A Bayesian Approach for Constrained Multi-Agent Minimum Time Search in Uncertain Dynamic Domains

Pablo Lanillos
Univ. Complutense Madrid
Av. Complutense s/n
Madrid, Spain
planillos@fis.ucm.es

Javier Yañez-Zuluaga
Univ. Complutense Madrid
Av. Complutense s/n
Madrid, Spain
fj.yanez@ucm.es

José Jaime Ruz
Univ. Complutense Madrid
Av. Complutense s/n
Madrid, Spain
jjruz@dacya.ucm.es

Eva Besada-Portas
Univ. Complutense Madrid
Av. Complutense s/n
Madrid, Spain
evabes@dacya.ucm.es

ABSTRACT

This paper proposes a Bayesian approach for minimizing the time of finding an object of uncertain location and dynamics using several moving sensing agents with constrained dynamics. The approach exploits twice the Bayesian theory: on one hand, it uses a Bayesian formulation of the objective functions that compare the constrained paths of the agents and on the other hand, a Bayesian optimization algorithm to solve the problem. By combining both elements, our approach handles successfully this complex problem, as illustrated by the results over different scenarios presented and statistically analyzed in the paper. Finally, the paper also discusses other formulations of the problem and compares the properties of our approach with others closely related.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem solving, control methods and search—*Heuristic methods; Plan execution, formation and generation*; G.3 [**Probability and Statistics**]: Probabilistic Algorithms

Keywords

Decision Making; Multi-agent System; Bayesian Search Theory; Bayesian Optimization Algorithm

1. INTRODUCTION

The constrained agent Minimum Time Search (MTS) problem in uncertain dynamic domains is a decision making process that involves two dynamic partakers: a searcher (sensing agent) with constrained dynamics and a searched object (the target) with uncertain initial location and uncertain dynamics [4, 9]. Its multi-agent version, presented in Fig. 1, is a straightforward extension with a set of sensing agents instead of a single one. In both versions, the objective is to determine the best set of agent actions that simultaneously 1) maximizes the probability of finding the target and 2) minimizes the time to detect it [20].

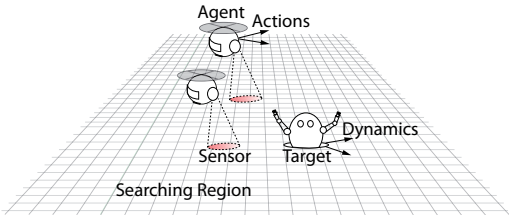

Figure 1: Multi-agent minimum time search

Modeling the uncertainty of the problem with a target initial location belief, probabilistic target motion model and probabilistic sensor model, the problem can be formalized as a Partially Observable Markov Decision Process (POMDP, [19]) defining the reward obtained by the agents during the search [7, 16] and solved with approximated techniques [1].

Alternative approaches appear by exploiting the special structure of the problem to consider it a sub-instance of a POMDP [4, 11] or by reformulating the problem to optimize 1) the probability of finding the target and 2) the detection time. Following this last approach, two types of works appear: those focused on optimizing only the first objective [2, 3, 4, 6, 12] and those that for the single-agent case optimize both using the expected time [9, 18] or a discounted detection probability function [9]. The optimization algorithms used to tackle the alternative formulations vary from gradient based techniques to heuristic methods, including Estimation of Distribution Algorithms (EDA, [10]) such as Cross Entropy Optimization (CEO, [17]).

The main contributions of our Bayesian proposal are:

1) *Two utility functions* to formulate the *dynamic multi-agent* MTS problem within the Bayesian framework: one based on the expected time and the other on a discounted probability function. In this regard, we ex-

GECCO'13, July 6-10, 2013, Amsterdam, The Netherlands.

tend our expressions in [9] for the single-agent ideal sensor case to the multi-agent general sensor case.

2) *A Bayesian Optimization Algorithm* (BOA, [14]) to solve the problem with the purpose of taking advantage during the optimization process of the dependencies induced by the probability models and utility functions into the search space. In this point, we tackle the problem with an EDA with higher searching possibilities than the CEO that we used in [9].
3) *A statistical analysis* of the results obtained over several scenarios by two configurations of the BOA (which allow/avoid it to learn the dependencies of the agent actions) with the two utility functions. We show that the utility function based on the expected time usually outperforms the one based on the discounted detection probability and that a better solution is usually found when the BOA learns the variables dependencies.
4) *A comparison of the properties* of our approach with other closely related solutions.

Our approach improves previous methods to solve the discrete MTS problem and is motivated by numerous applications such as determining the best route to find the survivors of a shipwreck accident [2] or a submarine in an uncertain water environment [8].

2. PROBLEM STATEMENT

In this section we describe the main elements of the MTS problem and the models used to describe their behavior.

The *world* is the space Ω where the target and the agents are contained, and where the search is done. It is defined as a 2 dimensional discretized grid whose $w_x * w_y$ cells can be identified by their indexes ($\Omega = \{1, 2, ..w_x * w_y\}$).

The *agents* are M independent moving sensing platforms that take active part in the search by making observations of the world at different time steps k. The location s_i^k of each agent $i \in \{1:M\}$ is a cell of the grid, always known and determined by the previous agent location s_i^{k-1} and the current control action u_i^k. In other words, given an initial agent location s_i^k and a sequence of N control actions $u_i^{k+1:k+N} = \{u_i^{k+1}, u_i^{k+2}, \cdots, u_i^{k+N}\}$, the trajectory of the agent $s_i^{k:k+N} = \{s_i^k, s_i^{k+1}, s_i^{k+2}, \cdots, s_i^{k+N}\}$, including the starting location s_i^k, is completely determined and viceversa. The actions are defined as the 8 cardinal directions that make an agent move from one cell to its adjacent ones.

The *target* is the object that we are looking for. Its location τ^k at every time step and dynamics are uncertain, although modeled by the initial target location probability distribution or belief $b_\tau^0 = P(\tau^0)$ and the motion model $P(\tau^k|\tau^{k-1})$ that represents the probability of the target moving to location τ^k from location τ^{k-1}. Besides, the lack of dependency of the target location on the agents locations in $P(\tau^k|\tau^{k-1})$ shows that the target is not evading from the agents. Finally, its worth noting that the problem with a static target is a special case of the moving one, where $P(\tau^k|\tau^{k-1})$ equals 1 when $\tau^k = \tau^{k-1}$ and 0 otherwise.

The *sensors*, placed in the agents, are the items capable of observing and collecting information about the world. The observations z_i^k taken by each agent i at time step k are used by the agents to update the target location belief and to decide which sequence of actions are the best. Assuming that the agent and sensor location are the same, i.e. s_i^k, the sensor behavior is modeled with the likelihood function $P(z_i^k|\tau^k, s_i^k)$ that represents the probability of obtaining the measurement z_i^k when the target and agent are respectively at τ^k and s_i^k. Besides, as we consider only two possible observations, target detection D and no target detection $\overline{D}$, $P(z_i^k = D|\tau^k, s_i^k) = 1 - P(z_i^k = \overline{D}|\tau^k, s_i^k)$. As an example, the ideal sensor only and always detects the target when the agent and target coincide (i.e. $P(z_i^k = D|\tau^k, s_i^k)$ equals 1 when $\tau^k = s_i^k$ and 0 otherwise).

Our solution to the MTS problem is the joint sequence of actions $u_{1:M}^{k+1:k+N} = \{u_1^{k+1:k+N}, u_2^{k+1:k+N}, \cdots, u_M^{k+1:k+N}\}$ (or agent trajectories $s_{1:M}^{k:k+N} = \{s_1^{k:k+N}, s_2^{k:k+N}, \cdots, s_M^{k:k+N}\}$) that minimize the time of finding the target, given the target location belief b_τ^k at time k, the target motion model $P(\tau^k|\tau^{k-1})$ and the agents' sensors models $P(z_i^k|\tau^k, s_i^k)$. Finally, to simplify future expressions, we will use D_i^k and $\overline{D}_i^k$ to represent $z_i^k = D$ and $z_i^k = \overline{D}$ respectively, and redefine $P(\tau^k|\tau^{k-1})$ as $T(\tau^{k-1})$ and $P(\overline{D}_i^k|\tau^k, s_i^k)$ as $\overline{O}(\{\tau, s_i\}^k)$.

3. UTILITY FUNCTIONS

In oder to formulate the MTS as an optimization problem we propose two different utility functions in sections 3.2 and 3.3, which simultaneously try to increment the chances of finding the target while minimizing the time required to do it. Section 3.1 shows the recursive expression used to calculate the joint target and non detection history probability, which is required to obtain the values of the utility functions. Each section also shows the simplified version of the functions for the static target case. The demonstration of the expressions has been deferred to Appendix A.

3.1 Joint Target and No Detection Probability

$P(\tau^{k+j+1}, \overline{D}_{1:M}^{k+1:k+j}|s_{1:M}^{1:k+N}, z_{1:M}^{1:k})$, also defined as $f_{\tau,s_{1:M}}^{k+j+1}$, is a probability function that appears naturally in the proposed utility functions and that represents the probability of the target location τ^{k+j+1} at time step $k+j+1$ and of no detecting it from time $k+1$ to $k+j$ given the complete agent trajectories $s_{1:M}^{1:k+N}$ and the real past measurements, $z_{1:M}^{1:k}$. Its values can be calculated progressively with Eq. (1) from $j = 2$ up to $j = N - 1$, considering that $f_{\tau,s_{1:M}}^{k+1} = \sum_{\tau^k \in \Omega} T(\tau^k) b_\tau^k$. Its important to highlight the close relation that exists between Eq. (1) and the Recursive Bayesian Estimation Filter (RBEF, [5]) that calculates $b_\tau^k = P(\tau^k|s_{1:M}^{0:k}, z_{1:M}^{1:k})$ given the initial target location belief b_τ^0, past agents trajectories $s_{1:M}^{0:k}$ and real observations $z_{1:M}^{1:k}$. The RBEF prediction step corresponds to the summation and $T(\tau^{k+j})$ in Eq. (1) while its update step is related to the $\prod_{i=1}^M \overline{O}(s^{k+j})$. However, the normalization factor of the filter does not appear in Eq. (1) and the measurement used to optimize the time (from $k+1$ to $k+j$) are no detections.

$$f_{\tau,s_{1:M}}^{k+j+1} = P(\tau^{k+j+1}, \overline{D}_{1:M}^{k+1:k+j}|s_{1:M}^{1:k+N}, z_{1:M}^{1:k}) = \sum_{\tau^{k+j} \in \Omega} T(\tau^{k+j}) \prod_{i=1}^{M} \overline{O}(\{\tau, s_i\}^{k+j}) f_{\tau,s_{1:M}}^{k+j} \quad (1)$$

When the *target is static*, the previous expression is simplified due to the properties of the motion model. The summation and $T(\tau^{k+j})$ disappear and Eq. (1) becomes:

$$f_{\tau,s_{1:M}}^{k+j+1} = \prod_{i=1}^{M} \overline{O}(\{\tau, s_i\}^{k+j}) f_{\tau,s_{1:M}}^{k+j} \quad (2)$$

Algorithm 1 Bayesian Optimization Algorithm

Require: $b_\tau^{k|k}$ ▷ Prior target location belief
Require: $s_{1:M}^{k}$ ▷ Initial agents location
Require: $T(\cdot)$ ▷ Target motion model
Require: $\overline{O}(\cdot)$ ▷ Sensor model
Require: MTS$(\cdot)$ ▷ MTS Utility function (Eq. (3) or (5))
Require: E ▷ Number of samples
$l \leftarrow 0$ ▷ Set iteration index
$q^l \leftarrow$ Initialize uniformly the action list BN distribution
$U_{1:E}^l \leftarrow$ Sample actions lists: $U_e^l = u_{1:M}^{k+1:k+N} \sim q^l$
$S_{1:E}^l \leftarrow$ Get trajectories: $S_e^l = s_{1:M}^{k+1:k+N} \leftarrow \{s_{1:M}^k, U_e^l\}$
$J_{1:E}^l \leftarrow$ Evaluate trajectories:
$J_e^l = \text{MTS}(S_e^l, b^k, T(\cdot), \overline{O}(\cdot))$
while no finished **do**
  $U'^l \leftarrow$ Select solutions with better $J_{1:E}^l$ from $U_{1:E}^l$
  $q^{l+1} \leftarrow$ Learn new action list BN distribution from U'^l
  $U_{1:E}^{l+1} \leftarrow$ Sample new actions lists:
  $U_e^{l+1} = u_{1:M}^{k+1:k+N} \sim q^{l+1}$
  $S_{1:E}^{l+1} \leftarrow$ Get new trajectories:
  $S_e^{l+1} = s_{1:M}^{k+1:k+N} \leftarrow \{s_{1:M}^k, U_e^{l+1}\}$
  $J_{1:E}^{l+1} \leftarrow$ Evaluate new trajectories:
  $J_e^{l+1} = \text{MTS}(S_e^{l+1}, b^k, T(\cdot), \overline{O}(\cdot))$
  $\{U_{1:E}^{l+1}, S_{1:E}^{l+1}, J_{1:E}^{l+1}\} \leftarrow$ Select solutions from
  $\{U_{1:E}^{l+1}, S_{1:E}^{l+1}, J_{1:E}^{l+1}\}$ and $\{U_{1:E}^{l}, S_{1:E}^{l}, J_{1:E}^{l}\}$
  based on the best values of $\{J_{1:E}^{l+1}, J_{1:E}^{l}\}$
  $l \leftarrow l + 1$
end while
return $s_{1:M}^{*k+1:k+N}$ (solution with best $J_{1:E}^{1:l-1}$)

3.2 Truncated Expected Time Utility Function

Due to the probabilistic nature of the problem, the searching time can be optimized by *minimizing* the Truncated Expected Time (TET) to find the target given the agents trajectories. In the general case, this can be computed with the following expression (which states that the TET is the summation of the probabilities of no detecting the target up to each time step $k+j$) and Eq. (1).

$$\text{TET}(s_{1:M}^{k:k+N}) = \sum_{j=1}^{N} P(\overline{D}_{1:N}^{k+1:k+j} | s_{1:M}^{1:k+N}, z_{1:M}^{1:k}) = \sum_{j=1}^{N} \sum_{\tau^{k+j} \in \Omega} \prod_{i=1}^{M} \overline{O}(\{\tau, s_i\}^{k+j}) f_{\tau, s_{1:M}}^{k+j} \quad (3)$$

When the *target is static*, Eq. (3) is valid, as the inner summation does not include the motion model. However, the $f_{\tau,s_{1:M}}^{k+j}$ of Eq. (3) has to be calculated with Eq. (2).

3.3 Discounted Detection Utility Function

An alternative way of formulating the search problem consists on *maximizing* the probability of detecting the target at any of the following $j = 1 : N$ time steps:

$$P\Big(\bigcup_{i=1:M, j=1:N} D_i^{k+j} | s_{1:M}^{1:k+N}, z_{1:M}^{1:k}\Big) = \sum_{j=1}^{N} \sum_{\tau^{k+j} \in \Omega} \left(1 - \prod_{i=1}^{M} \overline{O}(\{\tau, s_i\}^{k+j})\right) f_{\tau, s_{1:M}}^{k+j} \quad (4)$$

However, this utility function does not necessary optimizes the time, as the authors of [9, 18] point out. Nevertheless, as we suggest in [9] for the single-agent case, the time can be included within the external summation of Eq. (4) as a penalization discount factor λ^j. This creates the Discounted Detection (DD) utility function in Eq. (5), capable with the help of Eq. (1) of optimizing simultaneously the probability and time of detection.

$$\text{DD}(s_{1:M}^{k:k+N}) = \sum_{j=1}^{N} \lambda^j \sum_{\tau^{k+j} \in \Omega} \left(1 - \prod_{i=1}^{M} \overline{O}(\{\tau, s_i\}^{k+j})\right) f_{\tau, s_{1:M}}^{k+j} \quad (5)$$

When the *target is static*, Eq. (5) is valid too, and its $f_{\tau,s_{1:M}}^{k+j}$ has to be calculated with Eq. (2).

4. OPTIMIZATION ALGORITHM

The MTS is a problem whose complexity is NP-hard or higher [21]. Therefore, to find a good solution in a feasible time for grids with many cells, the problem is usually tackled with approximated optimization algorithms and/or heuristics [2, 3, 9, 18, 22]. Besides, due to the constraints imposed by the agents possible movements, it is easier to search for actions lists than for its corresponding trajectories as the action state space (with 8 possible values per action) is smaller than the location one (with $w_x * w_y$ possible values per location) and the actions lists create feasible trajectories. Moreover, if we analyze in detail any of the proposed utility functions for TET or DD, we can observe that the external summation partitions the variables in increasing subsets of state variables according to Eq. (6). Finally, although the search of actions lists is advantageous, the conversion from actions lists to trajectories creates a dependency among the values of the actions in the lists in the utility functions.

$$\text{MTS}(s_{1:M}^{k:k+N}) = \sum_{j=1}^{N} g(s_{1:M}^{k+1:k+j}, b^k, T(\cdot), \overline{O}(\cdot)) \quad (6)$$

Considering the characteristics of the problem, we apply an EDA to optimize it as it is an approach capable of exploiting the probability dependency that exists among the variables of the search space. Additionally, as the variables dependencies also depend on the starting belief b_τ^k, motion model $T(\cdot)$ and sensing model $\overline{O}(\cdot)$, the capability of the EDAs to learn these dependencies releases us from the necessity to impose them beforehand. Finally, among the existing EDAs, we implement the BOA schematized for our problem in Algorithm 1, because BOAs have successfully solved other complex optimization problems [15] and this selection lets us maintain the optimization method within the Bayesian framework. Therefore, the actions lists distribution q^l is represented as a Bayesian Network (BN, [13]) with $M \cdot N$ discrete 8-valued variables. The variables of the first BN q^0 are unconnected and their probability tables uniformly initialized. The dependencies of the variables of the following BNs q^l are learnt from the subset U'^l of the best samples in the sampled actions lists $U_{1:E}^l$, following the method proposed in [14], which is a greedy strategy for learning BNs that only adds those edges that imply a higher increment of the Bayesian Dirichlet metric [13]. The actions lists $U_{1:E}^l$ sampled from the BN q^l are converted into sampled trajectories $S_{1:E}^l$ that are evaluated by the selected utility function MTS$(\cdot)$. The best solutions among the current $S_{1:E}^{l+1}$ and previous $S_{1:E}^l$ iteration samples are used as the the samples of the following iteration $l+1$. The algorithm stops after a fixed number of iterations returning the best trajectory of all the iterations.

Table 1: Closely Related Solutions Comparison

Solution	MTS	D/C	Constrained	Multi-Agent	Moving T.	Optimality(Horizon)	Optimization type
[20]	✓	D,C				Global	Lagrange
[4]		D	✓		✓	Global(N)	DP
[2]		C	✓	✓		Local(1)	Greedy
[22]		D	✓	✓	✓	Local(2)+Expectation	Heuristic+NN
[3]		C	✓	✓		Local(1)	Greedy
[12]		C	✓	✓		Local(N)	Gradient
[18]	✓	D+C	✓			Local Approx(N)	Limited DFS
[6]		C	✓	✓		Local(N)	Explicit Gradient
[9]	✓	D	✓		✓	Local(N)	CEO
This work	✓	D	✓	✓	✓	Local(N)	BOA

5. RELATED WORK

In this section we discuss other formulations of the problem and study the closer related approaches and algorithms.

5.1 Alternative Formulations

MTS can also be formulated as a POMDP [19], adding to the problem definition a reward $R([s^k_{1:M}, \tau^k], u^k_{1:M})$ obtained by the agents given the extended state $[s^k_{1:M}, \tau^k]$ and actions $u^k_{1:M}$. Under this approach, followed in [7, 16], the expected reward $E\left[\sum_k \gamma^k R([s^k_{1:M}, \tau^k], u^k_{1:M}) \middle| b^0_\tau\right]$ is usually maximized. However, as the number of cells in the grid grows, the problem becomes intractable for exact POMDPs algorithms, and it has to be tackled with approximated methods [1].

Other approaches reduce the complexity of the problem exploiting some of its inner characteristics such as the binary detection/non-detection nature of the sensor model. For instance, [4] formulates the problem using a cell expansion and a backward recursive technique of Dynamic Programming (DP) to solve it exactly for small grids, while [11] takes into account the fact that the MTS is finished when the target is found to take out the expectation of the POMDP utility function and optimize it directly with a gradient-based method. However, neither [4, 11] optimize the time directly.

The approaches on next section and ours also tackle the search problem as a POMDP sub-instance. In our case, this strategy lets us optimize, without defining the POMDP reward $R(\cdot)$, utility functions directly related to the expected time and detection probability.

5.2 Closely Related Solutions

In the following, we compare our approach against other solutions closely related to it. Although the comparison is not exhaustive, it constitutes a compilation of the works that have motivated this paper. Besides, all the solutions, except [4], are tractable, i.e., they are able to compute a solution in an acceptable amount of time.

We compare these works based on the seven properties represented in the last seven columns of Table 1:

MTS, the most important for us, indicates which algorithms solve directly the task of locating the target in the minimum possible time. [2, 3, 4, 6, 12] are based on detection maximization, although maximizing the detection does not imply to find the target in minimum time [9, 18]. Indeed, the authors in [2] talk about minimizing Expected Time (ET), although they decline to use its expression due to complexity issues. [22] reduces the uncertainty using the entropy, although this utility function does not imply that the target is located in minimum time. Therefore, only [9, 18, 20] and this work tackle the MTS directly.

D/C indicates if the search is modeled over a continuous world (C) with piecewise linear control actions or in a discrete world (D) with a discrete set of actions. [18] constitutes a special case as it performs the search in the discrete space to obtain a set of points that are later used to determine a continuous space trajectory.

Constrained shows when the dynamic constraints of the agent are considered within the problem formulation.

Multi-Agent indicates if the solution is applicable to several agents. Additionally, [2, 3, 6, 12, 18, 22] are decentralize cooperative approaches, where the agents interchange the knowledge in a transparent way. Besides, in [22] the agents only interchange the information when they are in adjacent cells.

Moving Target shows when the target can be moving during the decision stage. In [2, 3, 6, 12, 20] the target is assumed to be static during the planning horizon. The rest assumes Markovian target motion.

Optimality(Horizon) indicates if the solution is local or globally optimal and the horizon (how many steps ahead the algorithm optimizes). Most of the solutions are not globally optimal in the horizon window because, due to the non-convexity of the problem, they get trapped in local optima (e.g. [6]). Besides, probabilistic optimization approaches like the ones that we have used in [9] and in this paper are not globally optimal, due to their sampling mechanisms. Finally, [22] uses the future expectation as an heuristic, to evaluate also the solutions beyond the horizon.

Optimization Type is more informative than comparative and shows which algorithm is used to solve the problem. [12] and [6] are essentially the same work, although [6] uses an explicit gradient based algorithm to speed up the computation. [22] uses an expectation as the heuristic and Neural Networks (NN) to learn the target location, [4] implements a DP solution, [18] applies a limited Depth First Search (DPS), our work in [9] uses CEO, and in this paper we tackle the MTS problem with a BOA.

Finally, we want to highlight the tight relation that exists between this work and our previous contribution in [9]. In

both cases, we optimize the TET and the DD utility functions and use an EDA. Nevertheless, this paper improves our previous contribution by solving the multi-agent problem for no necessary ideal sensor models and by using a probabilistic optimization algorithm that is able to exploit the inherent dependencies of the variables of the problem.

6. RESULTS

In order to characterize the performance of our approach over different starting conditions and probability models, we use the 5 scenarios schematized in the 5 graphs/columns of the the first row of Fig. 2. The selected scenarios differ in the number of agents (M), number of control actions (N), initial agents location (represented by the red stars), initial target location belief b_τ^k (which is no zero inside the black shapes) and target motion model (static (S) or dynamic (D), the last following the black and blue arrows). The same grid size ($20*20$) and sensor model (ideal) is used in all of them. Finally, in the same graphs we also show with the red arrows the beginning of the best trajectories of the agents.

Due to the probability nature of the MTS problem and sampling support of the BOA used to optimize TET or DD, the performance of our approach is statistically analyzed over the results obtained for 50 runs of the BOA for each utility function over each scenario. Moreover, in order to determine if our approach takes advantage of the capability of the BOA to learn the intrinsic dependencies of each scenario, we compare two different configurations of our BOA: using during the whole optimization a BN without connections[1] (i.e. *without dependencies*) or letting the BOA learn BNs with a maximum number of 3 parents per node (i.e. *with* a limited number of *dependencies*). Besides, for all the scenarios, BOA configurations and utility functions, we run on a Inter Core 2 Duo at 2 GHz a non-optimized Matlab + C implementation of Algorithm 1, with the following settings: $E = 10{\cdot}8{\cdot}N{\cdot}M$ samples, subsets U'^l of size $E/10$, and $\lambda = 0.9$ (when optimizing DD).

The statistics over 50 optimizations for each scenario, BOA configurations and utility functions are summarized in the following rows of Fig. 2:

2) TET/Iter, which shows the mean and standard deviation of the best TET value at each iteration l of the BOA with and without dependencies. The graphics of this row show how BOA with dependencies has better (lower) mean TET values than BOA without them.

3) TET/Time, which shows the mean of the best TET values obtained at the computation times (associated to each iteration) of both BOA configurations. That is, it represents the TET mean against the computation time needed to obtain the solutions instead of against the iteration number, to be able to study if the benefits of learning the dependencies of the BN is shadowed by the time needed by the greedy adding-edges algorithm to learn them. The graphics of this row show a higher computation time increase (dependent on the number $M \cdot N$ of action variables of the scenario) in the BOA with dependencies than in the BOA without them. As a consequence, the mean TET value of the BOA without dependencies is better than of the one with them initially sometimes, although the final value is always slightly/significantly better in the BOA with dependencies. Hence, in some scenarios the time increment of the dependencies learning allows us to find significantly better solutions.

4) DD/Iter, which shows the mean and standard deviation of the best DD value at each iteration l of both BOA configurations. The graphics of this row show how the BOA with dependencies has a better (higher) mean DD value than the BOA without them[2].

5) Dom, which shows the results of the dominance study presented in [9] among the two BOA configurations. The study uses the Wilcoxon test to determine if there is a significant difference between the number of times that each of the 50 solutions obtained by one configuration is better, in terms of the selected utility function, than the 50 solutions obtained by the other. The graphics of this row represents in blue/white, for each iteration and utility function, when the BOA with dependencies dominates the BOA without them/when there is not a significant statistical difference. Hence, this study complements TET/Iter and DD/Iter. On one hand, BOA with dependencies dominates BOA without them at the iterations when there is enough difference between the regions defined by the mean and standard deviation. On the other hand, both configurations are similar at the iterations when the regions overlap. However, from the dominance study we can not infer how big are the differences between the TET/DD values of both configurations. Finally, it is worth noting that this analysis shows that TET with dependencies dominates TET without them more often than its corresponding counterparts in DD do.

6) ETET, which shows for each iteration the *Experimental* TET value associated to both BOA configurations (with (w) and without (w/o) dependencies) and utility functions (DD and TET), in order to determine if one solution is usually better than the 3 remaining. To calculate ETET, we first generate 10000 target trajectories for each scenario sampling the initial target location from the initial target location belief b_τ^k and the remaining positions of the target trajectory from the motion model $P(\tau^k|\tau^{k-1})$. Next, we confront the 10000 target trajectories with the 50 solutions found at each iteration by each utility function & BOA configuration to determine the action step j where the target is found. We use these values to obtain experimentally $P(k \leq t \leq k+j)$ and ETET as $\sum_{j=1}^{N}(1-P(k \leq t \leq k+j))$ [20, 9]. The graphics of this row show that for the same utility function, the ETET value of the BOA with dependencies is usually better than the ETET values of the BOA without them. Besides, in all scenarios except C, the results obtained with TET are better regarding ETET than the ones obtained with DD.

The graphs of Fig. 2 allow us conclude that the mean and dominance results obtained with the BOA configuration that learns the dependencies are better or no worst than the

[1]Note that this configuration and CEO assume the probability independence of the problem variables.

[2]DD/Time is not represented, as the tendency in the time increment (due to the computational time needed to learn the BN) with respect the iteration number of DD/Iter is similar to the one presented in TET/Time.

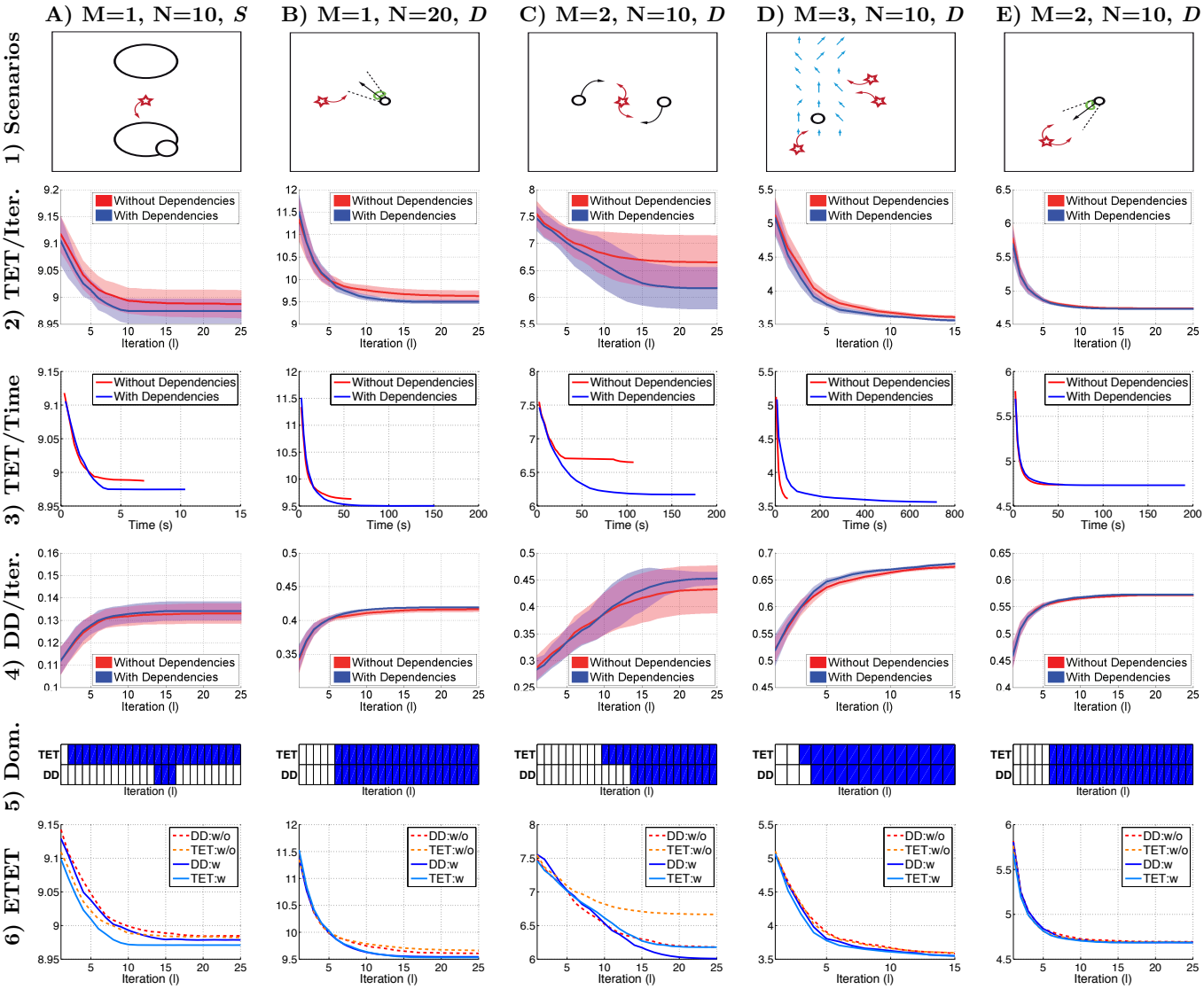

Figure 2: Scenarios and Performance Statistical Analsys

ones obtained without the dependency learning. However, the improvements depend on the scenario characteristics, which justify the variability of the observed behaviors:

Scenario A) The asymmetry in the search space associated to the original b_τ^k (due to the small black circle at the bottom of the scenario) originates the discrepancies and standard deviations observed with both BOA configurations: BOA with dependencies is better as it originally maintains the solutions that move the agent towards the north or the south. This static scenario has the smaller computation time differences too as its number of action variables ($N \cdot M = 10$) is the smallest.

Scenario B) The benefits of learning the dependencies are clearer in this dynamic scenario as the regions defined by the iterations, mean and small standard deviations do not overlap after generation $l = 5$. This happens because this scenario is asymmetric by definition, as its initial target location belief is concentrated in a cell of the map that is probabilistically moved towards the north west.

Scenario C) The benefits of learning the dependencies are clear in this dynamic scenario too. The discrepancies in the results with and without dependencies are bigger than in other cases, due to the scenario complexity originated by the circular spreading movements of the two circular black regions where the initial target location belief is concentrated.

Scenario D) The benefits of learning the dependencies are smaller again in this dynamic scenario that models a lost object in the sea that is moved by the probabilistic wind map associated to the blue arrows. Besides, this scenario has the higher computation time penalization as the BOA has to learn the dependencies of a BN with $N \cdot M = 30$ action variables.

Scenario E) In this dynamic scenario the results obtained with and without dependency learning are pretty similar. This occurs because the selected initial agent locations, init target location belief and target motion

model induce a symmetry in the scenario that makes equally good to explore the 2 possible solutions (a task facilitated by BOA with dependencies) or only one.

Another interesting result observed in the graphics is that TET is better than DD in terms of ETET in all scenarios but C. In other words, TET (which optimizes the truncated expected time) usually obtains the actions lists that allows the agents find the target sooner. But there are some cases where DD (which optimizes a discounted detection probability dependent on an additional tuning parameter λ) is the best choice. This is a consequence of the different landscapes induced by each utility functions and scenario characteristics over the search space.

Finally, we want to highlight that the computation time penalization of the BOA with dependencies can be reduced by optimizing the code or by using a different strategy to learn the BN, for instance, from the previous BN instead of from scratch. Besides, as the results converge before the maximum iteration number, the computation time can be reduced too with a convergence stop criteria. Additionally, the extra time is required for those scenarios whose results are significantly better in the BOA with dependencies.

7. CONCLUSIONS

This paper models, formulates and solves the constrained multi-agent Minimum Time Search (MTS) problem in uncertain dynamic discrete domains following a Bayesian approach. The uncertainty in the problem is 1) associated to the target initial location, target dynamics and agents sensors; and 2) modeled with the initial target location, target motion and sensor probability functions. Based on these probabilities, the problem is formulated as a decision making problem that optimizes the agents actions lists by either minimizing the Truncated Expected Time (TET) or maximizing the Discounted Detection (DD) probability function to find the target. Finally, in order to handle the inherent complexity of the problem, the optimization is carried out using a Bayesian Optimization Algorithm (BOA) capable of exploiting the dependencies that the probability and utility functions impose between the actions of the lists.

The paper also statistically compares the results obtained over 5 scenarios by our approach when 1) the utility function is TET or DD and 2) when the BOA is allowed or not allowed to learn the dependencies between the variables. The results show that TET obtains usually better solutions than DD and that learning the dependencies favors the search of the BOA at the expense of increasing its computation time.

Analysis of the influence of the number of allowed parents or samples in the results will be carried out in the future. Finally, it is worth highlighting that the extension to the continuous domain (in actions and space) is straightforward, by substituting the $\sum_{\tau^{k+j}}$ by integrals and the BOA by an EDA capable of handling continuous search spaces.

8. ACKNOWLEDGMENTS

This work has been supported by the Spanish Ministery of Education and Science under the research grants DPI2006-15661-C02-01 and DPI2009-14552-C02-01.

9. REFERENCES

[1] C. Amato, D. S. Bernstein, and S. Zilberstein. Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Auton Agent Multi-Agent Syst*, 21:293–320, 2010.

[2] F. Bourgault, T. Furukawa, and H. F. Durrant-Whyte. Optimal search for a lost target in a Bayesian world. In *FSR*, pages 209–222, 2003.

[3] F. Bourgault, T. Furukawa, and H. F. D. Whyte. Decentralized bayesian negotiation for cooperative search. In *IEEE Conf. Inteligent robots and Systems*, 2004.

[4] J. N. Eagle. The optimal search for a moving target when the search path is constrained. *Operations Research*, 32(5):1107–1115, 1984.

[5] T. Furukawa, F. Bourgault, B. Lavis, and H. F. Durrant-Whyte. Recursive bayesian search-and-tracking using coordinated UAVs for lost targets. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 2521–2526, 2006.

[6] S. K. Gan and S. Sukkarieh. Multi-UAV target search using explicit decentralized gradient-based negotiation. In *Proc. IEEE International Conference on Robotics and Automation*, 2010.

[7] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101:99–134, May 1998.

[8] B. Koopman. *Search and screening: general principles with historical applications*. Pergamon Press, 1980.

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

[10] P. Larranaga and J. A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic, 2002.

[11] G. Mathews. *Asynchronous decision making for decentralised autonomous systems*. PhD thesis, The University of Sydney, 2008.

[12] G. Mathews, H. Durrant-Whyte, and M. Prokopenko. Asynchronous gradient-based optimisation for team decision making. In *Decision and Control, 2007 46th IEEE Conference on*, pages 3145 –3150, dec. 2007.

[13] R. E. Neapolitan. *Learning Bayesian Networks*. Prentince Hall, 2003.

[14] M. Pelikan, D. Goldberg, and E. Cantu-Paz. Boa: The bayesian optimization algorithm. In *Genetic and Evolutionary Computation Conf.*, pages 525–532, 1999.

[15] M. Pelikan, K. Sastry, and E. Cantu-Paz. *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*. Springer, 2006.

[16] J. Pineau, G. Gordon, and S. Thrun. Anytime point-based approximations for large pomdps. *J. Artif. Int. Res.*, 27(1):335–380, Nov. 2006.

[17] R. Y. Rubinstein and D. P. Kroese. *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-carlo Simulation and Machine Learning*. Springer-Verlag, 2004.

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

[19] R. D. Smallwood and E. J. Sondik. The optimal

Table 2: Basic Probability Operations

Bayes Rule (BR)	$P(A,B\|C) = P(A\|B,C)P(B\|C)$
Marginalization	$P(A\|B) = \sum_C P(A,C\|B)$
$\cup \rightarrow \sum$	$P(\bigcup_{i=1:M,j=1:N} X_i^j) = \sum_{j=1:N} P(\bigcup_{i=1:M} X_i^j, \overline{X}_{1:M}^{1:j-1})$
$\cup \rightarrow \cap$	$P(\bigcup_{i=1:N} X^i) = 1 - P(\overline{X}^{1:N})$

control of partially observable markov processes over a finite horizon. *Operations Research*, 21(5):pp. 1071–1088, 1973.

[20] L. D. Stone. *Theory of optimal search / Lawrence D. Stone*. Academic Press, New York, 1975.

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

[22] Y. Yang, A. Minai, and M. Polycarpou. Decentralized cooperative search in uav's using opportunistic learning. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2002.

APPENDIX

A. UTILITY FUNCTIONS DERIVATIONS

In the appendix, we demonstrate the expressions presented in Section 3. The basic probability properties exploited during the derivations are presented in Table 2.

A.1 Joint Target and No Detection Prob.

The derivation of $f_{\tau,s_{1:M}}^{k+j+1}$ is shown in the following expression, where to go from Eq. (7) to Eq. (8) we marginalize over τ^{k+j}, from Eq. (8) to Eq. (9) apply the BR assuming independence of τ^{k+j+1} on any variable except τ^{k+j}, from Eq. (9) to Eq. (10) substitute the motion model by its alternative expression and apply the BR assuming independence of $\overline{D}_{1:M}^{k+j}$ on everything but τ^{k+j} and $s_{1:M}^{k+j}$, and from Eq. (10) to Eq. (11) assume that the measurements are independent on other measurements or agent locations and substitute the sensor model by its alternative expression.

$$f_{\tau,s_{1:M}}^{k+j+1} = P(\tau^{k+j+1}, \overline{D}_{1:M}^{k+1:k+j} | s_{1:M}^{1:k+N}, z_{1:M}^{1:k}) = \tag{7}$$

$$= \sum_{\tau^{k+j}\in\Omega} P(\tau^{k+j+1}, \tau^{k+j}, \overline{D}_{1:M}^{k+1:k+j} | s_{1:M}^{1:k+N}, z_{1:M}^{1:k}) \tag{8}$$

$$= \sum_{\tau^{k+j}\in\Omega} P(\tau^{k+j+1}|\tau^{k+j}) P(\tau^{k+j}, \overline{D}_{1:M}^{k+1:k+j} | s_{1:M}^{1:k+N}, z_{1:M}^{1:k}) \tag{9}$$

$$= \sum_{\tau^{k+j}\in\Omega} T(\tau^{k+j}) P(\overline{D}_{1:M}^{k+j} | \tau^{k+j}, s_{1:M}^{k+j}) \cdot \cdot P(\tau^{k+j}, \overline{D}_{1:M}^{k+1:k+j-1} | s_{1:M}^{1:k+N}, z_{1:M}^{1:k}) \tag{10}$$

$$= \sum_{\tau^{k+j}\in\Omega} T(\tau^{k+j}) \prod_{i=1}^{M} \overline{O}(\{\tau, s_i\}^{k+j}) f_{\tau,s_{1:M}}^{k+j} \tag{11}$$

A.2 TET Utility Function

As TET is a truncated version of the Expected Time (ET), in the following expression we show how to obtain ET, supposing, when required, the same independencies as in the previous case. According to [9, 20], ET can be calculated with Eq. (12) where $P(k \leq t \leq k+j)$ represents the probability of finding the target at any time between k and $k+j$, which is calculated as $P(\bigcup_{i=1:M,l=1:l} D_i^{k+l} | s_{1:M}^{1:k+\infty}, z_{1:M}^{1:k}))$ in Eq. (13). Besides, operation $\cup \rightarrow \cap$ is applied from Eq. (13) to Eq. (14), a marginalization over τ^{k+j} from Eq. (14) to Eq. (15), the BR from Eq. (15) to Eq. (16), and the measurement independence assumption plus the sensor model alternative expression from Eq. (16) to Eq. (17). Finally, in TET (Eq. (3)) we substitute ∞ by N in the external summation.

$$\mathrm{ET}(s_{1:M}^{k:k+\infty}) = \sum_{j=1}^{\infty} (1 - P(k \leq t \leq k+j)) \tag{12}$$

$$= \sum_{j=1}^{\infty} (1 - P(\bigcup_{i=1:M,l=1:j} D_i^{k+l} | s_{1:M}^{1:k+\infty}, z_{1:M}^{1:k})) \tag{13}$$

$$= \sum_{j=1}^{\infty} P(\overline{D}_{1:M}^{k+1:k+j} | s_{1:M}^{1:k+\infty}, z_{1:M}^{1:k}) \tag{14}$$

$$= \sum_{j=1}^{\infty} \sum_{\tau^{k+j}\in\Omega} P(\overline{D}_{1:M}^{k+1:k+j}, \tau^{k+j} | s_{1:M}^{1:k+\infty}, z_{1:M}^{1:k}) \tag{15}$$

$$= \sum_{j=1}^{\infty} \sum_{\tau^{k+j}\in\Omega} P(\overline{D}_{1:M}^{k+j} | \tau^{k+j}, s_{1:M}^{k+j}) \cdot \cdot P(\overline{D}_{1:M}^{k+1:k+j-1}, \tau^{k+j} | s_{1:M}^{1:k+\infty}, z_{1:M}^{1:k}) \tag{16}$$

$$= \sum_{j=1}^{\infty} \sum_{\tau^{k+j}\in\Omega} \prod_{i=1}^{M} \overline{O}(\{\tau, s_i\}^{k+j}) f_{\tau,s_{1:M}}^{k+j} \tag{17}$$

A.3 DD Utility Function

As the DD utility function (Eq. (5)) is an adaptation of the detection probability (Eq. (4)), in the following expression we demonstrate the last. Operation $\cup \rightarrow \sum$ is applied to go from Eq. (18) to Eq. (19), a marginalization over τ^{k+j} from Eq. (19) to Eq. (20), the BR from Eq. (20) to Eq. (21), operation $\cup \rightarrow \cap$ from Eq. (21) to Eq. (22), and the measurement independence assumption plus sensor model alternative expression from Eq. (22) to Eq. (23).

$$P(\bigcup_{i=1:M,j=1:N} D_i^{k+j} | s_{1:M}^{1:k+N}, z_{1:M}^{1:k}) = \tag{18}$$

$$= \sum_{j=1}^{N} P(\bigcup_{i=1:M} D_i^{k+j}, \overline{D}_{1:M}^{k+1:k+j-1} | s_{1:M}^{1:k+N}, z_{1:M}^{1:k}) \tag{19}$$

$$= \sum_{j=1}^{N} \sum_{\tau^{k+j}\in\Omega} P(\bigcup_{i=1:M} D_i^{k+j}, \tau^{k+j}, \overline{D}_{1:M}^{k+1:k+j-1} | s_{1:M}^{1:k+N}, z_{1:M}^{1:k}) \tag{20}$$

$$= \sum_{j=1}^{N} \sum_{\tau^{k+j}\in\Omega} P(\bigcup_{i=1:M} D_i^{k+j} | \tau^{k+j}, s_{1:M}^{k+j}) \cdot \cdot P(\tau^{k+j}, \overline{D}_{1:M}^{k+1:k+j-1} | s_{1:M}^{1:k+N}, z_{1:M}^{1:k}) \tag{21}$$

$$= \sum_{j=1}^{N} \sum_{\tau^{k+j}\in\Omega} \left(1 - P(\overline{D}_{1:M}^{k+j} | \tau^{k+j}, s_{1:M}^{k+j})\right) f_{\tau,s_{1:M}}^{k+j} \tag{22}$$

$$= \sum_{j=1}^{N} \sum_{\tau^{k+j}\in\Omega} \left(1 - \prod_{i=1}^{M} \overline{O}(\{\tau, s_i\}^{k+j})\right) f_{\tau,s_{1:M}}^{k+j} \tag{23}$$