Maximizing Mutual Information for Multipass Target Search in Changing Environments

Michael J. Kuhlman[1], *Student Member, IEEE*, Michael W. Otte[2] *Member, IEEE*
Donald Sofge[3], *Member, IEEE*, and Satyandra K. Gupta[4], *Senior Member, IEEE*

***Abstract*—Motion planning for multi-target autonomous search requires efficiently gathering as much information over an area as possible with an imperfect sensor. In disaster scenarios and contested environments the spatial connectivity may unexpectedly change (due to aftershock, avalanche, flood, building collapse, adversary movements, etc.) and the flight envelope may evolve as a known function of time to ensure rescue worker safety or to facilitate other mission goals. Algorithms designed to handle both expected and unexpected changes must: (1) reason over a sufficiently long time horizon to respect expected changes, and (2) replan quickly in response to unexpected changes. These ambitions are hindered by the submodularity property of mutual information, which makes optimal solutions NP-hard to compute. We present an algorithm for autonomous search in changing environments that uses a variety of techniques to improve both the speed and time horizon, including using ϵ-admissible heuristics to speed up the search.**

I. Introduction

When earthquakes and other disasters occur in remote areas, damage to transportation infrastructure can hinder search efforts by limiting ground access to the affected areas. Search for survivors is becoming increasingly supported by micro aerial vehicles (MAVs) as MAVs become cheaper and easier to deploy. For example, in the 2015 earthquake in Nepal, Canadian relief teams used 3 small MAVs to assess the area and direct relief efforts despite a lack of ground access to villages [1]. We consider a related scenario in which aerial vehicles fly at low altitudes, navigating rubble and other obstacles, through ruined buildings suspected of containing survivors. A critical yet previously unaddressed aspect of this scenario is that disaster environments are subject to change. Access to some areas may be unexpectedly lost due to aftershock, avalanche, flood, building collapse, etc., or gained as rescue teams clear pathways to unexplored areas. MAVs may also be required to follow a schedule of changing no-fly zones to share space with human rescue teams.

[1]M.J. Kuhlman is with the Department of Mechanical Engineering and the Institute for Systems Research, University of Maryland, College Park, MD 20742, USA mkuhlman@umd.edu

[2]M.W. Otte is a National Research Council RAP postdoc at Naval Research Laboratory, Washington, DC 20375, USA

[3]D. Sofge is with Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Washington, DC 20375, USA donald.sofge@nrl.navy.mil

[4]S.K. Gupta is with the Aerospace and Mechanical Engineering Department and the Center for Advanced Manufacturing at the University of Southern California, Los Angeles, CA 90089, USA skgupta@usc.edu

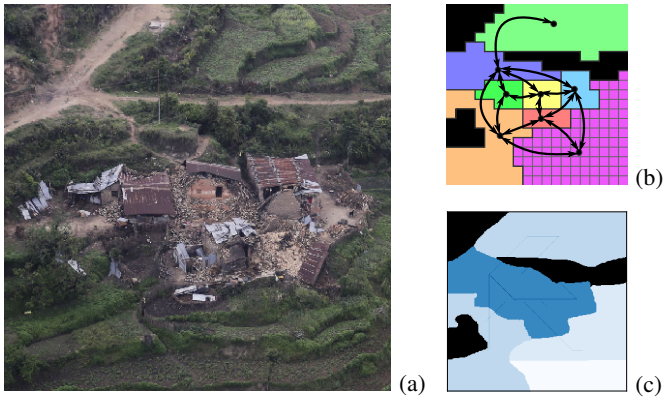


Fig. 1. Multipass coverage plan example for the 2015 Nepal Earthquake. (a) Aerial view. (b) Regions (color) with grids depicted in the lower-right region (grid size has been enlarged for illustration). (c) The resulting multi-pass search over the area at full resolution (color indicates pass count).

For this work we will focus on the following aspects of changing environments: targets may appear or disappear from view over time, and access to regions of the environment may only be available intermittently but known *a priori*. Formulating multi-target search in terms of maximizing mutual information allows us to formally reason about the tradeoffs inherent to a multi-pass search. For example, performing additional passes over a particular area provides a better understanding of that area given noisy sensors, but each subsequent pass has diminishing returns with respect to the amount of new information that is obtained. Maximizing mutual information in changing environments presents a variety of challenges. Algorithms must run quickly enough to react on-the-fly to unexpected changes, while using long enough time horizons to accurately model environmental changes that are scheduled *a priori*.

Many previous information theoretic techniques are insufficient to solve this type of problem. Both greedy approaches [2]–[4] and model predictive control [5] can react quickly enough to deal with unexpected changes, but are myopic and may get trapped by long-term changes—even long-term changes known *a priori*. However, there exist bounds in which greedy algorithms perform compared to the optimal solution [6,7]. Partially observable Markov decision processes (POMDPs) account for uncertainties [8], but suffer from a computational complexity that makes them ill-equipped to

deal with large environments that change in *unanticipated* ways. While discrete [9] or sample based information search algorithms [10] are capable of finding resolution optimal or asymptotically optimal solutions, respectively, they have previously only been applied in very simple static environments where the computational complexity of calculating mutual information scales exponentially with respect to time horizon.

In this paper we show how discrete search algorithms can be modified to enable planning through changing environments in a more efficient manner. First, we use a hierarchical data structure that groups similar and contiguous search grids into sets called *regions*. Branch and bound search is performed over regions instead of grids, which enables the time spent finding the first solution to be *linear* with the size of the evidence grid instead of exponential (it is still exponential with respect to the number of regions, which is deliberately kept as small as possible). We also exploit an equivalence that exists between different paths that cover the same search area, to prune redundant portions of the search space. In addition to using known bounding heuristics to speed branch and bound search, we developed a priority queue sorting heuristic that speeds search by biasing search effort in a beneficial way. We also experiment with an ϵ-admissible heuristic that dramatically increases search speed by about 2 orders of magnitude. Finally, we show that for evidence grids, mutual information can be implemented as a lookup table eliminating the complexity of computing the reward. These techniques are also applicable in static environments, and many of them (such as the ϵ-admissible heuristic) have not previously appeared in the information theoretic path planning literature.

II. Related Work

Probabilistic target search appears as early as 1956 [11]. Many variations of target search have been studied across a variety of communities, and surveys can be found [12]–[14]. We now discuss the subset of this previous work that is closely related to the current paper. We assume that an agent actively searches for targets until, possibly, a stopping criteria is met. Therefore, our survey of previous work will focus on problems requiring active search [15,16] (e.g., vs. the alternative problem where searchers are expected to travel to and then remain at locations well suited to static surveillance [17].

When (only) a single target exists, a common approach is to maintain a probability density function of the target's location within the search space that is refined analytically [5] or numerically [18] with respect to new observations taken as the searcher(s) move. This can be extended to the multi-target case by tracking either a predefined number m of targets in parallel [5,19]–[21], or using random finite sets to simultaneously define the likelihood that there are $1, \ldots, m$ targets [3,15]. An alternative (but not mutually exclusive) location-centric approach involves dividing the search space into a finite number of cells or grids, and then tracking the likelihood each cell/grid contains a target [3,22]–[24] or tracking the number of targets contained within each cell/grid [25,26]. Our work is similar to the cell/grid method, except that we model the world as a connected set of regions, and each region contains multiple cells.

Another consideration is if target(s) are assumed stationary, e.g., [3,4,18,24], or move according to a stochastic process [16,19] or other dynamics [5]. In this work we assume that targets are stationary.

Methods grounded in Bayesian inference have been widely studied [20,21,23,27,28], and are useful when target motion or sensor properties can be modeled probabilistically. Other methods inspired by probabilistic ideas calculate a heuristic value between 0 and 1 from the number of detections vs. attempted measurements to provide an estimate of how likely a cell is to contain a target [29]–[31].

If a cell/grid model is used such that the probability of target existence within each cell is the desired result of a search, then information theoretic methods (often grounded in Bayesian inference [2,4,18,32]) provide a means to reason about what parts of the search space are understood well or poorly, and how searcher movement along a potential search plan is likely to change the overall quantity of information we have about the target(s) locations within the environment [18]. Our work falls into this category.

The information theoretic benefits of visiting different locations have the property of submodularity (i.e., diminishing returns), and calculating the information theoretically optimal motion plan becomes intractable as the length of the plan increases. Searcher(s) often select joint actions that maximize expected information gain over a receding horizon to achieve tractability [5,15,16,18,20,28,32]. Computational complexity increases with horizon distance, but is often reduced in practice by greedily following the maximum gradient of mutual information [2]–[4]. Multiagent methods are exponentially complex with respect to the number of robots, and necessarily use shorter look-ahead distances as a result [2]–[4]. While our work plans over a fixed horizon, we consider a much deeper planning horizon than previous work. By performing branch and bound over a graph of regions (themselves composed of many cells) we are able to achieve much greater cell-wise look-ahead. We consider only a single searcher.

A hierarchical search is considered in [3] and mentioned as a possibility of future work in [4]. Both [3,4] envision the hierarchy as a quad-tree like data structure such that cells/grids are broken into multiple sub-grids based on a user defined threshold on the information contained in the cell/grid or target likelihood of existing within a cell. This contrasts with our approach (in which the grid vs. region resolutions are defined *a priori* and each region contains many cells), but we believe it would be possible to combine all of these ideas, but doing so is beyond the scope of the current paper.

In addition to work with moving targets [5,16,19], other work considering environmental dynamics includes [4,30] and focus on hostile environments that can destroy robots. These methods assume multiple robots exist, and focus on reaction to changing team size and accumulating knowledge about hazard locations. In contrast, we are interested in environmental

connectivity that changes with time. The term "uncertain environments" is used in [32] in the sense that the locations of targets are not known *a priori*, although the same thing could be said about most previous work. A time-varying information objective was explored in [10], when using sampling based motion planning for environmental information gathering.

Previous applications of branch and bound to the target search problem include [9,19,33]. The method presented in [19] considers the case of tracking a stochastically moving target and attempts to maximize the expected number of detections, while [33] considers a moving target but seeks to maximize the probability of target detection subject to fuel and risk constraints. In contrast to [19,33], we seek to maximize information regarding the location of multiple stationary targets. The most closely related work to our own is arguably [9] which uses branch and bound with Gaussian Processes for informative path planning. A significant difference is that the bounding heuristic used in [9] cannot account for multiple coverings of the search space—something we explicitly allow.

III. Problem Formulation and Approach

The workspace of the robot $W \subset \mathbb{R}^2$ contains all positions $w = (x, y)$ at which a target may exist. The configuration space C is the product of W with time T and any higher order spaces necessary to define robot trajectories, $C = W \times \dot{W} \times \ldots \times T$. The robot starts at point $c_{\text{start}} \in C$. A point in space time is denoted $(w, t) \in W \times T$. A feasible trajectory $\mathcal{T}$ is a curve through C that starts at c_{start} and that the robot is capable of following. A trajectory defined between time 0 and time t is denoted $\mathcal{T}_t$. Let $\mathscr{T}_t = \bigcup\{\mathcal{T}_t\}$ be the space of all valid trajectories between time 0 and t.

Information related to target search is stored in an evidence grid $\mathcal{E}$ which covers the workspace at a user defined resolution r. For notational convenience we assume $r = 1$ and evidence grid cell $g_{x,y}$ represents the set $[x, x + r) \times [y, y + r)$ for $x, y \in \mathbb{N}$ and $1 \leq x \leq x_{\max}$ and $1 \leq y \leq y_{\max}$, where $x_{\max}$ and $y_{\max}$ depend on the problem being solved. As the robot moves, the state of the evidence grid is updated as a function of measurements that it takes at cells. We assume measurements are taken at discrete times $t = 0, 1, 2, \ldots$.

For each cell $g \in \mathcal{E}$ define X to be the hidden state of whether or not the cell contains a target. X may be discrete or continuous. $Z_k \in \{0, 1\}$ is the kth sensor measurement in g and $\mathbf{Z}_{1:k}$ is the collection of k sensor measurements. $\mathbf{z}_{1:k}$ is an observed sequence of measurements (a realization of $\mathbf{Z}_{1:k}$). When it is necessary to denote the particular grid cell in which sensor measurements are taken we shall use superscripts, e.g., $\mathbf{z}^{x,y}_{1:k}$ is the observed sequence of measurements in $g_{x,y}$. Each cell in the evidence grid is assumed to be independent from all other cells, that is $X^{i,j} \perp X^{x,y}$ for $i \neq x$ and $j \neq y$. For all x, y, k we assume that $X^{x,y}$ is independent of time and thus $Z^{x,y}_k$ and $\mathbf{z}^{x,y}_{1:k}$ are not affected by the time(s) at which the measurements are taken. Given the forward sensor model $P(Z_k = z_k|X)$, we compute the inverse sensor model, (1):

$$P(X|\mathbf{z}_{1:k}) = \frac{P(Z_k = z_k|X, \mathbf{z}_{1:k-1})P(X|\mathbf{z}_{1:k-1})}{P(Z_k = z_k|\mathbf{z}_{1:k-1})} \tag{1}$$

in which all sensor measurements are conditionally independent. From this model it is possible to compute the mutual information $I(Z_{k+1}; X|\mathbf{z}_{1:k})$. In practice, this can be computed *a priori* for any process and sensor model, and tabulated on the number of positive and negative sensor measurements observed. This model can account for either static (consider the occupancy grid model) or stationary targets (target's presence at any moment of time is the outcome of a Bernoulli Process, but the target's distribution is independent of time). The outlined approach in this paper would need to be modified for nonstationary targets (whose target distribution is time dependent such as a Markov chain) or dynamic targets (whose belief distribution may be in more than one cell). We will assume the use of an occupancy grid model [34] which has closed form solutions through judicious use of basic properties from probability and information theory but use of other process/sensor models lie beyond the scope of this paper. Define submodular function $I(\mathbf{Z}_{k+1:k+q}; X|\mathbf{z}_{1:k})$ to be the total mutual information gathered by q future sensor observations. We note that in this approach, mutual information is used to predict the total information gain of future trajectories conditioned on any previous sensor measurements observed.

The sensor model also defines a *trajectory footprint* (or, footprint) for a given trajectory:

$$\Phi_{\mathcal{T}} = \{\cup_{x,y}\mathbf{Z}^{x,y}_{k+1:k+q} | \mathcal{T} \text{ makes } q \text{ observations at cell } g_{x,y}\}$$

where different sensors have different footprints (e.g., downward facing camera vs. scanning LIDAR). $\Phi_{\mathcal{T}}$ can be stored as a 2D array the same size as the evidence grid by defining $\Phi_{\mathcal{T}}[x, y] = q$. The cumulative mutual information collected by time t assuming the robot has followed $\mathcal{T}_t$ is given by the sum of the mutual information of the future sensor measurements and the hidden state conditioned on previously observed data within each individual grid:

$$R(\mathcal{T}_t) = \sum_{\mathbf{Z}^{x,y}_{k+1:k+q} \in \Phi_{\mathcal{T}}} I(\mathbf{Z}^{x,y}_{k+1:k+q}; X^{x,y}|\mathbf{z}^{x,y}_{1:k})$$

The problem of informative path planning with time constraints is defined as: Find $\mathcal{T}_t^*$, the trajectory of time duration t that maximizes the cumulative mutual information in $\mathcal{E}_t$,

$$\mathcal{T}_t^* = \arg\max_{\mathcal{T}_t \in \mathscr{T}_t} R(\mathcal{T}_t). \tag{2}$$

When computing the lookup table for mutual information, it holds that $I(\mathbf{Z}_{k+1:k+q}; X|\mathbf{z}_{1:k})$ can be computed from $I(Z_{k+1}; X|\mathbf{z}_{1:k})$ using the chain rule for mutual information:

$$I(\mathbf{Z}_{k+1:k+q}; X|\mathbf{z_{1:k}}) = \sum_{i=k+1}^{k+q} I(Z_i; X|Z_{i-1}, Z_{i-2}, \ldots, Z_{k+1}, \mathbf{z_{1:k}})$$

Using the exchangability property we define Binomial Random Variable $M_{k+1:i} = \sum_{j=k+1}^{i} Z_i$. Given $\mathbf{z}_{1:k}$, we can assign $n_{z1} = m_{1:k} = \sum_{i=1}^{k} z_i$ and $n_{z0} = k - m_{1:k}$ to be the number of positive and negative sensor readings, respectively.

$$I(\mathbf{Z}_{k+1:k+q}; X|\mathbf{z_{1:k}}) = \sum_{i=k+1}^{k+q} \sum_{m=0}^{i-k} P(M_{k+1:i} = m) I(Z_i; X|\mathbf{z_{1:i}})$$

This results in $\mathcal{O}(q^3)$ multiplications and $\mathcal{O}(q^3)$ calls to $I(Z_i; X|\mathbf{z})$, assuming such data is available efficiently. However, storing these solutions into a 3D lookup table $I[n_{z0}, n_{z1}, q]$ indexed by unsigned integers offers $\mathcal{O}(1)$ lookup speeds in random access memory and requires little memory (a 20x20x10 array of double precision floats takes 32 kilobytes of RAM).

A. Trajectory Equivalence

The computation time required to solve (2) can be significantly decreased vs. a naive search over all $\mathcal{T}_t \in \mathscr{T}_t$ as a consequence of our assumption that $X^{x,y}$ is independent of time for all x, y. Consequently, the order in which observations are recorded in $\Phi_{\mathcal{T}_t}$ does not affect the information gathered. Two trajectories are in the same equivalence class if their footprints and end configurations are equal. Further, if the set of future feasible trajectories is uniquely determined by the current configuration, then two trajectories belonging to the same equivalence class generate the same future feasible trajectories. This implies that we may prune a node in the search tree if it is found to be equivalent to another node in the queue, meaning that we only need to store one trajectory per equivalence class without affecting the quality of the final solution. In practice, this is accomplished by hashing all $(\Phi_{\mathcal{T}_t}, c)$ that we encounter to see if a new node generates a new equivalence class.

B. Search over regions instead of grids

Under special circumstances that are often reasonable in practice, the depth of the search for $t_{\max}$ can be decreased by orders of magnitude by searching over a hierarchical representation of W (e.g. Voronoi partition, visibility region, etc.). This is possible whenever the workspace can be divided into a set of disjoint regions $\mathcal{R}$ where each region contains a mutually exclusive contiguous subset of evidence grid cells, and each grid cell appears in exactly one region. See Fig. 1(b) for a set of regions with an example evidence grid.

Different trajectories are generated by selecting different actions, e.g. by either searching a given region, or by traversing between connected regions. The edges connecting regions may also be *time dependent*, modeling changing environmental connectivity such as no-fly zones, etc. We also assume that all trajectories start and stop at one of the nodes in the graph.

C. ϵ-admissible Branch and Bound Algorithm

We extend best first search branch and bound using a priority queue to create ϵ-admissible branch and bound. For general branch and bound for maximization problems (similar formulations exist for minimization problems), we want to maximize function $f(x)$ for $x \in X$ for solution space X with admissible heuristic $g(N)$ for arbitrary set $N \subseteq X$ such that $g(N) \geq \max_{x \in N} f(x)$, enabling branch and bound to reason about subsets of the solution space. The reward for the best known feasible solution is stored in the quantity B, initialized to $-\infty$ or heuristic solution B_0. Branching the node N generates multiple subspaces $N_i \subset N$. The branching function branch(N) will eventually generate all feasible solutions $\{x\} \in X$. Bounding determines if any solution in the subset defined by N can potentially beat the current best known solution, if $g(N) > B$. If $g(N) \leq B$, the optimal solution cannot be in N so the algorithm ignores the subset N for the rest of the search (it is said that branch and bound has *fathomed* N in this case). If a node has not been fathomed, the priority of the node, $P(N)$, is evaluated and N is placed on the priority queue. If $N = \{x\}$ where x is a complete feasible solution, then $B \leftarrow f(x)$ if $f(x) > B$. The algorithm terminates when the priority queue is empty.

***Theorem** 1:* branch and bound is complete when X is finite and is guaranteed to find the optimal solution.

Proof: Proof for an equivalent formulation can be found in Edelkamp and Shroedl [35]. □

For multipass target search, branch and bound is maximizing $f = R$ defined in (2) with $X = \mathscr{T}_t$. Each node or subset N in the search tree corresponds to a subset of solutions and is defined by $N = (c_t, \Phi_{\mathcal{T}_t})$. Branching of a node is done by using the motion model to generate children nodes for all available actions. Section III-D will define an admissible heuristic for use in branch and bound.

Suppose that instead of using an admissible heuristic we wish to speed up the search by more aggressively fathoming the search space. Define ϵ-admissible bounding heuristic $\tilde{g}(N) \geq \max_{x \in N} f(x) - \epsilon$ which underestimates the reward $f(x)$ by at most ϵ. This is closely related to Harris's *e-admissibility* criterion for bandwidth search [36]. ϵ-admissible branch and bound is therefore defined as branch and bound that uses an ϵ-admissible heuristic for bounding.

***Theorem** 2:* Using ϵ-admissible heuristic $\tilde{g}$ for bounding guarantees that the solution will be at most ϵ less than the optimal solution.

Proof: The only way this algorithm will not find the optimal solution x^* is if any subset N_j, where $x^* \in N_j$, is pruned. Suppose $\tilde{x}$ is a feasible solution and $f(\tilde{x}) = B$. Given that N_j is pruned, $\tilde{g}(N_j) \leq B$ but $\tilde{g}(N_j) \geq f(x^*) - \epsilon$. Further manipulation yields $\epsilon \geq f(x^*) - f(\tilde{x}) \geq 0$. □

D. Iterative Greedy Heuristic for Bounding

The major challenge for the bounding heuristic is to reason over potential future reward any partial solution can gather. A common approach to generate a heuristic is to relax the trajectory feasibility requirements by assuming that all actions are feasible at any given time, while assuming that the actions do not interact with each other (e.g. footprints do not overlap). We would like to maximize reward gathered for an action sequence of fixed duration, with any combination of actions feasible in any order. Suppose there are L actions to be made, and N actions available. Decision variable $a_i \in \{0, 1, \ldots, L\}$ with $i \in 1, \ldots, N$ is the number of times the ith action is taken, with decision vector $\mathbf{a} = (a_1, a_2, \ldots a_N)$. The reward is partitioned off in the sense that $R(\mathbf{a}) = \sum_{i=1}^{N} R_i(a_i)$. Define $\Delta R_i(a+1) = R_i(a+1) - R_i(a) \geq 0$. This is equivalent to saying that selecting action i the $(a_i + 1)$th time gives the agent a reward of $\Delta R_i(a_i+1)$. R_i is monotone increasing and

concave ($R_i(a+1) \geq R_i(a)$ but $\Delta R_i(a+2) \leq \Delta R_i(a+1)$). The goal is to maximize the objective function:

$$\max_{\mathbf{a}} \sum_{i=1}^{N} R_i(a_i) \quad \text{s.t.} \quad \sum_{i=1}^{N} w_i a_i \leq L \text{ and } a_i \geq \alpha_i \geq 0 \qquad (3)$$

where $w_i = 1$ implies that all actions are of unit duration α_i encodes subproblems for branch and bound by determining which actions have already been selected. Looking at this as a sequence of L decisions with $w_i = 1$, greedily selecting the next action will yield the optimal solution by construction. For example, if one were to sort *all actions* by their incremental reward $\Delta R_i(a_i)$, picking the L largest rewards would maximize the objective function. When sorted like this, $\Delta R_i(a_i)$ is always selected before $\Delta R_i(a_i + 1)$ by definition so a valid sequence of actions is always generated (pick action i the first time, pick action j the first time, then pick action i a second time, etc.). This relaxation of the original path planning problem generates an admissible heuristic satisfying the requirements for Theorem 1.

E. Priority Heuristic

We wish to discount future expected reward since future rewards are overestimated. For priority $p(N)$ for node N, current reward $R(N)$ heuristic $g(N)$, and discount factor $\alpha \in [0, 1]$, define the priority of a node to be:

$$P(N) = R(N) + \alpha(g(N) - R(N)) \qquad (4)$$

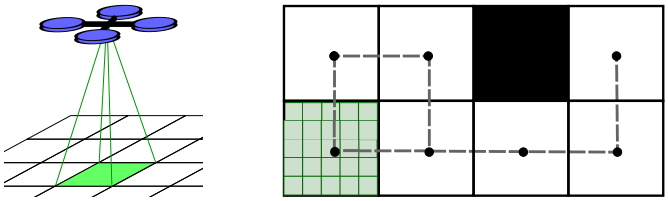
F. Benchmarks: Greedy Planner and Zelinsky's Algorithm

We contrast our approach to two alternate approaches, using an iterative greedy heuristic solver and Zelinsky's algorithm when applicable. For the greedy heuristic solver, we use the same environment and motion model that's used for branch and bound to generate candidate trajectories. However, the algorithm instead greedily maximizes total information gathered per unit of time for the next action. When the greedy heuristic is selecting the next best action, it is possible (but unlikely in natural environments) that two or more actions result in the same (maximal) amount of information gathered per unit of time. Such ties are resolved by selecting the action that acquires more total information. This is done iteratively until the remaining time budget expires.

Zelinsky's algorithm [37] is a value function based uniform coverage planning algorithm that plans directly on the 8-connected evidence grid. It may occasionally back itself into a corner, in which case we use Dijkstra's algorithm to plan a route to the nearest unvisited cell, and then continue running Zelinsky's algorithm. We note that Zelinsky's algorithm does not extend to space time and so cannot be applied to changing environments

IV. Experiments

For all experiments, the evidence grid is a 200x100 grid unless mentioned otherwise. We use the standard occupancy grid sensor model [34] where $X = 0$ or $X = 1$. $P(Z|X)$ is fully characterized by $P(Z = 1|X = 1) = p_d$ where $p_d = 0.85$ is the probability of detection and $P(Z = 1|X = 0) = p_f$ where $p_f = 0.15$ is the false positive rate. We assume that the quadrotor only senses the cell it is currently in, see Fig. 2(a).



(a) sensor model used in the simulations

(b) Example environment denoting regions, the connectivity graph and a portion of the evidence grid.

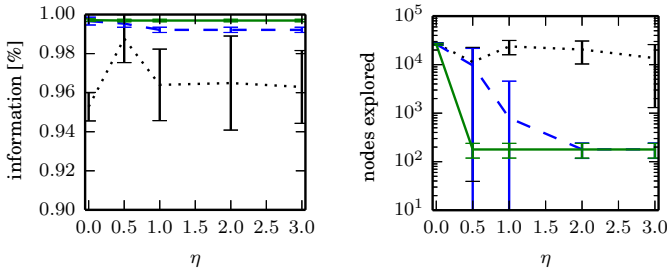
Fig. 2. 2(a): Sensor model observes a single cell in the occupancy grid. 2(b): example 7 region environment showing regions (black lines), their nodes (black circles) and edges (gray dashed lines) overlaying the evidence grid. Example portion of the evidence grid is shown in the bottom left region.

We procedurally generate environments randomly by creating rectangular regions that produce a tiling on the evidence grid. We randomly remove a fixed set of individual regions while maintaining graph connectivity. Environments will consist of either 12, 24, or 50 regions within the evidence grid. The 12 region environment started with 4x4 regions, while the 24 and 50 region environments started out with 8x4 and 10x10 regions respectively. Note that in this case, the 12 and 24 region environments have 15k cells accessible for observation, while in the 50 region case there are 10k accessible cells. The agent has the following actions: (1) traverse between connected regions, e.g. move left, down; (2) exhaustively search a region, or search the region until the time horizon expires. Each action has a given time duration and a trajectory footprint where one unit of time is required to traverse and sense one cell, and starts and ends at one of the graph nodes.

While our sensor model measures mutual information in bits, each environment doesn't necessarily contain the same amount of information. For each algorithm we will denote information gathered as the ratio of how much information the planned path will gather divided by the heuristic total information that could be potentially gathered (ignoring traversal times or motion constraints). Note that any ratio $\geq 100\%$ is not possible.

For our application, we use the ϵ-admissible heuristic $\tilde{g}(x) = g(x) - \epsilon$ where $\epsilon = \eta B$, where η is the minimum required performance improvement (e.g. $\eta = 0.5\%$ requires that new solutions are half a percent better than the current best solution). B is initialized to $-\infty$ as we do not start with a heuristic solution.

For each experiment $N_{trials} = 40$ different environments using the randomized environment generator were tested. Simulations were run on a Core i7-6920HQ with a 2.9 GHz clock with 64 GB RAM running 64 bit Ubuntu 14.04 LTS. The proposed branch and bound algorithm and the greedy algorithm were developed in Python, using NumPy for array operations and networkx for graph operations. Zelinsky's algorithm was implemented in the programming language



(a) information gathered (b) nodes expanded

Fig. 3. Results for Experiment 1. $(\alpha, \eta) = (0.9, 0.5)$ reduces the number of nodes expanded while not affecting solution quality. Green/solid is $\alpha = 0.9$, blue/dashed is $\alpha = 0.7$ and black/dotted is $\alpha = 0.5$.

Julia. We ran the following experiments to illustrate different properties of the proposed algorithm:

- Experiment 1: Sweep α, η on the 12 region environment to determine best parameters, and illustrate that the choice of priority heuristic and that ϵ-admissible bounding improve the performance of the planner.
- Experiment 2: Sweep the size of the evidence grid to show (in-)sensitivity to size of evidence grid
- Experiment 3: For set values of α, η, benchmark branch and bound on 12, 24, 50 region environments to greedy approach and Zelinsky's algorithm in uniform and nonuniform environments, and for static and time varying environments. This also shows how the complexity of the algorithm increases for more complex environments.

For uniform environments, all cells are initialized with the most uninformative prior of $P(X = 1) = \theta_0 = 0.5$. For the nonuniform environment, we will suppose that half of the regions (selected at random) have been visited twice by prior sensor measurements. Inspection of the mutual information look up table suggests that in this case, a nonuniform covering will gather more information. For the time varying environment in Experiment 3, we select 4 regions that are critical locations (their removal breaks the connectivity of the environment) at random and turn them into trap doors. All edges leading into or out of the region are disabled at regular time intervals. Each on/off cycle lasts 20% of the mission duration.

A. Experiment 1

We set $\alpha \in \{0.5, 0.7, 0.9\}$ and $\eta \in \{0.0, 0.5, 1.0, 2.0, 3.0\}$ in a 12 region environment. For these trials, the planner continues to plan until the priority queue is empty. Fig. 3(a) shows what percentage of information the planner is able to collect for given parameter set (α, η). We see that for $\alpha \geq 0.7$ collected information is insensitive to choice of η (this stems from the fact that for higher values of α, the first discovered solution is usually the best). In Fig 3(b) we observe how tweaking the heuristic used for ϵ-admissible bounding dramatically reduces the number of nodes expanded. For example setting $(\alpha = 0.9, \eta = 0.5)$ only explores 0.6% of the number of nodes explored when compared with setting $(\alpha = 0.9, \eta = 0.0)$.

TABLE I
UNIFORM STATIC ENVIRONMENTS, MEAN OF 40 TRIALS

Regions	Branch and Bound	
	info	time
12	99.2% ± 0.1%	1.15 ± 0.4
24	99.2% ± 0.1%	6.5 ± 6.9
50 (83% succ.)	99.2% ± 0.1%	128 ± 135

Regions	Greedy		Zelinsky	
	info	time	info	time
12	72% ± 11%	0.2 ± 0.03	99.7% ± 0.08%	< 0.1*
24	84% ± 9%	0.3 ± 0.02	99.5% ± 0.1%	< 0.1*
50	52% ± 19%	0.94 ± 0.18	78% ± 0.1%	< 0.1*

B. Experiment 2

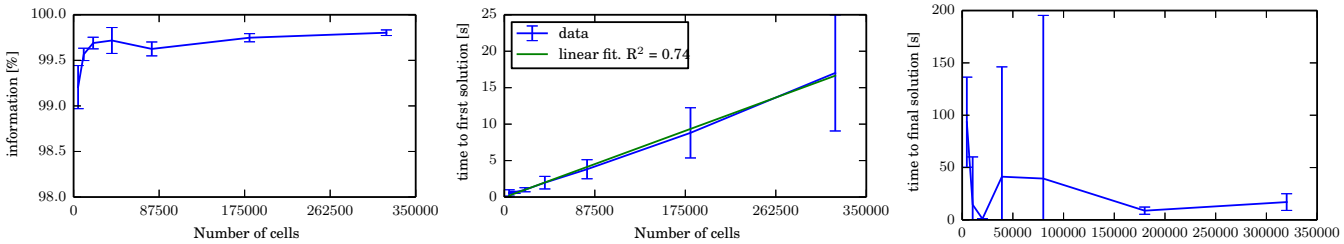
The results of Experiment 2 show that the time to the first solution is linear with the number of cells in the evidence grid (Fig. 4(b)) without affecting solution quality (Fig. 4(a)). This is due to the hierarchical nature of the formulation instead of planning directly in the evidence grid (which would be exponential in the number of grid cells). Time to algorithm termination (Fig. 4(c)) is not linear with the number of cells in part due to the effects of introducing caching in the heuristic. Further, the quality of the heuristic improves in larger environments since traversal times become insignificant with larger regions with longer search times.

C. Experiment 3

Fixing $\alpha = 0.8$ and $\eta = 0.5\%$, we will benchmark the proposed branch and bound algorithm to other algorithms in the 12, 24, and 50 region environments. Since we are pushing the proposed algorithm to the limits, branch and bound will not always find a solution for the more difficult environments. When branch and bound does not always find a solution, we will mention the success rate and provide statistics on the successful trials. We anticipate that future improvements in heuristic design will dramatically improve algorithm performance for the more difficult environments. Branch and bound will run until the first solution is found, or until 10,000 or more iterations have completed. We would like to note that Zelinsky's algorithm was implemented in the programming language Julia, which has speeds comparable to C and is much faster than Python.

Tables I, II and III summarize the statistics for the time to first solution in seconds (time), percentage of total available information gathered (info), and the total number of nodes explored (nodes), displaying averages and standard deviations.

1) Uniform Environment: Figure 5 shows example footprints that closely resemble *median* performance of each of the algorithms for the uniform environment. The footprints are the same size as the evidence grid, where white cells are unexplored by the trajectory, blue cells as being visited regions (darker cells have been visited more often), and black regions denote obstacles. Table I summarizes the statistics for the uniform, static environments.



(a) information gathered (b) time to first solution (c) time to terminate algorithm

Fig. 4. Results for Experiment 2. Runtime is linear with cell count and does not affect total information gathered by the proposed algorithm.

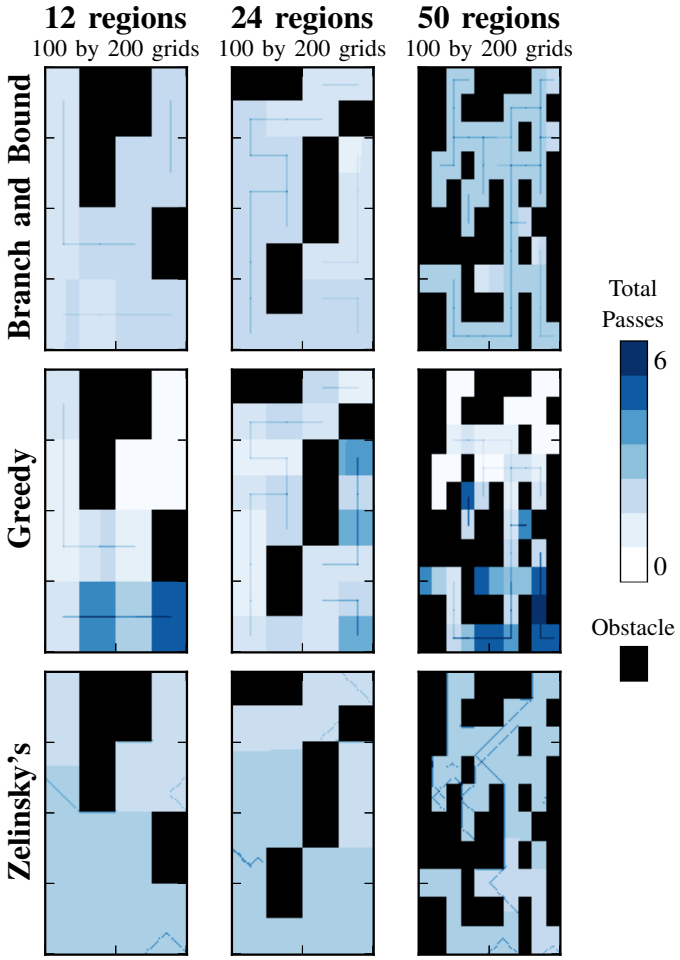


Fig. 5. Coverage in 100 by 200 grid static environments with uniform information. Different algorithms (rows) with respect to region count (columns). Even coverage is desired. The proposed branch and bound method and Zelinsky's algorithm perform well (Zelinsky's cannot be used in changing environments).

2) Nonuniform environment: We observe that branch and bound is able to handle the nonuniform environments as equally well as the uniform environments, while Zelinsky's algorithm suffers. Results are summarized in Table II.

3) Time varying environment: Experimental results in Table III illustrate that for branch and bound, solving the time varying example is significantly harder than the static environment. For example, consider that branch and bound is able to solve

TABLE II
NONUNIFORM STATIC ENVIRONMENTS, MEAN OF 40 TRIALS.

Regions	Branch and Bound	
	info	time
12	99.3% ± 0.2%	1.0 ± 0.5
24	99.3% ± 0.1%	9.6 ± 16
50 (80% succ.)	99.3% ± 0.1%	79 ± 80

Regions	Greedy		Zelinsky	
	info	time	info	time
12	70% ± 15%	0.19 ± 0.02	90% ± 1.6%	$< 0.1^*$
24	84% ± 9.4%	0.32 ± 0.03	90% ± 1.2%	$< 0.1^*$
50	72% ± 13%	0.72 ± 0.13	93% ± 0.4%	$< 0.1^*$

the static 24 region environment in 6.5 seconds on average, while taking over 58 seconds in the time varying example. More importantly, we note that there is a decrease in the success rate in the large environments We show that these success rates are sensitive to the heuristics used. For example, changing α from 0.8 to 0.2 results in dramatic improvements in success rate by encouraging depth-first like behavior, but at the expected cost of reducing solution quality.

TABLE III
UNIFORM TIME VARYING ENVIRONMENT, MEAN OF 40 TRIALS.

Regions	α	Branch and Bound	
		info	time
12	0.8	96% ± 1.7%	1.6 ± 1.5
24	0.8 (82.5% succ.)	97% ± 1.0%	58 ± 104
50	0.8 (12.5% succ.)	98% ± 0.6%	127 ± 136
50	0.2 (70% succ.)	76% ± 0.3%	73 ± 66

Regions	Greedy		Zelinsky	
	info	time	info	time
12	73% ± 14%	0.15 ± 0.01	N/A	N/A
24	76% ± 16%	0.3 ± 0.03	N/A	N/A
50	42% ± 0.2 %	0.83 ± 0.2	N/A	N/A

V. CONCLUSION

Since branch and bound reasons over all possible solutions, it discovers the highest reward path but takes the longest to compute. The ϵ-admissible heuristic shows that having a tighter bound dramatically reduces the number of nodes expanded later in the search. Zelinsky's algorithm is effective in uniform environments as expected but suffers in nonuniform environments. It is interesting to note that the greedy algorithm performs similarly in uniform, nonuniform and time varying

environments. However, the greedy algorithm is not effective enough to provide an initial heuristic solution for branch and bound as the solutions it finds are too low quality to prune any nodes during the search.

VI. Future Work

Designing the correct heuristics is essential for computing solutions quickly for target search and will be a focus for future work. We also plan to conduct experiments with MAVs searching for targets in a changing environment, which will require the generation of dynamically feasible motion primitives and coverage plans in a time-varying environment, and planning around dynamic obstacles that were not in the original plan. Finally, we plan to extend this work to a multiagent scenario by having multiple MAVs perform a coordinated search effort while sharing the information that they gather and the trajectories they plan to take in the future.

Acknowledgment

The authors would like to thank Dr. Dana Nau for his thoughts on heuristic search algorithms. This work was performed at NRL. Michael Kuhlman and Don Sofge were funded by ONR grant number N0001416WX01272, "Mobile Autonomous Navy Teams for Information Search and Surveillance (MANTISS)." Michael Otte was supported by ONR grant number N0001416WX01271.

References

[1] M. Kwong, "Nepal earthquake: Drones used by Canadian relief team," CBC News World, April 27, 2015. [Online]. Available: http://www.cbc.ca/news/world/nepal-earthquake-drones-used-by-canadian-relief-team-1.3051106

[2] B. J. Julian, M. Angermann, M. Schwager, and D. Rus, "Distributed robotic sensor networks: An information-theoretic approach," *Int. J. of Robot. Research*, vol. 31, no. 10, pp. 1134–1154, 2012.

[3] P. Dames, M. Schwager, D. Rus, and V. Kumar, "Active magnetic anomaly detection using multiple micro aerial vehicles," *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 153–160, 2016.

[4] M. Schwager, P. Dames, D. Rus, and V. Kumar, *A Multi-robot Control Policy for Information Gathering in the Presence of Unknown Hazards*. Springer Int. Publishing, 2017, pp. 455–472.

[5] J. R. Riehl, G. E. Collins, and J. P. Hespanha, "Cooperative search by UAV teams: A model predictive approach using dynamic graphs," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 47, no. 4, pp. 2637–2656, Oct. 2011.

[6] J. L. Williams, J. W. Fisher III, and A. S. Willsky, "Performance guarantees for information theoretic active inference," in *Artificial Intell. and Statistics Conf.*, vol. 2, Mar. 2007, pp. 620–627.

[7] A. Krause and C. E. Guestrin, "Near-optimal nonmyopic value of information in graphical models," *arXiv preprint arXiv:1207.1394*, 2012.

[8] N. Roy, G. Gordon, and S. Thrun, "Finding approximate POMDP solutions through belief compression," *J. of Artificial Intell. Research*, vol. 23, pp. 1–40, 2005.

[9] J. Binney and G. S. Sukhatme, "Branch and bound for informative path planning," in *IEEE Int. Conf. on Robot. and Autom.* IEEE, May 2012, pp. 2147–2154.

[10] G. A. Hollinger and G. S. Sukhatme, "Sampling-based robotic information gathering algorithms," *Int. J. of Robot. Research*, pp. 1271–1287, 2014.

[11] B. Koopman, "The theory of search. ii. target detection," *Operations Research*, vol. 4, no. 5, pp. 503–531, 1956.

[12] C. Robin and S. Lacroix, "Multi-robot target detection and tracking: taxonomy and survey," *Auton. Robots*, vol. 40, no. 4, pp. 729–760, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10514-015-9491-7

[13] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," *Auton. Robots*, vol. 31, no. 4, pp. 299–316, 2011.

[14] S. Waharte and N. Trigoni, "Supporting search and rescue operations with UAVs," in *Int. Conf. on Emerging Security Technologies*, Sept 2010, pp. 142–147.

[15] P. Dames and V. Kumar, "Autonomous localization of an unknown number of targets without data association using teams of mobile sensors," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 3, pp. 850–864, 2015.

[16] G. A. Hollinger, S. Yerramalli, S. Singh, U. Mitra, and G. S. Sukhatme, "Distributed data fusion for multirobot search," *IEEE Trans. Robot.*, vol. 31, no. 1, pp. 55–66, 2015.

[17] M. Schwager, J.-J. Slotine, and D. Rus, "Decentralized, adaptive control for coverage with networked robots," in *IEEE Int. Conf. on Robot. and Autom.* IEEE, May 2007, pp. 3289–3294.

[18] J. Tisdale, Z. Kim, and J. K. Hedrick, "Autonomous UAV path planning and estimation," *IEEE Robot. Autom. Mag.*, vol. 16, no. 2, pp. 35–42, 2009.

[19] A. R. Washburn, "Branch and bound methods for a search problem," *Naval Research Logistics*, vol. 45, 1998.

[20] E.-M. Wong, F. Bourgault, and T. Furukawa, "Multi-vehicle Bayesian search for multiple lost targets," in *IEEE Int. Conf. on Robot. and Autom.*, April 2005, pp. 3169–3174.

[21] T. Furukawa, F. Bourgault, B. Lavis, and H. F. Durrant-Whyte, "Recursive Bayesian search-and-tracking using coordinated UAVs for lost targets," in *IEEE Int. Conf. on Robot. and Autom.*, May 2006, pp. 2521–2526.

[22] J. P. Hespanha and H. Kizilocak, "Efficient computation of dynamic probabilistic maps," in *Mediterranean Conf. on Control and Autom.*, Jul. 2002.

[23] L. F. Bertuccelli and J. P. How, "Robust UAV search for environments with imprecise probability maps," in *IEEE Conf. on Decision and Control and the European Control Conferenc*, Dec 2005, pp. 5680–5685.

[24] A. Khan, E. Yanmaz, and B. Rinner, "Information exchange and decision making in micro aerial vehicle networks for cooperative search," *IEEE Trans. Control Netw. Syst.*, vol. 2, no. 4, pp. 335–347, 2015.

[25] A. Shende, M. J. Bays, and D. J. Stilwell, "Toward a mission value for subsea search with bottom-type variability," in *Oceans*, Oct 2012, pp. 1–5.

[26] H. Yetkin, C. Lutz, and D. Stilwell, "Utility-based adaptive path planning for subsea search," in *OCEANS'15 MTS/IEEE Washington*. IEEE, Oct. 2015, pp. 1–6.

[27] F. Bourgault, T. Furukawa, and H. E. Durrant-Whyte, "Coordinated decentralized search for a lost target in a Bayesian world," in *Int. Conf. on Intell. Robots and Systems*, vol. 1. IEEE/RSJ, Oct. 2003, pp. 48–53.

[28] F. M. Delle Fave, Z. Xu, A. Rogers, and N. R. Jennings, "Decentralised coordination of unmanned aerial vehicles for target search using the max-sum algorithm," in *Proc. of the AAMAS Workshop on Agents in Real Time and Dynamic Environment*, May 2010, pp. 35–44.

[29] P. Sujit and D. Ghose, "Optimal uncertainty reduction search using the k-shortest path algorithm," in *American Control Conf.*, vol. 4. IEEE, Jun. 2003, pp. 3269–3274.

[30] Y. Yang, A. A. Minai, and M. M. Polycarpou, "Decentralized cooperative search by networked UAVs in an uncertain environment," in *American Control Conf.*, vol. 6. IEEE, 2004, pp. 5558–5563.

[31] P. B. Sujit and D. Ghose, "Negotiation schemes for multi-agent cooperative search," *Proc. of the Institution of Mech. Engineers, Part G: J. of Aero. Eng.*, vol. 223, no. 6, pp. 791–813, 2009.

[32] M. Mirzaei *et al.*, "Cooperative multi-vehicle search and coverage problem in uncertain environments," in *Decision and Control and European Control Conf.*, Dec 2011, pp. 4140–4145.

[33] H. Sato and J. O. Royset, "Path optimization for the resource-constrained searcher," *Naval Research Logistics*, vol. 57, no. 5, pp. 422–440, 2010.

[34] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA USA: MIT Press, 2005.

[35] S. Edelkamp and S. Schroedl, *Heuristic Search: Theory and Applications*. Waltham, MA USA: Elsevier, 2011, ch. 5.

[36] L. R. Harris, "The heuristic search under conditions of error," *Artificial Intell.*, vol. 5, no. 3, pp. 217–234, 1974.

[37] A. Zelinsky, R. Jarvis, J. C. Byrne, and S. Yuta, "Planning paths of complete coverage of an unstructured environment by a mobile robot," in *Int. Conf. on Adv. Robot.* IEEE, May 1993, pp. 533–538.