Cell-to-switch assignment in cellular networks using barebones particle swarm optimization

Sotirios K. Goudos\textsuperscript{a)}, Konstantinos B. Baltzis, Christos Bachtsevanidis, and John N. Sahalos

RadioCommunications Lab., Dept. of Physics, Aristotle University of Thessaloniki
Thessaloniki 541 24, Greece
\textsuperscript{a}) sgoudo@physics.auth.gr

Abstract: Cell assignment is an important issue in the area of resource management in cellular networks. The problem is an NP-hard one and requires efficient search techniques for its solution in real-time. In this letter, in order to minimize the cabling and handoff costs, we use two novel discrete particle swarm optimization (PSO) algorithms, the barebones (BB) and the exploiting barebones (BBExp) PSO variants. The impact of network and algorithm parameters on the solution accuracy and computational cost of the methods is investigated. Comparisons with optimization methods in the literature demonstrate the benefits of our proposal.

Keywords: cell assignment, PSO, barebones PSO, integer programming, cellular networks

Classification: Wireless circuits and devices

References

1 Introduction

The effective assignment of cells to switches in order to minimize the cost of network deployment is a challenging issue in cellular networking.

The cell-to-switch assignment (CSA) problem consists of optimally assigning cells to network switches subject to certain constraints such as the call volume of each cell and the switches capacity [1]. The objective of the optimization is the reduction of implementation and operational costs. Usually, the cost function considers the cost of linking cells to switches (cabling cost) and the cost of handoff between different cells (handoff cost). The problem is an NP-hard one with exponential complexity and cannot be solved analytically in real size networks. Optimization techniques such as tabu search, Ant Colony Optimization (ACO) and genetic algorithms are commonly used in the literature [2, 3, 4].

In this letter, we solve the CSA problem using the barebones (BB) and the exploiting barebones (BBExp) variants of particle swarm optimization (PSO) [5]. It has been found that the two methods outperform popular evolutionary algorithms in integer programming test problems, e.g. [6, 7]. To the best of the authors’ knowledge, this is the first time they are applied to a CSA problem. In the following subsections, we discuss the cell assignment problem and the basic principles of the methods. A comparative study of BB and BBExp PSO against ACO [3] and binary PSO (BPSO) [8] investigates the impact of network and algorithms parameters on their performance and exhibits the merits of our proposal.

2 Problem formulation

We consider \( n \) unique and distinct cells in a given service area and \( m \) switches with known location and traffic parameters. The objective is the optimum assignment of cells to switches in order to minimize the total cost that comprises the handoff and cabling costs.

In single-homing CSA, each cell belongs to one cluster and it is assigned to one switch at a time. In this case, the objective function to be minimized
is [1]:
\[ \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ik} x_{ik} + \sum_{i=1}^{n} \sum_{j=1}^{n} h_{ij} (1 - y_{ij}) , \quad k = 1, \ldots, m \]  
(1)

where \( c_{ik} \) is the cabling cost per time unit between cell \( i \) and switch \( k \), \( x_{ik} \) is a parameter that takes the value one when cell \( i \) is assigned to switch \( k \) (otherwise, \( x_{ik} = 0 \)) and \( h_{ij} \) is the cost per time unit for the handoffs that occur between cells \( i \) and \( j \). The first term in (1) gives the total cabling cost; the second one is the total handoff cost per time unit among cells. Therefore, \( y_{ij} \) is defined as
\[ y_{ij} = \sum_{k=1}^{m} x_{ik} x_{jk}, \quad i, j = 1, \ldots, n \]  
(2)

Obviously, \( y_{ij} \) is one when cells \( i \) and \( j \) are connected to the same switch, otherwise it is zero. The product \( x_{ik} x_{jk} \) in (2) defines the variable
\[ z_{ijk} = x_{ik} x_{jk}, \quad i, j = 1, \ldots, n \quad \text{and} \quad k = 1, \ldots, m \]  
(3)

that is zero unless cells \( i \) and \( j \) are connected to switch \( k \). In this case, it takes the value one.

Cell assignment is subject to further constraints. The call handling capacity of each switch should not be violated at any time, i.e.
\[ \sum_{k=1}^{m} \lambda_i x_{ik} \leq M_k, \quad i = 1, \ldots, n \]  
(4)

where \( \lambda_i \) is the number of calls that cell \( i \) handles per unit time and \( M_k \) is the call handling capacity of switch \( k \). Also, each cell is assigned only to one switch, i.e.
\[ \sum_{k=1}^{m} x_{ik} = 1, \quad i = 1, \ldots, n \]  
(5)

The optimization problem defined by (1) and subject to (2)-(5), can be converted [1] to an integer programming one by replacing (3) with the
\[ 0 \leq z_{ijk} \leq x_{jk}, \quad x_{ik} \quad z_{ijk} \geq x_{ik} + x_{jk} - 1, \quad i, j = 1, \ldots, n \quad \text{and} \quad k = 1, \ldots, m \]  
(6)

In dual-homing, a cell is assigned to two switches. The problem was studied in [2] by considering and simultaneously optimizing two assignment patterns. Quantities \( x'_{ik}, y'_{ij}, z'_{ijk}, h'_{ij} \) and \( \lambda'_i \) are defined for the second pattern similarly to before. The single-homing constraints are used in both patterns while (1) becomes
\[ \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ik} x_{ik} + \sum_{i=1}^{n} \sum_{j=1}^{n} h_{ij} (1 - y_{ij}) + \sum_{i=1}^{n} \sum_{j=1}^{m} h'_{ij} (1 - y'_{ij}) , \quad k = 1, \ldots, m \]  
(7)

Additional constraints related to the link cost between cell \( i \) and switch \( k \) are:
\[ w_{ik} \geq x_{ik}, x'_{ik}, \quad i = 1, \ldots, n \quad \text{and} \quad k = 1, \ldots, m \]  
\[ w_{ik} \leq x_{ik} + x'_{ik}, 1 \]  
(8)
Parameter $w_{ik}$ is one when $x_{ik}$ or $x'_{ik}$ is one, i.e. when cell $i$ is assigned to switch $k$ in at least one of the two assignment patterns, otherwise it is zero.

3 Barebones Particle Swarm Optimization

Particle swarm optimization [5] is a population-based stochastic optimization technique inspired by the social behavior of birds flocking, where a swarm of individuals (particles) fly through the search space. The particles move in the search space by following the current optimum ones. The system is at first initialized with a population of random particles (solutions) and searches for optima by updating the particles positions in any iteration. Each particle position is updated by finding two optimum values, the particle’s best solution (fitness) achieved so far, $p_{best}$, and the global best value obtained so far by any particle, $g_{best}$. After finding $p_{best}$ and $g_{best}$, each particle updates its position and velocity. The algorithm is executed repeatedly until a specified number of iterations is reached or if the velocity updates are close to zero. Particles’ quality is measured using a fitness function that reflects the optimality of a particular solution. In recent years, PSO has been applied successfully to several engineering problems, such as parametric object recognition [9] and mobile network design [10].

Kennedy [6] proposed a new PSO approach, the BB PSO, where the standard PSO velocity equation is removed and replaced with samples from a normal distribution. In this method, the position update rule for the $n$th particle becomes

$$x_n = N \left( \frac{p_{best_n} + g_{best_n}}{2}, |p_{best_n} - g_{best_n}| \right)$$

where $N(.,.)$ denotes the normal distribution. The method allows particles with $p_{best_n}$ significant different from $g_{best_n}$ to make large step sizes towards it. When $p_{best_n}$ is close to $g_{best_n}$, step size decreases and limits exploration in favor of exploitation.

In [6], a variation of BB PSO, the BBExp PSO, was also proposed. In this method, approximately half of the time velocity is based on samples from a normal distribution; for the rest of the time, velocity is derived from the particle’s personal best position. The position update rule, (9), is modified into

$$x_n = \begin{cases} N \left( \frac{p_{best_n} + g_{best_n}}{2}, |p_{best_n} - g_{best_n}| \right), & U(0,1) > 0.5 \\ p_{best_n}, & \text{otherwise} \end{cases}$$

where $U(.,.)$ denotes the uniform distribution. In BBExp PSO, position updates equal $p_{best_n}$ for half of the time resulting in the improved exploitation of $p_{best_n}$ compared to the BB PSO. It has to be noticed that barebones PSO require no parameter tuning; for further information and applications, see [6, 7].
4 Results and discussion

In order to test the effectiveness and performance of BB and BBExp PSO in the CSA problem, we conducted 100 independent trials for each algorithm. The average results are presented and compared with results obtained from BPSO and ACO methods. All the algorithms were compiled with Borland C++ Builder 5.0 compiler and ran on a PC with Intel Core 2 Duo E8500 at 3.16 GHz with 4 GB RAM. In the proposed barebones PSO variants, the only parameter we set was the swarm size. In the examples presented here, this was set to 5 and 10 particles. The ACO parameters were the same as in [3]. For the BPSO, we set the learning factors $c_1$ and $c_2$ equal to two. Systems with varied number of cells and switches that range from 15 to 200 and from 2 to 7, respectively, were considered.

The percentage of successfully obtained solutions as a function of the number of cells and switches indicates the solution accuracy of the algorithms. Fig. 1a (1b), presents the results of the application of BB PSO, BBExp PSO, BPSO and ACO in a single-homing system for swarm size equal to 5 (10) particles. In the first case, BB PSO outperforms the other methods in systems with small complexity but as the complexity increases ($n/m = 150/6$ and $200/7$) BBExp gives the best results. As it was expected, solution accuracy decreases with system complexity. Similar conclusions are drawn when we use 10 particles, see Fig. 1b. In general, BB PSO outperforms the other methods for small $n$ and $m$. However, its performance degrades with system complexity; in this case, BBExp gives better results. In any case, at least one of BB and BBExp PSO is better than BPSO and ACO.

![Fig. 1. Successful solution vs cells/switch: a) 5 particles, b) 10 particles.](image_url)

The effectiveness of the proposed algorithms was further confirmed by the calculation of the computational time required for the derivation of the previous results. Figs. 2a and 2b show small differences in results between the four algorithms. In all cases, barebones PSO are slightly faster. BBExp PSO outperforms BB PSO as system complexity increases. The computational cost of the methods grows exponentially with $n/m$ and increases with the swarm size.
Fig. 2. Computational time vs cells/switch: a) 5 particles, b) 10 particles.

Figs. 1 and 2 shows that for small number of cells and switches BB PSO gives the best results; as system complexity increases, BB PSO performance is similar to BPSO and ACO. In this case, BBExp gives better results in terms of successful solutions and computational cost. It is noticed that barebones PSO outerperform BPSO and ACO in the cases that are studied here.

The previous examples refer to single-homing. Comparative results for single- and dual-homing systems are presented in Figs. 3a and 3b. In these examples, it is $n = 15$ and $m = 2$. We notice that in the dual-homing system BBExp PSO is slightly faster than the other methods. As it was expected [1], optimization in dual-homing is more time-consuming compared to single-homing.

Fig. 3. Computational time in single- and dual-homing systems: a) 5 particles, b) 10 particles.

5 Conclusions

BB and BBExp PSO variants are global optimizers designed for solving integer programming problems. In this paper, we have applied the above methods to the CSA problem in cellular networks. To the best of the authors’ knowledge, this is the first time these two algorithms are applied to cell-to-switch assignment. The derived results showed the improved performance of BB
and BBExp PSO compared to other popular discrete optimization methods in the literature, in terms of successfully obtained solutions and execution time. Comparison between BB PSO and BBExp PSO has shown that the second is more efficient when the complexity of the problem increases.