Applying Scatter Search to the Location Areas Problem

Sónia M. Almeida-Luz[1], Miguel A.Vega-Rodríguez[2], Juan A. Gómez-Pulido[2], and Juan M. Sánchez-Pérez[2]

[1] Polytechnic Institute of Leiria, School of Technology and Management, 2400 Leiria, Portugal
sluz@estg.ipleiria.pt
[2] University of Extremadura, Dept. Technologies of Computers and Communications, Escuela Politécnica, Campus Universitario s/n, 10071 Cáceres, Spain
{mavega,jangomez,sanperez}@unex.es

Abstract. The Location Areas scheme is one of the most common strategies to solve the location management problem, which corresponds to the management of the mobile network configuration with the objective of minimizing the involved costs. This paper presents a new approach that uses a Scatter Search based algorithm applied to the Location Areas scheme as a cost optimization problem. With this work we pretend to analyze and set the main parameters of scatter search, using four distinct test networks and compare our results with those achieved by other authors. This is a new approach to this problem and the results obtained are very encouraging because they show that the proposed technique outperforms the existing methods in the literature.

Keywords: Scatter Search, Location Areas problem, Location Management, Mobile Networks.

1 Introduction

The current mobile networks, more specifically the personal communication networks (PCN) [1], must support communications that enable all the users to make or receive calls at any time of the day and for, or from, any location. In order to these networks support the mobility of users and be able to find them, also when they change their location, it is necessary to consider mobility management and more precisely location management (LM) when the network infrastructures are defined.

The location management is divided into two main operations: location update that corresponds to the notification of current location, performed by mobile terminals when they change their location in the mobile network, and location inquiry that represents the operation of determining the location of the mobile user terminal, performed by the network when it tries to direct an incoming call to the user.

There exist several strategies of location management and they are divided into two main groups: static and dynamic schemes [2]. Static schemes are more used in the actual mobile networks. Furthermore, as static techniques, the most common ones are always-update, never-update, and location area (this one presented in section 2) schemes [2], among others.

In this paper, a Scatter Search (SS) based algorithm is used to find the best configuration for the location area scheme in a mobile network. Therefore, we present a new approach to this problem. Section 2 provides an overview of the Location Area (LA) problem and the involved costs. In section 3, the SS algorithm is described, as well as its parameters. In section 4, the experimental results are presented and an analysis over the obtained results is done with the intent of defining the best Scatter Search parameters. Finally, section 5 includes conclusions and future work.

2 Location Areas Problem

In cellular network systems it is very important to keep track of the location of the users, even when they move around without making or receiving calls, so as to consequently, be able to route calls to the users regardless of their location.

Location Areas (LA) scheme corresponds to an important strategy of location management, that is used with the objective of reducing signaling traffic caused by paging messages and location updates in cellular network systems.

In the LA scheme, the network is partitioned into groups of cells and each group corresponds to a region, or more precisely to a LA, as we can see in Fig. 1a, where we have a network with four LAs and each with several cells. In this scheme, when a mobile terminal moves to a new LA, its location is updated, which means a location update is performed. When the user receives an incoming call, the network must page all the cells of the new LA of the user, looking for its mobile terminal.

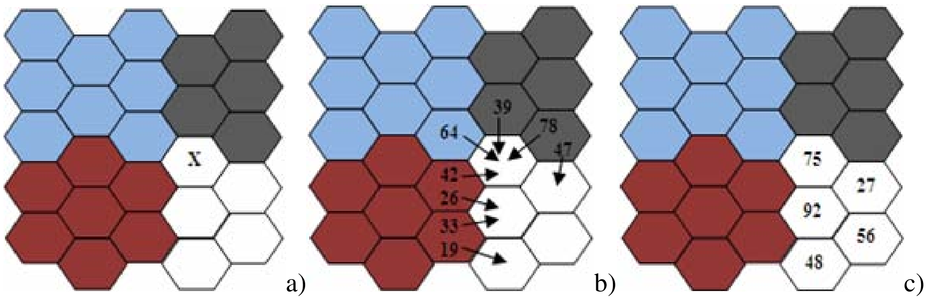

Fig. 1. a) Network partitioning in LAs; b) Entering flow of users; c) Incoming calls to one LA

2.1 Location Management Cost

The LA problem can be defined as the problem of finding an optimal configuration of location areas, minimizing the location management cost. The location management cost normally is divided into: location update cost and location paging cost [3], [4].

Location Update Cost. The location update (LU) cost corresponds to the cost involved with the location updates performed by mobile terminals in the network, when they change their location to another LA. Because of that, the number of location updates is normally caused by the user movements in the network. This means that, when we calculate the update cost for a certain LA, we must consider the entire network and look for the flow of users.

If we consider the network of Fig. 1b, it is possible to see the total number of users who enter in the white LA. To calculate the location update cost for that LA, we must sum up those numbers of users that enter (from another LA) on each cell of the LA:

$$N_{LU}= 47+78+39+64+42+26+33+19=348. \qquad (1)$$

Location Paging Cost. The location paging (P) cost is caused by the network when it tries to locate a user's mobile terminal, during the location inquiry, and normally the number of paging transactions is directly related to the number of incoming calls. The task of calculating the paging cost is simpler, because we only need to count the number of incoming calls in the selected LA and then multiply the value by the number of cells in the respective LA. Considering the incoming calls to the white LA shown in Fig. 1c, the calculus of paging cost is:

$$N_P= (75+27+92+56+48)\ x5=1490. \qquad (2)$$

Total Cost. The location management cost involves other parameters and components, but those are considered to be equal for all strategies [4]. Therefore, these other parameters do not influence the comparison of different strategies, and we will not consider them for the total cost. In conclusion, the combination of location update cost and location paging cost is sufficient to compare different strategy results.

The formula to calculate the total cost of location management [5] is:

$$\text{Cost}= \beta \text{ x } N_{LU} + N_P. \qquad (3)$$

The total cost of location updates is given by N_{LU}, the total cost of paging transactions is given by N_P, and finally β is a ratio constant used in a location update relatively to a paging transaction in the network. The cost of each location update is considered to be much higher than the cost of each paging transaction, due to the complex process that must be executed for each location update performed, and also because most of the time a mobile user moves without making any call [4]. Due to all of that, the cost of a location update is normally considered to be 10 times greater than the cost of paging, that is, β =10 [3].

For the white LA referred earlier, and presented in Fig. 1b and 1c, the total cost by (3) would be:

$$\text{Cost}= 10 \text{ x } 348 + 1490 = 4970. \qquad (4)$$

To calculate the total cost of the network with the configuration defined, which means with four LAs, would be necessary to make the calculus for each LA and then sum all the values and get the final total cost.

3 Scatter Search Algorithm

Scatter search (SS) is an evolutionary algorithm introduced by Glover in 1977 [6]. SS is characterized by five main components [7], [8]: *Diversification Generation method*, *Improvement method*, *Reference Set Update method*, *Subset Generation method* and *Solution Combination method.* The pseudo-code of the SS algorithm is presented in Fig. 2 (see [7], [8] for more details).

```
SS Algorithm
1:  Start with Population $P$=Ø. Use Diversification Generation method to create a
    solution $x$ and improve it with the Improvement method. If $x \notin P$ add $x$ to $P$.
    Repeat this step until get $PSize$ different solutions.
2:  Use the Reference Set Update method to create $RefSet = \{x^1,\ldots,x^b\}$ with $b/2$ best
    solutions and $b/2$ most diverse solutions (from the $b/2$ best solutions), of $P$.
3:  Evaluate the $RefSet$ solutions and order the solutions, using their fitness function.
4:  Make NewSolution=TRUE.
5:  while (NewSolution) do
6:     Make NewSolution=FALSE
7:     Use the Subset Generation method and create all different subsets
8:     while(Exist subsets not examined) do
9:           Select a subset and label it as examined
10:          Apply the Solution Combination method to the solutions of the subset
11:          Apply the Improvement method to each new solution obtained
             Considering $x$ as the improved solution:
12:          if ($f(x) < f(x^b)$) and ($x \notin RefSet$) then
13:                   Set $x^b = x$ and order solutions of $RefSet$
14:                   Make NewSolution = TRUE
15:          end if
16:    end while
17: end while
```

Fig. 2. Pseudo-code for Scatter Search algorithm

4 Experimental Results

In this section, we detail the source and preparation of the test networks, subsequently we explain the most relevant decisions and choices made in our algorithm implementation, then we expose our different experiments, present our results and finally we compare with other authors' results.

4.1 Test Networks Generation

In order to compare results we will use the same test networks of Taheri and Zomaya in [4] and in our previous work [9]. Each of these networks has a set of data for each cell including the cell identification, the number of total updates, that each cell may have, the number of calls received in each cell and also the number of updates to be considered by each cell whose neighbors change their LAs. In this work we use four distinct networks with respective sizes of 5x5, 5x7, 7x7 and 7x9 cells from [4], [9] with the objective of testing the performance of our SS approach applied to networks with distinct sizes.

4.2 Parameters Definition

Considering the details of SS algorithm implementation (Fig. 2), we decided to apply a local search in the boundary cells of each LA as the *improvement* method. We also defined in the *subset generation* method the definition of subsets of size 2. Relatively to the *combination* method we developed a crossover that could be applied to a maximum of four crossover points according to a predetermined probability. In conclusion, our SS uses four core parameters: initial population size *PSize*; reference set size *RSSize*; probability of combination (crossover) *Cr*; and the number of iterations of local search *nLS*. Furthermore, the *RSSize* is divided into two parameters, the size of the quality solutions *nQrs* and the size of the diversity solutions *nDrs*.

To start the experiments we set the following values: *PSize*=100; *RSSize*=10; *nQrs*=5; *nDrs*=5; *Cr*=0.2; *nLS*=1 based on what several authors suggest [7], [8].

4.3 Individuals Validation

When an individual is generated we must consider that an invalid configuration network may be created. This is because with the application of the algorithm it is possible that we have scattered LAs. This means that we may have cells, which are not connected, attributed to the same LA in distinct places of the network, but in reality that is not possible and we must correct or discard the individual.

To solve this problem we created a set of methods to validate and make feasible each potential solution. The first method is to split these scattered LAs into small ones. Then, the second method is defined to merge LAs, with the purpose of not having only one cell belonging to a LA, when all their neighbor cells belong to different LAs. Finally, after this, we have a third method to renumber the LAs because during all the process some LA numbers may have been deleted.

This process must be repeated for all the individuals that are generated, to assure that the final solution will be a valid one.

4.4 Simulation Results and Analysis

In our approach the fitness function, that is used to evaluate each solution, is defined according to the equation (3) presented in section 2.1. This fitness function corresponds to the calculus of the total cost of location management.

With the objective of studying in detail the best configuration of SS, we have executed five distinct experiments. In order to assure the statistical relevance of the results we have performed 30 independent runs, for each parameters' combination in each experiment. Due to space reasons, we only present the most important conclusions of all these experiments.

Experiment 1 – Defining the PSize. The first experiment had the objective of defining the best size of the population (parameter *PSize*). With the four test networks and using the initial values set to each parameter, we tested *PSize* with the values 10, 25, 50, 75, 100, 125, 150, 175 and 200. Analyzing the results, we observed that, considering the four test networks, the best configurations for this parameter were 75 and 100. Finally, we decided to proceed for the second experiment with *PSize*=100.

Experiment 2 – Defining the RSSize. The second experiment was defined to elect the optimal size of the *RefSet* that obtains the best result for all the networks. Using the *PSize*=100, determined in the first experiment, and maintaining the other initial parameters' values, we checked the following configurations for *RSSize*: 2, 4, 6, 8, 10, 12, 14, 16, 18 and 20. Evaluating the results, including the best and the average fitness, we noticed that from the *RSSize*=6 it generally obtained good fitness values for all networks, but it was with the *RSSize*=18 that the best average results were obtained.

Experiment 3 – Defining the nQrs and the nDrs. In the third experiment we intended to determine the division of the *RefSet* between quality *nQrs* and diversity *nDrs* solutions. So, in order to execute this experiment, we assigned *PSize*=100, *RSSize*=18 and maintained the initial values of *Cr* and *nLS*. Using these values we tested all the possible combinations of *nQrs* and *nDrs* (knowing that their sum must be 18). The obtained results were very similar for all the combinations. Because of that, we decided to maintain an equal division of 9 solutions for each subset (the standard configuration of SS).

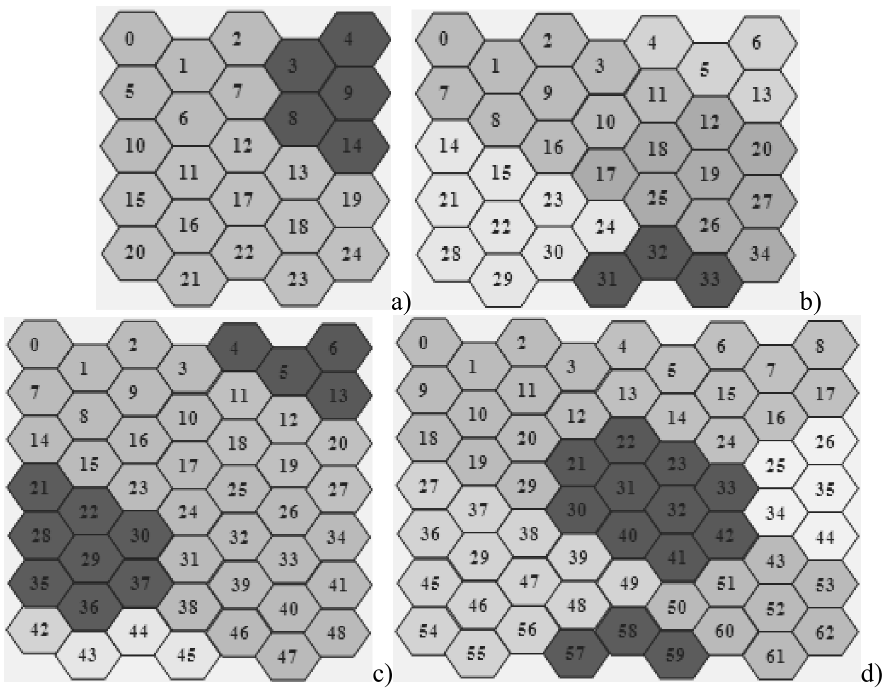

Fig. 3. The best LAs configuration: **a)** 5x5 **b)** 5x7 **c)** 7x7 **d)** 7x9 Networks

Experiment 4 – Defining the Cr. With the fourth experiment we pretended to elect the probability of combination (crossover) that obtains the best results for all the test networks. To proceed with this experiment we fixed the values obtained in the earlier experiments (*PSize*=100, *RSSize*=18, *nQrs*=9, *nDrs*=9), using the initial *nLS*=1, and testing the following values for *Cr*: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9. Analyzing the average results, we could conclude that for the small networks the lowest values of *Cr* performed better but, for the bigger networks the results were more similar and that was not so obvious. Therefore, we decided to proceed with *Cr*=0.2, the one that had already given good results.

Experiment 5 – Defining the nLS. The objective of this last experiment was to define the best value for the number of local search iterations *nLS*. We fixed the other parameters with the values obtained previously, and checked the following configurations for *nLS*: 1, 2, 3, 4, 5 and 6. Observing the results we might see clearly that it was with *nLS*=5 that the best results (lowest costs) were reached, because with *nLS*=6 the average of fitness values started to increase.

After these experiments we have reached the best configuration for the SS parameters applied to the LA problem: *PSize*=100, *RSSize*=18, *nQrs*=9, *nDrs*=9, *Cr*=0.2, and *nLS*=5. With these parameters we can achieve the best solutions (lowest fitness values) for all the four networks. The best LAs configuration that we have obtained for each network is shown in Fig. 3.

4.5 Comparing Our Results with Other Applied Algorithms

If we compare the results reached by this approach based on SS with the ones that we achieved in a previous work using a Differential Evolution (DE) based approach [9], we may say that SS performs better, because it always obtains equal or better solutions (see Table 1).

Furthermore, comparing the SS results with the ones presented by Taheri and Zomaya in [10], that present results obtained with other algorithms as GA (Genetic Algorithm), HNN (Hopfield Neural Network), SA (Simulated Annealing) and GA-HNNx (different combinations of Genetic Algorithm and Hopfield Neural Network, see [10]), our results are always equal or even better, as it is also possible to observe in Table 1. In particular, for the bigger networks (7x7 and 7x9), that is, the more complex ones, SS obtains the best results.

Table 1. Comparison of network costs achieved by different algorithms

Test Network	Algorithm							
	SS	**DE**	**GA**	**HNN**	**SA**	**GA-HNN1**	**GA-HNN2**	**GA-HNN3**
5x5	26990	26990	28299	27249	26990	26990	26990	26990
5x7	39832	39859	40085	39832	42750	40117	39832	39832
7x7	60685	61037	61938	63516	60694	62916	62253	60696
7x9	89085	89973	90318	92493	90506	92659	91916	91819

5 Conclusions and Future Work

In this paper we present a new approach based on SS algorithm with the objective of finding the best configuration for the LAs strategy in mobile networks.

We have studied in detail the best configuration of SS, applied to the LAs problem. The best parameters, after a big number of experiments with four distinct networks, are *PSize*=100, *RSSize*=18 with *nQrs*=9 and *nDrs*=9, *Cr*=0.2 and *nLS*=5.

Comparing the performance of SS algorithm with other artificial life techniques as differential evolution (DE), genetic algorithm (GA), hopfield neural network (HNN), simulated annealing (SA), and different combinations of GA and HNN (GA-HNNx) we may say that it performs well because outperforms the results obtained by those.

As future work we have the intention of using real data (like SUMATRA [11]) as input for generating the test networks and we have also planned the application of other evolutionary strategies to the LA problem, to compare results with those obtained by SS algorithm.

Acknowledgments. This work was partially funded by the Spanish Ministry of Science and Innovation and FEDER under the contract TIN2008-06491-C04-04 (the M* project). Thanks also to the Polytechnic Institute of Leiria, for the economic support offered to Sónia M. Almeida-Luz to make this research.

References

1. Pahlavan, K., Levesque, A.H.: Wireless Information Networks. John Wiley & Sons, Chichester (1995)
2. Wong, V.W.S., Leung, V.C.M.: Location Management for Next-Generation Personal Communications Networks. IEEE Network 14(5), 18–24 (2000)
3. Gondim, P.R.L.: Genetic Algorithms and the Location Area Partitioning Problem in Cellular Networks. In: 46th IEEE Vehicular Technology Conf. Mobile Technology for the Human Race, vol. 3, pp. 1835–1838 (1996)
4. Taheri, J., Zomaya, A.Y.: A Genetic Algorithm for Finding Optimal Location Area Configurations for Mobility Management. In: 30th Anniversary of the IEEE Conference on Local Computer Networks (LCN), pp. 568–577 (2005)
5. Subrata, R., Zomaya, A.Y.: Evolving Cellular Automata for Location Management in Mobile Computing Networks. IEEE Trans. Parallel & Distrib. Syst. 14(1), 13–26 (2003)
6. Glover, F.: Heuristics for integer programming using surrogate constraints. Decision Sciences 8, 156–166 (1977)
7. Martí, R., Laguna, M., Glover, F.: Principles of Scatter Search. European Journal of Operational Research 169, 359–372 (2006)
8. Laguna, M., Hossell, K.P., Martí, R.: Scatter Search: Methodology and Implementation in C. Kluwer Academic Publishers, Norwell (2002)
9. Almeida-Luz, S.M., Vega-Rodríguez, M.A., Gómez-Pulido, J.A., Sánchez-Pérez, J.M.: Solving the Location Area Problem by Using Differential Evolution. Journal of Communications Software and Systems 4(2), 131–141 (2008)
10. Taheri, J., Zomaya, A.Y.: A Combined Genetic-Neural Algorithm for Mobility Management. J. Mathematical Modelling and Algorithms 6(3), 481–507 (2007)
11. Stanford University Mobile Activity TRAces (SUMATRA) (May 2009), `http://infolab.stanford.edu/sumatra/`