An Evolutionary Approach to the Timed Multi-Agent Patrolling Problem

Vítor de Albuquerque Torreão
Recife, Pernambuco, Brazil
vitordeatorreao@gmail.com

Pablo Azevedo Sampaio
DC - UFRPE
Recife, Pernambuco, Brazil
pablo.sampaio@ufrpe.br

Péricles B. C. de Miranda
DC - UFRPE
Recife, Pernambuco, Brazil
pericles.miranda@ufrpe.br

***Abstract*—The Timed Multi-agent Patrolling is a complex multi-agent problem, which requires a group of agents to move cooperatively to obtain optimal results for the whole group. Patrolling a country's borders, watching the corridors of a building, monitoring maritime fleets, inspecting areas subject to gas leakage or fires, and traffic inspection are examples of applications of this problem. Different works have applied many heuristic techniques to solve the general case of the Multi-Agent Patrolling problem. Nonetheless, although it is a combinatorial problem, the application of evolutionary algorithms to directly address this problem has not been investigated yet. The main contribution of this work is the proposal of a new representation of individuals together with new operators to solve the patrolling problem. These operators were also assembled in four new complete evolutionary algorithms to address the problem. Empirical analysis using computer simulations shows that these approaches output effective solutions with competitive performances when compared to some traditional strategies proposed in the literature.**

***Index Terms*—multi-agent patrolling, multi-robot patrolling, coordination and patrolling, evolutionary algorithms**

I. Introduction

In this work, we call patrolling tasks those that require that a team of humans beings moves throughout a region repeatedly to perform some action in a set of previously chosen interest points in that region. Typical examples are security tasks where agents (e.g. guards) move continuously in a terrain to prevent or detect intruders. Besides, other patrolling tasks are related to continuous inspection and control, such as monitoring maritime fleets, inspecting areas subject to gas leakage, rescue tasks, traffic monitoring (by guards), continuous equipment maintenance and even cleaning floors [1]. Human factors such as boredom, distraction, and fatigue can deteriorate the performance of the human teams in theses tasks [2]. Therefore, the replacement of human beings by autonomous entities can potentially improve the reliability of theses systems.

To study the autonomous execution of these tasks, authors have modeled them using many different formal definitions [3]–[7]. In this work, we adopt the Timed Multi-Agent Patrolling (TMAP) definition which consists in coordinating a team of agents to walk throughout an environment repeatedly visiting a finite set of interest points during an arbitrary time, to optimize a metric (to measure the agents' performance) based on the timestamps of the visits [1]. This definition is sufficiently general to encompass the majority of works about patrolling found in the literature and is the definition of patrolling that we adopt in this paper.

The TMAP is ideal to study multi-agent coordination, because it is simple to be understood but also a challenging task allocation problem [5]. It has been proved that the TMAP problem is, at least, NP-Complete [8]. Therefore, most researchers have focused on heuristic solutions, created with some different approaches. Some solutions are based on specialized path-finding techniques [9], [10], some agents coordinate by leaving information in the environment [11], [12], other solutions use learning techniques [13] [14], others use multi-agent negotiation [4], other use information spreading through the environment [15], among many other specialized heuristics [9], [10], [16]. As TMAP is an optimization problem, one of the possibilities to solve it is by applying general-purpose optimization algorithms, like those from Evolutionary Computing paradigm (e.g. Genetic Algorithm (GA) and Evolutionary Strategy (ES)). However, a small number of related works were found in the literature, and they were all based on Ant Colony Optimization (ACO)[16], [17].

In this work, we propose the application of evolutionary algorithms to calculate solutions for specific instances of TMAP. As these techniques have not been investigated for TMAP, it was necessary to create novel evolutionary operators exclusively for the problem at hand. In especial, we proposed a novel structure of individuals to represent a solution for a TMAP instance properly, as well as new mutation and recombination operators for this structure. These operators were used in traditional Evolutionary Algorithm (EA), like GA and ES. To evaluate the proposal, we performed experiments comparing the output of these algorithms to some patrolling algorithms frequently used in literature [15] [8] over different sizes of agent societies in topologically distinct graphs.

This paper is organized as follows: section II provides a brief review of the literature regarding the TMAP problem; section III details the proposed evolutionary solution to the TMAP; section IV presents the experiments made to evaluate the effectiveness of those solutions; the paper ends in section V which provides a final discussion of the proposed solutions and presents some conclusions.

II. Background

The patrolling task has been previously studied with different approaches, but most of them agree in modeling the environment, where the agents patrol, as a graph [18]. This is because it allows the problem to be easily adapted to a variety of domains ranging from terrains to computer networks [10]. In addition to that, graphs are preferable because they are simple yet sophisticated enough to capture the relevant characteristics of the environment [1]. This is confirmed by [5] which has implemented solutions to the TMAP using graphs and later applied it in real world environments.

The first theoretical analysis of the TMAP was made in [8]. It was also proposed a solution based on the Travelling Salesman Problem (TSP), where a cycle which covers the entire graph is found, and the agent is configured to follow this cycle indefinitely. It is demonstrated that this strategy is optimal in the maximum idleness metric for only one agent. However, it is known that the TSP is an NP-Complete problem, so the authors have used Christofides' algorithm to find an approximation to the Hamiltonian cycle in polynomial time [19]. For the multi-agent scenario, the solution involves finding the cycle, then positioning every agent with a gap between them, and finally configuring them to follow the cycle. In [10], empirical comparisons found this Single Cycle strategy to be the best performing.

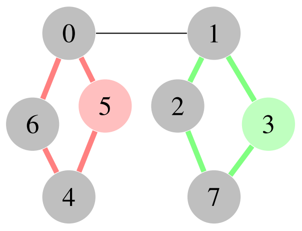
In [1], a new family of strategies for solving the TMAP is presented. These approaches are inspired by the way forces propagate in nature, specifically, Newton's second universal gravitation law. An imaginary mass is assigned to each interest point according to its idleness. These masses exert forces over the patrolling agents. The intensity of a given force is obtained by applying potential functions inspired by the universal gravitational law. The ending result is the agents move towards regions of the environment where there are more points of interest with high idleness. The work also presents simulation experiments which show that these gravitational strategies perform very well when compared to other traditional approaches.

Comparative works performed in [10] and [1] have shown that there are neither optimal nor clearly dominant strategies for the TMAP. This leaves the problem open for new solutions to be proposed. In fact, there have been many different approaches to solving it, for example, partition-based strategies [5] and ant colony-based [11], [12], [20].

In [17], a two-step hybrid approach is proposed. First, a genetic algorithm is used to position the agents as further apart as possible in the environment. Then, an ACO approach is used to calculate partitions in the graph. Therefore, the evolutionary algorithm is not being used to solve the patrolling problem itself, but rather to find the furthest positions to place the agents. In a later work, this same strategy is applied in a setting of the TMAP where agents move with different speeds and points of interest have distinct priorities [16].

One can see, then, even though EAs have been widely used for solving other optimization problems, its application to TMAP was not investigated yet. To apply this class of algorithms to TMAP it was necessary to define novel evolutionary operators. Therefore, in the following section of this paper we present the proposed EAs to solve the TMAP.

Fig. 1: A TMAP solution example. Two agents, red and green, patrol the environment. Each is responsible for a partition of the graph, where they will walk a cycle indefinitely. The red agent has its center at node 5, whereas the green agent is centered at node 3.



III. Proposed Approach

This paper proposes the use of EAs to obtain solutions to the TMAP. To apply this class of algorithms to a given problem, two aspects need to be defined. First, how the individuals are represented; and second, three operators: one for creating a new individual, one for mutating an existing individual into another and, lastly, one for creating new individuals from pre-existing ones, an operation known as crossover or recombination. These three operators will, typically, depend on the chosen representation. The most popular data structures used to represent individuals in EAs are the bit vectors and real numbers [21]. However, as discussed earlier, the TMAP is more naturally mapped into graph structures.

In this paper, an individual will be treated as being a solution to a specific instance of the TMAP, i.e. a society of agents which are going to walk the graph indefinitely, visiting the nodes as frequently as possible. This solution is then represented by a complex data structure that will be detailed as follows and is based on graph partitioning, as suggested by Chevaleyre et al. [8] and applied by Pippin et al. [5] and Lauri et al. [17], among others [11], [14].

Each agent is represented as a cycle which covers all vertices of a partition of the environment. Thus, a solution to the TMAP is a set of r cycles covering r partitions of the graph, where r is the number of agents. In addition to this, each cycle has a node marked as a *center* node. The center, however, does not need to be necessarily positioned at the middle of the cycle. The reason behind this design choice is to guide the crossover operator. If two agents, in two different individuals, have the same center, then they must be patrolling somewhat similar regions of the graph, and the recombination process can exchange these agents between the two individuals. Figure 1 shows a graphical representation of such a solution. It is important to note that these are not disjoint partitions, i.e. there may be intersections between two different partitions.

In the following subsections, each of the three operators (creation, mutation and crossover) will be detailed.

A. Creating Individuals

The process of creating a new individual was divided into three parts. Firstly, it is necessary to find r vertices to act as centers, where r is the number of agents. The second part is to select the vertices that will compose each agent's partition, which do not need to be disjoint. The center associated with a particular agent must be in its partition. Lastly, a cycle going through every vertex is calculated for each partition.

The result, after all these tasks are completed, is a solution to the TMAP with r agents, where each agent will possess a vertex as a center and a cycle to walk during the patrol. The routine composed by these three tasks should be called by the EA every time there is a need to create a completely new individual. For each of these three tasks, a heuristic was developed and will be explained in the following paragraphs.

For the task of selecting the centers, the developed heuristic tries to find a set of r vertices which are furthest apart from each other. The objective is to initialize individuals in a more intelligent way, to increase the chances of creating individuals with high fitness. It is important, however, that the strategy retains some level of randomness. Luke [21] alerts to the dangers of using a heuristic to guide the creation of individuals: it may seem to be improving the chance of generating good individuals, but could get the process stuck in a local minimum.

Therefore, this paper proposes an operator named Approximated Maximum Distance Centering (AMDC). In the beginning, it selects r centers randomly. Then, it calculates the sum of distances between the selected vertices. Next, it iterates until a given condition is met (such as a pre-defined number of iterations) and, at each iteration, a center would be chosen randomly. Then, every vertex that is adjacent to this center is tested as a hypothetical new center: if this change increases the sum of distances between the centers, the swap is confirmed; otherwise, the swap is reversed. The pseudo-code 1 illustrates this behaviour.

Pseudocode 1 *Approximated Maximum Distance Centering*

```
1:  procedure AMDC($G(V, E)$ , $r$)
      ▷ $G(V, E)$ is the graph and $r$ is the number of agents
      ▷ $N_G(v)$ is the neighbourhood of the vertex $v$ in graph $G$
2:      $C \leftarrow$ random subset of $V$ containing $r$ vertices
3:      $d \leftarrow$ sum of distances between vertices in $C$
4:      repeat
5:          $c \leftarrow$ randomly selected vertex in $C$
6:          for $v \in N_G(c)$ where $v \notin C$ do
7:              Replace $c$ for $v$ in $C$
8:              $d' \leftarrow$ sum of distances between vertices in $C$
9:              if $d > d'$ then
10:                 Replace $v$ for $c$ in $C$
11:             else
12:                 $d \leftarrow d'$
13:     until there is no time left
14:     Return $C$
```

For the task of creating partitions, the developed heuristic is represented in pseudo-code 2 and is described as follows. The first stage is to start a list, P, indexed by the vertices of the graph, where $P[v]$ is the center associated with v. For each center, c, a list of all the vertices in the graph ordered by their distance to c is kept in $L(c)$. Then, it will iterate its centers cyclically (the two outer loops from lines 8 to 15) until every vertex is associated with a center. At each iteration, for each center c, it will take the first element v from the ordered list $L(c)$ which has not yet been associated with a center, then associates v with the partition of c. The final result is that every vertex will be linked to a center to which it is close. Finally, for every vertex, the vertices in the shortest path from it to the associated center is added to the partition to guarantee it is connected. This strategy was named Heuristic Graph Partitioning (HGP).

Pseudocode 2 *Heuristic Graph Partitioning*

```
1:  procedure HGP($G(V, E)$, $C$)
      ▷ $G$ is the graph. $C$ is the output from AMDC
      ▷ $L(v)$ and $Pr(v)$ are key-value pairs structures
      ▷ $SP_G$ returns the shortest path between two vertices
      ▷ $POLL()$ removes the first element of a list
2:      $P \leftarrow \{\}$
3:      for $v \in V \mid v \in C$ do $P[v] \leftarrow v$
4:      for $v \in V \mid v \notin C$ do $P[v] \leftarrow -1$
5:      for $c \in C$ do
6:          $L(c) \leftarrow V - \{c\}$, ordered by their distance to $c$
7:          $Pr(c) \leftarrow \{\}$
8:      repeat
9:          for $c \in C$ do
10:             while $L(c) \neq \{\}$ do
11:                 $v \leftarrow$ POLL($L(c)$)
12:                 if $P[v] = -1$ then
13:                     $P[n] \leftarrow c$
14:                     break
15:     until $-1 \notin P$                ▷ Until every node is in a partition
16:     for $v \in V$ do
17:         $Pr(P[v]) \leftarrow Pr(P[v]) \cup SP_G(P[v], v)$
18:     Return Pr
```

Finally, after having selected the centers and defined the partitions, it is necessary to build a cycle that allows the agent to walk through all the vertices of its assigned partition indefinitely. It is possible to argue that the optimal cycle would be Hamiltonian and with minimum distance covered. Therefore, the operator for calculating this cycle was built based on heuristics for the TSP [22]. The heuristic used was the Nearest Neighbour Method [23], thus, the operator was named Nearest Neighbour Cycle Building (NNCB). A few adaptations were necessary because the TSP is defined only over complete graphs: whenever those heuristics referred to an edge between two vertices, the implemented version used the shortest path between the same two vertices, thus removing the requirement of completeness.

By combining AMDC, HGP and NNCB, it is possible to build an entirely new individual. To execute an EA, it is still necessary to define operators for mutation and recombination, which will be detailed next.

B. Mutation

The mutation operator developed for this research is called *Add-Sub Mutation*, because there is a 50% chance that the operator will add or subtract a randomly selected vertex from an agent's cycle. When adding a vertex v, the operation will search for another vertex v' already in the agent's cycle with the shortest distance to v. Then, it will include, in the agent's cycle, a path from v to v' and back to v. When removing a vertex, say v_i, the operator will remove the vertex and add the shortest path from the v_{i-1} and v_{i+1}. However, if $v_{i-1} = v_{i+1}$, then the operator will just remove v_i. The pseudo-code 3 illustrates this operator.

This operator uses a heuristic at the end of the process, known as k-change [24] or k-opt [23]. The k-change is a local search heuristic to the TSP. It tries to find the lowest cost cycle by applying small changes to an original cycle. This change consists in the substitution of k edges in the cycle. The most commonly used values for k are 2, 3 and 4 [23]. Since the objective of a mutation operator is to perform a slight change in an individual, this research used $k = 2$.

Pseudocode 3 *Add Sub Mutation*

```
1:  procedure ADD-SUB($G(V,E)$, $r$, $C$, $Pr$, $Cy$)
        ▷ $G$ is the graph, $r$ is the number of agents, $C$ is the list of
        centers. $Pr$ are the partitions and $Cy$ are the cycles, both are
        key-value pairs.
2:      $x \leftarrow$ a random number from 0 to 1
3:      $c \leftarrow$ a random element from $C$
4:      $v \leftarrow$ a random element from $V$ such that $v \notin C$
5:      if $x \geq 0.5$ then
6:          add $v$ to $Cy(c)$ ensuring connectivity
7:      else
8:          remove $v$ from $Cy(c)$ ensuring connectivity
9:      update $Pr(c)$
10:     repeat
11:         $Cy' \leftarrow$ 2-OPT($G[Cy]$)
12:         if COST($Cy'$) < COST($Cy$) then
13:             $Cy \leftarrow Cy'$
14:     until there is no more time
```

C. Crossover

Finally, the last necessary tool to build an evolutionary approach to the TMAP is the crossover operator. This is responsible for taking two already existing individuals, A and B, and combining their characteristics into new child individuals.

Since all individuals are solutions to the same instance of the TMAP, they possess the same number of agents. The operator developed for this research exchanges an agent from a selected individual for an agent of another individual in the population. It starts by randomly selecting an index, i, in the list of agents. Then, it verifies if the agents in position i have the same center in both solutions. In case they do, the operator will exchange one for the other. Otherwise, the operator will try to find an agent in individual B whose partition contains the center of the i^{th} agent in individual A. If an agent is found, the operator exchanges the j^{th} agent in B for the i^{th} agent in A. Otherwise, the operator will just exchange the individuals at position i in both individuals. Pseudo-code 4 helps visualize this behavior.

Pseudocode 4 *Simple Random Crossover*

```
1:  procedure SPC($G$, $r$, $A$, $B$)
        ▷ $G$ is the graph, $r$ is the number of agents.
        ▷ $A$ and $B$ are solutions. Each solution $x$ has 3 components: $C_x$,
        the list of centers; $Pr_x$ the partitions; and $Cy_x$ the cycles. Both
        $Pr_x$ and $Cy_x$ are key-value pairs.
2:      $i \leftarrow$ randomly chosen number in the range $1...r$
3:      $a \leftarrow$ a copy of $A$
4:      $b \leftarrow$ a copy of $B$
5:      if $C_a[i] = C_b[i]$ then
6:          exchange $C_a[i]$ and $C_b[i]$
7:          exchange $Pr_a(i)$ and $Pr_b(i)$
8:          exchange $Cy_a(i)$ and $Cy_b(i)$
9:      else
10:         if $\exists j \in 1...r$, s.t. $C_a[i] \in Pr_b(j)$ then
11:             exchange $C_a[i]$ and $C_b[j]$
12:             exchange $Pr_a(i)$ and $Pr_b(j)$
13:             exchange $Cy_a(i)$ and $Cy_b(j)$
14:         else
15:             exchange $C_a[i]$ and $C_b[i]$
16:             exchange $Pr_a(i)$ and $Pr_b(i)$
17:             exchange $Cy_a(i)$ and $Cy_b(i)$
18:     Return individuals $a$ and $b$
```

D. New TMAP solvers

We proposed a new representation of individuals together with evolutionary operators for initialization, mutation and crossover. For *selection*, any domain-independent operator can be used, e.g. *tournament*. All these operators can be combined in many ways to produce EAs to solve TMAP instances given as inputs. (The outputs are the cycles to be followed by each agent). In this research, we have created four specific such TMAP solvers by using these well known EAs: the ES $(\mu+\lambda)$, the ES (μ, λ), the GA and the steady-state GA [21]. In the next section we evaluate these algorithms in experiments.

IV. Experiments

This section aims at assessing the effectiveness of our approach by comparing them to other solutions from the literature. With this goal, we did experiments in a computer simulator, evaluating each solution with the Quadratic Mean of Intervals (QMI) metric. As shown in [1], this metric reflects a balance between the average frequency and the standard deviation of the intervals, so a solution that optimizes it is expected to make frequent visits in regular intervals.

A preliminary set of experiments (in the simulator) was designed and executed with the objective of finding the Evolutionary Algorithms (EAs) that performed better and also to tune some of their parameters. Due to lack of space, we only present the findings of these experiments. First, from the four EAs proposed, the two Genetic Algorithms (GAs) variants performed better than the Evolutionary Strategies (ES's) ones. Another finding is about the stopping criteria in the proposed evolutionary TMAP solvers. We based our stopping criteria on the overall number of fitness evaluations

Fig. 2: Maps used in the experiments. Sources: [1], [10]

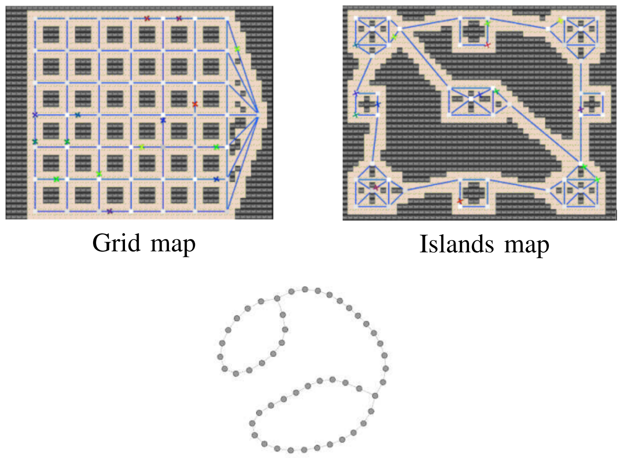


TABLE I: Ranks per size of the agent society

Algorithms	Number of agents			
	1 ag.	5 ag.	10 ag.	15 ag.
GA	1^{st}	2^{nd}	4^{th}	5^{th}
Steady state GA	3^{rd}	1^{st}	5^{th}	3^{rd}
grav(Edge,Ar,1,max)	1^{st}	6^{th}	6^{th}	5^{th}
grav(Node,Ar,1,max)	4^{th}	5^{th}	2^{nd}	2^{nd}
grav(Node,Ar,2,sum)	6^{th}	3^{rd}	1^{st}	1^{st}
Single cycle	4^{th}	3^{rd}	3^{rd}	3^{rd}

done by the algorithms, because fitness evaluation is the most time-consuming procedure, and we found out that stopping after $30,000$ fitness evaluations is enough to produce good outputs. The following parameters were also set based on these experiments: (i) both GAs were configured to use a 5-tournament selection operator; (ii) the population for the GA was set to 96 individuals; (iii) the population for the steady-state GA was set to 110 individuals;

Finally, the comparison experiments were conducted to compare the proposed GA and steady-state GA with Timed Multi-Agent Patrolling (TMAP) these solutions selected from literature for their good comparative results: the *Single Cycle* strategy [8] [10] and the three best gravitational strategies [15]: *grav(Node,Ar,2,sum)*, *grav(Node,Ar,1,max)* and *grav(Edge,Ar,1,max)*. For each solution, we run simulations in different maps (graphs) with different sizes of the societies of agents. We have chosen three maps from literature [1] [10] which are very different in topology, illustrated in figure 2. Each approach was simulated in each map with 1, 5, 10 and 15 agents. Before presenting the results, we first remark that the experiments produced a great mass of data that could be analyzed in some other ways, but for limitations of space we had to focus on the essential.

The results of the experiments display a non-trivial dependence on these two dimensions: the maps and the sizes of the society (i.e. number of agents). Therefore, we performed two separate comparisons by isolating one of these dimensions while collapsing the other. To compare the performance of the solutions in different sizes of society collapsing the maps dimension, we ranked the approaches from one (1^{st} place) to six (6^{th}) using the raw metric values. For each fixed number of agents, separate rankings were calculated (e.g. steady state GA was 2^{nd} in Grid map for 5 agents) then these rankings were averaged over all maps (e.g. steady state GA average position was 2.3 in all maps). Table I shows the final ranking based on the averages rankings just explained.

TABLE II: Ranks per map

Algorithms	Problems		
	$Cycles$	$Grid$	$Islands$
GA	5^{th}	2^{nd}	2^{nd}
Steady state GA	4^{th}	3^{rd}	4^{th}
grav(Edge,Ar,1,max)	6^{th}	6^{th}	6^{th}
grav(Node,Ar,1,max)	3^{rd}	1^{st}	3^{rd}
grav(Node,Ar,2,sum)	2^{nd}	4^{th}	1^{st}
Single cycle	1^{st}	5^{th}	5^{th}

The second comparison of the strategies ranked them in the different maps regardless of the society sizes. To collapse the dimension of the society sizes, a technique proposed in [1] was utilized. The raw values of the QMIs were all normalized by multiplying each of them by the number of agents. Then, these values were added together, giving a single value per map (for each solution) which we call Normalized Quadratic Mean of Intervals (NQMI). The following equation summarizes the idea: $NQMI = \sum QMI_S \times |S|$. Where S is the society of agents, $|S|$ is its size, and QMI_S is the QMI obtained by society S. Using the NQMI of the approaches, we ranked them in each map. The result is presented in Table II.

A third and final comparison can be done by collapsing both the maps and society sizes dimensions, producing a single ranking of the solutions. One way to do this is by calculating the average ranking of each strategy from the rankings by maps. The table III shows the final results.

TABLE III: Average Rankings over all maps and society sizes

Approach	Average Ranking
$grav(Node,\ Ar,\ 2,\ sum)$	2,333
$grav(Node,\ Ar,\ 1,\ max)$	2,333
GA	3,000
Steady-state GA	3,667
Single Cycle	3,667
$grav(Edge,\ Ar,\ 1,\ max)$	6,000

Now we give some comments about the results above. We first note that, in the results per society size, GA was the best performing strategy for a single agent, alongside the *grav(Edge,Ar,1,max)*. Indeed, GA and steady state GA performed very close to the best performing strategies for societies of sizes 1 and 5. For 10 and 15 agents, they had average performance but never placed last. We also highlight a stalemate between the GA variations: the standard GA had a better result for 1 and 10 agents, whereas the steady-state GA performed better for 5 and 15 agents. Analysing per topology, we highlight that the GA proposed in this work was the second best in both the grid and islands maps. The steady-state GA performed, on average, only slightly worse than GA, but surpassed it on one map (Cycles-Corridor), which, again, shows that there is no dominance of one of them over the other.

We also remark that the overall comparison puts the two algorithms proposed in this work in 3rd and 4th places. This is an important result that shows that the proposed evolutionary approaches had competitive performance when compared to four of the best solutions from literature.

V. Conclusion

This paper was motivated by the fact that, although the Timed Multi-Agent Patrolling (TMAP) is essentially an optimization problem, evolutionary algorithms have not been used directly to find solutions for it. Therefore, we presented a new representation along with new operators that can be combined to form evolutionary heuristics solvers for the TMAP.

It was also demonstrated, through computer simulations, that the proposed approaches are efficient in outputting solutions (i.e. cyclical paths for each agent) for the problem. The solutions produced have a performance in Quadratic Mean of Intervals (QMI) metric comparable to some important TMAP strategies reported in the literature.

Considering that the evolutionary operators proposed in this paper work by partitioning the environment between the agents and by optimizing the internal paths followed in each partition, a more general conclusion to this research is that *specializing agents in subregions of the graph yield good results to the TMAP*. Results presented in other works (e.g. [13]) corroborates with this conclusion.

This work can be extended in many ways in future works. It would be interesting to compare these evolutionary approaches to a larger number of other strategies from the literature. We also think it is important to test them in more graph topologies. With the data structure used in this paper to represent TMAP solutions, it is possible to develop other operators to replace the ones presented here. For instance, instead of the Nearest Neighbour heuristic, any other heuristic for the Travelling Salesman Problem (TSP) can be used as a cycle building operator. Also, it would be intriguing to investigate the possibility of designing completely random operators for creating individuals as they are commonly employed in other applications of EAs.

References

[1] P. A. Sampaio, "Patrulha temporal taxonomia, métricas e novas soluções," Ph.D. dissertation, Universidade Federal de Pernambuco, Recife, PE, Brazil, March 2013.

[2] E. Hernández, J. d. Cerro, and A. Barrientos, "Game theory models for multi-robot patrolling of infrastructures," *International Journal of Advanced Robotic Systems*, vol. 10, pp. 181–189, March 2013.

[3] R. Alberton, R. Carli, A. Cenedese, and L. Schenato, "Multi-agent perimeter patrolling subject to mobility constraints," *American Control Conference (ACC), 2012*, pp. 4498–4503, June 2012.

[4] C. Poulet, V. Corruble, and A. Seghrouchni, "Working as a team: Using social criteria in the timed patrolling problem," *Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on*, vol. 1, pp. 933–938, Nov 2012a.

[5] C. Pippin, H. Christensen, and L. Weiss, "Performance based task assignment in multi-robot patrolling," *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pp. 70–76, 2013. [Online]. Available: http://doi.acm.org/10.1145/2480362.2480378

[6] N. Agmon, S. Kraus, and G. A. Kaminka, "Multi-robot perimeter patrol in adversarial settings," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 2339–2345.

[7] J.-S. Marier, C. Besse, and B. Chaib-Draa, "Solving the continuous time multiagent patrol problem." in *ICRA*. Citeseer, 2010, pp. 941–946.

[8] Y. Chevaleyre, F. Sempe, and G. Ramalho, "A theoretical analysis of multi-agent patrolling strategies," *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3*, pp. 1524–1525, 2004. [Online]. Available: http://dx.doi.org/10.1109/AAMAS.2004.34

[9] A. Machado, G. Ramalho, J.-D. Zucker, and A. Drogoul, "Multi-agent patrolling: An empirical analysis of alternative architectures," *Proceedings of the 3rd International Conference on Multi-agent-based Simulation II*, pp. 155–170, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=1765317.1765332

[10] A. Almeida, G. Ramalho, H. Santana, P. Tedesco, T. Menezes, V. Corruble, and Y. Chevaleyre, "Recent advances on multi-agent patrolling," *Advances in Artificial Intelligence – SBIA 2004*, vol. 3171, pp. 474–483, 2004. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-28645-5_48

[11] Y. Elor and A. M. Bruckstein, "Autonomous multi-agent cycle based patrolling," *Proceedings of the 7th International Conference on Swarm Intelligence*, pp. 119–130, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1884958.1884970

[12] S. Koenig and Y. Liu, "Terrain coverage with ant robots: A simulation study," *Proceedings of the Fifth International Conference on Autonomous Agents*, pp. 600–607, 2001. [Online]. Available: http://doi.acm.org/10.1145/375735.376463

[13] H. Santana, G. Ramalho, V. Corruble, and B. Ratitch, "Multi-agent patrolling with reinforcement learning," *Autonomous Agents and Multi-agent Systems, 2004. AAMAS 2004. Proceedings of the Third International Joint Conference on*, pp. 1122–1129, July 2004.

[14] D. Portugal and R. P. Rocha, "Cooperative multi-robot patrol with bayesian learning," *Autonomous Robots*, pp. 1–25, 2015.

[15] P. A. Sampaio, G. Ramalho, and P. Tedesco, "The gravitational strategy for the timed patrolling," in *Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on*, vol. 1, Oct 2010, pp. 113–120.

[16] F. Lauri and A. Koukam, "Hybrid aco/ea algorithms applied to the multi-agent patrolling problem," *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pp. 250–257, July 2014.

[17] ——, "A two-step evolutionary and aco approach for solving the multi-agent patrolling problem," *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pp. 861–868, June 2008.

[18] K. H. Rosen, *Discrete Mathematics and Its Applications*, 6th ed. McGraw-Hill Higher Education, 2006.

[19] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," Carnegie-Mellon University, Pittsburgh, PA, USA, Tech. Rep., 1976.

[20] S. Doi, "Proposal and evaluation of a pheromone-based algorithm for the patrolling problem in dynamic environments," *Swarm Intelligence (SIS), 2013 IEEE Symposium on*, pp. 48–55, April 2013.

[21] S. Luke, *Essentials of Metaheuristics*, 2nd ed. Lulu, 2013, available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

[22] G. Gutin and A. Punnen, *The Traveling Salesman Problem and Its Variations*, ser. Combinatorial Optimization. Springer US, 2006. [Online]. Available: https://books.google.com.br/books?id=JBK_BAAAQBAJ

[23] C. Nilsson, "Heuristics for the traveling salesman problem," Tech. Report, Linköping University, Sweden, Tech. Rep., 2003.

[24] D. Marx, “Searching the k-change neighborhood for tsp is w[1]-hard,” *Oper. Res. Lett.*, vol. 36, no. 1, pp. 31–36, Jan. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.orl.2007.02.008