Distributed Coevolutionary Genetic Algorithm for Optimal Design of Ad Hoc Injection Networks

Grégoire Danoy, Pascal Bouvry
Faculty of Sciences, Technology and Communications
University of Luxembourg
Luxembourg
{gregoire.danoy, pascal.bouvry}@uni.lu

Enrique Alba
Department of Computer Science
University of Málaga
Málaga, Spain
eat@lcc.uma.es

Abstract— **Multi-hop ad-hoc networks allow establishing local groups of communicating devices in a self-organizing way. However, in a global setting such networks fail to work properly due to network partitioning. This means that users locally interacting could eventually spread and move away from each other and consequently loose their connections. Considering that devices are capable of communicating both locally (e.g. using Wi-Fi or Bluetooth) and additionally with remote devices (e.g. using GSM/UMTS links) the objective of our work is to optimize the way of inter-linking multiple network partitions. To this end we rely on small-world network properties, that consist in using special attributes like the clustering coefficient and the characteristic path length. In this paper we investigate the use of a distributed Cooperative Coevolutionary Genetic Algorithm (CCGA) and compare its performance to a generational and a steady state genetic algorithm (genGa and ssGA) for optimizing one instance of this topology control problem and present initial evidence of its capacity to solve it.**

I. Introduction

Multi-hop ad-hoc networks are networks composed of communicating devices capable of spontaneously interconnecting without any pre-existing infrastructure. The most popular wireless networking technologies available nowadays for building such networks are Bluetooth and IEEE802.11 (Wi-Fi). Devices in range to one another communicate in a point-to-point fashion. But such ad-hoc networks are intrinsically dynamic. Due to their limited transmission range, such networks face partitioning problems that penalizes their global efficiency. In real scenarios, one or more additional remote links have to be created to keep connected the different clusters of locally interacting users that dynamically move.

In this paper we consider the problem of optimizing the addition of such long-range links (e.g. GSM, UMTS or HSDPA) that are also called *bypass links* to interlink network partitions. To tackle this topology control problem, we use small-world properties as indicators for the good set of rules to maximize inter-link efficiency. Small-world networks [1] feature a high clustering coefficient (γ) while still retaining a small characteristic path length (L). A small path length corresponds to fewer hops, which is of importance for effective routing mechanisms as well as for the overall communication performance of the entire network. The clustering coefficient represents the connectivity in the neighborhood of each node and thus reflects the degree of information dissemination each single node can achieve. This finally motivates the objective of evoking small-world properties in such settings.

In order to optimize those parameters (maximizing γ, minimizing L) and to minimize the number of required bypass links in the network, we relied on Evolutionary Algorithms (EAs) [2] and more specifically on a distributed Cooperative Coevolutionary Genetic Algorithms (GA) [3] using Dafo, our distributed agent framework for evolutionary optimization. CCGA has already proved its ability for solving complex real-world problems [4]. We start by investigating the kind of evolution step more amenable to our problem by comparing the performance of a distributed CCGA, to both generational [5] and steady-state [6] GAs on a basic instance of a partitioned ad-hoc network.

The remainder of this paper is organized as follows. In the next section we give a detailed view on CCGA. Section III introduces Dafo, our distributed agent framework for evolutionary optimization. Then in Section IV, we provide details on the injection network problem; we address as well several small-world properties. In Section V presents the experiments and analyzes the results. The last section contains our conclusions and perspectives.

II. Coevolutionary algorithms

As for the "classical" genetic algorithm, the concept of coevolutionary algorithms comes from the biological observations [7]. Indeed, the nature is composed of several species that coevolve. And instead of evolving a population of similar individuals (like in classical Genetic Algorithms) representing a global solution, we consider the coevolution of subpopulations of individuals representing specific parts of the global solution. In the following subsections, we introduce CCGA, a Cooperative Coevolutionary Genetic Algorithms [3]. This algorithm was already applied (cf. [8]) for parallel and distributed optimization of a number of test functions known in the area of evolutionary computation. It was demonstrated that coevolutionary algorithms outperform a sequential GA. In the present article, we consider optimizing the Injection Networks problem by applying this cooperative coevolutionary algorithm.

Proceedings 21st European Conference on Modelling and Simulation
Ivan Zelinka, Zuzana Oplatková, Alessandra Orsoni ©ECMS 2007
ISBN 978-0-9553018-2-7 / ISBN 978-0-9553018-3-4 (CD)

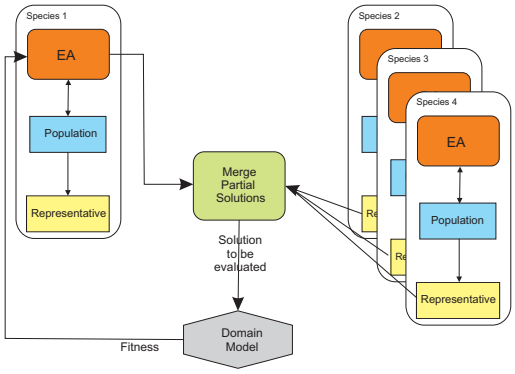


Fig. 1. Potter and De Jong's CCGA architecture

Algorithm 1: CCGA

```
gen = 0
foreach species_s do
    Pop_s(gen) = randomly initialized population
    evaluate fitness of each individual in Pop_s(gen)
end
while termination condition = false do
    gen = gen + 1
    foreach species_s do
        select Pop_s(gen) from Pop_s(gen − 1) based on fitness
        apply genetic operators to Pop_s(gen)
        evaluate fitness of each individual in Pop_s(gen)
    end
end
```

A. CCGA

Cooperative (also called symbiotic) coevolutionary genetic algorithms (CCGA) involve a number of independently evolving species which together form complex structures, well-suited to solve a problem. The fitness of an individual depends on its ability to collaborate with individuals from other species. In this way, the evolutionary pressure stemming from the difficulty of the problem favors the development of cooperative strategies and individuals. Potter and DeJong [3] developed a model in which a number of populations explore different decompositions of the problem. In Potter's system, each species represents a subcomponent of a potential solution. Complete solutions are obtained by assembling representative members of each of the species (populations). The fitness of each individual depends on the quality of (some of) the complete solutions it participated in, thus measuring how well it cooperates to solve the problem. The evolution of each species is controlled by a separate, independent evolutionary algorithm. In the initial generation *(t=0)* individuals from a given subpopulation are matched with randomly chosen individuals from all other subpopulations. A fitness for each individual is evaluated, and the best individual in each subpopulation is found. The process of *cooperative coevolution* starts form the next generation *(t=1)*. For this purpose, in each generation a cycle of operations is repeated in a round-robin fashion. Only one current subpopulation is active in a cycle, while the other subpopulations are frozen. All individuals from the active subpopulation are matched with the best values of frozen subpopulations. When the evolutionary process is completed a composition of the best individuals from each subpopulation represents a solution of a problem. Figure 1 shows the general architecture of Potter's cooperative coevolutionary framework, and the way each evolutionary algorithm computes the fitness of its individuals by combining them with selected representatives from the other species. Potter's methods have also been used or extended by other researchers, for instance Eriksson and Olsson [4] have used a cooperative coevolutionary algorithm for inventory control parameter optimization.

III. Distributed Agent-based evolutionary computation

We consider the opportunity to embed our players into software agents what is a convenient and elegant way to benefit from existing software infrastructure. Indeed using a multi-agent framework like Madkit [9] leverages us of writing low-level agent interaction behaviors and highly simplifies the agents distribution.

Since its introduction in the 70's, the agent technology has become synonymous with advanced computer software. The nature of real world problems has lead to the evolution of multi-agent systems in Distributed Artificial Intelligence (DAI). In this case each aspect of a problem is under the control of an agent and all the agents in the system interact to generate a global solution. In the proposed approach, the agent environment is composed of other evolving agents. In this article we choose Evolutionary Algorithms for modelling agents intelligence and the concept of agents organization for modelling agents interactions.

The concept of a "computational agent" becomes increasingly important in computer science, representing a new level of abstraction for software design. Distributed artificial intelligence/multi-agent systems are typically applied in two ways. In the first, the problem domain is itself distributed, e.g. telecommunications routing, and as such the multi-agent paradigm is a natural "fit" since each aspect of the system can be attributed to an agent. In the second, complex tasks are divided into multi-aspect problems to allow for the construction of a solution through the combination of a number of simpler (in respect to the global task) interacting agents. New issues arise when evolutionary computation is applied to the multi-agent paradigm. In these systems evolutionary algorithms must adapt to dynamic problem spaces, where changes are caused by the interactions of the agents in the environment of the global system. Agents evolve in a dynamic environment composed of other agents.
The use of evolutionary computing techniques in systems containing many interacting agents/entities goes back to the earliest days of experiments in machine intelligence. For example, Barricelli [10] used an abstract ecological model to examine the evolution of complexes

of cooperative entities, based in the idea of "symbiogenesis", the evolution of complexity by the bringing together of previously autonomous entities. However, nowadays most of coevolutionary computing consists of systems in which agents roles are predetermined as being either competitive or cooperative or a mixture of the two, i.e. agents are assigned particular tasks within the global system. Distributed problem solving by a multi-agent system represents a promising approach for solving complex computational problems. An agent-oriented problem-solving environment increases efficiency, capability and genericity by employing a set of agents communicating and co-operating to achieve their goals, i.e. that is to find local solutions that satisfy both their hard and soft constraints.

Our solution consists in providing a meta-level in the form of a distributed agent framework dedicated to evolutionary optimization including coevolutionary genetic algorithms. Modelling the algorithm with a multi-agent system makes explicit resolution strategy (i.e. the algorithm interaction graph) by using organizational models explicitly representing the roles and the interactions that are allowed for each agent. Using the agent technology also allows us to take benefit from existing multi-agent platforms and methodologies. The deployment and the distribution of the algorithm is thus ensured.

A. Framework Architecture

By providing a minimum of Java code concerning his optimization problem and a simple XML configuration file, the designer is capable of optimizing its function using various GAs that are generational GA, steady state GA and CCGA (that can be distributed). The XML file is used as an input file by the *Organizer Agent* for specifying information about the genetic algorithm, its parameters and information concerning the distribution if required.

Figure 2 represents a simple example of a distributed instance of Dafo on 3 different computers. Computers 2 and 3 run a *Slave Scheduler* Agent that first sends message *1* (which contains the IP address of Computer 1) to the *Communicator* Agent running in the Madkit platform's kernel in order to connect to Computer 1 (represented by message *2*). As soon as all *Communicators* have established a connection with the *Communicator* of Computer 1, one agent can communicate with any other agent, no matter on which computer it is (i.e. the computers are fully connected). Consequently, after this connection, all the other agents can communicate in a fully transparent way. Once all nodes are connected, the *Master Scheduler* sends message *3* that contains parameters (topology, population size, crossover operator, etc.) that will be used to instantiate the *Evolutionary Agents* (running a Simple GA), as represented by message *4*. The *Evolutionary Agents* can freely communicate with each other according to the CCGA architecture (as presented in Figure 1 and will send their partial solutions (message *4*) to the *Observer Agent* that is in charge of merging those

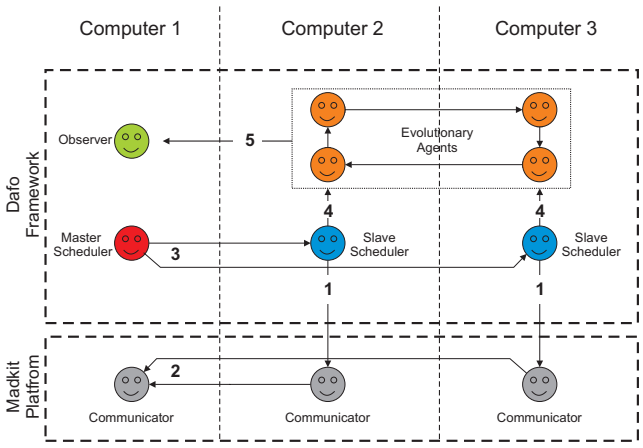


Fig. 2. Distributed Dafo Architecture

results and provide as output the global solution found.

Using Dafo makes juggling with the different versions of GAs easy. For instance, to use CCGA in distributed mode, it is sufficient to add the IP address and the port of the Node 1 and the total number of nodes in the configuration file.

IV. The Problem

This section introduces the injection network optimization problem using small-world properties. We first provide the reader with a definition of the injection network concept. Next we give details on what defines a small-world graph in this context.

A. Injection Networks

Due to several difficult (and practical) challenges that are inherent to mobile multi-hop ad-hoc networks, some past work advises the utilization of hybrid wireless networks, where a fixed infrastructure supports a higher connectivity among several clusters of ad hoc networks and avoids network partitioning [11] [12] [13]. However such a hybrid wireless network is often not feasible, because of economical and implementation issues. Alternatively, an infrastructureless setting is of interest where problems of restricted geographical regions are avoided. Helmy [14] focuses on long-range links for which the objective is to reduce the number of queries during the search for a given target node. Another approach introduces base stations to increase connectivity in ad hoc networks [15], thus realizing global reachability. Watts [1] introduces a spatially defined link, called *global edge*, with length-scaling properties to include spatial models in his investigations. Some approaches extend standard ad-hoc network models, by considering two different transmission ranges [16] [17], e.g. small distance Bluetooth links along with higher distance Wi-Fi links. Our initial motivation for the current investigation is based on the assumption that technologies like Bluetooth and Wi-Fi can be used to create ad-hoc communication links within the transmission range at no charge. Additional cellular network links such as GSM/UMTS/HSDPA might be employed by appropriately equipped devices to establish supple-

mentary communication links between two arbitrary devices. These links, however, will induce additional costs. Furthermore, we propose to implement that in a transparent way for the end user, i.e. linking in a mobile multi-hop ad-hoc network should be managed without explicit human interaction (self-organization). In summary, different approaches exist to augment ad-hoc networks with additional links. Different reasons exist for this need on increasing connectivity: e.g. to gain bandwidth between particular devices, and also to inter-link multiple ad-hoc network partitions. Consequently, we introduce the notion of *bypass links*.

Definition 1 (Watts) The spatial neighborhood $\Gamma_{tr}(v)$ of a node v is the set of nodes within transmission range tr of v.

Definition 2: A bypass link is a link *(u,v)* between nodes u and v with $u \notin \Gamma_{tr}(v)$.

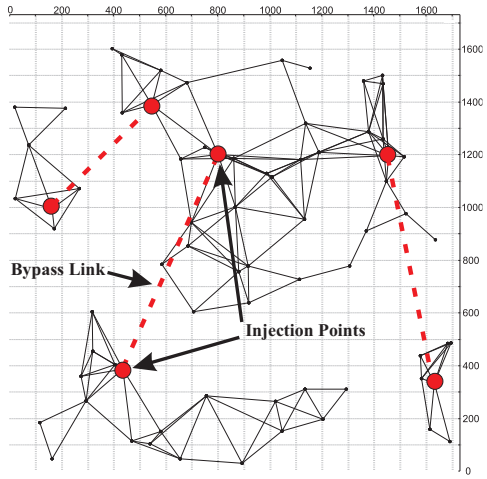


Fig. 3. Example of Injection Network

That is, a bypass link is a link which connects two nodes that are not in the same spatial neighborhood. Please note that elements of $\Gamma_{tr}(v)$ do not necessarily have to be connected to v in real settings. Practically, a bypass link can be built by using a cellular network as well as by using access points. Nevertheless, in our model a bypass link is counted as a single hop, thus simplifying the real topology behind that bypass link. Since we can dynamically control such bypass links, the network topology basically can be biased as needed, resulting in a topology control problem. Thus, this approach does not need to consider mobility models as mobility can be compensated via the bypass links. The injection communication paradigm is based on establishing bypass links between carefully selected devices. Herrmann et al. [18] called these dedicated communication points *hub nodes*. Depending on the overall objective, the selection of such devices can be driven by different factors, like for instance to obtain a high local clustering coefficient around the hub, or devices showing certain attributes (e.g. information available or services offered on a particular device). These dedicated devices used for establishing bypass links are called *injection points*. For self-organizing communication networks based on bypass links and injections points as described before we use the term *injection networks*.

Definition 3: Two nodes u and v are called injection points if a bypass link *(u,v)* exists between nodes u and v.

In order to study the small-world properties of such hybrid networks, we had to rely on some ad-hoc network simulator. In our case we used Madhoc [19], an application-level network simulator dedicated to the simulation of mobile ad hoc networks. The main motivation for using Madhoc is its ability to simulate hybrid networks, i.e. mixing different technologies (e.g. bluetooth/Wi-Fi for local connections and UMTS for long distance calls), and its graphical and batch modes of visualization.

B. Small-Worlds

Small-world networks [1] are a class of random graphs that exhibit two main characteristics: a small characteristic path length (L) and a high clustering coefficient (γ). A formal definition of these two graph measures is given below:

Definition 4 (Watts) The local clustering coefficient γ of one node v with k_v neighbors is

$$\gamma_v = \frac{|E(\Gamma_v^r)|}{\binom{k_v}{2}}$$

where $|E(\Gamma_v^r)|$ is the number of links in the relational neighborhood of v and $\binom{k_v}{2}$ is the number of possible links. The clustering coefficient is the average local clustering coefficient for all nodes of a network.

For example, in Figure 4, node a is connected to three nodes b, d and e. The maximum number of possible edges among these three nodes is three. The graph shows that only two out of those three possible edges exist (between b-e and d-e). The edge b-d is missing. So the clustering coefficient for node a is 2/3 or about 0.67. For Figure 4, this value is 0.67. In a physical sense, the clustering coefficient defines the extent to which nodes in the graph tend to form closely-knit groups that have many edges connecting each other in the group, but very few edges leading out of the group.

Definition 5 (Watts) The shortest path length $\overline{d}_v$ connecting each node $v \in V(N)$ of a network N to all other nodes is $d(v,j) \forall\, j \in V(N)$. The characteristic path length L is the median of all shortest paths.

The characteristic path length is a measure of the number of hops necessary to reach any node in the network from any other node. This indicates the degree of separation or connectivity between nodes in the graph. In Figure 4, node a can reach three of the nodes (b, d and e) through just one hop and the fourth node (c)

via two hops. So the characteristic path length for this node is:

$$Ł(a) = \frac{HopsToReachAllNodes}{NumberOfNodes} = \frac{(3\times1)+(1\times2)}{4} = 1.25$$

The characteristic path length (L) for the entire graph, which is the mean of the characteristic path length of all nodes, is equal to 1.2. The challenging as-

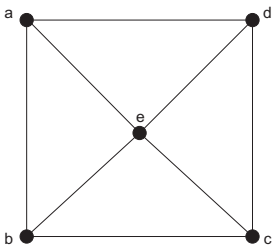


Fig. 4. Graph with $\gamma = 0.67$ and L = 1.2

pect in using small-world properties is that small-world networks combine the advantages of regular networks (high clustering coefficient) with the advantages of random networks (low characteristic path length).

C. Solution Encoding

Solution encoding is a major issue in this kind of algorithms since it will determine the choice of the genetic operators applied for exploring the search space. We have used a binary encoding of the solution in which each gene encodes an integer on 15 bits, that corresponds to one possible bypass link in the half-matrix of all possible links. For instance, if the maximum number of bypass links fixed a priori for the network that is optimized is 10, then for the genGA and the ssGA a chromosome will have 10 genes of 15 bits. Concerning CCGA, it will depend on the number of subpopulations used, if for instance CCGA is used with 5 subpopulations then one chromosome will have $\frac{10}{5} = 2$ genes of 15 bits. Figure 5 shows the example of a chromosome composed of 2 genes (thus the maximum number of created bypass links is 2) on a network of 5 stations. The 5×5 matrix represents all the possible links in the network including the already existing local links in the network (thus of the existing Wi-Fi connections) and the impossible links (i.e. links between two similar stations like station 1 - station 1) that are represented as shaded cells in the matrix (cells number 1, 7, 13, 19 and 25). In the example showed in Figure 5, the first gene (circled) with the integer value 3 stands for the connection between station 1 and station 3 in the corresponding matrix (also circled).

D. Fitness Function

As stated before we have used the Madhoc simulator to experiment injection networks optimization. Indeed, Madhoc allows to simulate and to visualize hybrid adhoc networks (using Wi-Fi, bluetooth, GSM, UMTS), to evaluate small world measures on them and to calculate the number of partitions in the network. In order

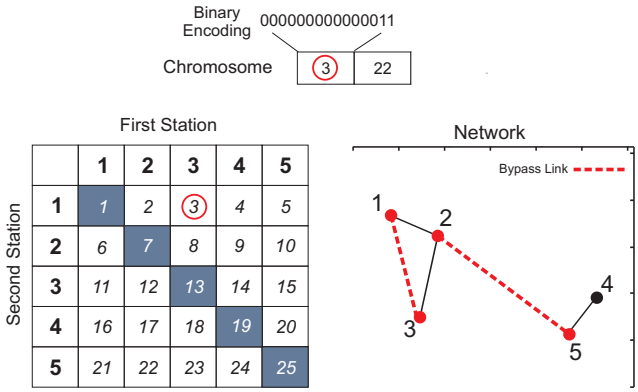


Fig. 5. Solution encoding example

to assign a fitness to the candidate solutions (i.e. sets of possible bypass links) of our algorithms, we use a unique cost function F which combines the two small world measures (L and γ) and the number of created bypass links. When computing the fitness function, we first test if the global network is connected. Indeed, since we use small-world properties as indicators, the network has to be connected in order to compute the characteristic path length (L) on the global network. Thus, if the optimized network is not connected, due to too few or not efficiently placed bypass links, the fitness value is a weighted term of the number of partitions in the network. On the contrary, if the network is connected, the fitness value is a linear combination of the small world measures (clustering coefficient and characteristic path length) and of the difference between the number of bypass links and the maximum number allowed. We look for maximizing the clustering coefficient and minimizing both characteristic path length and number of bypass links. Using this fitness function we thus have a maximization problem as defined in Algorithm IV-D.

Algorithm 1: Fitness Function

```
if Graph connected then
    F = α * γ + β * (L - 1) + δ * (bypassLinks -
    maxBypassLinks )
else
    fitness = 0.1 * numberOfPartitions
end
```

With weights experimentally defined:
$\alpha = 1$
$\beta = 1$ /(numberOfNodes -1)
$\delta = 2$ / (numberOfNodes * (numberOfNodes-1))

bypassLinks is the number of bypass links created in the simulated network by one solution, *maxBypassLinks* (defined a priori) is the maximum number of bypass links that can be created in the network, *numberOfPartitions* is the number of remaining partitions in the whole network after the addition of bypass links and *numberOfNodes* is the number of nodes in the global network.

V. Experimentations

This section presents the results obtained on the injection network optimization problem using the distributed CCGA compared to the results given by the generational GA (genGA) and the steady state GA (ssGA) . We first describe the parameters used for the three genetic algorithm. Next, the configuration of the network simulator is introduced and, finally the results obtained using the CCGA, genGA and ssGA are analyzed and compared.

The algorithms have been implemented in Java and tested on a single node for genGA, ssGA and CCGA and on 6 cluster-nodes for dCCGA (distributed CCGA) all nodes having a 3.7 GHz Xeon processor with 16 GB of RAM, running Debian Linux (with kernel 2.6.9-22) and Java version 1.5.0_05.

A. GA Parameterization

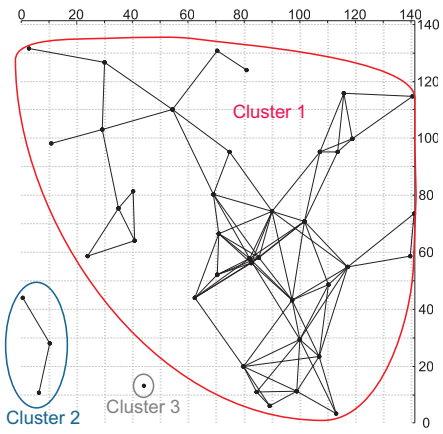


Fig. 6. Studied Networks with 3 clusters

In table I, we show the parameters used for genGA, ssGA, CCGA and dCCGA.

(d)CCGA was tested with 5 subpopulations. For all algorithms we used a randomly generated population composed of 100 individuals. The selection operator is a binary tournament selection (two individuals are selected and the fittest is copied into the intermediate population). The crossover operator is uniform crossover used with probability p_c=0.8. The mutation operator is bit flip mutation in which each allele of the chromosome is flipped with probability p_m= 1/chromosome_length. Concerning the generational GA and (d)CCGA we have added elitism: the best individual found in one generation is thus kept for the next generation.

TABLE I: Parameters used for genGA, ssGA, and (d)CCGA

Number of Subpopulations	5 (only for (d)CCGA)
(Sub)Population size	100 individuals
Termination Condition	50,000 function evaluations
Selection	Binary Tournament
Crossover operator	Uniform, p_c=0.8
Mutation operator	bit flip, p_m = 1/chrom_length
Elitism	1 individual (not for ssGA)

B. Madhoc Configuration

As stated before, the Madhoc simulator was used for managing the complex scenario posed by this injection network problem. We have defined a squared simulation area of 0.2 km^2 and tested with a density of 210 devices per squared kilometer. Each device is equipped with both Wi-Fi (802.11b) and UMTS technologies. The coverage radius of all mobile devices ranges between 20 and 40 meters in case of Wi-Fi. The studied network, as presented in Figure 6, here represents a snapshot of a mobile network in the moment in which a single set of users moved away from each other creating the clusters of terminals, that were obtained using the graphical mode of Madhoc. Used as example, the network with 3 clusters (center of Fig. 6) consists in 42 stations located in three partitions, the first partition has 38 nodes, the second one 3, and the third one has a single node. The number of possible connections in this 3-clusters network is $\frac{numberOfNodes^2}{2} = 882$, the number of existing Wi-Fi connections in this network is 116, thus the number of possible bypass links is 882-116 = 766. The clusters are selected purposely to be different and thus challenging.

TABLE II: Parameterization used in Madhoc

Surface	0.2 km^2
Node Density	210 / km^2
Number of Nodes	42
Partitions	3
Possible Links	766

C. Results

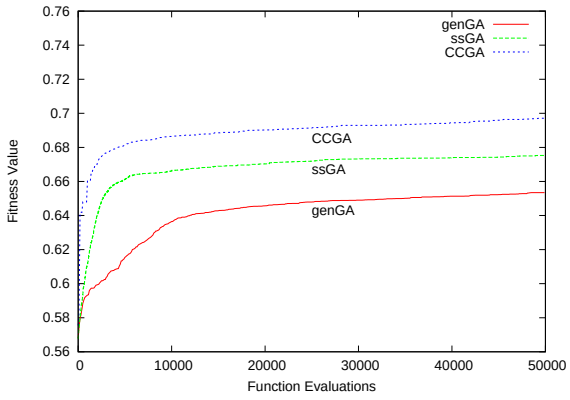


Fig. 7. Average results of 30 runs using genGA, ssGA and CCGA

Each result presented hereafter is the average obtained on 30 independent runs. In order to establish the statistical significance of the means, we first have checked that the data is normally distributed using the Kolmogorov-Smirnov test. If so, we then perform an ANOVA test so as to compare the means otherwise we use a Kruskal-Wallis test [20].

In Table III we show the averaged results for all 30 runs for each algorithm. As it can be seen in Table III, using a CCGA provides better results than both genGA and ssGA, genGA being the least performing

TABLE III: Results of all experiments

Network	GA	Crossover	Time	Result
3 Clusters	genGA	Uniform	58min 12s	0.6534
	ssGA	Uniform	70min 0s	0.6764
	CCGA	Uniform	138min 36s	**0.6971**
	dCCGA	Uniform	98min 55s	**0.6971**

one (with statistical confidence). This can be graphically observed in Figure 7, as well as the better convergence speed of CCGA compared to the other two GAs. As expected the computational time required for dCCGA is lower than for CCGA thanks to the distribution of the subpopulations, however it is still higher than panmictic GAs like genGA and ssGA due to the synchronization between subpopulation induced by the CCGA algorithm.

VI. Conclusion and Future Works

The results presented in this paper belong to an ongoing research on the injection network optimization problem using distributed coevolutionary genetic algorithms. Dafo, our distributed agent framework for evolutionary optimization, including coevolutionary genetic algorithms has been presented. The concept of injection network has been introduced as well as the utilization of small-world properties as indicators for inter-linking network partitions.

Experiments have been conducted using an ad-hoc network simulator on one network scenario composed of 42 stations. Three different GAs, generational, steady-state and cooperative coevolutionary, have been used, each one was tested using uniform crossover. The best result experimentally found on this problem, both in terms of best result found and convergence speed, was using the distributed CCGA. Initial evidence of the capacity of GAs and especially of coevolutionary GAs for solving this problem was also provided in this article.

As a future work, we plan to use some other coevolutionary GAs such as LCGA (Loosely Coupled Genetic Algorithm) to solve this problem. Our next research will also focus on the optimization of dynamic injection networks in which nodes move while optimum injection points are computed at the same time and thus bypass links have to be continuously created and destroyed in order to keep the network unpartitioned.

Acknowledgments

This work has been partially funded by the Spanish Ministry of Science and Technology and FEDER under contract TIN2005-08818-C04-01 (OPLINK project) and by the European CELTIC project (CARLINK).

References

[1] D. J. Watts, *Small Worlds – The Dynamics of Networks between Order and Randomness*. Princeton, New Jersey: Princeton University Press, 1999.

[2] T. Back, D. B. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*. Bristol, UK, UK: IOP Publishing Ltd., 1997.

[3] M. A. Potter and K. De Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving from Nature – PPSN III*. Berlin: Springer, 1994, pp. 249–257.

[4] R. Eriksson and B. Olsson, "Cooperative coevolution in inventory control optimisation," in *Proc. of the Third International Conference on Artificial Neural Networks and Genetic Algorithms*. University of East Anglia, Norwich, UK: Springer-Verlag, 1997.

[5] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[6] D. Whitley and J. Kauth, "GENITOR: A Different Genetic Algorithm," in *Proceedings of the Rocky Mountain Colorado, on Artificial Intelligence*, 1988, pp. 118–130.

[7] J. Paredis, "Coevolutionary life-time learning," in *Parallel Problem Solving from Nature – PPSN IV*. Berlin: Springer, 1996, pp. 72–80.

[8] F. Seredynski, A. Y. Zomaya, and P. Bouvry, "Function optimization with coevolutionary algorithms," in *Proc. of the International Intelligent Information Processing and Web Mining Conference*. Poland: Springer, 2003.

[9] O. Gutknecht and J. Ferber, "Madkit: a generic multi-agent platform," in *Proc. of the fourth international conference on Autonomous agents*. ACM Press, 2000, pp. 78–79.

[10] N. A. Barricelli, "Symbiogenetic evolution processes realized by artificial methods," *Methodos*, no. 9, pp. 35–36, 1957.

[11] P. Ratanchandani and R. Kravets, "A Hybrid Approach to Internet Connectivity for Mobile Ad Hoc Networks," in *Proceedings of IEEE WCNC*, 2003.

[12] A. Andronache, M. R. Brust, and S. Rothkugel, "Multimedia Content Distribution in Hybrid Wireless Networks Using Weighted Clustering," in *WMuNeP '06: Proceedings of the 2nd ACM international workshop on Wireless Multimedia Networking and Performance Modeling*. New York, NY, USA: ACM Press, 2006, pp. 1–10.

[13] N. I. T. Fujiwara and T. Watanabe, "A Hybrid Wireless Network Enhanced with Multihopping for Emergency Communications," in *ICC '04: Proceedings of the IEEE International Conference on Communications*, 2004, pp. 4177–4181.

[14] A. Helmy, "Small Large-Scale Wireless Networks: Mobility-Assisted Resource Discovery," 2002.

[15] O. Dousse, P. Thiran, and M. Hasler, "Connectivity in Ad-Hoc and Hybrid Networks," in *INFOCOM*, 2002.

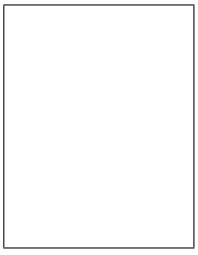
[16] D. Cavalcanti, D. Agrawal, J. Kelner, and D. F. H. Sadok, "Exploiting the Small-World Effect to Increase Connectivity in Wireless Ad Hoc Networks," in *ICT*, ser. Lecture Notes in Computer Science, J. N. de Souza, P. Dini, and P. Lorenz, Eds., vol. 3124. Springer, 2004, pp. 388–393.

[17] J. Li, C. Blake, D. S. D. Couto, H. I. Lee, and R. Morris, "Capacity of Ad Hoc Wireless Networks," in *MobiCom '01: Proceedings of the 7th annual international conference on Mobile Computing and Networking*. New York, NY, USA: ACM Press, 2001, pp. 61–69.

[18] K. Herrmann and K. Geihs, "Self-Organization in Mobile Ad hoc Networks based on the Dynamics of Interaction," Erlangen, Germany, 2003.

[19] L. Hogie, P. Bouvry, F. Guinand, G. Danoy, and E. Alba, "Simulating Realistic Mobility Models for Large Heterogeneous MANETS," in *Demo proceeding of the 9th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM'06)*. IEEE, October 2006.

[20] E. Alba, *Parallel Metaheuristics: A New Class of Algorithms*. Wiley-Interscience, 2005.



Grégoire Danoy received the Industrial Engineer Degree in Computer Science from Luxembourg University of Applied Sciences in 2003 and the M.S. degree in Web Intelligence from Ecole des Mines of Saint-Etienne, France, in 2004. He is currently working toward the PhD degree with the University of Luxembourg and the Ecole des Mines of Saint-Etienne (France). His current research interests include nature inspired algorithms, multi-agent systems, dynamic optimization.