Honey-pot Constrained Searching with Local Sensory Information

Bhaskar DasGupta* João P. Hespanha† James Riehl† Eduardo Sontag‡

October 20, 2005

Abstract

In this paper we investigate the problem of searching for a hidden target in a bounded region of the plane by an autonomous robot which is only able to use local sensory information. The problem is naturally formulated in the continuous domain but the solution proposed is based on an aggregation/refinement approach in which the continuous search space is partitioned into a finite collection of regions on which we define a discrete search problem. A solution to the original problem is obtained through a refinement procedure that lifts the discrete path into a continuous one.

We show that the discrete optimization is computationally difficult (NP-hard) but there are computationally efficient approximation algorithms to solve it. The resulting solution to the continuous problem is in general not optimal but one can construct bounds to gauge the cost penalty incurred due to (i) the discretization of the problem and (ii) the attempt to approximately solve the NP-hard problem in polynomial time.

Numerical simulations show that the algorithms proposed behave significantly better than naive approaches such as a random walk or a greedy algorithms.

1 Introduction

The problem addressed concerns searching for a hidden target by an autonomous agent. Suppose that a "honey-pot" is hidden in a bounded region $\mathcal{R}$ (typically a subset of the plane $\mathbb{R}^2$ or of the 3-dimensional space $\mathbb{R}^3$). The exact position $\mathbf{x}^*$ of the honey-pot is not known but we do know its *a-priori* probability density f. The goal is to find the honey-pot using an agent (called the *searcher*) that moves in $\mathcal{R}$ and is able to see only a "small region" around it. If the searcher gets "sufficiently close," it will detect the honey-pot and the search is over. We assume that the target is stationary and therefore the probability density f does not change with time. Honey-pot searching is thus an (open-loop) path planning problem where one seeks a path that *maximizes the probability* of finding the honey-pot, given some constraint on the time or fuel spent by the searcher.

*Dept. of Computer Science, University of Illinois at Chicago, Chicago, IL 60607-7053. Email: dasgupta@cs.uic.edu. Supported by NSF grants CCR-0296041, CCR-0206795, CCR-0208749 and a CAREER grant IIS-0346973.

†Dept. of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106-9560. Emails: hespanha@ece.UCSB.edu, jriehl@ece.ucsb.edu@ece.UCSB.edu. Supported by NSF grants ECS-0242798, CCR-0311084.

‡Dept. of Mathematics, Rutgers University, New Brunswick, NJ 08903. Email: sontag@hilbert.rutgers.edu. Supported in part by NSF grant CCR-0206789.

To formalize this problem, let $\mathcal{S}[x] \subset \mathcal{R}$ denote the set of points in $\mathcal{R}$ that the searcher can see from some position $x \in \mathcal{R}$. *Cookie cutter detection* corresponds to the special case where $\mathcal{S}[x]$ consists of a circle with fixed radius around x [7], but here we consider general detection regions.

Problem 1 (Continuous Constrained Honey-pot Search, cCHS)**.** Find a continuous path $\rho : [0, T] \to \mathcal{R}$, $T > 0$ starting anywhere in a given set $\mathcal{R}_{\text{init}} \in \mathcal{R}$ with $\|\dot{\rho}(t)\| \leq 1$ almost everywhere that maximizes the probability of finding the honey-pot given by

$$R[\rho] := \int_{\mathcal{S}_{\text{path}}[\rho]} f(x)dx, \tag{1}$$

subject to a constraint of the form

$$C[\rho] := \int_0^T c\big(\rho(t)\big)dt \leq L, \tag{2}$$

where $\mathcal{S}_{\text{path}}[\rho] := \bigcup_{t\in[0,T]} \mathcal{S}[\rho(t)]$ denotes the set of all points that the searcher can scan along the path ρ and L denotes a positive constant. □

One should emphasize that in the definition of the reward (1), we have an integral over the area $\mathcal{S}_{\text{path}}[\rho]$ and *not a line integral along the path* ρ (*cf.* Figure 1). The distinction may seem subtle but it is quite fundamental because if a searcher transverses the same location multiple times, a line integral would increase with each passage but the region $\mathcal{S}_{\text{path}}[\rho]$ that the searcher scans does not. This formulation does not prevent the path from returning to a point previously visited (which could be necessary) but does not reward the searcher for scanning the same location twice.

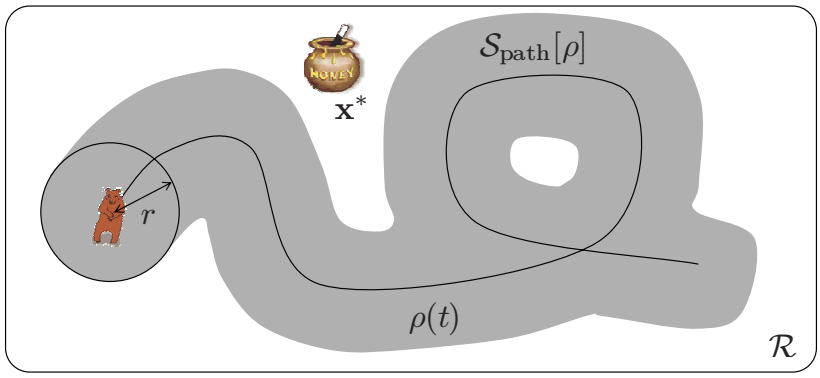


Figure 1. Region $\mathcal{S}_{\text{path}}[\rho]$ over which the integral in (1) should be computed for cookie cutter detection of radius r (*i.e.*, for $\mathcal{S}[x]$ defined to be a circle of radius r around x).

For bounded-time searches, $c(x) = 1, \forall x \in \mathcal{R}$ and L is the maximum time allowed for the search. For bounded-fuel searches, $c(\cdot)$ is the fuel-consumption rate and L is the total fuel available. The consumption rate can be position dependent when the terrain is not homogeneous. One can also "encode" obstacles in $c(\cdot)$ by making this function take large values in regions to be avoided.

The honey-pot search problem is inspired by the optimal search theory initiated by the pioneering work of Koopman [19] and further developed by Stone [27] and others. A summary of this work can be found in the surveys [6, 28]. The development of search theory was motivated by U.S. Navy operations during the Second World War, which included the search for targets in transit, setting up sonar screens, and protection against submarine attacks [20]. In the context of Naval operations, search theory has been used more recently in search and rescue operations by the U.S. Coast Guard [6], as well as to detect lost objects such as the H-bomb lost in the Mediterranean coast of

Spain in 1966, the wreck of the submarine USS Scorpion in 1968, or the unexploded ordnance in the Suez Canal following the 1973 Yom Kippur war. However, its application spans many other areas such as the clearing of land mines, locating parts in a warehouse, etc. The collection of papers [14] discusses several applications of search theory ranging from medicine to mining.

Until the 70s, most of the work in search theory decoupled the problems of finding the area that should be searched from that of finding a specific path for the searcher "covering" that area. This is sensible when (i) the cost-bound in (2) essentially poses a constraint on the total area that can be scanned and (ii) the optimal area turns out to be sufficiently regular so that one can find a continuous path ρ that sweeps it without overlaps. However, these assumptions generally only hold for time-constrained searches and Gaussian (or at least unimodal) *a-priori* target distributions. Complex distributions for f are likely to arise in many practical problems as discussed in [6]. The reader is referred, e.g, to [10, 29, 30] for numerical methods to efficiently compute target distributions based on noisy measurements.

More recently several researchers considered the so called *constrained search problem* where it is explicitly taken into account the fact that it must be possible to "cover" the area to be scanned using one or more searchers moving along continuous paths. Mangel [22] considered continuous search problems where the goal is to determine an optimal path that either maximizes the probability of finding the target in a finite time interval or minimizes the infinite-horizon expected time needed to find the target. In Mangel's formulation, this is reduced to an optimal control problem on the searcher's velocity $\dot{\rho}$, subject to a constraint in the form of a partial differential equation. In practice, this problem can only be solved for very simple *a-priori* target distributions f.

An alternative approach that proved more fruitful consists of discretizing time and partitioning the continuous space into a finite collection of cells. The search problem is then reduced to deciding which cell to visit on each time interval. Constraints on the searcher's motion can be expressed by only allowing it to move from one cell to adjacent ones [27]. At least when the time horizon is finite (and in some cases even when the time horizon is infinite [21]), the resulting optimal discrete search problem can be solved by finite enumeration of all possible solutions. However, this method scales poorly (exponentially!) with the number of cells. Eagle [4] noted that a discrete search can be formulated as an optimization on a partially observable Markov decision process (POMDP) and proposed a dynamic programing solutions to it. However, since the optimization of POMDPs is computationally very difficult, this approach is often not practical. Instead, Eagle and Yee [5] and Stewart [25, 26] formulated the discrete search as a nonlinear integer programming problem and proposed branch-and-bound procedures to solve it, which are optimal for the case considered in [5]. Hespanha *et al.* [11] proposed a (non-optimal) but computationally efficient greedy strategy that leads to capture with probability one, but no claims of optimality are made. The references [4, 5, 11, 25, 26] above considered the general case of a moving target but, as noted by Trummel and Weisinger [31], even the case of a stationary target is NP-Hard.

The starting point for this paper is the observation that most of the previous work based on spatial and temporal discretization ignored the continuous nature of the underlying problem. In fact, numerically efficient solutions to the cCHS problems exhibit two distinct sources of suboptimality: (i) the discretization procedure that converts the original continuous problem into a discrete one, and (ii) the attempt to approximately solve an NP-Hard problem in polynomial time. In this paper we address both issues:

1. Section 2 formalizes spatial and temporal discretization as an aggregation/refinement procedure and shows how it can formulated so that the final continuous path satisfies the constraint

(2), while providing some guaranteed reward (1). It also describes how to compute upper and lower bounds on the achievable (continuous) reward (1) by solving discrete problems.

2. Section 3 addresses the solution to the discrete optimization problems that arise out of the aggregation/refinement procedures described in Section 2. We show that these problems are NP-Hard but can be approximately solved in polynomial time. The algorithms proposed result in rewards no smaller than $1/(5+\epsilon)$ of its optimal value, where ϵ can be made arbitrarily small at the expense of increasing computation. When the problem exhibits additional structure on the costs and/or rewards, better worst-case bounds on the reward can be achieved.

The combined results of Sections 2 and 3 provide a computationally efficient approximate solution to the original cCHS problem. A subset of the results in this paper was presented at the 2004 American Control Conference.

2 Aggregation/refinement procedure

One can regard spatial and temporal discretization as part of an aggregation/refinement approach to solve the cCHS problem. In this type of approach one starts by aggregating the continuous search space $\mathcal{R}$ into a finite collection of regions on which a discrete search problem is defined (*cf.* Figure 2). From the solution to this problem, one then recovers a solution of the original problem through a refinement procedure that lifts the discrete path into a continuous one. This is inspired by

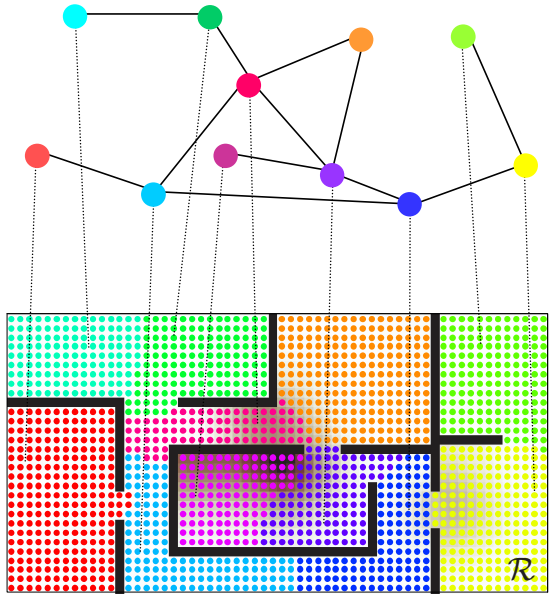


Figure 2. Aggregation of a continuous search space $\mathcal{R}$ into a collection of eleven regions. The black lines in the search space represent obstacles (walls) that cannot be crossed by the searcher. The specific partition of $\mathcal{R}$ shown in the figure is described in Section 4.

discrete abstractions of hybrid systems, where the behavior of a system with a state-space that has both discrete and continuous components is abstracted to a purely discrete system to reduce the complexity (*cf.* survey [1]). In our problem, the original system has no discrete components but we still reduced it to a discrete system by an abstraction procedure. A key difference between the results here and those summarized in [1] is that in general our abstraction procedure introduces some degradation in performance because the discretized system does not capture all the details of the original system. In particular, some information about the distribution of the honey-pot may

be lost in the abstraction. However, by allowing some performance degradation we can significantly enlarge the class of problems for which the procedure is applicable.

Given a partition[1] V of the continuous search space $\mathcal{R}$, an aggregation/refinement approach to solve the cCHS comprises the following three steps:

1. Construction of a graph $G = (V, E)$ whose vertices are the regions in V and with an edge $(v_1, v_2) \in E$ between any two regions $v_1, v_2 \in V$ for which the searcher can reach v_2 from v_1. To each vertex one associates a reward that provides a measure of the probability of finding the target in that region and to each edge one associates a cost that measures the cost incurred by moving the searcher between the corresponding regions.

2. Computation of a path on the graph G that maximizes an appropriately defined reward, subject to a cost constraint. This path consists of a sequence of vertices in G, *i.e.*, a sequence of regions in V.

3. Refinement of the discrete path into a continuous one that satisfies the continuous constraint (2) and that yields a continuous reward (1) at least as large as the reward obtained for the discrete problem.

This section addresses steps 1 and 3 above, whereas step 2 is relegated to Section 3. This paper does not explicitly address the problem of choosing the partition V, which we take as given. However, a careful selection of V will clearly help minimize the cost penalty introduced by this approach.

2.1 Aggregated Reward Budget Problem

To guarantee that the final continuous path satisfies the continuous constraint (2), and with continuous reward (1) at least as large as the reward obtained for the discrete problem, the path refinement algorithm must satisfy a worst-case bound of the following form:

Property 1 (Path refinement)**.** Given a discrete path $p = (v_1, v_2, \ldots, v_k)$, $v_1 \cap \mathcal{R}_{\text{init}} \neq \emptyset$ in G (possibly with the same region appearing multiple times), the refined continuous path $\rho : [0, T] \to \mathcal{R}$ is a continuous function, with $\rho(0) \in \mathcal{R}_{\text{init}}$ and $\|\dot{\rho}(t)\| \leq 1$ almost everywhere. Moreover, there exist functions $c_{\text{worst}} : V \times V \to [0, \infty)$, $r_{\text{worst}} : V \to [0, \infty)$, $|\cdot|_{\text{worst}} : V \to \mathbb{N}_{\geq 0}$ such that

$$C[\rho] \leq \sum_{i=1}^{k-1} c_{\text{worst}}(v_i, v_{i+1}), \qquad R[\rho] \geq \sum_{v \in V} r_{\text{worst}}(v)\ \min\{|v|_{\text{worst}}, \#(p, v)\}, \tag{3}$$

where $\#(p, v)$ denotes the number of times that the vertex v appears in the path p.

The existence of the functions $c_{\text{worst}}(\cdot)$, $r_{\text{worst}}(\cdot)$, and $|\cdot|_{\text{worst}}$ that satisfy (3) can be assumed without loss of generality. In fact, several options are possible for the selection of $r_{\text{worst}}(\cdot)$ and $|\cdot|_{\text{worst}}$. For example, one can define

$$|v|_{\text{worst}} = 1, \qquad r_{\text{worst}}(v) = \max_{x \in v} \int_{\mathcal{S}[x] \cap v} f(x) dx, \tag{4}$$

provided that the first time that v appears in the discrete path, the searcher goes to the point x^* where the max in (4) occurs. In this case, $\mathcal{S}[x^*] \cap v$ is guaranteed to be a subset of the area $\mathcal{S}[\rho]$

[1] We recall that a *partition* V of a set $\mathcal{R}$ is a collection of subsets of $\mathcal{R}$ such that $\bigcup_{v \in V} v = V$ and $v \cap v' = \emptyset$, $\forall v \neq v' \in V$

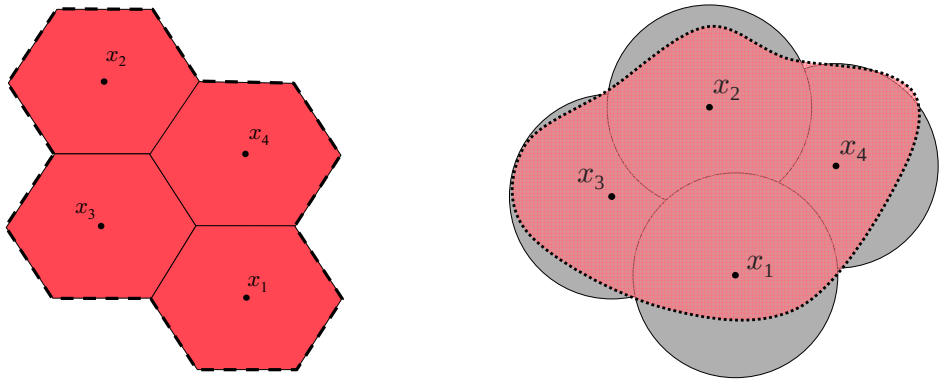
covered by the path and a reward of $\int_{\mathcal{S}[x^*]\cap v} f(x)dx$ is guaranteed. By setting $|v|_{\text{worst}} = 1$, the right-inequality in (3) does not require any additional reward for multiple visits to v. In practice, this may be very conservative, especially if $\int_v f(x)dx$ is much larger than $\int_{\mathcal{S}[x^*]\cap v} f(x)dx$. In this case, if there are k points $x_1, x_2, \ldots, x_k \in v$ for which

$$\int_{v\cap\mathcal{S}[x_i]\setminus\cup_{j=1}^{i-1}\mathcal{S}[x_j]} f(x)dx \geq r, \quad \forall i, \tag{5}$$

then one can set

$$|v|_{\text{worst}} = k, \qquad\qquad r_{\text{worst}}(v) = r, \tag{6}$$

provided that on the ith time that v appears in the discrete path p, the searcher goes to the point x_i. For the particular case in which the pdf f is approximately constant on v, finding such k becomes a form of covering problem in computational geometry (*cf.* Figure 3). In the example in



(a) (b)

Figure 3. Illustration of the worst-case rewards for a region v (dashed area) on which the pdf f is constant and equal to c. In the left plot we assumed that the detection area is hexagonal and the whole region can be exactly covered with 4 hexagons. In this case, we can set $|v|_{\text{worst}} = 4$ and $r_{\text{worst}}(v)$ equal to c times the area of the hexagon in (6). In the right plot, the region does not have a regular shape and the circles represent circular detection areas, which must necessarily overlap. In this case $|v|_{\text{worst}} = 4$ and $r_{\text{worst}}(v)$ is equal to c times the smallest area of one of the circles, excluding the area covered by the previous circles and the area outside the region v [cf. equation (5)], which probably corresponds to x_4 in this figure.

Section 4, the whole region was covered with small square tiles, the points x_i were chosen to be the centers of the tiles, and it was assumed that the area $\mathcal{S}[x_i]$ that can be seen from x_i is the square tile itself. Figure 3(a) shows a similar situation but with hexagonal tiles.

Several options are also possible for the selection of $c_{\text{worst}}(\cdot)$, *e.g.*,

$$c_{\text{worst}}(v, v') = \max_{x\in v, y\in v'} J[x, y], \tag{7}$$

where $J[x, y]$ denotes the minimum cost incurred when going from $x \in \mathcal{R}$ to $y \in \mathcal{R}$ (or some upper bound on it), which can be computed using numerous techniques available to solve shortest-path problems [8, 17, 18, 24]. Less conservative definitions of $c_{\text{worst}}(\cdot)$ are possible by taking the max in (7) only over points $x \in v$, $y \in v'$ where the max in (4) occurs or over the points x_i in (5). Note that in general $c_{\text{worst}}(v, v) > 0$ but one may generally keep this low by carefully selecting the order on which the the points x_i in (5) are visited.

The left-inequality in (3) guarantees that if the discrete path is selected so that

$$\sum_{i=1}^{k-1} c_{\text{worst}}(v_i, v_{i+1}) \leq L$$

then the refined path ρ satisfies the continuous constraint (2). Moreover, because of the right-inequality in (3), this path will have a reward (1) at least equal to $\sum_{v\in V} r_{\text{worst}}(v)\ \min\{|v|_{\text{worst}}, \#(p,v)\}$. This motivates the following graph-optimization problem:

Problem 2 (Discrete Aggregated Reward Budget, dARB)**.**

Instance: $\langle G, S, c, r, |\cdot|, L\rangle$, where $G = (V, E)$ denotes a directed graph with vertex set V and edge set E, $S \subset V$ set of initial vertices, $c : E \to [0, \infty)$ an edge-cost function, $r : V \to [0, \infty)$ a vertex-reward function, $|\cdot| : V \to \mathbb{N}$ a vertex-cardinality function, and L a positive integer.

Valid Solution: A (possibly self-intersecting) path[2] $p = (v_1, v_2, \ldots, v_k)$ in G with $v_1 \in S$ and $v_i \in V$, $\forall i$ such that $\sum_{i=1}^{k-1} c(v_i, v_{i+1}) \leq L$.

Objective: maximize the *total reward*

$$\sum_{v\in V} r(v)\ \min\{|v|, \#(p,v)\},$$

where $\#(p, v)$ denotes the number of times that the vertex v appears in the path p. □

Given a partition V of the region $\mathcal{R}$ and a path refinement algorithm satisfying Property 1, we can construct an instance $\langle G, S, c_{\text{worst}}, r_{\text{worst}}, |\cdot|_{\text{worst}}, L\rangle$ of the dARB Problem 2 by defining $G = (V, E)$ to be a fully connected graph whose vertices are the regions in V, $S := \{v \in V : v \cap \mathcal{R}_{\text{init}} \neq \emptyset\}$, and taking from Property 1 the edge-cost, the vertex-reward, and the vertex-cardinality functions. The dARB problem just defined is said to be *worst-case induced by the partition* V. It is then straightforward to prove the following result:

Lemma 1. *Consider an instance* $\langle G, S, c_{\text{worst}}, r_{\text{worst}}, |\cdot|_{\text{worst}}, L\rangle$ *of the worst-case dARB problem induced by a partition* V *and a path refinement algorithm satisfying Property 1.*

1. *Let* ρ *be a continuous path refined from a path* p *that is* admissible *for the worst-case induced dARB problem. Then* ρ *is admissible for the cCHS problem and*

$$R[\rho] \geq \sum_{v\in V} r(v)\ \min\{|v|, \#(p,v)\}. \tag{8}$$

2. *Denoting by* $R^*[L]$ *and* $R^*_{\text{worst}}[L]$ *the optimal rewards for the cCHS and the worst-case induced dARB problems, respectively, we have that*

$$R^*[L] \geq R^*_{\text{worst}}[L].$$

Proof of Lemma 1. The first statement has already been discussed and the second one results from the fact that selecting ρ to be a continuous path refined from an *optimal* path p^* for the worst-case induced dARB problem, we have that

$$R^*[L] \geq R[\rho] \geq \sum_{v\in V} r(v)\ \min\{|v|, \#(p^*,v)\} = R^*_{\text{worst}}[L],$$

where the first inequality follow from the definition of $R^*[L]$, the seconds from (8), and the last equality from the optimality of p^*. ■

[2]Often this type of paths are called walks.

2.2 Upper bound on the reward

We now formulate two dARB problems that can be used to construct lower bounds on the achievable reward for the cCHS problem. Such bounds can be used to determine whether or not the reward of a path obtained by refining the optimal solution to the worst-case dARB problem is very far from the optimal one.

To obtain an upper bound on the achievable reward, one needs to define "optimistic" instances $\langle G, S, c_{\text{best}}, r_{\text{best}}, |\cdot|_{\text{best}}, L\rangle$ of the dARB problem. The simplest of these can be formulated as follows: Take a partition V of the region $\mathcal{R}$ (not necessarily the same as for the worst-case dARB problem) and define $G = (V, E)$ to be a fully connected graph whose vertices are the regions in V; $S := \{v \in V : v \cap \mathcal{R}_{\text{init}} \neq \emptyset\}$;

$$c_{\text{best}}(v, v') = \min_{x \in v, y \in v'} J[x, y], \qquad \forall v, v' \in V,$$

where $J[x, y]$ denotes the minimum cost incurred when going from $x \in \mathcal{R}$ to $y \in \mathcal{R}$ (or some lower bound on it); and

$$|v|_{\text{best}} = 1, \qquad\qquad r_{\text{best}}(v) = \int_{\mathcal{S}[v]} f(x)dx,$$

where $\mathcal{S}[v] := \cup_{x \in v} \mathcal{S}[x]$ denotes the whole area that can be scanned from the region v. We call this the *type I best-case dARB problem induced by the partition V*. This problem is motivated by the "optimistic" assumption that as soon as one enters a particular region v, the searcher would be able to immediately collect the reward $\int_{\mathcal{S}[v]} f(x)dx$ associated with all the area that can be scanned from the region.

It is possible to construct a less conservative best-case instance of the dARB problem as follows: Take a partition V of the region $\mathcal{R}$ and a positive constant $\ell := \frac{L}{k}$, with k a positive integer. Using these, define the graph $G = (V, E)$ and the initial vertex set S as before, but with the edge cost defined by

$$c_{\text{best}}(v, v') = \max\Big\{\ell, \min_{x \in v, y \in v'} J[x, y]\Big\}, \qquad \forall v, v' \in V;$$

and the vertex reward $r_{\text{best}}(v)$, $v \in V$ defined to be an upper bound on the maximum reward for an instance of the cCHS Problem 1 starting anywhere in v with a cost bounded by ℓ, for the same probability density $f(\cdot)$ and continuous cost $c(\cdot)$ as the original cCHS problem. Each vertex cardinality $|v|_{\text{best}}$ should then be an upper bound on

$$\frac{\int_{\mathcal{S}_\ell[v]} f(x)dx}{r_{\text{best}}(v)},$$

where $\mathcal{S}_\ell[v]$ denotes the set of all points that could be scanned starting from the region v, with a cost not exceeding ℓ. We call this the *type II best-case dARB problem induced by the partition V and the cost-bound ℓ*. Computing tight bounds for the functions that define type II best-case induced dARB problems can be as hard as solving the original cCHS problem. In fact, for $\ell = L$ one essentially has to solve the original problem, but for smaller values of ℓ one only has to solve simpler cCHS for which the searcher motion is severely constrained. However, smaller values for ℓ will generally lead to looser bounds (possibly worse than those obtained for the type I dARB problem).

Theorem 1. *Consider an instance $\langle G, S, c_{\text{best}}, r_{\text{best}}, |\cdot|_{\text{best}}, L\rangle$ of either the type I or the type II best-case dARB problem induced by a partition V (and the cost-bound ℓ for type II). Denoting by $R^*[L]$ and $R^*_{\text{best}}[L]$ the optimal rewards for the cCHS and the best-case induced dARB problems, respectively, we have that*

$$R^*_{\text{best}}[L] \geq R^*[L]. \tag{9}$$

Proof of Theorem 1. Let $\rho : [0, T] \to \mathcal{R}$ be an admissible path for the cCHS problem that satisfies the cost constraint (2) with equality and achieves a reward larger than or equal to $R^*[L] - \delta$ for some small[3] $\delta \geq 0$.

To prove the result for a type I best-case dARB problem, we pick a sequence of times $t_1 := 0 < t_2 < \cdots < t_k := T$ such that $\rho(t) \in v_i$, $\forall t \in [t_i, t_{i+1}]$. Since $\rho(0) \in \mathcal{R}_{\text{init}}$, we must have $v_1 \in S$. Moreover, since $\rho(t_i) \in v_i$ and $\rho(t_{i+1}) \in v_{i+1}$,

$$c_{\text{best}}(v_i, v_{i+1}) = \min_{x \in v_i, y \in v_{i+1}} J[x, y] \leq \int_{t_i}^{t_{i+1}} c\big(\rho(t)\big)dt,$$

therefore $\sum_{i=1}^{k-1} c_{\text{best}}(v_i, v_{i+1}) \leq \int_0^T c\big(\rho(t)\big)dt \leq L$, which means that the discrete path $p = (v_1, v_2, \ldots, v_k)$ is admissible for the type I problem. To estimate the reward of this path, we note that

$$\mathcal{S}[\rho] \subset \bigcup_{i=1}^{k} \mathcal{S}[v_i],$$

where $\mathcal{S}[\rho]$ denotes the area scanned along the continuous path ρ and $\mathcal{S}[v_i]$ the whole area that could be scanned from the region v_i. Since $r_{\text{best}}(v) = \int_{\mathcal{S}[v]} f(x)dx$, we conclude that

$$R[\rho] = \int_{\mathcal{S}[\rho]} f(x)dx \leq \sum_{v \in V} r_{\text{best}}(v) \min\{1, \#(p, v)\}.$$

Since the left hand side of the above inequality is larger than or equal to $R^*[L] - \delta$ and the right-hand-side is the reward of an admissible path for the best-case dARB problem, we conclude that $R^*[L] - \delta \leq R[\rho] \leq R^*_{\text{best}}[L]$. Inequality (9) then follows from the fact that δ can be made arbitrarily close to zero.

To prove the result for a type II best-case dARB problem, we pick a sequence of times $t_1 := 0 < t_2 < \cdots < t_{k+1} := T$ such that

$$\int_{t_i}^{t_{i+1}} c\big(\rho(t)\big)dt = \ell := \frac{L}{k}, \qquad \forall i \in \{1, 2, \ldots, k\}.$$

Suppose now that we define a path $p = (v_1, v_2, \ldots, v_{k+1})$, where each v_i denotes the region on which $\rho(t_i)$ lies. Since $\rho(0) \in \mathcal{R}_{\text{init}}$, we must have $v_1 \in S$. This sequence is admissible for the best-case dARB problem because from the definition of c_{best} we conclude that

$$c_{\text{best}}(v_i, v_{i+1}) \leq \max\Big\{\ell, \int_{t_i}^{t_{i+1}} c\big(\rho(t)\big)dt\Big\} = \frac{L}{k}, \qquad \forall i \in \{1, 2, \ldots, k\},$$

[3]The need for $\delta > 0$ only arises when the optimal reward cannot be achieved for any admissible path.

and therefore $\sum_{i=1}^{k+1} c_{\text{best}}(v_{i-1}, v_i) \leq L$. To estimate the reward of p, we note that[4]

$$\mathcal{S}[\rho] = \bigcup_{i=1}^{k} \mathcal{S}_i = \bigcup_{v \in p} \bar{\mathcal{S}}[v], \tag{10}$$

where $\mathcal{S}_i$ denotes the set of all points that the searcher can scan on the interval $[t_i, t_{i+1}]$ and $\bar{\mathcal{S}}[v]$ the union of those $\mathcal{S}_i$ for which $v_i = v$, *i.e.*,

$$\mathcal{S}_i := \bigcup_{t \in [t_i, t_{i+1}]} \mathcal{S}[\rho(t)], \quad \forall i \in \{1, 2, \ldots, k\}, \qquad \bar{\mathcal{S}}[v] := \bigcup_{\substack{i \in \{1,\ldots,k\}: \\ v_i = v}} \mathcal{S}_i, \quad \forall v \in V.$$

From (10), we then conclude that

$$R[\rho] = \int_{\mathcal{S}[\rho]} f(x)dx \leq \sum_{v \in p} \int_{\bar{\mathcal{S}}[v]} f(x)dx. \tag{11}$$

Since the path segment $\rho : [t_i, t_{i+1}] \to \mathcal{R}$ has cost ℓ and starts in v_i, we conclude that $r_{\text{best}}(v_i) \geq \int_{\mathcal{S}_i} f(x)dx$, from which we obtain

$$\int_{\bar{\mathcal{S}}[v]} f(x)dx \leq \sum_{\substack{i \in \{1,\ldots,k\}: \\ v_i = v}} \int_{\mathcal{S}_i} f(x)dx \leq \sum_{\substack{i \in \{1,\ldots,k\}: \\ v_i = v}} r_{\text{best}}(v) \leq r_{\text{best}}(v)\, \#(p, v). \tag{12}$$

On the other hand, any point in $\bar{\mathcal{S}}[v]$ belongs to some $\mathcal{S}_i$ and therefore can be scanned using a path segment $\rho : [t_i, t_{i+1}] \to \mathcal{R}$ starting in v with cost ℓ. This means that $\bar{\mathcal{S}}[v] \subset \mathcal{S}_\ell[v]$ and we conclude from the definition of $|v|_{\text{best}}$ that

$$\int_{\bar{\mathcal{S}}[v]} f(x)dx \leq \int_{\mathcal{S}_\ell[v]} f(x)dx \leq r_{\text{best}}(v)\, |v|_{\text{best}}. \tag{13}$$

Using (12) and (13) in (11), we obtain

$$R[\rho] \leq \sum_{v \in p} r_{\text{best}}(v)\, \min\big\{|v|_{\text{best}}, \#(p, v)\big\}.$$

So for a type II best-case dARB problem we also conclude that (9) holds. ■

3 Discrete Aggregated Reward Budget (dARB) Problem

It turns out that the dARB problem introduced above is still computationally difficult. In fact, it is NP-hard even when one imposes significant structure on the graph and the cost, reward, and cardinality functions. In particular, this is the case when one restricts the cardinality function to be always equal to one and the graph to be planar with unit costs and rewards or when the graph is a *unit grid*, *i.e.*,

$$V = \Big\{(i, j) \in \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\}\Big\}, \qquad E = \Big\{\big\{(i, j), (k, \ell)\big\} : |i - j| + |k - \ell| = 1\Big\},$$

with unit costs and binary rewards[5]. The following lemma states these hardness results.

[4]With some abuse of notation, given a path $p = (v_1, v_2, \ldots, v_k)$ and a vertex v, we write $v \in p$ to express the statement that v is one of the v_i.

[5]For computational complexity results we restrict cost and reward values in our problems to take integer values. The *decision problem* for SdARB problems (as required for NP-hardness proofs) provides an additional real number R and asks if there is a valid solution of total reward *at least* R.

Lemma 2. *The dARB problem is NP-hard even when* $|v| = 1$ *for every* $v \in V$ *and*

(a) G *is planar bipartite with the maximum degree of any vertex equal to* 3, $c(e) = 1$ *for every* $e \in E$, *and* $r(v) = 1$ *for every* $v \in V$; *or*

(b) G *is a unit grid graph ,* $c(e) = 1$ *for every* $e \in E$, *and* $r(v) \in \{0, 1\}$ *for every* $v \in V$. □

Proof of Lemma 2. It suffices to prove the result for the special case when the graph G is undirected (since an undirected graph can be converted to an equivalent directed graph by replacing each undirected edge by two directed edges in opposite directions) and when $S = V$.

We prove part (a) as follows. The Hamiltonian path problem for a graph G with n vertices is NP-complete even if G is planar bipartite with maximum degree of any vertex equal to 3 [15]. One can see that by setting $r(v) = 1$ for every $v \in V$, $c(e) = 1$ for every $e \in E$, and $L = n - 1$, G has a Hamiltonian path if and only if the total reward collected by the instance of the dARB problem is n.

We prove part (b) as follows. It is known that the Hamiltonian path problem is NP-hard for graphs which are vertex-induced subgraphs of a unit grid [15]. Given an instance I of the Hamiltonian path problem on such graphs, we consider a corresponding instance I' of the dARB problem in which the graph is a *minimal* unit grid graph $G' = (V', E')$ *containing* the vertex-induced subgraph $G = (V, E)$ of n vertices in I, and

$$r(v) = \begin{cases} 1 & \text{if } v \in V \\ 0 & \text{if } v \in V' - V \end{cases}$$

and $L = n - 1$. If I has a Hamiltonian path then I' has a solution with a maximum total reward of n. Conversely, suppose that I' has a solution with a maximum total reward of n. Then, this solution must visit all the vertices in V using $n - 1$ edges and thus it is a Hamiltonian path of G. ■

3.1 Approximation Algorithms for the dARB problem

It turns out that the interesting cases of the dARB problem can be solved in polynomial time provided that one is willing settle for paths that do not quite yield the maximum reward. In particular, we are interested in the following slightly restricted version of the dARB problem for the remainder of this section.

Problem 3 (Symmetric Discrete Aggregated Reward Budget, SdARB)**.** The SdARB problem is the dARB problem with the following restrictions:

- the graph $G = (V, E)$ is *undirected*,
- $|v| = 1$ for every $v \in V$,
- the set S of initial vertices is equal to V. □

In the sequel, we denote by $\mathrm{OPT}_{\mathrm{SdARB}}(L)$ the optimum value of the objective function for a given instance $\langle G, s, c, r, |\cdot|, L\rangle$ of the SdARB problem, as a function of L.

Restricting our attention to problems with unit vertex-cardinality functions does not introduce loss of generality. In fact, given an instance $\langle G, S, c, r, |\cdot|, L\rangle$ of the dARB problem with $|v| > 1$ for

some vertices $v \in V$, we can construct another instance $\langle \bar{G}, \bar{S}, \bar{c}, \bar{r}, \|\cdot\|, L\rangle$ with $\|\bar{v}\| = 1$ for every $\bar{v} \in \bar{V}$. This is done by constructing the graph $\bar{G} = (\bar{V}, \bar{E})$ by expanding each vertex v in the graph $G = (V, E)$ into $|v|$ vertices with reward $r(v)$ connected to each other by edges with cost $c(v, v)$. Moreover, for any edge in G between vertices v_1 and v_2, the graph $\bar{G}$ must have edges with cost $c(v_1, v_2)$ between all vertices that result from the expansion of v_1 and all vertices that result from the expansion of v_2. The new start vertex set $\bar{S}$ consists of all vertices that result from the expansion of the vertices in S (see Figure 4). With this construction, for any feasible path of the new dARB problem we can construct a feasible path for the original problem with the same reward and vice versa. However, the new problem has larger input size since it has $\sum_{v\in V} |v| = O(\#V\#C)$ vertices and $\sum_{v\in V} |v|\big(|v|-1\big)+\sum_{(v_1,v_2)\in E} |v_1|\,|v_2| = O(\#V\#C^2+\#E\#C^2)$ edges, where $\#C := \max_{v\in V} |v|$.

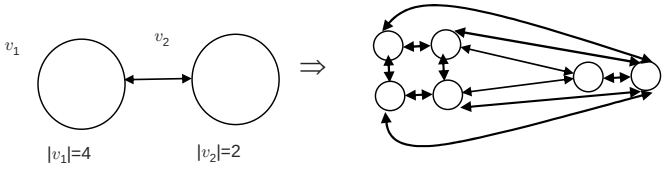


Figure 4. Illustration of the transformation of the given graph G to $\bar{G}$.

Taking the set of initial vertices S to be the whole vertex set V is somewhat restrictive but so far seems to be needed for our approximation results (*e.g.*, Proposition 1). In terms of the original continuous problem, this corresponds to the situation where the initial region $\mathcal{R}_{\text{init}}$ is the whole $\mathcal{R}$ or at least intersects all the $v \in V$. If this is not the case, then one may need to "provision" some cost to take the searcher from its initial position to an arbitrary point in $\mathcal{R}$.

To state the quality of approximation that we can obtain, we recall the following standard definition:

Definition 1. An *ε-approximate solution* (or simply an ε-approximation) of a maximization problem is a solution obtained in polynomial time with an objective value no smaller than $1/\varepsilon$ times the value of the optimum. Analogously, an ε-approximation of a minimization problem is a solution obtained in polynomial time with an objective value no larger than ε times the value of the optimum. □

Theorem 2. *Consider an instance of the SdARB problem. Then,*

(a) For any constant $\varepsilon > 0$, an δ-approximate solution to the RB problem can be found in polynomial time, where

$$\delta = \begin{cases} 2+\varepsilon & \text{if } c(e) = 1 \text{ for every } e \in E \\ 5+\varepsilon & \text{otherwise.} \end{cases}$$

(b) If $r(v) = 1$ for every $v \in V$ and $c(e) = 1$ for every $e \in E$, then a 2-approximate solution to the SdARB problem can be computed in $O(\#V + \#E)$ time, where $\#V$ and $\#E$ denote the number of vertices and edges, respectively. □

3.1.1 Proof of Theorem 2(a)

To prove statement (a) of Theorem 2, we need to consider a "dual" version of the SdARB problem which is defined as follows.

Problem 4 (Symmetric Discrete Aggregated Reward Quota, SdARQ)**.**

Instance: $\langle G, c, r, R\rangle$, where $G = (V, E)$ is an undirected graph with vertex set V and edge set E, $c : E \to [0, \infty)$ is an edge cost function, $r : V \to [0, \infty)$ is a vertex reward function, and R a positive integer.

Valid Solution: A (possibly self-intersecting) path $p = (v_1 = s, v_2, \ldots, v_k)$ in G with $v_i \in V$ such that $R[p] := \sum_{v \in P} r(v) \geq R$ where P denotes the *set* of vertices in the path p.

Objective: minimize the *total edge cost* $C[p] := \sum_{i=1}^{k-1} c(v_i, v_{i+1})$. □

To enable us to use existing results, we need to actually consider the "cycle" versions of the SdARB and SdARQ problems, which are defined as follows.

Problem 5 (Cycle version of SdARB)**.** This is same as the SdARB problem except that we need to find a (possibly self-intersecting) cycle instead of a (possibly self-intersecting) path, *i.e.*, we set $v_1 = v_k$ in Definition 2.

Problem 6 (Cycle version of SdARQ)**.** This is same as the SdARQ problem except that we need to find a (possibly self-intersecting) cycle instead of a (possibly self-intersecting) path, *i.e.*, we set $v_1 = v_k$ in Definition 4.

In the sequel we denote by $\mathrm{OPT}_{\mathrm{SdARB-cycle}}(L)$ the optimum value of the objective function for a given instance $\langle G, s, c, r, |\cdot|, L\rangle$ of the cycle version of the SdARB problem, as a function of L. Similarly, let $\mathrm{OPT}_{\mathrm{SdARQ-cycle}}(R)$ denotes the optimum value of the objective function for a given instance $\langle G, c, r, R\rangle$ of the cycle version of the SdARQ problem as a function of R.

Our strategy to prove statement (a) of Theorem 2 is shown in Figure 5. A good approximate solution to the cycle version of the dual SdARQ problem is shown to correspond to a good approximate solution of the cycle version of the SdARB problem via a binary search similar to that by Johnson *et al.* [16], path decompositions and Eulerian tours via doubling edges. To solve the cycle version of the SdARQ problem, we need to use the $(2 + \varepsilon)$-approximation results on the k-MST problem by Arora and Karakostas [2] that builds upon the 3-approximation results on the same problem by Garg [13]. Finally, the cycle versions of the problems can be translated to the path versions without any loss of quality of approximation.

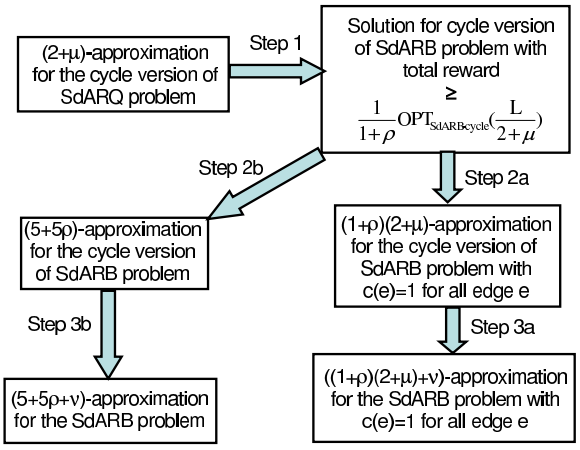


Figure 5. Series of reductions used to prove statement (a) of Theorem 2. Each reduction is *polynomial time*; μ, ν and ρ are arbitrary positive constants less than 1. The quantity ε in the theorem is equal to $5 + 5\rho + \nu$ for the general case and is equal to $(1 + \rho)(1 + \mu) + \nu$ for the case when $c(e) = 1$ for every $e \in E$.

Although the SdARB and SdARQ problems defined above seem to be novel, there are a few related problems that will be useful:

The budget and quota problems in [16]: These are similar to the problems defined above, except that they only searched for a subtree (vertex induced subgraph with no cycles), which may not necessarily be a path[6]. Nonetheless many of the ideas there are also of use to us.

Metric k-TSP An input to the metric k-traveling salesman problem is a weighted graph in which the edge costs satisfy the *triangle inequality*, *i.e.*, for every three vertices u, v and w the edges costs satisfy the inequality $c(u,v) + c(v,w) \geq c(u,w)$. The goal is to produce a *simple* (*i.e.*, non-self-intersecting) cycle of minimum total edge cost visiting at least k vertices that includes a specified vertex s. The authors in [2] provide a $(2+\varepsilon)$-approximate solution to this problem for any constant $\varepsilon > 0$ via a primal-dual schema.

Now we discuss the various steps in the proof as outlined in Figure 5.

Theorem 3. *For any constant $0 < \mu < 1$, we can design a polynomial-time $(2+\mu)$-approximation algorithm for the cycle version of the SdARQ problem.*

Proof of Theorem 3. Implicitly replace every vertex v with $r(v) > 0$ by one original vertex v' of reward 0 (which will be connected to other vertices in the graph) and an additional $r(v)$ vertices, each of reward 1, connected to v' with edges of zero cost. Obviously, an optimal solution of the SdARQ problem in the original graph remains an optimal solution of the SdARQ problem in this new graph.

Now we run the polynomial time $(2+\mu)$-approximation algorithm for the metric R-TSP problem in [2] on our new graph. We observe the following:

- The new zero-cost edges connected to a vertex v are dealt with implicitly in the TREEGROW step in [2] by coalescing the vertices together. This ensures that the algorithm indeed runs in polynomial time (as opposed to pseudo-polynomial time).
- The metric property of the graph is necessary to avoid self-intersection of the solution path in the R-TSP problem. Since the SdARQ problem allows self-intersection, we do not need the metric property.
- The authors in [2] solve the problem by finding a solution of the R-MST problem, doubling every edge and then taking short-cuts (to avoid self-intersection) to get *exactly* R vertices. However, for the SdARQ problem we do not count the reward of the same vertex twice, hence taking a simple Eulerian tour on the solution of the R-TSP problem with every edge doubled does not change the total reward.

As a result, we can obtain in polynomial time a $(2+\mu)$-approximate solution (for any constant $\mu > 0$) to the SdARQ problem. ■

Lemma 3. *Assume that we have a polynomial time δ-approximation algorithm for the cycle version of the SdARQ problem for some constant $\delta > 1$. Then, for* any *constant $\rho > 0$, there is a polynomial time approximation algorithm for the cycle version of the SdARB problem that returns a solution with a total reward of at least $\frac{1}{1+\rho}\,\mathrm{OPT}_{\mathrm{SdARB-cycle}}\left(\frac{1}{\delta}L\right)$.*

[6]By doubling every edge in a subtree and finding an Eulerian tour on the new graph, we do get a self-intersecting cycle; however, the total edge cost of such a path is twice that of the given subtree.

Proof of Lemma 3. Our proof is similar in nature to a proof in [16] that related the budget and quota problems for connected subtrees of a graph. We can do a binary search over the range $[0, \sum_{i=1}^{n} r(v_i)]$ for the total reward in the given approximation algorithm for the cycle version of the SdARQ problem to provide us in polynomial time a total reward value A such that the solution obtained by the δ-approximation algorithm has a total edge cost of at most L if the total reward is at least A, but has a total edge cost greater than L if the total reward is at least $A(1+\rho)$. We then output the solution to the cycle version SdARQ problem with a total reward of at least A as our approximate solution to the the cycle version of the SdARB problem. By choice of A, an optimal solution to the cycle version of the SdARQ problem with a total reward of $A(1+\rho)$ must have a total cost greater than $\frac{L}{\delta}$. Hence $\mathrm{OPT}_{\mathrm{SdARB-cycle}}\left(\frac{L}{\delta}\right) \leq A(1+\rho) \equiv A \geq \frac{1}{1+\rho}\mathrm{OPT}_{\mathrm{SdARB-cycle}}\left(\frac{L}{\delta}\right)$. ■

Remark 1. Setting $\delta = 2+\mu$ one can now perform Step 1 in Figure 5.

Proposition 1.

(a) $\mathrm{OPT}_{\mathrm{SdARB}}(\frac{L}{2+\mu}) \geq \frac{1}{5}\mathrm{OPT}_{\mathrm{SdARB}}(L)$ *for any constant* $0 < \mu < 1$.

(b) Assume that $c(e) = 1$ *for every edge* e. *Then,* $\mathrm{OPT}_{\mathrm{SdARB}}(\left\lceil \frac{L}{2+\mu} \right\rceil) \geq \frac{1}{2+\mu}\mathrm{OPT}_{\mathrm{SdARB}}(L)$ *for any constant* $0 < \mu < 1$.

Proof of Proposition 1. To prove (a), it suffices to show that $\mathrm{OPT}_{\mathrm{SdARB}}(\frac{L}{3}) \geq \frac{1}{5}\mathrm{OPT}_{\mathrm{SdARB}}(L)$ since $\mathrm{OPT}_{\mathrm{SdARB}}(\frac{L}{3}) \geq \mathrm{OPT}_{\mathrm{SdARB}}(\frac{L}{2+\mu})$. Assume that $p = (v_1, v_2, \ldots, v_k)$ is an optimal solution with a total reward of $\mathrm{OPT}_{\mathrm{SdARB}}(L)$. If $r(v_i) \geq \frac{1}{5}\mathrm{OPT}_{\mathrm{SdARB}}(L)$ for some v_i then our claim is true by picking this vertex. Otherwise, starting with the first edge in p, partition p into three disjoint subpaths such that total reward of the each of the first two subpaths is greater than $\frac{1}{5}\mathrm{OPT}_{\mathrm{SdARB}}(L)$ but the total reward of each of the first two subpaths excluding its last edge is at most $\frac{1}{5}\mathrm{OPT}_{\mathrm{SdARB}}(L)$. This ensures that the total reward of each of the first two subpaths is at most $\frac{2}{5}\mathrm{OPT}_{\mathrm{SdARB}}(L)$, Thus, the total reward of the last (third) subpath is at least $\mathrm{OPT}_{\mathrm{SdARB}}(L) - \frac{4}{5}\mathrm{OPT}_{\mathrm{SdARB}}(L) = \frac{1}{5}\mathrm{OPT}_{\mathrm{SdARB}}(L)$. At least one of the three subpaths must have a total edge cost of at most $\lfloor \frac{L}{3} \rfloor$.

To prove (b), note that an optimal path of total cost L can be partitioned into $\frac{L}{\lceil \frac{L}{2+\mu} \rceil} \leq 2+\mu$ disjoint paths each of total edge cost at most $\lceil \frac{L}{2+\mu} \rceil$. At least one of these paths must have a total reward of at least $\frac{1}{2+\mu}\mathrm{OPT}_{\mathrm{SdARB}}(L)$. ■

Remark 2. Proposition 1 allows us to perform Steps 2a and 2b in Figure 5.

Lemma 4. *Assume that we have a polynomial time* δ*-approximation algorithm (for some* $\delta > 1$*) for the cycle version of SdARB problem. Then, for any constant* $0 < \nu < 1$*, there is a polynomial time* $(\delta + \nu)$*-approximation algorithm for SdARB problem.*

Proof of Lemma 4. Let A' be the total reward returned by the δ-approximation algorithm for the cycle version of the SdARB problem. Hence, $A' \geq \frac{1}{\delta}\mathrm{OPT}_{\mathrm{SdARB-cycle}}(L)$. Setting $x = \lceil 1 + \frac{\delta}{\nu} \rceil$, since x is a constant, we can solve the SdARB problem for paths involving at most x edges in polynomial time (*e.g.*, trivially by checking all $\binom{|E|}{x} = O(|E|^x)$ subsets of x edges for a possible solution). Otherwise, an optimal solution for the SdARB problem (and hence for cycles version of

the SdARB problem as well) involves more than x edges. Now, by removing the least cost edge from the solution for A', we get a solution with total reward $A \geq \left(1 - \frac{1}{x}\right) A'$ of the SdARB problem. Since $A \geq \left(1 - \frac{1}{x}\right) A'$ and $\mathrm{OPT}_{\mathrm{SdARB-cycle}}(L) \geq \mathrm{OPT}_{\mathrm{SdARB}}(L)$, we have that

$$\frac{A}{\mathrm{OPT}_{\mathrm{SdARB}}(L)} \geq \left(1 - \frac{1}{x}\right) \frac{A'}{\mathrm{OPT}_{\mathrm{SdARB-cycle}}(L)}.$$

Moreover, $A' \geq \frac{1}{\delta} \mathrm{OPT}_{\mathrm{SdARB-cycle}}(L)$, therefore

$$\frac{A}{\mathrm{OPT}_{\mathrm{SdARB}}(L)} \geq \left(1 - \frac{1}{x}\right) \frac{1}{\delta} \geq \frac{1}{\delta + \nu},$$

where the last inequality holds by the choice of x. ■

Remark 3. Setting $\delta = 5 + 5\rho$ (respectively, $\delta = (1 + \rho)(2 + \mu)$) in the above lemma allows us to perform Step 3a (respectively, Step 3b) in Figure 5.

3.1.2 Proof of Theorem 2(b)

We do a depth-first-search (DFS) on G starting at any vertex s that computes a DFS tree. Now we replace every edge in the DFS tree by two edges, compute an Eulerian cycle and output the path consisting of the first L edges starting at s in this Eulerian cycle. Obviously, $\mathrm{OPT}_{\mathrm{SdARB}}(L) \leq L$. Since we replaced each undirected edge by two directed edges, we collect a total reward of at least $\frac{L}{2} + 1$.

3.2 Reducing the search space for the dARB problem

When the dARB is sufficiently small one may pursue an exhaustive search. In this section, we prove a few properties of the optimal solution to the dARB problem that can significantly decrease the search space for an exhaustive search. We consider instances $\langle G, S, c, r, |\cdot|, L\rangle$ of the dARB problems that are *subadditive*, meaning that the graph $G = (V, E)$ is fully connected and

$$c(v_1, v_2) + c(v_2, v_3) \geq c(v_1, v_3) + c(v_4, v_4), \qquad \forall v_1, v_2, v_3, v_4 \in V.$$

When $c(v, v) = 0$, $\forall v \in V$, this simply expresses a triangular inequality. A similar assumption is made in [21]. Typically, worst-case induced dARB problems are subadditive. In this case, the search space for optimal paths can be significantly reduced.

Theorem 4. *The maximum achievable reward for a subadditive dARB problem does not increase if we restrict the valid paths $p = (v_1, v_2, \ldots, v_N)$ to satisfy:*

(a) If $v_i = v_j$, $i < j$ then $v_k = v_i$ for every $k \in \{i, i+1, \ldots, j\}$.

(b) The number of times $\#(p, v)$ that a vertex $v \in V$ appears in p never exceeds $|v|$.

(c) If a vertex $v \in V$ appears in p and $r(v) > \min_{v_i \in p} r(v_i)$, then the number of times $\#(p, v)$ that v appears in p is exactly $|v|$.

In essence Theorem 4 allows one to simply focus the search on the *order one needs to visit the different vertices (without repetitions)*. This means that only an exhaustive enumeration of the vertex ordering is needed, because the time spent on each vertex is uniquely determined once an order has been chosen.

Theorem 4 also simplifies considerably the construction of paths refinement algorithms for which the bounds in Property 1 are tight. This is because in the discrete paths p that need to be refined, each region v only appears multiple times back to back (typically $|v|$ times) and therefore one can estimate quite precisely the value of $c_{\text{worst}}(v, v')$, especially for $v = v'$ (*cf.*, discussion in Section 4).

Proof of Theorem 4. For (a), consider a path $p = (v_1, v_2, \ldots, v_k)$ with total cost $C[p]$ for which $v_i = v_j$, $i < j$ but $v_{j-1} \neq v_i$. If we then construct a path

$$p' = \big(v_1, v_2, \ldots, v_{i-1}, v_i, v_j = v_i, v_{i+1}, \ldots, v_{j-1}, v_{j+1}, \ldots, v_N\big)$$

(v_j, which is equal to v_i, was moved to right after v_i), p' has exactly the same reward as p and, because of subadditivity, its cost $C[p']$ satisfies

$$C[p'] = C[p] - \big(c(v_{j-1}, v_j) + c(v_j, v_{j+1})\big) + c(v_i, v_i) + c(v_{j-1}, v_{j+1}) \leq C[p].$$

Since p' has the same reward as p and no worse cost, it will not increase the maximum achievable reward for the dARB problem. By induction, we conclude that any path for which (a) does not hold will also not improve the maximum achievable reward for the dARB problem.
For (b), consider a path p with total cost $C[p]$ for which v_i already appeared in p at least $|v_i|$ times before i. If we then construct a path

$$p' = \big(v_1, v_2, \ldots, v_{i-1}, v_{i+1}, \ldots, v_N\big)$$

(v_i was removed), p' has exactly the same reward as p and, because of subadditivity, its cost $C[p']$ satisfies

$$C[p'] = C[p] - \big(c(v_{i-1}, v_i) + c(v_i, v_{i+1})\big) + c(v_{i-1}, v_{i+1}) \leq C[p].$$

Since p' has the same reward as p and no worse cost, it will not increase the maximum achievable reward for the dARB problem. By induction, we conclude that any path in which v appears more than $|v|$ times will also not improve the maximum achievable reward for the dARB problem.
For (c), consider a path p with total reward $R[p]$ and total cost $C[p]$ for which the vertex v_i appears less than $|v_i|$ times and there is a vertex v_j such that $r(v_i) > r(v_j)$. If we then construct a path

$$p' = \big(v_1, v_2, \ldots, v_{i-1}, v_i, v_i, v_{i+1}, \ldots, v_{j-1}, v_{j+1}, \ldots, v_N\big)$$

(v_j was replaced by an extra v_i, right after the original one), its reward $R[p']$ satisfies

$$R[p'] \geq R[p] - r(v_j) + r(v_i) > R[p].$$

and, again because of subadditivity, its cost $C[p']$ satisfies

$$C[p'] = C[p] - (c(v_{j-1}, v_j) + c(v_j, v_{j+1})) + c(v_i, v_i) + c(v_{j-1}, v_{j+1}) \leq C[p].$$

Since p' has better reward and no worse cost than p, it will not increase the maximum achievable reward for the dARB problem. By induction, we conclude that any path for which (c) does not hold will also not improve the maximum achievable reward for the dARB problem. ■

4 Numerical results

In this section we compare the performance of the aggregation/refinement approach proposed in this paper with other (simpler) alternatives. To this effect we consider the rectangular search region in Figure 2, where the dark "walls" limit the motion of the searching agent. This region has been tiled with small squares of unit area and we assume that the searcher can move from the center of one tile to the center of any one of the four adjacent tiles with unit cost. Each of these tiles corresponds to one of the small circles visible in Figure 2. From the center of a tile, the searcher can see the whole tile and nothing more.

Within this region we generated many target density functions f and compared the performance of different search algorithms in terms of the reward achieved for different cost bounds L. The target density functions used were linear combinations of Gaussian distributions with randomized means and variances. The following three algorithms were compared:

Random walk The searcher starts at an arbitrary node and moves to a random adjacent node until the cost reaches the bound L.

Greedy algorithm The searcher starts at the node with maximum reward and moves to the unvisited adjacent node with highest reward until the cost reaches the bound L. If all adjacent nodes have the same reward or have all been previously visited, this algorithm chooses a random node.

Worst-case dARB algorithm The path followed by the searcher is obtained by solving a worst-case dARB problem induced by an appropriate partition of the region. For all the problem generated, the number of regions in the partition was small, which allowed us to solve the worst-case dARB problem by an exhaustive search.

The construction of the partition was fully automated and changed from one target density function to the next. The partition was obtained by constructing a graph with one tile per node and edges between adjacent tiles. This graph was partitioned using the algorithm in [9] by defining the cost of "cutting" an edge between adjacent tiles with rewards r_1 and r_2 to be

$$e^{-|r_1-r_2|}.$$

The algorithm in [9] was then used to compute a partition of the tiles that minimizes the total cost of all the "cuts." This partition will favor cutting edges with very different rewards because the cost of cutting such edges is low. The partition costs were normalized to obtain a doubly stochastic partition matrix [9]. Figure 2 shows a partition obtained in this fashion.

The path refinement algorithm used takes advantage of the results in Section 3.2, from which one concludes that the searcher will only be sent once to each element of the partition (perhaps staying there for several time steps). This means that the discrete paths that need to be refined will always be of the form

$$p = (\underbrace{v_1, v_1, \ldots, v_1}_{n_1 \text{ times}}, \underbrace{v_2, \ldots, v_2}_{n_2 \text{ times}}, \ldots \underbrace{v_k, \ldots, v_k}_{n_k \text{ times}}).$$

where the v_i are elements of the partition. To refine such path one constructs a continuous path that "sweeps" the n_1 tiles of v_1 with largest reward, then "sweeps" the n_2 tiles of v_2 with largest

reward, and so on until the region v_k. The graph partition algorithm guarantees that the reward on each region is fairly uniform and therefore it is straightforward to construct paths that "sweep" n_i tiles in the region v_i with a cost equal to n_i[7]. The corresponding worst-case edge-cost is then given by

$$c_{\text{worst}}(v, v') = \begin{cases} \max_{x \in v, y \in v'} d(x, y) & c \neq v' \\ 1 & v = v'; \end{cases} \tag{14}$$

the vertex-cardinality $|v|_{\text{worst}}$ is equal to the number of tiles in the region v; and the vertex-reward $r_{\text{worst}}(v)$ is equal to the average reward of the tiles in v. Note that since we sweep the n_i tiles of largest reward, we always get a reward of, at least, n_i times the average reward. In (14) $d(x, y)$ refers to the length of the shortest path between the tiles x and y. Figure 6 shows the regions swept by a typical optimal path.

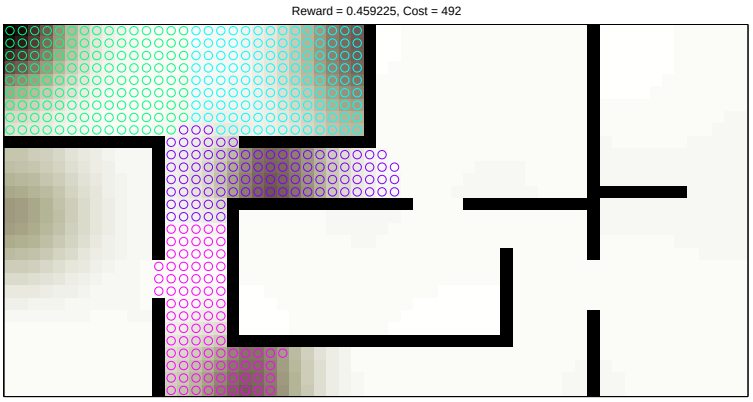


Figure 6. Solution to dARB problem. The circles represent tiles swept by the path.

Performance Evaluation

Figure 7 shows the reward attained by each algorithm as a function of the cost-bound. The results shown were averaged over 100 experiments. Not surprisingly both the dARB and the greedy algorithms greatly out-perform a random search. The greedy algorithm also performs poorly when the cost-bound is sufficiently large so as to allow the searcher to visit several local maxima of the probability density function. In this case, the dARB algorithm outperforms the remaining ones. The results of dARB are also more consistent, which results in smaller standard deviations.

5 Conclusions

We proposed an aggregation-based approach to the problem of searching for a hidden target by an autonomous robot using local sensory information. We investigated the cost penalty incurred due to (i) the discretization of the problem and (ii) the attempt to approximately solve the discrete NP-hard problem in polynomial time. Numerical simulations show that the algorithm proposed behaves significantly better than naive approaches such as a random walk or a greedy algorithms.

[7]We recall that we are associating a unit cost with the motion between adjacent tiles.

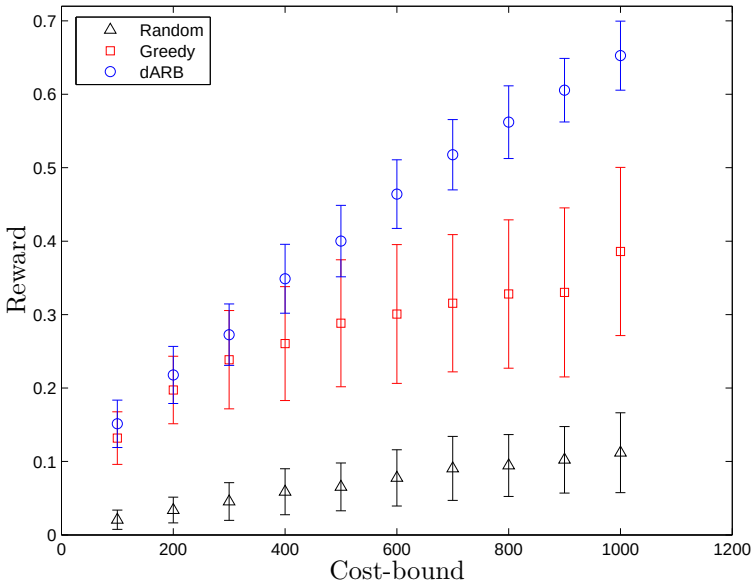


Figure 7. Comparison between random, greedy, and dARB algorithms for various cost-bounds. The cost-bound appears in the x-axis and the corresponding reward (*i.e.*, probability of capture) appears in the y axis. These results were obtained by averaging 100 trials and the error bars represent one standard deviation around the mean.

We are currently working on algebraic algorithms that produce partitions of the continuous search space for which the procedure proposed in this paper results in a small cost penalty. These algorithms are described in [9] and inspired by the results in [23] on state aggregation in Markov chains. Another avenue for future research is the search for mobile targets, or more generally, when the probability distribution f is not constant.

Acknowledgements

We would like to thank Robert Sloan and György Turán for pointing out reference [15] and for helpful discussions.

References

[1] R. Alur, T. Henzinger, G. Lafferriere and G. J. Pappas. *Discrete abstractions of hybrid systems*, Proceedings of IEEE, 88 (2), pp. 971-984, July 2000.

[2] S. Arora and G. Karakostas. *A* $2+\varepsilon$ *approximation for the k-MST problem*, 11^{th} ACM-SIAM Symposium on Discrete Algorithms, 754-759, 2000.

[3] D. V. Chudnovksy and G. V. Chudnovksy (editors). *Search Theory: Some Recent Developments*, Volume 112 of Lecture Notes in Pure and Applied Mathematics, Marcel Dekker, New York, 1989.

[4] J. N. Eagle. *Optimal Search for a Moving Target when the Search Path is Constrained*, Operations Research, 32(5), pp. 1107-1115, 1984.

[5] J. N. Eagle and J. R. Yee. *An Optimal Branch and Bound Procedure for the Constrained Path, Moving Target Search Problem*, Operations Research, 38 (1), pp. 110-114, 1990.

[6] J. R. Frost and L. D Stone. *Review of Search Theory: Advances and Applications to Search and Rescue Decision Support*, Technical Report CG-D-15-01, U.S. Coast Guard Research and Development Center, Gronton, CT, September, 2001 (URL `http://www.rdc.uscg.gov/reports/2001/cgd1501dpexsum.pdf`).

[7] Shmuel Gal. *Continuous Search Games*, in Chudnovksy and Chudnovksy [3], pp. 33-53.

[8] J. Hershberger and S. Suri. An optimal algorithm for euclidean shortest paths in the plane. *SIAM J. on Computing*, 28(6):2215–2256, 1999. ISSN 0097-5397.

[9] J. P. Hespanha *An efficient MATLAB Algorithm for Graph Partitioning*, Technical Report, University of California, Santa Barbara, CA, October 2004 (URL `http://www.ece.ucsb.edu/~hespanha/techreps.html`).

[10] J. P. Hespanha and H. Kizilocak. *Efficient Computation of Dynamic Probabilistic Maps*, 10th Mediterranean Conference on Control and Automata, July 2002.

[11] J. P. Hespanha, M. Prandini and S. Sastry. *Probabilistic Pursuit-Evasion Games: A One-Step Nash Approach*, 39th Conference on Decision and Control, volume 3, pp. 2272-2277, December 2000,

[12] M. R. Garey, D. S. Johnson and R. E. Tarjan. *The planar Hamiltonian circuit problem is NP-complete*, SIAM Journal of Computing, 5, 704-714, 1976.

[13] N. Garg. *A* 3*-approximation for the minimum tree spanning k vertices*, 37^{th} Annual Symposium on Foundations of Computer Science, 302-309, 1996.

[14] K. B. Haley and L. D. Stone (editors). *Search theory and Applications*, volume 8 of NATO Conference Series, Plenum Press, New York, 1980.

[15] A. Itai, C. H. Papadimitriou and J. L. Szwarcfiter. *Hamiltonian Paths in Grid Graphs*, SIAM Journal of Computing, 11 (4), 676-686, November 1982.

[16] D. S. Johnson, M. Minkoff and S. Phillips. *The prize collecting Steiner tree problem: theory and practice*, 11th ACM-SIAM Symposium on Discrete Algorithms, 760-769, 2000.

[17] S. Kapoor, S. N. Maheshwari, and J. S. B. Mitchell. An efficient algorithm for euclidean shortest paths among polygonal obstacles in the plane. *Discrete Comput. Geometry*, 18:377–383, 1997.

[18] J. Kim and J. P. Hespanha. Discrete approximations to continuous shortest-path: Application to minimum-risk path planning for groups of UAVs. In *Proc. of the 42nd Conf. on Decision and Contr.*, December 2003.

[19] B. O. Koopman. *Search and Screening*, Operations Evaluations Group Report N0. 56, Center for Naval Analyses, Alesandria, VA, 1946.

[20] B. O. Koopman. *Search and Screening: General Principles and Historical Applications*, Pergamon Press, New York, 1980.

[21] U. Lössner and I. Wegener. *Discrete sequential search with positive switch cost*, Mathematics of Operations Research, 7 (3), pp. 426-440, 1982.

[22] M. Mangel. *Search Theory: A Differential Equations Approach*, in Chudnovksy and Chudnovksy [3], pp. 55-101.

[23] R. Phillips and P. Kokotović. A singular perturbation approach to modeling and control of markov chains. *IEEE Trans. on Automat. Contr.*, 26(5):1087–1094, October 1981.

[24] J.-R. Sack and J. Urrutia, editors. *Handbook of Computational Geometry.* Elsevier, Amsterdam, 2000.

[25] T. J. Stewart. *Search for a Moving Target when Searcher motion is Restricted*, Computers and Operations Research, 6, pp. 129-140, 1979.

[26] T. J. Stewart. *Experience with a Branch and Bound Algorithm for Constrained Searcher Motion*, in Haley and Stone [14], pp. 247-253.

[27] L. D. Stone. *Theory of Optimal Search*, Academic Press, New York, 1975.

[28] L. D. Stone. *The Process of Search Planning: Current Approaches and Continuing Problems*, Operation Research, 31, 207-233, 1983.

[29] S. Thrun. *Robotic Mapping: A Survey*, in Exploring Artificial Intelligence in the New Millennium, G. Lakemeyer and B. Nebel (eds.), Morgan Kaufmann, 2002.

[30] S. Thrun, W. Burgard and D. Fox. *A probabilistic approach to concurrent mapping and localization for mobile robots*, Machine Learning and Autonomous Robots (joint issue), 31 (5), 1-25, 1998.

[31] K. E. Trummel and J. R. Weisinger. *The Complexiy of the Optimal Searcher Path Problem*, Operations Research, 34 (2), pp. 324-327, April 1986.