Multi-hop ad hoc networks allow establishing local groups of communicating devices in a self-organizing way. However, when considering realistic mobility patterns, such networks most often get divided in a set of disjoint partitions. This presence of partitions is an obstacle to communication within these networks. Ad hoc networks are generally composed of devices capable of communicating in a geographical neighborhood for free (e.g. using Wi-Fi or Bluetooth). In most cases a communication infrastructure is available. It can be a set of access point as well as a GSM/UMTS network. The use of such an infrastructure is billed, but it permits to interconnect distant nodes, through what we call ”bypass links”. The objective of our work is to optimize the placement of these long-range links. To this end we rely on small-world network properties, which consist in a high clustering coefficient and a low characteristic path length. In this article we investigate the use of three genetic algorithms (generational, steady-state, and cooperative coevolutionary) to optimize three instances of this topology control problem and present initial evidence of their capacity to solve it.

Keywords: Genetic algorithms, Optimization, Telecommunications, Performance analysis

1. Introduction

Multi-hop ad hoc networks are networks composed of communicating devices capable of spontaneously interconnecting without any pre-existing infrastructure. Among the possibilities of utilization, we can cite: positioning in cities, gaming, tourism, art
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. Such ad hoc networks are intrinsically dynamic. Due to their limited transmission range, these networks face partitioning problems that penalize 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), are also called bypass links, to inter-link 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 feature a high clustering coefficient ($\gamma$) 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 $\gamma$, minimizing $L$) and to minimize the number of required bypass links in the network, we here rely on Evolutionary Algorithms (EAs) and more specifically on Genetic Algorithms (GAs). These metaheuristics can provide low-cost operations in the optimization process and allow the designer to put some ”intelligence” or sophistication in the design (they are also known as computational intelligent tools). GAs have proved their ability for solving complex real-world problems; they have been extensively used thanks to their capacity of providing an accurate (possibly optimal) solution in a reasonable time. Since this problem is new to the metaheuristic community, we start by investigating the kind of evolution step more amenable to our problem by analyzing three proposals: a generational GA, a steady-state GA, and a cooperative coevolutionary GA on three different instances of a partitioned ad hoc network.

We further will investigate the influence of the solution representations and of the crossover operators on the final quality of the results, as an important methodological step in applying GAs to complex problems.

The remainder of this paper is organized as follows. In the next section we introduce the latest key related works. In Section III we give a detailed view on the injection network problem. Section IV provides details on our three algorithms, considering two representations and two crossover operators. The section is concluded by a detailed description of the fitness function we defined. Then, in Section V, we present the experiments and discuss the results. The last section contains our conclusions and perspectives.
2. Related Works

Mobile multi-hop ad hoc networks have brought several difficult (and practical) challenges. Among them, we will focus on network partitioning which is of importance for communication performance as well as routing mechanisms. Some past works advise 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\(^{22}\). However, such hybrid wireless networks are often not feasible, because of economical and implementation issues.

In order to tackle the same problem, some recent researches started investigating the advantages of bringing small-world properties to such networks. Let us highlight the following ones. Dousse in\(^9\) introduces base stations, connected through a fixed wired infrastructure, in order to increase connectivity in ad hoc networks, thus realizing global reachability. Helmy\(^12\) considers using the uniform distribution (of random links) with which the objective is to reduce the number of queries during the search for a given target node. In a similar way, Reznik et al. study the effect of randomly adding point-to-point wired links in a square grid ad hoc network in\(^3\). They provide an elaborate framework with a specific parameterized distribution for choosing long links. This helps to reduce the path length to a small power of the initial diameter. In his PhD dissertation\(^18\), Nguyen investigates the use of small-world graphs for network design. To this end, he extends Reznik’s framework by adding a cost (weight of long links) and congestion but he still studies square grid networks and his long-links construction scheme implies a high number of them. In\(^24\), Sharma also investigates the use of small-world properties for the addition of wired links in hybrid wireless sensor networks. His objective is to reduce the energy dissipation per node. Alternatively, an infrastructureless setting is of interest where problems of restricted geographical regions are avoided. Watts\(^27\) 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\(^6\), e.g. small distance Bluetooth links along with higher distance Wi-Fi links.

Some other researches focus on the optimization of this topology control problem using evolutionary algorithms, including genetic algorithms. In\(^8\), Lee uses a multi-objective genetic algorithm, named GA-OTC, for topology control in wireless sensor networks. The algorithm searches for the optimal clustering set of sensor nodes so as to fulfill the objectives which are to balance and minimize energy consumption. In\(^19\), Pandey uses a genetic algorithm to optimize the placement of special multi-interface devices called *drones* in an ad hoc heterogeneous network comprising of several underlying homogeneous networks. *p* drones have to be placed to improve the network connectivity while minimizing the network partitions and the number of interfaces on each drone.

However, in the current literature there is no research merging both the optimization of such hybrid ad hoc networks with the small-world properties of such
networks. This has motivated the definition of the problem we introduce in the following section.

3. The Problem

The problem we study in this article, consists in overcoming partitioning in ad hoc networks by optimizing the placement of long range links that we call *bypass 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 supplementary communication links, that we call *bypass links*, between two arbitrary devices. These links will induce additional costs. Let us formalize 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. 1. Example of an Injection Network.](image)

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.
The injection communication paradigm is based on establishing bypass links between carefully selected devices. Herrmann et al. \textsuperscript{13} called these dedicated communication points \textit{hub nodes}. We call these dedicated devices used for establishing bypass links \textit{injection points}.

\textbf{Definition 3.} Two nodes \(u\) and \(v\) are called injection points if a bypass link \((u,v)\) exists between nodes \(u\) and \(v\).

Injection points serve two different purposes: a point where information dissemination starts and where services are being placed (service placement, Herrmann et al. \textsuperscript{13}). In the first case, the injection point is of essential importance at the moment of receiving information and passing this information to the neighborhood. The injection point might represent a bottleneck, depending on the amount of data passing through. In addition, injection points become particularly attractive when offering a service. In fact, information dissemination can be seen as such a service that is usable by devices in the injection points surroundings. Different criteria for determining the injection point can be of importance. Supposing that, for instance, the device is highly clustered and thus one of the central members of a group, epidemic behavior for information spreading will take effect faster. Therefore, the current environment and the device’s relationship to its neighbors are important.

For self-organizing communication networks based on bypass links and injections points as described before we use the term \textit{injection networks}.

In this optimization problem, we consider small-world properties as indicators for the good set of rules to maximize the bypass links efficiency. Small-World networks \textsuperscript{27} are a class of random graphs that exhibit a small characteristic path length (\(L\)), indicating the degree of separation between the nodes in the graph, and a high clustering coefficient (\(\gamma\)), defining 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. The challenging aspect 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). 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 \textsuperscript{14}, 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, which greatly help in understanding the network design alternatives.
4. The Algorithms

In this section we first introduce the three genetic algorithms we used. Next we provide details on the two solution encodings and on the two crossover operators we applied. Finally we present the fitness function we have defined.

The use of evolutionary computation (EC) techniques to evolve solutions for both abstractions and real-life problems has seen a dramatic increase in popularity and success over the last decade. The most popular and widely applied EC technique is the sequential GA\textsuperscript{17}, whose computational scheme is based on a set (population) of potential solutions (individuals) on which it applies some stochastic operators in order to search for an optimum. It uses a single population (panmixia) of individuals and apply operators to them as a whole.

Past works have shown that the underlying iterative step of the GA is very influential in some applications\textsuperscript{2}. Therefore in this work we focus on a generational, a steady-state and a cooperative coevolutionary GA.

4.1. Generational Genetic Algorithm

The generational genetic algorithm (the ”Standard Genetic algorithm”)\textsuperscript{7} creates new offspring from the members of an old population using the genetic operators, placing these individuals in a new population\textsuperscript{11, 15} (although some of these new individuals might be identical to their parents). The basic step creates a whole new population that replaces the old one (the best individual is preserved: elitism). This algorithm is expected to preserve diversity for a large number of generations, showing coarse grain steps in which new populations are created (see Algorithm 1).

\begin{algorithm}
\caption{Generational GA.}
\begin{algorithmic}
\STATE Generate initial population $P_t$
\STATE Evaluate population $P_t$
\WHILE{Stopping criteria not satisfied}
\FOR{$i$ = 1 to pop\_size}
\STATE Select individuals $i_1$ and $i_2$ from $P_t$
\STATE $i' = \text{Crossover}(i_1, i_2)$
\STATE $i'' = \text{Mutate}(i')$
\STATE Insert individual $i''$ in $P_{t+1}$
\STATE $P_t = P_{t+1}$
\ENDFOR
\STATE Evaluate population $P_t$
\ENDWHILE
\end{algorithmic}
\end{algorithm}

4.2. Steady State Genetic Algorithm

The incremental/steady state genetic algorithm (ssGA)\textsuperscript{28} differs from the generational model in the sense that only a few individuals are replaced in each generation (one single individual in our case). A replacement/deletion strategy defines which
member(s) of the population will be replaced by the new offspring(s). In this paper the least fit individual is replaced by the offspring resulting from crossover and mutation of the selected individuals (see Algorithm 2).

**Algorithm 2**: Steady State GA.

- Generate initial population $P$.
- Evaluate population $P$.
- **while** Stopping criteria not satisfied **do**
  - Select individuals $i_1$ and $i_2$ from $P$.
  - $i' = \text{Crossover}(i_1, i_2)$.
  - $i'' = \text{Mutate}(i')$.
  - Evaluate ($i''$).
  - Insert individual $i''$ in $P$.

4.3. **Cooperative Coevolutionary Genetic Algorithm**

Just like the genetic algorithm are inspired from nature, the concept of coevolution (used as the foundation for coevolutionary algorithms) comes from biological observations. Indeed, nature is composed of several species that coevolve, that is whose individual evolution is directed by the the evolution of other species. Instead of considering a population of similar individuals representing a global solution (like classical genetic algorithms do), coevolutionary algorithms consider the coevolution of subpopulations of individuals representing specific parts of the global solution.

These algorithms were already applied in for parallel and distributed optimization of a number of test functions known in the area of evolutionary computation. It was then demonstrated that these two coevolutionary algorithms outperform a sequential GA.

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 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. Potter’s methods have also
been used or extended by other researchers, for instance Eriksson and Olsson
have used a cooperative coevolutionary algorithm for inventory control parameter
optimization.

Algorithm 3: CCGA.

\[
\begin{align*}
\text{gen} &= 0 \\
\text{foreach} \ species \ s \ do \\
& \quad \text{Pop}_s(\text{gen}) = \text{randomly initialized population} \\
& \quad \text{evaluate fitness of each individual in } \text{Pop}_s(\text{gen}) \\
\text{while} \ \text{termination condition} = \text{false} \ do \\
& \quad \text{gen} = \text{gen} + 1 \\
& \quad \text{foreach} \ species \ s \ do \\
& \quad \quad \text{select } \text{Pop}_s(\text{gen}) \text{ from } \text{Pop}_s(\text{gen} - 1) \text{ based on fitness} \\
& \quad \quad \text{apply genetic operators to } \text{Pop}_s(\text{gen}) \\
& \quad \quad \text{evaluate fitness of each individual in } \text{Pop}_s(\text{gen})
\end{align*}
\]

4.4. Solution Encodings

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.

4.4.1. First Encoding

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 a chromosome will have 10 genes of 15 bits. Figure 2 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 $5 \times 5$ half-matrix represents all the possible links in the network including the already existing local links in the network (i.e. the existing Wi-Fi connections). In the example showed in Figure 2, the first gene (circled) with the integer value 2 stands for the connection between station 1 and station 3 in the corresponding half-matrix (also circled).

4.4.2. Second Encoding

![Figure 2](image)

Fig. 2. Example of a chromosome composed of 2 genes on a network of 5 stations.

The second encoding is also binary. Each bit represents one possible bypass link in the network. If the bit value is 1 then the corresponding bypass link is created and if it is 0 it is not created. Let us take as example the same network as in Figure 2, in which the network is composed of 5 stations and 3 existing wireless links. The number of possible links in this network is

$$\frac{(N) \times (N - 1)}{2} = \frac{5 \times 4}{2} = 10$$

with $N =$ numberOfNodes. The number of possible bypass links is finally $10 - 3 = 7$. The resulting number of bits in the chromosome is 7, as shown in Figure 3. The half-matrix represents the possible links in the network and the light gray shaded cells represent the already existing wireless links that are not considered (i.e. links between stations 1-2, 2-3 and 4-5). In our example, the first bit in the chromosome thus stands for the possible bypass link between station 1 and station 3, which is created since its value is 1, the second bit stands for the link between station 1 and station 4, this one is not created since its bit value is 0, and so on. Contrary to the first encoding, this second encoding depends on the network size and on the number of existing links. Consequently, the bigger the network, the bigger the chromosome.
4.5. Crossover Operators

The crossover operation, also named as recombination, produces new individuals by combining the information contained in two or more parents. This is done by combining the values of the two parents. Our experiments have been conducted using both two-point crossover and uniform crossover to further investigate the interest and the influence of each operators using our two representations for the injection network problem.

In two-point crossover, two crossover positions are selected uniformly at random and the values between these points are exchanged. Then two new offspring are produced.

![Two Point Crossover](image)

Fig. 4. Two Point Crossover.

Single and multi-point crossover define cross points as places between loci where an individual can be split. Uniform crossover, however, generalizes this scheme to make every locus a potential crossover point. A crossover mask, the same length as the individual structure, is created at random and the parity of the bits in the mask indicates which parent will supply the offspring at every position.

Consider the following two individuals with 10 binary variables each (shown in Figure 5):

![Uniform Crossover](image)

Fig. 5. Uniform Crossover.

For each variable the parent who contributes its value to the offspring is chosen randomly with equal probability. Here, the offspring 1 is produced by taking the bit from parent 1 if the corresponding mask bit is 1 or the bit from parent 2 if the corresponding mask bit is 0. Offspring 2 is created using the inverse of the mask, usually.

Uniform crossover, like multi-point crossover, has been claimed to reduce the bias associated with the length of the binary representation, and the particular coding for a given parameter set. We thus investigate the effect of these two crossover
operators on the injection network problem since the problem at hands is new and thus unknown.

4.6. *Fitness Function*

As stated before, we relied on small-world networks properties so as to optimize the placement of the bypass links. We have conducted our experiments using the Madhoc simulator which to simulate and to visualize hybrid ad hoc networks (using Wi-Fi, bluetooth, GSM, UMTS). We extended Madhoc in order to make it support bypass links and to measure small world properties.

In order 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 $\gamma$) 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. 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 consequently have a maximization problem as defined in Algorithm 4.

**Algorithm 4: Fitness Function.**

```plaintext
if Graph connected then
    F = $\alpha * \gamma - \beta * (L - 1) - \delta * (bl - bl_{max})$
else
    fitness = $\xi * P$
end
```

With weights experimentally defined:

$\alpha = 1$

$\beta = 1 \div (N - 1)$

$\delta = 2 \div (N * (N - 1))$

$\xi = -0.1$

$bl$ is the number of bypass links created in the simulated network by one solution, $bl_{max}$ (defined a priori) is the maximum number of bypass links that can be created in the network, $P$ is the number of remaining partitions in the whole network after the addition of bypass links and $N$ is the number of stations in the global network.
5. Experimentation

This section presents the results obtained on the injection network optimization problem using the three GAs, generational, steady state and cooperative coevolutionary. We first describe the parameters used for the genetic algorithms. Next, the configuration of the network simulator is introduced and, finally the results obtained using the three GAs, using the two representations and the two crossover operators (2-point and uniform), are analyzed and compared.

The algorithms have been implemented in Java and tested on a 3.2 GHz Xeon processor with 4 GB of RAM, running Debian Linux (with kernel 2.6.9-22) and Java version 1.5.0_05.

5.1. GA Parameterization

Table 1. Parameters for genGA, ssGA and CCGA.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>100 indiv. (10x10 for CCGA)</td>
</tr>
<tr>
<td>Termination Condition</td>
<td>50,000 function evaluations</td>
</tr>
<tr>
<td>Selection</td>
<td>Binary Tournament</td>
</tr>
<tr>
<td>Crossover operators</td>
<td>2-point and uniform, ( p_c = 0.8 )</td>
</tr>
<tr>
<td>Mutation operator</td>
<td>bit flip, ( p_m = 1/\text{chrom}_\text{length} )</td>
</tr>
<tr>
<td>Elitism</td>
<td>1 individual (not for ssGA)</td>
</tr>
</tbody>
</table>

In table 1, we show the parameters used for genGA, ssGA and CCGA.

We used a randomly generated population composed of 100 individuals for genGA and ssGA and 10 subpopulations of 10 individuals for CCGA. The selection operator for genGA and CCGA is a binary tournament selection (two individuals are selected and the fittest is copied into the intermediate population). For ssGA we have used a replace-worst strategy. As stated before, the two crossover operators (separately) analyzed are 2-point and 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 \( p_m = 1/\text{chrom}_\text{length} \). Concerning the generational GA and CCGA we have added elitism: the best individual found in one generation is thus kept for the next generation.

5.2. Madhoc Configuration

As stated before, the Madhoc simulator was used for managing the complex scenarios posed by this injection network problem. Figure 6 shows how the genetic algorithms interact with Madhoc.

We have defined a squared simulation area of 0.2 \( km^2 \) and tested three different densities of 150, 210 and 350 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.
Table 2. Parameterization used in Madhoc.

<table>
<thead>
<tr>
<th></th>
<th>1 Cluster</th>
<th>3 Clusters</th>
<th>5 Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Surface</td>
<td>0.2 km²</td>
<td>0.2 km²</td>
<td>0.2 km²</td>
</tr>
<tr>
<td>Node Density</td>
<td>350/km²</td>
<td>210/km²</td>
<td>150/km²</td>
</tr>
<tr>
<td>Number of Nodes</td>
<td>70</td>
<td>42</td>
<td>30</td>
</tr>
<tr>
<td>Partitions</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Possible Links</td>
<td>2189</td>
<td>745</td>
<td>400</td>
</tr>
</tbody>
</table>

The studied networks, as presented in Figure 7, here represent a snapshot of mobile networks 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. As an example, the network with 3 clusters (center of Fig. 7) consists in 42 stations located in three partitions, the first partition has 38 stations, the second one 3, and the third one has a single station. The number of possible connections in this 3-clusters network is $N*(N-1)/2 = 861$. The number of existing Wi-Fi connections in this network is 116, thus the number of possible bypass links is 861-116 = 745. The clusters are selected purposely to be different and thus challenging.
5.3. Results

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.

Table 3. Results of all experiments.

<table>
<thead>
<tr>
<th>Network</th>
<th>GA</th>
<th>Representation</th>
<th>Crossover</th>
<th>Time (s)</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>genGA</td>
<td>1st Rep.</td>
<td>DPX</td>
<td>8176</td>
<td>0.6788</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UX</td>
<td>8083</td>
<td></td>
<td>0.6704</td>
</tr>
<tr>
<td>1 Cluster</td>
<td></td>
<td>2nd Rep.</td>
<td>DPX</td>
<td>28456</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UX</td>
<td>23192</td>
<td></td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>ssGA</td>
<td>1st Rep.</td>
<td>DPX</td>
<td>8427</td>
<td>0.6808</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UX</td>
<td>8852</td>
<td></td>
<td>0.6745</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2nd Rep.</td>
<td>DPX</td>
<td>19203</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UX</td>
<td>15284</td>
<td></td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>CCGA</td>
<td>1st Rep.</td>
<td>DPX</td>
<td>17802</td>
<td>0.7378</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UX</td>
<td>17730</td>
<td></td>
<td>0.7386</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2nd Rep.</td>
<td>DPX</td>
<td>42683</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UX</td>
<td>27290</td>
<td></td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>genGA</td>
<td>1st Rep.</td>
<td>DPX</td>
<td>4470</td>
<td>0.6620</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UX</td>
<td>3492</td>
<td></td>
<td>0.6554</td>
</tr>
<tr>
<td>3 Clusters</td>
<td></td>
<td>2nd Rep.</td>
<td>DPX</td>
<td>8913</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UX</td>
<td>7305</td>
<td></td>
<td>0.6851</td>
</tr>
<tr>
<td></td>
<td>ssGA</td>
<td>1st Rep.</td>
<td>DPX</td>
<td>3251</td>
<td>0.6764</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UX</td>
<td>4260</td>
<td></td>
<td>0.6752</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2nd Rep.</td>
<td>DPX</td>
<td>4179</td>
<td>0.6561</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UX</td>
<td>5189</td>
<td></td>
<td>0.6870</td>
</tr>
<tr>
<td></td>
<td>CCGA</td>
<td>1st Rep.</td>
<td>DPX</td>
<td>6669</td>
<td>0.7339</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UX</td>
<td>10016</td>
<td></td>
<td>0.7361</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2nd Rep.</td>
<td>DPX</td>
<td>16574</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UX</td>
<td>13889</td>
<td></td>
<td>0.7255</td>
</tr>
<tr>
<td></td>
<td>genGA</td>
<td>1st Rep.</td>
<td>DPX</td>
<td>1669</td>
<td>0.5619</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UX</td>
<td>1647</td>
<td></td>
<td>0.5525</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2nd Rep.</td>
<td>DPX</td>
<td>3110</td>
<td>0.4748</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UX</td>
<td>2622</td>
<td></td>
<td>0.5187</td>
</tr>
<tr>
<td>5 Clusters</td>
<td></td>
<td>1st Rep.</td>
<td>DPX</td>
<td>1701</td>
<td>0.6678</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UX</td>
<td>1743</td>
<td></td>
<td>0.6718</td>
</tr>
<tr>
<td></td>
<td>ssGA</td>
<td>2nd Rep.</td>
<td>DPX</td>
<td>2014</td>
<td>0.6433</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UX</td>
<td>2030</td>
<td></td>
<td>0.6561</td>
</tr>
<tr>
<td></td>
<td>CCGA</td>
<td>1st Rep.</td>
<td>DPX</td>
<td>8698</td>
<td>0.669</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UX</td>
<td>7295</td>
<td></td>
<td>0.6788</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2nd Rep.</td>
<td>DPX</td>
<td>8811</td>
<td>0.1057</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UX</td>
<td>8516</td>
<td></td>
<td>0.7015</td>
</tr>
</tbody>
</table>

In Table 3 we show the averaged results and the total computational time for all 30 runs for each algorithm, representation and crossover operator.

5.3.1. Comparison of the Algorithms

Comparing the three algorithms in terms of best results found, it clearly appears that CCGA outperforms the two panmictic algorithms by reaching the highest fitness for the three network scenarios (see Table 3). We can also see that ssGA always
performs better than genGA. Another interesting property is that the difference between the genGA and the ssGA increases as the number of simulated stations decreases (see 8 and 9). Between ssGA and CCGA, the behavior is opposite, i.e. the difference decreases as the number of simulated stations decreases. Regarding the execution time required, we see that CCGA requires from two to four times more computational time than the genGA and the ssGA, the bigger difference being observed on the 5-clusters network. The highest computational time is reached on the biggest network (i.e. 1-cluster network) by the CCGA using the second representation and 2-point crossover.

5.3.2. Comparison of the Representations
As can be seen in Table 3, using the first representation allows all the algorithms to find a solution in each experiment. Using the second representation highly deteriorates the results when the number of stations in the simulated network is high. Indeed no algorithm manages to find a solution that does not overpass the constraint of 10 bypass links on the 1-cluster network (see top-left corner of Figure 9. This difference does not exist when the number of stations decreases, since both genGA and ssGA obtain better results using the second representation on the 3-clusters
network. On the 5-clusters network, the results slightly deteriorated except for the CCGA with uniform crossover that obtains the overall best result. One drawback of the second representation is a slower convergence speed than with the first representation. Another disadvantage of the second representation concerns the execution time that is in average twice longer than with the first representation. This is due to the high number of bypass links that are created using this representation.

5.3.3. **Comparison of the Crossover Operators**

As it can be seen in Table 3, using the uniform crossover provides better results than 2-point crossover on the majority of the experiments. The contrary is only true when using the first representation for the generational and the steady state GAs respectively on the 3 network instances and on the first two network instances (1-cluster and 3-clusters). CCGA behaves the opposite way according to the crossover operator, since CCGA with uniform crossover always outperforms CCGA with 2-point crossover. Concerning the influence of the crossover operators on the execution time, we can see that ssGA behaves the opposite way from genGA and CCGA. Indeed ssGA requires more time when using uniform crossover (except on the 1-cluster network with the second representation) contrary to genGA and CCGA.
(except on the 3-clusters network with the first representation) that are faster with a uniform crossover.

5.3.4. Comparison of the Computational Speed

![Graph showing computational speed per algorithm and representation for each network.](image)

Fig. 10. Computational speed per algorithm and representation for each network.

Figure 10 shows the computational speed of each combination of algorithm and representation for the three network scenarios. Each column is thus the average of the computational time obtained using both two-point and uniform crossover. We can clearly observe that the bigger the network the bigger the calculation time. As previously mentioned, we can graphically see that using the second representation always increases the amount of time required. CCGA is always slower than genGA and ssGA. The two panmictic algorithms perform best using the first representation, the genGA being the overall best algorithm for the 1-cluster and 5-clusters networks and the ssGA for the 3-clusters network. Finally we can notice that in terms of computational time the ssGA is less sensible to the representation than the other two algorithms.

5.3.5. Analysis of the Problem

In this subsection we analyze the most complex problem among the three we have used, i.e. the 1-cluster network. We have studied the evolution of the number of bypass links created by each algorithm according to the number of function evaluations (see Figure 11). It is interesting to notice the difference of behavior between the two panmictic GAs and the coevolutionary GA. Indeed both generational and steady state GAs first start by decreasing the number of bypass links (the first 2500 evaluations for the ssGA and between 7500 and 12500 for the genGA) before raising as opposed to the CCGA that directly increases this value. Another noticeable difference is the perturbation that feature the two genGA curves contrary to the ssGA and the CCGA ones. The ssGA with two-point crossover reaches a lower number of
bypass links than the genGA with two-point crossover, and additionally provides a better final result. We can also observe that only CCGA reaches the limit of 10 bypass links, which could be considered as a worse behavior than the other two algorithms, but it allows much better final results.

Our second investigation focused on the stations in the network that are the most often elected as injection points. Figure 12 shows the number of times each station has been chosen as injection point in one experiment. This value is the average of all the algorithms that managed to obtain a non-partitioned network (see Table 3). The 20 stations that were chosen as best injection point candidate are represented using filled rectangles.

Figure 13 permits to locate in the network the same 20 stations which were the most often elected as injection point. It is interesting to notice that not only stations that are located inside high density areas are good injection point candidates (e.g. station 41 or 47), but also stations that interconnect high density areas, like station 46 and 61.
6. Conclusion and Future Works

The results presented in this paper belong to an ongoing research on the injection network optimization problem using genetic algorithms. 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 three network scenarios composed of 30, 42 and 70 stations. Three different GAs, generational, steady-state and cooperative coevolutionary, have been used, and each one was tested using two solution encodings and two crossover operators (2-point and uniform). The best combination experimentally found on this problem is a CCGA applying a uniform crossover either using the first representation for the 1-cluster and 3-clusters networks or using the second representation for the 5-clusters network. Initial evidence of the capacity of GAs for solving this problem were also provided in this article.

Our next research will 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

19. S. Pandey and P. Agrawal. A unifying architecture for maximal connectivity in hetero-


