IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 24, 2017, accepted July 20, 2017, date of publication August 29, 2017, date of current version September 19, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2739641

Heuristics for the Multi-Robot Worst-Case Pursuit-Evasion Problem

LÍVIA GREGORIN[1], SIDNEY N. GIVIGI[2], (Senior Member, IEEE), EDUARDO FREIRE[1], (Member, IEEE), ELYSON CARVALHO[1], AND LUCAS MOLINA[1]

[1]Department of Electrical Engineering, Universidade Federal de Sergipe, Aracaju 49100-000, Brazil
[2]Department of Electrical and Computer Engineering, Royal Military College of Canada, Kingston, ON K7K 7B4, Canada

Corresponding author: Sidney N. Givigi (sidney.givigi@rmc.ca)

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada through the NSERC Discovery Grants Program.

ABSTRACT There is a growing demand for the use of robots to assist humans in their tasks, especially those involving risks, such as search and rescue. For this reason, coordination among several robots has been a common option, and one of the ways to study and model these applications involves the problem of pursuit evasion. This paper extends the results presented earlier on the use of an evolutionary robotics approach to solve the worst case pursuit-evasion problem, in which evaders are considered arbitrarily fast and omniscient, while pursuers have limited sensing and communication capabilities, with no prior knowledge regarding environments, which are treated as discrete and can be multiply connected. First, a formulation based on random walk is offered. Then, the concept is extended to include a decentralized multi-robot control system based on a finite-state machine with state-action mapping defined by means of a genetic algorithm. Results show that the proposed system is able to decontaminate several types of maps, but does not generalize to all initial conditions, due to the incompleteness in the automaton mapping. Therefore, a complementary approach is presented in which random walk is used alternatively with the evolved automaton, indicating random actions in cases of states not sufficiently visited during evolution. In addition, a comparative analysis of the evolutionary approach and the random walk formulation is also carried out.

INDEX TERMS Evolutionary robotics, pursuit-evasion, clearing problem.

I. INTRODUCTION

The study of systems with multiple robots has intensified in the last two decades, mainly due to the amount of human assistance tasks whose execution can be facilitated by the coordinated work of several agents, such as search and rescue missions, demining, patrolling and exploration. One of the ways to model such applications is based on so-called pursuit-evasion games, in which different forms of competitive and cooperative behaviors can be studied depending on choices about pursuer and evader's dynamics, capture conditions and visibility constraints.

Among several taxonomies and models, the *worst-case adversarial search* [1], also called *the clearing problem* [2], stands out as a classical case that has the advantage of being applicable to one or several evaders, regardless of the behavior they present. This is because the scenario, firstly proposed by Parsons [3], is characterized by omniscient and arbitrarily small evaders, which can move at unbounded speed, but continuously, and can be treated as a noxious gas that has previously contaminated the environment, whereas the pursuers have several restrictions.

Over the years, worst case scenario has been studied under different aspects and, more recently, the interests have turned to the use of searcher agents with real restrictions in respect of movement and visibility. Since real robot applications with limited sensing with respect to the size of environments are desired, the clearing of any locations that are not limited to simply connected corridors requires the coordination of multiple robots to accomplish the task, otherwise the evader would always be able to recontaminate the environment.

In this sense, Kolling and Carpin [4] were the first to present a solution for coordination of multiple searchers in which no prior knowledge of environments is considered. It is a distributed approach, called Line-Clear, in which the pursuers are organized in multiple sweep lines and move according to their role in the formation, following walls or other close robots. The searchers' progress is only allowed if the distance between them does not exceed twice the radius of the sensor or the radius itself between the searcher and an obstacle. A topological map of the discovered environment is constructed and stored, describing possible line movements and their associated costs in terms of number of robots. Coordination only requires local communication between the leaders of different lines of robots whenever they meet. There is no concern about the minimum number of robots, given it is an unknown environment. And also because of this, it is not guaranteed an optimal way to clear the environment or to deal with obstacles. There is no proof of convergence, with theoretical guarantees for environment decontamination, and the algorithm is not well suited to multiply connected environments.

Later, Durham et al. [2] presented a distributed algorithm with theoretical capture guarantee for a sufficient number of robot searchers, without building or storing any map and with no global localization required. The method works maintaining a complete frontier coverage between cleared and contaminated environments, while expanding the cleared region according to a predetermined algorithm. When there are not enough searchers, the algorithm clears as much area as possible, ensuring no recontamination. Their proposed method stores and updates the global frontier based on local intersections and oriented arcs, using a limited amount of memory per robot. Moreover, their work presents real robot experiments, including a case in which one robot failed and the remaining became responsible for completing the task.

For discrete environments, Gonçalves *et al.* [5] presented an approach based on random walk combined with local constraints, without presenting proof of convergence associated to the method. Furthermore, the method requires global communication among the robots.

The present work proposes evolutionary robotics as a tool to generate a decentralized control command based on finite state machines in discrete environments. Preliminary results of the proposed method were published in [6]. The main contributions of this paper are *(i)* a formal representation of the worst-case pursuit-evasion problem based on random walk; *(ii)* the combination of this formulation with an evolutionary robotics approach; *(iii)* the proposition of a complementary *random walk/evolutionary* robotics approach; and *(iv)* a comparative analysis between a pure random walk solution, the evolutionary robotics approach and the combined complementary approach.

This paper is organized as follows. In Section II, we define the multi-robot pursuit-evasion problem. Section III discusses the representation of the problem as a graph and applies a random walk approach to demonstrate that randomly selected actions can guarantee the clearing of an environment. Section IV presents the proposed genetic algorithm methodology for solving the problem defined in Section II. Section V explains the complementary approach proposed. In Section VI, the simulation process and results are presented, as well as the comparative results. Finally, the conclusion is presented in Section VII.

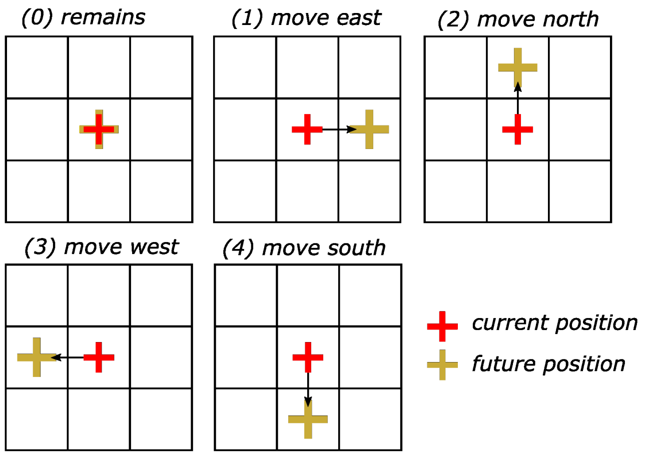

FIGURE 1. Available actions for the robots.

II. PROBLEM STATEMENT

This paper addresses the problem in which a team of N robotic searchers, also called pursuers, has to capture all evaders in an unknown, limited and planar environment that may be multiply connected. From now on, we will use only the term *robots* instead of pursuers or searchers.

The evaders to be captured move continuously at unbounded speed and have full knowledge about all robots' positions and actions. Then, they are treated as contamination, i.e., they are everywhere. Robots, on the other hand, have bounded speed and perfect but limited sensors, within a circumference of radius r_{sensor}. They can communicate with other robots that are at a maximum Euclidean distance of twice the radius r_{sensor}.

Given these restrictions, the capture can only be ensured by coordinated actions among robots, so that they perform a sweep in formation maintaining between them up to the maximum distance of the communication radius. Otherwise, the environment would be immediately recontaminated.

The task is treated in simulation, considering discrete maps represented by regular grids with resolution r. At each time step, robots occupy the center of the cell they are located and can only remain at the same cell or move to one of the nearest four cells in the von Neumann neighborhood (Figure 1). Evaders, otherwise, are assumed to be the size of a cell, in a way they can not hide on imperfections of the discretization process. Due to their features, evaders are not directly implemented, because they are at any not cleared part of the environment.

The house-built simulator, implemented in MATLAB®, where the experiments took place, and the sensing were inspired in the proposal of Gonçalves *et al.* [5]. Adjusts were made in the sensory model, that is treated in a more realistic way as we consider the sensing capabilities to be similar to a real robot. Figure 2 shows the sensory model and robot perceptions in the discrete environment. The robots can only sense the cells that are completely inside their sensing radius, marked in gray in Figure 2(a). The discretization process is

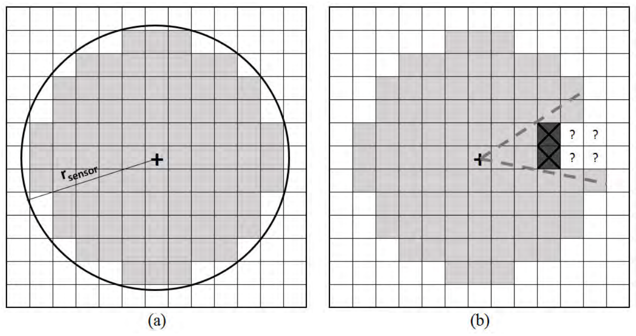

FIGURE 2. (a) The discrete sensory mask and (b) the robot vision.

assumed to has been previously carried out, thereby, whenever any point representing an obstacle is detected, a cell will be marked as occupied, represented by the two cells with a black X in Figure 2(b). Reversely, a cell becomes free whenever any free point is observed in a cell. The cells occluded by an obstacle are unknown and are not marked as free or occupied (case of the four cells behind the obstacle in Figure 2(b)).

Under the conditions of the proposed approach, no memory restriction is considered: robots store the history of visited cells, as well as labeling them about contamination and occupation according to their individual perceptions. Initial conditions to simulation require a formation such that robots fully communicate and establish a decontaminated area.

Based on these definitions, two proposed solutions to the clearing problem are presented. The necessary mathematical notation will be introduced when needed. Section III deals with the random walk approach and demonstrates that the algorithm would result in the whole environment being cleared in finite time.

III. THE RANDOM WALK APPROACH

In their work, Gonçalves *et al.* [5] presents a centralized discrete approach in which random walk is applied to move robot pursuers, while local restrictions avoid recontamination. However, no formal introduction of the problem is offered. Therefore, the mathematical representation and the demonstration that the environment is cleared by using this method are contributions of this paper.

This section is divided as follows. Section III-A introduces the notation necessary to describe the problem as a random walk. Section III-B analyzes the motion restrictions for the robots as a group. Section III-C introduces the problem as a graph. Section III-D describes how even with the restrictions in motion the robots can reach a final state in an environment of known dimensions. Finally, Section III-E demonstrates that starting from a feasible initial state, the final state can be reached with probability one.

A. INITIAL NOTATION

Let us first define a *discrete space* $\mathscr{V}$ that contains all the possible positions of the robots. Furthermore, let us consider,

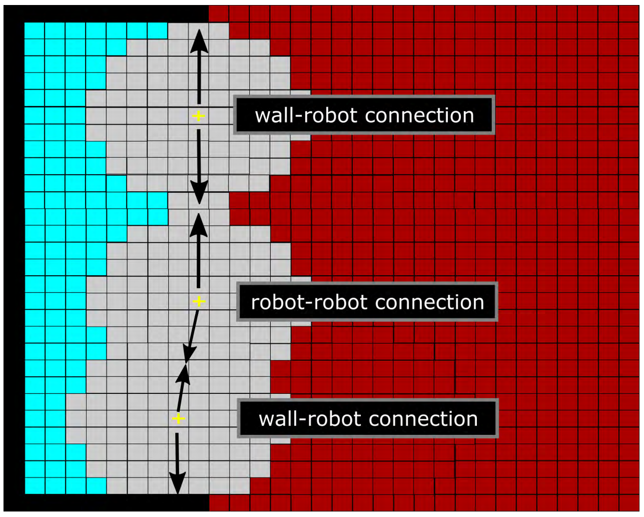

FIGURE 3. Robot neighborhood restrictions.

without loss of generality, that the space is rectangular, i.e., $|X| = \eta$ and $|Y| = \kappa$ with the state $\{x, y\} = {}_{xy}v \in \mathscr{V}$. Note that if we write $i = x * \kappa + y$, one can further define the state ${}_iv$ for $x \in \{1, \cdots, \eta\}$ and $y \in \{1, \cdots, \kappa\}$.

The position of all N robots is an element of the extended discrete state space $\mathscr{V}^N = \mathscr{V} \times \mathscr{V} \times \cdots \times \mathscr{V} \times \mathscr{V}$ where $v = (v_1, v_2, \cdots, v_{N-1}, v_N) \in \mathscr{V}^N$ is the position of all the robots in the environment. Moreover, for any two robots p and q, $v_p \neq v_q$, $\forall p \neq q$. Also, the state $v^t \in \mathscr{V}^N$ may be represented as a tuple of N integers, where the integer in the p^{th} position represents the state where robot p is located at time t.

As discussed in the previous section, each robot has at their disposal at most five actions: (0) remain where it is, (1) move to the east, (2) to the north, (3) to the west or (4) to the south (Figure 1). Some movements are not allowed and therefore their probability is set to zero (see discussion in Section III-B). If we define the action set $\mathscr{A}$ as the possible actions for each robot, the action set for all robots can be represented as $\mathscr{A}^N$. But, since the actions are related to the states, one can introduce the notation $A(v^t) \subset \mathscr{A}^N$ for the actual action set of permissible actions.

Finally, the choice of the actions $a \in A(v^t)$ is mediated by a probability distribution $p \in P(A(v^t))$, where $P(\cdot)$ is a probability distribution related to the motion restrictions to be discussed in the next section.

B. MOTION RESTRICTIONS

The motion restrictions presented in this section are the same introduced in [4]. However, we formally introduce them with the notation used in this paper. Recall that the position of all robots at time step t can be represented as a tuple v^t, that we further constrain to be only in a *feasible* set $\mathscr{F}$, i.e., $v^t \subset \mathscr{F}$. The feasible set $\mathscr{F}$ is defined as the positions of the robots such that from each robot $p \in \{1, \cdots, N\}$ either (1) two other robots or (2) a wall and a robot is sensed in the sensing area

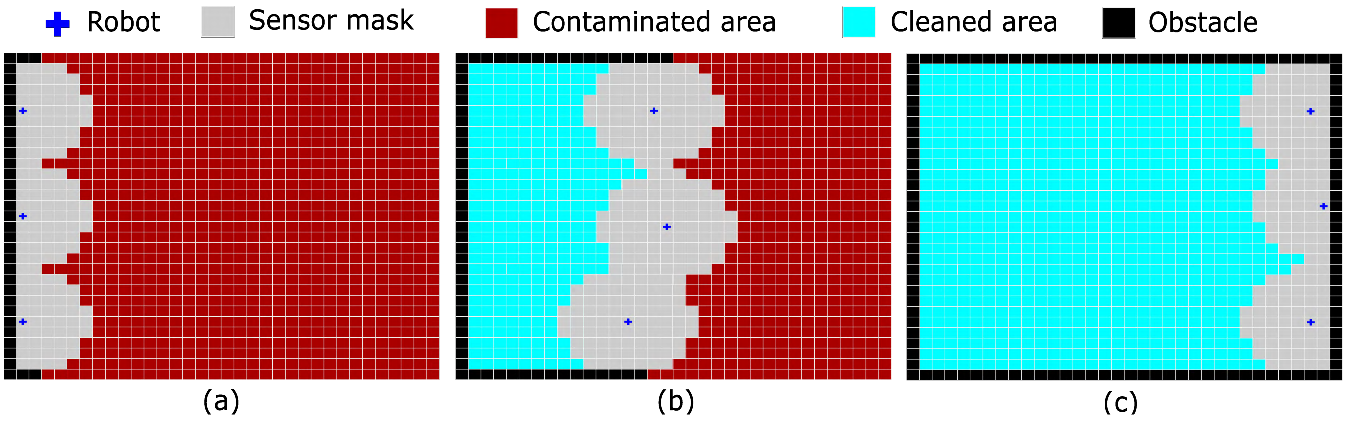

FIGURE 4. Representation of feasible states for the team of robots: (a) possible initial configuration (environment fully contaminated); (b) possible configuration during decontamination (environment partially cleared); (c) possible final configuration (environment fully cleared).

defined by the radius r_{sensor} (Figure 3). Notice that it is also possible for a robot to have two walls as neighbors, but this case will be left to be discussed in Section III-D. Finally, one of the sides of the line formed by the placement of the robots in a state $v^t \in \mathscr{F}$ is completely cleared, i.e., it is guaranteed that there is no evader in that region (Figure 4(a)).

Let us assume without loss of generality that the initial state $v^0 \in \mathscr{F}^0 \subset \mathscr{F}$, where $\mathscr{F}^0$ is the set of all possible initial states for the robots. As shown in Figure 3, this case is only satisfied if $\mathscr{F}^0$ represents all robots located in one of the sides of the environment (in the case shown in the figure, the west). Each robot can then choose their actions so that the new state is v^1. However, motions are restricted by the requirement of $v^1 \in \mathscr{F}$. This, in practice, defines the probability distribution $P(A(v^0)) = P^0 \in \mathscr{P}$ of the action set $A(v^0)$ as only considering the set of actions $a^0 \in A(v^0)$ that will result in $v^1 \in \mathscr{F}$. The probability distribution function is usually uniform, but it can also be non-uniform [7]. Moreover, notice that in order to apply these restrictions, it is required that the motion of each robot be decided (or at least validated) by a central process prior to execution. However, notice that the methods present in Sections IV and V do not assume that a central process is employed. Therefore, in the simulations of Section VI, this requirement is not employed.

By the same rationale, the choice of v^{n+1} is decided by the probability distribution $P(A(v^n)) = P^n$. Therefore,

$$(v^n \in \mathscr{F}) \iff (v^{n+1} \in \mathscr{F})$$

Notice that the restrictions in effect change the probability of transitioning from one state to another and, therefore, the probability distributions $P^0, P^1, \cdots, P^{n-1}, P^n$ are in general different, i.e., $P^0 \neq P^1 \neq \cdots, \neq P^{n-1} \neq P^n$.

In the experiments presented in Section VI, we have chosen v^0 to be all robots set in a line (Figure 4(a)). This guarantees that the necessary number of robots to perform the task is chosen if the environment is unknown but of known dimensions, i.e., we assume that the robots know nothing about the disposition of obstacles in the environment. Figure 4(a) shows the initial condition of the robots, Figure 4(b) shows the robots clearing an area. Notice that the area to the west is cleared whereas the area to the east is contaminated. Finally, Figure 4(c) shows the environment completely cleared.

C. PURSUIT-EVASION REPRESENTED AS A GRAPH

Let us consider the problem introduced above as a tuple

$$G = (\mathscr{V}^N, \mathscr{A}^N, \mathscr{P}(\cdot)) \tag{1}$$

with necessary parameters. Notice that we can alternatively describe the problem by a graph

$$G = (V, E) \tag{2}$$

where V are the vertices, which represent the states where the robots can be found, and E are the edges, which represent the permissible transitions between states. The vertices V are based on the permissible states $\mathscr{V}^N$ and the edges E are based on the allowed actions $\mathscr{A}^N$ and the probability distributions $\mathscr{P}$. Furthermore, for a finite environment of the type we are dealing with in this paper, the number of vertices V may be very large, but are always finite and countable. This is based on the fact that the environment is discrete and finite itself.

This problem represented as a graph has been previously considered in the literature [8]. However, in the previously defined problems, the vertices of the graph represents the possible locations of the robots [9]. In the representation used in this paper, the vertices are all the feasible states of the robots, i.e., $V = \mathscr{V}^N$.

Two features of the problem as represented by a graph must be considered. First that there is a path from the initial state of the robots to the final desired stated (the whole environment cleared). Second, that the final state of the graph is reachable in finite time. These two properties are discussed in the next two sections.

D. EXISTENCE OF A PATH

For the robots to be able to clear the whole environment, it is necessary that there be no contaminated cell. This means that starting at a state $v^0 \in \mathscr{F}^0$, a final state can be reached. Therefore, one needs to first define the set $\mathscr{F}_F \subset \mathscr{F}$ as

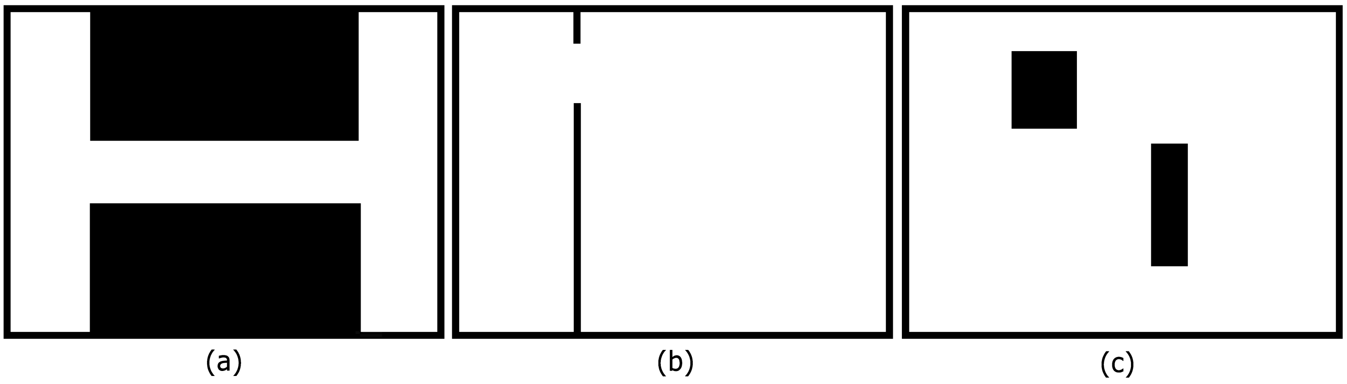

FIGURE 5. **Possible types of obstacles: (a) corridor, (b) door, (c) box.**

the states wherein in the environment is fully cleared. One important question is if the set of all possible feasible states can be built as a connected graph.

By construction, if the dimensions of the environment are known and the initial condition of the robots is $v^0 \in \mathscr{F}^0$ (as described in Section III-B), one can choose the required number of robots necessary to clear the whole environment assuming that the types of obstacles are known, although not their dimensions and locations. The obstacles to be considered are shown in Figure 5: corridors, doors and boxes.

In order to guarantee that there is a path from $v^0 \in \mathscr{F}^0$ to $v^F \in \mathscr{F}_F$, we need to determine if the restrictions discussed in Section III-B are sufficient to cover the whole space without allowing recontamination. This can be done if expansion is possible for one of the connected vertices of the graph to another [2].

Let us consider the three possibilities shown in Figure 5. First, the corridor. If a sufficient number of robots is considered, a state with the robots on the other side of the corridor is always feasible [2]. In the case when robots in front of the formation cannot move because they would allow recontamination of the environment, by the assumptions that there is a sufficient number of robots, there is a probability that the robots stop and wait for more robots to come to the front. Some algorithms can result in this behavior [2] and the probability that this will happen is one [2].

Second, a door can be seen as a special case of a corridor (a door is a corridor with length equal to the resolution of the map). Therefore, if the passage through a corridor results in feasible states, so does the passage through a door.

Finally, a box can be seen as a bifurcation. Given that we have enough robots, each side of a box can also been seen as a corridor and a feasible state can be found in the same way as it can be found in the previous cases. Therefore, a path of feasible states can be established from the initial state $v^0 \in \mathscr{F}^0$ to a final state $v^F \in \mathscr{F}_F$. Observe that the analysis done in this section is not valid for all types of obstacles, especially if the obstacle is non-convex. If more general obstacles are present, a more in depth analysis needs to be carried out.

In this section we established that there is a path from a initial state to a final state. However, it does not guarantee that the final state can be reached in finite time. This question will be tackled in the next Section.

E. REACHABILITY

Considering the problem described in Section II and represented by (1), an environment can be cleared if a final state of the graph (2) is reachable in finite time.

Let us start by defining reachability. For a final state $v^F \in \mathscr{F}_F$ to be *reachable*, one only needs to demonstrate that the *hitting time* $H(v^0, v^F) < \infty, \forall v^0 \in \mathscr{F}^0$.

If the probability distribution for all actions is uniform, one can define hitting time with the following recursive equation [10]

$$H(v^0, v^F) = 1 + \frac{1}{|A(v^0)|} \sum_{v^1 \in \Gamma(v^0)} H(v^1, v^F)$$

where $\Gamma(v^0)$ denotes the set of neighboring vertices of v^0. Intuitively, this equation means that the expected time to reach a state is dependent on the local connections of the graph and the probability distribution of the edges. Notice that if the probability distribution is not uniform, the equation would change, as the probabilities for the transitions would not be the same.

Treatment of the random walk defined in graphs has been done in several works. If the graph does not have loops (transitions to the same states), hitting time can be calculated more easily [11], [12]. In the case in which transitions to the same state are allowed (such as the problem defined in this paper), calculation of hitting time can be more complex, but there are also results that allow the calculation of hitting times if the graph can be reduced to a tree [13]. Unfortunately, this is not the case of the formulation introduced so far.

Some other results have shown that the random walk can be represented by a Markov chain [14], [15]. Moreover, notice that the graph in (2) with the restrictions discussed in Section III-B is irreducible and aperiodic. In this case, there are general methods for the calculation of upper bounds for hitting time [15], [16].

As it should be expected, hitting time is dependent on the number of vertices in the graph $G = (V, E)$ [13]. Moreover, it also depends on the *valence* of the graph, i.e., the number of connections coming into each vertice. Reference [15, Th. 2] in [15] shows that the hitting time in the general case (when no upper bound is enforced for the valence) is bounded to

$$H(v^0, v^F) \leq N(N-1)^2 \tag{3}$$

where N is the number of vertices in the graph. Therefore, the order for the time duration of the random walk is $O(N^3)$ and this has a series of implications for the problem at hand, of which we mention only one that is pertinent for the results discussed in this paper.

The number of vertices in the graph is potentially very large as it depends on the environment (size and disposition of obstacles) and the number of robots. Therefore, the method is not scalable. This is the main motivation for the use of the method discussed in the next section.

IV. THE EVOLUTIONARY ROBOTICS APPROACH

Evolutionary robotics is a field of research that applies artificial evolution, mainly genetic algorithms, to generate morphologies and control systems for autonomous mobile robots [17]. Evolutionary approaches have demonstrated benefits in a wide range of purposes, showing potential to build autonomous systems for problems of many dimensions, also able to deal with new information not predicted during the project, which arouses interest to investigate and expand their applications.

Despite the pursuit-evasion problem has been studied for years in evolutionary robotics (in that context called predator-prey), no research was carried out about the worst case scenario in a cooperative approach. Therefore this technique was chosen to develop this work, and a preliminary analysis was presented in [6].

Although most of the work in evolutionary robotics considers artificial neural networks as the controller paradigm [18], finite state machine has also shown several advantages [19] and was the tool selected to model the control system of robots in order to develop the ability to interact and sweep environments in a formation that does not allow recontamination.

As mentioned in the previous section, the main motivation for the use of the approach discussed in here is the scalability of the random walk solution. Moreover, we are interested in studying how well the evolutionary approach deals with this challenging problem.

Similar to the problem in (1) and (2), the proposed controller is a discrete automaton defined by a finite set of possible states, mapped to a finite set of motion actions, so that each robot is ruled independently by the same automaton model. The main difference is that the approach to be defined shortly is decentralized whereas the random walk approach is centralized (for it requires the check of motion actions by all robots prior to the actual execution of the actions). Also, the number of states (or vertices of the graph) is reduced by a clustering mechanism. Notice that if the same approach was used in the method of Section III, the graph would not necessarily cover the whole discrete space.

The proposed automaton is defined by the quintuple

$$G = (\mathscr{S}^N, \mathscr{A}^N, \delta, {}^0s, \mathscr{F}) \tag{4}$$

where:

- $\mathscr{S}^N$ is the finite set of possible states $s^n \in \mathscr{S}$, with $|\mathscr{S}| = 10$ and $\mathscr{S} = \mathscr{B}^{10}$, with $\mathscr{B} = \{0, 1\}$. Consequently, $s_i^n \in \mathscr{B}, n \in \{1, 2, ..., N\}$, i.e., each element of state vector is binary and defined according to the local perception of the n_{th} robot concerning the environment and the contamination border.
- $\mathscr{A}^N$ is the finite set of actions $\alpha \in \mathscr{A}$ associated to G, with $|\mathscr{A}| = 5$.
- $\delta : \mathscr{S}^N \times \mathscr{A}^N \rightarrow \mathscr{S}^N$ is the transition function. The future state $s^{t+1} \in \mathscr{S}^N$ of the robot group is defined by their current state $s^t \in \mathscr{S}^N$, their action $\alpha^t \in \mathscr{A}^N$ and the environment. Notice that it is not possible to define the target state from the occurrence of an action given a current state of a single robot. This happens because the new state is dependent on the actions of *all* robots and cannot be determined by the action of a single agent.
- ${}^0s$ is the initial state of the system, such that any state can be the initial one.
- $\mathscr{F}$ is the set of final states, in which any state can be the final one (this would be the $\mathscr{F}_F$ in the random walk approach).

Each dimension of the state vector is a binary digit, $s_i^n \in \{0, 1\}$, whose value is defined by agreement to the affirmative description of dimensions, according to Table 1. Then, the value assigned to each dimension will be 0 if the answer to the corresponding statement is false and 1 if it is true.

TABLE 1. State vector description.

Dimension	Description
1	Guarding a frontier
2	The frontier is safe
3	East cleared
4	North cleared
5	West cleared
6	South cleared
7	East free of collision
8	North free of collision
9	West free of collision
10	South free of collision

The first dimension refers to frontier status, as perceived by a robot, which will be *guarding a frontier* in two cases: (i) when the intersections of its sensory mask with obstacles or with other robots' mask has a neighborhood (eight adjacent cells) with a contaminated cell, without allowing recontamination, or (ii) when the robot is in a totally cleared area, but still communicates with the robots in the frontier. Consider the example of Figure 6 as an illustration of the robots' perceptions.

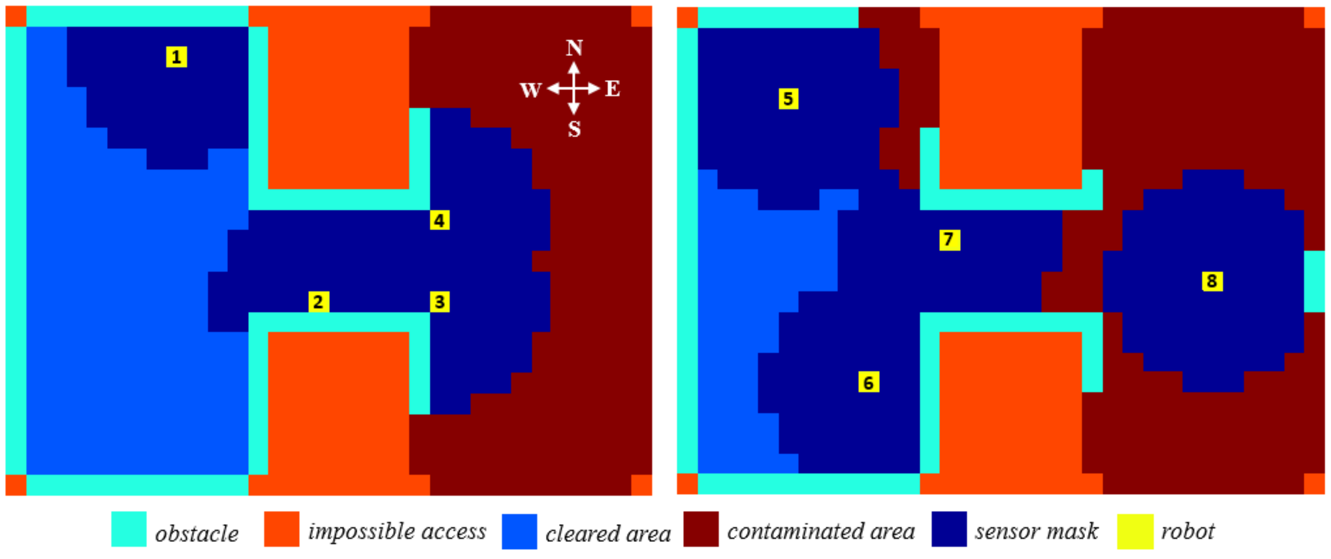

FIGURE 6. Examples of robots' states.

Robots 3 and 4 are an example of the first case, i.e, $s_1^n = 1$ for $n \in \{3, 4\}$, while robot 2 is an example of the second case, i.e, $s_1^2 = 1$. On the other hand, $s_1^n = 0$ for $n \in \{1, 5, 6, 7, 8\}$. Notice that the reasons for s_1^n being 0 are different for each robot. Robot 1 and 8 are not part of the frontier and, therefore, are not guarding it, so $s_1^1 = s_1^8 = 0$. Robots 5 and 7 are actually maintaining a frontier, but depending on their motion, they can break it. For example, if robot 5 moves in the north or west directions, or if robot 7 moves in the south or east directions, they lose the connection and will allow recontamination of the environment. For this reason, $s_1^n = 0$, for $n \in \{5, 6, 7\}$, as robot 6 is not communicating with an structured frontier. Moreover, observe that the state does not identify which frontier is unsafe. This choice was made in order to generalize the motion of the robots and will lead to a conservative behavior, but as results will show it still allows robots to clear different types of maps.

The safe information of the second dimension denotes whether the robot's motion can cause recontamination of previously cleared areas. Also in Figure 6, robots 5, 7 and 8 are unsafe frontier examples, i.e., $s_2^n = 0$ for $n \in \{5, 7, 8\}$. Robot 8 disengaged from the frontier and it is in a contaminated area, so its movement does not interfere with the safety of cleared area. Robot 1, although it cannot communicate with robots in the frontier, is considered aware that it was behind, due to moving history. Once the evolutionary process is interrupted in recontamination case, as will be detailed in the next section, the robot considers it is in a cleared area and then it is free to move anywhere, so its frontier is classified as safe, i.e., $s_2^1 = 1$.

Concerning dimensions 3 to 6, each robot labels the contaminated condition of cells with respect to the four possible motion directions. Notice that the directions do not correspond to the actual cardinal points, but are assigned at

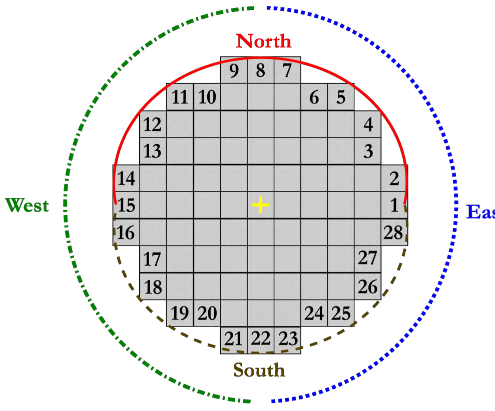

FIGURE 7. Directions for decontamination analysis.

initialization considering the initial position of the robots. This will be further discussed in the next section. The regions of sensory mask are divided according to Figure 7. Cells at the border of the mask always contribute in classification of more than one direction. For example, if cell number 3 indicates contamination, then this information will be assigned to east and also north direction.

Going back to the cases of Figure 6, only searchers 1, 2 and 6 indicate clean cells in all directions, so $s_i^1 = s_i^2 = s_i^6 = 1$ for $i \in \{3, 4, 5, 6\}$. Robot 8 is in a fully contaminated region, while robot 7 has signs of contamination in at least one sensor cell in each direction, then, $s_i^7 = s_i^8 = 0$ for $i \in \{3, 4, 5, 6\}$. Lastly, robots 3, 4 and 5 can define only west as cleared, so for $n \in \{3, 4, 5\}$, $s_5^n = 1$ e $s_i^n = 0$ with $i \in \{3, 4, 6\}$.

The last four dimensions concern the freedom of displacement, in order to avoid collisions. The avoidance of collision is not in the scope of this work and a low level controller is assumed. Therefore, actions that would take the robot to

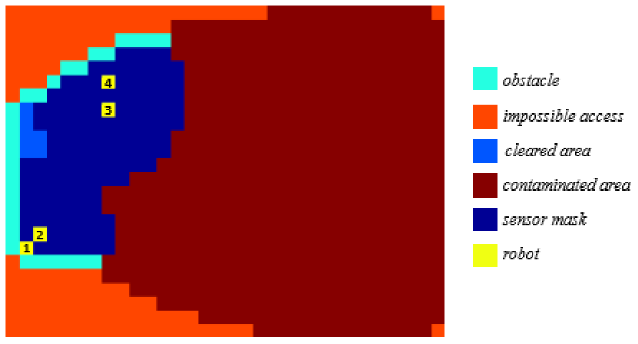

FIGURE 8. Collision Examples.

TABLE 2. Possible robot actions.

Action	Description
0	Robot does not move
1	Move to east
2	Move to north
3	Move to west
4	Move to south

hit obstacles or other robots are prohibited. To identify these cases, risks of collisions are defined differently among robots and between robot and obstacle. A robot has collision risk if (*i*) there is an obstacle in one of the four neighborhood cells or if (*ii*) there is another robot two cells away in any direction of its neighborhood (due to the fact that robots move synchronously). Some examples are presented in Figure 8.

In the figure, robot 1 has collision risk in all directions, i.e., $s_i^1 = 0$ for $i \in \{7, 8, 9, 10\}$. This is because the displacement to west or south causes a collision with the wall, and because the movement to north or east can cause a collision in two cases: if robot 1 moves to the east and robot 2 moves to the south, or if robot 1 moves north and robot 2 moves west. Therefore, dimensions 9 and 10 of robot 2 also show they are not free of collision, i.e., $s_9^2 = s_{10}^2 = 0$.

Robots 3 and 4 also are not free of collisions as there will be a collision if robot 3 moves north and robot 4 moves south. For all other directions the displacement is free, so $s_i^3 = 1$ for $i \in \{7, 9, 10\}$ and $s_i^4 = 1$ for $i \in \{7, 8, 9\}$.

Since each dimension can assume one of two values, we have a total of $2^{10} = 1024$ possible states for the automaton. Transition between two states depends on the motion action $\alpha \in \mathscr{A}$, and the future state depends on the environment and also on other robots' situations, so that it cannot be defined independently from the current state.

At each iteration, a robot's state is individually identified from among 1024 possible states, defining the scenario in which the robot is. Then, each state, marked with an identifier (id), must correspond to a single motion action α. The motion actions α can take integer values from 0 to 4, as described in Table 2.

In this work, evolutionary search was the chosen method to find a mapping to solve the problem. The genotype consists of 1024 motion actions, corresponding to the possible states of the system.

Motion actions with zero value, which correspond to the permanence of the robot in the same cell, are only allowed in cases of collision in all directions. This is done in order to accelerate the evolution of the decontamination behavior, not allowing robots to remain still without need.

Once the population of chromosomes is initialized, each of them is simultaneously incorporated as a control system of all robots present in the simulation. This is due to the fact that, in the proposed approach, the evolution does not operate on the state machine itself, but in the mapping of states and actions.

Unlike traditional optimization, in which the calculation of the cost function is performed directly on the individual's phenotype, evolutionary robotics brings the concept of embodied cognition, i.e., each candidate solution must take control of the robot and the evaluation takes place in emergent behavior from the interaction of the robot with its environment.

In the case of this study, the N simulated robots carry on their control system the same candidate solution, or the same chromosome, and the emergent behavior is evaluated at the end of iterations according to the fitness function

$$fitness = \begin{cases} 0, & Recontamination \\ \dfrac{1}{Contamination}, & Contamination > 0 \\ 1, & Contamination = 0 \end{cases} \quad (5)$$

such that

- *Recontamination* points out that robots opened formation, leading to contamination spreading throughout the environment, in which case the simulation of that chromosome is interrupted. Thus, even though there is no explicit prohibition of recontamination, the penalty is imposed in order for this behavior to disappear in the evolutionary process; and
- *Contamination* corresponds to the number of contaminated cells at the end of simulation, so that the evaluation is greater the lower the number of contaminated cells, and in cases of complete decontamination of the environment, the individual receives maximal fitness.

After evaluation of the population, all individuals are sorted and selected by tournament to be part of the next generation, through the genetic operators crossover and mutation.

The simulation of each individual is interrupted in cases of recontamination, complete environment decontamination or whether the limit number of iterations is reached. But the stopping criterion of the whole evolutionary algorithm is the established number of generations, when it is expected that there have been individuals that satisfy the decontamination task.

V. COMPLEMENTARY PROPOSED APPROACH

In order to investigate the impact of states not properly mapped in the automaton during the evolutionary process,

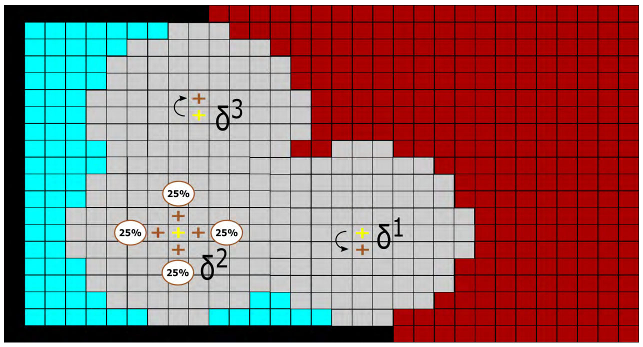

FIGURE 9. Example of moving choice in the complementary approach.

a complementary approach was also considered, inspired by both evolutionary and random walk approaches.

The approach consists in the insertion of random draws to move the robot in cases in which the identified state has been rarely visited during evolution and therefore considered not properly mapped to one of the α actions.

The random walk probability distribution applied in this complementary approach is uniform over the possible motion actions, similar to the one used in [5], but without the restrictions or the possibility of remaining in the same cell.

The random walk replaces the evolved automaton whenever the state identified by a robot does not meet the established cutoff point for the number of visits, a situation in which the mapping obtained in evolution is considered as unreliable. Figure 9 shows an example of how the choice of an automaton and the random walk differs.

In the example of the figure, δ is the transition function that associates a state with an action, that is, it represents the mapping. In the case of δ^1 and δ^3, the individually identified states were considered sufficiently visited, and therefore the mapping of the evolved automaton was accessed and led to the corresponding actions defined in the evolution process of the embedded solution: *move south* and *move north*, respectively.

However, δ^2 represents the situation of a robot that identifies a state whose number of visits does not exceed the established cutoff point, therefore, the transition function is replaced by a draw, in which one of the indicated actions is randomly chosen, under equal probability. Note that staying still at the current cell is not a valid option.

Applying this complementary approach, it is possible to investigate the impact of a different method on the rarely or unvisited states. The approach is used to verify if the definition of new actions tends to result in the improvement of performance and if those states really deserve more attention. In case this is true, a new on-line method could be used to update the selection of actions after the evolutionary algorithm converges.

VI. SIMULATION SETUP & RESULTS

In order to validate the ideas presented in the previous sections, several simulations were carried out. Section VI-A shows results for the random walk approach. These results are included as a baseline for comparison. Then, Section VI-B presents the results of the evolutionary robotics approach. Section VI-C analyses the use of the complementary approach for overcoming issues with the mapping of states to actions. Finally Section VI-D discusses all comparative results for all methods.

A. RANDOM WALK

For the simulations, four robots were considered with r_{sensor} = 150 cells, number sufficient to allow decontamination of all maps shown in Figure 10. The discretized maps are simply-connected, with 24×32 cells, and were chosen in order to represent parts of real environments such as narrow passages, rooms and corridors. For the random walk approach, in the case of Map 4, it was necessary to use five robots to fulfill the task. This was done because the resulting graph is larger and a much higher number of iterations would be necessary. For all other maps and in the simulations of the evolved solutions, only four robots were used.

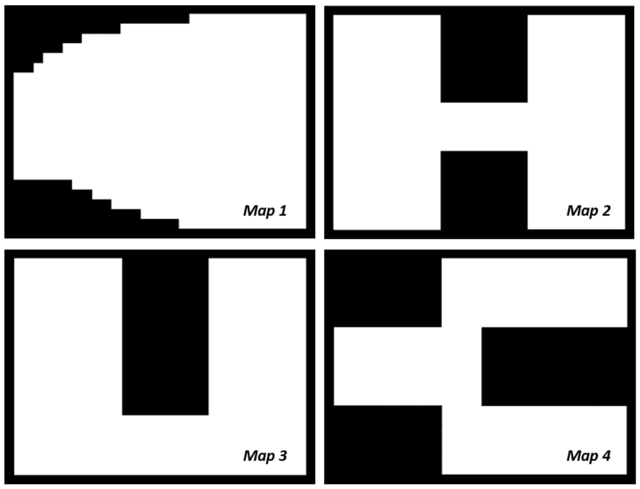

FIGURE 10. Maps used in the simulations.

Two different approaches can be used in order to define the feasible states (or valid vertices) of the graph presented in Section III. The first is to do an extensive search over all the possible states and create a graph with the feasible transitions respecting the restrictions described in Section III-B. However, this would be very time consuming as it depends on the size of the environment and number of robots.

The second is to dynamically apply the restrictions. In this case, the robots are allowed to select their actions independently and a supervisor decides if the motion is valid. Only after the restrictions are validated, the robots are allowed to move. This is also time consuming, but from an implementation point of view is much simpler. Therefore, this was the chosen approach.

The number of attempts to draw until all restrictions imposed on the work are fulfilled is much greater than the number of actual motions and was recorded in the

TABLE 3. Mean and standard deviation of valid iterations and of attempts on Gonçalves' approach.

	Valid iterations	Attempts
Map 1	318 ± 636	27869 ± 90602
Map 2	1422 ± 870	11320 ± 8180
Map 3	888 ± 573	30564 ± 32030
Map 4	7298 ± 17396	123557 ± 245864

experiments performed. Table 3 shows the mean and standard deviation of the valid and tentative iterations for each map.

As shown in the table, several attempts are necessary until a valid state can be reached. This greatly impacts the time required to complete the clearing of the environment. If larger maps and a larger number of robots are considered, the complexity will be significantly increased, which does not occur for the evolved automaton approach, as is shown in the next section.

B. EVOLUTIONARY ROBOTICS

For the evolution stage, the same configuration of the random walk experiments was used. However, few more settings are necessary.

Before simulation, the origin of each map is assigned as the position of the first robot, which is attached to the first cell available in the extreme southwest, considering the motion directions indicated in Figure 6. The origin serves as a reference point to all other robots, whose positions, in turn, are drawn within a narrow vertical range, but always aligned to the west end, so that they always start from a decontamination formation and with full communication.

Figure 11 depicts an individual used during evolution of the genetic algorithm. An individual represents a state machine and is ported to all the robots used in the simulation, i.e., the same controller is used for all robots. At each time step, the states of the robots are individually identified as discussed in Section IV. Each state, marked with an identifier, must have a corresponding action, as shown in Figure 11. The states are then mapped to the five possible actions of Table 2.

In the first generation, the chromosome of each individual in the population to be evolved is randomly initialized with 1024 motion actions, corresponding to the same number of possible states that robots can identify. In future generations, elitism occurs in 10% of the population, which in this case only indicates that such individuals performed better in specific positions that have been tested, but may not be replicated when new startup positions are set.

In order to complete the new population, parents are selected by simple tournament between any two individuals, i.e., two chromosomes of the current population are randomly selected as candidates and the one with the highest fitness is chosen for the group of parents. The process continues with reinsertion until the required number of parents is achieved. Once the assembly is complete, each pair of parents produce two children. If the crossover operator is not applied, the offspring are identical to the parents.

State Id	*State*										*Action*
1	0	0	0	0	0	0	0	0	0	0	$\mathbf{a_1}$
2	0	0	0	0	0	0	0	0	0	1	$\mathbf{a_2}$
3	0	0	0	0	0	0	0	0	1	0	$\mathbf{a_3}$
⋮	⋮										⋮
1022	1	1	1	1	1	1	1	1	0	1	$\mathbf{a_{1022}}$
1023	1	1	1	1	1	1	1	1	1	0	$\mathbf{a_{1023}}$
1024	1	1	1	1	1	1	1	1	1	1	$\mathbf{a_{1024}}$

FIGURE 11. State-Action Mapping.

TABLE 4. Genetic algorithm parameters.

Parameter	Value
Population size	200
Number of generations	200
Elitism	10%
Chromosome length	1024
Crossover rate	50%
Crossover points	20
Mutation rate	20%

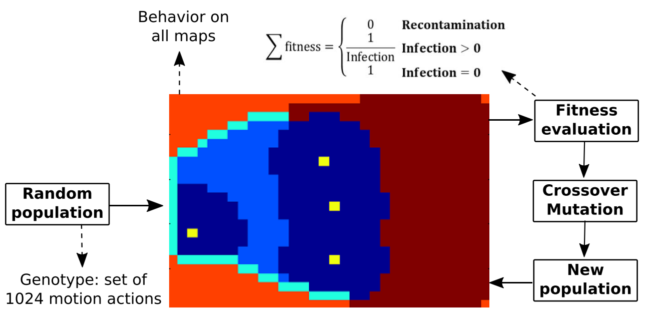
Behavior on all maps
$\sum \text{fitness} = \begin{cases} 0 & \textbf{Recontamination} \\ \frac{1}{\text{Infection}} & \textbf{Infection} > \textbf{0} \\ 1 & \textbf{Infection} = \textbf{0} \end{cases}$
Fitness evaluation
Crossover Mutation
New population
Random population
Genotype: set of 1024 motion actions

FIGURE 12. Evolutionary process flowchart.

The crossover and mutation rates were defined by tests inspired in the literature, and the number of crossover points was chosen to suit the size of the chromosome. Each chromosome has a 20% chance to mutate and, if chosen, new actions are drawn to 20% of the genes (state-action pair), respecting the prohibited actions, i.e, actions that would generate a collision. Table 4 describes all parameters used in the genetic algorithm.

Since not all states will be tested in each one of the maps, to find a solution able to adjust to the largest possible part of the state-action mapping, the evaluation of solutions was defined by the resulting behavior in all four maps presented, within the same evolutionary process. This means that the same candidate solution was tested in each map of Figure 10 and its evaluation was based on the sum of the partial evaluations performed on each map according to (5). If all maps are decontaminated the maximum fitness is achieved, i.e., $fitness = 4$.

The simulation on each map has a maximum duration of 70 time steps and can be interrupted earlier if the robots allow recontamination or if the clearing task is finished. The

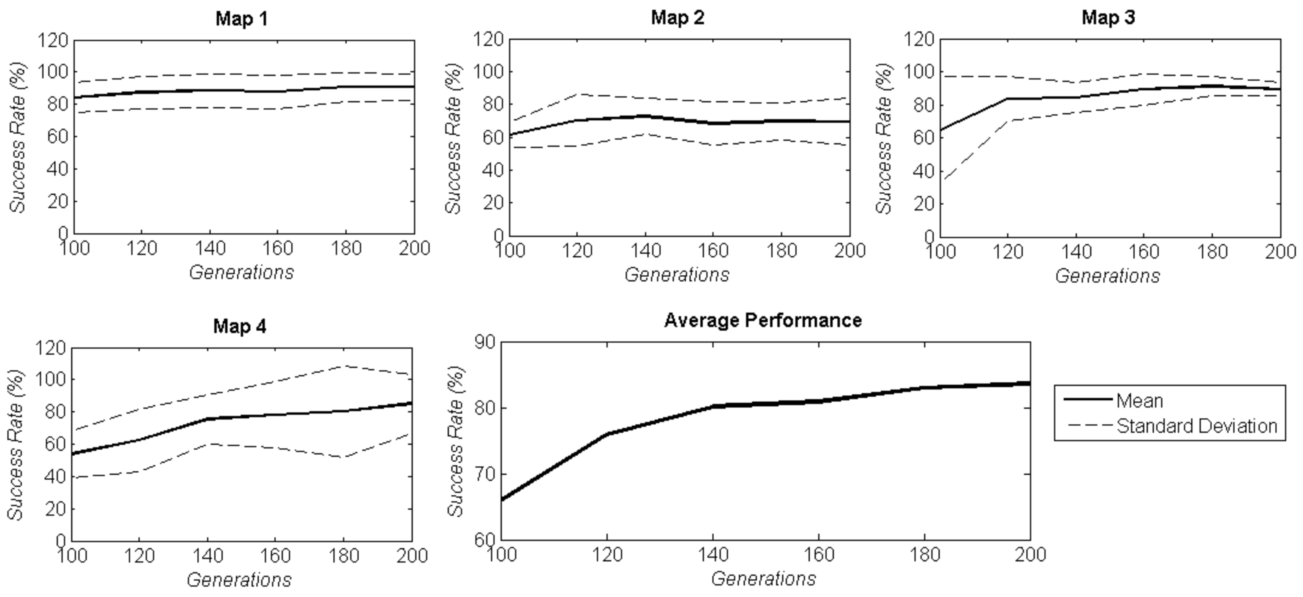

FIGURE 13. Success rate in decontamination task.

TABLE 5. Average success of last generation.

Map	**Evolutionary Experiment**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	99%	91%	98%	79%	86%	100%	92%	100%	100%	99%
2	86%	63%	72%	49%	77%	84%	76%	99%	90%	72%
3	83%	88%	91%	91%	93%	89%	83%	100%	94%	98%
4	95%	54%	93%	83%	100%	99%	99%	100%	98%	93%
Mean	***91%***	***74%***	***89%***	***76%***	***89%***	***93%***	***88%***	***100%***	***95%***	***90%***

complete flowchart of the evolutionary process performed is illustrated in Figure 12.

Ten runs of the same evolutionary process, carried out entirely in the house-built simulator implemented in MATLAB, were performed. To verify the quality of the solutions (i.e, the individuals that represent the state machine), which is related to the success of accomplishing the task, tests were conducted with the top ten individuals of each run, all starting from random initial positions. The initial conditions of tests and of the evolution process are independent, i.e., the positions chosen for the tests do not correspond to positions used during the evolution process.

The solutions were first tested in the same maps used in the evolution process. A set of 12 distinct positions for each map was arbitrarily selected at random and applied in all tests. For each one of the four maps, the solutions were tested every 20 generations, starting from generation 100, with the same set of initial conditions. Thus, the average performance of the decontamination task for the best individuals was obtained for each of the ten runs and for each map. The results are shown in Figure 13, with average and standard deviation of all tests. Table 5 shows the performance of the last generation solutions for each map, broken down by experiment.

Based on these results one can conclude that the solutions found during the evolution are not able to solve the clearing problem for all initial conditions. One of the causes for this issue may be the incomplete adjustment of the state machine, since only about 50% of the possible states are visited during evolution. Due to the fact that the initial conditions for the evolution and tests are different, unvisited states can be reached and actions chosen would be basically random.

It is also possible to notice in Figure 13 a reduction in the average decontamination of maps 2 and 3. These occurrences demonstrate the attempt of the evolutionary process to establish a compromise between successes in all maps, searching for more general behaviors. However, notice that the rate reduction for one map is always accompanied by an increase for another in the same generation, and that the overall average decontamination of maps always increases.

Subsequently, tests were performed on maps different from those used in evolution, in order to continue to check the quality and robustness of the solutions. The two multiply connected maps chosen are shown in Figure 14.

For these tests, new initial position sets with 32 different scenarios were generated for each map. Also, the tests were conducted varying the number of robots available on the maps, from four to nine robots, all of then implementing the

TABLE 6. Decontamination on multiply-connected maps.

Number of robots	Success rate	
	Map 5	Map 6
4	18% ± 12%	59% ± 20%
5	58% ± 17%	89% ± 10%
6	59% ± 20%	89% ± 13%
7	74% ± 20%	94% ± 13%
8	82% ± 17%	94% ± 8%
9	89% ± 14%	96% ± 6%

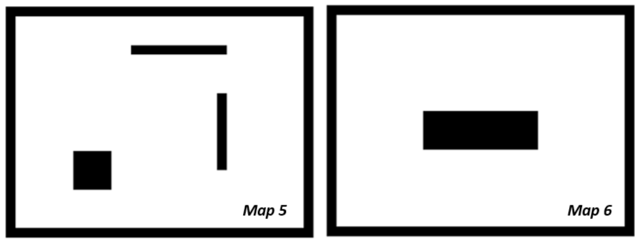

FIGURE 14. Multiply connected maps.

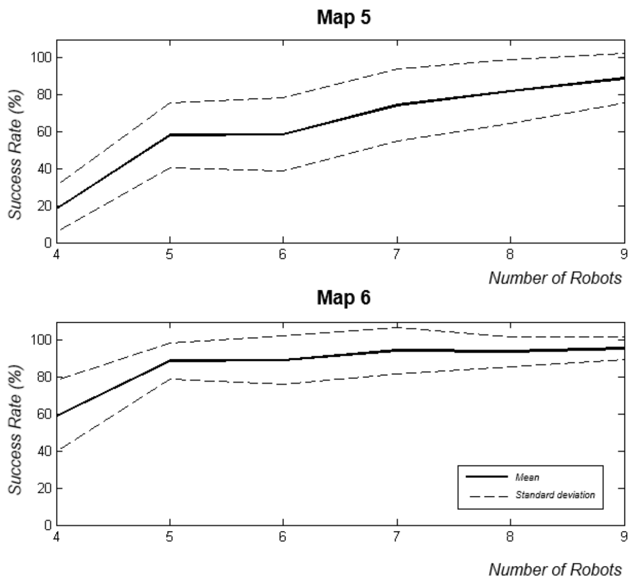

FIGURE 15. Success rate for new maps.

same control system based on the state-action mapping. This test was performed in order to verify the effect of redundancy in the system.

Table 6 presents, for each map and number of robots, the average and standard deviation of decontamination success of the top ten solutions of each one of the ten evolutionary experiments. The results are also graphically shown in Figure 15. One may notice that by increasing the number of robots, the decontamination rate of the maps also increases. Therefore, more redundancy leads to better solutions.

The last tests were performed on larger maps, called here *conjugated maps*. This nomenclature was used because the new maps are a connection of the already presented maps, as shown in Figure 16. Conjugated maps 2 and 4 have 24×64 cells and conjugated maps 1 and 3 have 24×92 and 24×126 cells, respectively.

The best ten individuals of the evolutionary experiments 8 and 9 presented in Table 5 were used in the conjugated maps. Experiments 8 and 9 were chosen because they were the experiments with the best average results. The tests were repeated 30 times for each map.

Considering the size and complexity of this conjugated maps, the tests were done first for a group of nine robots and then with another group of 15 robots. Also, a partial recontamination tolerance of 70 cells was allowed, that is, if part of the robots open the formation and less than 70 cells undergoes recontamination, the simulation is not interrupted such as the previous ones. Recontamination is necessary because the maps have not been used during the evolution phase. Therefore, certain situations will not have been seen before, what will lead to random motion, which, in turn, will cause recontamination.

Such number of 70 cells was stipulated based on the use of larger map and on some observations that, in some occasions, a partial recontamination does not compromise the accomplishment of the task. Furthermore, 70 cells represent only 2 to 5% of the total size of the maps. Simulations of up to 400 iterations were allowed for the conjugate map 3, and 200 for the other maps.

TABLE 7. Results for the conjugated maps (CM).

		CM 1	CM 2	CM 3	CM 4
9 robots	Clearing	5%	32%	1%	8%
	Iterations	176	130	331	151
15 robots	Clearing	26%	54%	7%	12%
	Iterations	162	122	328	166

Table 7 reports the average rate of success of the decontamination task per map for both selected experiments and two different group sizes. The average number of iterations spent for the successful cases is also presented in the table. It was observed that in most cases of failure several robots were trapped in decontaminated sections of the map and these states were not visited during the evolution process.

The recorded failures in the conjugated maps reinforce the hypothesis of incomplete automaton mapping, which does not guarantee visitation to all states, and points to a highly nonlinear cost function, so that small modifications in the genotype can lead to major changes in the system behavior.

In order to reach all possible states during the evolution process, a prohibitive amount of time would be required and it would imply the use of several maps with several different initial conditions, which would be close to the idea of an exhaustive search. However, we are interested in the investigation of the impact of such unmapped states on task accomplishment, and for that reason the complementary approach was proposed with the alternating application of random walk to move the robot in cases wherein the identified state has been rarely visited during evolution and therefore

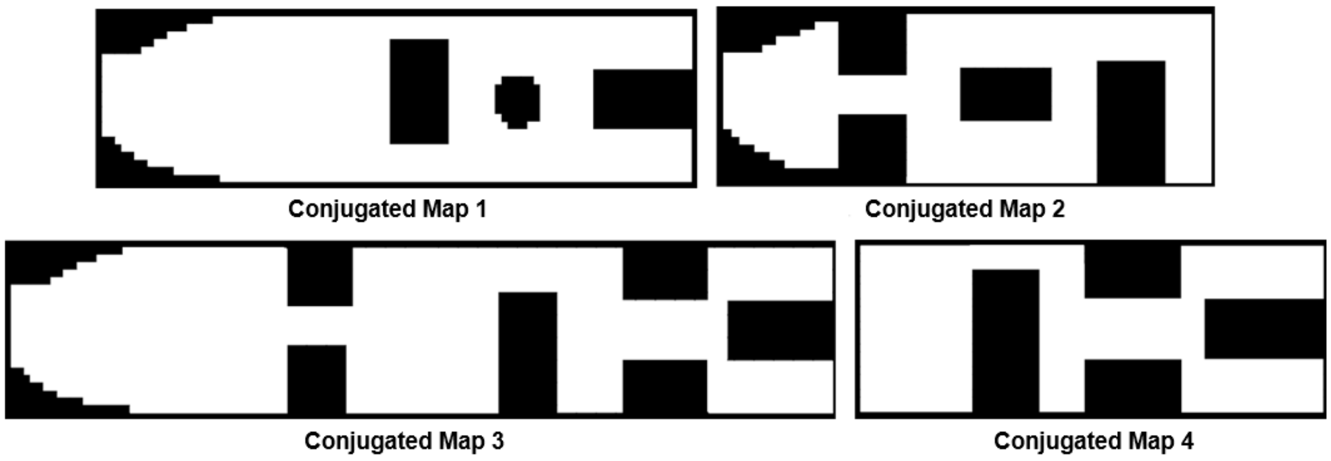

FIGURE 16. Conjugated maps.

TABLE 8. Results for conjugated maps (CM) with the complementary approach.

		CM 1	CM 2	CM 3	CM 4
9 robots	**Clearing**	10%	48%	8%	52%
	Iterations	199	156	413	213
	Draw	19.0%	18.9%	23.5%	18.4%
15 robots	**Clearing**	29%	70%	38%	57%
	Iterations	201	146	398	200
	Draw	23.3%	23.8%	27.3%	30.6%

not properly mapped to one of the actions. The next section presents the results obtained with this approach.

C. COMPLEMENTARY APPROACH

The conjugated maps of Figure 16 were chosen for the complementary approach tests and, as in the previous simulations with those same maps, groups of nine and 15 robots were used, and the partial recontamination of up to 70 cells was also allowed. Considering the existence of a random component, the tests were performed twice and the iteration limit accepted for the simulation was increased to 500, except for the conjugate map 3, where this value was set to 700. The iteration limit is increased in order to explore the random behavior of the Random Walk. Observe that for the evolutionary approach, increasing the limit does not have any effect as the actions are selected deterministically.

The cutoff point chosen was 100,000 visits, on an experimental basis and by observing the amount of visitation of the states. Table 8 shows the test results also performed with the individuals from experiments 8 and 9 in the same 30 positions used for the simulations whose results were presented in Table 7. In addition, it is also shown the percentage of iterations in which any of the robots opted for the draw, rather than the automaton mapped through evolution.

The results of Table 8 show an average improvement of 21% in the clearing rate in comparison to the tests carried out without the application of the draw. This indicates that the investigation of an alternative method to choose the action for these particular states is valid. In this paper we use a random walk, but one could use either a modified evolutionary process with a more complicated fitness function or some adaptation technique based on on-line learning to adjust the rarely visited states. It is also interesting to note, especially considering the size of the conjugated maps, that the frequency of the random walk usage and the number of iterations for convergence indicate that the robots were mostly still guided by the evolutionary approach.

D. COMPARATIVE RESULTS

In order to establish another parameter to evaluate the quality of solutions obtained through evolution, a comparative analysis between the random walk results presented in Section VI-A and the evolutionary approach was performed.

The comparisons are made using the maps of Figure 10 for the 12 initialization positions of the formation of each map. All tests were performed using four robots with the exception of Map 4, which required the use of five robots for the random walk. As discussed before, if four robots were to be used, the random walk approach would take a prohibitive amount of time.

The success rate for the evolutionary approach was 100% when the last generation of experiment 8 was used. The algorithm based on random walk was considered to have failed once in decontaminating the environment of Map 4 due to the high number of iterations (170,000) without the task being fulfilled. In the comparison of complexity, the iterations necessary for decontamination were considered, being recorded only when the experiments were successful.

In addition, some cases with very high number of iterations were considered outliers (only 2 simulations out of 96 were considered outliers) and were not included in the analysis to facilitate the visualization of the results, shown in Figure 17. Notice that these outliers were only generated in the random walk approach due to the stochastic nature of the approach.

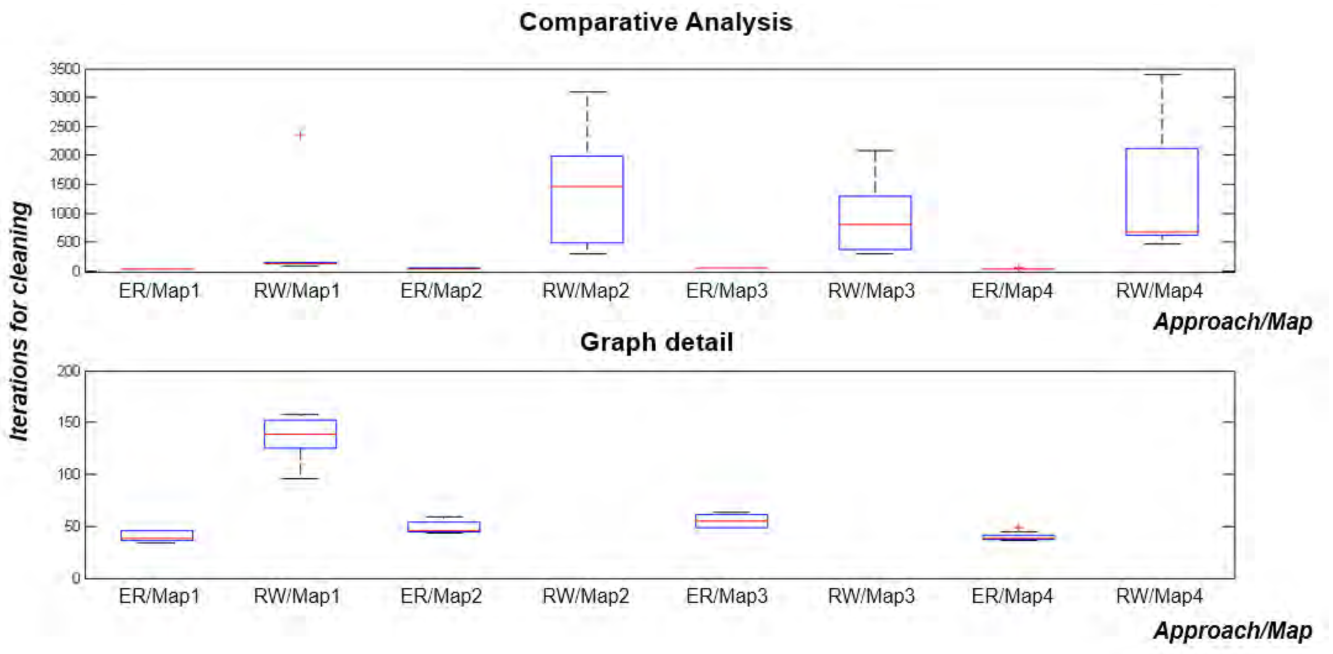

FIGURE 17. Comparative analysis between evolutionary (ER) and random walk (RW) approaches: full scale and detailed for up to 200 iterations.

Figure 17 presents the statistical comparison of the number of iterations between the evolutionary approach (ER) and the random walk approach (RW) for each of the four maps used. The top chart includes all the data, except for the tests considered outliers (with more than 3,500 iterations).

The bottom plot shows the details for up to 200 iterations, so that the results of the evolutionary approach are better visualized. The plots also show that the results of the two approaches are statistically different. Although the evolutionary approach requires a previous stage of evolution, once the solution is obtained, the decontamination time of the environment is considerably lower than in the application of the random walk, because the complete search process is performed with each new experiment.

VII. CONCLUSION

In this paper, solutions for the worst-case pursuit evasion problem were proposed. First a solution based on random walk and then a solution based on evolutionary robotics. Both solutions use the same problem formulation and a comparison of both was offered.

The random walk approach is based on a graph that is created based on restrictions that do not allow the environment to be recontaminated. The graph is also shown to cover the whole range of possibilities for the environment. Furthermore, the environment can be cleared in finite time, although the time can be very large.

The evolutionary approach discussed in the paper extended the results presented earlier for the solution of the worst-case pursuit evasion problem. All the robots in the group have the same control system based on a finite automaton, whose mapping of states in actions is subject to an evolutionary algorithm. Evolution is guided by the evaluation of the collective behavior resulting from robots actions, simulated in discrete maps.

The results point to reasonable solutions found with a relatively low number of generations. The derived control system is capable of solving the problem for some, but representative, types of maps, even some maps that were not presented during the evolutionary process. In addition, the automata were also able to generalize for several variations of initial conditions and to a larger number of robots.

It was observed that the solution does not generalize to all initial conditions. The reason for this could be related to the completeness of the finite state machine, for our method does not guarantee that the whole state machine is correctly derived. In order to investigate this issue, tests were done with a complementary approach, in which a random walk solution was combined with the evolutionary approach, determining the robots' actions when specific states were considered not have been sufficiently visited during evolution. The results showed that the control systems improved their success rate in the decontamination of the conjugate maps by 21%, considering the average of all the maps for groups of nine and 15 robots. For this reason, in the future, we intend to use a learning technique (Learning Automata) to adapt to the states not correctly defined by the evolutionary method or not visited during evolution.

On the other hand, the comparative analysis of results performed with the random walk approach showed that the proposed evolutionary process, despite requiring the previous adaption of the control system, presents a solution capable of decontaminating the same maps with a much smaller number of iterations.

Besides the development of the Learning Automata, in the future we intend to test our solution in Unmanned Air Vehicles (UAVs) to study the portability of the method.

REFERENCES

[1] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," *Auto. Robots*, vol. 31, no. 4, pp. 299–316, 2011.
[2] J. W. Durham, A. Franchi, and F. Bullo, "Distributed pursuit-evasion without mapping or global localization via local frontiers," *Auto. Robots*, vol. 32, no. 1, pp. 81–95, 2012.
[3] T. D. Parsons, "Pursuit-evasion in a graph," in *Theory and Applications of Graphs*. Berlin, Germany: Springer, 1978, pp. 426–441.
[4] A. Kolling and S. Carpin, "Multi-robot pursuit-evasion without maps," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2010, pp. 3045–3051.
[5] A. Gonçalves, E. Carvalho, L. Molina, and E. Freire, "Random walk-on-crossover problem," in *Proc. 12th Brazilian Symp. Intell. Autom. (SBAI)*, Oct. 2015, pp. 1460–1465.
[6] L. Gregorin, E. Freire, E. Carvalho, L. Molina, and S. Givigi, "Evolutionary robotics applied to the multi-robot worst-case pursuit-evasion problem," in *Proc. IEEE 7th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON)*, Oct. 2016, pp. 1–7.
[7] A. Acharyya, "Random walk with nonuniform angular distribution biased by an external periodic pulse," *Eur. J. Phys.*, vol. 37, no. 6, p. 065104, 2016. [Online]. Available: http://stacks.iop.org/0143-0807/37/i=6/a=065104
[8] V. Isler, S. Kannan, and S. Khanna, "Randomized pursuit-evasion in a polygonal environment," *IEEE Trans. Robot.*, vol. 21, no. 5, pp. 875–884, Oct. 2005.
[9] A. Kolling and S. Carpin, "Pursuit-evasion on trees by robot teams," *IEEE Trans. Robot.*, vol. 26, no. 1, pp. 32–47, Feb. 2010.
[10] L. Lovász, "Random walks on graphs: A survey," in *Combinatorics, Paul Erdős is Eighty*, vol. 2, D. Miklós, V. T. Sós, and T. Szőnyi, Eds. Keszthely, Hungary: János Bolyai Mathematical Society, 1996, pp. 353–398.
[11] M. Kang, "Random walks on a finite graph with congestion points," *Appl. Math. Comput.*, vol. 153, no. 2, pp. 601–610, 2004.
[12] S. Ikeda, I. Kubo, and M. Yamashita, "The hitting and cover times of random walks on finite graphs using local degree information," *Theor. Comput. Sci.*, vol. 410, no. 1, pp. 94–100, 2009.
[13] J. L. Palacios, "On hitting times of random walks on trees," *Stat. Probab. Lett.*, vol. 79, no. 2, pp. 234–236, 2009.
[14] H. Chen and F. Zhang, "The expected hitting times for finite Markov chains," *Linear Algebra Appl.*, vol. 428, nos. 11–12, pp. 2730–2749, 2008.
[15] G. F. Lawler, "Expected hitting times for a random walk on a connected graph," *Discrete Math.*, vol. 61, no. 1, pp. 85–92, 1986.
[16] J. L. Palacios, "Bounds on expected hitting times for a random walk on a connected graph," *Linear Algebra Appl.*, vol. 141, pp. 241–252, Nov. 1990.
[17] J. C. Bongard, "Evolutionary robotics," *Commun. ACM*, vol. 56, no. 8, pp. 74–83, 2013.
[18] S. Doncieux, N. Bredeche, J.-B. Mouret, and A. E. Eiben, "Evolutionary robotics: What, why, and where to," *Front. Robot. AI*, vol. 2, p. 4, Mar. 2015.
[19] L. König, S. Hettiarachchi, W. M. Spears, and S. Mostaghim, and H. Schmeck, "Decentralized evolution of robotic behavior using finite state machines," *Int. J. Intell. Comput. Cybern.*, vol. 2, no. 4, pp. 695–723, 2009.

LÍVIA GREGORIN received the master's degree in electrical engineering from the Universidade Federal de Sergipe (UFS), Aracaju, Brazil. She is currently a Lecturer with the Department of Electrical Engineering, UFS. Her current research interests include evolutionary robotics, electronics, and instrumentation.

SIDNEY N. GIVIGI (SM'14) received the Ph.D. degree in electrical and computer engineering from Carleton University, Ottawa, ON, Canada. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Royal Military College of Canada, Kingston, ON, Canada. His current research interests include autonomous systems and robotics.

EDUARDO FREIRE received the Ph.D. degree in electrical engineering from the Universidade Federal do Espírito Santo, Vitória, Brazil. He is currently an Associate Professor with the Department of Electrical Engineering, Universidade Federal de Sergipe, Brazil. His research is related to robotics, control, and artificial intelligence.

ELYSON CARVALHO received the Ph.D. degree in electrical engineering from the Universidade Federal de Campina Grande, Campina Grande, Brazil. He is currently an Assistant Professor with the Department of Electrical Engineering, Universidade Federal de Sergipe, Brazil. His research is related to robotics and instrumentation.

LUCAS MOLINA received the Ph.D. degree in electrical engineering from the Universidade Federal de Campina Grande, Campina Grande, Brazil. He is currently an Assistant Professor with the Department of Electrical Engineering, Universidade Federal de Sergipe, Brazil. His research is related to robotic manipulators, mobile robotics, and control.