Abstract — In many surveillance applications, there are different properties of the environment to check. For example, in the case of robots surveilling an industrial depot, one could be interested in verifying for fire alarms, intrusion alarms or bio-hazards. It is very hard to characterize the solution of this problem in terms of a unique utility function. Indeed, this would require to define a measure of the tradeoff among objectives, which are, by definition, incommensurable quantities. In this case, these tradeoffs should be tuned according to contingencies and is, in general, a very difficult task. In this paper, we present an approach to address such issues. In particular, we define the multi-robot multi-objective surveillance problem and show how this can be solved in terms of multi-objective heuristic search. The approach has been experimented based on an off-line planner including simulated and real robot plan execution results.

I. INTRODUCTION

In the last years, security issues have raised new research focused on surveillance. One of the “hot” topics is how to automate surveillance tasks based on robotic platforms and fixed sensors (e.g., [10]). In such a setting surveillance is a complex problem and poses many technical challenges, among which finding tradeoffs among different and noncommensurate attributes. Indeed, people are reluctant to delegate to automated systems critical (and difficult to model) choices based on which may depend human lives. Nevertheless, especially when surveilling large areas, there are huge amounts of data to process and a human based surveillance system may be very expensive and inefficient.

In this paper, we present a multi-objective solution to this problem. In particular, we define a set of attributes, i.e. objectives, for the system to optimize. Based on these attributes we find the set of Pareto optimal solutions and delegate to the human operator the choice of which particular solution to pick. In this way, on the one hand, we significantly reduce the space of possible solutions to present to the human operator; on the other one hand, we leave to the human operator, based on his experience, the choice of the tradeoffs among the different solutions. For example, the system might require to check some alarms relative to different emergencies: intrusion alarms, fire alarms or bio-hazards. Generally, the sooner you gather information on a specific alarm the better it is. Clearly, given the finite resources of a system, it may be required to trade-off among the different objectives. One could choose, for example, to give priority to fire alarms with respect to intrusion alarms.

The approach presented in this paper focuses on the use of teams of robots in surveillance problems. In particular, we present a case study on robotic surveillance which spans from theoretical modeling to validation on a real multi-robot system. We present the first application of multiobjective heuristic search to a realistic problem by formalizing and implementing a multi-robot multi-objective planner. The plans produced by our planner are abstract and can not be executed as they are on real robots. In order to overcome this limitation we have devised an algorithm that is able to transform abstract plans in equivalent executable plans (namely Petri Net Plans [20]) which are more robust and can guaranty a safe execution. The system has been implemented in the openRDK framework and has been tested both in simulation and on real robots.

In the remainder of this paper, we first provide an analysis of the related work on multi objective problem solving (Section II). Then, we describe the surveillance problem (Section III) and present a solving method based on the multi-objective heuristic search algorithm MOA∗ [18] (Section IV). We conclude (Section V) discussing robotic related issues, ranging from executability of plans and implementation to experimental validation.

II. RELATED WORK

Multi-objective problems have to trade-off between incom-mensurable quantities (i.e. different objectives). In general, for such problems, the utility of a solution is defined as a vector of utilities, one for each objective. For example, assume that we have two distinct objectives and three different solutions with the following utilities: ⟨1, 3⟩, ⟨3, 1⟩ and ⟨1, 1⟩. In this case, it is clear that the third solution is worse for each objective with respect to the others, nevertheless we have no way to choose among the first two. This reflects the fact that we can write a total preference relation (in this case in the form of an utility function) w.r.t. a single objective, but we can just define a partial one among complete solutions. It is commonly agreed, for example [7], [17] or [12], [18], that a solution for a multi-objective problem should be Pareto optimal. A solution o is Pareto optimal if there is no other solution which is as good as o for all objectives and strictly better for at least one. In the previous example, ⟨1, 1⟩ is not Pareto optimal because ⟨1, 3⟩ is, as good as ⟨1, 1⟩ for the first objective, and strictly better for the second. In this example, the Pareto optimal solutions are ⟨1, 3⟩ and ⟨3, 1⟩. The set of Pareto optimal solutions, in general, is not a singleton. It is, thus, necessary to devise a method to further guide our selection. Nevertheless, finding a refinement for Pareto optimality is still an open problem [18].

Agents which form a team are willing to maximize the performance of the team. In multi-objective problems this requirement is formally defined as Pareto optimality. The major drawback of Pareto optimality is that, in general, it defines a space of possible solutions. It is still an open problem to understand which solution should be selected within this
Most approaches in literature overcome this limitation by defining some form of tradeoff between the objectives to evaluate which Pareto optimal solution is the best. For example, they define preferences over the objectives (e.g. [19]), they reformulate the problem as a single objective one (e.g. [16]) or they tradeoff each agent’s utility with respect to the team’s interest [13]. In this paper, we assume that the solution to our multi-agent multi-objective problem is the entire set of Pareto optimal solutions. The issue of selecting the appropriate Pareto optimal solution is delegated to a human operator provided with a graphical representation of the Pareto Optimal space.

In many cases of interest, the space of possible solutions is not continuous, but discrete, and has to be build incrementally through the application of operators. These problems are formulated as graph search problems where the goal is to find a set of preferred paths from a start node to a set of goal nodes. The AI community has studied the problem from an algorithmic perspective, in particular by extending heuristic search methods, as $A^*$ [15], to the multi-objective case, and producing a set of approaches known as multi-objective heuristic search. The first notable result was $MOA^*$ [18], an extension of $A^*$, able to find the set of Pareto optimal solutions through a heuristic guided search. Nevertheless, $MOA^*$ suffers from exponential memory requirements, which prevent its application to real problems. Many different approaches have been devised to extend $MOA^*$ and can be categorized based on the different research goals they pursue: 1) improve the computational efficiency of the solving technique (e.g. [4], [6], [9], [11], [12]) and 2) generalize the class of solvable problems (e.g. [3], [8], [14]). In this work we will use $MOA^*$ as the main solving technique as it is the most well studied and established approach.

III. Multi-Objective Surveillance Problem

In this section, we present a multi-objective multi-robot surveillance problem. In particular, we assume that a team of robots has to surveil an environment which contains many (cheap) sensor nodes. These sensors are heterogeneous and can sense different properties of the environment, which can be associated to different alarms. For example, a thermal sensor can signal possible fire alarms or a motion sensor can signal an intrusion alarm. The task of the robots is to reach the locations and possibly confirm the alarms as soon as possible. The alarms are of different types corresponding to different types of events. In particular, each class of alarms corresponds to an objective for the system. For example, there may be an objective which requires to verify fire alarms and one which requires to verify intrusion alarms.

We consider an homogeneous multi-robot system, so all the robots are capable of performing the same set of actions. Each action has a specific duration $t_{\text{action}}$. Each robot can perform four different actions: move from one position of the map to another, wait in a specific position, check for a specific alarm, and stop acting. Furthermore, each robot has a battery, so it has a limited time of operation.

The environment is represented as a graph $G = (V, E)$. Each alarm is a specific node of the graph and each type of alarm correspond to an objective of the problem, so the objectives are subsets of the vertices of the map graph. Each robot must navigate the graph avoiding collisions with other robots and checking alarms when appropriate. The state of the system can be described by the positions of the robots on the map, the elapsed execution time and the number of verified alarms. The problem is a multi-agent and multi-objective extension of a Travelers Salesman Problem. In particular, can be seen as a generalization of the Discounted Reward TSP problem [2], due to need of checking alarms as soon as possible, and of the Orienteering problem [1] due to the time limit imposed to robots. Therefore we have defined our problem as the Multi-Objective Multi Agent Discounted Reward Orienteering ($MOMA\text{-DRO}$).

**Definition 1** The $MOMA\text{-DRO}$ Problem is a tuple

$\langle G, R, S_0, U, \mathcal{O}bj, T_{\max}\rangle$

where

- $G = (V, E)$ is the topological map of the environment represented as a graph of vertices $V$ and undirected edges $E$. In particular, each vertex represents a spatial resource which can only be occupied by one robot at the time.
- $R = \{1, \ldots, n\}$ is the set of robots.
- $S_0$ is an initial description of the state of the system.
- $\mathcal{O}bj = \{o_1, \ldots, o_m\}$ is a set of objectives. Each objective is a subset of the nodes $V$ (i.e., $\forall o_i \in \mathcal{O}bj$).
- $U$ is a set of utility functions $u_i$, one for each objective $i \in \mathcal{O}bj$.
- $T_{\max} = \{T_{1\max}, \ldots, T_{n\max}\}$ are maximum times of execution, one for each robot.

The solution of the $MOMA\text{-DRO}$ is the set of Pareto Optimal abstract multi-robot plans. We define a multi-robot plan as

$p = \{p_1, \ldots, p_n\}$

where each $p_i$ is the single robot plan for agent $i \in R$. In particular, $p_i$ is a sequence of pairs $(\alpha, t)$ of actions $\alpha$ and timestamps $t$ describing when $\alpha$ was performed. Plans are evaluated, with respect to each objective, using the set of utility functions using $U$ which compute the discounted reward of checked alarms.

![Fig. 1. A simple surveillance MOMA-DRO example.](image-url)
Example 1 Figure 1 shows an example of a surveillance scenario. We consider here a portion of the map of our department which has been used, in its entirety, for the validation on our approach on real robots (see Section V). The environment is represented as a graph where the distance between nodes is fixed: each vertex has a length of 1. Some nodes of the map represent the objectives of our problem. They correspond to the alarms that shall be checked by the robots. We use two type of them: fire alarms and intrusion alarms. In the example, in Figure 1, we disposed the objectives clustered in a small zone of the map. We should refer to this disposition as a Clustered Configuration.

IV. MULTI-OBJECTIVE HEURISTIC SEARCH

In this section, we present a solving technique for the MOMA-DRO problem. From a modeling perspective, a the machinery required to implement MOA as a Clustered Configuration in a small zone of the map. We should refer to this disposition as a Clustering Configuration.

In this section, we present a solving technique for the MOMA-DRO problem. From a modeling perspective, a the machinery required to implement MOA as a Clustered Configuration in a small zone of the map. We should refer to this disposition as a Clustered Configuration.

IV. MULTI-OBJECTIVE HEURISTIC SEARCH

In this section, we present a solving technique for the MOMA-DRO problem. From a modeling perspective, a the machinery required to implement MOA as a Clustered Configuration in a small zone of the map. We should refer to this disposition as a Clustered Configuration.

In this section, we present a solving technique for the MOMA-DRO problem. From a modeling perspective, a the machinery required to implement MOA as a Clustered Configuration in a small zone of the map. We should refer to this disposition as a Clustered Configuration.

In this section, we present a solving technique for the MOMA-DRO problem. From a modeling perspective, a the machinery required to implement MOA as a Clustered Configuration in a small zone of the map. We should refer to this disposition as a Clustered Configuration.

In this section, we present a solving technique for the MOMA-DRO problem. From a modeling perspective, a the machinery required to implement MOA as a Clustered Configuration in a small zone of the map. We should refer to this disposition as a Clustered Configuration.

In this section, we present a solving technique for the MOMA-DRO problem. From a modeling perspective, a the machinery required to implement MOA as a Clustered Configuration in a small zone of the map. We should refer to this disposition as a Clustered Configuration.

In this section, we present a solving technique for the MOMA-DRO problem. From a modeling perspective, a the machinery required to implement MOA as a Clustered Configuration in a small zone of the map. We should refer to this disposition as a Clustered Configuration.

In this section, we present a solving technique for the MOMA-DRO problem. From a modeling perspective, a the machinery required to implement MOA as a Clustered Configuration in a small zone of the map. We should refer to this disposition as a Clustered Configuration.

In this section, we present a solving technique for the MOMA-DRO problem. From a modeling perspective, a the machinery required to implement MOA as a Clustered Configuration in a small zone of the map. We should refer to this disposition as a Clustered Configuration.
The initial state

Definition 6 The initial state $S_0$ models through a plan $p = ((\alpha_1, r_1), \ldots, (\alpha_k, r_k))$ a global state $S$ (i.e., $S_0 \models_p S$ or simply $\models_p S$) iff:

- $S_1 = \text{suc}(S_0, r_1, \alpha_1)$
- $\forall i \in \{1 \ldots k - 1\}, S_i = \text{suc}(S_{i-1}, r_i, \alpha_i)$
- $S = S_k = \text{suc}(S_{k-1}, r_k, \alpha_k)$

In particular, we can prove that:

Theorem 2 The state space is a DAG.

The utility of a plan is described by a tuple real values. Each element represents the utility for a specific objective. The utility takes into account, for each objective, the number of checked alarms discounted in time.

Definition 7 Given a plan $p$, an initial state $S_0$ and the global state $S$ such that $S_0 \models_p S$, the utility of a plan $p$ is a tuple:

$$(u_1(S), \ldots, u_m(S))$$

where each $u_i(S)$ represents the utility of $p$ w.r.t. objective $i$. In particular,

$$u_i(S) = \sum_{\{g, t\} \in S.c \land g \in o_i} \gamma^t$$

where $\gamma = [0 \ldots 1]$ represents the discount rate for delaying the check of a location.

B. Heuristics

As any heuristic search approach MOA* requires the definition of heuristics. A first heuristic is used to extract the set ND of Pareto-Optimal nodes from the open list of the algorithm. We refer to this heuristic as the h Heuristic. A second heuristic is used to select from ND the next node to be expanded by the algorithm. We refer to this heuristic as the ND Heuristic. In our case, to select the appropriate global state $S^* \in ND$ we have defined an heuristic that prefers solutions which have a better worse case scenario:

$$S^* = \max_{S \in NS} \min_{o \in Obj} (f_o(S))$$

The $h$ Heuristic is similar to the one used in $A^*$. As a matter of fact, we may consider the $h$ Heuristic as an extension of the heuristic in $A^*$ to the multi-objective case. In particular, the heuristic for a given global state $S$ can be computed as:

$$\forall o \in Obj \quad f_o(S) = g_o(S) + h_o(S)$$

where $g_o(S) = u_o(S)$.

The $h_o$ function is computed based on the minimum distances between the alarms and every possible position of the robots in the environment. We define $\text{mindist}(l, g)$ as the function which returns the cost (in time) of the minimum path form $l$ to $g$. The function is implemented as a look-up table pre-computed with the Floyd-Warshall algorithm on the topological representation of the environment.

Definition 8 For each objective $o \in Obj$, $h_o$ is defined as:

$$h_o(S) = \sum_{r \in R} h_o^r(s)$$

where each $h_o^r(s)$ is the contribution of the robot $r$ to the utility of objective $o$ and is defined as:

$$h_o^r(S) = \sum_{d \in \text{achieve}(r, o)} \gamma^d$$

where $\text{achieve}(r, o)$ is a set of times $d$ such that:

$$d = S_r.t + t_{i}^{(i)} + i \cdot t_{r, \text{check}}^* < T_{r, \text{max}}$$

where finally:

- $S_r.t$ is the current execution time of robot $r^*$.
- $t$ is the current time of the robot $r^*$.
- $t_{r, \text{check}}^*$ is the duration of the check action.
- $t_{i}^{(i)}$ is the $i$-th element of the ordered set $T_{r, \text{check}}^*$ which contains all the distances, calculated with $\text{mindist}(l, g)$ from the current position of $r^*$ to all the alarms in $o_i$.
- $i$ is the index of $t_{i}^{(i)}$.

Given the particular utility structure of the problem we know that any global state $S$ such that $\forall o \in Obj \quad h_o(S) = 0$ must not be expanded. In particular, for each successor global state $S$:

- IF $\forall o \in Obj \quad h_o(S) = 0 \land g_o(S) = 0$ THEN $S$ is not insert in the open list.
- IF $\forall o \in Obj \quad h_o(S) = 0 \land g_o(S) \neq 0$ THEN $S$ is not insert in the open list, but $S$ is inserted in the goal set.
- OTHERWISE $S$ is expanded as in MOA*.

In this way we can prune parts of the search space which we know will not lead to a visited goal location.

The last theorem of this section states a fundamental property of the $h$ Heuristic:

Theorem 3 The heuristic is admissible.
scheduling of the robot actions that correspond to a Pareto optimal solution of the problem defined in Figure 1. An analysis of the execution of this plan is given in the next section.

V. AN EXAMPLE ON MULTI-ROBOT SURVEILLANCE

In order to verify the proposed approach, we implemented and tested it on a multi-robot system. The plans generated with the algorithm described in the previous section (we refer to them as abstract plans) have been transformed in executable plans and actually executed by a team of mobile robots.

More specifically, we used a multi-robot plan representation formalism called Petri Net Plans (PNP) [20]. This formalism allows for defining many high-level constructs of multi-agent plans and provides for decomposition of a central multi-robot plan to actual single-robot plans that can be executed in a distributed fashion by the robots.

The implementation work that has been accomplished in order to test the proposed method comprises the following steps: 1) transformation of an abstract plan into an executable multi-robot PNP; 2) generation of single-robot PNPs from the multi-robot PNP; 3) implementation of the actions used in the plans; 4) execution and validation of the approach on both a simulated and a real multi-robot team.

A. From abstract plans to executable plans.

Plans provided by the MOA* algorithm are sequences of actions (one for each agent), that represent the solution of the problem. However, they are generated by consider actions at a high level and without taking into account the problems arising in a real time execution of the plan.

For example, Table I shows the initial part of the plan represented in Figure 2. While at a high level this plan is correct, since the robots avoid to go to the position P16 at the same time, this plan is not executable since important information are missing: when it is safe for Robot 2 to start its action to go to P16?

In order to correctly execute the abstract plans, we have added synchronization actions between the robots. Table II contains an example of solution: Robot 1 notifies Robot 2 with a direct communication that it has reached position P16 and is about to leave from there; this allows Robot 2 to move to that position without the risk of deadlocks during the execution.

By adding these synchronization actions, the multi-robot plan is equivalent to the abstract plan generated by the MOA* algorithm, but enforces precedence between actions executed by different robots and prevents possible conflicts on resources and consequent deadlocks.

More specifically, a synchronization is inserted in the plan when the plan of Robot \(i\) contains an action goto(x,y) and the plan of Robot \(j\) contains an action goto(z,y). Since the MOA* algorithm guarantees that these two actions will not be executed in parallel, a send action is inserted in the plan after the first of the two goto actions and a receive action is inserted in the plan before the second one.

B. Distributed execution of multi-robot plans

As already mentioned, the abstract plans are transformed in a multi-robot PNP containing the actions defined by the MOA* algorithm and the synchronization action added as shown before.

In order to execute this multi-robot plan on a team of robots, it is necessary to generate a set of single-robot plans that can be executed in a distributed fashion and that guarantee the correctness of the execution. We adopted a mechanism for generating single-robot plans for distributed execution of a multi-robot plan based on PNP, which is described in [20]. By applying this method we obtain a set of single-robot PNPs that can be executed on the robots and that guarantee a correct distributed execution of the multi-robot PNP and consequently of the plans generated by the MOA* algorithm.

C. Experiments and Results

The robot software has implemented and integrated by using the OpenRDK framework\(^1\) [5], which provides the basic modules for handling the mobility of robots. The system was tested both on Player/Stage simulator and both with two real robots. The quantitative results presented in this section have been obtained with the simulated system, while videos on the execution on a real multi-robot system are provided in www.dis.uniroma1.it/~ziparo/mo_surv.html.

The multi-robot system was tested in 6 situations, varying the number of robots, the number of active alarms and the difficulty of the initial situation (i.e., positions of robots and alarms in the environment), that are listed in the following table.

<table>
<thead>
<tr>
<th>Situation</th>
<th>Robots</th>
<th>Active alarms</th>
<th>Initial state</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>easy</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>difficult</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>easy</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
<td>difficult</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>4</td>
<td>easy</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>5</td>
<td>easy</td>
</tr>
</tbody>
</table>

\(^1\)openrdk.sf.net

TABLE I
PORTION OF THE ABSTRACT PLAN IN FIGURE 2

<table>
<thead>
<tr>
<th>Robot 1</th>
<th>Robot 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>goto(P17,P16)</td>
<td>wait</td>
</tr>
<tr>
<td>goto(P16,P15)</td>
<td>goto(P18,P16)</td>
</tr>
<tr>
<td>goto(P15,P12)</td>
<td>goto(P16,P15)</td>
</tr>
</tbody>
</table>

... ...

TABLE II
A SAFE PLAN

<table>
<thead>
<tr>
<th>Robot 1</th>
<th>Robot 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>goto(P17,P16)</td>
<td>send</td>
</tr>
<tr>
<td>goto(P16,P15)</td>
<td>receive</td>
</tr>
<tr>
<td>goto(P15,P12)</td>
<td>goto(P18,P16)</td>
</tr>
</tbody>
</table>

... ...
In order to validate the plans generated by MOA* algorithm, we have defined a simple greedy algorithm and compared the execution of the plans generated by the two algorithms. The greedy algorithm returns multi-agent plans that minimize with a greedy search the number of goto actions in the plan.

The results of this comparison are summarized in Figure 3. As expected, the two algorithms have different advantages: MOA* always finds an optimal solution, but it fails to compute it when the search space is too large (i.e., with 3 and 4 agents); on the other hand, the greedy algorithm always finds a solution even on large-scale problems, but it is not guaranteed to be optimal, therefore, in some cases, the execution of the greedy plans fails because of deadlocks that are not detected and solved.

![Figure 3. Average intervention time (MOA* vs. greedy).](image)

We also performed a comparison between multi-robot and single-robot approaches, using the same situations described before. For each situation, we executed a plan with only one robot and a plan with two or more robots. In these tests, we used MOA* plans when available and greedy plans otherwise. The results shown in Figure 4 confirm that using a multi-robot system allows for an effective increase of performance. This also confirms that the MOA* algorithm, and the greedy algorithm in some cases, correctly solve conflicts on resources avoiding deadlocks in an effective way.

![Figure 4. Average intervention time (single-robot vs. multi-robot).](image)

VI. CONCLUSIONS

In this paper, we introduced a multi-objective multi-robot surveillance problem. The problem has been addressed both in its theoretical and practical aspects. The problem has been formally modeled as a search problem in a state space which has been proven to be a Finite State Machine with a direct acyclic structure. We showed how the problem can be solved through heuristic search, in particular using MOA*. Moreover, we addressed the issues of executability translating abstract plans generated by MOA* in executable plans represented through Petri Net Plans. The results have been compared with a greedy algorithm and the system has been validated both in a simulated environment and on real robots. The system has been shown to be effective for small teams of robots and for a limited number of objectives. Future work will address the issue of scaling up the approach to large teams with possibly many objectives through the use of more effective domain specific heuristics.

REFERENCES