Applied Soft Computing 116 (2022) 107992

Contents lists available at ScienceDirect

Applied Soft Computing

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

A takeover time-driven adaptive evolutionary algorithm for mobile user tracking in *pre*-5G cellular networks

Zakaria Abdelmoiz Dahi [a,b,*], Enrique Alba [c], Gabriel Luque [c]

[a] Dep. de Lenguajes y Ciencias de la Computación, Fac. ETSI Informática, University of Malaga, Spain
[b] Dep. Fundamental Computer Science and Its Applications, Fac. NTIC, University of Constantine 2, Algeria
[c] ITIS Software, Edificio Ada Byron, University of Malaga, Spain

A R T I C L E I N F O

Article history:
Received 4 April 2019
Received in revised form 8 October 2021
Accepted 12 October 2021
Available online 27 October 2021

Keywords:
Cellular networks
Evolutionary algorithms
Adaptation

A B S T R A C T

Cellular networks are one of today's most popular means of communication. This fact has made the mobile phone industry subject to a huge scientific and economic competition, where the quality of service is key. Such a quality is measured on the basis of reliability, speed and accuracy when delivering a service to a user no matter his location or behaviour are. This fact has placed the users' tracking process among the most difficult and determining issues in cellular network design. In this paper, we present an adaptive bi-phased evolutionary algorithm based on the takeover time to solve this problem. The proposal is thoroughly assessed by tackling twenty-five real-world instances of different sizes. Twenty-eight of the state-of-the-art techniques devised to address the users' mobility problem have been taken as the comparison basis, and several statistical tests have been also conducted. Experiments have demonstrated that our solver outperforms most of the top-ranked algorithms.

1. Introduction

Because of the affordability and easy accessibility of their services, cellular networks have become one of the most widespread means of communication. Actually, the GSM[1] association estimated that in 2017 nearly 50% of the globe's population was using mobile communications services [1]. This has made the telephony industry a highly competitive field, one in which the quality of service is a determinant factor. This quality is reflected by how quick and accurately a service is provided to a user; no matter the type of the service and the location or the comportment of this user are. Technically speaking, this depends on how fast and well a mobile user can be located. In fact, like most of present-day's wireless networks (e.g. sensor [2], vehicular [3], etc.), the principle of a cellular network is based on the mobility of its users. This problem has become more crucial with the emergence of the Internet of Things (IoT) and heterogeneous networks.

The importance of this task is even more accentuated since all communications pass through a bandwidth. The bandwidth, therefore, is also a valuable resource, economically and technically speaking. As the number of subscribers has tremendously increased, the optimal use of this resource has quickly become a big issue. Actually, some studies have demonstrated that the communication traffic generated when attempting to locate mobile users engender more than 33% of the messages transiting via the bandwidth [4]. All these evidences have made the users' tracking process one of today's most crucial and determining tasks in cellular networks.

Metaheuristics are efficient tools that can be employed to tackle such complex problems. Evolutionary Algorithms (EAs) are one of the most promising [5]. However, their efficiency is greatly influenced by the quality of their parameters' tuning. This is generally done over many thorough and time-consuming processes so that one can get the best configuration to a certain problem. Besides, such tuning involves a profound understanding of the algorithm utilised and the problem being solved [6]. So, every change in the problem characteristics is likely to compromise the effectiveness of the algorithm and a new tuning cycle will be mandatory each time to retrieve the desired efficiency. This restricts the utilisation of most of metaheuristics to skilled professionals within abstract research rather than inexperienced users in real-world environments.

To cope with this issue, much research effort is deployed to design algorithms that can automatically adjust the value of the algorithm parameters without involving an outer agent. Many adaptation schemes have been already presented in the literature,

* Corresponding author at: Dep. de Lenguajes y Ciencias de la Computación, Fac. ETSI Informática, University of Malaga, Spain and Dep. Fundamental Computer Science and Its Applications, Fac. NTIC, University of Constantine 2, Algeria.
E-mail addresses: zakaria.dahi@uma.es, zakaria.dahi@univ-constantine2.dz (Z.A. Dahi), eat@lcc.uma.es (E. Alba), gabriel@lcc.uma.es (G. Luque).

[1] GSM: Global System for Mobile communications.

https://doi.org/10.1016/j.asoc.2021.107992

but they can be categorised as deterministic, adaptive or self-adaptive [7]. The adaptive policy is the one researched in this work. The aim of an adaptation strategy is to find the adequate balance between the algorithm's exploitative and exploratory capacities. The quality of this balance can be expressed in terms of the convergence rate of the algorithm itself. Selection pressure models are mathematical tools that permit studying/modelling the convergence rate of a given algorithm subject to some selection method [8,9]. However, until now the use of such models has been restricted to theoretical analysis and no practical use has been done.

In this paper, we present an adaptive bi-phased evolutionary algorithm for solving the Mobility Management Problem (MMP). Unlike most of previous adaptive EAs, designed in the literature [10,11], the EA used in our approach does not guide the *current* steps of adaptation on the basis of some information collected from *previous* runs. Indeed, our approach utilises a mathematical model of the *future* expected algorithm convergence in order to control the adaptation within the *current* iteration.

The robustness, efficiency and scalability of the proposal have been assessed by solving twenty-five differently-sized realistic instances of the problem. In addition, twenty-eight of state-of-the-art solvers devised to address the same problem are taken as a comparison basis, and numerous statistical tests have also been conducted.

The remainder of the paper is structured as described. In Section 2, we introduce fundamental concepts related to mobility management, EAs' adaptation and selection pressure models. In Section 3, we carry out a literature review of mobility management solvers, the takeover models and the GA advances. In Section 4, we present our approach. Sections 5–7 are consecrated to the experimental study. Finally, we wrap up the paper in Section 8.

2. Fundamental background

We present, all along this section, some introductory notions about the management of users' mobility in *pre*-5G networks, adaptation and selection pressure models for EAs.

2.1. Management of user's mobility in pre-5G networks

Mobility Management (MM) is about locating users to whom a certain service (e.g. phone call, photo sharing, etc.) is destined. Whatever the generation of network being used (e.g. 2G, 3G, 4G, etc.), this task is accomplished by some component in the network's system [12]. For each user, this element stores and manages two pieces of mobility data: the cell containing the user and the area (a group of cells designated with an identical identifier) to which this cell belongs. Therefore, the mobile user needs to update his identifier only when he penetrates a new area and this by sending a *Location-Update* (LU) to the MM Core (MMC) [13].

When a service (e.g. Skype call, Whatsapp message, etc.) is destined to a user, the network must discover first in which cell of the network the user is located. To do so, the MMC executes a polling cycle in which *paging* messages are transmitted to all cells within the user's area. The way location update or paging are carried out is defined by the MM scheme. Regarding this evidence, the effectiveness of the MM task stands in how good and efficient the used policy is for managing and minimising the paging and LU costs.

Taking the LU task, for instance, two principal approaches exist: static and dynamic (see Fig. 2.1). Within the dynamic approach, the LUs are based on the variability of the mobility patterns and user calls. This requires usually the on-line gathering and processing of data, which consumes significant computing resources. By contrast, the LUs in the static policy depend on the changes of network's topology (i.e. independent of user characteristics), which allows an effective implementation and has low computational needs. Therefore, the static strategy is the one investigated here. This approach can be further classified into four schemes: the location area, reporting cell, never-update and always-update. The never and always-update schemes engender either excessive costs of LU or paging. So, they are barely implemented in real networks. That being said, the rest of the schemes are those used in real-life scenarios [14]. Taking the location area and Reporting Cell Schemes (RCS), both are investigated in the literature but a real keen interest is noticeable for the second scheme (see Section 3.1). Taking this into account, the RCS is the one researched in this paper. In the case of this scheme, the network's cells are labelled as Reporting or Non-Reporting Cells (RCs or NRCs). The mobile user performs an LU only if he/she enters an RC, while in NRCs, he/she can move freely without acknowledging a new location. The prior work in [14] has shown that yet with knowledge from the network, finding the best configuration of the RCs is an NP-hard problem [13].

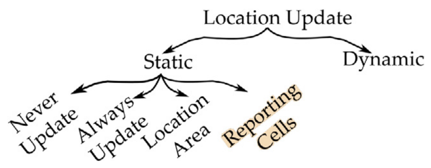

Fig. 2.1. Location-update strategies in *pre*-5G cellular networks.

With consideration to the paging process, two policies exist: *standard* and *selective*. In the standard paging strategies (e.g. blanket polling), all cells of the user's area are simultaneously paged, while selective schemes consider some properties (e.g. velocity, distance, etc.) when paging the user's area cells. Previous study in [14] has demonstrated that the selective paging strategies require the collection of supplementary user information and a modification of the system's implementation, whereas the standard strategy does not necessitate further knowledge (e.g. user location, etc.). This places it among the most functional and widely used schemes [14]. Besides, another noteworthy fact is that the standard scheme (blanket polling) is the one originally implemented within the RCS. Thus, it is the one investigated within this work.

The RCS was first devised by Bar-Noy and Kessler [15]. Then, the Reporting Cell Problem (RCP) was modelled and formulated as a binary optimisation problem by Hać and Zhou [16]. In the next sections, we introduce both the RCP solution representation and mathematical formulation presented by the authors of that paper.

2.1.1. Solution representation

The RCP solution can be described as a binary vector $\overrightarrow{X}$. Each vector $\overrightarrow{X} = \{x_1, \ldots, x_D\}$ represents a unique configuration of the network, where D is the network's size. Each bit x_k form $\overrightarrow{X}$ (where $k = 1, \ldots, D$) represents the status (i.e. RC or NRC) of the k^{th} cell. If $x_k = 1$, the k^{th} cell is acknowledged as an RC, otherwise ($x_k = 0$), it is recognised as an NRC. Fig. 2.2 illustrates the standard RCP solution representation.

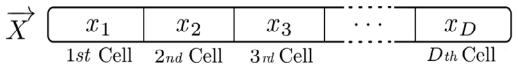

Fig. 2.2. Representation of an RCP solution.

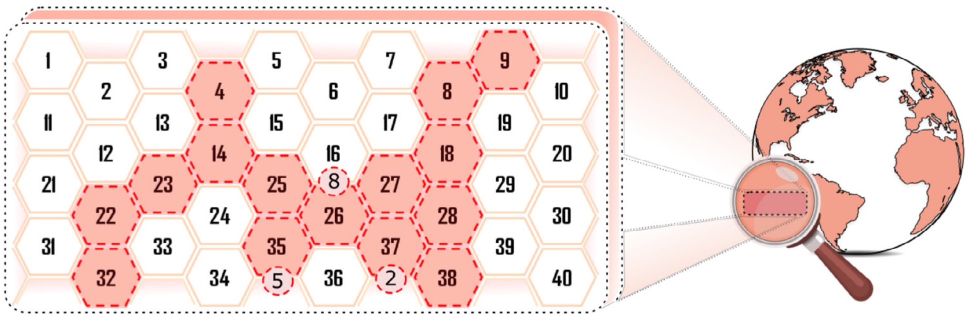

Fig. 2.3. Illustration of vicinity calculation.

2.1.2. Objective function

The RCP fitness function introduced by Haćand Zhou [16] is defined by Eq. (2.1).

$$\min_{\overrightarrow{X}=\{x_1,\ldots,x_D\}} \quad Q(\overrightarrow{X}) = \varsigma.\sum_{i=1}^{\vartheta} L_i + \sum_{j=1}^{D} P_j.V_j \tag{2.1}$$

The function Q is used for evaluating the goodness of some solution $\overrightarrow{X}$, where L_i represents the number of LUs performed within the i^{th} reporting cell (ϑ is the set of RCs in the network). P_j is the amount of paging operations performed within the j^{th} cell (D is the number of the network's cells). Lastly, V_j is the vicinity of the j^{th} cell. The variable ς is used to reflect that the LU is considered to be more significant than the paging cost. Thus, for the sake of performing a reliable comparison with state-of-the-art solvers, we utilise the same value of ς ($\varsigma = 10$) employed in all works that tackled the RCP [1,4,17].

A cell's vicinity is described as the maximum number of cells that must be browsed when an incoming service is intended for this cell. Besides, the way vicinity is computed depends on the cell's type (RC or NRC). The vicinity value of an RC is the number of NRCs (including the RC itself) that can be reached from this cell without going through another RC. This being said, an NRC can be reached from several RCs. Consequently, the vicinity of an NRC can be described as the highest vicinity value of the RCs from where this NRC can be reached [13].

Fig. 2.3 illustrates an example of a configuration for a network of forty cells with fifteen RCs (those marked with a dashed contour). The vicinity of the 36th NRC is the highest vicinity of its neighbours: the 35th, 26th and 37th RCs. The values of the vicinity for those cells are 5, 8 and 2, respectively. Therefore, the vicinity of the 36th NRC is 8.

2.2. Dynamic adaptation within EAs

The uniqueness of every optimisation problem lies in some features that make it distinguishable from another problem (e.g. epistasis, multi-modality and deceptiveness). This goes even further since despite being within the same problem, each instance might be different from another. Consequently, most metaheuristics nowadays are problem and even instance-dependent. Indeed, their effectiveness is greatly correlated with the quality by which their parameters are tuned. This tuning is done through several exhaustive phases to better fit the instance or the problem at hand. Therefore, every change in the instance/problem features may reduce the algorithm's efficacy. Besides, such tuning involves a profound knowledge about both the addressed problem and the algorithm used. All these constraints restrain the utilisation of hand-tuned (i.e static) algorithms to specialists within pure theoretical research rather than inexperienced users in practical real-life scenarios.

As a promising solution to this pitfall, dynamically adaptive algorithms seem to be a good substitute to hand-tuned ones. Adaptation within evolutionary algorithms is a tentative to make the EA's search process adjust to better fit the characteristics of the optimisation problem being engaged. This stands in conceiving mechanisms that evolve the parameters' values without an outer intervention. Considering the problem and the algorithm at hand, various parameters can be adapted and also several policies can be employed to do so (see Fig. 2.4). Actually, many adaptation schemes have been previously devised but all can be classified as adaptive, deterministic and self-adaptive [7].

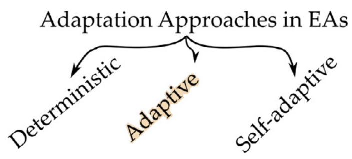

Fig. 2.4. Dynamic adaptation strategies within EAs.

Adaptation in the deterministic approach tunes the value of a certain parameter using a predefined equation. The noteworthy fact in this scheme is that; to determine the adaptation, no exchange of information is made between the algorithm and another external agent. In contrast with the first strategy, the adaptive approach is based on the feedback obtained from the algorithm throughout the adaptation. Finally, in the self-adaptive policy, both the problem's variables and the parameters being adjusted are encoded as an individual (i.e. solution) that evolves by applying the algorithm's operators [13]. In this work, we focus on the adaptive scheme.

2.3. Selection pressure models within EAs

The efficacy of an evolutionary algorithm basically lies in its ability of finding the appropriate balance between its exploitative and exploratory capacities. A good measure of *how* this balance evolves can be reflected by the convergence of the population itself. Thus, the convergence rate is key in any evolutionary algorithm. Indeed, much research effort has been made to design some analytical approaches that allow the selection pressure evolution of different selection methods to be studied. The majority of the models presented in the literature are built upon the following question; starting from a single best solution, how long does it take to the latter to take over the entire population? To answer this question, many models make two fundamental assumptions. The first supposition is that no perturbation operator is used and that the population evolves using only the selection. The second assumption is that at the beginning of the search process, the population contains only one copy of the best individual. In light of this, two important concepts appeared, the growth curve and the takeover time. Fig. 2.5 shows a concrete instance of these two concepts for a population undergoing a binary tournament selection [9,18].

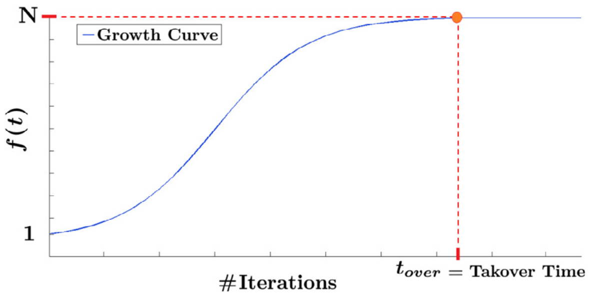

Fig. 2.5. Growth curve and takeover time.

Considering a population of N individuals, the *growth curve* can be mathematically described as a function, f, that expresses in terms of algorithm iterations, t, the number of copies of the best individual, $f(t)$, contained in the population, whereas the *takeover time* can be described as the first iteration t_{over}, where the function $f(t_{over}) = N$. In other words, the takeover time can be thought of as the time needed for the algorithm to fill the entire population with the best individual.

3. Related work

In this section, we carry out a literature review of state-of-the-art solvers proposed to tackle the MMP and also the mathematical models for the growth curve and finally the GA's advances.

3.1. Mobility management

The reporting cell scheme was firstly proposed by Bar-Noy and Kessler in [15] and later formulated by Hać and Zhou in [16] as a binary problem. In both papers, the authors stated that the RCP was NP-complete to solve, which encouraged the utilisation of approximate search techniques instead of exhaustive ones. Fig. 3.1 presents a literature review of the MMP's solvers.

Considering the MMP's objective function formulation and the data used for assessing the devised solvers, the literature can be divided into three groups of works, where the first one treats a first formulation of the MMP's fitness function and a set of 12 networks [1,4,29–31,34,35]. Although, it is to be noted that some works of this 1st group treats additional instances [1,30,34]. The second group uses a different MMP fitness function called "*cost per call arrival*" and experiments are conducted on up to 6 networks [27,28,32,37,38,41,42,47–50]. The final group are isolated works that use the MMP fitness function and all 12 networks used by the 1st group and also some networks from those of the 2nd group [39,40,45,46], or only some networks from the 2nd group [44] or propose new networks [17]. In all the above-cited works: (I) no one treated all the networks existing in the literature (i.e. those of the 1st and 2nd groups), (II) no one treated both MMP's fitness function (i.e. those of the 1st and 2nd groups), (III) the number of state-of-the-art solvers considered as a comparison basis is small (no more than 5) since no work compared its results against all the solvers devised in the literature. All these shortfalls do not ensure the encompassiveness and reliability of the obtained results and conclusions. As far as the authors' knowledge, the work in [13] is the only one to have considered both MMP's objective functions, performed experiments on the largest set of networks (25 instances: those of the 1st and 2nd groups and also proposed 7 new real-world-sized networks) and compared its results against the largest set of solvers (26 algorithms).

Now, considering the experimental validation part, as shown in Table 1, most of the works done on the MMP suffer from severe shortfalls that make the reliability of their results more questionable. Indeed, some works use experiments' and algorithms' settings that are different from those used in the literature, and then they even compare their results with those same works. Also, the choice of the state-of-the-art solvers taken as a comparison basis was not justified and those selected were very few, not enough diversified and not the top-ranked ones. No explanation of the choice of the treated networks was given and those selected were neither diverse nor include large real networks. Most of the works did not assess the devised algorithms based on other criteria than efficiency. Furthermore, they do not report statistical tests to assess the correctness of the results. Even when applied, no justification of the choice of the tests was given, especially that they have to be chosen carefully to ensure the reliability of the conclusions. No thorough sensitivity analysis was made on the devised approach to provide more understanding and explanation about the strength/weakness of the proposal.

Regarding the above-highlighted pitfalls, for the sake of reliability, our work goes even further than what has been done in [13] and the rest of the literature and this by considering both MMP's formulations. Furthermore, to ensure a fair and sound comparison, we use, in our work, the same experiments' and algorithms' settings as those employed in the literature we are comparing with. We perform our experimentations on the largest set of networks ever used, including the largest real-world networks ever studied in the literature (25 networks). Also, we compare our approach against the largest, ever considered in the literature, set of existing solvers (28 algorithms). The techniques that we take as a comparison basis and the networks used during the experimentations have been chosen carefully to be enough encompassing and represent several properties and complexities. This has been done to assess our approach based on several criteria (e.g. efficiency, reliability and robustness) and prove with certitude its efficiency. We go further by applying well-chosen thorough statistical tests to prove the correctness of our results. Finally, we perform a profound parameter-sensitivity analysis to provide a further understanding of our proposal.

3.2. Takeover time models

The first papers to study the concept of takeover time of some EAs using different selection methods were those of Goldberg et al. in [51,52]. Since then, numerous mathematical models have been introduced to express both takeover time and growth curve for different structured and unstructured evolutionary algorithms. Table 2 presents a literature review of the predominant models that were previously devised.

Many studies like those in [60,63,64], have stressed that the accuracy of a given model depends on the degree to which the conditions of its use match the assumptions that were considered during its design. Thus, the majority of the models in Table 2 as well as others in the literature are either restricted by the type of algorithm they were designed for (e.g. structured EAs: cellular [56–62] and distributed [63,64]), the type of problem tackled (e.g. dynamic [8]) or their assumptions (e.g. the population evolves using only the selection and no variation operators are used like mutation, crossover, etc.). Keeping in mind these facts, their accuracy will be weak in our case, since we are investigating an unstructured EA that evolves the population using selection and variation operators (mutation and crossover) to solve a static optimisation problem.

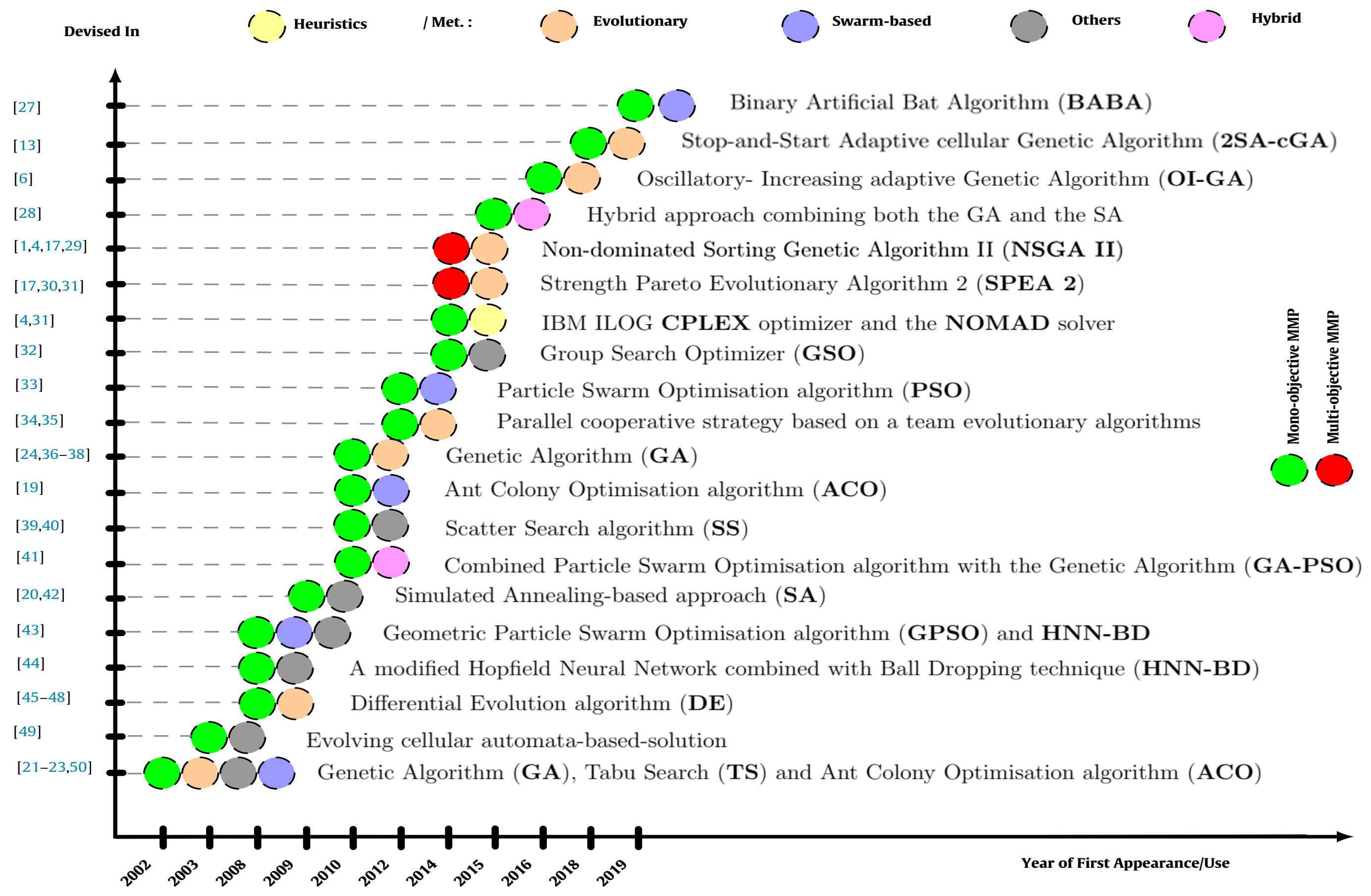

Fig. 3.1. Literature survey of the MMP solvers [19–26].

Table 1
Experimental pitfalls in the MMP's literature.

Experiments' and algorithms' settings	Solvers compared with	Chosen benchmarks	Statistical tests	Parameter sensitivity analysis
[24,28,36–38,42]	[24,28,31,35–38,41,42,47]	[1,24,28–32,36,37,39–48,50]	[24,28–32,34–38,41–45,47,48,50]	All

Table 2
Literature review of takeover models.

Envir.	Alg. structured	Proposed in	Based on	Devised for
		[53,54]	Logistic formula	/
		[55]	Studied the Logistic formula	/
		[56,57]	Hypergraphs	EAs with structured population
		[58]	Studied the hypergraphs and logistic formula models	Structured populations (array and ring)
		[59]	/	Toroidal cEAs
	Yes	[60]	Stochastic mathematical model	Asynchronous and synchronous EAs with structured population
Stationary		[61]	Idealised selecto-Lamarckian model	Structured population-based EAs
		[62]	Punctuated equilibria model	Cellular EAs
		[60,63,64]	Compared both hypergraph and logistic models in [53,56]	/
		[60,63,64]	Another version of the logistic model called LOG2	Distributed EAs
		[65]	/	/
	No	[66]	/	Genetic algorithm
		[9,18]	Rayleigh Distribution	Population-based algorithms
Dynamic	/	[8,25]	/	/

Therefore, in our work we employ the model proposed in [9, 18]. It has many advantages over other models. First, it has the benefit of being simple (i.e. it uses only a single value to describe the behaviour of any population-based algorithm under any selection method and even variation operators). Second, the entire model is governed by the value of only one parameter. Additionally, it enables the possibility of predicting the control parameter and estimating its value with a relatively low number of steps, but still maintaining a high level of exactness [18]. Furthermore, it was designed with the ability to handle multiple types of selection such as proportional and tournament-based ones. Finally, the assumptions made when proposing this model are similar to those used here, like selection with binary tournament and bit-flip mutation. All these factors increase the accuracy of this model within our framework.

3.3. A bird's eye view of the GA's advances

Since Holland presented the GA in [67], many works have been conducted on it leading to several advances. For so many reasons, the enthusiasm for the GA was and still going intensively that it is hard to enumerate all of these breakthroughs. Indeed, several extensive surveys exist on each specific topic where the GA has been studied such as hybridisation, application domains (e.g. data mining, artificial intelligence, etc.), design of its operators (e.g. crossover, mutation, etc.) [5], its variants (e.g. parallelism, multi-objective, dynamic, etc.) [68] and countless other topics. Taking into consideration all of this, one of the most noticeable GA advances are the LTGA (Linkage Tree Genetic Algorithm) [69], LT-GOMEA (Linkage Tree Gene-pool Optimal Mixing Evolutionary Algorithm) [70], P3 (Parameter-less Population Pyramid) [71], DSMGA-II (Dependency Structure Matrix Genetic Algorithm II) [72] and the 3LOa (Linkage Learning based on Local Optimisation algorithm) [73]. It is to be noted that the analysis we provide here about each solver is made regarding the original work where it has been proposed. Otherwise, we will explicitly cite the work(s) that support(s) our statement(s).

Similarly, all the above-cited solvers use linkage learning. Roughly speaking, the LT-GOMEA, P3 and the DSMGA-II are greatly based on the original LTGA and partially inspired by each other. However, LT-GOMEA uses a strategy for optimal gene mixing based on computing the frequencies and probabilities of the appearance of some substructures after recombination of other given structures. The P3 uses a pyramid-like structure of populations and has no parameter to tune, while the LTGA has one parameter and the LT-GOMEA is parameterless. Taking the DSMGA-II, it uses a new linkage tree called incremental linkage sets and exploits the idea of optimal mixing taken from the LT-GOMEA. At last, the 3LOa employs disturbances and local search to check which genes are dependent on one another.

When going more into the details, one can see that the LTGA takes advantage of the links between the problem's variables based on a linkage tree constructed using a hierarchical clustering algorithm, while the LT-GOMEA is based on both the linkage tree and optimal mixing using the calculation of the frequency/probability of appearance of substructures after variation. Now, as to the P3, it uses a pyramid-like structure of populations and exploits the linkage tree using cluster algorithms. The DSMGA-II employs pairwise incremental linkage model based on a dependency structure matrix and variants of optimal mixing called restricted and back mixing. Finally, the 3LOa is partially inspired by P3. It disturbs and optimises the genotype with a local optimisation algorithm. If this triggers any other genes, then they are found to be dependent.

Each solver has been assessed for solving a set of problems. The LTGA, LT-GOMEA, P3 and DSMGA-II have been applied for solving deceptive trap functions (e.g. order-k deceptive, concatenated, folded, cyclic, etc.) and nearest-neighbour NK-landscapes. Additionally, the LT-GOMEA has been used for solving the one-max problem. Also, both P3 and DSMGA-II have been applied for solving Ising spin glasses and MAX-SAT problems as well. Finally, the 3LOa is assessed using overlapping problems, concatenations of deceptive functions, hierarchical problems and discretised Rastrigin problem.

The experiments have highlighted several advantages of each technique. For example, the P3 might require, in some cases, smaller populations [74]. As for the LT-GOMEA, it appeared that the resolution time and population size are smaller than basic GA or the EDA using univariate, marginal product structures or linkage tree. Also, results showed that P3 does not require any tuning. When it comes to the DSMGA-II, experiments demonstrated that it requires fewer fitness evaluations to solve the problem than other solvers such as the LT-GOMEA, hBOA and P3 do. Finally, the 3LOa employs the 3LO (Linkage Learning based on Local Optimisation) which is a linkage learning technique that does not report false linkage.

Bearing in mind that since our proposal attempts to enhance the MMP's solving, we will be comparing our approach with techniques that were specifically designed or at least previously applied to solve the MMP. This is done to avoid any scattered comparison with other general-purpose algorithms that go beyond the scope of our work. Nonetheless, the LTGA, LT-GOMEA, P3, DSMGA-II and the 3LOa could be also an interesting direction to consider in the future so as to enhance the state-of-the-art of MMP's solving.

4. The proposed approach

In this section, we introduce the mathematical model used in our work and then we explain our proposal's search process. The source code of the latter is accessible at[2].

4.1. The studied mathematical model: Rayleigh distribution

The model we use in our work is the one proposed in [9,18]. It is based upon the cumulative function of the Rayleigh distribution. This can be described as a function R that evolves in terms of a given variable ξ, where $\xi \in [0, +\infty]$ and σ is a parameter, where $\sigma > 0$. Eq. (4.1) defines the Rayleigh probability density function, while the cumulative function of the Rayleigh distribution can be formulated using Eq. (4.2).

$$R_1(\xi) = \frac{\xi}{\sigma^2} e^{-\xi^2/(2\sigma^2)} \tag{4.1}$$

$$R_2(\xi) = 1 - e^{-\xi^2/(2\sigma^2)} \tag{4.2}$$

Fig. 4.1 represents the cumulative function of the Rayleigh distribution using several settings of the parameter σ.

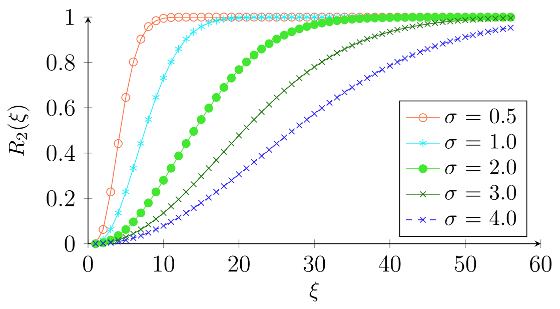

Fig. 4.1. Cumulative distribution function of the Rayleigh distribution.

As can be seen in this figure, the growth curve shape is similar to the one shown in Fig. 2.5. In addition, a systematic study was made on this function in [9,18]. The authors of those works found that using this model, more specifically the parameter's σ value, some interesting properties like the algorithm's convergence speed can be expressed. An estimation of this parameter

[2] The TD-EA source code: https://github.com/Zakaria-Dahi/TD-EA_for_MMP.git.

can be accurately computed using Eq. (4.3) and this by taking a certain number of points, S, of the Rayleigh distribution.

$$\widehat{\sigma} \approx \sqrt{\frac{1}{2S}\sum_{i=1}^{S}\xi_i^2} \tag{4.3}$$

Given this, the model based upon the Rayleigh distribution can be defined using Eq. (4.4), where $R(t)$ is the growth curve at iteration t.

$$R(t) = 1 - e^{-t^2/(2\sigma^2)} \tag{4.4}$$

4.2. Search process

The approach we present here is a Takeover Time-driven adaptive bi-phased Evolutionary Algorithm (TD-EA) to tackle the MMP. After the initialisation, the TD-EA's iteration consists of the cyclic application of selection, either an optimisation phase or a local search, evaluation and replacement. Fig. 4.2 shows the overall framework of our proposal and in the upcoming sections we provide more details about each phase.

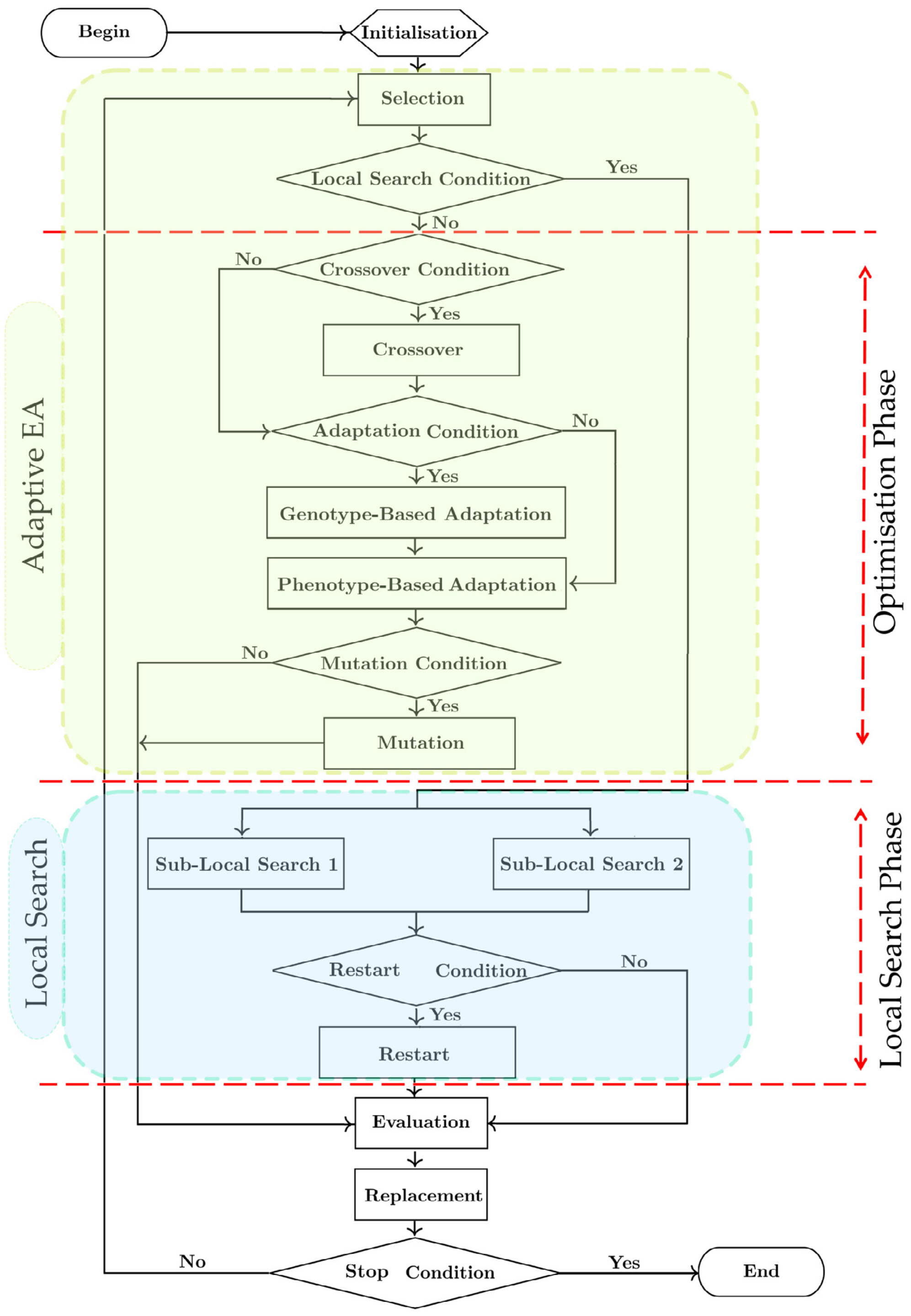

Fig. 4.2. Flowchart of the TD-EA.

4.2.1. Initialisation

The proposed TD-EA starts by generating a population of N binary individuals. Each individual $\overrightarrow{X} = \{x_1, \ldots, x_D\}$, constitutes a potential network configuration (i.e. solution), where D is the number of cells in that network. Every single bit x_j from a given individual $\overrightarrow{X}$, represents the status (reporting or non-reporting) of the j^{th} cell from the network. Fig. 4.3 exemplifies the canonical representation of a population for the MMP.

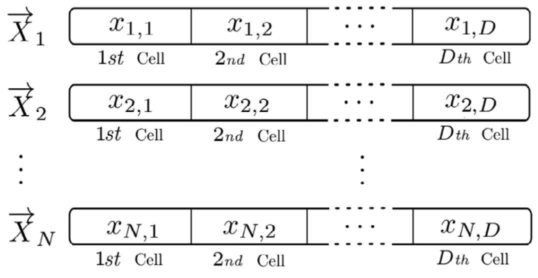

Fig. 4.3. Representation of the TD-EA's population for the MMP.

This initial population is randomly generated. After that, the quality of each individual is measured using the objective function described in Eq. (2.1). After, all the individuals are ranked according to their respective fitnesses and then the best individual, $\overrightarrow{G}$, is extracted.

It is important to state that all the parameters' values of the TD-EA have been set based on our preliminary experiments or on state-of-the-art works.

4.2.2. Selection

The selection phase is a step where a group of parents is chosen to create the couples that will undergo either the optimisation phase or the local search. If the optimisation phase is the one performed, we create $\frac{N}{2}$ couples by selecting N parents, whereas in the case the local search is the one achieved, we create $\frac{(N-M)}{2}$ couples by choosing $(N-M)$ parents $(N \gg M)$. In our case M is fixed to 100 individuals.

For both optimisation and local search phases, the parents are selected by means of binary tournament. The process is reiterated $\frac{N}{2}$ times in the case of the optimisation phase and $\frac{(N-M)}{2}$ when performing the local search. It should be kept in mind that in our implementation of the binary tournament, no condition prevents a parent from being selected more than once when creating a couple.

4.2.3. Optimisation phase

The optimisation phase represents the main core of our proposed approach. It consists in applying the mutation and crossover on the $\frac{N}{2}$ couples created during the selection phase. In addition, a key feature of the optimisation phase is the adaption of the mutation probability. The pseudo-code of Algorithm 1 illustrates the main steps of the optimisation phase which we will break down in the sections below.

Algorithm 1. The Optimisation Phase

```
1: for every created couple do
2:     Crossover.
3:     Adaptation: genotype and phenotype-based.
4:     Mutation.
5: end for
```

A. Crossover

This operator is ruled by the probability P_c. For each couple, a number ω is randomly drawn from a Standard Uniform Distribution (SUD). Afterwards, if $\omega \leqslant P_c$, the crossover is performed, otherwise the parents of that couple are copied and considered as new offspring. The crossover applied in our approach is a two-point crossover.

B. Adaptation

State-of-the-art works conducted on the GA [6,13,75] and our experiments have indicated that the principal parameter driving the convergence of the GA is the mutation probability P_m. Thus, this is the one that will undergo the adaptation.

The adaptation performed within our TD-EA is built upon the hypothesis that the efficacity of an adaptation strategy lies in its efficiency when dealing with three issues: *When to adapt* (i.e. period of application of adaptation)? *What should be the direction of adaptation* (i.e. increasing, decreasing or stabilising the value)? *What should be the amplitude of the adaptation* (i.e. by how much do we increase or decrease the value)?

Thus, in our present work we have designed a strategy that deals with all these issues. First, we define the direction of the adaptation using a *phenotype-based strategy*. Then, the amplitude of the adaptation is set using a *genotype-based strategy*. Finally, for both adaptation strategies, we define an *application time*. Every single one of these steps is now described in more detail.

B.1. Phenotype-based adaptation

The main goal of this strategy is to decide the direction in which (i.e. increasing, decreasing or stabilising) the P_m value will evolve during the succeeding iterations. For this purpose, we exploit the information contained in the fitness function (i.e. phenotype) of the produced offspring. In fact, for every two offspring previously-created by applying the crossover, a mutation is applied according to two values P_m and αP_m, respectively. This will generate four new mutated offspring which are evaluated and the best two are kept.

This process is reiterated for every selected couple. Then, at iteration t, if the number of offspring created by executing a mutation according to the probability αP_m is greater than the one produced using the probability P_m, the value of the mutation probability for iteration $t+1$ will be *increased* using Eq. (4.5). Otherwise, the mutation probability throughout the next iterations will be *decreased* using Eq. (4.6). Note that α is an amplification factor drawn within the interval [1.1,1.5], while γ is an attenuation parameter that is drawn from [0.5,0.9]. The Eqs. (4.5) and (4.6) are mainly based on some offline experiments and also inspired from the "*constant gain*" and the "*declining adaptive*" mutation schemes previously proposed in [76].

$$P_m(t+1) = P_m(t).\alpha \tag{4.5}$$

$$P_m(t+1) = P_m(t).\gamma \tag{4.6}$$

It needs to be also noted that since the effect of employing either the value P_m or αP_m can be seen immediately in the fitness value during the evaluation phase, the phenotype-based adaptation is performed in every one of the TD-EA's iterations.

B.2. Genotype-based adaptation

A fitness-driven adaptation strategy has the benefit of guiding the P_m value in the next iteration towards a value that produces better offspring (greedy evolution). However, it does not take the population diversity into account, and in the long term it makes the population collapse (converge). In fact, previous experiments allowed us to deduce that the step sizes of adaptation (γ and α) are the main factors driving the algorithm's convergence speed.

Thus, we have designed a genotype-based adaptation that rules the *amplitude of adaptation* of the mutation probability (values of γ and α) so as to prevent the population from collapsing.

This strategy is based on the information contained in the population evolution (i.e. genotype). Our approach here is similar to the mathematical oracle presented in [9,77]. The idea is to adapt the values of both variables γ and α according to the algorithm's convergence speed. The latter is the value of the parameter σ defined in Eq. (4.3) of the Rayleigh model (see Section 4.1).

Concretely, in our case, S is the number of iterations in which the variable ξ is computed. The value of the latter is the number of sub-optimal solutions in the population. It is formulated using Eq. (4.7), where τ is the number of optimal solutions in the population and N is the population's size.

$$\xi(t) = (1 - \frac{\tau}{N}).100 \tag{4.7}$$

Taking into consideration the definition of selection pressure models given in Section 2.3, the growth curve gives the number of copies of the best individual at a given iteration. However, in real methods this is not possible since only one best individual is authorised to evolve from one generation to another. So, the concept of optimal solutions becomes relative in real-life techniques. For this reason, in our approach the number of optimal solutions, τ, in the population are designated as the individuals whose fitness value is $\leqslant \eta Q(\overrightarrow{G})$ and $\geqslant Q(\overrightarrow{G})$, where η is a number > 1 and $Q(\overrightarrow{G})$ is the fitness value of the best individual in the ongoing iteration. The value of the parameter η can be set in compliance with the problem addressed, but in our case it is set to 1.02.

Then, using the value of the computed variable σ, we judge if the TD-EA is converging too fast or too slow. Depending on this judgement, the adaptation direction of both variables γ and α is decided. Firstly, we set three rules to define each state of convergence: fast, slow and stable.

- **State 1:** $\sigma < \epsilon \equiv$ *fast* convergence.
- **State 2:** $\sigma > \upsilon \equiv$ *slow* convergence.
- **State 3:** $\upsilon < \sigma < \epsilon \equiv$ *stable* convergence.

It is noteworthy to state that the variables ϵ and υ are bounds defining the algorithm's convergence state. In our case, υ is set to 60 when the size of the individual is smaller than or equal 36 bits ($D <= 36$), otherwise 50 ($D > 36$), while ϵ is set to 70 no matter what the individual's size is. Then, depending on the state of convergence, the direction of adaptation of γ and α is decided. We set three rules to manage this process.

- **Rule 1:** *fast* convergence $\Rightarrow$ decrease both γ and α using a step φ.
- **Rule 2:** *slow* convergence $\Rightarrow$ increase both γ and α by a step φ.
- **Rule 3:** *stable* convergence $\Rightarrow$ keep both γ and α constant.

For both parameters γ and α, the adaptation starts exactly from the middle of their respective intervals [0.5,0.9] and [1.1,1.5] and the adaptation step size, φ, is kept constant. It is set to the value $\frac{(0.9-0.5)/2}{(T/S)}$ for γ and $\frac{(1.5-1.1)/2}{(T/S)}$ for α, where T is the total number of iterations the TD-EA is authorised to perform. All of this is done to give a fair chance to both values of γ and α to reach, in some extreme cases of constant decreasing or increasing, the boundaries of their respective intervals.

It is essential to know that unlike the phenotype-based adaptation, the effect of the genotype-based one does not appear immediately. In fact, a change in the population convergence needs a certain number of iterations, S, in order to be noticeable. So, the genotype-based adaptation is only applied each S iteration. On the basis of the analysis that has been conducted in [9], S is set to 30 iterations. Fig. 4.4 shows an example of the resulting evolution of the mutation probability when using both phenotype and genotype-based adaptations.

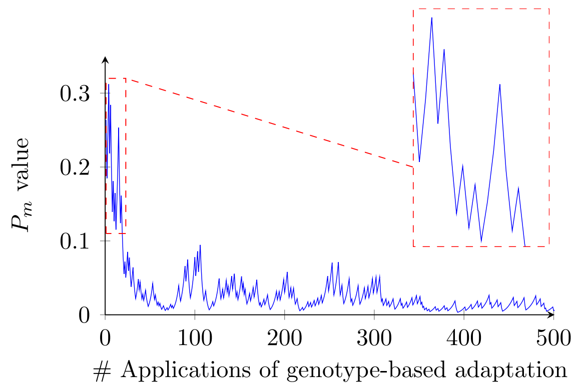

Fig. 4.4. Evolution of the P_m value when using both phenotype and genotype-based strategies.

On the basis of the state-of-the-art studies [6,75], the value of mutation probability evolves within the interval $[1/D, 0.5]$. The adaptation of the mutation probability starts from the middle of that interval. This is done to a give a fair chance to the P_m value to reach the boundaries of its interval (in case of extreme continuous increasing or decreasing evolution).

C. Mutation

In the same way as crossover, the mutation is also controlled by a probability P_m. For each two offspring produced by the crossover, a number ω is randomly drawn from an SUD. After that, if $\omega \leqslant P_m$, the mutation is executed on the current element of that offspring, otherwise it is kept as it is. The mutation used within our work is a bit-flip where a given element is mutated by flipping its value to the opposite state.

4.2.4. Local search condition

The local search phase is achieved once the population stops evolving and the algorithm stagnates. In other terms, it is achieved once the takeover time is reached (see Section 2.3). Considering the definition of the genotype-based adaptation given in Section 4.2.3, the stagnation of the population means that all the individuals in the population are considered to be optimal. Thus, the number of sub-optimal solutions, ξ, becomes null. By substituting the value of ξ in Eq. (4.3), the convergence, σ, of the algorithm also becomes null. However, a thorough study allowed us to find that in general once the $\sigma = 0$, the fitness value continues evolving over a given period. We realised that it obeys a square root function of the individual's size. In light of this, we define the condition of applying the local search as the time when $\sigma = 0$ and remains null over $S\sqrt{D}$ iterations.

Once the local search condition has been fulfilled, the optimisation phase is never performed again, and the algorithm starts performing only local search in each iteration.

4.2.5. Local search phase

It is composed of two sub-local searches that are performed in parallel. Each of them evolves a part of the population. The first sub-local search evolves $(N - M)$ individuals while the other one evolves the remaining M individuals. After the local search phase, the offspring produced are the union of the offspring created by both sub-local searches. The pseudo-code of Algorithm 2 summarises the main phases of the local search steps and throughout the sections below we provide more details about each of them.

Algorithm 2. The Local Search Phase

```
1: Perform in parallel the first and second local searches.
2: if condition of restart is fullfilled then
3:     Restart.
4: end if
```

Note that when applying the local search for the first time, the population is constituted by the best individual obtained so far in the optimisation phase and (($N - M$) - 1) new binary individuals created randomly. The remaining M individuals are created using the second sub-local search (see Section 4.2.5(B)).

A. The first sub-local search

It evolves $\frac{(N-M)}{2}$ couples created within the selection phase (see Section 4.2.2). It consists in sequentially applying a Half Uniform Crossover (HUX) to each one of those couples. This is governed by a probability P_c whose value is identical to the one utilised in the optimisation phase (see Section 4.2.3).

Considering two parents of some couple, the HUX exchanges half the elements where the two parents differ. The elements that will be exchanged are selected randomly according to a uniform distribution. To perform the HUX on a given couple, two conditions must be fulfilled. A number ω is randomly generated from an SUD. The first condition is fulfilled if $\omega \leqslant P_c$, while the other condition is that the number of elements that differ between each two parents of that couple must be greater than a threshold β. The latter was set to $\frac{D}{4}$ based on state-of-the-art studies that used the HUX [78].

B. The second sub-local search

The second local search is a heuristic we developed specifically to solve the MMP. It creates each time M new individuals necessary to fill up the population. Those individuals are copies of the best individual found in the previous iteration. However, every one of those M individuals contains a unique combination among all combinations possible for a specific neighbourhood. Six neighbourhoods with equal sizes are considered. Each neighbourhood represents a group of cells from the network. A combination of a certain neighbourhood consists of a unique set of states of the cells from that neighbourhood.

Let us first start by defining the used neighbourhoods. Given a network with a length of ψ cells. Each and every one of these six neighbourhoods is an area with a length of $\frac{\psi}{2}$ cells and 2 cells width. Fig. 4.5 shows the neighbourhoods resulting for a network of 6 cells width and 12 cells length.

Taking the first neighbourhood, {$cell_1, cell_2, cell_3, cell_4, cell_5, cell_6, cell_{13}, cell_{14}, cell_{15}, cell_{16}, cell_{17}, cell_{18}$}, as an example; a combination of this neighbourhood could be for instance {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1}. Using this neighbourhood definition, each neighbourhood contains 2^{ψ} combinations. In each iteration of the second sub-local search, we apply to all M copies of the best individual one of these combinations. This procedure is repeated in every TD-EA iteration until all combinations contained in the first neighbourhood have been browsed. Then, the same process is reiterated with the second neighbourhood, the third one and so on. Once all the neighbourhoods have been browsed, the second sub-local search is never applied again. Thus, the local search phase will consist only of the first sub-local search. Nevertheless, one should keep in mind that if the best individual changes through iterations, the second sub-local search is reiterated each time starting with the first neighbourhood.

C. Restart

It is a phase in which (N - 1) of the individuals are reinitialised. We compare the population produced in the previous iteration with the new offspring. If they are judged to be similar, the threshold β used when performing the HUX crossover in the first sub-local search is decreased by 1 each time. The restart procedure is performed once the value of β reaches 0.

The restart step consists in keeping the best individual found so far and randomly generating the (N - 1) remaining individuals. This is done according to a cataclysmic probability ϱ. The latter was set to 0.6 on the basis of some offline experiments. For each element of these (N - 1) individuals, a number ω is randomly drawn from an SUD. If ω *is lesser than or equal to* ϱ, the element will be given 1 as a value, or else it will be set to 0. It needs to be mentioned also that when performing the restart phase, the value of the threshold β is again set to $\frac{D}{4}$.

4.2.6. Evaluation

Upon completion of either the optimisation phase or the local search one, the quality of the produced offspring is measured using the function represented by Eq. (2.1).

4.2.7. Replacement

The replacement is a phase in which the composition of the population during the following iterations is decided. Our proposal uses a $(\mu + \lambda)$ generational elitist strategy. This means, the best N individuals from the union of both the offspring produced in the ongoing iteration and the population produced in the previous iteration survive and the best individual $\overrightarrow{G}$ is updated.

The termination of the replacement step marks the ending of an iteration of the TD-EA. The whole process is reiterated until the maximum number of fitness evaluations is reached.

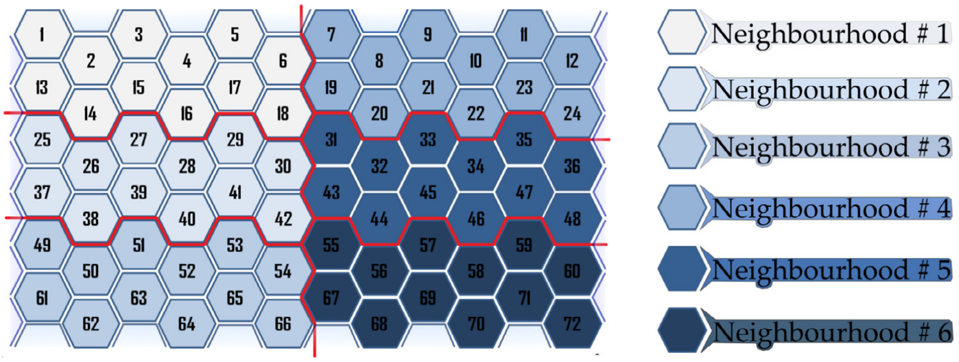

Fig. 4.5. Neighbourhood configuration for a network of 12×6 cells.

5. Experimental study and analysis

Throughout this section, first, we begin by describing the experiments conducted to assess our proposal's performances. Afterwards, we present and discuss the obtained numerical results.

5.1. Experiments' conception and settings

Our experiments are executed in a cluster composed of nineteen machines. They are all run with a Linux OS. The cluster has a computing power of ninety-four cores: sixteen machines have three cores, one machine has eight cores and two machines have forty-eight cores each. The High Throughput Computing (HTC) framework is used to manage the cluster. The implementation has been achieved using Matlab R2014a.

The proposal has been assessed based on its scalability, efficiency and robustness. This has been done by carrying the experiments over twenty-five differently-sized realistic instances (i.e. networks). In addition, the comparison has been performed against twenty-eight state-of-the-art solvers designed to address the MMP. The networks have been organised in three groups, where the first collection consists of twelve networks (three networks of each size: 4×4, 6×6, 8×8 and 10×10 cells). These instances were studied and used in [1,4,6,13,17,29–31,34,39,40,43,45,46] and are accessible in[3].

The second collection of instances includes six networks (one network of each size: 4×4, 6×6, 8×8, 7×9, 9×11 and 19 cells). They were studied in [13,28,32,33,37,38,41,42,44,46,48–50] and obtainable in [42,44,50].

The third collection of instances consists of six real-world scaled networks of sizes 12×12, 14×14, 16×16, 18×18, 20×20 and 30×30 cells. These instances were recently proposed in [13]. They were inspired from six months of real and diverse communication records that are accessible in [79]. Besides, we utilise another network of 30×30 cells. This instance has been previously studied in [34,35] and is obtainable in[4]. It is also worth mentioning that the TD-EA's source code as well as all the test cases used in our work are publicly available at[5].

Each single one of these twenty-five networks is arranged as D couples of attributes (L_i, P_i), where $i = 1, \ldots, D$ and D is the network's size. P_i and L_i represent the paging and LU costs, respectively, associated with the i^{th} cell.

To ensure a fair and reliable comparison, we used experimental settings similar to those used in all works addressing the MMP. Therefore, we considered 175 000 fitness evaluations as the stop criterion of the experiments. The proposal evolves a population of 175 individuals and was run 1000 iterations over 30 executions. All along the set of runs, various results are recorded like the worst, the best, the mean and deviation of the fitness values. It is also noteworthy that within the next sections, the words "*network*" and "*instance*" denote the same signification.

5.2. Experiments' results and analysis

All over this section, the obtained numerical results are presented. It is necessary to keep in mind that when reviewing the literature, one can notice that the authors who have used the first or the second collection of networks followed two strategies of comparison. In the *first* one, some authors rank the solvers by considering the best result they obtained over 30 runs. Thus, neither the mean nor the deviation of fitness values is reported. Alternatively, in the *second* strategy, the authors sort the algorithms by considering the mean of fitnesses achieved over all the executions.

For the purpose of ensuring an equitable comparison, the fitness deviation given in Tables 4, 5, 7 and 8 is calculated in the same way as in papers where the solvers taken as comparison basis were proposed. It is done using Eq. (5.1) and expressed as a percentage, where "*Best*" is the best value obtained and "*Mean*" is the average fitness value.

$$Dev\% = \left(\frac{Mean}{Best} - 1\right).100 \tag{5.1}$$

Since the use of the first or the second strategy of comparison depends on the availability of the information needed for each type of comparison, it is important to state that the symbol "–" in Tables 3–8 is used in case the corresponding value was not provided in the literature. Finally, the metric "*Rank*" in Tables 4, 5, 7 and 8 represents the ranking of solvers on the basis of the mean fitness value.

5.2.1. First collection of networks

Taking into account the *first* strategy, Table 3 shows the comparison between our proposal and state-of-the-art solvers. The metric "*# BRL*" designates for how many networks a given solver could reach the Best Result known in the Literature (BRL). Then again, Tables 4 and 5 present the results of the comparison according to the *second* strategy. By considering the metric "*Mean*", we highlighted, in bold, the best results.

It is worth stating also that even if the MMP is a single-objective problem, some works have already used multi-objective algorithms to solve it and have compared their results against MMP's single-objective techniques [1,4,17,29–31]. To do so, the authors of these works split the MMP's single-objective function defined in Eq. (2.1) into two objective functions defined using Eqs. (5.2) and (5.3), where the variables are the same as those used in Eq. (2.1). Then, multi-objective solvers such as NSGA-II and SPEA2 are used to tackle the MMP as a bi-objective problem. Once the solving process done, a Pareto Front is obtained. Inside the latter, the authors look for the solution(s) that best minimise(s) the same MMP's single-objective function that we use in our work (i.e. defined in Eq. (2.1) and which can also be formulated using Eq. (5.4)). For further details, please refer to the original works [1,4,17,29–31]. Considering the fact that these previously-cited works addressed the same MMP's single-objective function as the one studied in our work, we had to include in our comparison the results they report when solving the MMP using multi-objective solvers.

$$\min_{\overrightarrow{X}=\{x_1,\ldots,x_D\}} \qquad f_1(\overrightarrow{X}) = \sum_{i=1}^{\vartheta} L_i \tag{5.2}$$

$$\min_{\overrightarrow{X}=\{x_1,\ldots,x_D\}} \qquad f_2(\overrightarrow{X}) = \sum_{j=1}^{D} P_j.V_j \tag{5.3}$$

$$\min_{\overrightarrow{X}=\{x_1,\ldots,x_D\}} \qquad Q(\overrightarrow{X}) = \varsigma.f_1 + f_2 \tag{5.4}$$

From results in Table 3, one can notice that the TD-EA outperformed 11 out of 14 solvers and obtained results as good as those reached by 2 out of 14 state-of-the-art algorithms.

With due consideration to results in Tables 4 and 5, one can remark that our TD-EA was able to outperform all the other state-of-the-art techniques in 1 out of 12 networks (*largest network*: network 1 of size 10×10 cells). Here, also, the finding that needs to be noticed is that, as far as the authors know, our TD-EA is the first algorithm that reached this new best solution for this

[3] Collection 1: http://oplink.lcc.uma.es/problems/mmp.html.

[4] Collection 3-B: http://arco.unex.es/rc/.%5C%5Credes%5C%5CTN-13-30x30.txt.

[5] The TD-EA source code and test cases: https://github.com/Zakaria-Dahi/TD-EA_for_MMP.git.

Table 3
Networks of 4×4, 6×6, 8×8 and 10×10 cells: number of networks where the BRL has been reached.

Solver	Network												# BRL
	4×4 cells			6×6 cells			8×8 cells			10×10 cells			
	1	2	3	1	2	3	1	2	3	1	2	3	
NOMAD [31]	**98 535**	**97 156**	**95 038**	177 647	200 069	175 620	316 328	301 833	271 637	404 447	371 091	382 180	3
CPLEX [31]	**98 535**	**97 156**	**95 038**	181 677	200 990	186 481	375 103	351 505	407 457	514 504	468 118	514 514	3
GRASP [34]	**98 535**	**97 156**	**95 038**	175 072	184 142	175 500	316 373	299 616	270 830	402 376	369 979	383 321	3
PBIL [34]	**98 535**	97 262	**95 038**	173 804	**182 331**	175 564	313 336	292 667	265 853	390 489	362 574	378 033	4
GA [34]	**98 535**	**97 156**	**95 038**	176 032	**182 331**	176 994	312 395	294 391	265 792	391 025	364 354	378 926	4
ABC [34]	**98 535**	**97 156**	**95 038**	175 041	**182 331**	176 148	312 558	297 560	268 366	396 456	366 568	381 725	4
OI-GA [6]	**98 535**	**97 156**	**95 038**	**173 701**	**182 331**	**174 519**	311 171	**287 149**	**264 204**	387 104	359 623	372 938	8
DE [46]	**98 535**	**97 156**	**95 038**	**173 701**	**182 331**	**174 519**	308 401	**287 149**	**264 204**	386 681	358 167	371 829	9
HNN-BD [43]	**98 535**	**97 156**	**95 038**	**173 701**	**182 331**	**174 519**	308 929	**287 149**	**264 204**	386 351	358 167	**370 868**	9
SPEA2 [30]	**98 535**	**97 156**	**95 038**	**173 701**	**182 331**	**174 519**	308 702	**287 149**	**264 204**	386 721	358 392	**370 868**	9
GPSO [43]	**98 535**	**97 156**	**95 038**	**173 701**	**182 331**	**174 519**	308 401	**287 149**	**264 204**	385 972	359 191	**370 868**	9
SS [39]	**98 535**	**97 156**	**95 038**	**173 701**	**182 331**	**174 519**	**307 695**	**287 149**	**264 204**	**385 927**	357 714	**370 868**	11
NSGA-II [1]	**98 535**	**97 156**	**95 038**	**173 701**	**182 331**	**174 519**	308 702	**287 149**	**264 204**	**385 927**	**357 368**	**370 868**	11
TD-EA	**98 535**	**97 156**	**95 038**	**173 701**	**182 331**	**174 519**	**307 695**	**287 149**	**264 204**	386 474	**357 368**	**370 868**	11
2SA-cGA [13]	**98 535**	**97 156**	**95 038**	**173 701**	**182 331**	**174 519**	**307 695**	**287 149**	**264 204**	**385 927**	**357 368**	**370 868**	12

Table 4
Networks 1–3 of 4×4 and 6×6 cells: best, worst, average and fitness deviation.

Network			Solver								
			OI-GA [6]	HNN-BD [43]	GPSO [43]	CPLEX [31]	NOMAD [31]	NSGA-II [1]	SPEA2 [30]	2SA-cGA [13]	TD-EA
4×4 cells	1	Best	98 535	98 535	98 535	98 535	98 535	98 535	98 535	98 535	98 535
		Worst	98 535	–	–	–	–	–	–	98 535	98 535
		Mean	**98 535.00**	98 627.00	**98 535.00**	**98 535.00**	100 366.00	**98 535.00**	**98 535.00**	**98 535.00**	**98 535.00**
		Dev (%)	–	0.09	0.00	0.00	2.54	0.00	0.00	0.00	0.00
		Rank	1	2	1	1	3	1	1	1	1
	2	Best	97 156	97 156	97 156	97 156	97 156	97 156	97 156	97 156	97 156
		Worst	97 156	–	–	–	–	–	–	97 156	97 156
		Mean	**97 156.00**	97 655.00	**97 156.00**	**97 156.00**	100 065.00	**97 156.00**	**97 156.00**	**97 156.00**	**97 156.00**
		Dev (%)	–	0.51	0.00	0.00	2.92	0.00	0.00	0.00	0.00
		Rank	1	2	1	1	3	1	1	1	1
	3	Best	95 038	95 038	95 038	95 038	95 038	95 038	95 038	95 038	95 038
		Worst	95 038	–	–	–	–	–	–	95 038	95 038
		Mean	**95 038.00**	95 751.00	**95 038.00**	**95 038.00**	100 131.00	**95 038.00**	**95 038.00**	**95 038.00**	**95 038.00**
		Dev (%)	–	0.75	0.00	0.00	4.30	0.00	0.00	0.00	0.00
		Rank	1	2	1	1	3	1	1	1	1
6×6 cells	1	Best	173 701	173 701	173 701	181 677	177 647	173 701	173 701	173 701	173 701
		Worst	173 701	–	–	–	–	–	–	173 701	175 241
		Mean	**173 701.00**	174 690.00	174 090.00	181 677.00	189 263.00	**173 701.00**	**173 701.00**	**173 701.00**	173 855.00
		Dev (%)	–	0.56	0.22	0.00	4.44	0.00	0.00	0.00	0.09
		Rank	1	4	3	5	6	1	1	1	2
	2	Best	182 331	182 331	182 331	200 990	200 069	182 331	182 331	182 331	182 331
		Worst	182 331	–	–	–	–	–	–	182 331	182 331
		Mean	**182 331.00**	182 430.00	**182 331.00**	200 990.00	221 889.00	**182 331.00**	**182 331.00**	**182 331.00**	**182 331.00**
		Dev (%)	0.00	0.05	0.00	0.00	6.27	0.00	0.00	0.00	0.00
		Rank	1	2	1	3	4	1	1	1	1
	3	Best	174 519	174 519	174 519	186 481	175 620	174 519	174 519	174 519	174 519
		Worst	174 519	–	–	–	–	–	–	174 519	174 519
		Mean	**174 519.00**	176 050.00	175 080.00	186 481.00	182 289.00	174 605.00	174 711.00	**174 519.00**	**174 519.00**
		Dev (%)	–	0.87	0.32	0.00	2.66	0.13	0.19	0.00	0.00
		Rank	1	5	4	7	6	2	3	1	1

instance. Moreover, in 6 out of 12 instances (all networks of size 4×4 cells, network 2 and 3 of size 6×6 cells and network 2 of size 8×8 cells), our TD-EA could get results as good as those obtained by the best state-of-the-art algorithm.

Finally, in 5 out of 12 networks (network 1 of sizes 6×6 and 8×8 cells, network 3 of size 8×8 cells, networks 2 and 3 of size 10×10 cells), our approach is outperformed by only one of the state-of-the-art algorithms. Again, it must be brought to light that in those 5 out of 12 instances, the TD-EA is outperformed by a different algorithm each time. Indeed, no algorithm could outperform our approach in all these instances, while our TD-EA is always ranked at a competitive position no matter the algorithm that has outperformed it or the instances being solved are.

5.2.2. *Second collection of networks*

Similarly to the first collection of networks, various works that studied the second collection of networks also used either the first or the second strategy of comparison. Therefore, as we did in the previous section, Table 6 uses the first strategy (i.e. the best solutions achieved) to compare our proposal against state-of-the-art-algorithms, while Table 7 uses the second strategy (i.e. the mean of the fitness values).

It is important to keep in mind that the majority of works that studied the second collection of networks used a modified objective function known as *cost per call arrival*. This is described using Eq. (5.5), where $Q(\overrightarrow{X})$ defines the objective function formulated by Eq. (2.1), D is the number of cells in the network and P_i is the

Table 5
Networks 1-3 of 8×8 and 10×10 cells: best, worst, average and fitness deviation.

Network			Solver								
			OI-GA [6]	HNN-BD [43]	GPSO [43]	CPLEX [31]	NOMAD [31]	NSGA-II [1]	SPEA2 [30]	2SA-cGA [13]	TD-EA
8×8 cells	1	Best	311 171	308 929	308 401	375 103	316 328	308 702	308 702	307 695	307 695
		Worst	312 925	–	–	–	–	–	–	312 792	312 355
		Mean	312 308.38	311 351.00	310 062.00	375 103.00	326 008.00	308 859.00	**308 822.00**	310 571.87	311 213.32
		Dev (%)	–	0.78	0.53	0.00	2.19	0.05	0.06	0.95	1.14
		Rank	7	6	3	9	8	2	1	4	5
	2	Best	287 149	287 149	287 149	351 505	301 833	287 149	287 149	287 149	287 149
		Worst	289 771	–	–	–	–	–	–	287 149	287 149
		Mean	287236.41	**287 149.00**	287 805.00	351 505.00	312 497.00	**287 149.00**	**287 149.00**	**287 149.00**	**287 149.00**
		Dev (%)	–	0.00	0.22	0.00	2.59	0.00	0.00	0.00	0.00
		Rank	2	1	3	5	4	1	1	1	1
	3	Best	264 204	264 204	264 204	407 457	271 637	264 204	264 204	264 204	264 204
		Worst	265 324	–	–	–	–	–	–	264 353	264 786
		Mean	264 533.16	264 695.00	264 475.00	407 457.00	289 309.00	264 396.00	264 279.00	**264 257.47**	264 288.83
		Dev (%)	–	0.18	0.10	0.00	3.09	0.09	0.07	0.02	0.03
		Rank	6	7	5	9	8	4	2	1	3
10×10 cells	1	Best	387 104	386 351	385 972	514 504	404 447	385 927	386 721	385 927	386 474
		Worst	394 176	–	–	–	–	–	–	388 954	389 441
		Mean	390 531.47	387 820.00	387 825.00	514 504.00	415 740.00	387 416.00	387 764.00	387 148.03	**386 993.11**
		Dev (%)	–	0.38	0.48	0.00	1.82	0.20	0.22	0.32	0.27
		Rank	7	5	6	9	8	3	4	2	1
	2	Best	359 623	358 167	359 191	468 118	371 091	357 368	358 392	357 368	357 368
		Worst	370 021	–	–	–	–	–	–	361 299	361 575
		Mean	362 320.88	359 036.00	359 928.00	468 118.00	379 725.00	**358 777.00**	359 077.00	359 050.73	359 504.77
		Dev (%)	–	0.24	0.20	0.00	1.31	0.16	0.12	0.47	0.60
		Rank	7	2	6	9	8	1	4	3	5
	3	Best	372 938	370 868	370 868	514 514	382 180	370 868	370 868	370 868	370 868
		Worst	382 155	–	–	–	–	–	–	378 121	377 255
		Mean	376 900.28	374 205.00	373 722.00	514 514.00	391 627.00	371 349.00	**371 331.00**	372 854.10	374 097.70
		Dev (%)	–	0.89	0.76	0.00	0.81	0.15	0.10	0.54	0.94
		Rank	7	6	4	9	8	2	1	3	5

Table 6
Networks of 4×4, 6×6, 8×8, 7×9, 9×11 and 19 cells: number of BRLs reached.

Solver	Network						
	4×4 cells	6×6 cells	8×8 cells	7×9 cells	9×11 cells	19 cells	# BRL
HNN-BD [26,44] (*we report the results given in* [46])	–	–	–	123 474	243 414	–	0 among 2
GA [50] (*we report the results given in* [46])	92 883	229 556	436 283	–	–	–	0 among 3
ACO [50] (*we report the results given in* [46])	92 883	211 291	436 886	–	–	–	0 among 3
DE [46]	92 883	211 278	436 269	120 904	243 957	–	0 among 5
TS [50] (*we report the results given in* [46])	92 883	211 278	436 283	–	–	–	1 among 3
BABA [27]	**85 165**	214 312	459 962	–	–	–	1 among 3
SA [42]	–	–	–	–	–	**5239**	1 among 1
GA-SA [28]	92 882	**211 273**	**436 030**	–	–	–	1 among 1
SS [39]	92 833	211 278	436 269	**120 052**	**242 914**	–	2 among 5
2SA-cGA [13]	**85 165**	214 312	458 474	123 473	242 990	**5239**	2 among 6
TD-EA	**85 165**	214 312	458 474	123 473	242 990	**5239**	2 among 6

paging load associated with the ith cell.

$$\min_{\vec{X}=\{x_1,\dots,x_D\}} \quad \Theta(\vec{X}) = \frac{Q(\vec{X})}{\sum_{i=1}^{D} P_i} \tag{5.5}$$

Giving consideration to Table 6 results, it is possible to observe that our TD-EA outperforms 9 out of 10 state-of-the-art techniques by reaching 2 out of 6 BRLs. In addition, the TD-EA is one of the three algorithms in the literature able to get a new best result for the instance of 4×4 cells. Also, as far as we know, our proposed TD-EA is one of the two algorithms in the literature capable to attain 2 out of 6 BRLs.

Looking at the results in Table 7, one can notice that in 2 out of 5 instances (networks with sizes 4×4 and 7×9 cells), our proposal outperforms all state-of-the-art algorithms, while in 3 out of 5 networks (instances with sizes 6×6, 8×8 and 9×11 cells), the state-of-the-art solvers could outperform our TD-EA.

5.2.3. *Third collection of networks*

For the sake of performing a more complete analysis, we have selected, based on the results in Tables 3–5, some of the best approaches devised to solve the MMP: the GPSO [43], the OI-GA [6], the DE [46] and the 2SA-cGA [13] to compare them against our proposal. All algorithms were executed in our hardware platform to attempt solving the most realistic and complex scenarios in our benchmark (the third collection of instances). Also, in this same line of thoughts, one should bear in mind that network 1 of size 30×30 cells was inspired from six months of real and diverse communication records [13], while network 2 of similar size was previously researched in [34,35].

Table 8 shows the outcome of the aforementioned comparison. Based on the metric "*Mean*", we highlight the best results in bold. It is paramount to state that the parameters of GPSO, OI-GA, DE and 2SA-cGA used in our experiment are those utilised in the original papers. In addition, the fitness deviation is computed using Eq. (5.1) and is expressed in percentage. One can observe, in this table, that our TD-EA could be competitive to state-of-the-art algorithms in 6 out of 7 networks. Now, the noteworthy thing

Table 7
Networks of 4×4, 6×6, 8×8, 7×9 and 9×11 cells: best, worst, average and fitness deviation.

Network		Solver												
		GA [50]	GA-PSO [41]	GSO [32]	DGSO [32]	BPSO [33]	PBVM [38]	TS [50]	ACO [50]	DE [48]	DEL [48]	IDE [48]	2SA-cGA [13]	TD-EA
4×4 cells	Best	12.252	–	12.252	12.252	–	12.252	12.252	12.252	–	–	–	11.234	11.234
	Worst	12.373	–	12.252	12.252	–	12.273	12.252	12.252	–	–	–	11.234	11.234
	Mean	12.253	–	12.252	12.252	–	12.255	12.252	12.252	–	–	–	**11.234**	**11.234**
	Dev (%)	0.006	–	0.000	0.000	–	0.008	0.000	0.000	–	–	–	0.000	0.000
	Rank	3	–	2	2	–	4	2	2	–	–	–	1	1
6×6 cells	Best	11.471	–	11.426	11.426	11.471	11.471	11.471	11.471	11.471	11.471	11.471	11.636	11.636
	Worst	12.030	–	11.471	11.471	11.471	11.573	11.471	11.471	11.471	11.471	11.471	11.636	11.636
	Mean	11.511	–	11.456	**11.432**	11.471	11.474	11.471	11.471	11.471	11.471	11.471	11.636	11.636
	Dev (%)	0.343	–	0.853	0.676	0.000	0.014	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Rank	5	–	2	1	3	4	3	3	3	3	3	6	6
8×8 cells	Best	13.782	13.782	13.782	13.782	13.782	13.782	13.782	13.782	13.782	13.782	13.782	14.483	14.483
	Worst	14.671	13.947	14.102	13.883	13.893	14.042	13.999	14.007	13.923	13.892	13.782	14.495	14.483
	Mean	14.005	13.829	13.791	**13.780**	13.793	13.809	13.791	13.860	13.798	13.784	13.782	14.486	14.483
	Dev (%)	1.619	0.000	0.497	0.037	0.034	0.045	0.071	0.569	0.116	0.014	0.000	0.019	0.000
	Rank	10	8	4	1	5	7	4	9	6	3	2	12	11
7×9 cells	Best	–	–	–	–	34.538	–	–	–	–	–	–	34.538	34.538
	Worst	–	–	–	–	35.873	–	–	–	–	–	–	34.681	34.538
	Mean	–	–	–	–	34.576	–	–	–	–	–	–	34.549	**34.538**
	Dev (%)	–	–	–	–	0.142	–	–	–	–	–	–	0.036	0.000
	Rank	–	–	–	–	3	–	–	–	–	–	–	2	1
9×11 cells	Best	–	–	–	–	42.969	–	–	–	–	–	–	42.969	42.969
	Worst	–	–	–	–	44.633	–	–	–	–	–	–	43.515	44.413
	Mean	–	–	–	–	43.423	–	–	–	–	–	–	**43.171**	43.479
	Dev (%)	–	–	–	–	0.346	–	–	–	–	–	–	0.145	0.366
	Rank	–	–	–	–	2	–	–	–	–	–	–	1	3

Table 8
Networks of 12×12–30×30 cells: best, worst, average and fitness deviation.

Network		Solver				
		OI-GA	GPSO	DE	2SA-cGA	TD-EA
12×12 cells	Best	15 023	16 112	14 767	14 767	14 767
	Worst	15 440	17 277	14 806	14 780	14 885
	Mean	15 183.20	16 693.73	14 775.40	**14 767.87**	14 785.77
	Dev (%)	1.07	3.61	0.06	0.01	0.13
	Rank	4	5	2	1	3
14×14 cells	Best	18 454	21 838	17 308	17 308	17 308
	Worst	19 158	23 233	17 358	17 395	17 631
	Mean	18 762.97	22 509.67	17 333.40	**17 329.43**	17 398.60
	Dev (%)	1.67	3.08	0.15	0.12	0.52
	Rank	4	5	2	1	3
16×16 cells	Best	26 010	33 479	23 200	23 199	23 195
	Worst	28 562	38 070	23 394	23 304	23 885
	Mean	27 310.27	35 599.10	23 254.90	**23 239.60**	23 312.27
	Dev (%)	5.00	6.33	0.24	0.18	0.51
	Rank	4	5	2	1	3
18×18 cells	Best	32 794	43 084	26 257	26 257	26 257
	Worst	35 747	47 587	26 429	26 325	27 429
	Mean	34 343.17	45 605.73	26 326.40	**26 278.33**	26 371.60
	Dev (%)	4.72	5.85	0.27	0.08	0.44
	Rank	4	5	2	1	3
20×20 cells	Best	42 779	54 510	32 486	32 303	32 397
	Worst	46 139	62 318	32 742	32 593	33 308
	Mean	44 537.10	59 233.03	32 554.47	**32 424.50**	32 605.97
	Dev (%)	4.11	8.66	0.21	0.38	0.65
	Rank	4	5	2	1	3
30×30 (1) cells	Best	117 342	153 693	58 944	61 874	59 150
	Worst	130 824	184 784	59 462	64 102	61 269
	Mean	121 977.20	171 554.73	**59 121.00**	62 982.67	60 058.53
	Dev (%)	3.95	11.62	0.30	1.79	1.54
	Rank	3	4	1	3	2
30×30 (2) cells	Best	6 880 903	8 374 708	5 551 632	5 504 949	5 420 485
	Worst	7 378 851	10 052 437	5 604 376	5 581 954	5 472 371
	Mean	7 136 752.50	9 341 891.47	5 584 741.43	5 551 192.63	**5 442 089.43**
	Dev (%)	3.72	11.55	0.60	0.84	0.40
	Rank	4	5	3	2	1

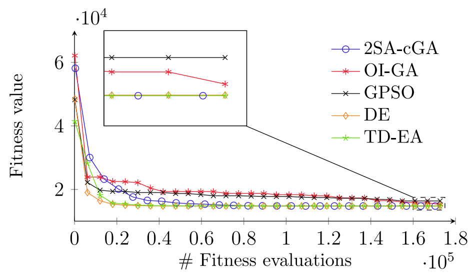

Fig. 5.1. Network with size of 12×12 cells.

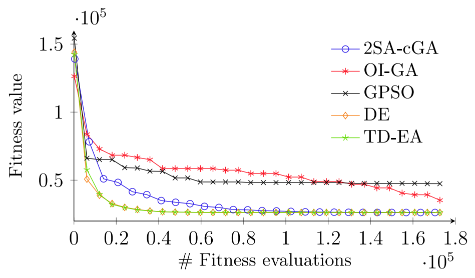

Fig. 5.4. Network with size of 18×18 cells.

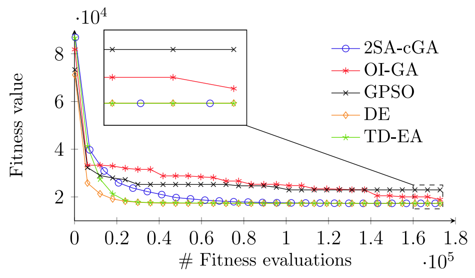

Fig. 5.2. Network with size of 14×14 cells.

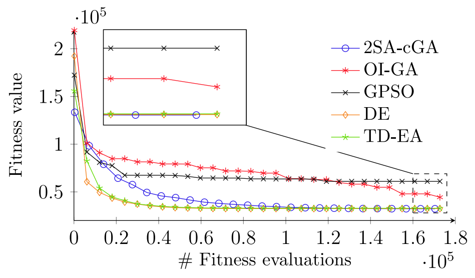

Fig. 5.5. Network with size of 20×20 cells.

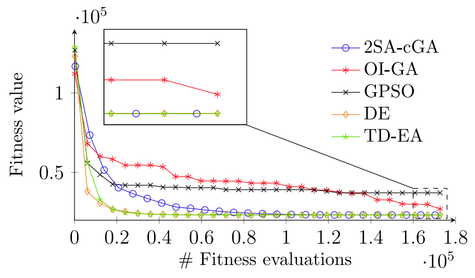

Fig. 5.3. Network with size of 16×16 cells.

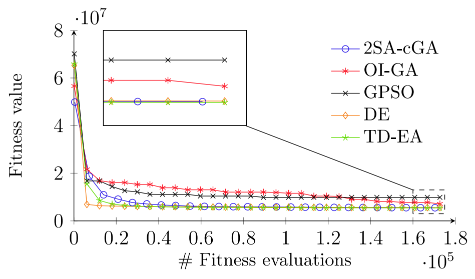

Fig. 5.6. Network with size of 30×30 (2) cells.

to be noticed is that the TD-EA was capable of outperforming all state-of-the-art techniques in the *largest* instance used in our study (900 cells). Besides, all the attained results are significant like we will see in Section 5.3.

Figs. 5.1–5.6 exemplify the fitness value evolution in the best execution performed by the TD-EA, 2SA-cGA, DE, OI-GA and the GPSO for the different networks of this set. We can note in these figures that the algorithms' behaviour is not the same for all instances. In fact, GPSO has a swift convergence speed that becomes slow (even stagnates) after some iterations. During the first iterations, the OI-GA's convergence velocity is slow when compared with the TD-EA or the GPSO. Nonetheless, while GPSO starts to converge, the OI-GA keeps on evolving discontinuously (i.e. with stagnation gaps). Unlike either OI-GA or GPSO, our TD-EA and the 2SA-cGA have regular and continuous convergence rates.

Now, regarding the MMP's features, besides being an NP-hard, multimodal binary problem, the results in Tables 3–5 and 8 show the high scalability of its instances, where most solvers have a tendency to efficiently solve small-sized networks and fail to address average and high-dimensional ones. Also, on the basis of Tables 6–8, even for small-sized problems, it appears that it is not guaranteed to efficiently solve them. Actually, one can note that most optimisers fail to tackle the smallest networks. This might point out the high complexity (e.g. multimodality, trays, etc.) of the MMP independently of the size of the network at hand. Bearing in mind these properties, our proposal could be efficient when applied on problems with features similar to those of the MMP. Except for these facts, very few can be said about the MMP's characteristics. Indeed, making reliable statements about it implies a sophisticated theory and experiments on fitness and landscape analysis. Indeed, the amount of work and its purpose go beyond the topic of our contribution and should be treated in a separated paper.

5.3. Statistical tests, results and analysis

All along this section, thorough statistical tests are carried out to verify the statistical correctness of the results given in Table 8. The distribution normality and variance homogeneity of the data samples are checked using the Kolmogorov–Smirnov and the Bartlett tests, respectively. In view of the results of both aforementioned tests, we apply a Kruskal–Wallis as a variance analysis test. This is done to discover whether the means of the results got by every algorithm are different or not. Taking into account the results of this last test, we apply a post-hoc test to discover where the difference occurs and eventually find which algorithm is the best.

All conducted tests are hypothesis tests, where S_T represents the result of the statistical test, P_V is the probability value returned by the test and ζ is the significance level. All the tests are executed using $\zeta = 5\%$ as a level of significance, and all are applied to a sample of data. The latter represents the results achieved by every algorithm for each instance over 30 executions.

5.3.1. Analysis of variance: Kruskal–Wallis test

Table 9 gives the results found after performing a Kruskal–Wallis test, where one can remark that for all networks' sizes, H_0 has been rejected. Thus, a post-hoc test needs to be carried to find out where the difference occurs and eventually deduce which algorithm is the best.

Table 9
ANOVA test: the Kruskal–Wallis results.

Size	P_V	S_T	Null-hypothesis
12×12	$1.3638\ 10^{-26}$	127.4542	Rejected
14×14	$8.6423\ 10^{-27}$	128.3809	Rejected
16×16	$4.4679\ 10^{-24}$	115.6799	Rejected
18×18	$2.5144\ 10^{-25}$	121.5319	Rejected
20×20	$1.6754\ 10^{-26}$	127.0363	Rejected
30×30 (1)	$9.0572\ 10^{-30}$	142.3056	Rejected
30×30 (2)	$1.111\ 10^{-29}$	141.8897	Rejected

5.3.2. Post-Hoc test

To verify its null-hypothesis, the post-hoc executes a sequence of comparisons between some reference algorithm (A) and a second algorithm (B). Table 10 presents the obtained results.

Regarding the post-hoc test results, one can remark that they confirm some of the results in Table 8. However, they provide a clearer and more reliable interpretation of other results. Indeed, one of the important things to be noticed is that in 5 out of 7 instances highlighted in green (12×12, 16×16, 18×18, 30×30 (1) and 30×30 (2)), the post-hoc test failed to reject the null-hypothesis that both 2SA-cGA and TD-EA are achieving the same set of results. This supports the evidence that both algorithms are having equal efficiency in terms of fitness value.

The same thing goes for the TD-EA and DE. In 5 out of 7 instances highlighted in green (12×12, 16×16, 18×18, 20×20 and 30×30 (1)), the post-hoc test could not reject the null-hypothesis that the results attained by both algorithms are similar. This fact, again, supports the idea that both DE and TD-EA are achieving the same efficiency in terms of fitness value.

6. TD-EA vs. 2SA-cGA: A comparative study

In this section, we carry out a thorough theoretical and numerical comparison between the approach we present here (TD-EA) and the best solver (2SA-cGA) [13], encountered across the literature.

6.1. Theoretical comparison

Both works, the one we conduct here and the one made in [13], aim at devising new efficient solvers for the MMP. So, in order to perform methodical, correct and constructive research and for a reliable comparison with state-of-the-art solvers, it is mandatory and unavoidable for us to follow, here, the same experimental settings as those used in [13]. Technically speaking, it is obligatory that in both works we use the same mathematical formulation of the MMP, the same experimental parametrising, benchmarks, comparison metrics, etc. Now, when going back to the main contribution of both works: the algorithm we present here (TD-EA) and the one devised in [13] (2SA-cGA), both algorithms are totally independent and completely different from each other at several levels. On the basis of the explanations given all over both works and especially in Sections 4–6 of our paper and Sections 3.2-3.10 and 4 in [13], we can cite, as an example among many, of where these differences occur :

First, the "*algorithms' type*". The TD-EA we devise here is a panmictic evolutionary algorithm, while the 2SA-cGA presented in [13] is a cellular genetic algorithm. The TD-EA evolves a non-structured population using a hybrid solver that combines an evolutionary algorithm with a local search, while the 2SA-cGA evolves a structured population (toroidal-grid-like shaped) using a genetic algorithm. On the basis of this first fact, many other differences are noticeable between both proposals: (1) their structure, (2) search processes, (3) search operators, (4) paradigms employed in each of them, (5) their parametrising, (6) the novelties and contributions they contain, (7) their research conclusions, etc. Let us, in what is coming, give further details about these dissimilarities.

The "*structure*" and how the TD-EA and 2SA-cGA are built are totally different. This includes their architecture, the sequencing and the order in which their search operators are executed. This leads us to the second point of divergence between the TD-EA and 2SA-cGA, which is their "*search operators*". Actually, the latter is one of the principal origins in the difference of structure. Both the TD-EA and 2SA-cGA use search operators that are different in terms of number (i.e. how many), type (i.e. the working mechanism) and placement (i.e. position inside the algorithm), etc. Although, since both proposals can be classified as evolutionary metaheuristics, naturally, the only point where both approaches seem to partially join is the use of some "*concepts*" related to evolutionary computing. These concepts are the initialisation, selection, crossover, mutation, replacement and the application of parameter's adaptation at some moment in their execution. Still, it is very important to note that these are "*high-level*" common features, which means that it is nothing more than sharing the "*concept*" or the "*naming*". Indeed, even here, both the TD-EA and 2SA-cGA perform such phases in a way that completely differ from one another. As a matter of fact, technically speaking, "*When*", "*Where*" and "*How*" these steps are done is absolutely different in each algorithm.

Considering the difference in their structures and search operators, another dissimilarity between the TD-EA and 2SA-cGA can also be noted in their "*search processes*" which is a key element in search algorithms. As a matter of fact, this means that both solvers' working mechanism and how they perform their search is completely different. Now, with regard to all the aforementioned divergence points that exist between both approaches, we can find that they are the source of other fundamental differences in the "*paradigms employed*" and those on which the TD-EA and 2SA-cGA are based. This includes the philosophy of search, structuring the population, hybridisation of algorithms and so on. Also, the previously-cited dissimilarities are caused by those same differences in the paradigms employed in the TD-EA and 2SA-cGA. Probably one of the most noticeable and undeniable ones is the parameters' adaptation. In fact, the 2SA-cGA adapts its parameter on the basis of some basic offline formula, while the TD-EA evolves its parameter on the basis of the future expected behaviour of the algorithm and this using a mathematical oracle based on two concepts: growth curve and takeover time.

Another important aspect in evolutionary computing is the algorithm "*parametrising*" and especially search operators at all levels. Here also, another difference can be noticed between the TD-EA and the 2SA-cGA. All of these, lead us to the differences we can find in the experimental level, which are the "*results and the research conclusions*". Indeed, it is clear that both proposals are leading, in their own way, to separate, different and independent research findings. Now, of course many other differences can be found between both the solver we present here and the one devised in [13], but just bearing in mind all the above dissimilarities, one can note one final major difference between both works : the "*novelty*" and the "*contribution they contain*". Table 11 summarises some of the principal similarities and differences that can be found between both the work we conduct here and the one made in [13].

Table 10
Post-hoc test results.

Algorithm A	Algorithm B	12×12	14×14	16×16	18×18	20×20	30×30 (1)	30×30 (2)
TD-EA	OI-GA	$3.00\ 10^{-06}$	$1.14\ 10^{-02}$	$8.22\ 10^{-06}$	$3.07\ 10^{-06}$	$9.59\ 10^{-05}$	$5.96\ 10^{-07}$	$1.03\ 10^{-14}$
TD-EA	GPSO	$2.38\ 10^{-14}$	$2.99\ 10^{-08}$	$2.85\ 10^{-13}$	$6.43\ 10^{-14}$	$1.24\ 10^{-11}$	$5.75\ 10^{-15}$	$1.05\ 10^{-25}$
TD-EA	2SA-cGA	$1.00\ 10^{-00}$	$4.89\ 10^{-03}$	$1.00\ 10^{-00}$	$9.86\ 10^{-01}$	$2.99\ 10^{-02}$	$6.04\ 10^{-02}$	$5.31\ 10^{-02}$
TD-EA	DE	$1.00\ 10^{-00}$	$4.89\ 10^{-02}$	$1.00\ 10^{-00}$	$1.00\ 10^{-00}$	$1.00\ 10^{-00}$	$1.13\ 10^{-01}$	$1.64\ 10^{-06}$
OI-GA	GPSO	$5.17\ 10^{-02}$	$7.42\ 10^{-02}$	$7.48\ 10^{-02}$	$7.45\ 10^{-02}$	$7.49\ 10^{-02}$	$7.49\ 10^{-02}$	$7.49\ 10^{-02}$
OI-GA	2SA-cGA	$9.47\ 10^{-10}$	$1.57\ 10^{-10}$	$2.28\ 10^{-08}$	$1.28\ 10^{-10}$	$1.42\ 10^{-12}$	$7.49\ 10^{-02}$	$1.64\ 10^{-06}$
OI-GA	DE	$2.22\ 10^{-06}$	$1.29\ 10^{-08}$	$2.73\ 10^{-06}$	$3.08\ 10^{-04}$	$2.38\ 10^{-04}$	$1.84\ 10^{-14}$	$5.31\ 10^{-02}$
GPSO	2SA-cGA	$1.83\ 10^{-19}$	$4.58\ 10^{-20}$	$5.11\ 10^{-17}$	$3.49\ 10^{-20}$	$7.57\ 10^{-23}$	$8.85\ 10^{-07}$	$2.57\ 10^{-14}$
GPSO	DE	$1.51\ 10^{-14}$	$2.21\ 10^{-17}$	$5.46\ 10^{-14}$	$7.72\ 10^{-11}$	$5.20\ 10^{-11}$	$2.25\ 10^{-25}$	$4.72\ 10^{-07}$
2SA-cGA	DE	$1.00\ 10^{-00}$	$1.00\ 10^{-00}$	$1.00\ 10^{-00}$	$9.23\ 10^{-02}$	$1.53\ 10^{-02}$	$1.31\ 10^{-06}$	$1.43\ 10^{-01}$

Table 11
TD-EA vs. 2SA-cGA: a theoretical comparison.

Comparison part	Comparison aspect	Different	Similar
MMP problem	Problem modelling		x
	Mathematical formulation		x
Experimental validation	Experiments' settings		x
	Comparison metrics		x
	Benchmark instances		x
	# SOTA the proposal is compared with	x	
Contribution/Proposal	Algorithm's type	x	
	Algorithm's structure	x	
	Algorithm's search operators	x	
	Algorithm's search process	x	
	Paradigms employed/ based on	x	
	Parametrising	x	
	Results and research conclusions/findings	x	
	Novelties and contributions	x	

6.2. Numerical comparison

The numerical comparison between the TD-EA and 2SA-cGA has been done over all the 25 networks of the 1st, 2nd and 3rd collections. The comparison takes into account four criteria. The first criterion is the efficiency in terms of statistical results presented in Table 10 and mean values in Tables 3–8. On the basis of the results in Tables 3–10, it is to bear in mind that regarding the best result achieved all along 30 executions, both TD-EA and 2SA-cGA achieve identical results in 20 out of 25 instances, the TD-EA outperforms the 2SA-cGA in 3 out of 25 instances and the 2SA-cGA outperforms the TD-EA in 2 out of 25 instances. Taking now the average of the results obtained throughout 30 executions, the TD-EA and the 2SA-cGA achieve similar results in 11 out of 24 instances, the TD-EA outperforms the 2SA-cGA in 5 out of 24 instances and the 2SA-cGA outperforms the TD-EA in 8 out of 24 networks. One should also keep in mind that based on the average results in 30 executions, the TD-EA outperforms the 2SA-cGA in the two largest realistic networks ever studied in the literature (900 cells): networks 1 and 2 of size 30×30 cells.

The second criterion is "*# Hits*", which represents the number of times (among 30 executions) the algorithm could reach the best result known in the literature. The third criterion is "*Execution time*". It represents the average time needed (in seconds) for the algorithm to complete one execution. The last criterion, "*FE Needed*", represents the average number of fitness evaluations needed for an algorithm to reach the best result known in the literature. It is worth noting that the metric "*FE Needed*" considers all the fitness evaluations between the "*start*" and the "*termination*" of the algorithm (whether the 2SA-cGA or the TD-EA) and it also includes the "*total*" number of fitness evaluations performed by all the algorithm's components (whether the 2SA-cGA or the TD-EA). Indeed, considering the TD-EA for instance, the "*FE Needed*" represents the average of the total number of fitness evaluations performed by all the TD-EA's components including both the optimisation and local search phases.

Table 12 and Fig. 6.1 represent the results of such comparison, where the best results are emboldened and coloured in green. Taking into account the metrics "efficiency" and "*# Hits*", both TD-EA and 2SA-cGA display approximately similar performances with a tiny superiority for the 2SA-cGA over the TD-EA. In addition, regarding the criteria "*Execution time*" and "*FE needed*", our proposed TD-EA outperforms the 2SA-cGA. Indeed, in 4 networks (16% of the cases), it reaches the BRLs more often than the 2SA-cGA does and in 14 instances (56% of the cases), the same times as 2SA-cGA does. (II) In 16 instances (64% of the cases), our proposal takes less fitness evaluations than the 2SA-cGA to reach the BRLs. (III) In 19 instances (76% of the cases), our approach execution time is shorter than the one of the 2SA-cGA.

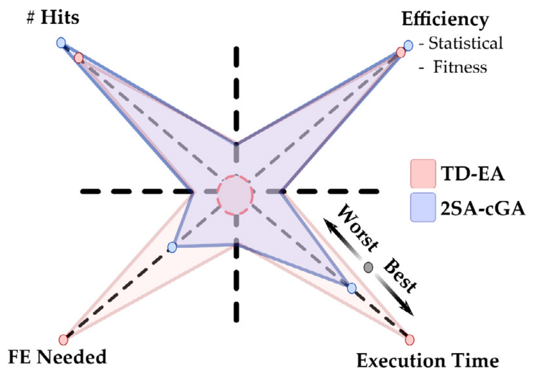

Fig. 6.1. TD-EA vs. 2SA-cGA: # hits, FE needed and execution time.

7. The TD-EA: A component efficiency analysis

In order to give further understanding of the source of the TD-EA's efficiency, we perform in this section a profound analysis of the main components that constitutes our proposal. As it can be seen in Section 4, the TD-EA has two principal constituents:

Table 12
TD-EA vs. 2SA-cGA.

Networks		Execution time (s)		# Hits (among 30 exe)		FE needed	
		TD-EA	2SA-cGA	TD-EA	2SA-cGA	TD-EA	2SA-cGA
1st collection	4×4 (1)	**630.943**	824.125	**30**	**30**	**2916.667**	11 600.000
	4×4 (2)	**739.072**	814.656	**28**	**28**	**3212.500**	14 042.857
	4×4 (3)	**691.086**	843.626	**30**	**30**	**2811.667**	10 466.667
	6×6 (1)	**2 241.385**	2 918.756	27	**30**	**23 262.037**	49 800.000
	6×6 (2)	**2 145.827**	2 832.698	**30**	**30**	**8 691.667**	42 026.667
	6×6 (3)	**2 438.858**	2 798.797	**28**	**28**	**11 112.500**	48 014.286
	8×8 (1)	**6 324.602**	6 841.234	**3**	1	**109 468.000**	114 800.000
	8×8 (2)	**4 951.019**	6 796.665	**30**	**30**	**44 881.267**	78 533.333
	8×8 (3)	**5 041.298**	6 778.263	16	**17**	**62 509.250**	101 670.590
	10×10 (1)	**8 252.369**	13 021.427	0	**1**	None	**114 800.000**
	10×10 (2)	**9 389.050**	13 238.905	**1**	**1**	171 079.000	**154 800.000**
	10×10 (3)	**10 437.926**	15 832.301	**4**	1	154 711.250	**136 800.000**
2nd collection	4×4	**121.744**	213.792	**30**	**30**	**2718.333**	6466.667
	6×6	**340.090**	595.567	**0**	**0**	None	None
	8×8	6218.159	**1 423.710**	**0**	**0**	None	None
	7×9	6747.040	**1 488.498**	**0**	**0**	None	None
	9×11	14 722.314	**11 644.408**	**0**	**0**	None	None
	19	771.853	**223.369**	**30**	**30**	**3406.667**	15 093.333
3rd collection	12×12	**15 399.197**	15 541.150	20	**28**	**78 120.000**	107 785.714
	14×14	**26 190.939**	26 748.731	1	**15**	**119 175.000**	137 600.000
	16×16	**40 286.036**	41 458.463	**3**	0	**97 591.667**	None
	18×18	**61 817.631**	62 525.031	6	**8**	**109 783.333**	166 550.000
	20×20	**93 283.923**	94 295.348	0	**1**	None	**165 200.000**
	30×30 (1)	631 712.950	**439 778.120**	**0**	**0**	None	None
	30×30 (2)	692 930.790	**453 510.530**	**1**	0	**174 825.000**	None

the "*optimisation*" and the "*local search*" phases. The aim of our study is to make some hypothesis on whether the improvements displayed by the TD-EA are due to the local search phase, the optimisation phase or both.

The results of Table 13 represent the number of times (e.g. maximum, minimum and average) the local search phase is executed throughout the 30 runs of the TD-EA. As it can be seen, in 19 out of 25 instances (76% of the cases: green-shaded), the local search is not executed. This means that the TD-EA only performs the optimisation phase and that its efficiency is due only to that component. In this same line of thoughts, we go further by trying to prove/reject such hypothesis. To do that, we executed the TD-EA using the same experimental settings employed previously in our study (see Section 5.1). The only difference this time is that we are discarding the local search phase. Technically speaking, the TD-EA will execute only the optimisation phase. In the following, we call this TD-EA's variant as TD-EA-{LS}.

The results in Table 14 are those obtained when comparing both the TD-EA and the TD-EA-{LS} on the basis of several metrics. We can cite, as an example of these criteria, the "best", the "worst" and the "Mean" of the fitness value obtained through 30 executions. The metric Δ(Q) represents the difference between the average fitness obtained by the TD-EA and the TD-EA-{LS} (see Fig. 7.1). The metrics "*# Hits*" and "*FE Needed*" stand for the number of times an algorithm (whether the TD-EA or the TD-EA-{LS}) could achieve the best result known in the literature and also what is the average of fitness evaluations needed to do so. It is also worth noting that the metric "*FE Needed*" considers all the fitness evaluations between the "*start*" and the "*termination*" of the algorithm (whether the TD-EA or the TD-EA-{LS}) and it also includes the "*total*" number of fitness evaluations performed by all the algorithm's components. Indeed, considering the TD-EA for instance, the "*FE Needed*" represents the average of the total number of fitness evaluations performed by all the TD-EA's components including both the optimisation and local search phases, while for the case of the TD-EA-{LS}, it considers only the optimisation phase.

Table 13
Number of times the local search phase is executed.

Collection	Size	Network	Min	Max	Average	STD
1st	4×4	1	0	0	0.000	0.000
		2	0	0	0.000	0.000
		3	0	0	0.000	0.000
	6×6	1	0	0	0.000	0.000
		2	0	0	0.000	0.000
		3	0	0	0.000	0.000
	8×8	1	1527	2291	1858.400	215.436
		2	1006	2113	1599.900	229.836
		3	1616	2324	2044.967	175.802
	10×10	1	939	1539	1348.767	143.570
		2	655	1061	819.567	71.796
		3	839	1743	1187.133	184.217
2nd	4×4	–	0	0	0.000	0.000
	6×6	–	0	0	0.000	0.000
	8×8	–	0	0	0.000	0.000
	19	–	0	0	0.000	0.000
	7×9	–	0	0	0.000	0.000
	9×11	–	0	0	0.000	0.000
3rd	12×12	–	0	0	0.000	0.000
	14×14	–	0	0	0.000	0.000
	16×16	–	0	0	0.000	0.000
	18×18	–	0	0	0.000	0.000
	20×20	–	0	0	0.000	0.000
	30×30	1	0	0	0.000	0.000
		2	0	0	0.000	0.000

Finally, the metric "H_0" represents the results of the statistical test, where "+" means that the null-hypothesis has been accepted (i.e. the TD-EA and TD-EA-{LS} have similar mean ranks), while "–" designates the alternative scenario. It is important to bear in mind that green-shaded cells signify that both the TD-EA and TD-EA-{LS} have similar results, while blue-shaded cells mean that the TD-EA outperforms the TD-EA-{LS} and finally red-shaded cells designates the opposite scenario.

On the basis of the metrics "Mean" and "Δ(Q)" in Table 14, one can notice that in 7 out of 25 instances, both the TD-EA and TD-EA{LS} achieve the same results. For the remaining instances, the difference is barely noticeable. Although it is worth to mention

Table 14
The TD-EA vs. TD-EA-{LS}.

Collection	Size	Network	Algorithm	*Best*	*Worst*	*Mean*	Dev (%)	Δ(Q)	# Hits	FE needed	H_0
1st	4×4	1	TD-EA	98 535.000	98 535.000	**98 535.000**	0.000	0.000	**30**	**2916.667**	+
			TD-EA-{LS}	98 535.000	98 535.000	**98 535.000**	0.000		**30**	3301.667	
		2	TD-EA	97 156.000	97 156.000	**97 156.000**	0.000	0.000	28	**3212.500**	+
			TD-EA-{LS}	97 156.000	97 156.000	**97 156.000**	0.000		**30**	3430.000	
		3	TD-EA	95 038.000	95 038.000	**95 038.000**	0.000	182.600	**30**	2811.667	+
			TD-EA-{LS}	95 038.000	100 516.000	95 220.600	0.192		29	**2456.034**	
	6×6	1	TD-EA	173 701.000	175 241.000	173 855.000	0.090	154.000	27	**23 262.037**	+
			TD-EA-{LS}	173 701.000	173 701.000	**173 701.000**	0.000		**30**	24 266.667	
		2	TD-EA	182 331.000	182 331.000	**182 331.000**	0.000	0.000	**30**	**8691.667**	+
			TD-EA-{LS}	182 331.000	182 331.000	**182 331.000**	0.000		**30**	8820.000	
		3	TD-EA	174 519.000	174 519.000	**174 519.000**	0.000	0.000	28	11 112.500	+
			TD-EA-{LS}	174 519.000	174 519.000	**174 519.000**	0.000		**30**	**10 313.333**	
	8×8	1	TD-EA	307 695.000	312 355.000	**311 213.320**	1.140	322.880	**3**	109 468.000	+
			TD-EA-{LS}	307 695.000	312 819.000	311 536.200	1.248		1	**76 475.000**	
		2	TD-EA	287 149.000	287 149.000	**287 149.000**	0.000	910.133	**30**	44 881.267	–
			TD-EA-{LS}	287 149.000	296 191.000	288 059.133	0.317		26	**31 742.308**	
		3	TD-EA	264 204.000	264 786.000	**264 288.830**	0.030	172.037	**16**	62 509.250	+
			TD-EA-{LS}	264 204.000	265 324.000	264 460.867	0.097		15	**36 761.667**	
	10×10	1	TD-EA	386 474.000	389 441.000	**386 993.110**	0.270	3354.790	0	None	–
			TD-EA-{LS}	387 543.000	397 904.000	390 347.900	0.724		0	None	
		2	TD-EA	357 368.000	361 575.000	**359 504.770**	0.600	1638.830	**1**	171 079.000	–
			TD-EA-{LS}	358 167.000	365 402.000	361 143.600	0.831		0	None	
		3	TD-EA	370 868.000	377 255.000	**374 097.700**	0.940	2519.567	**4**	154 711.250	–
			TD-EA-{LS}	370 868.000	381 160.000	376 617.267	1.550		1	**50 925.000**	
2nd	4×4	–	TD-EA	11.234	11.234	**11.234**	0.000	0.000	**30**	2718.333	+
			TD-EA-{LS}	11.234	11.234	**11.234**	0.000		**30**	**2555.000**	
	6×6	–	TD-EA	11.636	11.636	**11.636**	0.000	0.000	0	None	+
			TD-EA-{LS}	11.636	11.636	**11.636**	0.000		0	None	
	8×8	–	TD-EA	14.483	14.483	**14.483**	0.000	0.008	0	None	+
			TD-EA-{LS}	14.483	14.536	14.491	0.054		0	None	
	19	–	TD-EA	13.679	13.679	**13.679**	0.000	0.000	**30**	**3406.667**	+
			TD-EA-{LS}	13.679	13.679	**13.679**	0.000		**30**	3441.667	
	7×9	–	TD-EA	34.538	34.538	**34.538**	0.000	0.261	0	None	–
			TD-EA-{LS}	34.538	41.000	34.799	0.754		0	None	
	9×11	–	TD-EA	42.969	44.413	**43.479**	0.366	0.082	0	None	+
			TD-EA-{LS}	42.969	44.450	43.561	1.377		0	None	
3rd	12×12	–	TD-EA	14 767.000	14 885.000	14 785.770	0.130	1.003	**20**	78 120.000	+
			TD-EA-{LS}	14 767.000	14 879.000	**14 784.767**	0.120		18	**70 000.000**	
	14×14	–	TD-EA	17 308.000	17 631.000	**17 398.600**	0.520	6.867	1	119 175.000	+
			TD-EA-{LS}	17 308.000	17 529.000	17 405.467	0.563		**2**	**108 325.000**	
	16×16	–	TD-EA	23 195.000	23 885.000	23 312.270	0.510	4.337	**3**	**97 591.667**	+
			TD-EA-{LS}	23 199.000	23 801.000	**23 307.933**	0.470		0	None	
	18×18	–	TD-EA	26 257.000	27 429.000	26 371.600	0.440	39.833	**6**	**109 783.333**	+
			TD-EA-{LS}	26 257.000	26 912.000	**26 331.767**	0.285		4	147 612.500	
	20×20	–	TD-EA	32 397.000	33 308.000	32 605.970	0.650	23.070	**0**	None	+
			TD-EA-{LS}	32 370.000	33 028.000	**32 582.900**	0.658		**0**	None	
	30×30	1	TD-EA	59 150.000	61 269.000	**60 058.530**	1.540	52.603	0	None	+
			TD-EA-{LS}	58 849.000	61 343.000	60 111.133	2.145		**1**	**162 575.000**	
		2	TD-EA	5 420 485.000	5 472 371.000	5 442 089.430	0.400	3283.597	1	174 825.000	+
			TD-EA-{LS}	5 408 420.000	5 472 165.000	**5 438 805.833**	0.562		**4**	**150 587.500**	

that the largest differences are found in networks 8×8 and 10×10 of the 1st collection. This is quite logical since the results of Table 13 show that those instances are the only ones where the local search phase is executed. This supports the hypothesis that the difference in the TD-EA's efficiency might be due to the optimisation phase.

Now, on the basis of the metric "# Hits", both the TD-EA and the TD-EA-{LS} achieve the same results in 10 out of 25 instances, while the TD-EA could outperform the TD-EA-{LS} in 9 instances and the TD-EA-{LS} could beat the TD-EA in 6 out of 25 cases. Although, when considering the metric "FE needed", it can be seen that as the size of the instances increases, the TD-EA-{LS} has a tendency to outperform the TD-EA and this by reaching the best known results in the literature in much smaller number of fitness evaluations. Despite the fact that it is hard to make firm conclusions about the efficacy of stochastic algorithms, we can confirm (with a reasonable confidence) that in this particular study, the source of the efficiency of the TD-EA

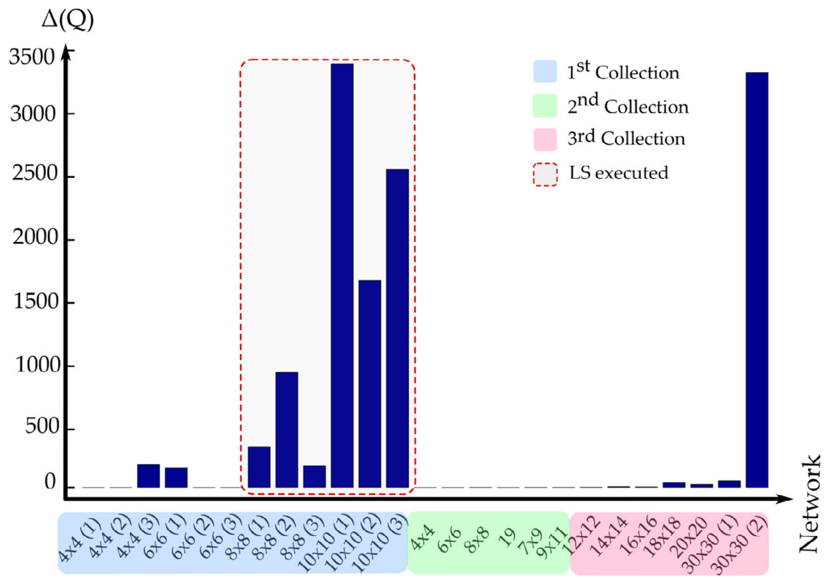

Fig. 7.1. TD-EA vs. TD-EA-{LS}: Δ(Q).

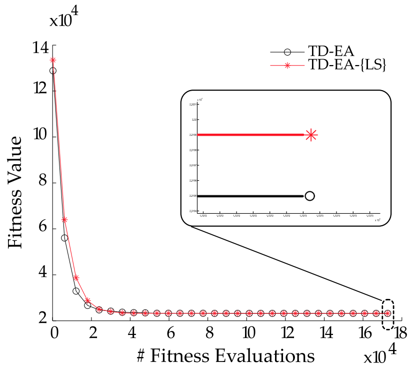

Fig. 7.2. Network 16×16 cells.

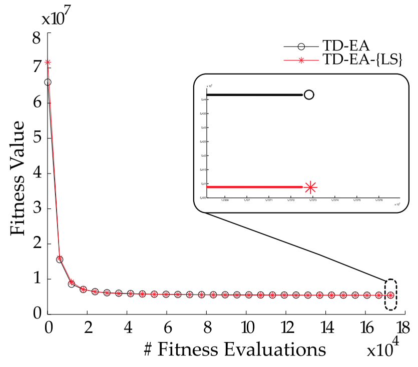

Fig. 7.4. Network 30×30 cells.

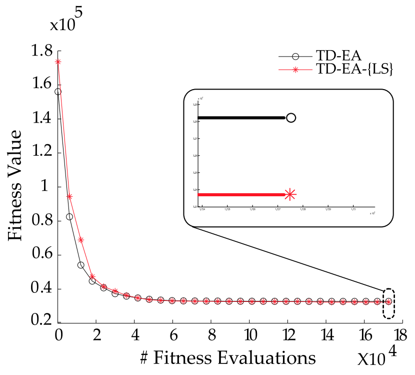

Fig. 7.3. Network 20×20 cells.

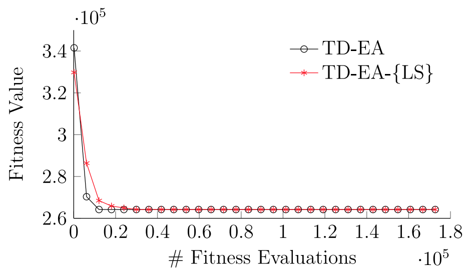

Fig. 7.5. Network 8×8 cells.

is mainly due to the optimisation phase. This being said, in some cases (e.g. networks 8×8 and 10×10 cells), it is also due to the local search phase.

Figs. 7.2–7.13 represent the fitness evolution for both the TD-EA and the TD-EA-{LS} when tackling instances of 4×4 - 30×30 cells. The figures have been created by choosing, on purpose, executions where both the TD-EA and the TD-EA-{LS} end-up reaching the same result or at least very close ones. Our goal is to analyse the behaviour of both algorithms when providing the same efficiency. As it can be seen in Figs. 7.2–7.13, the comportment of both algorithms is very similar (even identical) in most of the cases. Indeed, for almost all the networks, the TD-EA and TD-EA-{LS} display the same convergence speed and the same smooth fitness evolution at overall. All the above-mentioned facts

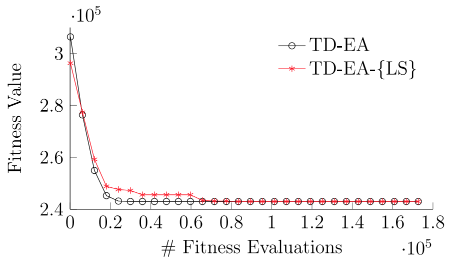

Fig. 7.6. Network 9×11 cells.

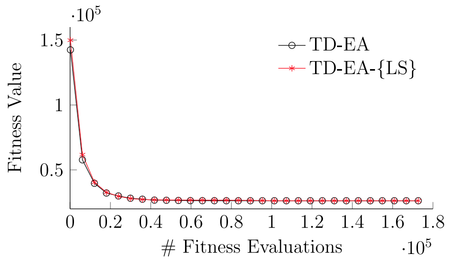

Fig. 7.7. Network 18×18 cells.

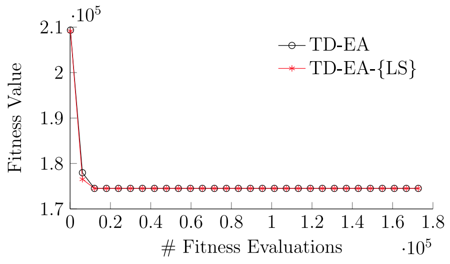

Fig. 7.8. Network 6×6 cells.

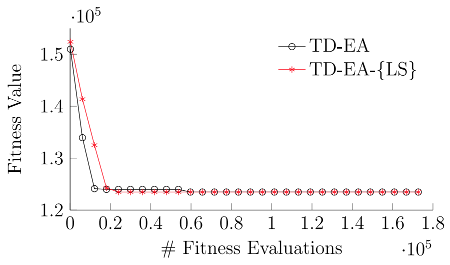

Fig. 7.9. Network 7×9 cells.

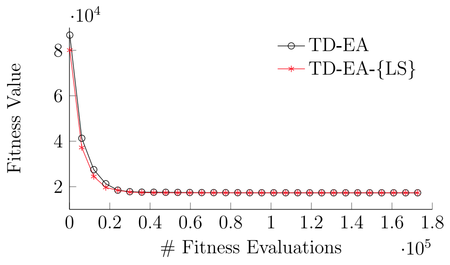

Fig. 7.10. Network 14×14 cells.

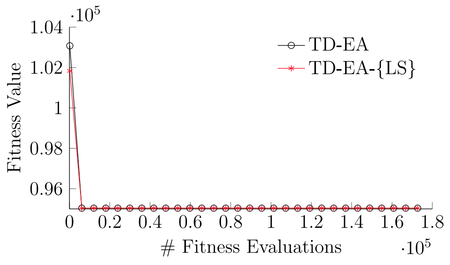

Fig. 7.11. Network 4×4 cells.

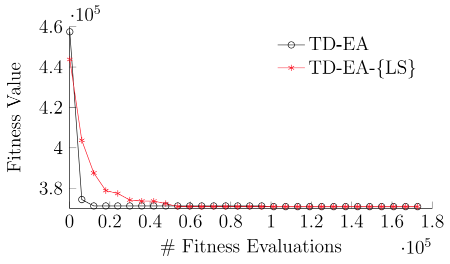

Fig. 7.12. Network 10×10 cells.

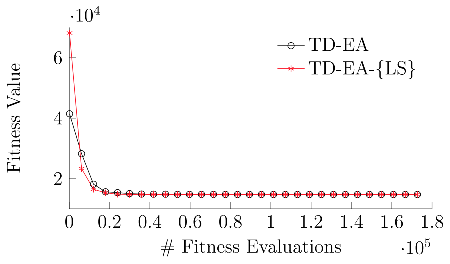

Fig. 7.13. Network 12×12 cells.

support, once again, the hypothesis that in 76% of the cases, the TD-EA's efficiency is coming from the "optimisation phase".

Another important thing to notice in Table 14 is that removing the local search has enhanced the efficiency of the TD-EA in the two largest instances of 30×30 cells. Indeed, for the 1st network of size 30×30, the TD-EA-{LS} could reach a new best result that has never been obtained by another solver in the literature (including the TD-EA). For the 2nd network of 30×30 cells, the TD-EA-{LS} reached a new value of the metrics "# Hits" and "Mean" that has never been obtained in the literature (including by the TD-EA).

8. Conclusions and future work

In this work, we have presented a new adaptive evolutionary algorithm based on takeover models for solving and addressing the users' mobility management issue in *pre*-5G telephony networks. Our proposed algorithm has been thoroughly assessed by tackling 25 realistic instances of different sizes and organised in three sets. Besides, the proposal has been compared with 28 state-of-the-art MMP's solvers. The results' significance has been verified by carrying out thorough statistical tests.

The attained results have demonstrated that our TD-EA algorithm is more efficient, scalable and robust than most of the state-of-the-art techniques and could also achieve very competitive results compared to those of one of the top-ranked optimisers: the 2SA-cGA. Furthermore, our proposed solver could manage to get new (i.e. never obtained before in the literature) best solutions in some instances, obtain results as good as those achieved by state-of-the-art algorithms in 5 instances and could outperform every solver in the literature in 2 instances including the largest network we have been using through our assessment. Considering the third collection of networks, the statistical results have shown similar efficiency between the TD-EA and two of the top-ranked optimisers: 2SA-cGA and the DE. Our proposed TD-EA could also beat the 2SA-cGA in the two most realistic and largest networks ever studied in the literature (900 cells).

As for future work, we think that the use of LTGA [69], LT-GOMEA [70], P3 [71], DSMGA-II [72] and the 3LOa [73] could be a promising direction to explore so as to enhance the MMP's state-of-the-art solvers. Furthermore, we intend to test our proposal on the real-world field and this by involving telephony operators. Also, we seek to tackle larger realistic instances and include communication traffic changes using SUMO (Simulation of Urban Mobility) and OpenStreetMap platform.

CRediT authorship contribution statement

Zakaria Abdelmoiz Dahi: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review and editing, Visualization, Supervision, Project administration. **Enrique Alba:** Conceptualization, Methodology, Validation, Resources, Writing – original draft, Writing – review and editing, Visualization, Supervision, Project administration. **Gabriel Luque:** Conceptualization, Methodology, Validation, Writing – original draft, Writing – review and editing, Visualization, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research is partially funded by the Universidad de Málaga, Consejería de Economía y Conocimiento de la Junta de Andalucía and FEDER under grant number UMA18-FEDERJA-003 (PRECOG); MCIN/AEI/10.13039/501100011033 under grant number PID 2020-116727RB-I00 (HUmove) and under TAILOR ICT-48 Network (No 952215) funded by EU Horizon 2020 research and innovation programme. Funding for open access charge is supported by the Universidad de Málaga/CBUA. The views expressed are purely those of the writer and may not in any circumstances be regarded as stating an official position of the European Commission. We also acknowledge that some instances studied in our work were previously inspired from the CRAWDAD dataset spitz/cellular. The authors would like also to address special thanks to Mrs Malika Belaifa and Mrs Zeineb Dahi for their help in creating the realistic problem benchmarks.

Appendix. List of acronyms

Table A.15 presents the set of all the acronyms used in the text. It is also worth noting that one can find, in the text, the same acronyms ending with an "s" that indicates the plural form.

Table A.15
List of acronyms and their explanations.

Acronym	Explanation
ACO	Ant Colony Optimisation algorithm
BABA	Binary Artificial Bat Algorithm
BRL	Best Result known in the Literature
GA-PSO	combined Particle Swarm Optimisation algorithm with the Genetic Algorithm
DE	Differential Evolution algorithm
DSMGA-II	Dependency Structure Matrix Genetic Algorithm II
EA	Evolutionary Algorithm
GA	Genetic Algorithm
GPSO	Geometric Particle Swarm Optimisation algorithm
GSO	Group Search Optimiser
HNN-BD	modified Hopfield Neural Network combined with Ball Dropping technique
HUX	Half Uniform Crossover
HTC	High Throughput Computing
IoT	Internet of Things
LU	Location Update
LTGA	Linkage Tree Genetic Algorithm
LT-GOMEA	Linkage Tree Gene-pool Optimal Mixing Evolutionary Algorithm
MM	Mobility Management
MMC	Mobility Management Core
MMP	Mobility Management Problem
NSGA-II	Non-dominated Sorting Genetic Algorithm II
NOMAD	Nonlinear Optimisation with the Mesh Adaptive Direct search algorithm
NRC	Non-Reporting Cell
OI-GA	Oscillatory-Increasing adaptive Genetic Algorithm
PSO	Particle Swarm Optimisation algorithm
P3	Parameter-less Population Pyramid
RC	Reporting Cell
RCP	Reporting Cell Problem
RCS	Reporting Cell Scheme
SS	Scatter Search algorithm
SA	Simulated Annealing
SUD	Standard Uniform Distribution
2SA-cGA	Stop-and-Start Adaptive cellular Genetic Algorithm
SPEA 2	Strength Pareto Evolutionary Algorithm 2
TS	Tabu Search
TD-EA	Takeover Time-Driven adaptive bi-phased Evolutionary Algorithm
TD-EA-{LS}	Takeover Time-Driven adaptive bi-phased Evolutionary Algorithm without Local Search
3LOa	Linkage Learning based on Local Optimisation algorithm
3LO	Linkage Learning based on Local Optimisation

References

[1] V. Berrocal-Plaza, M.A. Vega-Rodriguez, J.M. Sanchez-Perez, A multi-objective study of the gaussian cluster paging in the reporting cells strategy, Appl. Soft Comput. 28 (2015) 332–344.

[2] A. Achour, L. Deru, J. Christophe Deprez, Mobility management for wireless sensor networks a state-of-the-art, Procedia Comput. Sci. 52 (2015) 1101–1107.

[3] M.C. Chuang, Authentication and Mobility Management Mechanisms in VANETs, Scholars' Press, 2014.

[4] V. Berrocal-Plaza, M.A. Vega-Rodriguez, J.M. Sanchez-Perez, Studying the reporting cells strategy in a realistic mobile environment, in: Proceedings of the 6th World Congress on Nature and Biologically Inspired Computing, (NaBIC), IEEE, 2014, pp. 29–34.

[5] G. Pavai, T.V. Geetha, A survey on crossover operators, ACM Comput. Surv. 49 (4) (2016).

[6] Z.A. Dahi, C. Mezioud, E. Alba, A novel adaptive genetic algorithm for mobility management in cellular networks, in: Proceedings of the 11th International Conference on Hybrid Artificial Intelligent Systems, (HAIS), Springer, 2016, pp. 225–237.

[7] E. Alba, B. Dorronsoro, The exploration/exploitation tradeoff in dynamic cellular genetic algorithms, IEEE Trans. Evol. Comput. 9 (2) (2005) 126–142.

[8] Y. Bravo, G. Luque, E. Alba, Takeover time in dynamic optimization problems, in: Proceedings of the IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments, (CIDUE), IEEE, 2013, pp. 25–30.

[9] G. Luque, E. Alba, Math oracles: A new way of designing efficient self-adaptive algorithms, in: Proceedings of the 15th Annual Conference

Companion on Genetic and Evolutionary Computation, in: (GECCO '13 Companion), ACM, 2013, pp. 217–218.
[10] M.H. Alavidoost, M. Tarimoradi, M.H. Fazel Zarandi, Fuzzy adaptive genetic algorithm for multi-objective assembly line balancing problems, Appl. Soft Comput. 34 (2015) 655–677.
[11] P. Zhou, Y. Zhao, Z. Zhao, T. Chai, Source mapping and determining of soil contamination by heavy metals using statistical analysis, Artif. Neural Net. Adapt. Genet. Algorithm J. Environ. Chem. Eng. 3 (4, Part A) (2015) 2569–2579.
[12] Z.A. Dahi, Optimisation Problem Solving in the Field of Cellular Networks (Ph.D. thesis), Constantine 2 University, 2017.
[13] Z.A. Dahi, E. Alba, A. Draa, A stop-and-start adaptive cellular genetic algorithm for mobility management of gsm-lte cellular network users, Expert Syst. Appl. 106 (2018) 290–304.
[14] S.M. Razavi, Tracking Area Planning in Cellular Networks (Ph.D. thesis), Department of Science and Technology, Linkoping University, 2011.
[15] A. Bar-Noy, I. Kessler, Tracking mobile users in wireless communications networks, IEEE Trans. Inform. Theory 39 (6) (1993) 1877–1886.
[16] A. Hać, X. Zhou, Locating strategies for personal communication networks, a novel tracking strategy, IEEE J. Sel. Areas Commun. 15 (8) (1997) 1425–1436.
[17] V. Berrocal-Plaza, M.A. Vega-Rodríguez, J.M. Sánchez-Pérez, Optimizing the mobility management task in networks of four world capital cities, J. Netw. Comput. Appl. 51 (2015) 18–28.
[18] G. Luque, E. Alba, Analyzing the behaviour of population-based algorithms using rayleigh distribution, in: Proceedings of the 12th International Conference on Parallel Problem Solving from Nature, (PPSN XII), Springer, 2012, pp. 417–427.
[19] S.S. Kim, I.H. Kim, V. Mani, H.J. Kim, D.P. Agrawal, Partitioning of mobile network into location areas using ant colony optimization, ICIC Int. ICIC Express Lett. B 1 (1) (2010) 39–44.
[20] F. Mehta, P. Swadas, 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, Universal Publishers, Gujarat, 2009, pp. 208–213.
[21] R. Subrata, A.Y. Zomaya, Artificial life techniques for reporting cell planning in mobile computing, in: Proceedings of the Workshop on Biologically Inspired Solutions to Parallel Processing Problems, (BioSP3), 2002.
[22] R. Subrata, A.Y. Zomaya, Artificial life techniques for reporting cell planning in mobile computing, in: Proceedings of the 16th International Parallel and Distributed Processing Symposium, (IPDPS '02), IEEE, 2002, pp. 75–95.
[23] R. Subrata, A.Y. Zomaya, Location management algorithms based on biologically inspired techniques, in: Handbook of Algorithms for Mobile and Networking and Computing, Vol. 16, Taylor & Francis, 2006, pp. 363–390.
[24] S.R. Parija, P. Addanki, P.K. Sahu, S.S. Singh, 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), Vol. 327, Springer, 2015, pp. 823–830.
[25] Y. Bravo, G. Luque, E. Alba, Takeover time in evolutionary dynamic optimization: From theory to practice, Appl. Math. Comput. 250 (2015) 94–104.
[26] J. Taheri, A.Y. Zomaya, Bio-inspired algorithms for mobility management, in: Proceedings of the the International Symposium on Parallel Architectures, Algorithms, and Networks, in: ISPAN '08, IEEE Computer Society, USA, 2008, pp. 216–223.
[27] S. Swayamsiddha, Prateek, S.S. Singh, S. Parija, D.K. Pratihar, Reporting cell planning-based cellular mobility management using a binary artificial bat algorithm, Heliyon 5 (3) (2019).
[28] M.H. Vekariya, P.B. Swadas, Hybrid genetic algorithm-simulated annealing approach for reporting cell planning problem of location management, Int. J. Innov. Res. Comput. Commun. Eng. 3 (6) (2015).
[29] V. Berrocal-Plaza, M.A. Vega-Rodríguez, J.M. Sánchez-Pérez, 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), Vol. 8480, Springer, 2014, pp. 285–295.
[30] V. Berrocal-Plaza, M.A. Vega-Rodriguez, J.M. Sanchez-Perez, 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), Vol. 299, Springer, 2014, pp. 1–10.
[31] V. Berrocal-Plaza, M.A. Vega-Rodriguez, J.M. Sanchez-Perez, 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), Vol. 8584, Springer, 2014, pp. 212–223.
[32] D. Wang, C. Xiong, W. Huang, Group search optimizer for the mobile location management problem, Sci. World J. 2014 (12) (2014).
[33] S.S. Kim, G. Kim, J.H. Byeon, J. Taheri, Particle swarm optimization for location area mobility management, Int. J. Innov. Comput. Inf. Control 8 (12) (2012) 8387–8398.
[34] D.L. González-Álvarez, A. Rubio-Largo, M.A. Vega-Rodríguez, S.M. Almeida-Luz, J.A. Gómez-Pulido, J.M. Sánchez-Pérez, Solving the reporting cells problem by using a parallel team of evolutionary algorithms, Log. J. IGPL 20 (4) (2012) 722–731.
[35] A. Rubio-Largo, D.L. Gonzalez-Alvarez, M.A. Vega-Rodriguez, S.M. Almeida-Luz, J.A. Gomez-Pulido, J.M. Sanchez-Perez, 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), Vol. 73, Springer, 2010, pp. 71–78.
[36] C.A. Baburaj, K. Alagarsamy, 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), Vol. 132, Springer, 2011, pp. 321–328.
[37] C.A. Baburaj, M.C.A.M Phil, K. Alagarsamy, A review on various network results for reporting cell planning in genetic algorithm technique, Int. J. Eng. Sci. Technol. 2 (2010) 4088–4094.
[38] M. Patra, S.K. Udgata, 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), Vol. 7077, Springer, 2011, pp. 248–256.
[39] S.M. Almeida-Luz, M.A. Vega-Rodriguez, J.A. Gomez-Pulido, J.M. Sanchez-Perez, 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), Vol. 73, Springer, 2010, pp. 145–152.
[40] S.M. Almeida-Luz, M.A. Vega-Rodriguez, J.A. Gomez-Pulido, J.M. Sanchez-Perez, 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), Vol. 6086, Springer, 2010, pp. 534–543.
[41] L. Wang, G. Si, 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), IEEE, 2010, pp. 1160–1163.
[42] F. Mehta, P. Swadas, A simulated annealing approach to reporting cell planning problem of mobile location management, Int. J. Recent Trends Eng. Technol. 2 (2) (2009) 355–367.
[43] E. Alba, J. García-Nieto, J. Taheri, A. Zomaya, 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, Vol. 4974, Springer, 2008, pp. 1–10.
[44] J. Taheri, A.Y. Zomaya, A modified hopfield network for mobility management, Wirel. Commun. Mob. Comput. 8 (3) (2008) 355–367.
[45] S. Almeida-Luz, M.A. Vega-Rodriguez, J.A. Gomez-Pulido, J.M. Sanchez-Perez, Applying differential evolution to the reporting cells problem, in: Proceedings of the International Multiconference on Computer Science and Information Technology, (IMCSIT), IEEE, 2008, pp. 65–71.
[46] S.M. Almeida-Luz, M.A. Vega-Rodriguez, J.A. Gomez-Pulido, J.M. Sanchez-Perez, Differential evolution for solving the mobile location management, Appl. Soft Comput. 11 (1) (2011) 410–427.
[47] B. Patel, P. Bhatt, A differential evolution algorithm for reporting cell problem in mobile computing, Int. J. Eng. Res. Technol. (IJERT) 3 (3) (2014).
[48] W. Wang, F. Wang, Q. Pan, F. Zuo, Improved differential evolution algorithm for location management in mobile computing, in: Proceedings of the International Workshop on Intelligent Systems and Applications, (ISA), IEEE, 2009, pp. 1–5.
[49] R. Subrata, A.Y. Zomaya, Evolving cellular automata for location management in mobile computing networks, IEEE Trans. Parallel Distrib. Syst. 14 (1) (2003) 13–26.
[50] R. Subrata, A.Y. Zomaya, A comparison of three artificial life techniques for reporting cell planning in mobile computing, IEEE Trans. Parallel Distrib. Syst. 14 (2) (2003) 142–153.
[51] D.E. Goldberg, Sizing populations for serial and parallel genetic algorithms, in: Proceedings of the 3rd International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc., 1989, pp. 70–79.
[52] D.E. Goldberg, K. Deb, J.H. Clark, Genetic algorithms, noise, and the sizing of populations, Complex Syst. 6 (1991) 333–362.
[53] J. Sarma, K.A. De Jong, An analysis of the effects of neighborhood size and shape on local selection algorithms, in: Proceedings of the 4th International Conference on Evolutionary Computation for Parallel Problem Solving from Nature, Springer, 1996, pp. 236–244.
[54] J. Sarma, K.A. De Jong, An analysis of local selection algorithms in a spatially structured evolutionary algorithm, in: Proceedings of the 7th International Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco (CA), 1997, pp. 181–186.
[55] U. Kamath, C. Domeniconi, K.A. De Jong, An analysis of a spatial EA parallel boosting algorithm, in: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, in: (GECCO '13), ACM, 2013, pp. 1053–1060.

[56] J. Sprave, A unified model of non-panmictic population structures in evolutionary algorithms, in: Proceedings of the International Congress on Evolutionary Computation, (CEC '99), Vol. 2, IEEE, 1999, p. 1391.
[57] J. Sprave, A Unified Model of Non-Panmictic Population Structures in Evolutionary Algorithms, Tech. rep., Univeristy of Dortmund Reihe Computational Intelligence Collaborative Research Center 531, Design and Management of Complex Technical Processes and Systems by means of Computational Intelligence Methods, 2010.
[58] G. Rudolph, On takeover times in spatially structured populations: Array and ring, in: Proceedings of the 2^{nd} Asia-Pacific on Genetic Algorithms and Application, Global-link Publishing Company, 2000, pp. 144–151.
[59] M. Giacobini, E. Alba, A. Tettamanzi, M. Tomassini, Modeling selection intensity for toroidal cellular evolutionary algorithms, in: Proceedings of the Conference on Genetic and Evolutionary Computation, (GECCO), Springer, 2004, pp. 1138–1149.
[60] M. Giacobini, M. Tomassini, A. Tettamanzi, Takeover time curves in random and small-world structured populations, in: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, in: (GECCO '05), ACM, 2005, pp. 1333–1340.
[61] R. Nogueras, C. Cotta, Analyzing meme propagation in multimemetic algorithms: Initial investigations, in: Proceedings of the Federated Conference on Computer Science and Information Systems, (FedCSIS), IEEE, 2013, pp. 1013–1019.
[62] D. Simoncini, S. Verel, P. Collard, M. Clergue, Centric selection: A way to tune the exploration/exploitation trade-off, in: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, in: (GECCO '09), ACM, 2009, pp. 891–898.
[63] E. Alba, G. Luque, Growth curves and takeover time in distributed evolutionary algorithms, in: Proceedings of the International Conference on Genetic and Evolutionary Computation, (GECCO), Springer, 2004, pp. 864–876.
[64] G. Luque, E. Alba, Parallel Genetic Algorithms: Theory and Real World Applications, Springer, 2011, pp. 55–71, ch. Theoretical Models of Selection Pressure for Distributed GAs.
[65] G. Luque, E. Alba, Selection pressure and takeover time of distributed evolutionary algorithms, in: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, in: (GECCO '10), ACM, 2010, pp. 1083–1088.
[66] A. Ceroni, M. Pelikan, M. Pelikan, D.E. Goldberg, Convergence-Time Models for the Simple Genetic Algorithm with Finite Population, Tech. rep., University of illinois at Urbana-Champaign, Departement of General Engineering, Illinois Genetic Algorithms Laboratory, 2001.
[67] J.H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, 1975.
[68] D. Whitley, A. Sutton, Genetic algorithms - a survey of models and methods, 2012.
[69] D. Thierens, The linkage tree genetic algorithm, in: R. Schaefer, C. Cotta, J. Kołodziej, G. Rudolph (Eds.), Parallel Problem Solving from Nature, PPSN XI, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 264–273.
[70] D. Thierens, P.A. Bosman, Optimal mixing evolutionary algorithms, in: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, in: GECCO '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 617–624.
[71] B.W. Goldman, W.F. Punch, Parameter-less population pyramid, in: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, in: GECCO '14, Association for Computing Machinery, New York, NY, USA, 2014, pp. 785–792.
[72] S.-H. Hsu, T.-L. Yu, Optimization by pairwise linkage detection, incremental linkage set, and restricted / back mixing: DSMGA-II, in: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, in: GECCO '15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 519–526.
[73] M.W. Przewozniczek, M.M. Komarnicki, Empirical linkage learning, IEEE Trans. Evol. Comput. 24 (6) (2020) 1097–1111.
[74] B.W. Goldman, W.F. Punch, Fast and efficient black box optimization using the parameter-less population pyramid, Evol. Comput. 23 (3) (2015) 451–479.
[75] M.M.J. Kabir, S. Xu, B.H. Kang, Z. Zhao, Discovery of interesting association rules using genetic algorithm with adaptive mutation, in: Proceedings Part II of 22^{nd} International Conference on Neural Information Processing, (ICONIP), Springer, 2015, pp. 96–105.
[76] D. Thierens, Adaptive mutation rate control schemes in genetic algorithms, in: Proceedings of the 2002 Congress on Evolutionary Computation, (CEC '02), Vol. 1, 2002, pp. 980–985.
[77] G. Luque, E. Alba, Designing an efficient self-adaptive parallel algorithm using math oracles, in: L. Barolli, I. Woungang, O.K. Hussain (Eds.), Advances in Intelligent Networking and Collaborative Systems, Springer International Publishing, 2018, pp. 313–325.
[78] L.J. Eshelman, The CHC Adaptive Search Algorithm: How to have safe search when engaging in nontraditional genetic recombination, Found. Genet. Algorithms (1991) 265–283.
[79] M. S., CRAWDAD dataset spitz/cellular (v. 2011-05-04), Downloaded from https://crawdad.org/spitz/cellular/20110504, http://dx.doi.org/10.15783/C71P4C, may, 2011.