Ant Colony Optimization for Multi-UAV Minimum Time Search in Uncertain Domains[☆]

Sara Perez Carabaza[a,1], Eva Besada-Portas[a], Jose A. Lopez-Orozco[a], Jesus M. de la Cruz[a]

[a] *Departamento de Arquitectura de Computadores y Automatica*
Universidad Complutense de Madrid, Spain

Abstract

This paper presents a new approach based on Ant Colony Optimization (ACO) to determine the trajectories of a fleet of Unmanned Air Vehicles (UAVs) looking for a lost target in the minimum possible time. ACO is especially suitable for the complexity and probabilistic nature of the Minimum Time Search (MTS) problem, where a balance between the computational requirements and the quality of solutions is needed. The presented approach includes a new MTS heuristic that exploits the probability and spatial properties of the problem, allowing our ant based algorithm to quickly obtain high-quality high-level straight-segmented UAV trajectories. The potential of the algorithm is tested for different ACO parameterizations, over several search scenarios with different characteristics such as number of UAVs, or target dynamics and location distributions. The statistical comparison against other techniques previously used for MTS (ad hoc heuristics, Cross Entropy Optimization, Bayesian Optimization Algorithm and Genetic Algorithms) shows that the new approach outperforms the others.

Keywords: Ant Colony Optimization, Probabilistic Path Planning, UAVs, Minimum Time Search

1. Introduction

In Minimum Time Search (MTS) problems, there are one or several targets in unknown locations that need to be found as soon as possible. The objective of the problem consists of maximizing the probability of finding the target at the earliest, by wisely using the available information about the target locations and dynamics (provided by sources such as maps or witnesses). MTS belongs to the wider search problem family, and its distinguishing feature is the importance of time. It has many real world applications, ranging from the target location

*Corresponding author
Email addresses: sapere04@ucm.es (Sara Perez Carabaza), ebesada@ucm.es (Eva Besada-Portas), jalopez@ucm.es (Jose A. Lopez-Orozco), jmcruz@ucm.es (Jesus M. de la Cruz)

Figure 1: Two UAVs searching for survivors after an earthquake, where time is a critical factor.

problems of the Second World War to current search and rescue operations, wildlife monitoring or tactical reconnaissance [1, 2, 3, 4]. An example application, schematized in Fig. 1, is searching for survivors after natural disasters, where the time of search is critical to find them alive.

In our case, the search is carried out by one or several Unmanned Aerial Vehicles (UAVs) that have sensors capable of providing detection measurements about the target. UAVs are used because their cooperation can significantly improve the efficiency of the search operation, as they are a great advantage in scenarios with abrupt terrain or that imply certain dangers.

A probabilistic approach to the search problem enables to handle the uncertainty sources of the problem and return optimized UAVs search trajectories. As a consequence of the high complexity of the search problem (NP-hard, [5]), and in order to find high-quality high-level straight-segmented UAV trajectories in an appropriated time, the problem is often simplified and tackled with ad hoc heuristics [6] or generic approximated optimization methods such as Cross Entropy Optimization (CEO, [7]), Bayesian Optimization Algorithms (BOA, [8]) and Genetic Algorithms (GA, [9]). On one hand, the ad hoc heuristics are especially designed to construct high-level UAV trajectories that exploit the intrinsic properties of search problems. On the other one, the generic optimization methods achieve overall good UAVs trajectories by 1) sampling the distribution learned from promising solutions identified, evaluating them with an objective function, in previous iterations of the approach (CEO and BOA) or by 2) combining them (GA). Hence, the existing approaches for tackling MTS exploit problem specific information *either directly* (using specific heuristics) *or indirectly* (by means of the objective function).

For this reason, this work presents a new approach based on Ant Colony Optimization (ACO, [10]), a technique that *naturally combines* the learning capabilities of the pheromone trails left by overall good UAV trajectories (according to the objective function) and the benefits of a problem specific heuristic. Besides, ACO has already been used to tackle successfully other high-level straight-segmented UAV trajectory optimization problems (e.g. to minimize

the threat exposure and travelled distance to a final point [11], and the distance required to cover a given area [12]) or the MTS problem with low-level smooth-curved UAV trajectories [13].

Our new approach uses a new heuristic function especially designed for MTS that enables ACO to initially generate high-quality solutions and accelerates the convergence of the algorithm. This is a great benefit in our problem, where a balance between the quality of solutions and the computational cost is needed. Besides, two different encodings are analyzed: one closer to the original ACO [10] and the other to the codification used by other search algorithms [7, 8, 14, 15, 16, 17]. Finally, in order to show the capabilities of the new approach, the performance of both codifications is analyzed over different search scenarios and statistically compared against several existing MTS approaches that generate solutions *either* exploiting problem specific properties (ad hoc heuristics) or employing the information generated/learned from previous iterations of the optimization method (CEO, BOA and GA). By contrast, our new algorithm combines the benefits of the heuristic specifically introduced in this paper for the MTS problem and of its supporting optimizer (ACO).

In summary, the contributions of this paper are: solving the MTS problem using ant colony optimization techniques to obtain high-level trajectories for the searching UAVs and the proposal of a new heuristic for MTS that rates the possible actions of the UAVs based on their locations and on the probability information about the target location.

This paper is organized as follows. In Section 2 the formulation of the MTS problem is introduced. Next, in Section 3 the state of the art of the closest related works is analyzed. Then, in Section 4 the proposed MTS algorithm based on ACO is presented. Later, in Section 5 its performance with different parameterizations over several search scenarios is analyzed and compared to other approaches previously used for MTS. And finally, in Section 6 the main conclusion of the work and some open research lines are discussed.

2. Formulation of the MTS Problem

This section introduces the notation and probability models of the search problem, presents the objective function proposed in [8] for MTS, and illustrates its use over a simplified scenario.

2.1. Notation and Preliminaries

In MTS, the exact target location is unknown, but it can be represented with a random variable ν^t. The search problem starts with the assumption of a probability distribution function that models the target location information available before the search starts. This probability function can be represented with a grid-based probability map (belief map, $b(\nu^0)$), where each cell corresponds to a discretization of the search area and has an associated probability that the target is present in it [6, 7, 8, 9, 18]. In order to construct the initial probability map, the search area is discretized into a grid Ω of $w_x * w_y$ cells

(i.e. $\Omega = 1 : w_x * w_y$), and a probability of target presence is given to each cell (based on the available information, obtained from sources such as maps, witnesses or intelligence). As it is assumed that the target is inside the search area, the initial probability map satisfies $\sum_{\nu^0 \in \Omega} b(\nu^0) = 1$.

A bayesian approach is typically used to keep track of the target probability function as time passes and the sensors of a fleet of U different UAVs make observations of the search region. The bayesian approach allows to start from the prior target belief $b(\nu^0)$ and to update it with the sensor measurements z_u^t (obtained at time t by the u-th UAV) through a sensor model $P(z_u^t = D|\nu^t, s_u^t)$, which gives the probability of target detection ($z_u^t = D$) conditioned by the target and the UAV positions (ν^t and s_u^t). Besides, by considering only target detection and no detection measurements (i.e. $z_u^t = D$ and $z_u^t = \overline{D}$), $P(z_u^t = \overline{D}|\nu^t, s_u^t) = 1 - P(z_u^t = D|\nu^t, s_u^t)$. In addition, in the case that the target is not static, the belief is also updated considering the target dynamic information through a target motion model $P(\nu^t|\nu^{t-1})$, which represents the probability that the target moves from one cell ν^{t-1} to another ν^t. This method, called Recursive Bayesian Estimation (RBE, [19]), iterates through two steps and updates the belief with the probabilistic information about the target motion (prediction step, Eq. (1)) and the sensor measurements (update step, Eq. (2)):

$$\hat{b}(\nu^t) = \sum_{\nu^{t-1} \in \Omega} P(\nu^t|\nu^{t-1}) b(\nu^{t-1}) \tag{1}$$

$$b(\nu^t) = \frac{1}{\xi} \prod_{u=1:U} P(z_u^t|\nu^t, s_u^t) \hat{b}(\nu^t) \tag{2}$$

where ξ is just a normalization factor used to ensure that the target presence belief inside the area is always one (i.e. $\sum_{\nu^t \in \Omega} b(\nu^t) = 1$). Therefore, with the use of the sensor and target dynamic models, RBE forecasts and updates the probability map with the measurements $z_{1:U}^{1:t}$ obtained by the UAVs while following the search trajectories $s_{1:U}^{0:t}$.

2.2. MTS Objective Function

The search problem can be formulated as an optimization problem whose solutions are the UAVs search trajectories. One of the most commonly used utility functions to evaluate the search trajectories of the UAVs is to maximize the cumulative probability of finding the target (in short, the probability of target detection, P_d). However, maximizing the chances of finding the target does not guarantee to minimize the time to localize it [8]. Hence, in the MTS problem, which can be considered as a subproblem of the search problem family where time is critical, the expected time of detection (ET, [7, 8, 20]) is usually optimized. The ET can be computed, using Eq. (3), adding up to infinity the probability $P(\overline{D}_{1:U}^{1:t}|s_{1:U}^{0:t})$ of no detecting the target up to each time step.

$$ET(s_{1:U}^{0:\infty}) = \sum_{t=1}^{\infty} P(\overline{D}_{1:U}^{1:t}|s_{1:U}^{0:t}) \tag{3}$$

The probability of no having detected yet the target at time t ($P(\overline{D}_{1:U}^{1:t}|s_{1:U}^{0:t}) \triangleq 1 - P(\cup_{t=1:N,u=1:U} D_u^t|s_{1:U}^{0:t}) = 1 - P_d(s_{1:U}^{0:t})$), can be expressed, according to [8] and as Eq. (4) states, in terms of the unnormalized probability map $\tilde{b}(\nu^t)$. This last function is computed, according to Eqs. (5) and (6), combining two steps similar to those of the RBE (Eqs. (1) and (2)), but: 1) without a normalization factor and 2) assuming no target detection (i.e. with $z_u^t = \overline{D}$). For the initial case (at $t = 0$) of the recursive process stated by Eqs. (5) and (6), $\tilde{b}(\nu^0) = b(\nu^0)$.

$$P(\overline{D}_{1:U}^{1:t}|s_{1:U}^{0:t}) = \sum_{\nu^t \in \Omega} \tilde{b}(\nu^t) \tag{4}$$

$$\bar{b}(\nu^t) = \sum_{\nu^{t-1} \in \Omega} P(\nu^t|\nu^{t-1})\tilde{b}(\nu^{t-1}) \tag{5}$$

$$\tilde{b}(\nu^t) = \prod_{u=1:U} P(z_u^t = \overline{D}|\nu^t, s_u^t)\bar{b}(\nu^t) \tag{6}$$

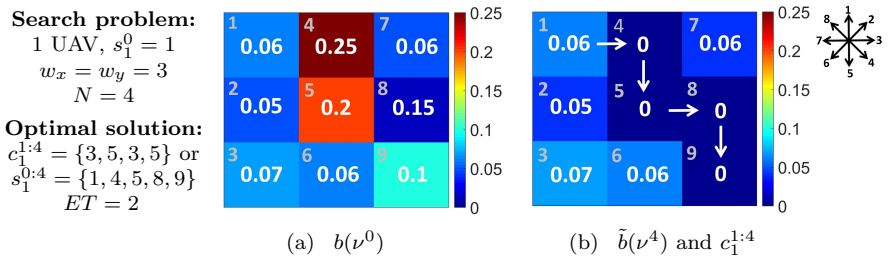
Note that the values of the summation terms for calculating ET in Eq. (3) are bounded ($0 \leq P(\overline{D}_{1:U}^{1:t}|s_{1:U}^{0:t}) \leq 1$) and decrease as t grows ($P(\overline{D}_{1:U}^{1:t}|s_{1:U}^{0:t}) \leq P(\overline{D}_{1:U}^{1:t-1}|s_{1:U}^{0:t-1})$). Besides, they reach a null value at the first time step where $P(\overline{D}_{1:U}^{1:t}|s_{1:U}^{0:t}) = 0$. This happens because Eq. (6) lacks of normalization term and because the values of the unnormalized probability map $\tilde{b}(\nu^t)$ decrease in the cells ν^t under each UAV location s_u^t . Moreover, $P(\overline{D}_{1:U}^{1:t}|s_{1:U}^{0:t}) \approx 0$ when the sensors have gathered almost all the probability along $s_{1:U}^{0:t}$, and therefore the remaining probability of no having detected the target yet becomes negligible.

Nevertheless, the rapidly growth of number of possible solutions of the search problem as the length of the trajectories increases derives in the optimization of trajectories of limited horizon N. This approach is taken in several search probability problems (e.g. [7] or [9]), and in the case of minimizing the ET, the truncated version presented in Eq. (7) is often used ([7, 8, 18]) due to the decreasing behavior of its summation terms. From now on, we would indifferently refer to ET or its truncated version.

$$ET(s_{1:U}^{0:N}) = \sum_{t=1}^{N} P(\overline{D}_{1:U}^{1:t}|s_{1:U}^{0:t}) \tag{7}$$

Finally, the search trajectories can be determined either by the sequence of visited cells $s_{1:U}^{0:N}$, or by the corresponding sequence of actions $c_{1:U}^{1:N}$ and the UAV initial positions $s_{1:U}^{0}$. Moreover, this work considers that the UAV moves from one cell to its neighbors following one of the 8 cardinal directions, allowing ACO to determine UAV straight-segmented trajectories based on sequences of high-level control actions (as in [6, 7, 8]).

Summarizing, the problem at hand consists in minimizing the expected target detection time of trajectories of fixed length (defined by a sequence of N high-level control actions) of a fleet of U UAVs. The ET of the possible trajectories (in our case, codified by the discrete values of the 8 cardinal directions) is



(a) $b(\nu^0)$ (b) $\tilde{b}(\nu^4)$ and $c_1^{1:4}$

Figure 2: Example search scenario. a) Initial probability map. b) MTS optimal solution.

obtained adding up the unnormalized belief at each time instant (Eqs. (7) and (4)), and the unnormalized belief is updated considering the target probability motion model and UAV no detection probability function with the recursive process of Eqs. (5) and (6).

2.3. Illustrative Example

To illustrate and evaluate our approach we use, as in [7, 8, 9], the ideal sensor model described by Eq. (8), which considers a detection probability equal to one when the UAV and the target are over the same cell and equal to zero otherwise. Hence, $P(z_u^t = \overline{D}|\nu^t, s_u^t) = 0$ in the cell ν^t overflown by the UAV, and $P(z_u^t = \overline{D}|\nu^t, s_u^t) = 1$ otherwise. However, more complex sensor models like the radar in [4] or the model with false negatives in [16] could also be considered.

$$P(z_u^t = D|\nu^t, s_u^t) = \begin{cases} 1 & \nu^t = s_u^t \\ 0 & \nu^t \neq s_u^t \end{cases} \tag{8}$$

The selected example is shown in Fig. 2, where the search area is discretized into a $3 * 3$ grid and one UAV starts the search at s_1^0 from cell 1 (the cell indexes are displayed on the upper left corner of each cell). Fig. 2(a) shows the initial belief $b(\nu^0)$, whose values, written in the centre of the cell and indicated with a colormap (where warmer colors are associated to higher target presence probabilities and colder colors to lower probabilities), add up to 1 ($\sum b(\nu^0) = 1$). Figure 2(b) shows with white arrows the optimal trajectory of length $N = 4$ for a unique UAV that carries an ideal sensor and its corresponding final unnormalized probability map $\tilde{b}(\nu^4)$. Finally, the compass rose at the right shows the actions encoding used for the eight cardinal directions.

The UAV trajectory can either be described as the sequence of cells $s_1^{0:4} = \{1, 4, 5, 8, 9\}$ or by the initial position $s_1^0 = 1$ and cardinal actions $c_1^{1:4} = \{\text{east, south, east, south}\} = \{3, 5, 3, 5\}$. As previously explained, the expected time of a trajectory is computed adding (with Eqs. (7) and (4)) the remaining unnormalized belief at each time instant, which is updated (with Eqs. (5) and

(6)) using the target dynamic and sensor probability models. In this example, after the first no detection measurement at $t = 1$, the unnormalized belief at cell $i = 4$ (under the UAV s_1^1 location) it set to zero, so $\sum_{\nu^1 \in \Omega} \tilde{b}(\nu^1) = 0.75$. Next, as the target is static, the unnormalized belief is not redistributed, and the following measurement at $t = 2$ makes $\tilde{b}(\nu^2)$ at cell $i = 5$ equal to zero, and $\sum_{\nu^2 \in \Omega} \tilde{b}(\nu^2) = 0.55$. Following the same procedure, in the third time step $\tilde{b}(\nu^3 = 8) = 0$ and $\sum_{\nu^3 \in \Omega} \tilde{b}(\nu^3) = 0.4$, and in the fourth time step $\tilde{b}(\nu^4 = 9) = 0$ and $\sum_{\nu^4 \in \Omega} \tilde{b}(\nu^4) = 0.3$. For these reasons, the final unnormalized belief of the overflown cells has became zero and the expected detection time for the optimal trajectory is $ET(s_1^{0:4}) = 0.75 + 0.55 + 0.4 + 0.3 = 2$.

It is worth noting that for this simple example, the number of possible solutions is low and therefore the optimal solution can be easily computed. However, neglecting the forbidden actions that lead the UAVs out of the search area, the number of possible solutions/trajectories increases exponentially with the number of actions to optimize as 8^{N*U}. Moreover, as the number of cells in the scenario $(w_x * w_y)$ grows, the computational time required in the belief update process also increases. Consequently, for more complex scenarios, the calculation of all possible trajectories in order to choose the optimal one (the one with the lowest ET) becomes intractable, and heuristics/optimization algorithms for discrete decision variables (e.g. actions to perform or sequence of visited cells) are generally used.

3. Related Work

In this section we discuss different existing approaches and formulations of the search problem. This state of art is not exhaustive, as it is a compilation of the works that have motivated this paper. Therefore, we focus on search works that consider high-level UAV trajectories specified with discrete decision variables as a sequence of adjacent cells or as a sequence of the actions defined by the cardinal directions, leaving out other works such as [18, 19, 20] that do not share those properties. Besides, the selection ranges from the historical background of our work to its updated counterparts.

The comparison in this section shows the main differences among our work and others, taking into account five properties, explained below and represented in the columns of Table 1, where works are sorted by publishing date.

Objective Function indicates the criterium optimized, which directly depends on the specific search problem tackled by each work. We are specially interested in MTS approaches, highlighted by the gray background on the second column of the table. Within this group, the algorithms in [7, 8] and ours optimize the expected value of the required time to find the target (ET), while [6] evaluates the solutions by computing the mean of the finding times (Mean Time To Find, MTTF) obtained in several Monte Carlo simulations that start with a different initial target position sampled from the initial belief. A different search strategy consists on maximizing the probability of target detection (P_d, [9, 14, 23]). However, maximizing P_d does not imply to find the target in

Table 1: Search works comparison table.

Work	Obj. Function	Multi-Agent	Moving Target	Optimization Method	Optimality	Ad hoc Heur.
[14]	P_d		✓	POMDP-DP	Global(N)	
[15]	P_d+Cov.	✓		Q-learning	Local(2)+Heur(N)	
[9]	P_d			GA/Greedy/LHC	Local(N)	
[7]	ET/DTR	✓	✓	CEO	Local(N)	
[16]	P_d	✓		MIQP	Local Approx.(N)	
[8]	ET/DTR	✓	✓	BOA	Local(N)	
[21]	TSP for $\bar{b}(\nu^t) > threshold$	✓		NN/GA	Local Approx.(N)	
[22]	P_d + Entropy + Coop.	✓	✓	MPC	Local Approx.(N)	
[17]	P_d	✓		MIQP	Local Approx.(N)	
[6]	MTTF				Local Approx.	✓
[23]	P_d		✓	BILP	Local Approx.(N)	
This work	ET	✓	✓	ACO	Local(N)	✓

minimum time, as this objective function does not take into account the time order of the measurements [7, 17]. For this reason, the intermediate alternative approach in [7, 8] temporary discounts the probability of detection (Discounted Time Reward, DTR), giving more importance to the probability collected by the earlier measurements. Finally, the strategy in [15] maximizes a linear combination of the probability of detection (P_d) and a Coverage (Cov.) criterium; the method in [22] optimizes a linear combination of P_d, the entropy of the environment and a fleet Cooperation (Coop.) criterium; and the approach in [21] set it up as a Traveling Salesman Problem (TSP) among the cells of the searching region where the unnormalized belief $\bar{b}(\nu^t)$ is bigger than a given threshold.

Multi-Agent points out whether the search algorithm is applicable for several searchers [7, 8, 15, 16, 17, 21, 22] or for a unique UAV [9, 6, 14, 23]. In general, a multi-agent approach allows the algorithm to benefit from a greater capability to overfly bigger areas in less time, and therefore it usually provides more efficient solutions.

Moving Target denotes if the algorithm can consider the target dynamics. The algorithms in [7, 8, 14, 23, 22] and this work assume a Markovian target motion model (where the current target state is only dependent on the previous state), while the approaches in [9, 6, 15, 16, 17, 21] search for a static target.

Optimization Method shows the variability in the techniques used to solve the search problem. Due to the high complexity of the problem, it is commonly solved with approximated methods such as Genetic Algorithms (GA, [9, 21]), Cross Entropy Optimization (CE0, [7]), Bayesian Optimization Algorithm (BOA, [8]), Reinforcement Learning Techniques (Q-learning, [15]), Greedy

Heuristics [9], or Nearest Neighbors (NN, [21]). Alternatively, [14] implements a Partially Observable Markov Decision Process (POMDP) Dynamic Programming (DP) solution, while [16, 17] formulate the problem as a Mixed-Integer Linear and Quadratic Programming (MIQP), [22] as a Model Predictive Controller (MPC), and [23] as a Binary Integer Linear Programming (BILP). Finally, [6] does not apply any optimization method, but it evaluates (via Monte Carlo simulations) the expected time of detection from the UAV trajectories obtained with several deterministic techniques (e.g. following spiral trajectories) and greedy heuristics (e.g. moving towards the cell of maximum probability) designed ad hoc for search problems with static targets and a unique UAV.

Optimality (Horizon) indicates if the solution is local or globally optimal, and the optimization horizon N. In this regard, it is worth noting that the approach in [14] is the unique that assures the global optimum in the horizon window, but it is only applicable to small search scenarios due to its high computational time. The majority of the solutions, not globally optimal in the horizon window, are classified as locally optimal (Local, as a consequence of the nature of the algorithms [7, 8, 9, 15]) or as locally approximated (Locally Approx., due to the approximations performed in the objetive function [16, 17, 21, 22, 23] or used to construct the solutions [6]). Finally, [15] is local optimal for horizons of two steps, but the evaluation of the solutions includes a heuristic that considers the following N time steps, with the purpose of improving the solutions of the algorithm.

Ad hoc Heuristic points out if a heuristic specially designed for the search problem is used to *build* the solutions. While most of the studied works [7, 8, 9, 14, 15, 16, 17, 21, 22, 23] use well known techniques in order to optimize a search related objective function, they do not consider any constructive heuristic related with the problem during the generation of their solutions. On the contrary, the ant-colony-based algorithm employed in this work considers the information provided by a new MTS heuristic function in the generation process of its solutions. Finally, [6] is classified as an approach with an ad hoc constructive heuristic because it contains a compendium of methods (e.g. spiral trajectories) specially suitable for search scenarios, while the heuristic used in [15] is not considered within this category as it is not used for generating the algorithm solutions, but as a part of the optimization objective function.

In short, our method is a new efficient suboptimal approach especially targeted for MTS problems with moving targets and searching UAVs following straight-segmented trajectories, which benefits from the use, within the ACO framework, of a new heuristic designed ad hoc for MTS.

Finally, it is worth noting that a MTS planner supported by ACO-R ([24], an ant colony optimizer for problems with *Real-coded* decision variables, which does not have the heuristic support of the approach used in this paper) has been recently published for optimizing the detection time of a fleet of UAV following smooth-curved trajectories [13]. That method can be used for obtaining low-level UAV trajectories at the expenses of a higher computational time (required to perform, in the evaluation step of the objective function, the realistic simulations that generate the UAV trajectories given the low-level real-coded

actions). Hence, the alternative presented in this paper is targeted to quickly obtain high-level trajectories for UAVs capable of following straight-segmented trajectories (e.g. quadrotors). Besides, these types of trajectories can undergo a smoothing second-optimization step to make them flyable by other types of UAVs.

4. Ant Colony Optimization for Minimum Time Search

In this section, we introduce the main characteristics of our new MTS algorithm based on ACO, explain its two different encodings, detail the methods used to construct the pheromone table, present the heuristic function that we have developed for the MTS problem, describe the process followed to build new solutions, and come out with the pseudo-code of the MTS ACO-based planner.

4.1. Introduction to Ant Colony Algorithms.

ACO is inspired by the foraging activity of natural ants, capable of finding through pheromone deposit the shortest path between a food source and their nest. It is an iterative algorithm that uses a set of artificial ants to construct, in each generation, M solutions combining the information of a problem specific heuristic with the information learned from previous iterations, saved in a pheromone table. When certain stop condition is reached, ACO returns the best solution (ant trajectory) found so far.

The first ant colony based algorithm (Ant System, [10]) was originally applied to solve the Traveling Salesman Problem (TSP): finding the shortest closed loop that traverses once a group of cities. Since then, different variants of ant algorithms have successfully been used to solve a variety of problems, such as quadratic assignment [25], generating test data for software [26], or several path planning problems with fixed initial and final destinations such as optimizing the trajectory length [12] or the threat exposure [11]. In this regard, it is worth noting that although the two last examples are close to our problem, in MTS only the initial UAVs locations are fixed and the objective function is a probabilistic-based function.

Two more recent popular variants of ACO, Max-Min Ant System (MMAS, [27]) and Ant Colony System (ACS, [28]), maintain the basic idea of the original Ant System, but implement different methods to update the pheromones with the purpose of avoiding stagnation (i.e. that all the ants select the same path and stop exploring the search space). We use MMAS to solve our problem because we can benefit from a generation in parallel of the ants tours, which is not possible with the ACS variant because it has a local pheromone update rule that changes the pheromones after each ant step. Besides, in MMAS, maximum and minimum pheromones bounds are set with the purpose of limiting the relative difference between the pheromones trails, in order to avoid stagnation.

4.2. Solving MTS with MMAS.

In addition to the design of a MTS specific heuristic, there are other aspects to take into account when solving MTS with ACO, which differentiate our MTS-ACO algorithm from the original TSP-ACO implementation. First, while in the TSP implementation the initial positions are set at random, in MTS all ants start from the UAVs initial positions. This happens because in TSP the tours are circular and the initial position is not important, while in MTS we want to obtain the search trajectories that commence at the UAVs initial positions. This is a common characteristic of the path planning problems, where the agents initial positions are also given [11, 12]. Second, while in TSP all cities are connected, in MTS the nodes (map cells) are connected according with a grid and each UAV/ant can only move from its current node to its neighbors. Therefore, in each step of an ant tour we only have to compute the transition rule for the connections allowed by the eight possible actions.

In addition, we propose and analyze two different encodings of the MTS-ACO algorithm: ACO-NODE and ACO-TIME. In ACO-NODE encoding, the pheromones learn the best actions to perform at each node, while in ACO-TIME the pheromones learn the best actions to perform at each time step. Therefore, the first type of encoding is closer to the TSP node-to-node codification, while the second is closer to earlier estimation distribution algorithms implementations for MTS such as CEO [7] or BOA [8].

4.2.1. Pheromones.

The pheromone deposit is a positive feedback mechanism that enables ants to learn the best tours from the trails followed by previous ants.

In ACO-NODE, the pheromone table learns the best actions to perform at each node by each UAV, so $\boldsymbol{\tau}_{\text{NODE}}$ (the pheromone table of ACO-NODE) is a 3D matrix of size $8 * (wx * wy) * U$, whose elements $\tau_{\text{NODE}}[a, i, u]$ are distributed in rows that correspond to each action (a), columns to each cell of the map (i), and depth to each UAV (u). In ACO-TIME, the pheromone table learns the best actions to perform at each time step by each UAV, so $\boldsymbol{\tau}_{\text{TIME}}$ (the pheromone table of ACO-TIME) is a 3D matrix of size $8 * N * U$, whose elements $\tau_{\text{TIME}}[a, t, u]$ are distributed in rows that correspond to each action (a), columns to each time step (t), and depth to each UAV (u).

The pheromone update process of MMAS takes place at the end of each algorithm iteration, and consists of a pheromone reinforcement step (Eq. (9) for ACO-NODE and Eq. (11) for ACO-TIME), a pheromone evaporation process (Eq. (10) for ACO-NODE and Eq. (12) for ACO-TIME), and a bounding round.

• ACO-NODE:

$$\tau_{\text{NODE}}[{}^{*}c_u^t, {}^{*}s_u^{t-1}, u] \leftarrow \tau_{\text{NODE}}[{}^{*}c_u^t, {}^{*}s_u^{t-1}, u] + \frac{1}{ET({}^{*}s_{1:U}^{0:N})} \tag{9}$$

$$\tau_{\text{NODE}}[a, i, u] \leftarrow (1 - \rho) \cdot \tau_{\text{NODE}}[a, i, u] \tag{10}$$

• ACO-TIME:

$$\tau_{\text{TIME}}[{}^{*}c_u^t, t, u] \leftarrow \tau_{\text{TIME}}[{}^{*}c_u^t, t, u] + \frac{1}{ET({}^{*}s_{1:U}^{0:N})} \tag{11}$$

$$\tau_{\text{TIME}}[a, t, u] \leftarrow (1-\rho) \cdot \tau_{\text{TIME}}[a, t, u] \tag{12}$$

where ${}^{*}c_u^t$ and ${}^{*}s_u^t$ stand for the action and location (of the u-th UAV at time t) of the best solution of the current algorithm iteration, and ρ is the evaporation rate parameter. Hence, on one hand, Eq. (9) shows that the pheromone values corresponding to the best solution of an iteration (${}^{*}s_{1:U}^{0:N}$) are intensified by a factor inversely proportional to its expected time, i.e. higher reinforced as better (lower) ET. Therefore, in ACO-NODE the corresponding actions ${}^{*}c_u^t$ applied at ${}^{*}s_{1:U}^{0:N}$ are intensified. On the other hand, Eq. (11) of ACO-TIME shows how the corresponding actions ${}^{*}c_u^t$ of each time-step ($\forall t = 1 : N$) are intensified. Besides, in contrast to the pheromone reinforcement that only depends on the best solution, the pheromone evaporation with $\rho \in (0, 1)$ in Eqs. (10) and (12) is applied to the whole pheromone matrix. Finally, although MMAS imposes pheromone bounds $[\tau_{min}, \tau_{max}]$ to all the pheromones in order to avoid stagnation, we set some null values in the pheromone table of ACO-NODE to avoid the ants from choosing the forbidden actions that lead the UAVs outside the search zone.

To show how the best trajectories are learned, the pheromone tables of the example of Fig. 2 obtained after 2 iterations of both ACO variants are represented below. The pheromone elements that correspond to the optimal solution are highlighted: in ACO-TIME the emphasized rows in each column of $\boldsymbol{\tau}_{\text{TIME}}$ directly encode the optimal sequence of actions (${}^{*}c_1^{1:4} = \{3, 5, 3, 5\}$), while in ACO-NODE, the emphasized elements encode which action (row) should be taken in each node (column) of the trajectory ${}^{*}s_1^{0:3} = \{1, 4, 5, 8\}$. The high values of the highlighted elements indicate how the optimal path has already been learned for this simple scenario.

$$\boldsymbol{\tau}_{\text{NODE}} = \begin{array}{c} \text{nodes} \\ \left[\begin{array}{ccccccccc} 0 & 0.48 & 0.48 & 0 & 0.48 & 0.48 & 0 & 0.48 & 0.48 \\ 0 & 0.48 & 0.48 & 0 & 0.48 & 0.48 & 0 & 0 & 0 \\ \boxed{0.73} & 0.48 & 0.48 & 0.48 & \boxed{0.73} & 0.48 & 0 & 0 & 0 \\ 0.27 & 0.01 & 0 & 0.57 & 0.32 & 0 & 0 & 0 & 0 \\ 0.48 & 0.48 & 0 & \boxed{0.73} & 0.48 & 0 & 0.48 & \boxed{0.73} & 0 \\ 0 & 0 & 0 & 0.48 & 0.48 & 0 & 0.48 & 0.48 & 0 \\ 0 & 0 & 0 & 0.48 & 0.48 & 0.48 & 0.48 & 0.48 & 0.48 \\ 0 & 0 & 0 & 0 & 0.48 & 0.48 & 0 & 0.48 & 0.48 \end{array}\right] \end{array} \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \text{actions} \end{array}$$

$$\boldsymbol{\tau}_{\text{TIME}} = \begin{array}{c} \text{time} \\ \begin{bmatrix} 0.48 & 0.48 & 0.48 & 0.48 \\ 0.48 & 0.48 & 0.48 & 0.48 \\ \boxed{0.73} & 0.48 & \boxed{0.73} & 0.48 \\ 0.48 & 0.48 & 0.48 & 0.48 \\ 0.48 & \boxed{0.73} & 0.48 & \boxed{0.73} \\ 0.48 & 0.48 & 0.48 & 0.48 \\ 0.48 & 0.48 & 0.48 & 0.48 \\ 0.48 & 0.48 & 0.48 & 0.48 \end{bmatrix} \text{actions} \end{array}$$

Both codifications have advantages and disadvantages. On one hand, in each ant step of ACO-TIME, the actions that take the ants outside the search region have to be eliminated, while in ACO-NODE we ensure that the trajectories are always valid by fixing to 0 the pheromones corresponding to the actions that will lead the ants outside the search region. On the other hand, as the number of actions to optimize N is usually less than the total number of cells, the pheromone table in ACO-TIME is more compact and requires less memory.

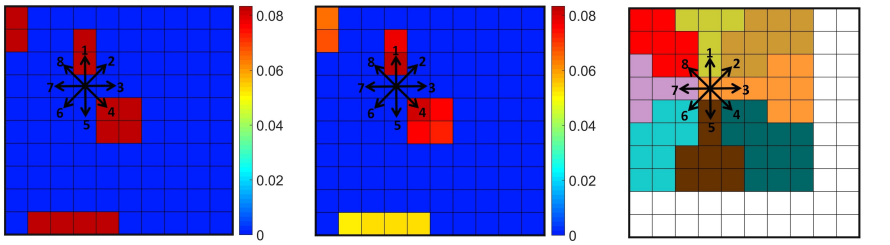
4.2.2. Heuristic.

For generating new solutions, ACO combines the information learned in previous iterations with the information given by a problem specific heuristic.

The heuristic we propose for MTS is a spatial function that depends on the current position s_u^t of the ant/UAV and on the current predicted unnormalized belief $\bar{b}(\nu^t)$. The calculation of the heuristic value for action a of UAV u located at node i at time t, given by Eq. (13), can be divided in two steps. First, the current predicted unnormalized belief $\bar{b}(\nu^t)$ is linearly weighted taking into account the distance from the cells of the map j to the current UAV/ant at cell i, giving higher values to the cells that are closer to the UAV. Second, the heuristic value $\eta(a, i, t)$ associated to action a of the UAV at cell i at time t is obtained adding up the values of the weighted unnormalized belief contained in the associated $triangle(a, i, l)$. This triangle is defined by the perpendicular bisector associated to action a starting at cell i and by length $l = N - t$. In this way, the heuristic function gives higher values to the actions that point towards the highest and closer probability areas. Moreover, it considers only the cells that are reachable from the UAV, that is, the cells that are not further that $N - t$.

$$\eta(a, i, t) = \sum_{j \in triangle(a,i,N-t)} f(distance(i, j))\bar{b}(\nu^t = j) \tag{13}$$

To better explain the heuristic we use the search scenario of $10 * 10$ cells of Fig. 3(a), where $\bar{b}(\nu^t)$ has four high probability areas and the UAV location $s_u^t = i$ is in the center of the arrows that represent the eight possible actions. Fig. 3(b) shows the distance weighted belief and Fig. 3(c) the associated triangles for each of the eight possible actions. After adding up the weighted probabilities of the cells $j \in triangle(a, i, N - t)$, the highest heuristic value is given to action



(a) $\bar{b}(\nu^t)$ (b) Distance weighted $\bar{b}(\nu^t)$ (c) $triangle(a, s_u^t, N-t=4)$

Figure 3: Heuristic sketch.

4, followed by action 1 and action 8. That is, $\eta(4, s_u^t, t) > \eta(1, s_u^t, t)$ because the triangle of action 4 includes the highest probability zone, and $\eta(1, s_u^t, t) > \eta(8, s_u^t, t)$ because the probability area included in the triangle of action 1 is closer to the UAV position, and thus it has a higher weight. Finally, the heuristic values for the rest of the actions are equal and the probability area at the bottom of the map is out of reach for the UAV, and thus it is not taken into account in any of the heuristic values.

4.2.3. Solutions construction.

The M artificial ants of an ACO iteration obtain for each UAV their action sequence ($c_u^{1:N}$) combining the information from the heuristic and from the pheromones.

To do it, the actions are sampled according to the probability rule in Eq. (14) for ACO-NODE+H (ACO with NODE encoding plus Heuristic) and in Eq. (15) for ACO-TIME+H (ACO with TIME encoding plus Heuristic), which return the probability that the action a should be chosen at time step t for UAV u. Note that the difference between both equations is due to the fact that in ACO-NODE+H the pheromone table $\boldsymbol{\tau}_{\text{NODE}}$ contains the previous learned information about the best actions to perform at each node, and in ACO-TIME+H the pheromone table $\boldsymbol{\tau}_{\text{TIME}}$ contains the information learned about the best actions to perform at each time step.

• ACO-NODE+H :

$$p(a,t,u) = \frac{(\tau_{\text{NODE}}[a, s_u^t, u])^{\,\alpha}\,(\eta(a, s_u^t, t))^{\beta}}{\sum_{a=1:8}(\tau_{\text{NODE}}[a, s_u^t, u])^{\,\alpha}\,(\eta(a, s_u^t, t))^{\beta}} \tag{14}$$

• ACO-TIME+H :

$$p(a,t,u) = \frac{(\tau_{\text{TIME}}[a, t, u])^{\,\alpha}\,(\eta(a, s_u^t, t))^{\beta}}{\sum_{a=1:8}(\tau_{\text{TIME}}[a, t, u])^{\,\alpha}\,(\eta(a, s_u^t, t))^{\beta}} \tag{15}$$

Eqs. (14) and (15) state that actions with higher pheromones and heuristic values are more likely to be chosen by the ants. Moreover, their parameters α

and β control the relationship between the pheromone and heuristic influence. Finally, by making $\beta = 0$, we can disable the heuristic to create two variants of ACO without heuristic (ACO-NODE and ACO-TIME), which only use the pheromones trails to determine the best UAVs trajectories for MTS. And by making $\alpha = 0$, we can disable the pheromones effects of both encodings to create one variant of ACO without pheromones (ACO-H), which only uses the heuristic to determine the best UAVs trajectories for MTS.

4.3. MTS ACO-based Planner

This section introduces our planner in an algorithmic form (detailing the pseudo-code of ACO-NODE+H and highlighting the modification required for ACO-TIME+H) and analyzes their computational complexity.

4.3.1. MTS ACO-NODE+H Planner

The inputs, steps, and outputs of the proposed MTS planner for the node encoding with heuristic are presented in Algorithm 1, where we extend the notation previously used in the paper with the following elements. On one hand, bold subindexed variables ($\boldsymbol{s}_m, \boldsymbol{c}_m, \boldsymbol{ET}_m$) respectively stand for the whole trajectory, actions sequence and ET obtained by each ant m in an iteration of the algorithm. On the other one, bold superindexed variables (${}^{ib}\boldsymbol{s}$,${}^{ib}\boldsymbol{c}$,${}^{ib}\boldsymbol{ET}$,${}^{gb}\boldsymbol{s}$,${}^{gb}\boldsymbol{c}$,${}^{gb}\boldsymbol{ET}$) represent the information of the best solution of each algorithm main iteration (iteration best, *ib*) or of all iterations (global best, *gb*).

The planner inputs (requirements) are: initial positions of the UAVs $s^0_{1:U}$, number of control actions of the trajectories N, probability functions (target initial probability map $P(\nu^0)$, target dynamic model $P(\nu^t|\nu^{t-1})$ and sensor model $P(z^t_u = \overline{D}|\nu^t, s^t_u)$), and MMAS parameters (number of ants M, pheromone influence α, heuristic influence β and evaporation rate ρ). The planner outputs are the best trajectory found by the algorithm (${}^{gb}\boldsymbol{s}$) and its corresponding expected time of detection (${}^{gb}\boldsymbol{ET}$).

The steps of the algorithms are the following. It starts initializing the global best variables (${}^{gb}\boldsymbol{s}$,${}^{gb}\boldsymbol{c}$,${}^{gb}\boldsymbol{ET}$) and the pheromone table ($\boldsymbol{\tau}_{\text{NODE}}$, nullifying the values of the pheromones corresponding to the actions that lead the UAVs outside the search area). Within the main iteration loop (lines 3 up to 27), M solutions/ants are constructed (line 13) step by step combining (line 12, Eq. (14)) the information obtained by the heuristic function (line 11, Eq. (13)) and the pheromones. Besides, the unnormalized belief $\tilde{b}(\nu^t)$ is updated with the target motion (line 8, Eq. (5)) and sensor no-detection measurements (line 16, Eq. (6)) as the value of $\bar{b}(\nu^t)$ is required to compute the heuristic function (line 11). Moreover, in line 17 we take advantage of this calculation to compute the ET of the solutions by iteratively adding up each of its summation terms (Equations (4) and (7)), instead of obtaining the ET once the whole trajectory is available at line 19. In other words, in order to reduce the computational cost of the algorithm, we have interlaced the operations required to construct the solution and compute its ET. Once the whole population of ants have finished constructing the solutions, the best solution obtained within the current

Algorithm 1 ACO-NODE+H

Require: $N, s^0_{1:U}$ ▷ Number of control actions and initial UAVs locations
Require: $P(\tau^0), P(\tau^t|\tau^{t-1}), P(\overline{D}^t|z^t, s^t_a)$ ▷ Target and sensor models
Require: M, α, β and ρ ▷ ACO parameters
$^{gb}\boldsymbol{ET} \leftarrow \infty, ^{gb}\boldsymbol{s} \leftarrow [], ^{gb}\boldsymbol{c} \leftarrow []$ ▷ Initialize fitness function (global best solutions)
$\boldsymbol{\tau}_{\text{NODE}} \leftarrow InitializePheromoneNode()$ ▷ Initialize pheromone table
while no finished **do** ▷ Main loop
  for m=1:M **do** ▷ Loop for each ant in the population (it can be parallelized to speed the algorithm)
    $ET(s^0_{1:U}) \leftarrow 0$ ▷ ET due to the initial location of the UAVs
    $\tilde{b}(\nu^0) \leftarrow b(\nu^0)$ ▷ Initialize unnormalized belief
    for t=1:N **do** ▷ Solution construction loop
      $\bar{b}(\nu^t) = \sum_{\nu^{t-1} \in \Omega} P(\nu^t|\nu^{t-1})\tilde{b}(\nu^{t-1})$ ▷ Predict the unnormalized belief, Eq. (5)
      for u=1:U **do** ▷ Loop for each UAV
        $i \leftarrow s^{t-1}_u \in s^{0:t-1}_{1:U}$ ▷ Get the current location (cell) of the UAV
        $\eta(a, i, t) = \sum_{j \in triangle(a,i,N-t)} f(distance(i,j))\bar{b}(\nu^t = j)$ ▷ Calculate de heuristic value, Eq. (13)
        $p(a,t,u) = \frac{\left(\tau_{\text{NODE}}[a, s^t_u, u]\right)^{\alpha}\left(\eta(a, s^t_u, t)\right)^{\beta}}{\sum_{a=1:8}\left(\tau_{\text{NODE}}[a, s^t_u, u]\right)^{\alpha}\left(\eta(a, s^t_u, t)\right)^{\beta}}$ ▷ Calculate the ACO probability to sample from, Eq. (14)
        $c^t_u \sim p(a,t,u)$ ▷ Sample the cardinal action to move to the following cell
      end for
      $s^{0:t}_{1:U} \leftarrow \{s^{0:t-1}_{1:U}, c^t_{1:U}\}$ ▷ Add new components to UAVs trajectories
      $\tilde{b}(\nu^t) = \prod_{u=1:U} P(z^t_u = \overline{D}|\nu^t, s^t_u)\bar{b}(\nu^t)$ ▷ Update the unnormalized belief, Eq. (6)
      $ET(s^{0:t}_{1:U}) \leftarrow ET(s^{0:t-1}_{1:U}) + \sum_{\nu^t \in \Omega} \tilde{b}(\nu^t)$ ▷ Evaluate the ET due to new segments of the trajectories
    end for
    $\boldsymbol{s}_m \leftarrow s^{0:t}_{1:U}, \boldsymbol{c}_m \leftarrow c^{1:t}_{1:U}, \boldsymbol{ET}_m \leftarrow ET(s^{0:N}_{1:U})$ ▷ Store the information obtained for the current ant
  end for
  $[^{ib}\boldsymbol{ET}, ^{ib}\boldsymbol{s}, ^{ib}\boldsymbol{c}] \leftarrow SelectBest(\boldsymbol{ET}_{1:M}, \boldsymbol{s}_{1:M}, \boldsymbol{c}_{1:M})$ ▷ Select main loop iteration best solution
  $[^{gb}\boldsymbol{ET}, ^{gb}\boldsymbol{s}, ^{gb}\boldsymbol{c}] \leftarrow SelectBest([^{gb}\boldsymbol{ET}, ^{ib}\boldsymbol{ET}], [^{gb}\boldsymbol{s}, ^{ib}\boldsymbol{s}], [^{gb}\boldsymbol{c}, ^{ib}\boldsymbol{c}])$ ▷ Select best overall (global) solution
  $\tau_{\text{NODE}}[^{ib}c^t_u, ^{ib}s^{t-1}_u, u] \leftarrow \tau_{\text{NODE}}[^{ib}c^t_u, ^{ib}s^{t-1}_u, u] + 1/^{ib}\boldsymbol{ET}$ ▷ Pheromone reinforcement with best iteration solution, Eq. (9)
  $\tau_{\text{NODE}}[a, i, u] \leftarrow (1-\rho) \cdot \tau_{\text{NODE}}[a, i, u]$ ▷ Pheromone evaporation for all pheromone values, Eq. (10)
  $[\tau_{min}, \tau_{max}] \leftarrow PheromoneLimits(^{gb}\boldsymbol{ET})$ ▷ Determine pheromone limits taking into account $^{gb}\boldsymbol{ET}$, Ref. [27]
  $\tau_{\text{NODE}} \leftarrow max\{\tau_{min}, min\{\tau_{max}, \tau_{\text{NODE}}\}\}$ ▷ Pheromones are kept between the allowed range
end while
return $^{gb}\boldsymbol{s}, ^{gb}\boldsymbol{ET}$ (solution with minimum ET)

population (line 21, *ib*) and so far (line 22, *gb*) are identified and stored. Next, the pheromone update process takes place: the pheromone values corresponding to the iteration best solution (*ib*) are reinforced (line 23), all pheromone values are evaporated (line 24) and bounded (line 26) between the pheromone limits ($[\tau_{min}, \tau_{max}]$), computed in line 25 following the process described in [27].

4.3.2. MTS ACO-TIME+H Planner

Although Algorithm 1 corresponds to the node encoding, it can be used for ACO-TIME+H after modifying the lines highlighted in navy blue with the following changes. In line 2, all the values of the time encoding pheromone table $\boldsymbol{\tau}_{\text{TIME}}$ have to be initialized to the same value. In line 12, the probability distribution has to be obtained with Eq. (15) and modified to avoid the actions that lead the UAVs outside the search area. And, in lines 23, 24 and 26, the pheromone reinforcement of Eq. (11), the pheromone evaporation of Eq. (12) and the pheromones bounds obtained according to [27] have to be applied.

4.3.3. Computational cost of the MTS Planner

Finally, we calculate the computational complexity of our planner. To do it, we can study Algorithm 1 and observe that within the main loop (between lines 3 and 27), there are three nested loops: for each ant of the population, for each time step of the trajectory and for each UAV of the fleet. Besides, as the beliefs are defined over the $w_x * w_y$ cells of the search region Ω, the cost of the computational demanding operations in Eqs. (5), (13) and (6) are respectively proportional to $(w_x w_y)^3$, $(w_x w_y)^2$ and $(U(w_x w_y)^2)$. Hence, the computational complexity of the main loop is $O(MN((w_x w_y)^3 + U(w_x w_y)^2)) = O(MN w_x^2 w_y^2 (w_x w_y + U))$, what implies that the computational cost is 1) significantly dependent on the number of cells in Ω, on the number of ants and on the number of time steps of the trajectory, and 2) that is slightly modified by the numbers of UAVs. However, when we make $M = 8 * N * U$ (as in the experiments presented in Section 5), the computational complexity of the main loop becomes $O(UN^2(w_x w_y)^3)$. Finally, it is worth noting that the computational cost for some scenarios can be reduced. For instance, in static scenarios, Eq. (5) becomes $\bar{b}(\nu^t) = \tilde{b}(\nu^{t-1})$. Hence the general computation complexity of the main loop becomes $O(MN w_x^2 w_y^2 U)$ and the one obtained when $M = 8 * N * U$ $O(U^2 N^2 (w_x w_y)^2)$.

In order to calculate the total complexity of the planner, we can also account for the number of iterations of the main loop. However, this total complexity will not provide information about the convergence speed of the algorithm, which is highly dependent of the probability functions of the problem (initial belief, target motion model and sensor detection function). Hence, in the following section we will analyze our MTS planner temporal performance over different scenarios to be able to determine, empirically, the time needed to optimize them.

5. Results

This section shows the potential of the proposed MTS algorithm based on ACO. First, we describe several MTS scenarios and the evaluation setups, next we analyze the performance of multiple parameterizations of the different ACO variants and, finally, we compare them against three other MTS optimization algorithms (CEO, BOA and GA) and heuristics.

5.1. Scenarios Setup

We analyze the results of all methods over six search scenarios, several of them already used in [8] to compare CEO and BOA performance in MTS.

All scenarios have the same grid size ($w_x = w_y = 20$) and differ in the initial probability map $b(\nu^0)$, target motion model $P(\nu^t|\nu^{t-1})$, number of steps of the horizon N, number of UAVs U and their initial locations s_u^0.

Their properties are sketched in Fig. 4: the UAVs initial positions are indicated with gray circles; the target motion models are outlined with white arrows; and the initial beliefs are represented with colored maps (where cells with higher probabilities of target presence are represented with warmer colors). Other characteristics of each scenario are detailed in the labels at the top of each graphic of Fig. 4 and below.

Scenario A has two high probability areas equally spaced from the UAV initial position. Its difficulty is due to the small difference in ET of the solutions that go to each of the probability zones, as the probability of the area situated in the south is only slightly higher.

Scenario B has the belief initially concentrated in the center of the search area and as time passes the probability moves towards the southeast. The UAV has to intercept and gather the probability mass as soon as possible.

Scenario C complexity lies on the circular spreading movements of two initially concentrated probability areas. Each of the two UAVs in this setup should follow and gather one of the two probability masses.

Scenario D has a complex target dynamic model that simulates the movements of a lost boat in the sea, which is obtained from a probabilistic wind map. The two central UAVs are expected to intercept the belief and the bottom left UAV to follow the target displacements.

Scenario E initial belief is concentrated in the center of the area and as time passes it spreads out towards the initial position of the two UAVs, which should first move towards the belief and then turn back to overfly the remaining probability.

Scenario F has two static high probability areas on each side of a single UAV. The UAV needs to go east first, towards the highest probability area, and then fly back over the same trail, towards the other probability zone.

5.2. Representative Solutions

Fig. 5 shows one of the final solutions (UAVs trajectories represented with the colored lines) obtained with ACO-NODE+H (as the results of Section 5.5

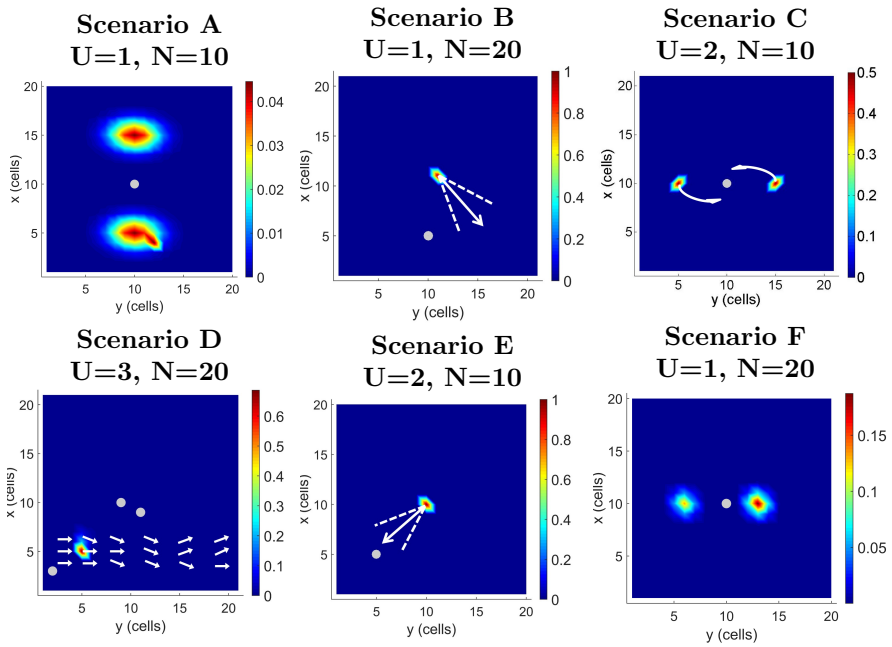


Figure 4: Search scenarios.

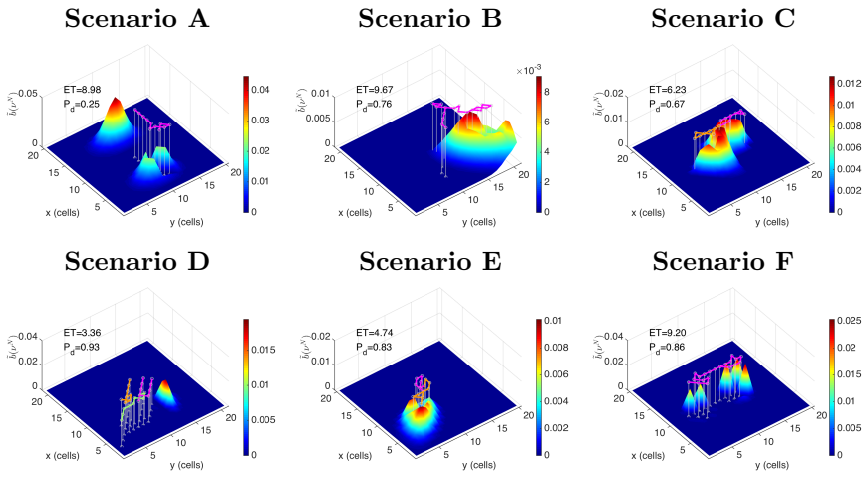


Figure 5: Solutions obtained with ACO-NODE+H.

show that it produces slightly better results than ACO-TIME+H) and its corresponding updated unnormalized probability map $\tilde{b}(\nu^N)$ at the end of the trajectory. The fitness function of the selected solution (ET) and its associated probability of detection (P_d) are also indicated within each graph.

It is important to notice that due to the high P_d gathered by the UAVs sensors in most of the scenarios, the colormaps and z-axis limits of $\tilde{b}(\nu^N)$ in Fig. 5 have been rescaled with respect to the initial beliefs in Fig. 4. For instance, in Scenario F the value of the red color of the highest probability area in the original belief is slightly bigger than 0.15 (see bottom right graphic of Fig. 4), while the value of the red color of the highest unnormalized belief areas in $\tilde{b}(\nu^N)$ is 0.025 (see bottom right graphic of Fig. 5). This happens because the UAV has overflown both initial probability areas gathering most of the belief.

The results of Fig. 5 show that the solutions returned by ACO overfly sooner the areas with higher chances of target presence (those with higher $\tilde{b}(\nu^t)$), making the UAVs cooperate in order to gather the unnormalized belief as soon as possible. For instance, in Scenario A the heuristic in ACO-NODE+H makes the algorithm prefer the actions that lead the UAVs to the highest probability area, helping to overcome the difficulty of the scenario. Besides, in Scenario C both UAVs cooperate, by each one following one of the highest spreading probability areas. Moreover, in Scenarios C and D, where the spreading probability movements complicate the search, the UAVs fly first to the areas with higher concentrated belief to achieve soon a high probability of target detection. Or in Scenario F, the heuristic helps the algorithm to identify the high probability area first and then make the UAV turn in the opposite direction to overfly the unnormalized belief finally gathering 86% of the initial probability ($P_d = 0.86$).

5.3. Comparison Method

Due to the stochastic nature of many of the approaches compared in this paper, a statistical analysis of their results is needed. Therefore, for each search scenario we store the computation time[1] and the solutions with the best ET ($^{gb}\boldsymbol{ET}$) obtained at the end of each iteration by 50 runs of each algorithm. With this information, we calculate the mean computation time of each algorithm iteration and the information represented in the two types of comparative graphs detailed below and represented in Figs. 6, 7, 8, 9 and 10. Note that the graphics within each of the columns of these figures are associated to two scenarios, whose information is divided by a horizontal line (e.g. in Fig. 6 the top two graphics of the first column correspond to Scenario A while the bottom two graphics within the same column belong to Scenario D).

On one hand, we apply the Wilcoxon test to compare the results obtained at the mean computation time of each iteration by the different algorithms/variants in order to determine if the results are statistically different. We present the

[1] All algorithms are implemented in Matlab and run over a 2.5 GHz Intel Core i7 with 6GB RAM PC with Windows 7. Besides, the operations used to evaluate the ET of the solutions are speed up using the Matlab Parallel Toolbox.

results of this test in the *dominance graphs* of the first and third rows of Figs. 6, 7, 9 and 10; and of the first and fourth rows of Fig. 8. To build these dominance graphs, we first compare all algorithms/variants against a base algorithm (indicated in the caption of each figure). This comparison is performed with the results obtained at the mean computation time of each iteration of the base algorithm against the results obtained by the iteration with the closest (equal or small) mean computation time of the other algorithms. Next, we represent the outcomes of the comparison for the different approaches (indicated in the y-axis) at the different computation times of the base algorithm (displayed at the x-axis) using the following colors: green if the approach indicated in the y-axis dominates the base algorithm, red if the base algorithm dominates the approach indicated in the y-axis, gray if there is not statistical difference, and black when the method in the y-axis has still not finished when the first iteration of the base algorithm has ended. Therefore, the results of the iterations of the algorithms in green are statistically better than the results of the base algorithm, in red statistically worse and in gray similar. Note that, as one algorithm/variant can not dominate itself, the corresponding row of the base algorithm is always completely gray.

On the other hand, we also use the stored values to obtain, for each algorithm and the mean computation time of each iteration, the mean ET and its standard deviation. This information is presented in the *ET evolution graphs* that appear in the second rows of each scenario in Figs. 6, 7, 9 and 10; and in the last two rows of each scenario in Fig. 8. These ET evolution graphs represent in different colors (for each algorithm or parameterization under study) and against their computation time, the mean ET and the shadowed area that delimits its standard deviation. Therefore, those algorithms/parameterizations that show lower ET mean values sooner, converge to a better solution quicker. Besides, over the mean ET curve, we mark with dots the mean computation time of every 10 iterations of each algorithm/variant. Hence, the reader can 1) determine the computation time required by 10 iterations of an algorithm/variant by observing the computation time of two consecutive dots of its ET curve and 2) compare the scenarios computation costs based on the density of these dots.

Finally, it is worth noting that both types of graphics complement each other. The dominance graphs show at the computation time of the base algorithm/variant if there is a statistical difference between the algorithms/variants under analysis, but they do not show how different they are. The ET evolution charts graphically represent the magnitude of the ET difference against the computation time, but they lack of the statistical significance information of the dominance graphs. For instance, the dominance graph of Scenario D represented in the first column and third row of Fig. 6 shows that the parameterizations labelled as 2, 3, 8, 9 11, 12, 14, 15, 17 and 18 are statistically better (in green) than the base parameterizations (labelled as 1) during the 5 initial seconds and that become worse (in red) that the base parameterization after 15 seconds. Or equivalently, that for Scenario D the base parameterization becomes the best of the ones analyzed in Fig. 6 after 15 seconds. However, this dominance graph does not show the magnitude of the improvement, which is represented in the

Table 2: ACO parameter configurations under study.

Config. Labels	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
ρ	0.5	0.5	0.5	0.5	0.5	0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.02	0.02	0.02	0.02	0.02	0.02
α	1	1	1	2	2	2	1	1	1	2	2	2	1	1	1	2	2	2
β	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3

ET evolution curves of the first column and fourth row of Fig. 6, that for the same scenario show how the mean of the ET of the first configuration (labelled as Conf. 1) converges quicker to better (lower ET) solutions than the other three parameterizations represented within this graphic (labelled as Conf. 2, 7 and 10).

5.4. MTS ACO Performance Analysis

This section presents the results of the analysis of our ACO planner under both encodings (NODE and TIME) and different parameterizations, obtained by modifying the ACO parameters (ρ, α, β) and enabling/disabling the use of the pheromones and heuristic function.

The common characteristics of the ACO planner in all the analysis of this work are the following. Similarly to [8], we set bigger population sizes (M) for more complex scenarios, by making them proportional to the planning horizon N, the number of UAVs U and the number of possible actions (i.e, the population size $M = 8 \cdot N \cdot U$). Besides, a predefined maximum computation time long enough to see the convergence values of the configurations is set as stop condition, and the values of τ_{min} and τ_{max} are determined and recomputed each time a new best solution is found, following the methodology proposed in [27].

5.4.1. General Parameterization Analysis

The best set of values of ACO parameters depends on the given problem, its heuristic and the available computational time. Hence, the analysis presented in this section is intended to find the set of parameters that proportionate high quality solutions of MTS in reasonable time. To obtain them, we have tested 18 different configurations of ACO parameters, varying ρ, α and β within a range of values ($\rho = \{0.5, 0.1, 0.02\}$, $\alpha = \{1, 2\}$, $\beta = \{1, 2, 3\}$) that can be typically found in the literature [10, 27, 29]. Each configuration appears in a different column of Table 2, where it is labelled with the numbers appearing in its first row and parameterized with the values of ρ, α and β shown underneath. The main results of the analysis for ACO-NODE+H (ACO with NODE encoding plus Heuristic) and ACO-TIME+H (ACO with TIME encoding plus Heuristic) are respectively presented in Figs. 6 and 7, and the selection of the base configuration (Conf. 1) of the dominance graphs is justified in Appendix A.

The dominance graphs of Fig. 6 for ACO-NODE+H show, in all scenarios but A, that after 10 seconds the majority of the other configurations are usually worse (in red) than Conf. 1 and that, after a while, Conf. 2, 7 and 10 are usually as good (in gray) as Conf. 1 in at least three scenarios. Hence, we represent

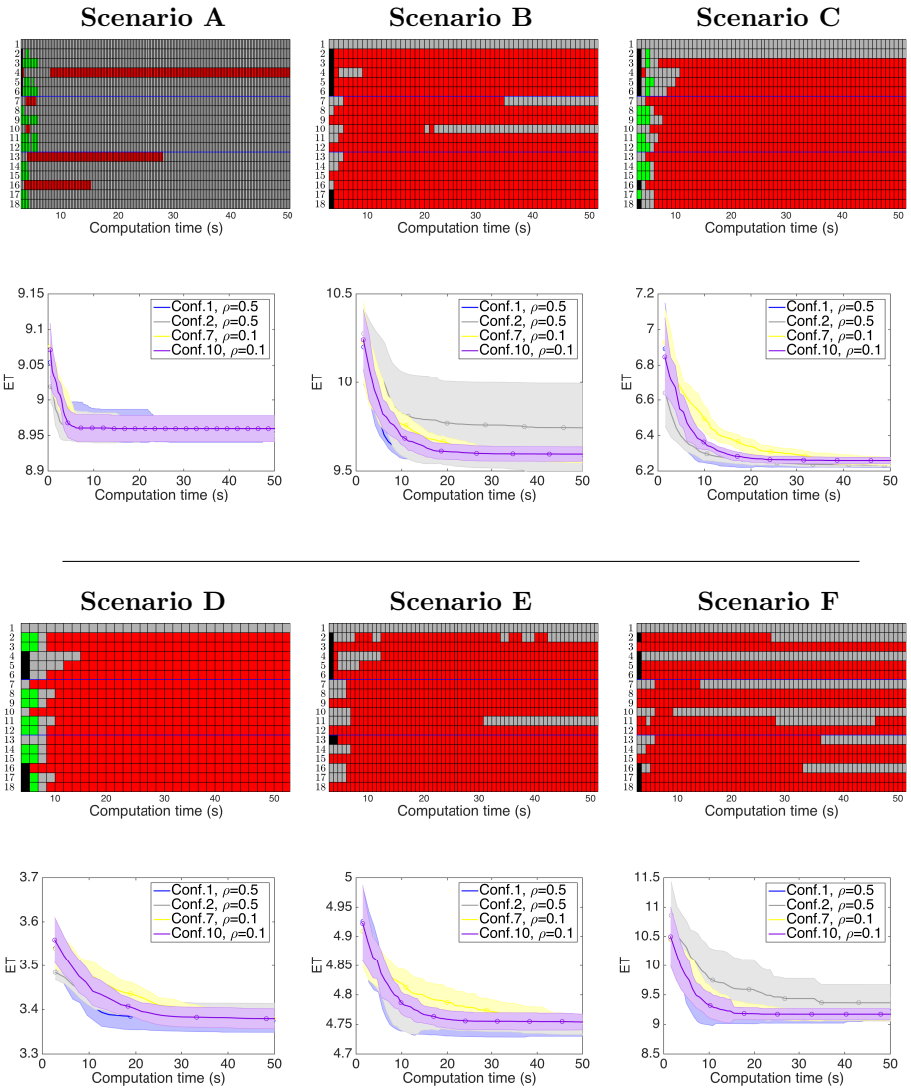


Figure 6: Comparison of all parameter configurations for ACO-NODE+H (base parameterization = Conf. 1).

the ET evolutionary curves of these four configurations and observe that Conf. 1 converges quicker to a final ET value similar to the final ones obtained by the others. Besides, Conf. 2 presents a higher ET variance in several scenarios; for Scenarios B and F Conf. 10 becomes equally good as Conf. 1 quicker than Conf. 7; and for Scenario A the majority of the configurations are equally good (the dominance graph is almost gray).

Analogously, the dominance and ET evolution graphs of all parameter con-

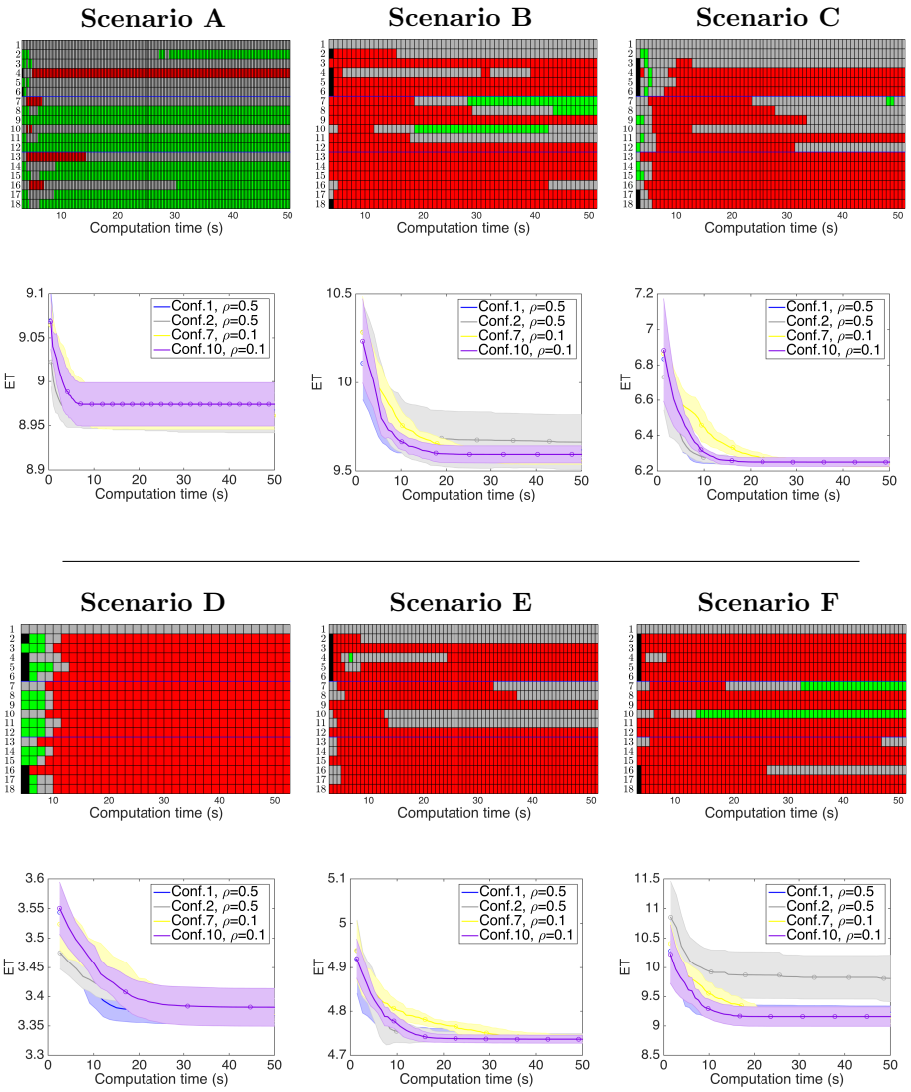


Figure 7: Comparison of all parameter configurations for ACO-TIME+H (base parameterization = Conf. 1).

figurations for ACO-TIME+H are presented in Fig. 7. The dominance graphs suggest us to represent the ET evolution graphs of Conf. 1, 2, 7 and 10, because 1) the majority of the configurations are usually worst (in red) that the base (Conf. 1) in all scenarios but A; and 2) Conf. 2, 7 and 10 become as good (in gray) as or better (in green) than Conf. 1 in four scenarios. Again, the ET evolution graphs show that Conf. 1 converges quicker than the others to solutions with similar ET.

Table 3: ACO encodings and heuristic variants.

Short Label	Pheromone table	Heuristic	α	β	ρ	M
NODE+H	$\boldsymbol{\tau}_{\text{NODE}}$	✓	1	1	0.5	$8 \cdot N \cdot U$
NODE	$\boldsymbol{\tau}_{\text{NODE}}$		1	0		
TIME+H	$\boldsymbol{\tau}_{\text{TIME}}$	✓	1	1		
TIME	$\boldsymbol{\tau}_{\text{TIME}}$		1	0		
H	-	✓	0	1	-	

From the analysis of the graphics of both encodings and the values of the parameters of Table 2, we can draw the following conclusions. First, four configurations (two with $\rho = 0.5$ and two with $\rho = 0.1$) produce pretty similar results. Second, within $\rho = 0.5$, although Conf. 2 starts at better solutions than Conf. 1 in some scenarios, Conf. 1 usually converges quicker to good solutions than Conf. 2. Third, within $\rho = 0.1$, Conf. 10 converges quicker to good solutions than Conf. 7. And fourth, although Conf. 1 converges also quicker than Conf. 10, in ACO-TIME+H Conf. 10 is able to slightly improve the results of Conf. 1 in some scenarios. These facts show how the balance (obtained through the combined values of ρ, α and β) is achieved for our MTS planner: when the evaporation rate is higher ($\rho = 0.5$), the values of the pheromones and heuristic influence parameters should be low and equally considered ($\alpha = \beta = 1$); when the evaporation rate is lower ($\rho = 0.1$) the pheromones importance should be increased ($\alpha = 2, \beta = 1$). Finally, due to quickest convergence of Conf. 1 to good solutions, we will select its parameterization for both ACO encodings.

Finally, it is worth noting that we have also tested the performance of the configurations with different fixed M, but the results showed that the variants that make $M = 8 \cdot N \cdot U$ usually outperformed the others.

5.4.2. Encodings and Heuristic Analysis

In order to evaluate the capabilities of the two ACO encodings and the power of the new heuristic that we have proposed for the MTS problem, we compare the results obtained with the five variants of ACO: the two ones with the heuristic enabled (ACO-NODE+H and ACO-TIME+H), the two ones with the heuristic disabled (ACO-NODE and ACO-TIME) and the one with only heuristic information (ACO-H). Hence, the results in this section will let us determine if our approach benefits mainly from the pheromone trails associated to the good solutions identified by the evaluation of the objective function, from the new heuristic introduced in this paper for MTS or from the capabilities of MMAS (the supporting ACO approach of our planner) to combine both of them.

The performance for each scenario of the five variants under study, labelled in this section without the ACO- prefix for simplicity and set up according to the parameterization specified in Table 3, is presented in Fig. 8. The corresponding graphs of the second and third rows of each scenario are respectively the ET evolution graphs of the variants with pheromones and with heuristic.

The dominance evolution graphs in Fig. 8 show that 1) the variants without heuristic (NODE and TIME) or without pheromone trails (H) are statistically dominated (in red) by the base variant (NODE+H) and that 2) the dominance behavior (red, gray, green) of the variants with heuristic (NODE+H and TIME+H) depend on the scenario. However, these graphs do not show the magnitude of the improvement, which can be observed in the ET evolution curves that show that the ACO variants with heuristic (NODE+H, TIME+H and H) provide better solutions from the first iterations and quickly converge to solutions that are at least as good as those finally found by the variants without heuristic (NODE and TIME). Therefore, the new MTS heuristic is adequate for the problem and helps the algorithm to reach higher quality solutions in less computational time.

Besides, NODE+H and TIME+H obtain solutions with lower ET sooner than H. Hence, we can conclude that although the heuristic by itself is able to achieve high-quality results quickly, the pheromone deposit and the heuristic-pheromone combination mechanisms of MMAS help indeed the algorithm to achieve higher quality results faster.

Finally, respect the two encodings analyzed, NODE outperforms TIME in scenarios A, B, C. So, learning the best actions to perform at each node seems to be better for the MTS problem when the heuristic is disabled. However, when the heuristic is enabled NODE+H and TIME+H reach similar high quality solutions quickly. Hence, enabling the MTS heuristic helps ACO-TIME+H to overcome the drawbacks of its encoding and to find high quality solutions too.

5.5. Comparison of ACO Performance with other MTS Approaches

In this section we compare the results of the new ACO-based approach against the results obtained by the approaches of Table 1 that have been previously used in the MTS problem [6, 7, 8] or that can be used straightforward [9]. The other methods in Table 1 have not been tested because the technique in [14] does not scale well for scenarios of 20x20 cells, and the approaches in [15, 16, 16, 17] have been adapted to exploit properties of its objective function (and therefore, they are not meant to be applied directly for optimizing ET).

In particular, we use: 1) the NODE encoding for this comparison as it produces slightly better results than the TIME encoding, and 2) the same fitness function and sensor models for evaluating all the approaches.

5.5.1. Comparison with ad hoc heuristics tested for MTS

In this section we describe and compare, against our MTS heuristic, the ad hoc heuristics that are presented/analyzed in [6] for generating a single-UAV high-level searching trajectory for minimum time search problems with static targets. All but ours are deterministic approaches that generate only one straight-segmented trajectory for the UAV according to the following rules:

- **Heuristic towards the Global Maximum (HGM)** moves, at each time step t, the UAV according to the cardinal direction that will make the UAV arrive quicker to the cell of higher $\bar{b}(\gamma^t)$. Additionally, it sequentially

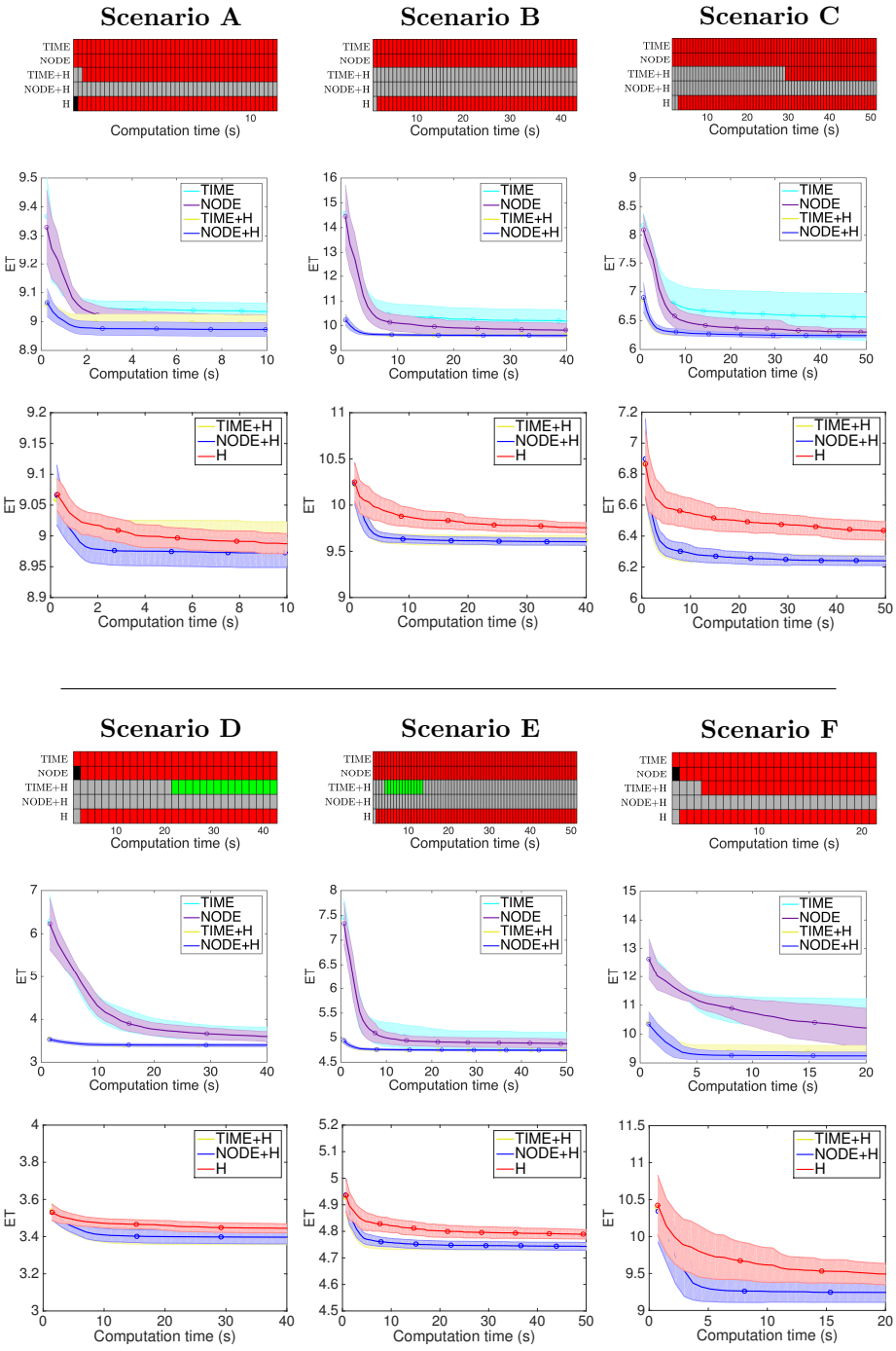


Figure 8: Comparison of ACO variants with and without heuristic (base variant =NODE+H).

updates $\tilde{b}(\gamma^t)$ taking into account the UAV locations and no-detection sensor function. It has a straightforward implementation, but when multiple unconnected probability areas within the search region contain the maximum value, it makes the UAV move between the different areas without collecting the remaining probability within each of them. Besides, when it is directly used for multiple UAVs, it makes all the fleet move towards the same point.

- **Heuristic towards the Local Maximum (HLM)** moves, at each time step t, the UAV according to the cardinal direction that will make the UAV arrive quicker to the cell of higher unnormalized $\bar{b}(\gamma^t)$ within a circle of radius r_{HLM} around the UAV (i.e for those γ^t where $distance(\gamma^t, s_i^t) < r_{\text{HLM}}$). Besides, to avoid getting stuck in a local minimum around the UAV location, this heuristic increments $r_{\text{HLM}} \leftarrow r_{\text{HLM}} + \Delta_{\text{HLM}}$ when the maximum of $\bar{b}(\gamma^t)$ falls outside the circle around the UAV location s_i^t during several consecutive time steps N_{HLM}. Its strong dependency with the initial UAV positions can make it a better choice than HGM for distributing a fleet of multiple UAVs.

- **Heuristic Spiral (HS)** moves the UAV towards the cell with higher $\bar{b}(\gamma^t)$ and once there, it describes a spiral trajectory around that cell during several time steps N_{HS}. The movement to the maximum followed by a spiral is repeated as many times as permitted by the number of steps N of the UAV trajectory. Although this way of proceeding makes this heuristic ideal for static scenarios with mixtures of circular gaussians, it is not straightforward applicable in dynamic scenarios, where the center of the spiral should be moved to follow the displacements of the target.

In short, in order to show the power of our new heuristic for MTS, we compare its behavior (by itself, without the pheromones) against others (HGM, HLM, HS) already tested in [6] for MTS. Although all of them take into account $\tilde{b}(\gamma^t)$, ours presents an *stochastic* behavior that combines and weights $\tilde{b}(\gamma^t)$ accordingly to eight regions around the cardinal directions and to the distance between the UAV and target locations, while the others show a *deterministic* greedy/spiral behavior towards/around the cell of highest probability.

On one hand, to set up our heuristic, we use ACO+H (i.e. our approach without pheromones) with its parameterization within last row of Table 3. On the other, for the remaining heuristics we make $r_{\text{HLM}} = 5$ initially, and fix $N_{\text{HLM}} = 5$, $\Delta_{\text{HLM}} = 5$ and $N_{\text{HS}} = 5$. Besides, for all scenarios, we run 50 times our stochastic heuristic within ACO+H and once the deterministic maximal heuristics (HGM, HLM). Finally, we only run once HS for the static scenarios, as its deterministic behavior should be adapted for dynamic scenarios to make the center of the spiral follow the motion of the target belief.

The results of the analysis over the six search scenarios are presented in Fig. 9. The dominance graphs show that in all Scenarios but F, our heuristic (H) dominates the result of the others (HGM, HLM and HS). This happens because our method directs the UAV towards *regions with closer and higher*

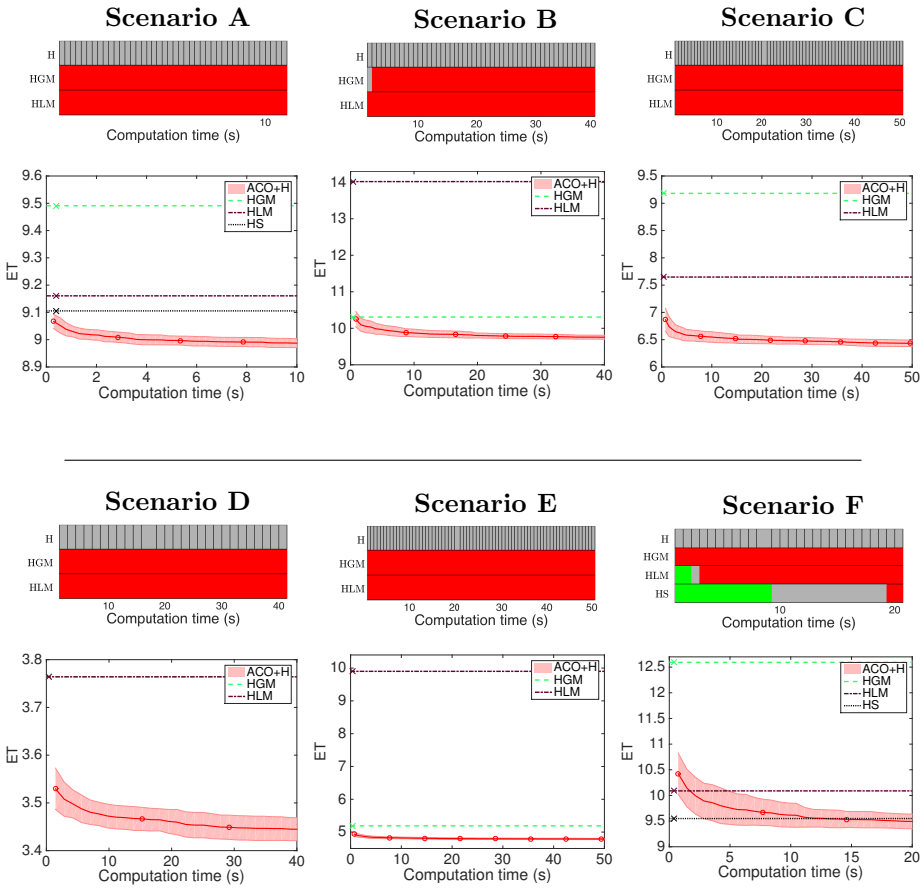


Figure 9: Comparison of implemented ad-hoc heuristics (base variant = ACO+H).

probability of target presence, while the others move the UAV towards *the cell with the global/local maximum belief* in order to follow its location (HGM and HLM) or perform spirals around it (HS). The initial better behavior (in green) of HLM and HS in the dominance graph of Scenario F occurs because in that particular static case, the initial target belief is a mixture of circular gaussians, and therefore going towards the maximal probability cell and performing spiral displacements around it is an overall good strategy. The ET evolution graphs show the stochastic behavior of our approach (since there is a variance around the mean ET line) and the deterministic behavior of the others (represented by a unique line and with an initial single dot associated to the time required to obtain their solutions). Besides, in some scenarios (e.g. in Scenario B and E) HGM is better (has lower ET) than HLM, and in others the opposite happens (e.g. in Scenarios A, C and F). Moreover, in all scenarios but F, our heuristic by itself (that is without its combination with the pheromones trails due to the

objective function) achieve overall better results than the others in the initial steps, and in Scenario F our approach achieves as good solutions as the others in only 10 seconds.

In other words, the new heuristic that we propose in this paper is a good method by itself to generate promising solutions for MTS, which are further improved by our ACO-based planner due its combination with the pheromone mechanism within MMAS. Finally, it is worth noting the high dependency of the results of HLM and HS with their configuration parameters and the characteristics of the search scenarios.

5.5.2. Comparison with optimization methods

In this section we compare against ACO-NODE+H other optimization methods that have been previously used for MTS or that can be used straightforward for generating high-level straight-segmented searching UAVs trajectories. Their characteristics are presented below from the general idea to specific details:

- **Cross Entropy Optimization (CEO)** learns the probability distribution of the solutions from the best solutions of each iteration, which are randomly generated initially [30]. It is first applied to the MTS problem in [7], where it learns the probability distributions of the best actions to perform by the UAVs at each time step. Indeed, each iteration of the CEO in [7] can be divided in two steps: first the UAVs trajectories are sampled from the learned probability distribution and evaluated, and then a new probability distribution is estimated (under the assumption that the actions are independent on each other) by counting the number of times that each action is performed at each time step among the best solutions of the iteration.

- **Bayesian Optimization Algorithm (BOA)** also evaluates the sampled solutions of each iteration to learn a Bayesian Network (BN, [31]), which is a probability model more complex than the one used in CEO. It is applied to solve the MTS problem in [8], optimizing the best actions to perform at each time instant and capturing the relationships among the actions of the best solutions with a BN learned with a greedy algorithm that requires a higher computation time than the action counting performed in CEO. However, after performing a statistical comparison of different BN learning strategies for the MTS problem, we substitute the one used in [8] by the one described in [32] that combines K2 learning strategy and Dirichlet metric, as it converges quicker to similar ET values.

- **Genetic Algorithm (GA)** is a well-known evolutionary algorithm that mimics the process of natural selection, where the population of candidate solutions gets better fitness at each iteration through mechanisms of crossover, mutation and selection [33]. It is the only technique of the analysis that does not learn a probability distribution. It is applied in [9] to maximize the P_d of a static target searched by from several UAVs that can move following only four cardinal directions. That work considers two

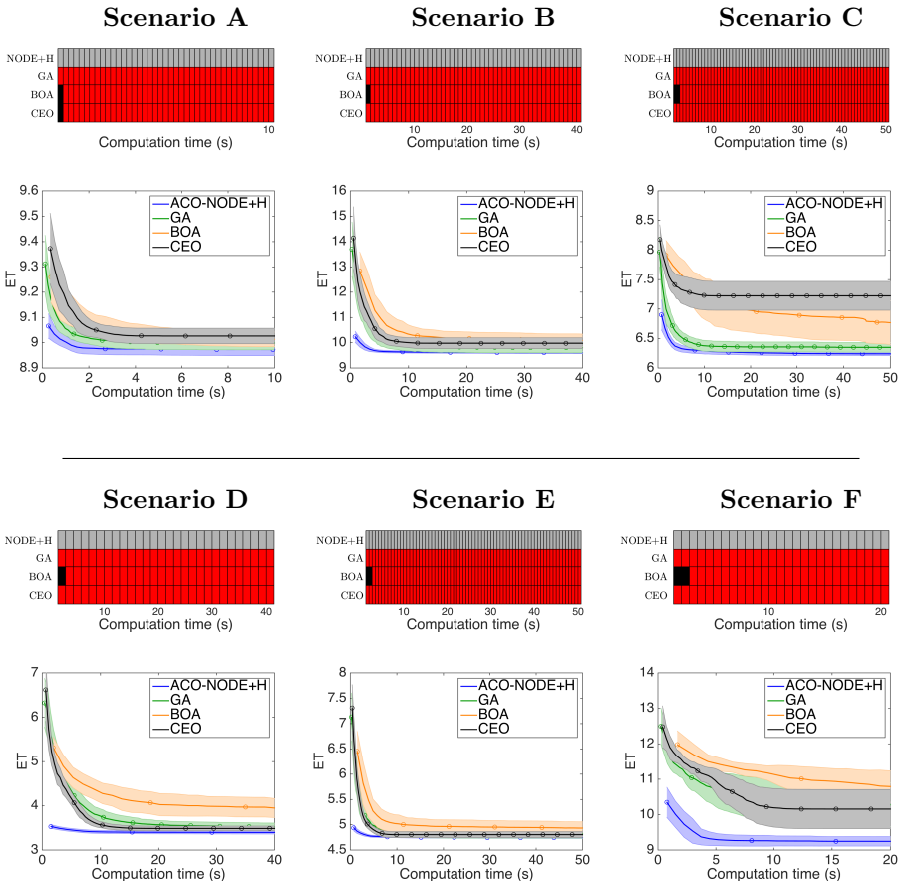


Figure 10: Comparison of all implemented MTS algorithms (base variant = ACO-NODE+H).

different path encodings: as a sequence of directions or as a sequences of nodes. We select the first because it is closer to the approaches analyzed in this work and because it induces a smaller search space. Moreover, the other encoding has two drawbacks: the resulting trajectories after crossover may have different lengths (needing to be truncated or extended) and the parents need to have at least one common node to be able to cross them.

In short, we compare three optimization methods (ACO-NODE+H, CEO and BOA) that learn the distribution of the best solution (ACO through the pheromones, and CEO and BOA through a probability distribution) and one optimization approach (GA) that builds solutions by mixing previous ones. Besides, our new proposal (ACO-NODE+H) includes a new heuristic that is not considered by the other optimization methods (CEO, BOA and GA).

The population size considered in the algorithms of this section is also pro-

portional to the planning horizon N, number of UAVs U and number of actions (i.e. $M = 8 * N * U$), and the stop condition for all the algorithms is a maximum predefined computational time. It is worth mentioning that we have also performed a statistical analysis over different parameterizations of CEO, BOA and GA to make a fair comparison against good versions of all of them.

The results of the statistical analysis over the six search scenarios are presented in Fig. 10. The dominance graphs show that ACO-NODE+H outperforms the other algorithms in all scenarios, while the ET evolution graphs show that ACO-NODE+H finds solutions of similar (Scenarios B and E) or higher quality (Scenarios A, C, D and F) than the others in significantly less computational time. Besides, the ET evolution curves show that CEO and BOA present the slowest convergence. For instance, in Scenario C, CEO converges into suboptimal solutions, while BOA has not yet converged after 50 seconds. Additionally, the ET evolution graphs show that GA presents better results than CEO and BOA, but ACO-NODE+H converges faster in all scenarios. This happens because while all methods but ACO-NODE+H start from random initial populations and only use the information of previous iterations to improve their solutions, ACO-NODE+H combines the MTS heuristic with previously learned information. Therefore, the heuristic generates solutions with better fitness from the first iterations, guides the search and accelerates the convergence of the algorithm to better solutions, which are usually found in just a few seconds.

In addition, since all algorithms have the same population size and the dots over the mean ET curve show the computation time of every 10 iterations of each method, it is possible to compare the computational time of each algorithm iteration. It can be observed that GA and CEO are the algorithms with lower computational time per iteration, that ACO-NODE+H employs an extra time (to calculate its heuristic) and that BOA is the algorithm that requires the biggest time by iteration (due the complexity of computing the underlying BN). Besides, comparing the dots of the same algorithm for different scenarios, we can observe how the computation time of all of them grows as the number of UAVs (U) and trajectory time steps (N) are increased.

6. Conclusions

This work presents a new approach based on an ant colony optimizer for problems with discrete decision variables (MMAS) to minimize the searching time of a fleet of UAVs following high-level straight-segmented trajectories. These trajectories are generated combining the information of 1) promising solutions identified in the previous iterations of the method and of 2) a new heuristic especially designed for the MTS problem. Besides, this paper presents two variants of the approach supported by two different encodings (NODE and TIME) for the pheromone table, one closer to the original ACO and the other closer to the typical codification used in other search algorithms.

The potential of this new approach is tested over six search scenarios and compared against several heuristics (HGM, HLM, HS)/optimization algorithms (CEO, BOA and GA) designed ad hoc or previously tested in MTS. The analysis

of the new approach shows that some of its parameterizations are consistently good over a range of scenarios, that the new heuristic proposed for MTS helps our approach to generate pretty good solutions quickly, and that the combination of the pheromone trails and of our heuristic improves the solutions further in just a few seconds. The comparison against the other approaches shows that 1) our heuristic provides usually quicker better solutions than the other heuristics, and that 2) our method obtains better or similar solutions than the other optimization algorithms in significantly less computational time. On one hand, this happens because our heuristic drives the UAVs towards promising zones instead towards the cells with maximal probability by the other heuristics (HGM, HLM, HS). On the other one, it occurs because the optimization approaches previously applied to the search problem only use the information of the best solutions identified by the previous iterations of each method. In other words, the capability of MMAS of combining the effects of a good MTS heuristic with the pheromone trails of good solutions, allows it to obtain much higher quality solutions from the first iterations.

The main drawback of the approach is that the trajectories obtained by the method presented in this paper can only be directly flown by UAVs capable of following straight-segmented trajectories (e.g. quadrotors) and have to be smoothed/optimized in a second step for others (e.g. fixed-wing UAVs, provided that the distance between the nodes is large enough). Besides, so far, this method only optimizes a unique objective (the ET of target detection), leaving out others (e.g. fuel consumption, avoiding forbidden areas) often considered in more realistic planners [18, 34]. Finally, it is worth noting that in spite of the good results of our current heuristic for MTS, we are planning to combine it with others that are known to be good in certain types of scenarios (e.g. HS for mixtures of circular gaussians or lawn-movements for uniformly distributed probability search regions).

7. Acknowledgements

This work was supported by Airbus under the SAVIER AER-30459 project.

References

[1] G. A. Hollinger, Search in the physical world, Master's thesis, Carnegie Mellon University (2010).

[2] T. H. Chung, G. A. Hollinger, V. Isler, Search and pursuit-evasion in mobile robotics: A survey, Autonomous Robots 31 (4) (2011) 299–316.

[3] A. Khan, Coordinated unmanned aerial vehicles for surveillance of targets, Ph.D. thesis, Alpen-Adria Universität, Klagenfurt (2015).

[4] P. Lanillos, E. Besada-Portas, J. A. Lopez-Orozco, J. M. de la Cruz, Minimum time search in uncertain dynamic domains with complex sensorial platforms, Sensors 14 (8) (2014) 14131–14179.

[5] K. E. Trummel, J. R. Weisinger, The complexity of the optimal searcher path problem, Operations Research 34 (2) (1986) 324–327.

[6] M. Meghjani, S. Manjanna, G. Dudek, Multi-target rendezvous search, in: IEEE International Conference on Intelligent Robots and Systems, 2016, pp. 2596–2603.

[7] P. Lanillos, E. Besada-Portas, G. Pajares, J. J. Ruz, Minimum time search for lost targets using cross entropy optimization, in: Proc. IEEE/RSJ Int. Conf. Intell. Robots and Systems, 2012, pp. 602 –609.

[8] P. Lanillos, J. Yañez Zuluaga, J. J. Ruz, E. Besada-Portas, A bayesian approach for constrained multi-agent minimum time search in uncertain dynamic domains, in: Genetic and Evolutionary Computatio (GECCO), 2013, pp. 391–398.

[9] L. Lin, M. Goodrich, UAV intelligent path planning for wilderness search and rescue, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2009, pp. 709–714.

[10] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: Optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics: Systems 26 (1) (1996) 29–41.

[11] C. Zhang, Z. Zhen, D. Wang, M. Li, UAV path planning method based on ant colony optimization, in: IEEE Chinese Control and Decision Conference, 2010, pp. 3790–3792.

[12] A. Jevtic, D. Andina, A. Jaimes, J. Gomez, M. Jamshidi, Unmanned aerial vehicle route optimization using ant system algorithm, in: 5th International Conference on System of Systems Engineering, IEEE, 2010, pp. 1–6.

[13] S. Perez-Carabaza, J. Bermudez-Ortega, E. Besada-Portas, J. A. Lopez-Orozco, J. M. de la Cruz, A multi-UAV minimum time search planner based on ACOR, in: Proceedings of the Genetic and Evolutionary Computation Conference, (GECCO), 2017, pp. 35–42.

[14] J. N. Eagle, The optimal search for a moving target when the search path is constrained, Oper. Res. 32 (5) (1984) 1107–1115.

[15] Y. Yang, A. Minai, M. Polycarpou, Decentralized cooperative search in UAV's using opportunistic learning, in: AIAA Guidance, Navigation, and Control Conference and Exhibit, 2002, p. 4590.

[16] N. Lo, J. Berger, M. Noel, Toward optimizing static target search path planning, in: IEEE Symposium on Computational Intelligence for Security and Defence Application, 2012, pp. 1–7.

[17] J. Berger, N. Lo, M. Barkaoui, Static target search path planning optimization with heterogeneous agents, Annals of Operations Research (2016) 1–18.

[18] S. Perez Carabaza, E. Besada Portas, L. O. J. Antonio, J. M. de la Cruz, A real world multi-uav evolutionary planner for minimum time target detection, in: Genetic and Evolutionary Computation (GECCO), 2016, pp. 981–988.

[19] F. Bourgault, T. Furukawa, H. F. Durrant-Whyte, Optimal search for a lost target in a bayesian world, in: FSR, 2006, pp. 209–222.

[20] A. Sarmiento, R. Murrieta-Cid, S. Hutchinson, An efficient motion strategy to compute expected-time locally optimal continuous search paths in known environments, Advanced Robotics 23 (12-13) (2009) 1533–1560.

[21] A. Khan, E. Yanmaz, B. Rinner, Information exchange and decision making in micro aerial vehicle networks for cooperative search, IEEE Transactions on Control of Network Systems 2 (4) (2015) 335–347.

[22] R. Chang-jian, Q. Xiao-ming, , G. Xu-ning, Distributed cooperative search control method of multiple uavs for moving target, International Journal of Aerospace Engineering.

[23] M. Raap, S. Meyer-Nieberg, S. Pickl, M. Zsifkovits, Aerial vehicle search-path optimization a novel method for emergency operations, Journal of Optimization Theory and Applications (2016) 1–19.

[24] K. Socha, M. Dorigo, Ant colony optimization for continuous domains, European Journal of Operational Research.

[25] T. Stützle, M. Dorigo, ACO algorithms for the quadratic assignment problem, in: New Ideas in Optimization, McGraw-Hill Ltd., UK, 1999, pp. 33–50.

[26] C. Mao, L. Xiao, X. Yu, J. Chen, Adapting ant colony optimization to generate soft data for software structural testing, Swarm and Evolutionary Computation (2015) 23–36.

[27] T. Stützle, H. H. Hoos, Max-Min ant system, Future Generation Computer Systems (2000) 889–914.

[28] M. Dorigo, L. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation (1997) 53–66.

[29] M. Gentile, A theoretical consideration of the parameters of the max min ant system, Master's thesis, Universidad Politecnica de Madrid (2015).

[30] R. Rubinstein, The cross-entropy method for combinatorial and continuous optimization, Methodology And Computing In Applied Probability 1 (2) (1999) 127–190.

[31] E. Cant-Paz, M. Pelikan, M. Pelikan, D. E. Goldberg, D. E. Goldberg, Boa: The bayesian optimization algorithm, in: Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation, Morgan Kaufmann, 1999, pp. 525–532.

[32] H. Karshenas, A. Nikanjam, B. H. Helmi, A. T. Rahmani, Combinatorial effects of local structures and scoring metrics in bayesian optimization algorithm, in: Proceedings of the First ACM/SIGEVO Summit on Genetic and Evolutionary Computation, ACM, 2009, pp. 263–270.

[33] D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989.

[34] E. Besada-Portas, L. de la Torre, J. M. de la Cruz, B. de Andrs-Toro, Evolutionary trajectory planner for multiple uavs in realistic scenarios, IEEE Transactions on Robotics 26 (4) (2010) 619–634. doi:10.1109/TRO.2010.2048610.

[35] E. Besada-Portas, L. de la Torre, A. Moreno, J. L. Risco-Martn, On the performance comparison of multi-objective evolutionary uav path planners, Information Sciences 238 (2013) 111 – 125.

Appendix A. Complementary Analysis

This appendix shows an illustrative example of the complementary analysis that we have performed to select the base algorithm/variants to compare against in the dominance graphics of Figs. 6, 7, 8, 9 and 10.

The analysis consists in, for each scenario and different computation times, simultaneously comparing, without a previous selection of a base, all configurations/algorithms under study. To do it, we build a table that, for each scenario and configuration/algorithm, displays the mean and standard deviation of the best ET at the associated closer iteration (with equal or immediately low computation time). Besides, we represent, for each scenario and computation time, the *overall dominance* relationships of each of the configurations against the others, using the method in [35] and the graphics of Fig. A.11, where the color of each cell of the graph determines if the configuration in the y-axis dominates (white), is equivalent (gray) or is dominated (black) by the configuration in the x-axis. Therefore, the configurations that perform better for the corresponding scenario and computation time are the ones whose rows have more white cells. In this way, these tables and overall dominance graphs complement each other, the graphs allows to see at a glance which configurations dominate all the others and the tables the magnitude of the improvement of the fitness function (ET).

The results of the data presented in Table A.4 and Fig. A.11 for analysing ACO-NODE+H configurations at 50 seconds, show how Conf. 1, 2, 7 and 13 usually dominate the others and that Conf. 1 (our selected base) dominates a few more configurations than the other three when considering all the scenarios.

Finally, note that although the same type of analysis has been performed at different computation time for the different set of configurations/algorithms, this appendix only displays the data of ACO-NODE+H configuration analysis at computation time of 50 seconds.

Table A.4: ET mean and standard deviation values for ACO-node configurations at 50 seconds.

Conf. Labels	Scenarios					
	A	B	C	D	E	F
1	8.96 ± 0.02	9.58 ± 0.04	6.23 ± 0.01	3.38 ± 0.03	4.74 ± 0.01	9.27 ± 0.22
2	8.95 ± 0.00	9.74 ± 0.25	6.24 ± 0.01	3.40 ± 0.02	4.75 ± 0.01	9.37 ± 0.31
3	8.95 ± 0.00	10.06 ± 0.23	6.26 ± 0.01	3.41 ± 0.02	4.87 ± 0.11	10.22 ± 0.70
4	8.98 ± 0.02	9.68 ± 0.13	6.28 ± 0.03	3.41 ± 0.04	4.75 ± 0.02	9.43 ± 0.46
5	8.96 ± 0.01	9.97 ± 0.27	6.31 ± 0.06	3.41 ± 0.03	4.77 ± 0.05	9.72 ± 0.37
6	8.96 ± 0.01	10.24 ± 0.10	6.36 ± 0.08	3.43 ± 0.02	4.93 ± 0.08	10.78 ± 0.39
7	8.95 ± 0.00	9.58 ± 0.03	6.26 ± 0.03	3.38 ± 0.01	4.76 ± 0.02	9.10 ± 0.03
8	8.95 ± 0.00	9.63 ± 0.04	6.29 ± 0.02	3.40 ± 0.01	4.76 ± 0.02	9.64 ± 0.29
9	8.95 ± 0.00	10.07 ± 0.22	6.30 ± 0.03	3.41 ± 0.01	4.84 ± 0.06	10.35 ± 0.28
10	8.96 ± 0.02	9.60 ± 0.04	6.26 ± 0.02	3.38 ± 0.02	4.75 ± 0.01	9.17 ± 0.10
11	8.95 ± 0.00	9.61 ± 0.04	6.28 ± 0.04	3.39 ± 0.02	4.75 ± 0.02	9.38 ± 0.32
12	8.95 ± 0.00	10.05 ± 0.25	6.30 ± 0.03	3.40 ± 0.00	4.85 ± 0.07	10.35 ± 0.36
13	8.95 ± 0.00	9.71 ± 0.04	6.37 ± 0.04	3.43 ± 0.02	4.79 ± 0.02	9.32 ± 0.10
14	8.95 ± 0.00	9.76 ± 0.06	6.34 ± 0.03	3.43 ± 0.02	4.78 ± 0.02	9.51 ± 0.23
15	8.95 ± 0.00	10.03 ± 0.20	6.35 ± 0.03	3.45 ± 0.01	4.81 ± 0.03	10.38 ± 0.47
16	8.95 ± 0.00	9.68 ± 0.03	6.33 ± 0.04	3.42 ± 0.02	4.78 ± 0.02	9.26 ± 0.09
17	8.95 ± 0.00	9.74 ± 0.06	6.32 ± 0.03	3.41 ± 0.01	4.78 ± 0.02	9.58 ± 0.29
18	8.95 ± 0.00	10.04 ± 0.17	6.34 ± 0.01	3.44 ± 0.02	4.82 ± 0.03	10.41 ± 0.35

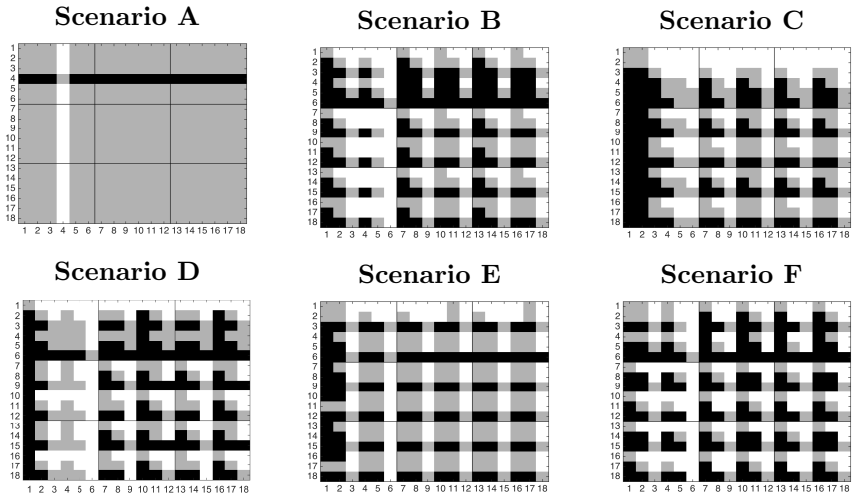


Figure A.11: Overall dominance comparison for ACO-NODE+H configurations at 50 seconds.