

A Novel Adaptive Genetic Algorithm for Mobility Management in Cellular Networks

Zakaria Abd El Moiz Dahi¹(✉), Chaker Mezioud¹, and Enrique Alba²

¹ Department of Fundamental Computer Science and Its Application,
Constantine 2 University, Constantine, Algeria
{zakaria.dahi,chaker.mezioud}@univ-constantine2.dz

² Dep. de Lenguajes y Ciencias de la Computación, Málaga University,
Málaga, Spain
eat@lcc.uma.es

Abstract. Metaheuristics are promising tools to use when addressing optimisation problems. On the other hand, most of them are hand-tuned through a long and exhaustive process. In fact, this task requires advanced knowledge about the algorithm used and the problem treated. This constraint restricts their use only to pure abstract scientific research and by expert users. In such a context, their further application by non-experts in real-life fields will be impossible. A promising solution to this issue is the inclusion of adaptation within the search process of these algorithms. On the basis of this idea, this paper demonstrates that simple adaptation strategies can lead to more flexible algorithms for real-world fields, also more efficient when compared to the hand-tuned ones and finally more usable by non-expert users. Seven variants of the Genetic Algorithm (GA) based on different adaptation strategies are proposed. As benchmark problem, an NP-complete real-world optimisation problem in advanced cellular networks, the mobility management task. It is used to assess the efficiency of the proposed variants. The latter were compared against the state-of-the-art algorithm: the Differential Evolution algorithm (DE), and showed promising results.

Keywords: Evolutionary computation · Adaptation · Cellular networks

1 Introduction

Due to the limitations of the exhaustive-based search methods, metaheuristics appear as a promising tool to use when tackling optimisation problems. The Genetic Algorithm (GA) is one of the first-proposed Evolutionary Algorithms (EAs) [10], and also one of the most studied ones.

The main motivation behind the use of metaheuristics is ultimately solving real-world problems, which often requires a problem-specific algorithm design. The complex customisation of algorithms to a specific problem either needs a specialist or limits the field of application to scientific research. Indeed, most

of the scientific works propose powerful algorithms without further consideration for their industrial application. Furthermore, there is a growing demand for optimisation software usable by non-specialists within an industrial development environment. A promising way to do this is to supply the industry with intelligent algorithms containing a mechanism that modifies the parameters without external control [1]. Many schemes of adaptation exist in the literature, but all can be regrouped in three categories [1]: deterministic, adaptive and self-adaptive algorithms. In this paper we are interested in the first class.

Customer tracking and mobility management are key factors in communication networks in general and especially in cellular networks (2G, EDGE, GPRS, 3G, 3G+, LTE, 4G). In fact, any dysfunction of this task may alter the service quality of the network itself causing call drops and other issues. Regarding the actual high standards of the mobile phone industry, these kinds of failure are unacceptable for the telephony operators and the users as well. Thus, the mobility management is one of the most challenging optimisation issues in cellular networks. It has been formulated as a binary optimisation problem and was proven to be NP-complete [8].

This paper *investigates/proves* whether simple strategies of adaptation can lead to more flexible, efficient and usable algorithms. Seven deterministically-adaptive variants of the genetic algorithm are proposed based on several adaptation strategies with different complexities and behaviour. The performances of the proposed variants is assessed by solving the mobility management problem. The state-of-the-art algorithm Differential Evolution (DE) is taken as an efficient competitor.

The remainder of the paper is structured as follows. In Sect. 2, basic concepts related to the GA, adaptation within EAs and finally the mobility management problem are presented. In Sect. 3, the proposed variants of the genetic algorithm are introduced. Section 4 is dedicated to the experimental results and their discussion. Finally, we conclude the paper in Sect. 5.

2 Basic Concepts

In this section, we introduce basic concepts of the genetic algorithm, adaptation within evolutionary algorithms and finally the mobility management task.

2.1 The Genetic Algorithm

The genetic algorithm was first proposed by Holland [10]. It was born from the attempt to mimic the natural process of reproduction and evolution. Thus, the idea of the GA is to evolve a set of individuals toward a fitter state by applying a series of selection, variation, evaluation and replacement phases.

The GA starts by *initializing* a population of N individuals \vec{X} of size L , where each individual $\vec{X}_{i=1\dots N} = \{X_{(i,1)}, X_{(i,2)}, \dots, X_{(i,L)}\}$. The latter are evaluated using the fitness function of the problem being addressed. Then, the best individual \vec{X}_* is extracted.

After, at each iteration of the algorithm, firstly we *select* M individual parents \vec{P} according to a given strategy. Secondly, the selected parents $\vec{P}_{i=1..M}$ are perturbed using *variation* operators such as the mutation and crossover in order to produce $(N - M)$ new offspring \vec{O} . The crossover consists of exchanging parts of the individual parents at some chosen points. The amount of information exchanged and the number of crossover operations performed are ruled by both the type of crossover used and the crossover probability $P_c \in [0,1]$. After, the mutation operator randomly mutates the values of genes in each produced offspring $\vec{O}_{i=1..(N-M)}$. The number of mutation operations is determined by a mutation probability $P_m \in [0,1]$. Thirdly, the produced offspring are *evaluated* using the objective function of the problem being tackled.

Finally, having the old population of parents and the newly-produced offspring, a replacement phase is performed in order to determine what will be the composition of the population for the next iteration. However, the GA mechanics are left as a black box due to the existence of several types of selection, crossover, mutation and replacement.

2.2 Adaptation Within Evolutionary Algorithms

Adaptation within EAs reflects the attempt to mimic processes of natural evolution. State-of-the-art EA implementations often require a deep algorithmic knowledge from the user in order to choose appropriate parameters values for solving a specific optimisation problem. Even for an expert, the parameters configuration for an optimal performance is hard to find. The idea of automatic adaptation is to evolve these parameters values according to a given scheme. A classification of these schemes can be made according on how they are performed. Many classifications exist, but we opt for the taxonomy proposed in [1].

Parameter Tuning: Is the scheme used whenever the parameters have constant values throughout the run of the EA (i.e. there is no adaptation). Consequently, an external agent or a manual mechanism is needed to tune the desired parameters and to choose the most appropriate values.

Dynamic: It appears whenever there is some mechanism that modifies a parameter without any external control. The class of EA that uses this scheme can be further subdivided into other specific subschemes.

- **Deterministic:** This scheme takes place if the value of a parameter is tuned constantly by some deterministic rule. This rule modifies the strategy parameter deterministically without using any feedback from the EA. This scheme is the one we are interested in our work.
- **Adaptive:** This scheme takes place if there is some form of *feedback* got from the EA that is used to set the direction *or/and* magnitude of the change to the strategy parameters.
- **Self-adaptive:** In this scheme the parameters are encoded with the variables and can evolve as well as the solution itself.

2.3 Studied Adaptation Strategies

Many parameters could be adapted such as P_c , the selection pressure, the population size, ... etc. But in our work the one that will undergo the adaptation is the mutation probability P_m . This is very simple but it could be very powerful at the same time. Adaptation in general is about three issues: When to adapt (i.e. period)? In which direction (i.e. increase or decrease)? How much (i.e. the amplitude of change)? Our scientific methodology consists of using strategies that deal with these issues in different manners and with an increasing level of complexity and with different directions of evolution in order to represent a wide range of mathematical adaptation behaviour. We firstly opt for the use of a canonical *monotone adaptation: linearly-decreasing* and *linearly-increasing*. Then, we use a more complex adaptation strategy (i.e. non-monotone) called *oscillatory*. In this perspective, we opt for the *sinusoidal wave* with a *decreasing* and *increasing amplitude* [6]. Finally, we use several specialized state-of-the-art strategies whose the efficiency is well-established. Thus, we opt for the ones studied in [4, 7, 9].

This is done in order to answer some questions such as: does adaptation allow producing better algorithms than the hand-tuned ones? Does designing more efficient algorithms imply using more complex adaptation schemes? Ultimately, is the adaptation a real solution for conceiving better algorithms?

It is worth to mention that within the next sections the variable N corresponds to the size of the population, L refers to the length of the individual, $Maxit$ refers to the maximum number of iterations and finally t represents the actual iteration.

The Linear Adaptation Strategy: This strategy is based on adding (i.e. when increasing) or subtracting (i.e. when decreasing) at each iteration “ t ” of the algorithm a constant “*step*” from the previous P_m value at iteration $(t - 1)$. The step size used is computed using Formula (1).

$$Step = \frac{UB - LB}{Maxit} \tag{1}$$

$$P_m(t + 1) = P_m(t) \pm Step \tag{2}$$

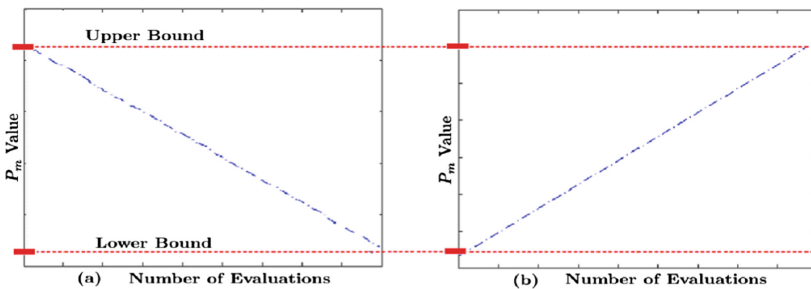


Fig. 1. Linear adaptation strategies: (a) decreasing and (b) increasing

Variables UB and LB are the upper and lower bounds of the interval where the P_m value is supposed to evolve. Figure 1(a) and (b) show the resulting evolution of P_m value when using the linearly-decreasing and linearly-increasing adaptation strategies, respectively.

The Oscillatory Adaptation Strategy: This strategy is based on the formula defining a *sinusoidal wave* with a *decreasing* or an *increasing amplitude*. Formula (3) defines the sinusoidal wave with an increasing amplitude, whereas Formula (4) highlights the new terms that differentiate the sinusoidal wave with an increasing from the one with a decreasing amplitude.

$$P_m(t) = P_{m_0} * \left(\sin(2\pi * freq * t) * \frac{t}{Maxit} + 1 \right) \tag{3}$$

$$\frac{Maxit - t}{Maxit} \tag{4}$$

The variable “ $freq$ ” is a parameter that controls the frequency of the wave and P_{m_0} is the value from where the evolution starts. Figure 2(a) and (b) show the resulting evolution of the P_m value when using the oscillatory-decreasing and oscillatory-increasing adaptation strategies, respectively.

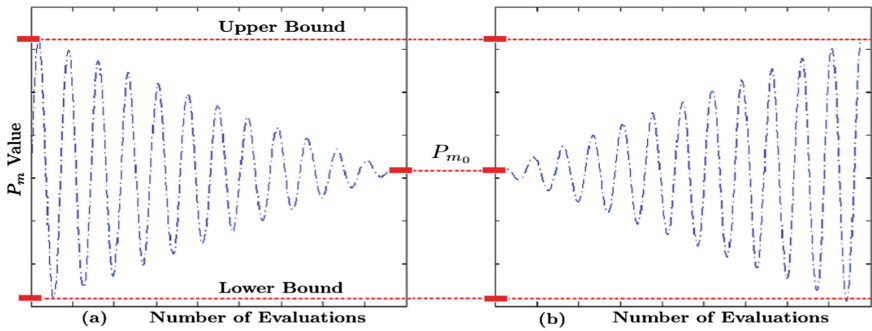


Fig. 2. Oscillatory adaptation strategies: decreasing and increasing amplitude

State-of-the-art Adaptation Strategies. These strategies were proposed in [4,7,9]. In the following, each one of them is introduced.

- **Bäck Strategy:** It was proposed by Bäck et al. [4]. It is defined by Formula (5).

$$P_m(t) = \left(2 + \frac{L - 2}{Maxit - 1} * t \right)^{-1} \tag{5}$$

- **Forgaty Strategy:** It was proposed by Fogarty [7]. It is defined by Formula (6).

$$P_m(t) = \frac{1}{240} + 0.113752^t \tag{6}$$

- **Hesser Strategy:** It was proposed by Hesser and Männer in [9]. It is defined by Formula (7), where C_1 , C_2 and C_3 are constants.

$$P_m(t) = \sqrt{\frac{C_1 \exp(-C_3 t/2)}{C_2 N \sqrt{L}}} \quad (7)$$

2.4 Mobility Management in Cellular Networks

In order to assess the efficiency of our proposed approaches, we could go for an academic problem as much of the literature does. But instead, we wanted to improve the interest of our study by considering a real-problem such as mobility management.

Tracking mobile users is a key factor and a sensitive issue in cellular phone networks (2G, EDGE, GPRS, 3G, 3G⁺, LTE, 4G) [11] and other wired or wireless networks (vehicular, sensor, etc.). The optimisation of this task implies the optimisation of two sub-tasks which are: the paging and the location-update. Two main schemes of mobility management exist; the paging scheme and the location area-update-based scheme. The latter includes two main types; dynamic and static. The reporting cell scheme is one of the most practical static-location-update-based schemes [11]. In this last one, one seek to find the optimal network configuration that determines the reporting and non-reporting cells.

The reporting cell scheme was first proposed by Hac and Zhou in [8]. They assessed that it is NP-complete. Then, it has been modelled in [12] as a binary optimisation problem, where the objective function to optimize is defined by Formula (8).

$$\text{Minimize: } L_c = \beta * \sum_{i \in RC} N_{Lu}(i) + \sum_{i=0}^{Tc} N_P(i) * V(i) \quad (8)$$

Here $N_{Lu}(i)$ is the location-update cost associated to the i^{th} cell and RC is the set of reporting cells in the network. $N_p(i)$ is the paging cost associated to the i^{th} cell and Tc is the number of cells in the network. $V(i)$ is the vicinity factor associated to the i^{th} cell. The vicinity factor depends on the type of the cell itself. If it is a reporting cell, the vicinity factor can be calculated as the number of non-reporting cells from where one can reach the reporting cell. If the cell is a non-reporting cell, the vicinity factor can be calculated as the maximum vicinity factor of the reporting cells from where one can reach the non-reporting cell. The parameter β is a ratio to balance the objective function. In fact, the cost of location-update is considered to be 10 times much higher than the cost of paging. Thus, usually in the literature β is set to 10. Figure 3 illustrates a potential solution of the reporting cell problem for a network with eight cells, where the 1st, 4th and the 8th cells are reporting cells, while the 2nd, 3rd, 5th, 6th and the 7th are non-reporting cells.

1	0	0	1	0	0	0	1
1st	2nd	3rd	4th	5th	6th	7th	8th

Fig. 3. Solution instance for the mobility management problem

3 The Proposed Approach

In this paper, seven deterministically-adaptive variants of the genetic algorithm are proposed. The proposed variants are enhanced generational elitist genetic algorithms. In this section, a detailed description of the main operators and phases of the proposed approaches is given. Please note that the term L used in the next sections refers to the length of the chromosome. In the case of the mobility management problem, L corresponds to the size of the network.

3.1 Initialisation

At this step, a first population of N chromosomes is generated randomly by assigning to each gene of each chromosome either the value 1 or 0. A random number “ $rand$ ” is generated from a standard uniform distribution. If $rand \geq 0.5$ then the gene will have the value 1, otherwise it will have the value 0.

3.2 Selection

In this step, $(N/2)$ couples are formed from the N chromosomes existing in the population. A binary tournament selection is used to select the parents and create the couples that will constitute the mating pool that will undergo the reproduction phase. To create a couple, the best of two randomly-chosen parents is mated with a second individual selected from two other randomly-chosen parents. The process is repeated $(N/2)$ times.

3.3 Reproduction

In this step, the $(N/2)$ couples of parents will undergo a series of genetic operators (i.e. crossover and mutation) to produce N new offspring.

Crossover: A two-point crossover is performed. This mechanism is ruled by a probability of crossover $P_c \in [0,1]$. In our case P_c is set to 1. For each couple of parents, a random number “ $rand$ ” is generated from a standard uniform distribution. If $rand \leq P_c$, the crossover is performed on the current couple, otherwise the parents are copied and considered as new offspring.

When applying the two-point crossover, two splitting-points are chosen randomly from the interval $[1, L]$. Then, the parents are swapped at these points. The zones delimited by the head, the first switch point, the second switch point and the tail of the two parents are exchanged.

Mutation: After the crossover operator, mutation is applied on the produced offspring. This operator is ruled by a mutation probability $P_m \in [0,1]$. But in our case it is set to $(1/L)$. The mutation consists in a bit-flip. For each gene of each one of the produced offspring, a random number “*rand*” is generated from a standard uniform distribution. If $rand \leq P_m$, the mutation is performed on the current gene. The gene is then flipped to the other state, otherwise it is kept as it is. In this work the value of P_m is driven using different adaptation strategies explained in Sect. 2.3.

3.4 Evaluation and Replacement

In this step, the produced offspring are evaluated using Formula (8). A generational elitist strategy (μ, λ) is used to extract the N best individuals (i.e. from the union of both produced offspring λ at generation (t) and the parents μ at generation $(t-1)$) that will constitute the new population of individuals at generation $(t+1)$. The latter will undergo the next generation of genetical operators. Detailed pseudo-code of the proposed approaches is available in the hyperlink¹.

On the basis of the strategy of adaptation used for the mutation probability P_m , we propose in this work seven new adaptive variants of the genetic algorithm. The first two proposed variants are based on a linear adaptation. The LD-GA (for Linearly-Decreasing adaptation strategy based Genetic Algorithm) that uses a linearly-decreasing strategy. The second variant called LI-GA (for Linearly-Increasing adaptation strategy based Genetic Algorithm) that is based on a linear increasing strategy.

The two next variants are based on the oscillatory adaptation. The OD-GA (for Oscillatory-Decreasing adaptation strategy based Genetic Algorithm) which is based on a sinusoidal strategy with a decreasing amplitude. The OI-GA (for Oscillatory-Increasing adaptation strategy based Genetic Algorithm) that is based on a sinusoidal strategy with an increasing amplitude.

The three last adaptation strategies are the state-of-the-art ones. Firstly, the B-GA (for Bäck adaptation strategy based Genetic Algorithm) which is based on the strategy proposed by Bäck et al. [4]. Secondly, the F-GA (for Forgy adaptation strategy based Genetic Algorithm) which is based on the strategy proposed by Forgy in [7]. Finally, the H-GA (for Hesser adaptation strategy based Genetic Algorithm) which is based on the strategy proposed by Hesser et al. [9].

4 Experimental Results and Analysis

All our tests are carried out using an Intel I3 core with 2 GB Ram and a Linux OS. The implementation is done using Matlab 7.12.0 (R2011a). Experiments are run using a cluster of 16 physical machines. Parallel processing of fitness evaluations is performed. The latter are equally split over 4 cores.

¹ Appendix: <http://tinyurl.com/Appendix-Hyperlink>.

The scalability of the proposed variants is assessed using different-sized instances of the reporting cell problem. Twelve instances of 4×4 , 6×6 , 8×8 and 10×10 cells are used. The instances were generated on the basis of realistic patterns. The instances are provided by the University of Malaga, Spain. They have been previously used in [2,3,5] and are available in².

On the basis of state-of-the-art works that already studied the GA behaviour and parameters [1], for the LI-GA, LD-GA, OI-GA and OD-GA, the Lower Bound (*LB*) is set to $(1/L)$, where L is the chromosome length (in our case the size of the network). The Upper Bound (*UB*) is set to 0.3. For the OI-GA and OD-GA, the frequency *freq* is set to 0.7 and P_{m_0} is set to the middle of the interval $[(1/L), 0.3]$ after a parameter tuning experiment. Finally, for the variant H-GA, the constant C_1 , C_2 and C_3 are set to 0.4, 0.6 and 0.9, respectively.

The proposed variants are compared to one of the state-of-the-art metaheuristic to solve the mobility management task in cellular networks, which is the Differential Evolution algorithm (DE) [2,3].

For consistency and fair comparison with the DE, we use the same experimental parameters used in [2,3]. Thus, experiments are performed until reaching the termination criterion of 175000 fitness evaluations (including the evaluation of the initial population). We use 175 individuals, 1000 iterations and all the algorithms are repeated for 30 runs. Several results are reported such as: the *best*, the *worst* fitness, the *mean* and *standard deviation* of the fitnesses over 30 executions. We also report *#hits*; which represents how many times the proposed variants were able to reach the results obtained by the differential evolution algorithm. The variable *#evals*; represents the mean of the number of fitness evaluations needed to the proposed variants to obtain the same best results as the ones obtained by the differential evolution algorithm. The variable *Time* is the average time needed to the proposed approach to reach the same results reached by the differential evolution and it is expressed in seconds.

It is worth to mention that metrics *#hits*, *#evals* and *Time* are recorded every time the proposed variants achieve the same/better results obtained by the DE. The comparison is based on the best solution reported through 30 executions. Statistical analysis tests are also performed such as the one-sample Kolmogorov-Smirnov test, the Brown-Forsythe test, the Kruskal-Wallis one-way analysis of variance test and a multiple comparison test is used to determine which is the best “self* plan”. All tests are performed using a level significance equal to 5%. Detailed-numerical results of the above-mentioned tests are available in the hyperlink³.

4.1 Numerical Results

Table 1 shows the numerical results obtained when using the proposed variants of the genetic algorithm to tackle networks of size 4×4 , 6×6 , 8×8 and 10×10 cells, respectively. It is worth to mention that the symbol “-” is used whenever the proposed variants achieve results worst than the ones obtained by the DE.

² Benchmark Instances: <http://oplink.lcc.uma.es/problems/mmp.html>.

³ Appendix: <http://tinyurl.com/Appendix-Hyperlink>.

4.2 Discussion and Interpretation

On the basis of the metric “*Mean*” of Table 1, one can note that all the variants except the B-GA achieve the same results for networks 1, 2 and 3 of size 4×4 cells. Considering now networks of size 6×6 cells, the OI-GA variant is the best one, whereas for network 2, the OI-GA, OD-GA and F-GA are the best ones. For network 3, the OI-GA and OD-GA are the best variants. For networks of size 8×8 cells, the F-GA is the best one, while for networks 2 and 3, the OI-GA is the best one. Finally, for networks of size 10×10 cells, the OI-GA is assessed to be the best variant for networks 1, 2 and 3. Also, on the basis of the metric “*Mean*” of Table 1, the B-GA is assessed to be the worst variant in all networks, whereas the OI-GA is assessed to be the best one for all networks.

When considering the metric “*#evals*” of the best variants found in each one of the problem instances, one can observe that F-GA is the best variant when tackling network 1 of size 4×4 cells, whereas for network 2 it is the LD-GA the best one and finally for network 3, the LI-GA is assessed to be the best one. Let us consider now networks of size 6×6 cells, the H-GA is assessed to be the best variant for networks 1, 2 and 3. When tackling networks of size 8×8 cells, the OD-GA is found to be the best variant for networks 2 and 3. Finally, when tackling instance 3 of size 10×10 cells, the F-GA is found to be the best variant.

Taking again the same metric, the worst variant assessed for networks 1 and 3 of size 4×4 cells is the OD-GA, while for network 2 it is H-GA. When tackling networks of size 6×6 cells, OI-GA is assessed to be the worst variant. Finally, the LD-GA is found to be the worst variant for network 2 of size 8×8 cells, whereas for network 3 it is OI-GA the worst one.

Considering now the metric “*#hits*” of Table 1, all the variants except the B-GA achieve the same results for networks of size 4×4 cells. Let us take now networks of size 6×6 cells, the OI-GA is assessed to be the best variant for network 1, while for network 2 it is OI-GA, OD-GA and F-GA the best ones. For network 3, both OI-GA and OD-GA are the best variants. When tackling networks of size 8×8 cells, the OI-GA is found to be the best variant for networks 2 and 3. Finally, for network 3 of size 10×10 cells, the F-GA is found to be the best variant. Also, on the basis of the metric “*#hits*” of Table 1, the B-GA is found to be the worst variant in all problem instances, whereas the OI-GA is assessed to be the best one.

With regards to results of Table 1, the proposed variants are able to achieve results as good as those obtained by the differential evolution algorithm in 8 out of 12 instances. On the other hand, the proposed variants were outperformed by the DE in 3 out of 12 networks: network 1 of size 8×8 cells and networks 1 and 2 of size 10×10 cells. Finally, the proposed variant F-GA was able to outperform the differential evolution algorithm in network 3 of size 10×10 cells. This fact is even more interesting since we are proposing adaptive variants of a simple algorithm that is able to outperform a more complex and specialized one.

Statistical results of the Kruskal-Wallis and also the Post-Hoc tests⁴ support the numerical results obtained in Table 1. The null-hypothesis is rejected for each

⁴ Appendix: <http://tinyurl.com/Appendix-Hyperlink>.

one of the instances which confirms that a difference exists in the distribution of the proposed variants. Also, statistical ranking of the Post-Hoc confirm the promising efficiency of the proposed variants.

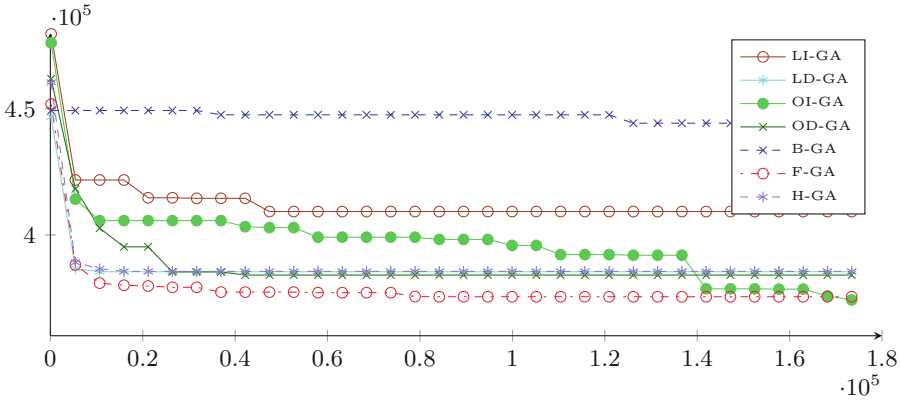


Fig. 4. Fitness evolution for network 3 of size 10×10 cells

Figure 4 shows the fitness evolution when using the deterministically-adaptive variants of the genetic algorithm for tackling network 3 of size 10×10 cells. One can see that the F-GA has the quickest convergence rate and B-GA has the slowest one, while the remaining variants OI-GA, OD-GA, LI-GA, LD-GA and H-GA have a moderate convergence rate when comparing with the F-GA and B-GA. However, no clear link can be seen between the type of adaptation strategy (i.e. linear and oscillatory) or the direction of evolution (i.e. increasing and decreasing) and the produced convergence rate.

5 Conclusions

In this paper, a deterministically-adaptive genetic algorithm based on seven adaptation strategies was presented. The proposed variants were assessed for solving the mobility management problem in advanced cellular networks. The state-of-the-art algorithm differential evolution was taken as a comparison basis.

The experiments showed that the proposed approach based on the sinusoidal adaptation is the best one. They also showed that the use of adaptation within EAs can lead to more efficient algorithms when comparing to the hand-tuned ones. They demonstrated also that designing more efficient adaptive algorithms does not imply using more complex adaptation schemes. As perspective, we intend to study more complex and unexplored strategies of adaptation in order to solve efficiently the mobility management in advanced cellular networks.

Acknowledgments. Prof. Alba acknowledges partial support from Spanish-plus-FEDER project TIN2014-57341-R

References

1. Alba, E., Dorronsoro, B.: The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. *IEEE Trans. Evol. Comput.* **9**(2), 126–142 (2005)
2. Almeida-Luz, S., Vega-Rodríguez, M., Gomez-Pulido, J., Sanchez-Perez, J.: Applying differential evolution to the reporting cells problem. In: Proceedings of the International Multiconference on Computer Science and Information Technology (IMCSIT), pp. 65–71. IEEE (2008)
3. Almeida-Luz, S., Vega-Rodríguez, M., Gómez-Púlido, J., Sánchez-Pérez, J.: Differential evolution for solving the mobile location management. *Appl. Soft Comput.* **11**(1), 410–427 (2011)
4. Bäck, T., Schütz, M.: Intelligent mutation rate control in canonical genetic algorithms. In: Michalewicz, M., Raś, Z.W. (eds.) *ISMIS 1996*. LNCS, vol. 1079, pp. 158–167. Springer, Heidelberg (1996)
5. Berrocal-Plaza, V., Vega-Rodríguez, M.A., Sánchez-Pérez, J.M.: A strength pareto approach to solve the reporting cells planning problem. In: Murgante, B., et al. (eds.) *ICCSA 2014, Part VI*. LNCS, vol. 8584, pp. 212–223. Springer, Heidelberg (2014)
6. Dahi, Z., Chaker, M., Draa, A.: Deterministically-adaptive genetic algorithm to solve binary communication problems: application on the error correcting code problem. In: Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication (IPAC 2015), pp. 79:1–79:5 (2015)
7. Fogarty, T.: Varying the probability of mutation in the genetic algorithm. In: Proceedings of the 3rd International Conference on Genetic Algorithms, pp. 104–109. Morgan Kaufmann Publishers Inc. (1989)
8. Hac, A., Zhou, X.: Locating strategies for personal communication networks: a novel tracking strategy. *IEEE J. Sel. Areas Commun.* **15**(8), 1425–1436 (1997)
9. Hesser, J., Manner, R.: Towards an optimal mutation probability for genetic algorithms. In: Schwefel, H.-P., Männer, R. (eds.) *PPSN 1990*. LNCS, vol. 496, pp. 23–32. Springer, Heidelberg (1991)
10. Holland, J.: Genetic algorithms and adaptation. *Adapt. Control of Ill-Defined Syst.* **16**, 317–333 (1984)
11. Razavi, S.: Tracking area planning in cellular networks. Ph.D. thesis, Department of Science and Technology Linköping University Norrköping Sweden (2011)
12. Subrata, R., Zomaya, A.: 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)