Expert Systems With Applications 106 (2018) 290–304

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A stop-and-start adaptive cellular genetic algorithm for mobility management of GSM-LTE cellular network users

Zakaria Abdelmoiz Dahi[a,*], Enrique Alba[b], Amer Draa[a]

[a] Department of Fundamental Computer Science and Its Applications, Constantine 2 University, Constantine, Algeria
[b] Department of Lenguajes y Ciencias de la Computación, Malaga, Spain

ARTICLE INFO

Article history:
Received 30 November 2017
Revised 6 February 2018
Accepted 28 February 2018
Available online 1 March 2018

Keywords:
Cellular networks
Cellular Genetic Algorithms
Adaptation

ABSTRACT

The optimisation of the user tracking process is one of the most challenging tasks in today's advanced cellular networks. In this paper, we propose a new low-complexity adaptive cellular genetic algorithm to solve this problem. The proposed approach uses a torus-like structured population of candidate solutions and regulates interactions inside it by using a bi-dimensional neighbourhood. It also automatically adapts the algorithm's parameters and regenerates the algorithm's population using two algorithmically-light operators. In order to draw reliable conclusions and perform an encompassing assessment, extensive experiments have been conducted on 25 differently-sized realistic networks. The proposed approach has been compared against 26 state-of-the-art algorithms previously designed to solve the mobility management problem, and a thorough statistical analysis of results has been performed. The obtained results have shown that our proposal is more efficient and algorithmically less complex than most of the state-of-the-art solvers.

1. Introduction

The popularity and accessibility of the services provided by cellular networks made them leaders in the field of communication. According to the GSM[1] association, in 2017 almost half of the world is using mobile communication services (Berrocal-Plaza, Vega-Rodriguez, & Sanchez-Perez, 2015a). This has made the mobile industry a field with tremendous technological and economical competition, where service quality is a key component. Quality can be measured by how good, reliable and fast a service is delivered/requested to/from a user. This has made the Mobility Management System (MMS) a sensitive core since it handles necessary information for the user's mobility management. Studies have shown that messages generated by the MMS when trying to locate a mobile user induce more than 33% of the signalling traffic passing through the bandwidth (Berrocal-Plaza, Vega-Rodriguez, & Sanchez-Perez, 2014d; Dahi, 2017). Besides the fact that the bandwidth is the most economically-valuable resource, such overhead results in many issues such as network congestion, service failure and communication failure, that are unacceptable by both operators and costumers (Dahi, Mezioud, & Alba, 2016). This has made the optimisation of the users tracking one of the most challenging tasks in today's 2G, 3G, and 4G networks.

The high complexity of the Mobility Management Problem (MMP) (Bar-Noy & Kessler, 1993; Hać & Zhou, 1997) has made of metaheuristics a promising solution. Many approaches have been previously devised in the literature and are summarized in Table 1.

Cellular Genetic Algorithms (cGAs) are Evolutionary Algorithms (EAs) that organise the population as a grid in which each node represents an individual and where interactions between the latter are restricted to a neighbourhood. Like the cGAs, most of today's algorithms proposed to solve problems such as the MMP are problem or instance-dependent. Actually, their performance is greatly related to the quality of their parameter tuning. The latter is performed over many exhaustive and long processes in order to better suit a particular problem or instance. In addition, such tuning requires advanced knowledge of the algorithm used and the problem being addressed (Dahi et al., 2016). So, any change in the problem/instance properties will probably decrease the efficiency of the algorithm and a new loop of tuning will be necessary each time in order to recover the sought efficiency.

All these facts restrict the use of most of today's hand-tuned (static) metaheuristics to experts within pure abstract research instead of practitioners within real-life industrial environments. As a solution to this shortfall, many research efforts are being deployed in order to design algorithms containing mechanisms that can adapt automatically the value of the algorithm's parameters in

* Corresponding author.
E-mail addresses: zakaria.dahi@univ-constantine2.dz (Z.A. Dahi), eat@lcc.uma.es (E. Alba), draa.amer@gmail.com (A. Draa).

[1] GSM: Global System for Mobile communications.

https://doi.org/10.1016/j.eswa.2018.02.041
0957-4174/

Table 1
Literature review: state-of-the-art solvers of the mobility management problem.

	The approach	Proposed in
Mono	Genetic Algorithm (GA), Tabu Search (TS) and Ant Colony Optimisation algorithm (ACO)	(Subrata & Zomaya, 2002a; 2002b; 2003a; 2006)
	Geometric Particle Swarm Optimisation algorithm (GPSO) and Hopfield Neural Network with Ball Dropping (HNN-BD)	(Alba et al., 2008)
	A modified Hopfield Neural Network combined with Ball Dropping technique (HNN-BD)	(Taheri & Zomaya, 2008)
	Differential Evolution algorithm (DE)	(Almeida-Luz, Vega-Rodriguez, Gomez-Pulido, & Sanchez-Perez, 2008; Almeida-Luz et al., 2011; Patel & Bhatt, 2014; Wang et al., 2009)
	Simulated Annealing-based approach (SA)	(Mehta & Swadas, 2009a; 2009b)
	Combined Particle Swarm Optimisation algorithm with the Genetic Algorithm (GA-PSO)	(Wang & Si, 2010)
	Ant Colony Optimisation algorithm (ACO)	(Kim, Kim, Mani, Kim, & Agrawal, 2010)
	Scatter Search algorithm (SS)	(Almeida-Luz et al., 2010a; 2010b)
	Parallel cooperative strategy based on a team evolutionary algorithms	(González-Álvarez et al., 2012; Rubio-Largo et al., 2010)
	Genetic Algorithm (GA)	(Baburaj & Alagarsamy, 2011; Baburaj et al., 2010; Parija, Addanki, Sahu, & Singh, 2015; Patra & Udgata, 2011)
	Particle Swarm Optimisation algorithm (PSO)	(Kim et al., 2012)
	Group Search Optimizer (GSO)	(Wang et al., 2014)
	Hybrid approach combining both the GA and the SA	(Vekariya & Swadas, 2015)
	Evolving cellular automata-based-solution	(Subrata & Zomaya, 2003b)
	IBM ILOG CPLEX Optimizer and the Non-linear Optimization with the Mesh Adaptive Direct search (NOMAD)	(Berrocal-Plaza et al., 2014c; 2014d)
	Oscillatory-Increasing adaptive Genetic Algorithm (OI-GA)	(Dahi et al., 2016)
Multi	Strength Pareto Evolutionary Algorithm 2 (SPEA 2)	(Berrocal-Plaza et al., 2014b; 2014c; Berrocal-Plaza et al., 2015b)
	Non-dominated Sorting Genetic Algorithm II (NSGA II)	(Berrocal-Plaza et al., 2014a; Berrocal-Plaza et al., 2014d; 2015a; Berrocal-Plaza et al., 2015b)

order to better suit the problem or the instance being addressed without the intervention of an external agent. However, such an adaptation is generally resulting in more complex algorithms than static ones with extra hidden-tunable parameters and computationally heavier calculations. Many approaches of adaptation have already been proposed in the literature, but all can be classified as (1) *deterministic* (uses off-line deterministic rule), (2) *adaptive* (uses the algorithm's feedback) or (3) *self-adaptive* (parameters are evolved along with the problem solution) (Alba & Dorronsoro, 2005). The first category, deterministic, is the one studied in our work.

In this paper, we propose an efficient and low-complexity adaptive cellular genetic algorithm for solving the MMP. The scalability, efficiency and robustness of the proposed approach have been assessed by solving twenty-five realistic instances with different sizes. In addition, twenty-six of the top-ranked algorithms designed to solve the MMP are taken as a comparison basis and several statistical analysis tests have been performed.

The remainder of this work is structured as follows. In Section 2, we introduce basic concepts of mobility management, cGAs and adaptation within EAs. In Section 3, we present the proposed approach. Section 4 is dedicated to the experimental study. Finally, we conclude the paper in Section 5.

2. Preliminaries

In this section, we present basic notions of the mobility management in cellular networks, the cellular genetic algorithm and adaptation within EAs.

2.1. The mobility management of users

The mobility management consists of tracking mobile users to whom a given service (e.g. messages, calls, etc.) is destinated. Roughly speaking, it is widely believed that no matter the network's generation or technology being used, this task is performed by a given component in the system architecture. For each mobile user, this component keeps trace and manages two mobility pieces of information: the cell where the user is situated and the area to which this cell belongs. The latter is a set of cells that share the same and unique identifier. Thus, a user's mobile can freely move within cells of the same area without updating its area identifier. Each time a mobile user enters a new area, he has to update his identifier by notifying the mobility management core using a *location-update* message.

When a given service (e.g. video-call) is destinated to a user, the network has to know first in which cell the concerned mobile user is situated. Having the identifier of the user's area, the mobility management system performs a polling cycle. In each polling cycle, *paging* messages are sent to all the cells within the user's area in order to locate the mobile user.

A mobility management scheme defines how the location update and paging are performed. For the location-update task, two approaches exist: dynamic and static (see Fig. 1(a)). In the first approach, updates are based on the change of the user calls and mobility patterns, which usually requires the on-line collection and processing of data and consumes significant computing power, while updates in the static approach are independent of the user's characteristics. In fact, they are based on the network's topology change, which allows efficient implementation and low computational requirements. Thus, the static approach is the one studied in our work. In this last approach, we can cite the always-update, never-update, reporting cell and location area schemes. Both always and never-update schemes are almost never used in real networks since they imply either extreme costs of paging or location update, while the remaining schemes are considered as the standard location-update schemes and are practically implemented in real-life networks (Razavi, 2011). Considering these facts, the reporting-cell scheme is the one studied in our work. Within this scheme, cells of the network are labelled either as *reporting* or *non-reporting* cells. The user's mobile performs a location update only if it enters a reporting cell, while it can freely move within non-reporting cells without acknowledging its new location. Previous works (Razavi, 2011), showed that even with some helpful data got about the network, finding the optimum arrangements of the reporting cells is an NP-hard problem.

Let us consider now the paging task, two main schemes exist: *standard* and *selective* (see Fig. 1(b)). In the static paging schemes,

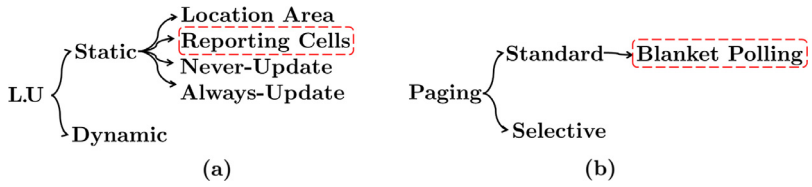

Fig. 1. (a) location-update and (b) paging schemes in cellular networks.

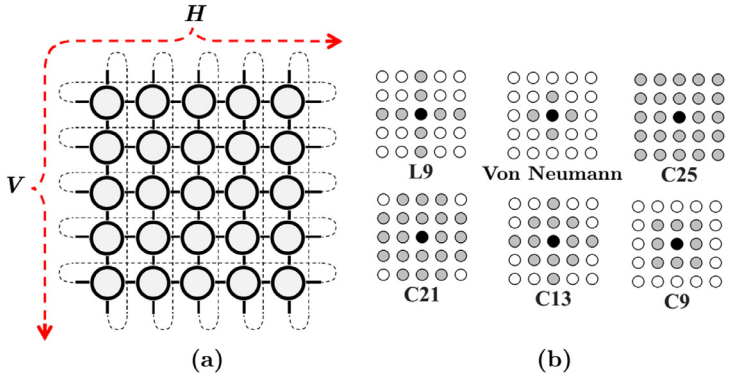

Fig. 2. (a) Example of grid and (b) neighbourhood configurations in cGAs.

such as the blanket polling, all cells of the user's area are paged simultaneously, while in selective schemes the user's area cells are paged selectively on the basis of a given feature (e.g. distance, velocity, etc.). Previous works (Razavi, 2011) showed that even if selective paging schemes reduce the signalling overhead, their use requires a modification of the system implementation and the collection of additional user information, whereas the standard scheme does not require extra knowledge of the user location, which makes it one of the most practical and commonly-used schemes in today's networks (Razavi, 2011). In addition, it is worth to mention that the blanket polling is the paging scheme originally implemented in the reporting-cell scheme. Thus, the latter is the one considered in our work.

Under the light of these facts, the efficiency of the mobility management task relies on the efficiency of the used scheme for both managing and reducing the generated location update and paging costs. The reporting cell scheme was first proposed by Bar-Noy and Kessler (1993). Then, the Reporting Cell Problem (RCP) has been mathematically formulated as a binary optimisation problem by Hać and Zhou (1997).

2.2. Cellular genetic algorithms

A Cellular Genetic Algorithm (cGA) belongs to the class of structured algorithms, where the population is organised as a grid with a length of H nodes and a width of V nodes and where each node of that grid represents a given individual (Alba & Dorronsoro, 2008). The latter interacts with a specific number of individuals according to a given neighbourhood. Many grid shapes and neighbourhoods have been proposed and studied in the literature. Fig. 2(a) represents an instance of a toroidal square grid with H and V equal to 5, whereas Fig. 2(b) illustrates the Von Neumann (or the so-called L5 or NEWS), L9, C9, C13, C21 and C25 neighbourhoods.

The cGA takes as an input a set of individuals (i.e. solutions), and makes them evolve through iterations toward a fitter state by applying a series of operators. However, unlike the basic GA (*unstructured*), the cGA applies these operators sequentially on each individual by browsing the grid node by node. After initialisation, a classical iteration of the cGA consists of applying selection, variation operators, evaluation and replacement. The pseudo-code of Algorithm 1 summarizes the general framework of the basic cGA.

Algorithm 1 The Cellular Genetic Algorithm.

```
1: Initialisation.
2: while termination criterion is not reached do
3:   for every individual in the grid do
4:     Selection within the individual neighbourhood.
5:     Variation operators: crossover and mutation.
6:     Evaluation.
7:     Replacement within the individual neighbourhood.
8:   end for
9: end while
```

The cGA starts by initialising a population of N individuals, where each individual $\vec{X} = \{x_1, x_2, \ldots, x_D\}$, represents a possible solution to the D-dimensional problem being addressed. Each element x_j is a problem variable to be optimised, where $j = 1, \ldots, D$. Theoretically speaking, all initialisation strategies of the classical GA are applicable to the cGA. We can cite, for instance, the heuristic and random ones. After initialisation, the quality of all individuals is evaluated using the objective function of the problem being solved. Then, the best individual $\vec{G}$ is extracted.

Once the initialisation phase is achieved, the cGA selects a given number, M, of parents according to a given strategy. Again, here also all selection strategies of the unstructured GA can be applied to the cGA. Some of them are ranking or tournament-based and others are fitness-proportionate, but all aim at selecting the most promising parents to undergo the variation phase (Sivanandam & Deepa, 2007). Also, it is worth stating that the selection is performed within the neighbourhood of the processed individual.

After selection, the chosen parents go through the breeding process. The latter stands in applying successively crossover and mutation. For both operators, the amount and way the information is exchanged are ruled by a probability (P_c and P_m) and the type of crossover or mutation used. Many types of crossover have been proposed in the literature, we can cite for instance the single-point, two-point and uniform crossovers. Some mutations have

also been previously-devised. We can cite, for example, the bit-flip and Cauchy mutations (Sivanandam & Deepa, 2007).

Having the new offspring produced by applying successively both crossover and mutation, the quality of these individuals is assessed using the objective function of the problem being addressed.

Once the produced offspring are evaluated, a replacement phase is performed, in order to decide what will be the individual composition for the next iteration. It has to be noted that replacement within the cGA consists of deciding *who* and *when* the individual will be inserted in the new population. In this perspective, theoretically, all replacement strategies of the classical GA are applicable to the cGA, such as the (μ, λ), $(\mu + \lambda)$, $(1 + 1)$ replacement strategies. However, for deciding when the new individual will be inserted in the population, two main strategies exist: *synchronous* and *asynchronous*. The synchronous replacement stands in placing the newly-produced individual in an auxiliary population and once all individuals are browsed, the auxiliary population replaces the old one. On the other hand, the asynchronous replacement strategy stands in replacing directly in the current population the individual being processed by the newly-created one.

In order to accomplish one iteration of the cGA, the whole process has to be repeated until all the nodes (i.e. individuals) of the grid are browsed. Again, it is worth mentioning that the grid nodes are browsed in a very specific sequence. After, the best individual $\vec{G}$ is updated. Then, the classical cGA iteration is repeated until a given termination criterion is reached.

2.3. Adaptation in evolutionary algorithms

The particularity of each optimisation problem stands in some specific properties that make it different from another problem. Even for the same problem, each instance has its own characteristics that distinguish it from another instance. We can cite, for example, multi-modality, epistasis and deceptiveness (Rothlauf, 2011). Thus, most of today's metaheuristics are problem or instance-dependent. In fact, their efficiency highly depends on the fine-grained tuning of their parameters. The latter are tuned through long and exhaustive phases in order to better suit the problem/instance tackled. So, any variation in the problem/instance properties might change the efficiency of the algorithm. Moreover, such tuning needs advanced knowledge on both the algorithm used and the problem tackled. These facts limit the use of static algorithms to experts in pure abstract research instead of novice users in real-life environments (Alba & Dorronsoro, 2005; Dahi et al., 2016).

As a solution to this issue, dynamic algorithms appear as a good alternative to the static ones. Adaptation in EAs like in any other metaheuristic, is an attempt to make the algorithm's search process evolve in order to better suit the properties of the problem tackled. Technically speaking, it consists of designing mechanisms that allow evolving the value of some parameters without an external control (Dahi et al., 2016). Depending on the algorithm used and the problem tackled, many parameters can undergo such adaptation and many strategies can be used to do so. Indeed, many adaptation approaches have been proposed but all can be grouped in three schemes (Alba & Dorronsoro, 2005): deterministic, adaptive and self-adaptive.

Deterministic adaptation occurs whenever the value of a given parameter is tuned using some predefined formula. It is worth to mention that in this scheme, no information from the algorithm or any other external agent are exchanged in order to rule the adaptation. Unlike the first scheme, the *adaptive* one takes place if some feedback is got from the algorithm during the search process. Finally, the *self-adaptive* scheme consists of evolving the parameter's value along with the problem solution. In fact, the variables of the problem as well as the value of the parameter being adapted are all encoded as a solution that evolves using the algorithm operators. Although, it is to be noted that the deterministic scheme is the one studied in this work.

However, even if dynamic adaptation seems to be the ideal alternative to the classical hand-tuning, studies such as in Alba and Dorronsoro (2005) showed that most of dynamic algorithms proposed in the literature involve computationally-heavy policies that rule adaptation, and additional hidden parameters that make the use of hand-tuned algorithms more beneficial than dynamic ones. So, it is clear that low-complexity and computational lightness are the principal features of any dynamic algorithm in order to be useful.

3. Stop-and-start cGA for the RCP

In this section, we present a Stop-and-Start cellular Genetic Algorithm (2SA-cGA) for solving the MMP. First, we start by explaining solution encoding and the objective function of the mobility management problem. Then, we give more insight into the phases of the proposed approach.

3.1. Problem formulation

In this section, we introduce both solution representation and the mathematical formulation introduced by Haćc and Zhou (1997) in their modelling of the mobility management problem.

3.1.1. Solution encoding

A solution for the reporting cell problem is represented by a binary-coded vector $\vec{X}$. Each solution vector $\vec{X} = \{x_1, \ldots, x_D\}$, represents a potential configuration of the network, where D is the size of the network. Each element $x_j \in \vec{X}$, where $j = 1, \ldots, D$, represents the state (i.e. reporting or not) of the j^{th} cell in the network. If $x_j = 1$, the j^{th} cell is considered as reporting, while when $x_j = 0$, it is considered as a non-reporting cell. Fig. 3 illustrates the canonical solution representation for the RCP.

$\vec{X}$	x_1	x_2	x_3	x_4	$\cdots$	x_D
	$1st$ Cell	$2nd$ Cell	$3rd$ Cell	$4th$ Cell		Dth Cell

Fig. 3. Solution representation for the RCP.

Fig. 4 represents a possible solution for the RCP. It corresponds to a configuration for a network with six cells, where the 1^{st} and 4^{th} cells are considered as reporting, while the remaining ones are non-reporting cells.

$\vec{X}$	1	0	0	1	0	0
	$1st$ Cell	$2nd$ Cell	$3rd$ Cell	$4th$ Cell	$5th$ Cell	$6th$ Cell

Fig. 4. Example of solution representation for the RCP.

3.1.2. Fitness function

The RCP objective function proposed by Haćc and Zhou (1997) is defined in Eq. (1).

$$\min_{\vec{X}=\{x_1,\ldots,x_D\}} Z\left(\vec{X}\right) = \beta * \sum_{i=1}^{W} L_i + \sum_{j=1}^{D} P_j * V_j \tag{1}$$

The RCP objective function, Z, is used to assess the quality of a potential solution vector $\vec{X}$, where the variable L_i represents the number of location updates performed in the i^{th} reporting cell. The value W is the set of reporting cells in the network, while P_j is the number of paging transactions performed in the j^{th} cell, where D

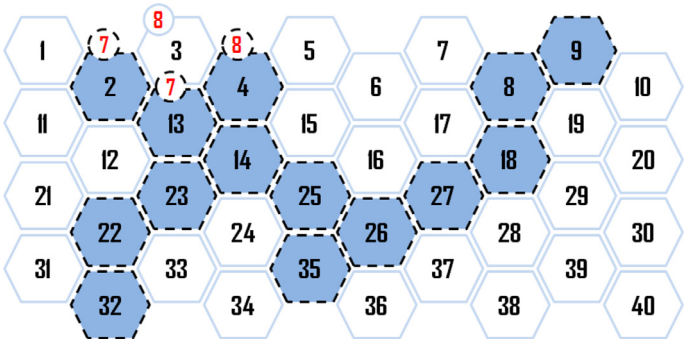

Fig. 5. Example of vicinity calculation.

is the number of cells in the network. Finally, V_j is the vicinity factor of the j^{th} cell. Since the location update is more important than the paging cost, the variable β is a weight used to balance the objective function. For reliability and comparison purposes with the state-of-the-art algorithms, we use the same value of β ($\beta = 10$) as the one used in all previous works treating this formulation of the reporting cell problem (Berrocal-Plaza et al., 2014d; 2015a; Berrocal-Plaza, Vega-Rodríguez, & Sánchez-Pérez, 2015b).

The vicinity of a cell is the maximum number of cells that have to be browsed in case an incoming call occurs in this cell. Concretely, the vicinity value of a reporting cell is the maximum number of non-reporting cells (including the reporting cell itself), reachable from this cell without entering another reporting cell. A non-reporting cell can be reachable from different reporting cells. So, the vicinity of a non-reporting cell can be defined as the maximum vicinity value of the reporting cells from which we can reach this non-reporting cell. It is also to be noted that the cells vicinity values are not static neither universal and therefore depend on the network configuration at hand and computed on the fly.

Fig. 5 represents a potential configuration of a network with 40 cells, where fourteen of them are reporting and the remaining are non-reporting. The vicinity of the 3^{rd} non-reporting cell is the maximum vicinity value of its neighbouring 2^{nd}, 13^{th} and 4^{th} reporting cells. The vicinity values of these cells are 7, 7 and 8, respectively. Thus, the vicinity value of the 3^{rd} non-reporting cell is 8.

3.2. The search process

In this section, we present the newly-proposed Stop-and-Start Adaptive cellular Genetic Algorithm (2SA-cGA). Like the classical cGA does, the 2SA-cGA takes as an input a population of N individuals and makes them evolve toward a fitter state by applying in each iteration a series of operators. Each individual $\vec{X} = \{x_1, x_2, \ldots, x_D\}$ represents a potential solution to the mobility management problem for a network of D cells, where each element x_j / $j = 1, \ldots, D$, corresponds to the state (i.e. reporting or non-reporting) of the j^{th} cell. Fig. 6 illustrates the canonical representation of the 2SA-cGA's population for the mobility management problem.

However, unlike the unstructured GA, each individual in the 2SA-cGA represents a node on a grid with length H and V width (see Fig. 2). After population initialisation, a classical iteration of the 2SA-cGA consists of applying sequentially on each individual in the grid: selection, crossover, mutation, evaluation and finally replacement. Once the whole grid browsed, two of the main features of the proposed 2SA-cGA are applied: adaptation of the mutation probability and population's restart.

The pseudo-code of Algorithm 2 and flowchart in Fig. 7 give the general framework of the proposed approach, and in the following sections we give more insight into each of the 2SA-cGA's phases. It

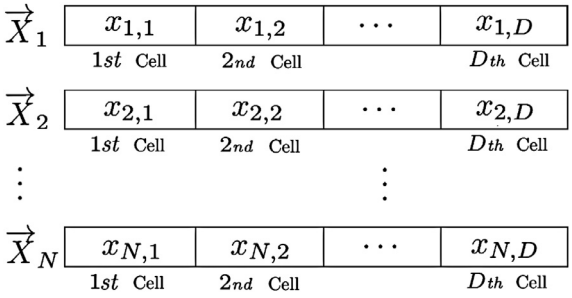

Fig. 6. Population representation for the MMP.

Algorithm 2 2SA-cGA pseudo-code.

```
1: while stop criterion is not reached do
2:   for i=1:V do
3:     for j=1:H do
4:       % ———————— Form the (i, j)th couple ———————— %
5:       Select the individual at position (i, j) as 1st parent.
6:       Select 2nd parent by performing a binary tournament on the
         NEWS neighbourhood of 1st parent.
7:       % ———————————— Crossover ———————————— %
8:       Apply DPX1-best crossover on (i, j)th couple.
9:       % ———————————— Mutation ———————————— %
10:      Apply bit-flip mutation on produced offspring.
11:      % ———————————— Evaluation ———————————— %
12:      Compute the fitness value of the offspring.
13:      %———————— Replacement: Synchronous (a) ———————— %
14:      Compare parent at position (i, j) to the newly-produced
         offspring and copy the best to the auxiliary population.
15:    end for
16:  end for
17:  %———————— Replacement: Synchronous (b) ————————- %
18:  Consider the auxiliary population as the new one.
19:  %———————————— Stop and Restart ———————————— %
20:  if restart condition is checked then
21:    Restart the population.
22:  end if
23:  % ———————————— Adaptation ———————————— %
24:  Adapt the mutation probability P_m.
25: end while
```

is worth mentioning that the 2SA-cGA uses a Von Neumann neighbourhood and a toroidal grid (3D) (see Fig. 2).

3.3. Initialisation

The initialisation step consists of creating N binary individuals. Each element x_j of each individual $\vec{X}$ is initialised by generating a random number ω from a standard uniform distribution. Then, if $\omega \geq 0.5$, x_j will have the value 1, otherwise 0. Once the population is initialised, the quality of all individuals in it are evaluated by computing their fitness values (see Section 3.7). Then, the best individual $\vec{G}$ is updated.

3.4. Selection

For the $(i, j)^{th}$ individual being browsed, a couple is formed, where the $(i, j)^{th}$ individual is considered as the 1^{st} parent, while the 2^{nd} one is selected by performing a binary tournament on the Von Neumann neighbourhood of the $(i, j)^{th}$ individual.

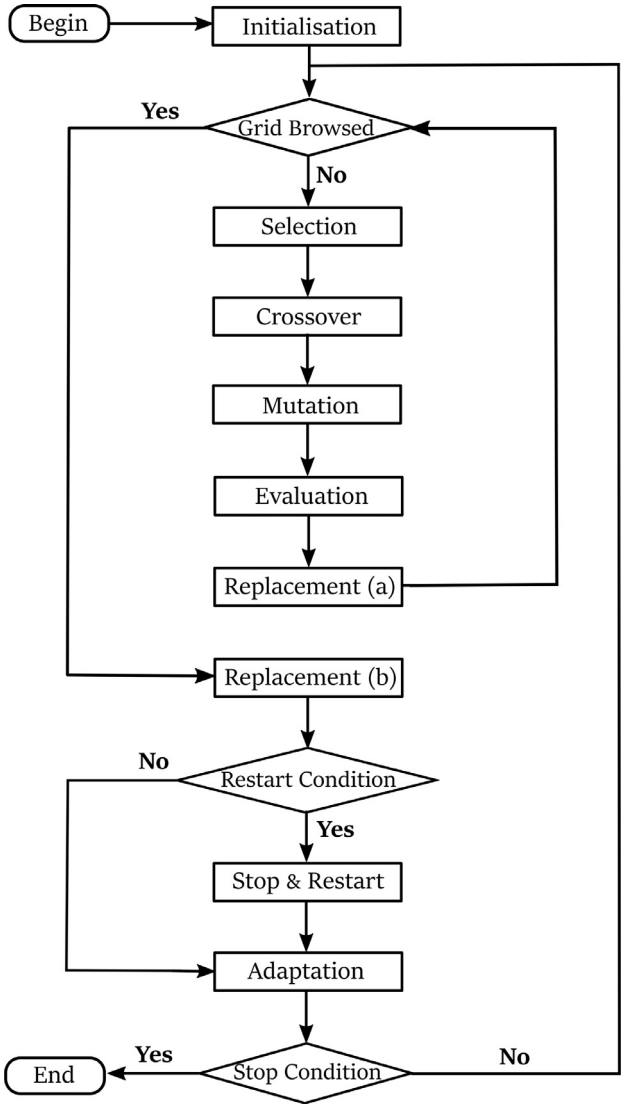

Fig. 7. The proposed 2SA-cGA.

3.5. Crossover

It is the first variation operator that the $(i, j)^{th}$ couple undergo. A random number ω is generated from a standard uniform distribution. Then, if $\omega \leq P_c$, the crossover is applied, otherwise the parents are considered as the new offspring. In our work we present a variant of the DPX1 crossover called DPX1-*best*. In a first step, the DPX1-*best* consists of applying a two-point crossover by generating two switching points ϕ_1 and ϕ_2 from the interval [1, D] according to a uniform distribution. Then, the parts delimited by both the heads, ϕ_1, ϕ_2 and the tails of both parents of the $(i, j)^{th}$ couple are swapped. This, will result in two new offspring.

As a second step of the DPX1-*best*, the offspring that is closer to the best individual $\vec{G}$ in terms of Hamming distance is kept.

3.6. Mutation

After crossover, mutation is performed on the produced offspring. For each gene, a random number ω is generated from a standard uniform distribution. Then, if $\omega \leq P_m$, mutation is applied on the corresponding element, otherwise nothing is done. The mutation type used in the proposed 2SA-cGA is a bit-flip mutation. It consists of inverting the value of the corresponding gene to the opposite state (1 or 0).

3.7. Evaluation

After applying both mutation and the crossover, the quality of the produced offspring is assessed by computing its fitness value using Eq. (1) defining the objective function of the MMP.

3.8. Replacement

The proposed 2SA-cGA uses a synchronous policy of replacement, in which both the $(i, j)^{th}$ individual being processed and the newly-produced offspring are compared in terms of fitness. The newly-produced individual is copied to an auxiliary population if it has a fitness better or equal to the one of the previous individual.

The end of the replacement phase marks the end of a canonical cycle of treating an individual in the 2SA-cGA grid. Then, the whole process is repeated on the following individual in the grid. When the final individual is processed, it marks the end of the classical iteration of the proposed 2SA-cGA. At this step, the auxiliary population replaces the old one. Then, the best individual $\vec{G}$ is updated.

3.9. Stop and restart

After each iteration of the 2SA-cGA, a *stop and restart* phase might or not be performed. This phase is applied if two conditions are fulfilled. The first one is that the size of the individual is *inferior or equal to* 64 bits (i.e. $D \leq$ 64). The second condition is that the current iteration is equal to (3T/5), where T is the maximum number of iterations that the 2SA-cGA is allowed to perform. These two restart conditions are formulated on the basis of several fine-grained offline experiments. In addition, it is to be noted that if the stop and restart phase is performed once, it will never be performed again.

Concretely, this phase consists of reinitialising N - 1 individuals of the current population. The latter are set as identical copies of the best individual $\vec{G}$. Then, a given percentage P_α of randomly-chosen bits in each of these individuals is altered by switching their values to either 1 or 0.

3.10. Adaptation

In this phase, we adapt the value of the mutation probability using the equation proposed by Bäck and Schütz (1996). It is defined using Eq. (2), where P_m is the mutation probability value at iteration t and T is the maximum number of iterations that the algorithm is allowed to perform.

$$P_m(t) = \left(2 + \frac{D-2}{T-1} * t\right)^{-1} \quad (2)$$

Fig. 8 represents the P_m value evolution when using the above-introduced adaptation strategy. It is to be noted that Fig. 8 is representative of the evolution behaviour resulting when tackling each problem instance, but it does not stand for all the networks. This is due to the fact that the bounds of the interval where the P_m value evolve change from one size of problem to another.

The choice was made for this adaptation strategy on the basis of previous offline fine-grained experiments. Also, unlike most of the adaptation strategies proposed in the literature, this adaptation scheme has the particularity of being simple and computationally light and does not add any extra hidden parameters that need to be tuned. In fact, no extra information is exchanged or necessary to compute the new value of P_m. Furthermore, it removes the need for setting an initial value of the mutation probability or the need

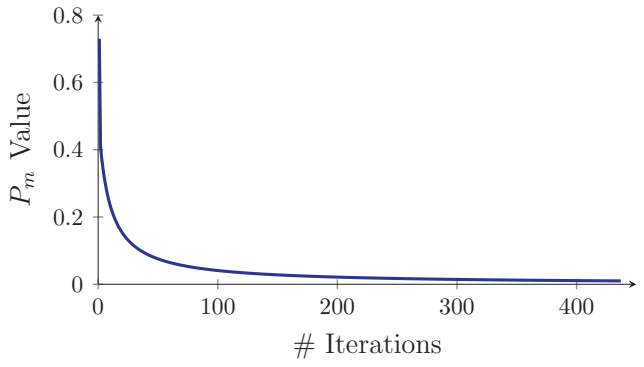

Fig. 8. P_m value evolution using Bäck and Schütz adaptation.

for any further parameters that tunes this adaptation strategy. As a matter of fact, $P_m(1)$ can be computed by setting $t = 1$ in Eq. (2).

4. Experimental design, results and analysis

In this section, we start by giving a description of the experiments performed in order to investigate the performances of the proposed approach. Then, we present the obtained results and their discussion.

4.1. Experimental design

Our experiments have been run on a cluster of 19 heterogeneous machines with a total of 94 cores, where 16 machines had 3 cores, 1 machine was with 8 cores and 2 machines were with 48 cores each. The cluster was managed by the framework HTCondor[2]. All machines were running under a Linux OS (Ubuntu 14 distribution). Implementation was done using Matlab R2014a.

In order to investigate the efficiency, scalability and robustness of the proposed approach, twenty-five differently-sized instances have been used in order to illustrate several complexities, sizes and characteristics. Also, the proposal is compared against twenty-six state-of-the-art algorithms that have been designed to solve the MMP. The instances are grouped in three sets, the first set contains twelve networks generated on the basis of realistic patterns. They consist of three networks of 4×4, 6×6, 8×8 and 10×10 cells. They were used in Berrocal-Plaza, Vega-Rodríguez, and Sánchez-Pérez (2014a); Berrocal-Plaza, Vega-Rodriguez, and Sanchez-Perez (2014b, 2014c, 2014d, 2015a); Berrocal-Plaza et al. (2015b); Dahi et al. (2016) and are available in[3].

The second set of instances contains six networks of sizes 4×4, 6×6, 8×8, 7×9, 9×11 and 19 cells. Like the first set, the instances of this second set have been generated on the basis of realistic patterns. They were used in Almeida-Luz, Vega-Rodriguez, Gomez-Pulido, and Sanchez-Perez (2011); Baburaj, Phil, and Alagarsamy (2010); Kim, Kim, Byeon, and Taheri (2012); Mehta and Swadas (2009b); Patra and Udgata (2011); Subrata and Zomaya (2003a,b); Taheri and Zomaya (2008); Vekariya and Swadas (2015); Wang, Xiong, and Huang (2014); Wang and Si (2010); Wang, Wang, Pan, and Zuo (2009) and are available in Mehta and Swadas (2009b); Subrata and Zomaya (2003a); Taheri and Zomaya (2008).

The third set of instances are new bigger instances we propose in our work. The instances are based on six months of communication record (from 31/08/2009 to 27/02/2010) of the German politician Malte Spitz. The data are stored by the Deutsche Telekom and available in[4]. This set contains six networks of sizes: 12×12, 14×14, 16×16, 18×18, 20×20 and 30×30 cells. Besides, we use another instance of size 30×30 cells. The latter has been studied previously in González-Álvarez et al. (2012); Rubio-Largo et al. (2010) and available in.[5]

Each network of each set consists of D pairs of attributes (L_i,P_i), where $i = 1, \ldots, D$ and D is the size of the network. L_i and P_i represent the location update and the paging costs associated to the i^{th} cell, respectively.

For a fair comparison, we use the same experimental parameters used in all the state-of-the-art works treating the mobility management problem. Thus, the experiments are conducted until reaching a termination criterion of 175,000 fitness evaluations. We use a population of 400 individuals. The algorithm is run about 437 iterations, over 30 independent runs. Many results are reported such as the best and the worst fitness values and the mean and deviation of the fitness values over the whole set of runs. It is worth to note that in the following sections, the terms "*instance*" and "*network*" refer to the same meaning.

The crossover probability P_c is set to 1, while the cataclysmic probability P_α is set to 50%.

4.2. Numerical results and discussion

In this section, we present the results obtained when tackling the instances of the first, second and third sets.

4.2.1. First set of instances

Some works that have been conducted on this set of networks rank the algorithms on the basis of the best solution achieved over 30 executions. Thus, they do not report the mean or the deviation of the fitness values over the whole set of executions; while other works rank the algorithms on the basis of the mean of their fitness over the whole set of executions.

We include here what is publicly available in the literature and try to include a lot of information on our own proposal so as to ease future comparisons versus our results.

Table 2 represents the comparison between the proposed approach and the state-of-the-art algorithms whose best solution is available in the literature. The metric "# *Optima*" represents the number of optima (known so far in the literature) reached by each algorithm, whereas Tables 3 and 4 compare our approach against the state-of-the-art algorithms whose mean and deviation are available. On the basis of the metric "*Mean*" the best results are highlighted in bold. For a fair comparison, the deviation reported in Tables 3 and 4 is computed like it was done in works from where the algorithms taken as comparison basis were proposed. It is calculated using Eq. (3), where "*Mean*" is the average fitness value and "*Best*" is the best value achieved. Both are taken from Tables 3 and 4. It is also to be noted that the deviation is expressed in terms of percentage.

$$Dev\,(\%) = \left(\frac{Mean}{Best} - 1\right) * 100 \tag{3}$$

It is worth mentioning that in Tables 3–6, the symbol "-" is used whenever the corresponding value is not available in the literature. The metric "*Rank*" represents the order of algorithms on the basis of the mean of their fitnesses.

The proposed 2S-cGA is compared against NOMAD (Berrocal-Plaza, Vega-Rodriguez, & Sanchez-Perez, 2014c), CPLEX (Berrocal-Plaza et al., 2014c), GRASP (González-Álvarez et al., 2012), PBIL (González-Álvarez et al., 2012), GA (González-Álvarez et al., 2012),

[2] High Throughput Computing (HTC).
[3] Set 1: http://oplink.lcc.uma.es/problems/mmp.html.
[4] Set 3-A: http://crawdad.org/spitz/cellular/20110504.
[5] Set 3-B: http://arco.unex.es/rc/.%5C%5Credes%5C%5CTN-13-30x30.txt.

Table 2
Best solution achieved when tackling networks of sizes 4×4, 6×6, 8×8 and 10×10 cells.

Size	Network	NOMAD	CPLEX	GRASP	PBIL	GA	ABC	OI-GA	DE	HNN-BD	SPEA2	GPSO	SS	NSGA-II	2SA-cGA
4×4	1	**98,535**	**98,535**	**98,535**	**98,535**	**98,535**	**98,535**	**98,535**	**98,535**	**98,535**	**98,535**	**98,535**	**98,535**	**98,535**	**98,535**
	2	**97,156**	**97,156**	**97,156**	97,262	**97,156**	**97,156**	**97,156**	**97,156**	**97,156**	**97,156**	**97,156**	**97,156**	**97,156**	**97,156**
	3	**95,038**	**95,038**	**95,038**	**95,038**	**95,038**	**95,038**	**95,038**	**95,038**	**95,038**	**95,038**	**95,038**	**95,038**	**95,038**	**95,038**
6×6	1	177,647	181,677	175,072	173,804	176,032	175,041	**173,701**	**173,701**	**173,701**	**173,701**	**173,701**	**173,701**	**173,701**	**173,701**
	2	200,069	200,990	184,142	**182,331**	**182,331**	**182,331**	**182,331**	**182,331**	**182,331**	**182,331**	**182,331**	**182,331**	**182,331**	**182,331**
	3	175,620	186,481	175,500	175,564	176,994	176,148	**174,519**	**174,519**	**174,519**	**174,519**	**174,519**	**174,519**	**174,519**	**174,519**
8×8	1	316,328	375,103	316,373	313,336	312,395	312,558	311,171	308,401	308,929	308,702	308,401	**307,695**	308,702	**307,695**
	2	301,833	351,505	299,616	292,667	294,391	297,560	**287,149**	**287,149**	**287,149**	**287,149**	**287,149**	**287,149**	**287,149**	**287,149**
	3	271,637	407,457	270,830	265,853	265,792	268,366	**264,204**	**264,204**	**264,204**	**264,204**	**264,204**	**264,204**	**264,204**	**264,204**
10×10	1	404,447	514,504	402,376	390,489	391,025	396,456	387,104	386,681	386,351	386,721	385,972	**385,927**	**385,927**	**385,927**
	2	371,091	468,118	369,979	362,574	364,354	366,568	359,623	358,167	358,167	358,392	359,191	357,714	**357,368**	**357,368**
	3	382,180	514,514	383,321	378,033	378,926	381,725	372,938	371,829	**370,868**	**370,868**	**370,868**	**370,868**	**370,868**	**370,868**
# Optima		3	3	3	4	4	7	8	9	9	9	9	11	11	**12**

Table 3
Results achieved when tackling networks 1, 2 and 3 of sizes 4×4 and 6×6 cells.

		4×4 cells					6×6 cells				
Network	Algorithm	Best	Worst	Mean	Dev (%)	Rank	Best	Worst	Mean	Dev (%)	Rank
1	NSGA-II	98,535	–	**98,535.00**	0.00	1	173,701	–	**173,701.00**	0.00	1
	SPEA2	98,535	–	**98,535.00**	0.00	1	173,701	–	**173,701.00**	0.00	1
	OI-GA	98,535	98,535	**98,535.00**	–	1	173,701	173,701	**173,701.00**	–	1
	HNN-BD	98,535	–	98,627.00	0.09	2	173,701	–	174,690.00	0.56	3
	GPSO	98,535	–	**98,535.00**	0.00	1	173,701	–	174,090.00	0.22	2
	CPLEX	98,535	–	**98,535.00**	0.00	1	181,677	–	181,677.00	0.00	4
	NOMAD	98,535	–	100,366.00	2.54	3	177,647	–	189,263.00	4.44	5
	2SA-cGA	98,535	98,535	**98,535.00**	0.00	1	173,701	173,701	**173,701.00**	0.00	1
2	NSGA-II	97,156	–	**97,156.00**	0.00	1	182,331	–	**182,331.00**	0.00	1
	SPEA2	97,156	–	**97,156.00**	0.00	1	182,331	–	**182,331.00**	0.00	1
	OI-GA	97,156	97,156	**97,156.00**	–	1	182,331	182,331	**182,331.00**	0.00	1
	HNN-BD	97,156	–	97,655.00	0.51	2	182,331	–	182,430.00	0.05	2
	GPSO	97,156	–	**97,156.00**	0.00	1	182,331	–	**182,331.00**	0.00	1
	CPLEX	97,156	–	**97,156.00**	0.00	1	200,990	–	200,990.00	0.00	3
	NOMAD	97,156	–	100,065.00	2.92	3	200,069	–	221,889.00	6.27	4
	2SA-cGA	97,156	97,156	**97,156.00**	0.00	1	182,331	182,331	**182,331.00**	0.00	1
3	NSGA-II	95,038	–	**95,038.00**	0.00	1	174,519	–	174,605.00	0.13	2
	SPEA2	95,038	–	**95,038.00**	0.00	1	174,519	–	174,711.00	0.19	3
	OI-GA	95,038	95,038	**95,038.00**	–	1	174,519	174,519	**174,519.00**	–	1
	HNN-BD	95,038	–	95,751.00	0.75	2	174,519	–	176,050.00	0.87	5
	GPSO	95,038	–	**95,038.00**	0.00	1	174,519	–	175,080.00	0.32	4
	CPLEX	95,038	–	**95,038.00**	0.00	1	186,481	–	186,481.00	0.00	7
	NOMAD	95,038	–	100,131.00	4.30	3	175,620	–	182,289.00	2.66	6
	2SA-cGA	95,038	95,038	**95,038.00**	0.00	1	174,519	174,519	**174,519.00**	0.00	1

ABC (González-Álvarez et al., 2012), OI-GA (Dahi et al., 2016), DE (Almeida-Luz et al., 2011), HNN-BD (Alba, García-Nieto, Taheri, & Zomaya, 2008), SPEA2 (Berrocal-Plaza et al., 2014b), GPSO (Alba et al., 2008), SS (Almeida-Luz, Vega-Rodriguez, Gomez-Pulido, & Sanchez-Perez, 2010a), NSGA-II (Berrocal-Plaza et al., 2015a). However, it is worth to mention that the results of the GPSO reported in Tables 2–4 are the ones published in Alba et al. (2008) and are of 10 executions. But, since previous comparisons against state-of-the-art solvers run for 30 executions (Almeida-Luz et al., 2010a; 2010b; Berrocal-Plaza et al., 2014c; 2014d) or 31 executions (Berrocal-Plaza et al., 2014a; Berrocal-Plaza et al., 2014b; 2015a) have been already done in some works, we assume it is feasible in our work also.

On the basis of the results of Table 2, one can see that our proposed 2SA-cGA outperformed all state-of-the-art algorithms. It is to be noted also that to the best knowledge of the authors, the 2SA-cGA is the first algorithm in the literature that could reach *all* best solutions for the twelve networks of sizes 4×4, 6×6, 8×8 and 10×10 cells.

With regards to results of Tables 3 and 4, one can first note that for 7 out of 12 networks (Networks 1, 2 and 3 of sizes 4×4 and 6×6 cells and Network 2 of size 8×8 cells) our proposed 2SA-cGA could achieve results as good as the ones achieved by state-of-the-art algorithms. However, the interesting thing to notice here is that in each of those instances the algorithms giving the best results are not the same. Indeed, there is no algorithm obtaining the best

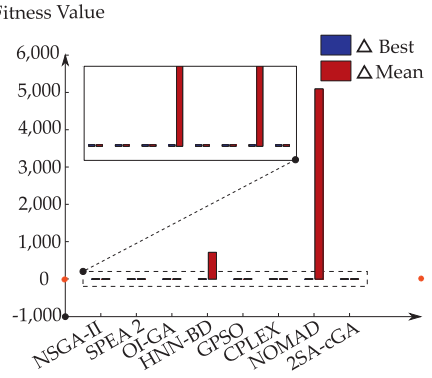

Fig. 9. Network 3 of 4×4 cells.

Table 4
Results achieved when tackling networks 1, 2 and 3 of sizes 8×8 and 10×10 cells.

Network	Algorithm	8×8 cells					10×10 cells				
		Best	Worst	Mean	Dev (%)	Rank	Best	Worst	Mean	Dev (%)	Rank
1	NSGA-II	308,702	–	308,859.00	0.05	2	385,927	–	387,416.00	0.20	2
	SPEA2	308,702	–	**308,822.00**	0.06	1	386,721	–	387,764.00	0.22	3
	OI-GA	311,171	312,925	312,308.38	–	6	387,104	394,176	390,531.47	–	6
	HNN-BD	308,929	–	311,351.00	0.78	5	386,351	–	387,820.00	0.38	4
	GPSO	308,401	–	310,062.00	0.53	3	385,972	–	387,825.00	0.48	5
	CPLEX	375,103	–	375,103.00	0.00	8	514,504	–	514,504.00	0.00	8
	NOMAD	316,328	–	326,008.00	2.19	7	404,447	–	415,740.00	1.82	7
	2SA-cGA	307,695	312,792	310,571.87	0.95	4	385,927	388,954	**387,148.03**	0.32	1
2	NSGA-II	287,149	–	**287,149.00**	0.00	1	357,368	–	**358,777.00**	0.16	1
	SPEA2	287,149	–	**287,149.00**	0.00	1	358,392	–	359,077.00	0.12	4
	OI-GA	287,149	289,771	287,236.41	–	2	359,623	370,021	362,320.88	–	6
	HNN-BD	287,149	–	**287,149.00**	0.00	1	358,167	–	359,036.00	0.24	2
	GPSO	287,149	–	287,805.00	0.22	3	359,191	–	359,928.00	0.20	5
	CPLEX	351,505	–	351,505.00	0.00	5	468,118	–	468,118.00	0.00	8
	NOMAD	301,833	–	312,497.00	2.59	4	371,091	–	379,725.00	1.31	7
	2SA-cGA	287,149	287,149	**287,149.00**	0.00	1	357,368	361,299	359,050.73	0.47	3
3	NSGA-II	264,204	–	264,396.00	0.09	3	370,868	–	371,349.00	0.15	2
	SPEA2	264,204	–	264,279.00	0.07	2	370,868	–	**371,331.00**	0.10	1
	OI-GA	264,204	265,324	264,533.16	–	5	372,938	382,155	376,900.28	–	6
	HNN-BD	264,204	–	264,695.00	0.18	6	370,868	–	374,205.00	0.89	5
	GPSO	264,204	–	264,475.00	0.10	4	370,868	–	373,722.00	0.76	4
	CPLEX	407,457	–	407,457.00	0.00	8	514,514	–	514,514.00	0.00	8
	NOMAD	271,637	–	289,309.00	3.09	7	382,180	–	391,627.00	0.81	7
	2SA-cGA	264,204	264,353	**264,257.47**	0.02	1	370,868	378,121	372,854.10	0.54	3

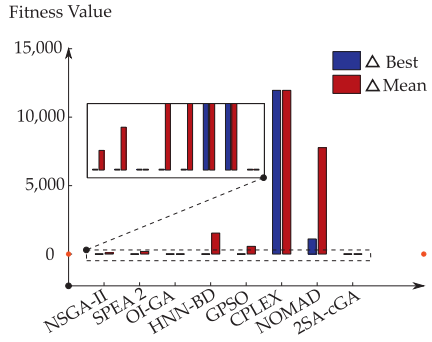

Fig. 10. Network 3 of 6×6 cells.

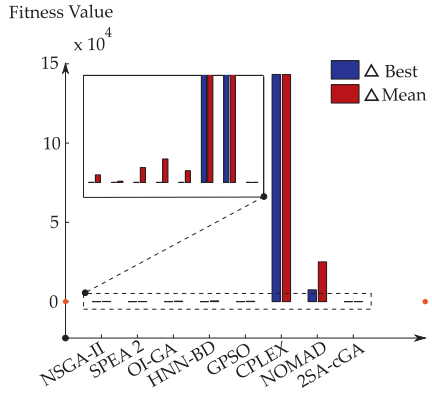

Fig. 11. Network 3 of 8×8 cells.

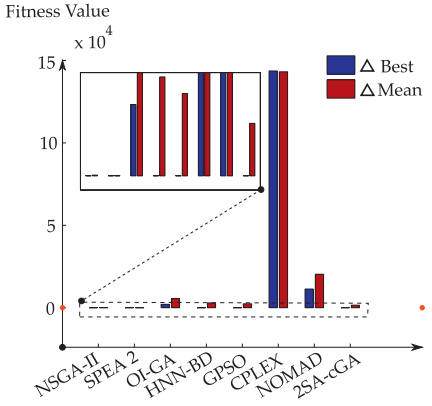

Fig. 12. Network 3 of 10×10 cells.

results for all those instances, while the 2SA-cGA is always giving results as good as the ones obtained by the best algorithms.

In addition, on the basis of the results of Tables 3 and 4, one can see that the proposed 2SA-cGA could outperform all state-of-the-art algorithms in 2 out of 12 instances (Network 3 of size 8×8 cells and Network 1 of size 10×10 cells). However, it is to be noted that the proposed approach is slightly outperformed by the SPEA2 in 2 out of 12 instances (Network 1 of size 8×8 cells and Network 3 of size 10×10 cells) and the NSGA-II in Network 2 of size 10×10 cells.

Figs. 9–12 are bar-based representations of the $\Delta best$ and $\Delta mean$ fitness values found by each algorithm when tackling network 3 of sizes 4×4, 6×6, 8×8 and 10×10 cells. The metrics $\Delta best$ or $\Delta mean$ are the difference between the best (or mean) fitness value obtained by each algorithm and the best (or mean) obtained by the best algorithm. Therefore, the best algorithms in Figs. 9–12 are those having the *smallest* bars. As it can be seen, Figs. 9–12 support the findings of Tables 2–4. It is to be noted also that further graphical results can be found at[6].

4.2.2. Second set of instances

Like for the first set of instances, some works that treated the second set of instances also ranked the algorithms on the basis of the best solution achieved over 30 executions and no other met-

[6] Appendix URL: https://tinyurl.com/Appendix-Link.

Table 5
Best solution achieved when tackling networks of sizes 4×4, 6×6, 8×8, 7×9, 9×11 and 19 cells.

Size	HNN-BD	GA	ACO	DE	TS	SA	GA-SA	SS	2SA-cGA
4×4 cells	–	92,833	92,833	92,833	92,833	–	92,882	92,833	**85,166**
6×6 cells	–	229,556	211,291	211,278	**211,278**	–	**211,272**	211,278	214,313
8×8 cells	–	436,283	436,886	436,269	436,283	–	**436,029**	436,269	458,469
7×9 cells	123,474	–	–	120,904	–	–	–	**120,052**	123,474
9×11 cells	243,414	–	–	243,957	–	–	–	**242,914**	242,988
19 cells	–	–	–	–	–	**5,239**	–	–	**5,239**
# Optima	0 out of 2	0 out of 3	0 out of 3	0 out of 5	1 out of 3	1 out of 1	2 out of 3	2 out of 5	2 out of 6

Table 6
Results achieved when tackling networks of sizes 4×4 and 6×6 cells.

Size	Algorithm	Best	Worst	Mean	Dev (%)	Rank	Size	Best	Worst	Mean	Dev (%)	Rank
4×4	GA	12.252	12.373	12.253	0.006	3	6×6	11.471	12.030	11.511	0.343	5
	TS	12.252	12.252	12.252	0.000	2		11.471	11.471	11.471	0.000	3
	ACO	12.252	12.252	12.252	0.000	2		11.471	11.471	11.471	0.000	3
	DE	–	–	–	–	–		11.471	11.471	11.471	0.000	3
	DEL	–	–	–	–	–		11.471	11.471	11.471	0.000	3
	IDE	–	–	–	–	–		11.471	11.471	11.471	0.000	3
	GA-PSO	–	–	–	–	–		–	–	–	–	–
	GSO	12.252	12.252	12.252	0.000	2		11.426	11.471	11.456	0.853	2
	DGSO	12.252	12.252	12.252	0.000	2		11.426	11.471	**11.432**	0.676	1
	BPSO	–	–	–	–	–		11.471	11.471	11.471	0.000	3
	PBVM	12.252	12.273	12.255	0.008	4		11.471	11.573	11.474	0.014	4
	2SA-cGA	11.234	11.234	**11.234**	0.000	1		11.636	11.636	11.636	0.000	6

Table 7
Results achieved when tackling network of size 8×8 cells.

Size	Algorithm	Best	Worst	Mean	Dev (%)	Rank
8×8	GA	13.782	14.671	14.005	1.619	10
	TS	13.782	13.999	13.791	0.071	4
	ACO	13.782	14.007	13.860	0.569	9
	DE	13.782	13.923	13.798	0.116	6
	DEL	13.782	13.892	13.784	0.014	3
	IDE	13.782	13.782	13.782	0.000	2
	GA-PSO	13.782	13.947	13.829	0.000	8
	GSO	13.782	14.102	13.791	0.497	4
	DGSO	13.782	13.883	**13.780**	0.037	1
	BPSO	13.782	13.893	13.793	0.900	5
	PBVM	13.782	14.042	13.809	0.045	7
	2SA-cGA	14.483	14.495	14.486	0.0187	11

Table 8
Results achieved when tackling networks of sizes 7×9 and 9×11 cells.

Size	Algorithm	Best	Worst	Mean	STD	Rank
7×9	BPSO	34.538	35.873	34.576	0.142	2
	2SA-cGA	34.538	34.681	**34.549**	0.036	1
9×11	BPSO	42.969	44.633	43.423	0.346	2
	2SA-cGA	42.969	43.515	**43.171**	0.145	1

rics such as the mean and the deviation of the fitness values were provided. On the other hand, other works ranked the algorithms on the basis of the mean of the fitness values achieved through 30 executions.

Under the light of these facts, Table 5 groups the results when comparing our proposed approach against all state-of-the-art algorithms whose the best solution is available in the literature. The metric "# *Optima*" represents the number of optima (known so far in the literature) reached by each one of the algorithms.

Tables 6–8 regroup the results when comparing our proposed approach against all state-of-the-art solvers whose the mean of fitness values is available. It is worth mentioning that the works that rank algorithms on the basis of the mean of fitness values, used a slightly modified fitness function called *cost per call arrival*. The latter is defined using Eq. (4), where $Z(\vec{X})$ represents the objective function defined by Eq. (1), D is the number of cells in the network and P_i is the paging cost associated to the i^{th} cell. It is also worth mentioning that in Tables 6–8, the symbol "-" is used whenever the corresponding value is not available in the literature.

$$\min_{\vec{X}=\{x_1,\ldots,x_D\}} \frac{Z\left(\vec{X}\right)}{\sum_{i=1}^{D} P_i} \tag{4}$$

The proposed 2SA-cGA is compared against the HNN-BD (Taheri & Zomaya, 2008), GA (Subrata & Zomaya, 2003a), ACO (Subrata & Zomaya, 2003a), DE (Taheri & Zomaya, 2008), TS (Subrata & Zomaya, 2003a), SA (Mehta & Swadas, 2009b), GA-SA (Vekariya & Swadas, 2015), SS (Almeida-Luz et al., 2010a), DE (Wang et al., 2009), IDE (Wang et al., 2009), GA-PSO (Wang & Si, 2010), GSO (Wang et al., 2014), DGSO (Wang et al., 2014), BPSO (Kim et al., 2012), PBVM (Patra & Udgata, 2011).

Considering results of Table 5, one can observe that the proposed 2SA-cGA could outperform *all* state-of-the-art algorithms by reaching two optima out of six. In addition, it is to be noted that to the best knowledge of the authors, the 2SA-cGA is the only solver in the literature that reached a new optimum for network of size 4×4 cells.

On the basis of results of Tables 6–8, one can observe that in 3 out of 5 instances (networks of sizes 4×4, 7×9 and 9×11 cells) the proposed approach could outperform all state-of-the-art algorithms. On the other hand, the proposed 2SA-cGA is outperformed in 2 out of 5 instances (networks of sizes 6×6 and 8×8 cells).

4.2.3. Third set of instances

On the basis of the results of Tables 2–4, we compare the proposed 2SA-cGA against three of the most efficient state-of-the-art algorithms designed to solve the mobility management problem: the GPSO proposed in Alba et al. (2008), the OI-GA devised in Dahi et al. (2016) and the DE proposed in Almeida-Luz et al. (2011).

Table 9 shows the numerical results of this comparison. On the basis of the metric "*Mean*", the best results are highlighted in bold. Figs. 13–18 are the graphical representations of the $\Delta best$ and $\Delta mean$ fitness values found by each algorithm. The metrics $\Delta best$ or $\Delta mean$ are the difference between the best (or mean)

Table 9
Results achieved when tackling networks of sizes 12×12, 14×14, 16×16, 18×18, 20×20 and 30×30 cells.

Size	Algorithm	Best	Worst	Mean	Dev (%)	Rank	Size	Best	Worst	Mean	Dev (%)	Rank
12×12	2SA-cGA	14,767	14,780	**14,767.87**	0.01	1	14×14	17,308	17,395	**17,329.43**	0.12	1
	DE	14,767	14,806	14,775.40	0.06	2		17,308	17,358	17,333.40	0.15	2
	OI-GA	15,023	15,440	15,183.20	1.07	3		18,454	19,158	18,762.97	1.67	3
	GPSO	16,112	17,277	16,693.73	3.61	4		21,838	23,233	22,509.67	3.08	4
16×16	2SA-cGA	23,199	23,304	**23,239.60**	0.18	1	18×18	26,257	26,325	**26,278.33**	0.08	1
	DE	23,200	23,394	23,254.90	0.24	2		26,257	26,429	26,326.40	0.27	2
	OI-GA	26,010	28,562	27,310.27	5.00	3		32,794	35,747	34,343.17	4.72	3
	GPSO	33,479	38,070	35,599.10	6.33	4		43,084	47,587	45,605.73	5.85	4
20×20	2SA-cGA	32,303	32,593	**32,424.50**	0.38	1	30×30(1)	61,874	64,102	62,982.67	1.79	2
	DE	32,486	32,742	32,554.47	0.21	2		58,944	59,462	**59,121.00**	0.30	1
	OI-GA	42,779	46,139	44,537.10	4.11	3		117,342	130,824	121,977.20	3.95	3
	GPSO	54,510	62,318	59,233.03	8.66	4		153,693	184,784	171,554.73	11.62	4
30×30(2)	2SA-cGA	5,504,949	558,1954	**5,551,192.63**	0.84	1						
	DE	5,551,632	56,04,376	55,84,741.43	0.60	2						
	OI-GA	6,880,903	7,378,851	71,36,752.50	3.72	3						
	GPSO	8,374,708	10,052,437	93,41,891.47	11.55	4						

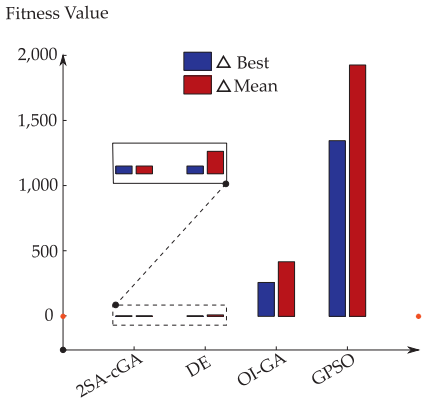

Fig. 13. 12×12 cells.

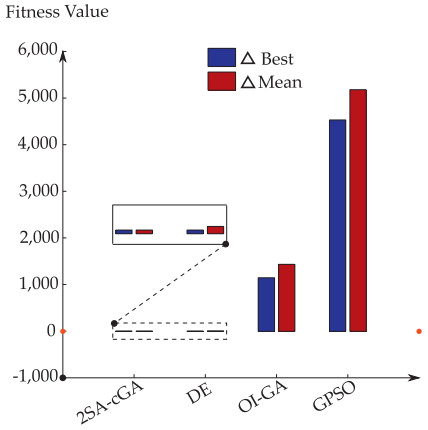

Fig. 14. 14×14 cells.

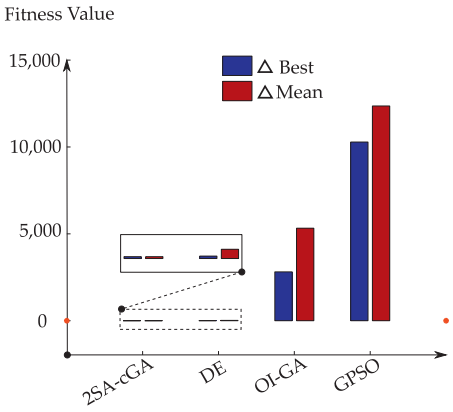

Fig. 15. 16×16 cells.

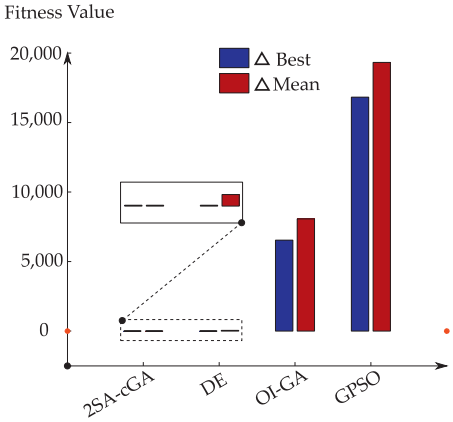

Fig. 16. 18×18 cells.

fitness value obtained by each algorithm and the best (or mean) obtained by the best algorithm. Therefore, the best algorithms in Figs. 13–18 are those having the *smallest* bars. For more insight, further graphical results can be found at[7]. It is to be noted also that the parameters of the GPSO, OI-GA and DE used within this experiment are the ones used within their original works. Also, the deviation is computed using Eq. (3) and expressed in percentage.

[7] Appendix URL: https://tinyurl.com/Appendix-Link.

It is worth to mention that Network 1 of size 30×30 cells is the one based on the communication traces of the German politician Malte Spitz, whereas Network 2 of the same size is the one previously-used in the literature (González-Álvarez et al., 2012; Rubio-Largo et al., 2010).

With regard to the results of Table 9, we can observe that the proposed 2SA-cGA outperforms the GPSO, OI-GA and DE in 6 out of 7 instances.

Figs. 19–24 illustrate the fitness value evolution for a randomly-chosen execution achieved by the 2SA-cGA, the OI-GA, GPSO and

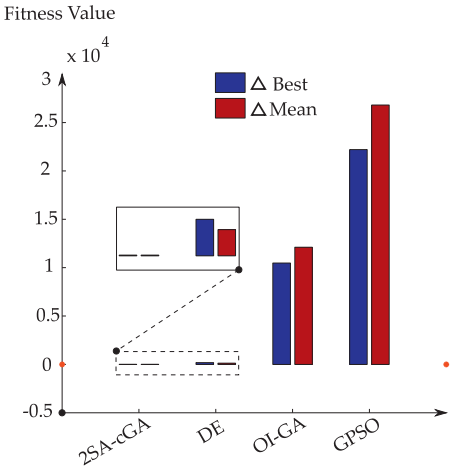

Fig. 17. 20×20 cells.

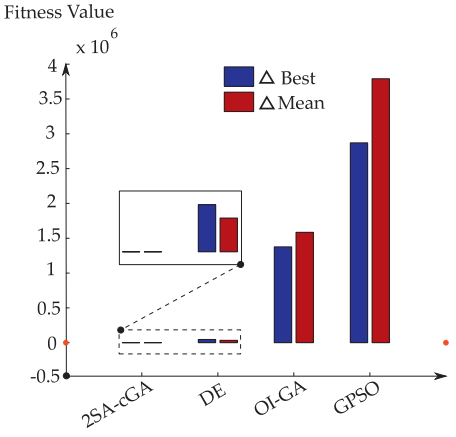

Fig. 18. 30×30 (2) cells.

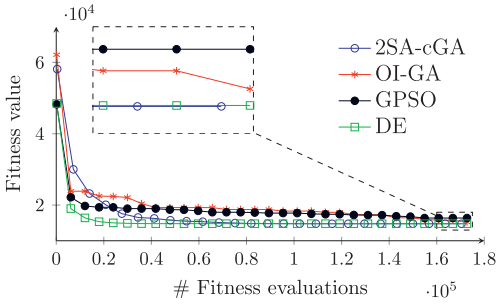

Fig. 19. 12×12 cells.

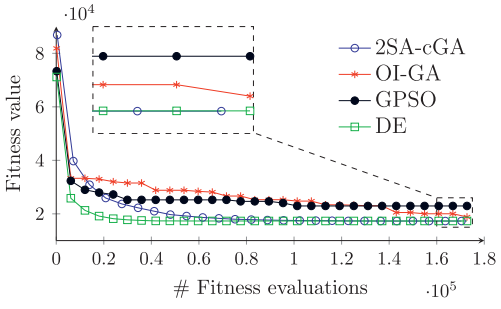

Fig. 20. 14×14 cells.

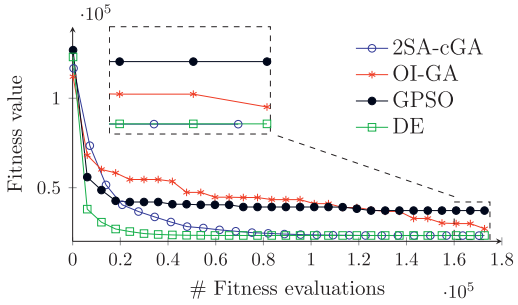

Fig. 21. 16×16 cells.

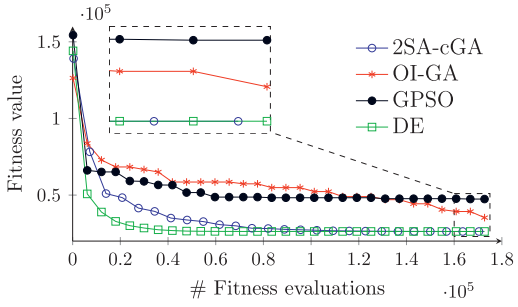

Fig. 22. 18×18 cells.

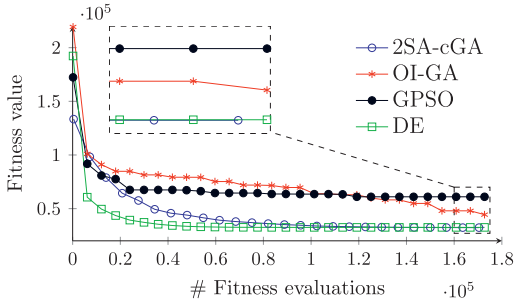

Fig. 23. 20×20 cells.

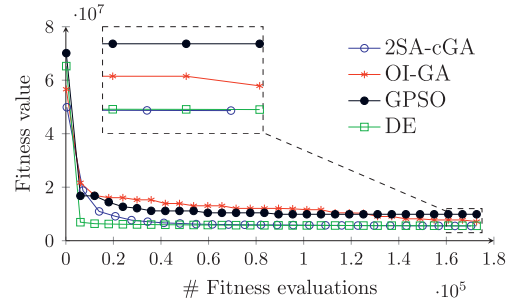

Fig. 24. 30×30 (2) cells.

the DE when tackling networks of sizes 12×12, 14×14, 16×16, 18×18, 20×20 and Network 2 of size 30×30 cells.

Considering the fitness value evolution in Figs. 19–24, we can see that the behaviour of the 2SA-cGA, OI-GA and GPSO is the same for all sizes of networks. In fact, the GPSO has a quick convergence rate that ends by a stagnation after a given number of iterations. The OI-GA has a slow convergence rate in the first iterations when compared to the 2SA-cGA or the GPSO, but when the GPSO starts converging, the OI-GA continues evolving with gaps of stagnation. The DE has the quickest convergence rate since it quickly stagnates starting from the few first iterations. Finally, unlike both OI-GA and GPSO, our 2SA-cGA has a smooth and continuous convergence.

Fig. 25. Grid representation: (a) 12×12, (b) 14×14, (c) 16×16, (d) 18×18, (e) 20×20 and (f) $30 \times 30(2)$ cells.

Fig. 25(a)–(f) represent the grey-coded representation of the grid obtained at the end of a randomly-chosen execution when tackling networks of sizes 12×12, 14×14, 16×16, 18×18, 20×20 and $30 \times 30(2)$ cells, where lightened pixels represent the individuals with the best fitness values, while those with dark pixels are the individuals with the worst fitness values.

With regards to Fig. 25(a)–(f), one can see that for small instances (12×12 cells), the proposed approach converges totally (all the grid nodes have the same colour), while for a little bit bigger instances (14×14, 16×16 and 18×18 cells), divergence starts appearing within the population in the form of clusters (the grid contains spots of the same colour). Finally, for bigger instances of 20×20 and 30×30 cells, the 2SA-cGA's population diverges totally.

4.3. Statistical analysis: post-hoc test

In this section, we perform a series of statistical tests in order to investigate the significance of the obtained results in Table 9. First, before deciding what kind of test to perform (i.e. parametric or non-parametric), we verify two assumptions: normality of distribution and variance homogeneity. The first assumption is verified using a one-sample Kolmogorov-Smirnov test, while the second is checked by applying a Bartlett test. On the basis of the results of the two above-mentioned tests, we perform after a Kruskal-Wallis one-way analysis of variance test. For more insight, results of the Kruskal-Wallis test are available at[8]. Also, it is to be noted that all tests are conducted using a significance level equal to 5%.

Considering the results of the Kruskal-Wallis test, we perform in this section a post-hoc test to find where the difference occurs and ultimately deduce what is the best algorithm. The null-hypothesis, H_0, of this test is that the distribution of the results obtained by two algorithms are equal. To verify H_0, the post-hoc performs a series of pairwise comparisons between a reference algorithm (A) and another algorithm (B). For each instance, this test ranks both algorithms A and B on the basis of their results achieved in each execution. Then, on the basis of the mean of these ranks, it statistically orders the algorithms. Table 10 represents the results of this statistical ranking.

It is worth mentioning that the post-hoc test assigns the best ranks to algorithms with the highest fitness values. However, since we are dealing with a minimisation problem, the best algorithms

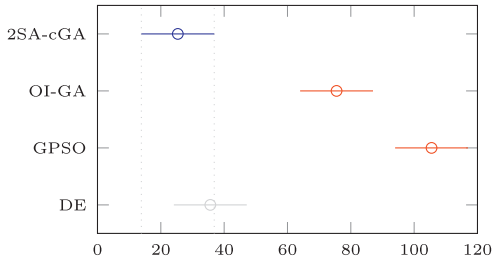

Fig. 26. P-H test: 12×12 cells.

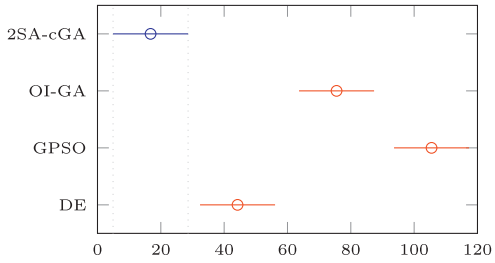

Fig. 27. P-H test: 30×30 (2) cells.

are those having the lowest statistical ranking. On the basis of the statistical ranking, we can see that our approach outperforms all state-of-the-art solvers in all sizes except for Network 1 of size 30×30 cells, where the DE could outperform the proposed 2SA-cGA.

Figs. 26 and 27 correspond to the results of the post-hoc test for the smallest and biggest networks. The segments represent the interval of distribution of ranking for each execution.

4.4. On the neighbourhood's choice and its influence

The neighbourhood's size and shape are one of the principal factors influencing the behaviour of cellular Evolutionary Algorithms (cEAs) (Alba & Dorronsoro, 2008). Indeed, the overlapped small neighbourhoods of cEAs induce *exploration*, while *exploitation* takes place inside each neighbourhood (Alba & Dorronsoro, 2005). In this case, the algorithm's selection pressure is a good indicator of the neighbourhood's influence on the search capabilities of the cEA (Alba & Dorronsoro, 2008).

In order to characterize/control, this time, the selection pressure of cEAs, several works such as those of Sarma and De Jong (1996, 1997), Alba and Troya (2000) and Alba and Dorronsoro (2005) led

[8] Appendix URL: https://tinyurl.com/Appendix-Link.

Table 10
Statistical ranking.

Algorithm	12×12	14×14	16×16	18×18	20×20	30×30 (1)	30×30 (2)
2SA-cGA	**25.37**	**26.72**	**26.97**	**20.52**	**17.98**	45.50	**16.77**
OI-GA	75.50	75.50	75.50	75.50	75.50	75.50	75.50
GPSO	105.50	105.50	105.50	105.50	105.50	105.50	105.50
DE	35.63	34.28	34.03	40.48	43.02	**15.50**	44.23

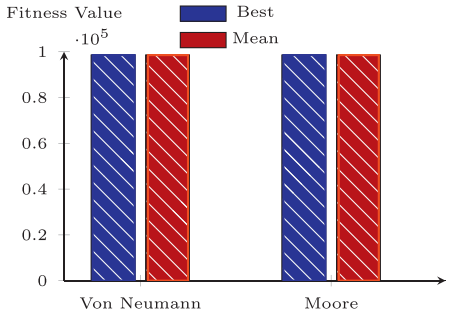

Fig. 28. Network 1 of size 4×4 cells .

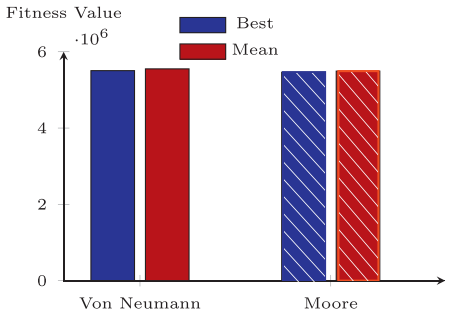

Fig. 29. Network 2 of size 30×30 cells.

to the appearance of the "*ratio*" concept. The latter quantifies the relationship between the neighbourhood and the grid (Alba & Dorronsoro, 2008). Mathematically speaking, it has been defined as the ratio between the neighbourhood's radius and the population's radius. For more mathematical insight of both the radius and ratio concepts, one can refer to Alba and Dorronsoro (2005, 2008). It has been proved that the ratio value has a direct influence on whether promoting the exploratory or exploitative capabilities of the algorithm. As a matter of fact, studies conducted in Alba and Dorronsoro (2005, 2008) showed that two algorithms with different population sizes and neighbourhoods but with the same ratio value displayed the same behaviour. Other works, such as in Sarma and De Jong (1997), proved also that algorithms with a similar ratio value show a similar selection pressure.

With regards to the neighbourhood's type and size used in this work, the study performed by Gordon, Mathias, and Whitley (1994) showed that small neighbourhoods are more suitable for tackling complex problems like it is the case for the mobility management problem. In that same perspective, the *Von Neumann* and *Moore* neighbourhoods have been proved to be the most commonly used ones (Alba & Dorronsoro, 2008). In our case, when calculating the radius of both *Von Neumann* and *Moore* neighbourhoods, they are found to be 0.89 and 1.15, respectively (difference of the order of $10^{-2} = 0.26$). Now, when computing the ratios of the proposed approach using either the *Von Neumann* and *Moore* neighbourhoods, they are found to be 0.0357 and 0.0459, respectively (difference of the order of $10^{-4} = 0.0102$). These quasi-identical ratios suggest that the two neighbourhoods will have the same influence on the search capabilities of the proposed approach.

This turns to be more evident especially that works such as Alba and Dorronsoro (2005, 2008) showed that two algorithms using two different population sizes and two different neighbourhoods (one using a population of 32×32 individuals and an *L5* neighbourhood and the second using a population of 64×64 individuals and a *C21* neighbourhood) display an identical selection pressure during iterations. Numerically speaking, the first algorithm had a ratio of 0.0695, while the second had a ratio of 0.0555 (difference of the order of $10^{-4} = 0.0140$).

All of these facts confirm that the use either the *Von Neumann* or *Moore* neighbourhoods will result in a closely similar *selection pressure evolution, search behaviour* and probably *efficiency*.

Figs. 28 and 29 are bar-based representations of the best and mean fitness values obtained by the proposed approach when using the Von Neumann and Moore neighbourhoods for tackling the *smallest* (16 cells) and the *largest* (900 cells) instances of the MMP. The best algorithms are represented with *dashed* bars. As it can be seen, the obtained results supports the above-drawn conclusions. However, for a deeper insight, further results of this comparison are available at[9].

[9] Appendix URL: https://tinyurl.com/Appendix-Link.

4.5. Algorithmic complexity study

Besides its efficiency, the proposed 2SA-cGA is proved to be less complex than the best state-of-the-art solvers. In fact, one can note that the adaptation feature added to the cGA is a simple arithmetic operation with constant complexity $\mathcal{O}(1)$, which makes that the proposed 2SA-cGA has the same complexity as the basic cGA ($\mathcal{O}(N)$). Taking now the first set of instances, the best state-of-the-art solver is the NSGA-II, but according to the work where it has been first proposed, it has a complexity of $\mathcal{O}(MN^2)$ where M is the number of objective function to be optimised (in our case $M = 2$ (Berrocal-Plaza et al., 2015a)), while N is the size of the population. When considering the second and third set of instances, the best state-of-the-art solvers are static algorithms with linear complexity, $\mathcal{O}(N)$.

These facts prove that our approach is not only numerically efficient, but also has a low complexity versus most of state-of-the-art algorithms.

5. Conclusion

In this paper we have proposed a low-complexity adaptive cGA for solving the mobility management problem in advanced cellular networks. The experiments are conducted on twenty-five differently-sized realistic instances. The results have been compared to those of twenty-six state-of-the-art solvers, several statistical tests have been conducted as well. The results of the experiments showed that our approach is more robust, scalable, efficient and less complex than most of state-of-the-art algorithms.

Taking th first set of instances, we could reach 12 out of 12 of the best results known in the literature. Also, in 7 out of 12 instances, we could achieve results as good as those obtained by the best solvers known in the literature, while we conceived to outperform all state-of-the-art solvers in 2 out of 12 instances. When considering the second set of instances, we could reach 2 out of 6 best results known in the literature and outperform all state-of-the-art solvers in 3 out of 5 instances. Finally, with regards to the third set of instances, we could surpass all state-of-the-art algorithms in 6 out of 7 instances.

As future perspective of research, we plan to deploy our algorithm in a company. Also, we want to go more realistic and include it in the frameworks Network Simulator 2 and 3 (NS2 and NS3) using data from the OpenStreetMap platform.

Acknowledgements

Authors would like to address special thanks to Mrs Malika Belaifa and Mrs Zeineb Dahi for their help in creating the realistic benchmarks. Also, authors would like to thank Mr Louai Rabeh Dahi for his help in treating and checking the numerical results. Prof. Alba acknowledges partial funding from Spanish-plus-FEDER

MINECO project moveON TIN2014-57341-R, TIN2016-81766-REDT and TIN2017-88213-R.

References

Alba, E., & Dorronsoro, B. (2005). The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. *IEEE Transactions on Evolutionary Computation, 9*(2), 126–142. doi:10.1109/TEVC.2005.843751.

Alba, E., & Dorronsoro, B. (2008). *Cellular genetic algorithms* (1st). Springer Publishing Company, Incorporated.

Alba, E., García-Nieto, J., Taheri, J., & Zomaya, A. (2008). New research in nature inspired algorithms for mobility management in GSM networks. In *Proceedings of the EvoWorkshops on applications of evolutionary computing: EvoCOMNET, EvoFIN, EvoHOT, EvoIASP, EvoMUSART, EvoNUM, EvoSTOC, and EvoTransLog: 4974* (pp. 1–10). Springer. doi:10.1007/978-3-540-78761-7_1.

Alba, E., & Troya, J. (2000). Cellular evolutionary algorithms: Evaluating the influence of ratio. In *Proceedings of the international conference on parallel problem solving from nature VI, (PPSN-VI): 1917* (pp. 29–38). Springer.

Almeida-Luz, S., Vega-Rodriguez, M. A., Gomez-Pulido, J. A., & Sanchez-Perez, J. M. (2008). Applying differential evolution to the reporting cells problem. In *Proceedings of the international multiconference on computer science and information technology, (IMCSIT)* (pp. 65–71). IEEE. doi:10.1109/IMCSIT.2008.4747219.

Almeida-Luz, S. M., Vega-Rodriguez, M. A., Gomez-Pulido, J. A., & Sanchez-Perez, J. M. (2010). A scatter search based approach to solve the reporting cells problem. In *Proceedings of the 5th international workshop on soft computing models in industrial and environmental applications, (SOCO): 73* (pp. 145–152). Springer. doi:10.1007/978-3-642-13161-5_19.

Almeida-Luz, S. M., Vega-Rodriguez, M. A., Gomez-Pulido, J. A., & Sanchez-Perez, J. M. (2010). Solving the reporting cells problem using a scatter search based algorithm. In *Proceedings of the 7th international conference on rough sets and current trends in computing, (RSCTC): 6086* (pp. 534–543). Springer.

Almeida-Luz, S. M., Vega-Rodriguez, M. A., Gomez-Pulido, J. A., & Sanchez-Perez, J. M. (2011). Differential evolution for solving the mobile location management. *Applied Soft Computing, 11*(1), 410–427. doi:10.1016/j.asoc.2009.11.031.

Baburaj, C. A., & Alagarsamy, K. (2011). Generalized N x N network concept for location and mobility management. In *Proceedings of the 1st international conference on computer science and information technology, (CCSIT): 132* (pp. 321–328). Springer. doi:10.1007/978-3-642-17878-8_32.

Baburaj, C. A., Phil, M. C. A. M., & Alagarsamy, K. (2010). A review on various network results for reporting cell planning in genetic algorithm technique. *International Journal of Engineering Science and Technology, 2*, 4088–4094.

Bäck, T., & Schütz, M. (1996). Intelligent mutation rate control in canonical genetic algorithms. In *Foundations of intelligent systems: 1079* (pp. 158–167). Springer.

Bar-Noy, A., & Kessler, I. (1993). Tracking mobile users in wireless communications networks. *IEEE Transactions on Information Theory, 39*(6), 1877–1886. doi:10.1109/18.265497.

Berrocal-Plaza, V., Vega-Rodríguez, M. A., & Sánchez-Pérez, J. M. (2014). Non-dominated sorting and a novel formulation in the reporting cells planning. In *Proceedings of the 9th international conference on hybrid artificial intelligence systems, (HAIS): 8480* (pp. 285–295). Springer.

Berrocal-Plaza, V., Vega-Rodriguez, M. A., & Sanchez-Perez, J. M. (2014). A strength pareto approach and a novel formulation in the reporting cells planning. In *Proceedings of the international joint conference, (SOCO'14, CISIS'14 and ICEUTE'14): 299* (pp. 1–10). Springer. doi:10.1007/978-3-319-07995-0_1.

Berrocal-Plaza, V., Vega-Rodriguez, M. A., & Sanchez-Perez, J. M. (2014c). A strength pareto approach to solve the reporting cells planning problem. In *Proceedings of the 14th international conference on computational science and its applications, (ICCSA): 8584* (pp. 212–223). Springer. doi:10.1007/978-3-319-09153-2_16.

Berrocal-Plaza, V., Vega-Rodriguez, M. A., & Sanchez-Perez, J. M. (2014d). Studying the reporting cells strategy in a realistic mobile environment. In *Proceedings of the 6th world congress on nature and biologically inspired computing, (NABIC)* (pp. 29–34). IEEE. doi:10.1109/NaBIC.2014.6921900.

Berrocal-Plaza, V., Vega-Rodriguez, M. A., & Sanchez-Perez, J. M. (2015). A multiobjective study of the Gaussian cluster paging in the reporting cells strategy. *Applied Soft Computing, 28*(0), 332–344. doi:10.1016/j.asoc.2014.12.005.

Berrocal-Plaza, V., Vega-Rodríguez, M. A., & Sánchez-Pérez, J. M. (2015). Optimizing the mobility management task in networks of four world capital cities. *Journal of Network and Computer Applications, 51*, 18–28. doi:10.1016/j.jnca.2015.01.002.

Dahi, Z. A. (2017). *Optimisation problem solving in the field of cellular networks*. Constantine 2 University Ph.D. thesis..

Dahi, Z. A., Mezioud, C., & Alba, E. (2016). A novel adaptive genetic algorithm for mobility management in cellular networks. In *Proceedings of the 11th international conference on hybrid artificial intelligent systems, (HAIS)* (pp. 225–237). Springer. doi:10.1007/978-3-319-32034-2_19.

González-Álvarez, D. L., Rubio-Largo, A., Vega-Rodríguez, M. A., Almeida-Luz, S. M., Gómez-Pulido, J. A., & Sánchez-Pérez, J. M. (2012). Solving the reporting cells problem by using a parallel team of evolutionary algorithms. *Logic Journal of the IGPL, 20*(4), 722–731. doi:10.1093/jigpal/jzr016.

Gordon, V., Mathias, K., & Whitley, D. (1994). Cellular genetic algorithms as function optimizers: Locality effects. In *Proceedings of the ACM symposium on applied computing, (SAC)* (pp. 237–241). ACM.

Hać, A., & Zhou, X. (1997). Locating strategies for personal communication networks, a novel tracking strategy. *IEEE Journal on Selected Areas in Communications, 15*(8), 1425–1436. doi:10.1109/49.634783.

Kim, S. S., Kim, G., Byeon, J. H., & Taheri, J. (2012). Particle swarm optimization for location area mobility management. *International Journal of Innovative Computing Information and Control, 8*(12), 8387–8398.

Kim, S. S., Kim, I. H., Mani, V., Kim, H. J., & Agrawal, D. P. (2010). Partitioning of mobile network into location areas using ant colony optimization. *ICIC International, ICIC Express Letters, Part B: Applications, 1*(1), 39–44.

Mehta, F., & Swadas, P. (2009). Comparison of simulated annealing approach and a novel tracking strategy to reporting cell planning problem of mobile location management. In *Proceedings of the international conference on signals, systems and automation* (pp. 208–213). Universal Publishers, Gujarat. doi:10.1109/ICECS.2010.5724723.

Mehta, F., & Swadas, P. (2009). A simulated annealing approach to reporting cell planning problem of mobile location management. *International Journal of Recent Trends in Engineering and Technology, 2*(2), 355–367.

Parija, S. R., Addanki, P., Sahu, P. K., & Singh, S. S. (2015). Cost reduction in reporting cell planning configuration using soft computing algorithm. In *Proceedings of the 3rd international conference on frontiers of intelligent computing: Theory and applications, (FICTA): 327* (pp. 823–830). Springer. doi:10.1007/978-3-319-11933-5_93.

Patel, B., & Bhatt, P. (2014). A differential evolution algorithm for reporting cell problem in mobile computing. *International Journal of Engineering Research & Technology (IJERT), 3*(3), 2175–2180.

Patra, M., & Udgata, S. K. (2011). Soft computing approach for location management problem in wireless mobile environment. In *Proceedings of the 2nd international conference on swarm evolutionary and memetic computing, (SEMCCO): 7077* (pp. 248–256). Springer. doi:10.1007/978-3-642-27242-4_29.

Razavi, S. M. (2011). *Tracking area planning in cellular networks*. Department of Science and Technology, Linkoping University Ph.D. thesis..

Rothlauf, F. (2011). *Design of modern heuristics: Principles and application* (1st). Springer.

Rubio-Largo, A., Gonzalez-Alvarez, D. L., Vega-Rodriguez, M. A., Almeida-Luz, S. M., Gomez-Pulido, J. A., & Sanchez-Perez, J. M. (2010). A parallel cooperative evolutionary strategy for solving the reporting cells problem. In *Proceedings of the 5th international workshop on soft computing models in industrial and environmental applications, (SOCO): 73* (pp. 71–78). Springer. doi:10.1007/978-3-642-13161-5_10.

Sarma, J., & De Jong, K. (1996). An analysis of the effect of the neighborhood size and shape on local selection algorithms. In *Proceedings of the international conference on parallel problem solving from nature IV, (PPSN-IV): 1141* (pp. 236–244). Springer.

Sarma, J., & De Jong, K. (1997). An analysis of local selection algorithms in a spatially structured evolutionary algorithm. In *Proceedings of the seventh international conference on genetic algorithms, (ICGA)* (pp. 181–186). Morgan Kaufmann.

Sivanandam, S. N., & Deepa, S. N. (2007). *Introduction to genetic algorithms* (1st). Springer.

Subrata, R., & Zomaya, A. Y. (2002). Artificial life techniques for reporting cell planning in mobile computing. In *Proceedings of the workshop on biologically inspired solutions to parallel processing problems, (BioSP3)*.

Subrata, R., & Zomaya, A. Y. (2002). Artificial life techniques for reporting cell planning in mobile computing. In *Proceedings of the 16th international parallel and distributed processing symposium, (IPDPS '02)* (pp. 75–*). IEEE.

Subrata, R., & Zomaya, A. Y. (2003). A comparison of three artificial life techniques for reporting cell planning in mobile computing. *IEEE Transactions on Parallel and Distributed Systems, 14*(2), 142–153. doi:10.1109/TPDS.2003.1178878.

Subrata, R., & Zomaya, A. Y. (2003). Evolving cellular automata for location management in mobile computing networks. *IEEE Transactions on Parallel and Distributed Systems, 14*(1), 13–26. doi:10.1109/TPDS.2003.1167367.

Subrata, R., & Zomaya, A. Y. (2006). Location management algorithms based on biologically inspired techniques. In *Handbook of algorithms for mobile and networking and computing: 16* (pp. 363–390). Taylor & Francis.

Taheri, J., & Zomaya, A. Y. (2008). A modified hopfield network for mobility management. *Wireless Communications and Mobile Computing, 8*(3), 355–367. doi:10.1002/wcm.582.

Vekariya, M. H., & Swadas, P. B. (2015). Hybrid genetic algorithm-simulated annealing approach for reporting cell planning problem of location management. *International Journal of Innovative Research in Computer and Communication Engineering, 3*(6), 5388–5395.

Wang, D., Xiong, C., & Huang, W. (2014). Group search optimizer for the mobile location management problem. *The Scientific World Journal, 2014*(12), 2–8.

Wang, L., & Si, G. (2010). Optimal location management in mobile computing with hybrid genetic algorithm and particle swarm optimization (GA-PSO). In *Proceedings of the 17th IEEE international conference on electronics, circuits and systems, (ICECS)* (pp. 1160–1163). IEEE. doi:10.1109/ICECS.2010.5724723.

Wang, W., Wang, F., Pan, Q., & Zuo, F. (2009). Improved differential evolution algorithm for location management in mobile computing. In *Proceedings of the international workshop on intelligent systems and applications, (ISA)* (pp. 1–5). IEEE. doi:10.1109/IWISA.2009.5072906.