Optimal Search for Multiple Targets in a Built Environment

Haye Lau, Shoudong Huang and Gamini Dissanayake
ARC Centre of Excellence for Autonomous Systems (CAS)
University of Technology, Sydney
NSW, Australia
{hlau, sdhuang, gdissa}@eng.uts.edu.au

Abstract – **The main contribution of this paper is an algorithm for autonomous search that minimizes the expected time for detecting multiple targets present in a known built environment. The proposed technique makes use of the probability distribution of the target(s) in the environment, thereby making it feasible to incorporate any additional information, known a-priori or acquired while the search is taking place, into the search strategy. The environment is divided into a set of distinct regions and an adjacency matrix is used to describe the connections between them. The costs of searching any of the regions as well as the cost of travel between them can be arbitrarily specified. The search strategy is derived using a dynamic programming algorithm. The effectiveness of the algorithm is illustrated using an example based on the search of an office environment. An analysis of the computational complexity is also presented.**

Index Terms – Multiple targets, target search, dynamic programming, topological map, probability distribution

I. Introduction

In many rescue and security scenarios, it is necessary to efficiently direct mobile responders to find and reach phenomena of interest. In an Urban Search and Rescue (USAR) setting, the objective is to locate and render aid to the victim as quickly as possible in a race against diminishing survival windows. In security applications, the targets or events are to be found and investigated by capable mobile agents, minimizing the opportunity for harm. In both of the above contexts, the key is to have the searcher physically reach the target(s) as quickly as possible.

If the targets locations are completely unknown, the search problem becomes that of area coverage. In many cases, however, there may be partial information available about the target locations, for example, a distress signal, witness reports "He's in the back of the office", informed guesses based on past patterns, or information gathered through a network of sensors such as motion detectors distributed in the environment [1]. The key issue is then to find an efficient algorithm that can guide the selective search process by making use of all the available information [2].

One of the natural ways to capture the available information is to represent it as the likelihood of the target presence in the search space. In the case of finding targets in free space, such as boats lost at sea, Bourgault et al. [3] employed a Bayesian approach where the target probability density function (PDF) is used as prior information. As rescue air vehicles cover the ocean, the target PDF is updated using the model of the sensor and expected target motion. Optimal trajectories for the search are those that maximize the cumulative probability of detection over a limited time horizon. The key assumptions used in [3] are that the PDF of the target locations is smooth and the search space is free from obstacles constraining searcher motion. The global optimum can only be found if the time horizon is infinite, however, this is not computationally tractable. In practice, a short planning horizon is used to achieve reasonable but globally suboptimal trajectories.

In the case of a built environment consisting of, for example, floors in a building, or rooms on a number of floors, the likelihood of target presence can be described by discrete probabilities associated with specific regions. Early search problems such as [4] sought to minimize the expected cost to find a target among a series of discrete cells given a target distribution, detection probability and search costs. This formulation however assumes that the time required to reach and examine each cell is independent of previous action. Clearly, this assumption is violated in a typical indoor environment where travel time to any given region is a function of the current location of the agent. Furthermore, motion of the searcher is constrained and depends on the topology of the environment. The more realistic constrained search or optimal searcher path problem [5][6] embodies these movement restrictions, but most existing work assumes uniform regions that take identical search times and identical travel times from each region to its immediate neighbors.

This paper addresses the multiple target search problem in a known environment which can be described by a set of connected regions. The available information about the targets is expressed by the expected proportion of targets present in each region. A search strategy that minimizes the expected average time for detection of each target is presented. In contrast to much of the existing literature, the proposed algorithm deals with an environment where multiple targets may be present and allows travel and search costs of individual regions to be arbitrarily specified. In a related research, Lössner and Wegener [7] also examined a scenario with non-uniform search costs, but this work only focused on the conditions necessary for the existence of optimal solutions to a single target problem.

The optimal strategy proposed in this paper is based on dynamic programming. While dynamic programming is known to be computationally expensive, it will be shown that the special structure of the search problem can be exploited to obtain an efficient solution. For example, the search strategy for an environment containing 14 distinct regions with non-zero target probability can be computed at a cost of less than 20 seconds in a high end PC.

The paper is arranged as follows. The search problem is formally described in section II. Section III develops the dynamic programming algorithm for obtaining the optimal search schedule. Section IV describes the computational aspects of the algorithm and discusses the issue of computational complexity. The effectiveness of the proposed algorithm is presented using simulated search scenarios in an office environment in section V. Section VI provides the conclusions and discusses some possible extensions to the proposed approach.

II. Problem Description

Searching looks for objects of interests – targets. A general single-robot search problem for a known environment can be described as follows.

Given knowledge of:

- The *environment structure and searcher capability*
- A priori *information of the targets*

Find a search strategy such that the *search efficiency* is maximized.

The following defines the specific search problem addressed in this paper.

A. *Environment Structure and Searcher Capability*

Typical indoor environments are composed of connected regions (floors, rooms, halls or corridors). In this paper, we assume that the environment can be decomposed into a set of connected regions (Fig. 1). The distances between each pair of connected regions are assumed to be known, along with the size (and structure) of each region. It is assumed a map of the environment is available and the searcher can self-localize and move between regions as required. Since the size of each region and the distance between two connected regions are known, the time necessary to search a region and the minimum time needed to move from a region to each of its immediate neighbours can be easily computed.

Once a searcher is on an edge leading to another region, it is committed to keep moving until reaching the destination. On arrival in a region, it can take the following two actions:

1. Search the region it is in
2. Move to an adjoining region without searching

The search of a region is expected to detect all the targets present there, although an extension to the case where there is a finite probability of missed detection can be accommodated.

A topological map can be used to describe the environment. The complete search area is partitioned into a weighted undirected graph G = (N, E), where each node denotes a region *i*, that requires a search time T_i. n = |N| is the total number of regions. An edge exists if one can travel from one region directly to another. The weight A_{ij} denotes the time necessary to move from region *i* to *j* (Fig. 1). The set of all the adjoining nodes to region *i* is denoted by $E(i)$. Typical indoor environments are not fully connected. Therefore, $E(i)$ would only be a small subset of N. The location of a robot at time *t* is *x*(*t*). The motion of the searcher is described by:

1. When searching region *i*:

$$x(t) = i \rightarrow x(t+T_i) = i \tag{1}$$

2. When moving from region *i* to *j*:

$$x(t) = i \rightarrow x(t+A_{ij}) = j,\ j \in E(i) \tag{2}$$

B. *Information of the Targets*

The most convenient form of describing the information available about the targets is to use the expected proportion of targets in each region. If there is no prior knowledge, the expected proportion of targets in a given region may be set proportional to its size.

At time *t*, the expected proportion of the remaining targets in region *i* is given by:

$$p_i(t),\ i = 1 \ldots n \tag{3}$$

Note that:

$$\sum_{i=1}^{n} p_i(t) = 1 \tag{4}$$

except at the termination of the search when all targets have been found. In this case:

$$p_i(t) = 0,\ i = 1 \ldots n \tag{5}$$

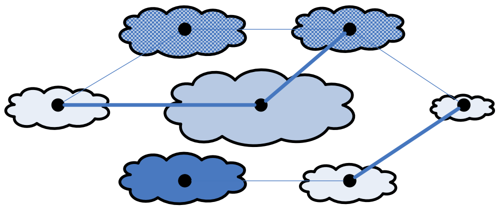

Fig. 1 Pictorial view of an environment decomposed into a set of regions, with different expected proportion of targets (shading), search time (cloud size) and travel costs (line thickness)

C. *Target Information Update*

Consider the actions described by (1) and (2). If the searcher at time *t* chooses to move from region $x(t) \in \{1 \ldots n\}$ to a neighboring region *j*, then the information about the targets stays unchanged until time $t + A_{x(t)j}$. If the searcher chooses to search the region *x(t),* then the expected target proportions are updated for the following cases:

1. With probability $1 - p_{x(t)}(t)$:

$$p_{x(t)}(t+T_{x(t)}) = 0; p_i(t+T_{x(t)}) = \frac{p_i(t)}{1-p_{x(t)}(t)}, i \neq x(t) \tag{6}$$

2. With probability $p_{x(t)}(t)$:

$$p_{x(t)}(t+T_{x(t)}) = 0\ ;\ p_i(t+T_{x(t)}) = 0\ ,\ i \neq x(t) \tag{7}$$

Equation (7) reflects the final scenario where all the targets have been found. When there are targets remaining to be found, the probability mass is distributed amongst the unsearched regions as described by (6).

D. *Search Efficiency*

When looking for a single target, two typical measures used for search efficiency are (i) time to detect, and (ii) probability of detection within a given time window. In the case where multiple targets are present, there is a variety of objectives to choose from. For example, minimizing the time to find all the targets or to find the very first target, or maximising the number of targets found in a fixed time period, are all reasonable objectives.

In search and rescue scenarios, on the one hand, we aim to find all the targets, on the other, we want to find most (if not all) of the targets as quickly as possible. This motivates us to choose "minimization of the average time to find a target" as the objective measuring the search efficiency, provided all targets are found. For example, when there are 10 targets, one search plan finds the first 2 targets in 5 minutes and the remaining 8 targets in 30 minutes. The average time to find a target is (2*5+8*30)/10=25 minutes. This is not as desirable as finding 8 targets at 7 minutes and the last 2 targets at 35 minutes, where the average time is (8*7+2*35)/10=12.6 minutes.

In practice, since the exact locations of the targets are not known, we can only minimize "the expected average time to find a target" (instead of the true average time) provided all the targets are found. In the special case when there is only a single target in the environment, this objective is equivalent to minimizing the expected time of detection.

E. *Discrete Time Description of the Problem*

A searcher needs to choose an action at discrete instants of time. Let $u(k)\in\{s,E(x(k))\}$ be the action after *k* time steps while $x(k)$ and $p_i(k)$, $i=1...n$, denote the searcher location and expected target proportions, respectively, after *k* control actions. The action stored in $u(k)$ denotes a transition to a neighboring region or if equaling '*s*', an action to investigate the region where the robot already is. Note that the actual time interval between two time steps may be different in general.

For an initial robot location $x(0)$ and any given control sequence $u(0),u(1),u(2),...,$ we can determine $T(i), i=1...n,$ the time needed to finish the search of region *i*, starting from the initial robot location $x(0)$. The expected average target detection time can then be computed by:

$$T(u(0),u(1),...)=T(1)\cdot p_1(0)+T(2)\cdot p_2(0)+...T(n)\cdot p_n(0) \quad (8)$$

The optimum search problem can now be written as: Given a map of the environment (such as Fig. 1), the initial robot location $x(0)$ and an initial expected proportion of targets $p_1(0),...,p_n(0)$, decide a sequence of $u(k)$ actions to minimize the expected average time to find a target given by (8).

Because there can be an infinite number of control sequences, it is in general not possible to compute a minimal value of (8) (together with the optimal sequence) directly.

III. Approach

In this section, a method for obtaining the optimum sequence of actions using dynamic programming is presented. Although dynamic programming tends to be computationally expensive in general, it will be shown that the structure of the specific problem lends itself to an efficient implementation of this algorithm.

A. *Value Function*

It can be seen that the expected target proportion $p_1,...,p_n$ contains sufficient information to derive the optimal search strategy for the robot. In fact, following a similar argument as in section 2 of [7], it can be shown that at each time step *k*, the optimal control action $u^*(k)$ only depends on the robot location $x(k)$ and expected target proportion $p_1(k),...,p_n(k)$. So we define the value function as follows.

For any robot location $x\in\{1,2,...,n\}$ and any feasible expected target proportion $p_1,...,p_n$ the value function $V(x,p_1,...,p_n)$ is the minimum expected average time, starting from the current time, to find a target given the specified robot location and expected target proportions.

B. *Dynamic Programming Equation*

By the principle of optimality, the following Dynamic Programming Equation (DPE) is obtained: for any $(p_1,...,p_n)\neq(0,\dots,0)$,

$$V(x,p_1,...,p_n)=\min\begin{bmatrix} T_x+(1-P_x)\cdot V(x,\dfrac{p_1}{1-p_x},...,\dfrac{p_{x-1}}{1-p_x},0,\dfrac{p_{x+1}}{1-p_x},...,\dfrac{p_n}{1-p_x}), \\ \min\limits_{j\in E(x)}\{A_{xj}+V(j,p_1,...,p_n)\} \end{bmatrix} \quad (9)$$

In other words, the minimal expected average time is equal to the lesser of (i) the minimal expected time if the robot chooses to search the area it is in, equaling the search time T_x plus the minimal expected time calculated from when the search is finished, and (ii) the minimal time for the robot to move to any region $j\in E(x)$ plus the minimal expected average time calculated from when the robot is in that new region j.

The boundary condition that represents the case when the search is concluded and thus no more time is needed is:

$$V(x,0,...,0)=0, x\in\{1,2,...,n\} \quad (10)$$

IV. Computation issue

A. *Computation of the Optimal Control Actions*

The following is an algorithm that can be used to compute the value function and the optimal control action.

Algorithm:

Step 1. Initialize the value function V_0, for example choose $V_0(x,p_1,\cdots,p_n)=0$ for all $x,p_1,\cdots,p_n$. Arbitrary values can be selected provided the boundary condition (10) is met.

Step 2. Compute function V_{i+1} using the dynamic programming (DP) recursion for $(p_1,\cdots,p_n)\neq(0,....,0)$

$$\begin{aligned} &V_{i+1}(x,p_1,...,p_n) \\ &=\min\begin{bmatrix} T_x+(1-P_x)\cdot V_i(x,\dfrac{p_1}{1-p_x},...,\dfrac{p_{x-1}}{1-p_x},0,\dfrac{p_{x+1}}{1-p_x},...,\dfrac{p_n}{1-p_x}), \\ \min\limits_{j\in E(x)}\{A_{xj}+V_i(j,p_1,...,p_n)\} \end{bmatrix} \end{aligned} \quad (11)$$

Step 3. Stop when $V_{i+1}(x, p_1, ..., p_n) = V_i(x, p_1, ..., p_n)$, for all $x, p_1, \cdots, p_n$.

Then the final function $V_i(x, p_1, \cdots, p_n)$ satisfies the DPE and is the value function that we seek. The optimal control law $u^*(x, p_1, \cdots, p_n)$ can be obtained simultaneously in the last step of iteration. In fact, if in the final iteration searching gives the minimal time in (11), then the action 's' should be taken. Otherwise the searcher should move to the region j that achieves the lowest value. It is possible for more than one control action to yield the same minimum value.

B. *Computation Complexity*

Because there is an infinite number of feasible expected target proportions $p_1, ..., p_n$ (for example p_1 can be any real number between 0 and 1), the DP algorithm presented above cannot be applied directly. A naïve way is to discretize the continuous state space first and only compute the value function at the discretized states. However, this may cause some numerical problems due to the discretization and a fine discretization will lead to unacceptable computation cost.

Actually, by a close look at the specific search problem considered in this paper, it can be seen that once the initial expected target proportion $p_1(0), ..., p_n(0)$ is given, the total number of all the possible target proportion generated from some feasible control actions is finite -- 2^{n_0}, where n_0 is the number of regions with non-zero probability. Accordingly, only the value function and the optimal control actions for these target proportions (together with different robot locations) need to be computed. As there are n regions for the robot to be in, the total number of states $(x, p_1, \cdots, p_n)$ in the DP algorithm is $n2^{n_0}$. This makes it possible to compute the exact value function and optimal control law using the proposed DP algorithm at dramatically reduced computation cost.

A research implementation of the algorithm in MATLAB executes in under 20 seconds on a Pentium M 1.4-Ghz computer, for the environment shown in Fig. 2 where $n_0 = 14$ and $n = 17$. When all seventeen regions have non-zero target probability ($n_0 = 17$), paths are generated within four minutes. Despite the NP-hard nature of the problem [8], the proposed algorithm is viable in this scenario.

It should be noted that in doing so, the optimal strategy $u^*(x, p_1, \cdots, p_n)$ is only available for the states that can be generated from the initial expected target proportion $p_1(0), ..., p_n(0)$. If additional information becomes available (e.g. through an embedded sensor network) contradicting the original assumptions for the target proportions, then the value function and optimal control action need to be recomputed based on the updated targets information. Since the computation time for the planning is reasonably short, this is acceptable in practice.

V. Simulations

A. *Scenario and Simulations*

This section illustrates the proposed algorithm using a simulated search of an office floor in the University of Technology, Sydney. The floor is divided into 17 regions as shown in Fig. 2a. The cost of travel (shown in parenthesis) between regions was computed by planning the shortest distance path between the nodes for a robot with a speed of 0.5 m/s. The estimated costs are obtained by computing minimum distance paths between the regions, giving due consideration to the presence of walls, doors etc. In Figure 2, however, regions are linked by straight lines for clarity. The search cost per region is set proportional to its area. Further details of the simulation and a number of animations are available in http://www.eng.uts.edu.au/~hlau/indoor_search.

B. *Example Track*

Fig. 2(a-b) shows a part of the optimal search plan produced when the searcher starts from region 9. The prior expected target proportion is represented using the degree of shading in Fig. 2a (for example room 17 has the highest proportion of targets). The sequence of rooms searched following the plan is start-17-16-15-14-13-12-11-2-3-1-10.

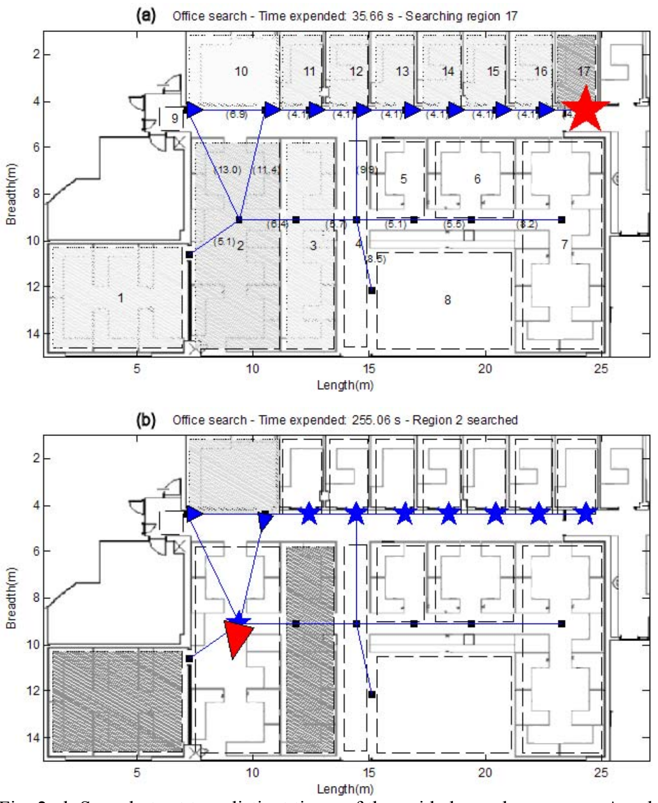
(a) Office search - Time expended: 35.66 s - Searching region 17
(b) Office search - Time expended: 255.06 s - Region 2 searched
Breadth(m)
Length(m)
10
11
12
13
14
15
16
17
9
5
6
1
2
3
4
7
8
(6.9)
(4.1)
(13.0)
(11.4)
(9.8)
(6.4)
(5.7)
(5.1)
(5.5)
(6.2)
(8.5)

Fig. 2a-b Snapshots at two distinct times of the guided search sequence. A red triangle depicts the current searcher position, while a red star indicates that a region is currently being swept. Blue stars mark the regions already searched and blue triangles show the regions the robot bypassed without searching. 2(a) shows the status of the environment after 9 time steps while 2(b) shows the simulation after 24 time steps

The strategy is clearly not greedy with respect to either distance or probability. For example, the robot bypasses regions 10-16 and proceeds to search region 17, despite starting at region 9. On the other hand, after completing the search of region 17, it spends time searching the regions 16-

11, although region 2 now has the highest proportion of targets. The proposed algorithm generates schedules which strike an optimal balance between anticipated costs and potential rewards.

C. *Comparison with Shortest Paths*

The following example demonstrates how the algorithm leverages available information about the likely distribution of targets to improve search effectiveness. If we know nothing of where the targets could be and each region requires insignificant effort to investigate (e.g., the searcher only needs to go there and see), then the optimal plan is equivalent to a shortest coverage path through the set of nodes. For example, given equal target probability in regions 1, 2, 3, 6, 7, 10, 11, 12, the proposed algorithm generates a shortest possible route which requires at most 78.34s to search if it succeeds on the very last region. However, if we are privy to information suggesting region 7 is more likely to contain targets (Fig. 3), the sequence generated has the worst case time of 80s. While appearing counterintuitive, the new route (start-10-11-12-6-7-3-2-1) is designed to result in a lower expected time than that achievable if the geometrically shortest path was followed.

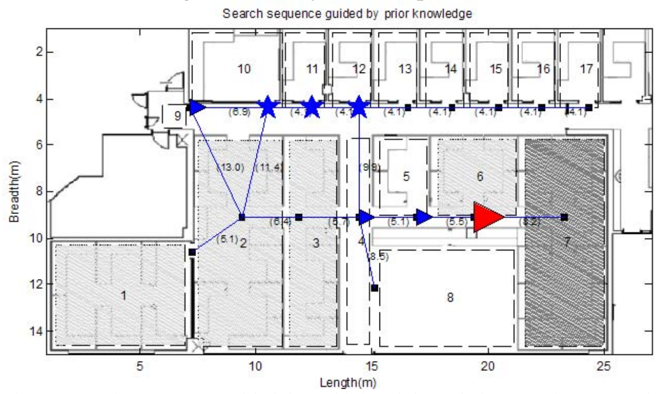

Fig. 3 Search sequence guided by additional knowledge. A shortest path would search leftwards first. Investigating the right side earlier however results in the minimal expected time.

This was in fact confirmed by using randomly generated target locations in 10,000 trials. As expected, the strategy of using the shortest path also results in the shortest expected time for detection when targets are indeed evenly distributed. However, when the expected target proportion is not uniform, it can be seen that the proposed strategy results in smaller expected time to detect than that achieved by the shortest geometric route, requiring an average time of 42.2s versus 47.7s, a difference of 12.9 percent.

This example illustrates that when there is additional data about likely target presence, the algorithm is able to make the most of the information. On the other hand, in the absence of such guidance, the plan generated is nevertheless the best that can be expected given the complete lack of knowledge.

D. *Comparison with Heuristics*

Optimal action sequences to detect a single target are compared against three common heuristics to show the relative performance of this approach and the respective influences of search and travel costs. The heuristics are characterized by the criteria used to select the next possibly occupied region to search at each time step, defined as follows:

1) Maximize the highest probability of detection
2) Minimize the cost of travel and search
3) Maximize the ratio of detection probability and cost

In the latter two, the cost is the time required to travel directly to the target region and then immediately search it. Despite being locally optimal, the techniques generate reasonable trajectories for comparison, particularly on small graphs. The third is known to be also optimal for a restricted version of this problem [4] and is the one-step version of the utility greedy heuristic used in [9].

The results after 10,000 runs of the four methods with randomly generated target locations in the same environment are collated in Table II. To examine the relative effect of varying search and travel costs, two additional scenarios where the search cost is zero and large compared to the travel cost (as outlined in Table I) are considered to examine the relative effect of the costs.

TABLE I
SCENARIO PROPERTIES

Scenario	**Target Distribution**	**Search and Travel Costs**
1	11 regions with non-zero target probability	Search costs and travel costs are similar
2	Same as 1	Search cost is significantly larger than the travel cost
3	Same as 1	Search cost is zero

TABLE II
SCENARIO RESULTS

Scenarios	Mean Time to Detection (s)			
	DP algorithm	Maximize detection probability	Minimize travel and search costs	Maximize detection probability / cost
1	169	237.6(40.6%)	185.1(9.5%)	184.7(9.3%)
2	559.8	762.4(36.2%)	644.9(15.2%)	569.9(1.8%)
3	54.7	105.3(92.5%)	57(4.2%)	74.6(36.4%)

The three heuristics are compared with the optimal strategy generated by DP. As can be seen, maximizing the ratio of detection probability versus cost performed well for the heavily contrived scenario 2, since a searcher's current position has almost no bearing on the time required to cover any other cell when search costs dominate the travel time. This then satisfies the condition identified in [4] for a locally optimal solution to be also globally optimal in minimizing expected cost. In the absence of this greedy choice property (i.e. in larger indoor areas), the advantage of the proposed algorithm becomes more pronounced as the cost to reach the next destination increasingly depends on the current region. When there are no search costs (scenario 3), only the travel time remains to be minimized, rendering a shortest path heuristic competitive in the absence of target information.

VI. Discussion and Conclusions

The core challenge in this search problem, be it for rescue or surveillance applications, is to determine where to go, given what we know. The key benefit of the approach presented lies in its ability to take advantage of additional available target information to create a more effective plan of action (Fig. 4). The high-level strategy can then sensibly coordinate the searching of individual areas. Instead of just covering all the regions as quickly as possible, when formulating an optimal plan, the search and travel costs of the regions are traded off against how likely they are to contain targets in the first place. The use of expected target proportions across the area as a basis renders this approach flexible in the sources of information that can be incorporated. For instance, low costs sensors distributed throughout buildings could provide a coarse estimation based on sound, motion or heat. Similarly the last known contact position could form an estimate to accelerate the search.

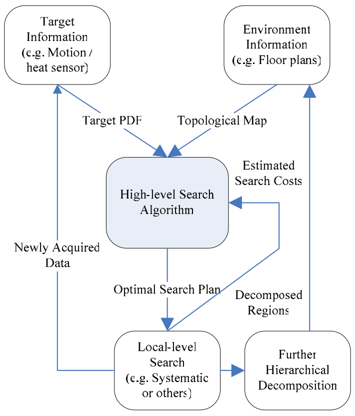
Target Information (e.g. Motion / heat sensor)
Environment Information (e.g. Floor plans)
Target PDF
Topological Map
Estimated Search Costs
High-level Search Algorithm
Newly Acquired Data
Optional Search Plan
Decomposed Regions
Local-level Search (e.g. Systematic or others)
Further Hierarchical Decomposition

Fig. 4 Approach in context

Although this paper concentrates on a static environment with targets remaining stationary, the computational cost of the algorithm is such that re-planning to incorporate the effect of any new information gathered (reflected in a new probability distribution for the target location) is feasible.

The computational cost of the approach is related to the number of regions with non-zero target probability. While the technique is tractable for an environment with around 14 regions (in under 20s), a sub-optimal hierarchical decomposition is necessary for tackling larger environments or when faced with stricter timing constraints. Decomposing a large area into constituent parts has been used as a natural way of managing complexity in search related problems. [10] divides a structure into regions to coordinate the tracking of targets between cells, [11] extracts the topology in architectural plans for navigation, while [12] solves a Pursuit-Evasion problem by first partitioning visibility regions. Of most direct interest, [2] describes the USAR scenario where a disaster site is segmented into regions with different survivor probability, thereby facilitating more rational decision-making over where to focus resources. Hierarchical decomposition could be achieved for this technique along structural lines (e.g. from urban block to buildings to floors), or by grouping the target probabilities of regions remote from the searcher. The latter in particular allows for planning in graduated granularity.

When the flexibility of supporting a more general detection function is not necessary, only an order of searching the non-empty regions is required. The search problem in this paper may then be posed as that of a Maximum Collection Problem with Time-Dependent Rewards [13]. We are currently investigating optimal techniques for solving the problem on small map sizes.

Work is underway on search in dynamic environments and multi-robot search, to extend the ideas presented in this paper.

Acknowledgment

This work is supported by the ARC Centre of Excellence Programme, funded by the Australian Research Council (ARC) and the New South Wales State Government.

References

[1] B. Krishnamachari and S. Iyengar, “Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks,” *IEEE Transactions on Computers*, vol. 53, no. 3, pp. 241-250, 2004.
[2] R. R. Murphy, “Biomimetic search for urban search and rescue,” *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 2073-2078, 2000.
[3] F. Bourgault, T. Furukawa and H. F. Durrant-Whyte, “Coordinated Decentralized Search for a Lost Target in a Bayesian World,” *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, October, 2003, Las Vegas, Nevada.
[4] L. D. Stone, *Theory of Optimal Search*: Academic Press, 1975.
[5] J. N. Eagle and J. R. Yee., “An Optimal Branch-and-Bound Procedure for the Constrained Path, Moving Target Search Problem,” *Operations Research*, vol. 38, no. 1, pp. 110-114, January-February 1990.
[6] B. DasGupta, J. Hespanha and E. Sontag, “Aggregation-based Approaches to Honey-pot Searching with Local Sensory Information,” *Proceedings of the American Control Conf.*, pp. 1202-1207, Boston, June 2004.
[7] U. Lössner and I. Wegener, “Discrete Sequential Search with Positive Switch Cost,” *Mathematics of Operations Research*, vol. 7, no. 3, August 1982
[8] K. E. Trummel and J. R. Weisinger, “The Complexity of the Optimal Searcher Path Problem,” *Operations Research*, vol. 34, no. 2, pp. 324-327, March-April 1986.
[9] A. Sarmiento, R. Murrieta and S. Hutchinson, “An Efficient Strategy for Rapidly Finding an Object in a Polygonal World,” *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, pp. 1153-1158, 2003
[10]B. Jung and G. S. Sukhatme, “A Generalized Region-based Approach for Multi-target Tracking in Outdoor Environments,” *IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2189-2195, 2004.
[11]A. Murarka and B. Kuipers, “Using CAD drawings for robot navigation,” *2001 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 3494, pp. 678-683, 2001.
[12]B. P. Gerkey, S. Thrun and G. Gordon, “Visibility-based pursuit-evasion with limited field of view,” *Proceedings of the National Conference on Artificial Intelligence*, pp. 20-27, 2004.
[13]E. Erkut and J. Zhang, “The Maximum Collection Problem with Time-Dependent Rewards,” *Naval Research Logistics*, vol. 43, pp. 749-763, 1996.