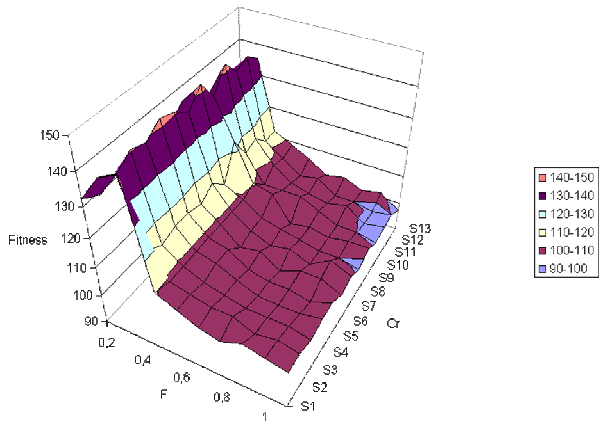


Fig. 3 – F and Cr parameters fine tuning.

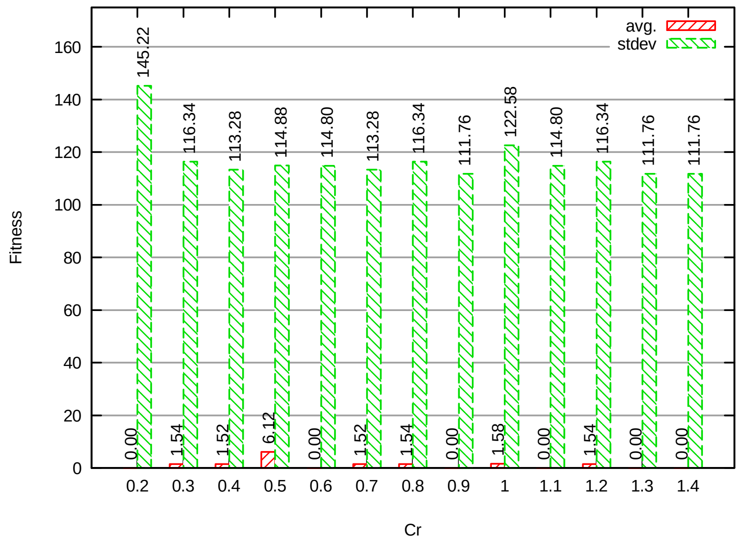


Fig. 4 – Cr reliability based on the F=0.2.

The optimal value for F parameter was set to 0.2 as it clearly emerges from Fig. 3 as the most appropriate. The optimal Cr parameter (0.6) was chosen based on the maximum average fitness values and minimal standard deviation within the 0.2 F axes (Fig. 4). Fig. 5 depicts the optimal number of BTS to use, which is weighted by the usage of the fitness function (Eq. 2). The results point out that 45 is the optimal number of antennas to deploy.

Although the problem has not been tackled in a multi-objective fashion, it was possible to build an approach to the Pareto front concerning the maximum cover rate and minimal deployed BTS objectives (Fig. 6). The construction of the Pareto front required 3,000 experiments (30 runs x [1…100] BS). Fig. 7 depicts the convergence attained by the usage of DE within a 50,000 fitness evaluation boundary. Although experiences had been prepared with a stopping criterion of 200,000 fitness evaluations, no better results were obtained after 50,000.

Figure 8 plots the wall clock times of the runs relatively to the number of considered antennas. Specifically, the plot presents, for every count of antennas, the best wall clock execution time (out of the 30 experiments that were run for every number of antennas), the average and the standard deviation observed for these 30 experiments. From the plot, it clearly emerges that, for each count of antennas, the best execution times progress linearly, an indication that the algorithm scales linearly with the number of antennas. As an aside note, the linear growth of the best execution times also indicates that the fastest execution times were all achieved by the fastest machines (type F).

The high divergences between the average and the best execution times stems from the computer pool’s heterogeneity (as stated before, type F machines are roughly 70% faster than type A machines). In fact, the tasks submitted to Condor were configured to prioritize scheduling over the fastest machines (this way, when two or more machines are available, Condor selects the fastest one). The influence of the machines' heterogeneity on the results is further confirmed by the discrepancy among the average execution times, with some average execution times being close to the respective best time, while others are as far as 70% apart. This also applies to the high fluctuation of the standard deviations.

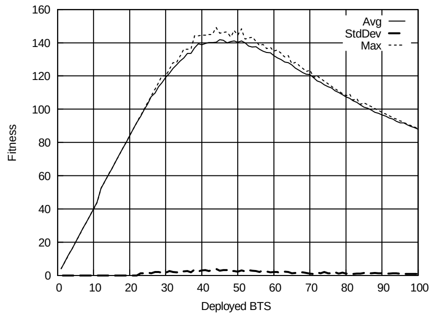


Fig. 5 – Optimal number of BTS (fitness/deployed BTS).

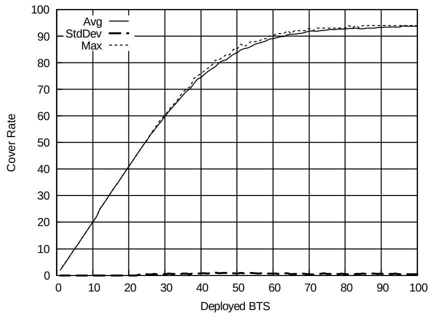


Fig. 6 – Approach to the Pareto Front.

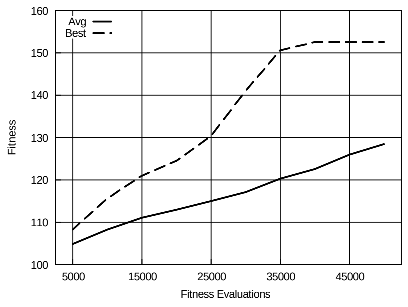


Fig. 7 – DE Algorithm progression series.

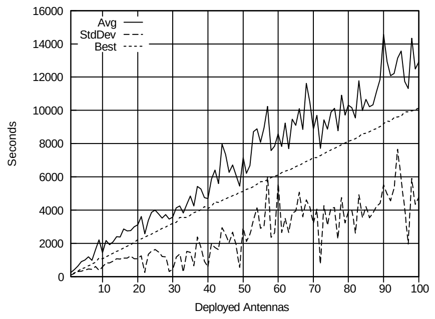


Fig. 8 – Execution profiling.

6 Related work

Vega-Rodríguez et al. [23] resorted to the BOINC middleware system [24] to perform a wide-scale study of the applicability of the Population-Based Incremental Learning algorithm to the RND problem. Comparatively to the Condor approach, the BOINC middleware requires not only a specifically tailored application (the application needs to use the BOINC client API), but also a whole server-side infrastructure (used (1) to distribute tasks and collect results and (2) to give feedback to resource donors through forums and contribution ranks). Moreover, in order to participate in the computation, resource donors need to be attracted to the project and explicitly register their machines to the project. On the contrary, the Condor approach is mostly transparent, since only rather simple submit files need to be prepared, with practically no changes required for the applications itself. However, a BOINC-based project has the potential to reach several thousands of resource donors, where all expenses (computer acquisition and maintenance, network bandwidth, etc.), apart the server-side ones, are supported by the volunteers, while Condor is limited to single-geographical sites (mostly local area environments) and thus has a much lower potential for scalability. It should be noted that Vega-Rodríguez et al. used a smaller map representation and only 349 possible BS positions. The BS positions were tailored so that it would be possible to fit the BS nicely in order to get 100% coverage without a single overlap, using a square wave model, so results are not comparable at all. Furthermore, the computing power required for this magnitude of problems is lowermost when compared to our current work.

7 Conclusion and future work

DE convergence has been proven on a large-scale real-world problem, although the quality of the results are non-conclusive yet, since no related work has been developed at this echelon, i.e., no other work that we are aware of use this magnitude of combinatorics. Using Condor-based high throughput computing offers a credible framework for providing a significant speed-up to evolutionary optimisation experimentation in science and engineering, especially in cases where dedicated resources are not possible to achieve.

Future work includes the study of other bio-inspired algorithms, such as VNS, Scatter Search and GRASP, as a mean of finding a comparison base on problems of this order of magnitude. In this research line, we will continue using desktop grid computing with Condor in order to speedup all our experiments and create more effective algorithms resorting to the Alchemi platform to gather parallelism, building-up, exploiting and comparing resulting algorithms that are also suitable to tackle the RND problem. Another line of research is the fine tuning of the Condor platform itself via a task application-level checkpointing independent API, so that interrupted executions under a Condor host can be resumed, reducing the amount of lost computing power.

References

1. M. Garey and D. Johnson. Computers and Intractability: a Guide to the Theory of NP-completeness. Freeman and Cº, 1979.
2. Thain ,D.,Tannenbaum,T., Livny, M.”Distributed Computing in practice: the Condor Experience”. Concurrency and Computation Practice and Experience, 17(2-4), 2005.
3. Mendes S., Gomez-Pulido J., Vega-Rodriguez M., Sánchez-Perez J., “A Differential Based Algorithm to Optimize the Radio Network Design Problem”, 2nd IEEE International Conference on e-Science and Grid Computing, Amsterdam, Netherlands, December 2006.
4. Mendes S., Gomez-Pulido J., Vega-Rodríguez M., Pereira M., Sanchez-Pérez J., “Fast Wide Area Network Design Optimisation using Differential Evolution”, AdvComp 2007 – International Conference on Advanced Engineering Computing and Applications in Sciences, Papeete, French Polynesia, Tahiti, November 2007.
5. P. Calegari, F. Guidec, P. Kuonen, and D. Kobler. Parallel island-based genetic algorithm for radio network design. Journal of Parallel and Distributed Computing, 47:86–90, 1997.

6. P. Calegari, F. Guidec, P. Kuonen,. Combinatorial Optimization Algorithms for Radio Network Planning. Journal of Theoretical Computer Science, Vol. 263, issue 1-2 pages 235–265, 2001.
7. E. Alba. Evolutionary Algorithms for Optimal Placement of Antennae in Radio Network Design, NIDISC'04, Santa Fe, New Mexico, USA, pp. 168, 2004.
8. Price, K. and R. Storn (2006), Web site of DE as on July 2006, the URL of which is: http://www.ICSI.Berkeley.edu/ ~storn/code.html
9. S. Khuri and T. Chiu. Heuristic algorithms for the terminal assignment problem. In Proceedings of the 1997 ACM Symposium on Applied Computing, 1997.
10. Price, K. and R. Storn (1997), “Differential Evolution – A simple evolution strategy for fast optimisation”. Dr. Dobb’s Journal, 22 (4), 18 – 24 and 78.
11. Vasan A. and Raju K., Optimal Reservoir Operation Using Differential Evolution, Birla Institute of Technology and Science, Pilani, Rajasthan 333031, 2004.
12. Joshi R. and Sanderson A., Minimal Representation Multisensor Fusion Using Differential Evolution. IEEE 0-8186-8138-1, 1997.
13. Vega-Rodríguez M., Gomez-Pulido J., Alba E., Pérez D., Mendes S., Molina G., Different Evolutionary Approaches for Selecting the Optimal Number and Locations of Omnidirectional BTS in a Radio Network, Eurocast, Eleventh International Conference on Computer Aided Systems Theory, Spain, Feb. 2007.
14. Gomez-Pulido J., Web site of Optimización y Ambientes de Red (OPLINK::UNEX) as on september 2006, the URL of which is: http://oplink.unex.es/
15. Hussein A. Abbass and Ruhul Sarker. The Pareto Differential Evolution Algorithm, International Journal on Artificial Intelligence Tools, Vol. 11, No. 4, pp. 531--552, 2002.
16. Storn R. and Price K., Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech. report TR-95-012, ICSI, 1995.
17. Storn R., On the usage of differential evolution for function optimization. NAFIPS 1996, Berkeley, pp519-523, 1996.
18. Price K., An Introduction to Differential Evolution. In new methods for optimization. ISBN:0-07-709506-5, 1999, Mc-Graw-Hill, UK.
19. Salomon R., Reevaluating Genetic Algorithm Performance under Coordinate Rotation of Benchmark Functions; A survey of some theoretical and practical aspects of genetic algorithms. BioSystems, 39(3):263-278, 1996.
20. Lampinen J., and Zelinka I. Mixed Variable Non-Linear Optimization by Differential Evolution, Proceedings of Nostradamus ’99, 2nd International Prediction Conference, Oct. 7-8, 1999.
21. Mayer,Uwe F. NBenchProject (http://www.tux.org/~mayer/linux/), 2007.
22. OPLINK: http://oplink.lcc.uma.es/problems/rnd.html, January 2007.
23. Miguel A. Vega-Rodríguez, David Vega-Pérez, Juan A. Gómez-Pulido and Juan M. Sánchez-Pérez, “Radio Network Design Using Population-Based Incremental Learning and Grid Computing with BOINC", Lecture Notes on Computer Sciences, 4448: Applications of Evolutionary Computing. Berlin Heidelberg, 2007.
24. Anderson, D., “BOINC: A System for Public-Resource Computing and Storage”, Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, 2004.