Evolutionary Optimization of Intruder Interception Plans for Mobile Robot Groups

Wojciech Turek, Agata Kubiczek, and Aleksander Byrski(✉)

AGH University of Science and Technology, Krakow, Poland
{wojciech.turek,olekb}@agh.edu.pl, agat.kubiczek@gmail.com

Abstract. The task of automated intruder detection and interception is often considered as a suitable application for groups of mobile robots. Realistic versions of the problem include representing uncertainty, which turns it into NP-hard optimization tasks. In this paper we define the problem of indoor intruder interception with probabilistic intruder motion model and uncertainty of intruder detection. We define a model for representing the problem and propose an algorithm for optimizing plans for groups of mobile robots patrolling the building. The proposed evolutionary multi-agent algorithm uses a novel representation of solutions. The algorithm has been evaluated using different problem sizes and compared with other methods.

Keywords: Intruder detection · Mobile robot motion planning · Evolutionary computing

1 Introduction

The problem of securing buildings from unauthorized intrusion is often considered a very suitable application for groups of mobile robots. Repetitive monitoring of a given area, systematic observation and possible interaction with hostile intruders fits perfectly the popular Dull-Dirty-Dangerous rule of robotization. This is one of few tasks where robots can be more efficient than humans, even when leaving financial expense aside.

These reasons have motivated many researchers to consider various variants of the problem over the last few decades. The comprehensive survey presented in [7] locates the background of the problem in operations research, graph theory, classical search theory and differential games. Existing approaches are grouped into two main categories: *pursuit-evasion* games and *probabilistic search*.

The *pursuit-evasion* games, including the well-recognized *cops and robbers* game [1], focus on finding the worst-case guarantee of finding an intruder in the specified environment. The assumptions typically include faultless motion and detection and the aim is to minimize the number of pursuers, which provide the guarantee.

J. M. F. Rodrigues et al. (Eds.): ICCS 2019, LNCS 11536, pp. 642–655, 2019.
https://doi.org/10.1007/978-3-030-22734-0_47

More realistic assumptions of detecting an intruder require using different methods and lead to probabilistic results rather than guarantees. The problems described as *probabilistic search* focus on modeling robots' sensors inaccuracy and uncertainty of intruder's decisions.

The variant of the intruder interception problem, presented in this paper, is considered an open problem in the domain. A group of mobile robots operates in a known, indoor environment. Each robot is able of detecting an intruder with limited probability. Intruder's decisions are uncertain, there is no guarantee of particular decision in specific circumstances. The environment can be equipped with static sensors, also imperfect. The aim is to reduce the probability of intruder's presence in the secured area.

In this paper we present the definition of the intruder interception problem, which represents realistic assumptions of intruders' unpredictability and inaccuracy of detection. We propose a novel approach to modeling of the problem, which represents crucial features of the problem. The model of intruder(s) was constructed based on a quasi-probabilistic measure spreading across the graph nodes (parts of building's model). The robots were dispatched based on redirection lists present in the nodes. Such redirection lists were used for defining an evolutionary algorithm (encoded in a dedicated genotype and evaluated by a predefined fitness function) for finding and optimizing plans for mobile robots groups. We evaluate the solution in different scenarios and compare the results with a reference, greedy algorithm. We also compare two variants of evolutionary algorithm, showing significant superiority of the Evolutionary Multi-Agent approach.

In the next section the excerpt from state-of-the-art regarding the intruder interception is provided. Next the problem is discussed and its novel model is presented. In the subsequent section the planning algorithm is shown and later the experimental evaluation is presented. Finally the paper is concluded, showing other applications of the proposed method.

2 Existing Approaches to the Intruder Interception Problem

The considered problem of indoor intruder interception using mobile robots is located within the wider area, often referred to as target detection and tracking. A comprehensive review of related work has been presented by Chung et al. in [7] and recently further extended by Robin et al. [14]. The area includes problems like target identification, tracking and following, however the most challenging open problems are related to finding moving targets [9].

The target detection and tracking taxonomy (fragment presented in Fig. 1) divides the mobile search problem into three categories. The firs one, *Capture*, is also often referred to as pursuit-evasion. Particular problems within this category share common aim: to ensure that no intruder was left in the secured area. Providing such guarantees forces strong assumptions concerning motion and detection model, moving the solutions far from reality.

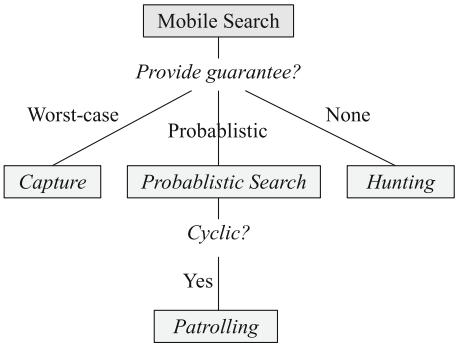

Fig. 1. Taxonomy of mobile search problems [14]

Typical approach to pursuit-evasion problem modelling is using graphs for representing environments. With rooms represented by vertexes and accessibility relation modelled by edges, the problem becomes a discrete planning task, often called the *graph-clear* [2]. In its basic form, each pursuer clears a node in which it is located. Pursuers must form a formation, which splits the graph into clear and potentially contaminated parts. Among various existing variants, the work presented in [11] is important in the context of our work. Besides the new variant of the model, which reflects additional features of reality, the authors show, that the problem of finding the minimal number of pursuers is NP-hard for general graphs.

Two other categories of mobile search problems, that is probabilistic search and hunting, do not provide guarantees and therefore focus on finding minimal number of robots needed. The task is defined as a planning problem, which aims at minimization of intruder presence probability over time, with many variants, of course. Typically more realistic models of the environment, the intruders and perception are considered. In [8] the environment is represented as a grid, and the intruder is modelled using the partially observable Markov decision process. In [13] a particle filter is used to represent possible locations of intruder; the aim is to maximize of number of visible particles while potential fields are used to control the pursuers. An example of optimization is presented in [15], where local optimization of trajectories is used. The common feature of the considered problems is the exponential complexity which prevents from finding global optimum.

Analysis of the existing approaches shows that among the three major problem definition dimensions (environment model, target motion model, pursuer sensing model) the environment model implies most consequences. The dependency between the used environment model and problem complexity has been discussed in [10], where the need for simplifying the model is underlined. Most of existing approaches use metric maps or grid representations, although graph-based models are typically far more efficient. However, automated transformations of metric data into a meaningful topological model is not straightforward.

The solution proposed in this paper uses our previously proposed method [16]. It generates a list of spaces from a provided list of walls, identifies passages between spaces and generates a graph. The graph lacks semantic information, however it can be automatically enriched with various metric data derived from the features of the spaces.

According to [14], further investigation is needed in the area of realistic models and their efficient processing. Also focus on sub-optimal solutions is expected due to the nature of the problem. The variant of the intruder interception problem, presented in this paper, is located within these areas.

3 Problem Definition and Model

In order to define precisely the problem considered in this paper, let us assume that there is a known building. The topological map of the building is known, it is composed of rooms and passages between rooms. It can be created manually or by using methods presented in [16]. In the normal situation nobody incidental is present (no human security), only the robots and potential intruder(s). In the case of intruder(s) trespassing, he should be quickly intercepted – and this is the task of the robot-security group.

Certain knowledge regarding the presence of intruders in a particular rooms can become available at any time – e.g. derived from a set of static sensors, present in strategic places of the building. Alternatively, the information about possible intrusion can be deduced from the structure of a building, assuming rooms with windows on the ground floor more prone.

Certain knowledge about possible movements of the intruder can also be available. The intruder's velocity, aims and decisions can be considered: unknown, random, deliberative or adversarial. The knowledge about intruder's decisions have probabilistic character – only the probability of certain actions can be predicted.

The group of robots can be located in one place or spread around the building. Robots can be heterogeneous, differing in velocity and perception capabilities. Each robot is able to detect an intruder located in the same room, with a certain probability, as the intruder can be overlooked.

The planning task for the group of robots is centralized. The goals of the planning is to:

- plan the movement of the robots in such way, that the probability of intruders' presence is minimized in the shortest possible time,
- adopt to the detected changes and new information when one is available – the system should work continuously.

With these assumptions, let us model the building as a graph, $G = (V, E)$, where V is a set of labels of the nodes. $V = \{v_1, \ldots, v_k\}, k \in \mathbb{N}$ (k is the maximum number of vertices). Each node is associated with one room, while one room is represented by d nodes, where d is the number of passages of the room. So a node can be interpreted as one passage (door) connecting a room with other

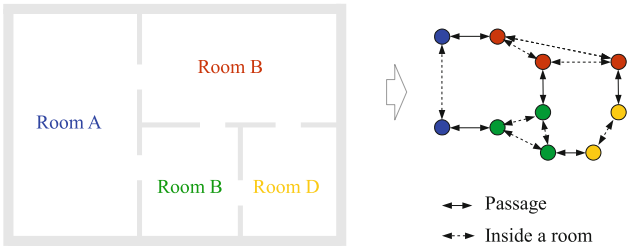

Fig. 2. Example of the proposed graph model of a building

rooms (see Fig. 2). Each passage is represented by two nodes, one in each of the connected rooms.

E is the set of edges $E = \{e_{1,2}, e_{1,3}, \ldots, e_{j,k}\}, i, j \in \mathbb{N}, \{i, j\} \in \{1, \ldots, k\}^2$, $i \neq j$. Typically there are edges fully connecting all nodes associated with one room. Additional edges represent passages – these can be directional. Each edge has a weight, which is proportional to the distance (or travel time) between passages represented by the nodes.

By using this modelling method, a complex building can be represented as a weighted graph, representing the topology of rooms and passages, enriched with distances between passages within the rooms. It allows finding paths and storing additional information about model state (information about robots and intruders).

Let us define a probabilistic model of an intruder, which can be located in a certain room, by defining a quasi-probabilistic measure of intruder's presence in a certain room $\mathbb{R} \ni x_n \geq 0$. This value can exceed 1 (so it is not a probability per se), and can be treated as a certain estimate of a number of intruders present at a certain node of the graph.

To define the dynamics of intruders and robots, let us assume the existence of discrete time units. In each time unit an intruder can stay at the current graph node n or start moving to an adjacent node. The decisions are driven by intruder's model, a certain probability p_i. The presence measure changes in each of the nodes according to the following equation:

$$x'_n \leftarrow x_n - \sum_{i=1}^{m} p_i x_n + \sum_{i=1}^{m} r_i \tag{1}$$

where x'_n is the presence measure in the next time unit (short for $x_n(t+1)$ as the next observed value of $x_n(t)$) and p_i is the probability of moving of an intruder from n-th to i-th node. $\mathbb{R}^+ \ni r_i \geq 0$ is the presence metric value of an intruder approaching from the i-th, which has been subtracted from the x_i value k time steps earlier (k depends on the edge length between i and n and the intruder's velocity).

The values of intruder presence measures are computed for each of the nodes in each observable moment. Thus overall we get a certain quasi-probabilistic

measure of the presence of intruders in the graph, that gets "distributed" through its nodes and edges according to the Eq. 1.

When the robot comes to visit certain room (any of the room's nodes), the intruder presence measure in the room (nodes and edges) is decreased according to the following equation:

$$x'_n \leftarrow x_n \cdot (1 - p_d) \tag{2}$$

where p_d is the probability of detecting of the intruder by the robot.

4 Planning Algorithm

Based on the graph depicting the building, defined in the previous section, the planning algorithm is constructed. Note, that the graph represents presence measure of one or more intruders, and this presence distribution changes over time. At each observable time moment, the planning algorithm can assess the place of the most probable spotting of the intruder. Here two algorithms will be presented, the former is very easy though efficient, and will be used as a reference one, while the latter is one of tangible results of the research presented.

4.1 Greedy Reference Algorithm

Assuming $R = \{r_1, \ldots, r_k\}$ the set of robot labels, a simple greedy algorithm, assigning the robots, present in the node v_x, to the next nodes may be proposed. The algorithm takes into consideration the number of already realized robot visits in the neighboring nodes:

```
function assignRobotsToNextNodes(v_x, map)
    neighbors ← getNeighborsOfNode(v_x, map)
    sortAccordingToAscendingVisitCount(neighbors)
    nextNode ← getFirst(neighbors)
    incrementVisitCount(nextNode)
    return nextNode
end function
```

thus the robots present in the node v_x will be assigned with the next target – one of the neighboring nodes, while the less visited nodes will be prioritized.

Note that the proposed algorithm uses the local knowledge present in the graph to dispatch the robots, and although it is centralized, it works very efficiently, even for relatively large graphs. Such algorithm must be used instead of deterministic, "brute-force" planning as the considered problem apparently belongs to NP class (it is quite similar to VRP problems).

Locally-focused algorithms are prone to stucking in local extrema, therefore being aware that such algorithm can deliver certain (and easy to attain), useful, sub-optimal result, we might turn towards using state-of-the-art global optimization algorithm to make these results better.

4.2 Plan Optimization

Finding satisfactory solutions to the considered, *NP*-hard planning problem justifies using general-purpose global-optimization metaheuristics, like the well-known evolutionary algorithms. It is to note that some of such algorithms were subjected to detailed formal analysis, showing the proof that they will be able to reach the solution, wherever it is in the search space [3,17]. Every possible evolutionary algorithm (and many other metaheuristics), in order to solve a problem, require a proper definition (and encoding) of a problem, construction of the fitness function in the problem domain and construction of the variation operators. The main novelty of the proposed approach is the solution definition and encoding, which does not focus on paths of particular robots, but rather represents routes to cover.

Let us introduce the concept of a dispatch list $s \in S$, which is a list of node labels of a certain length dl. A valid dispatch list for node v can contain only labels of nodes adjacent to v and the label of v. A valid dispatch list can contain duplicates of node labels. The dispatch list encodes consecutive redirections of robots entering the node; once the list is finished, dispatching starts from the first item again.

The search space S includes all the possible dispatch lists of a certain length dl for a given graph (of n nodes):

$$S = V^{dl \cdot n} \tag{3}$$

a vector of dispatch lists, belonging to such space, can be depicted as:

$$S \ni s = (d_1, d_2, \ldots, d_n) \tag{4}$$

where

$$d_i = (v_r, \ldots, v_s) \tag{5}$$

where $||d_i|| = dl$ and $(v_r, \ldots, v_s)$ are all adjacent to v_i or equal v_i.

As an example, imagine we have simple, full graph consisting of four nodes: V = {A, B, C, D}. Considering dispatch lists of the length equal to 3, one of possible solutions of the problem can look as follows:

$$((A, B, C), (D, C, B), (B, C, A), (D, C, A)) \tag{6}$$

This sample solution will be interpreted as a sequence of orders: "When the first robot arrives to node A, send it again to node A, then send the second one to B. When the first robot arrives to node B, send it to node D, then send the second one to C, the third one to B and the fourth one again to D", and so on.

The goal is to find such dispatch list, that the intruder will be localized in the shortest possible time. Coming back to the definition of the presence measure, we can construct the fitness function as a simple sum of the presence measure over the whole graph after a fixed time:

$$f = \sum_{i=1}^{n} x_i \tag{7}$$

and check this value after certain, predefined time of dispatching the robots according to the current schedule. Thus the evaluation function is defined. Computing the fitness function value requires performing the simulation of the process: intruder spreading and robots movements.

The remaining steps are crossover and mutation. The representation treated as dispatch lists for all the nodes may be very easily subjected to any crossover working with combinatorial problems, e.g. discrete crossover. So the offspring will randomly inherit subsequent elements of the dispatch list either from one or another parent, e.g. let us focus only on the dispatch list for two parents and offspring:

$$crossover((\mathbf{B}, \mathbf{D}, B), (D, C, \mathbf{B})) \rightarrow (B, D, B) \tag{8}$$

assuming that these two dispatch lists belong to the same node (A), and this node is connected to all other nodes.

The mutation is also very simple. Focusing on the dispatch list for a certain node, its elements can be randomly altered:

$$mutation(D, \mathbf{C}, B) \rightarrow (D, B, B) \tag{9}$$

again the considered node A is connected with all other ones.

The presented approach has been implemented and tested using two evolutionary metaheuristics: an Evolutionary Algorithm and an Evolutionary Multi-Agent System.

4.3 Evolutionary Algorithm

The first type of evolutionary algorithms, namely genetic algorithm was proposed by Holland and updated by Rechenberg, Schwefel, Fogel, Michalewicz and many others [18]. This optimization metaheuristic requires using a dedicated encoding of a problem as a genotype vector, a random construction of a population of individuals (a set of such vectors) and submitting this population to a cycle of operations, which would process the population according to evolutionary inspiration by Darwin's theory of evolution. Holland's original algorithm used binary encoding and proportional selection. There do exist many other popular encodings, e.g. real-valued, discrete, tree-based and many more (relevant to particular problems to be solved).

In the presented solution we are using an Evolutionary Algorithm (similar to the one proposed by Michalewicz [12]). It consists in realization of the following sequence of operations: (1) random initialization of a population (minding the constraints of the problem), (2) evaluation of the individuals by running the simulation of intruders and robots, (3) selection of a mating pool, (4) generation of the next population by means of crossover i.e. creation of new individuals based on randomly selected parents, (5) introduction of random aberrations to the solutions of the new population (mutation). At this point the algorithm returns to (2). The stopping condition of the loop is specified by the limit of real computations time.

4.4 Evolutionary Multi-agent System

Another evolutionary-type algorithm used in the experiments presented in this paper is the Evolutionary Multi Agent-System [6]. This metaheuristic has a firm formal background proving its correctness [3]. It may be treated as an approach (proposed first by Cetnarowicz in 1996 and developed significantly since then) to put together autonomy of agent-based systems with the optimization capabilities of evolutionary-inspired metaheuristics.

Agents in EMAS represent solutions to a given optimization problem. They are located on islands representing distributed structure of computation. The islands constitute local environments, where direct interactions among agents may take place. In addition, agents are able to change their location, which makes it possible to exchange information and resources all over the system.

In EMAS, phenomena of inheritance and selection – the main components of evolutionary processes – are modeled via agent actions of *death* and *reproduction*. As in the case of classical evolutionary algorithms, inheritance is accomplished by an appropriate definition of reproduction. Core properties of the agent are encoded in its genotype and inherited from its parent(s) with the use of variation operators (mutation and recombination). Moreover, an agent may possess some knowledge acquired during its life, which is not inherited. Both inherited and acquired information (phenotype) determines the behavior of an agent. It is noteworthy that it is easy to add mechanisms of diversity enhancement, such as allotropic speciation (cf. [5]) to EMAS. It consists in introducing population decomposition and a new action of the agent based on moving from one evolutionary island to another.

Many optimization tasks, which have already been solved with EMAS and its modifications, have yielded better results than certain classical approaches. They include, among others, optimization of neural network architecture, multi-objective optimization, multimodal optimization and financial optimization. EMAS has thus been proved to be a versatile optimization mechanism in practical situations. A summary of EMAS-related review is given in [4].

5 Experiments and Results Evaluation

In order to evaluate the two algorithms implementing the presented plan optimization method, a series of experiments was carried out. Three different building models have been used in the tests. The first one was a ring of 8 rooms, each having 2 doors. The second (Fig. 3) was far more complex, with four stories and multiple rooms. The last one, presented in Fig. 4, is a model of a real, existing building.

The experiments were performed on a single computer with 4 core CPU and 12 GB of RAM. Each test case has been run 30 times. The diagrams present aggregated results of all launches. Each case is represented by the series of data: the best result obtained in any launch for the given fitness evaluations count, the worst result for the given evaluations count, and the average value for all launches.

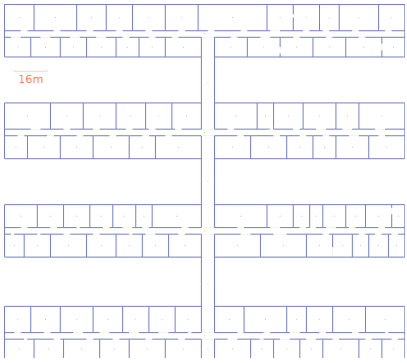

Fig. 3. Large, multi-storey building

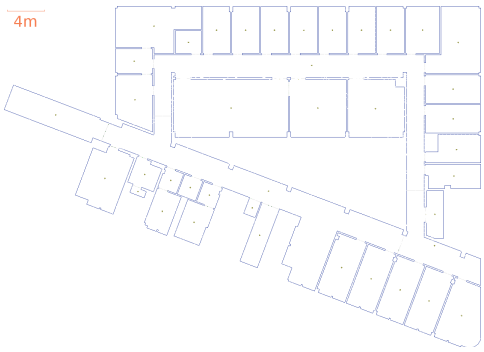

Fig. 4. Real-life building

5.1 Test Cases

The purpose of the experiments was to evaluate the performance of the algorithm depending on parameters configuration and the test building structure. Four test cases have been proposed. The first one includes simple building scheme and its goal is to compare the results of two different evolutionary algorithms and a greedy one for the same parameters configurations. Another two test cases involve the real-life building and were created to examine the effect configuration has on evolutionary algorithm performance. Two parameters have been analyzed - dispatch list's length and population size. The last test case involves complex multi-storey building. It examines how number of robots exploring the building affects the results of evolutionary and greedy algorithm.

5.2 Results

The results of the simple building experiment are presented in Fig. 5. Separate series illustrate the fitness values for EMAS and for conventional evolutionary algorithms. Constant value represents the greedy algorithm result.

Both evolutionary algorithms produce much better results than the greedy approach and is this case the fitness reduces to value close to zero. That means the solution close to the optimum could be achieved. At first the conventional EA produced better results but eventually the best results come from EMAS. Moreover, the EMAS offers much better results for worst-case scenario (comparing worst results achieved in any launch for the given fitness evaluations count).

The experiments involving real-life building were designed to examine the influence of configuration on the results. In the first case the dispatch list's length was analyzed. The results are presented in Fig. 6, for better readability only average series were included. It can be observed that there is a dispatch list length which yield the best solutions. For the analyzed building list of 10–12 elements seems optimal. Increasing the length of the list improves the results to

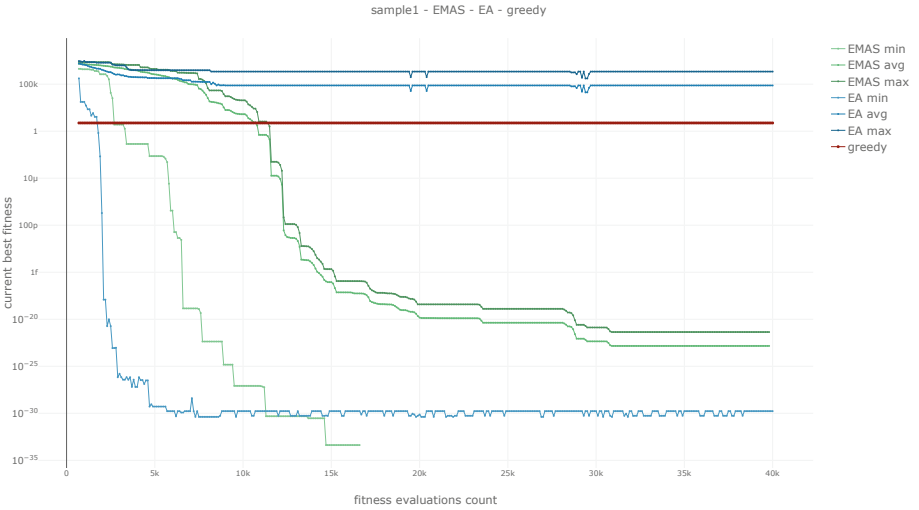

Fig. 5. Results of both evolutionary and greedy algorithms for the small building case

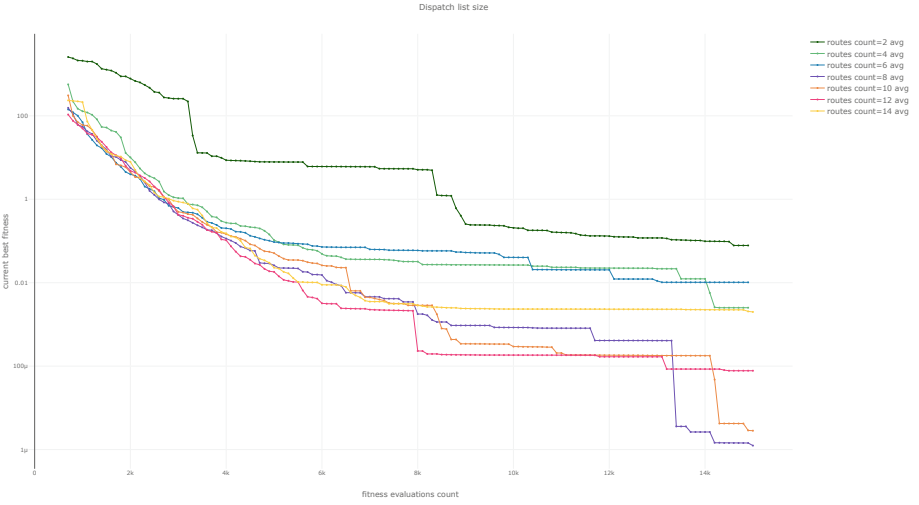

Fig. 6. Evolutionary algorithm results for real-life building calculated for various dispatch list lengths

some extent, as the solution space extends. However, when the list is too long the last elements of solution are never used and they impede the computation.

The second real-life scenario involved examining the population size's impact on the performance. In order to obtain best results, the dispatch list size has been set to the value determined in previous experiment. The results are presented in Fig. 7 - as before, only average series are included. Similar to the dispatch

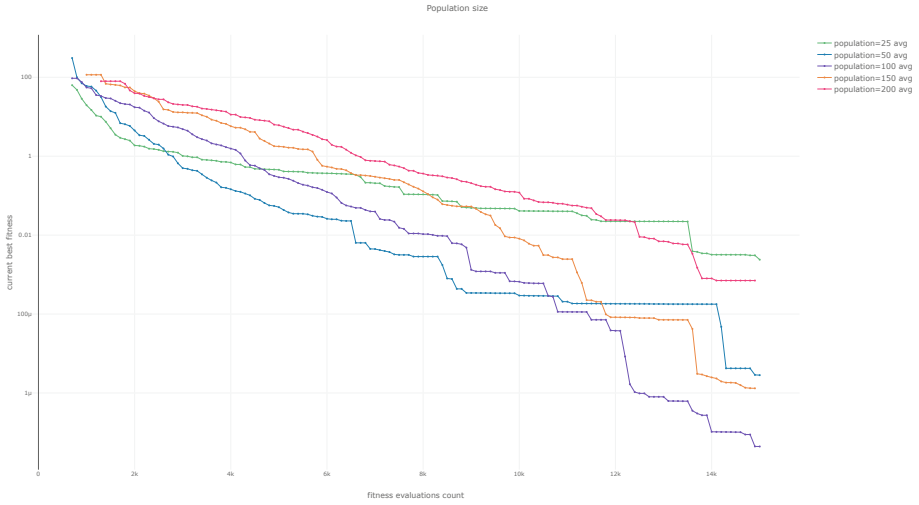

Fig. 7. Evolutionary algorithm results for real-life building calculated for various population sizes

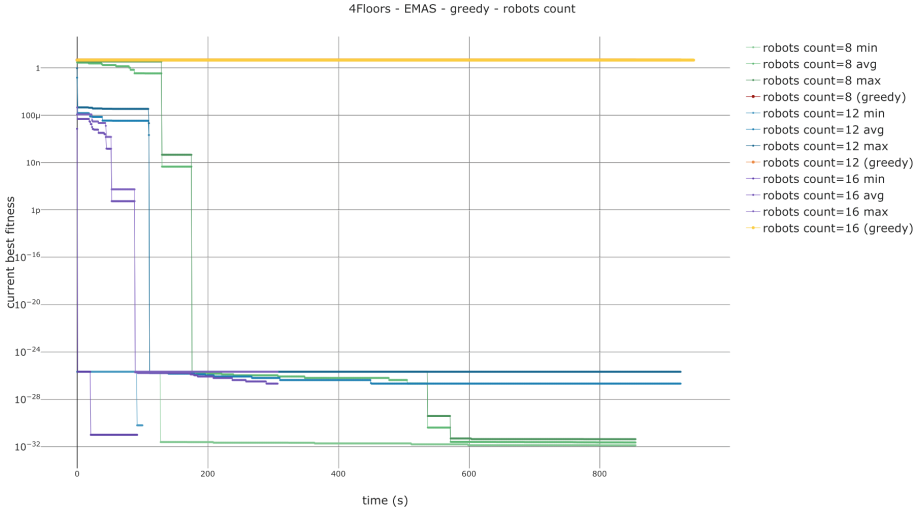

Fig. 8. Results of EMAS and greedy algorithms calculated for multi-storey building and various robots count

list size, an optimum population size can be determined and for this case equals 100 individuals. If the population is too small it becomes less diverse and this can cause stopping the algorithm in local optimum. However, too big population implies more fitness evaluations and therefore fewer epochs of evolution.

The last experiment examined the influence of number of robots exploring the building on the algorithms' performance. The results are presented in Fig. 8.

This test case involved complex multi-storey building and comparison of EMAS and greedy approach results. As can be expected, for both methods performance improved with the increasing robots' count, however the improvement in case of the greedy algorithm is hardly visible, also due to logarithmic scale on the vertical axis. The EMAS algorithm found far better solutions and made use of more numerous groups of robots. Larger number of robots allowed finding good solutions faster, which is visible in the first 200 s of optimization.

6 Conclusions and Further Work

In this paper a novel approach to planning of intruder interception by a group of mobile robots is proposed. A dedicated intruder model has been constructed, based on a quasi-probabilistic presence measure. Actual plan of the mobile robots movement was constructed based on redirection lists distributed among the nodes of the graph. These lists became parts of directly encoded genotype, subjected to a process of evolutionary optimization.

The evolutionary approach to solving this problem (based on EA and EMAS algorithms) were compared with a dedicated (very natural) greedy algorithm and it turned out that the evolutionary approach (in particular EMAS) prevailed significantly, being able to lower the presence measure in the whole graph to the greatest extent.

The proposed method of intruder localization by the means of multiple robots turns out to be quite similar to VRP or MTSP problems. In future we are planning to test existing VRP/MTSP benchmarks in order to prove the applicability of the proposed model to those very important hard computation problems, showing the generality of the proposed approach.

Acknowledgments. The research presented in this paper was partially supported by the funds of Polish Ministry of Science and Higher Education assigned to AGH University of Science and Technology.

References

1. Aigner, M., Fromme, M.: A game of cops and robbers. Discrete Appl. Math. **8**(1), 1–12 (1984)
2. Alspach, B.: Searching and sweeping graphs: a brief survey. Le matematiche **59**(1–2), 5–37 (2004)
3. Byrski, A., Schaefer, R., Smołka, M.: Asymptotic guarantee of success for multi-agent memetic systems. Bull. Pol. Acad. Sci. Tech. Sci. **61**(1) (2013)
4. Byrski, A., Drezewski, R., Siwik, L., Kisiel-Dorohinicki, M.: Evolutionary multi-agent systems. Know. Eng. Rev. **30**(2), 171–186 (2015)
5. Cantú-Paz, E.: A survey of parallel genetic algorithms. Calculateurs Paralleles, Reseaux et Systems Repartis **10**(2), 141–171 (1998)
6. Cetnarowicz, K., Kisiel-Dorohinicki, M., Nawarecki, E.: The application of evolution process in multi-agent world (MAW) to the prediction system. In: Proceedings of the 2nd International Conference on Multi-Agent Systems (ICMAS 1996). AAAI Press (1996)

7. Chung, T.H., Hollinger, G.A., Isler, V.: Search and pursuit-evasion in mobile robotics. Auton. Rob. **31**(4), 299 (2011)
8. Ferrari, S., Fierro, R., Tolic, D.: A geometric optimization approach to tracking maneuvering targets using a heterogeneous mobile sensor network. In: Proceedings of the 48th IEEE Conference on Decision and Control (CDC), pp. 1080–1087 (2009)
9. Garg, V., Tiwari, R.: A chronological review of the approaches used for multi robot navigation. In: International Conference on Recent Trends in Engineering, Science Technology - (ICRTEST 2016), pp. 1–8, October 2016
10. Hollinger, G., Singh, S., Djugash, J., Kehagias, A.: Efficient multi-robot search for a moving target. Int. J. Rob. Res. **28**(2), 201–219 (2009)
11. Kolling, A., Carpin, S.: Pursuit-evasion on trees by robot teams. IEEE Trans. Rob. **26**(1), 32–47 (2010)
12. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer, Heidelberg (1996). https://doi.org/10.1007/978-3-662-03315-9
13. Mottaghi, R., Vaughan, R.: An integrated particle filter and potential field method applied to cooperative multi-robot target tracking. Auton. Rob. **23**(1), 19–35 (2007)
14. Robin, C., Lacroix, S.: Multi-robot target detection and tracking: taxonomy and survey. Auton. Rob. **40**(4), 729–760 (2016)
15. Sarmiento, A., Murrieta-Cid, R., Hutchinson, S.: An efficient motion strategy to compute expected-time locally optimal continuous search paths in known environments. Adv. Rob. **23**(12–13), 1533–1560 (2009)
16. Turek, W., Cetnarowicz, K., Multan, M., Sośnicki, T., Borkowski, A.: Modeling buildings in the context of mobile robotics. Research Reports of Warsaw University of Technology. Electronics (194), 165–174 (2014)
17. Vose, M.D.: The Simple Genetic Algorithm: Foundations and Theory. MIT Press, Cambridge (1998)
18. Whitley, D., Sutton, A.M.: Genetic algorithms - a survey of models and methods. In: Rozenberg, G., Bäck, T., Kok, J.N. (eds.) Handbook of Natural Computing, pp. 637–671. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-540-92910-9_21