RAIRO-Oper. Res. 60 (2026) 2189–2226
https://doi.org/10.1051/ro/2026063

RAIRO Operations Research
www.rairo-ro.org

NAVIGATION ALGORITHMS FOR OPTIMAL PATHFINDING IN THE STOCHASTIC OBSTACLE SCENE PROBLEM

Elvan Ceyhan[1,*], Polat Charyyev[2] and Li Zhou[1]

Abstract. In the stochastic obstacle scene (SOS) problem (a continuous counterpart of the Canadian traveler's problem), a navigating agent seeks a low-cost route from a source to a target in the presence of obstacles whose true/false status is initially uncertain. We study the discretized SOS setting and develop a family of penalty-based navigation heuristics that incorporate both sensor-derived obstacle probabilities and a disambiguation cost incurred upon querying an obstacle at its boundary. Our main methodological contribution is a parametric unification of classical penalty rules via the ACS(k) family, together with norm-based approximations that relate ACS(k) to existing penalties. We evaluate the resulting heuristics through extensive Monte Carlo experiments across sensor-accuracy regimes, disambiguation costs, and obstacle densities, and we summarize which penalty-growth behaviors are preferable under each regime. The results provide practical guidance for selecting a penalty-based navigation rule for a given sensor and operational cost configuration.

Mathematics Subject Classification. 90C27, 90C59, 90C15, 05C85, 60G55.

Received July 18, 2024. Accepted May 27, 2026.

1. Introduction

Stochastic obstacle scene (*SOS*) *problem*, introduced by Papadimitriou and Yannakakis [19], is one of the most challenging stochastic optimization problems in literature, and its graph-theoretic (*i.e.*, discrete) version is called the *Canadian traveler's problem* (CTP). SOS problem pertains to finding an optimal path (*e.g.*, a path with minimum cost) of a navigating agent (NAVA) from a start point to a target point. Recently, both continuous and graph-theoretic versions of the problem have been studied and several efficient heuristics have been proposed to tackle them (see, *e.g.*, [5, 8, 15, 17]).

For the discretized SOS problem, the BAO* algorithm of Aksakalli [1], the simulated risk (SR) disambiguation algorithm of Fishkind *et al.* [15], the reset disambiguation (RD) algorithm of Aksakalli *et al.* [5], and the distance to termination (DT) algorithm of Aksakalli and Ari [3] are some of the well performing algorithms. The cost incurred from disambiguation of each obstacle is added to the overall traversal burden. To avoid ambiguity, we reserve the term *length* for Euclidean path or walk length, *cost* for realized traversal length plus disambiguation costs actually incurred, and *risk* for the planned weighted objective used by the navigation rule when selecting

Keywords. Network traversal and blocking, Canadian traveler's problem, penalty function, heuristic algorithm, obstacle disambiguation.

[1] Department of Mathematics and Statistics, Auburn University, Auburn, AL 36849, USA.
[2] MAP Akademi, Levent Çarşı, Beşiktaş, Istanbul, Turkey.
*Corresponding author: ceyhan@auburn.edu

a route. In the SOS context, the RD algorithm is proven to attain the optimal expected length, only in the case of two-node graphs with several edges, or disjoint paths where the beginning and end points are the same. However, showing the optimality of the navigation algorithms in the SOS problem for the broader setting of multiple nodes and edges is still an open problem. For other related work in this area, see Ye and Priebe [26]; Ye *et al.* [27]; Aksakalli and Ceyhan [4]; Alkaya *et al.* [7].

Several variants of the CTP problem have been introduced and effective heuristic algorithms are proposed. For example, in Bar-Noy and Schieber [8] the *recoverable CTP* was introduced where a blocked edge does not remain blocked forever, and each vertex v has a *recovery* time $t(v) \in [0, \infty)$ for edges adjacent to v. They also introduced the k-CTP, in which an upper bound of k blocked edges is given. Nikolova and Karger [17] studied another variant of the CTP where the cost of edges is assumed to follow independent and identical distributions (iid). Bnaya *et al.* [9] introduced the sensing cost in CTP which is closely related to our work. In the original CTP, the status of an edge (*i.e.*, whether it is blocked or not) can only be revealed upon reaching a vertex adjacent to that edge. This kind of sensing is called *local* sensing. On the other hand, *remote* sensing refers to revealing the status of an edge from a distant location [10]. The sensing cost in the CTP is analogous to the disambiguation cost in the discretized SOS problem. Bnaya *et al.* [11] studied the repeated-task CTP where a number of NAVAs need to travel with the same goal, minimizing their combined traversal cost. Recently, Aksakalli *et al.* [6] developed an AO^* based exact algorithm, called the CAO^* algorithm, for the CTP and they showed that the empirical performance is better than other exact algorithms.

This paper provides a structured methodological extension of penalty-based navigation heuristics for the discretized SOS problem. We introduce a parametric penalty family and an accompanying framework for comparing penalty-growth behaviors. Our contributions are:

(i) **New penalty structures:** We propose additive penalty functions, including *AP* and the *ACS(k)* family, that extend classical RD- and DT-type penalties while preserving computational simplicity.
(ii) **Parametric unification via approximation:** We show that ACS(k) can approximate existing penalty functions over relevant probability ranges under L_1, L_2, and L_∞ norms, thereby placing classical heuristics within a continuous parametric spectrum indexed by k.
(iii) **Comparative empirical evaluation:** We conduct extensive Monte Carlo experiments across sensor-accuracy regimes, disambiguation costs, and obstacle densities to characterize how penalty-growth structure influences traversal cost and to provide guidance on selecting ACS(k) (and related heuristics) under different operating conditions.

The contribution is methodological and empirical as we study and compare heuristic navigation rules within the established SOS/CTP paradigm.

We introduce the SOS problem in Section 2, including both its continuous and discretized formulations. Section 3 then develops the general methodological framework for penalty-based navigation, clarifying the distinction between planned risk and realized traversal cost. Section 4 introduces the penalty structures studied in this paper, including the new ACS(k) family, and presents the approximation-based unification of ACS(k) with existing heuristics. Section 5 reports the empirical evaluation, including sensitivity analyses with respect to sensor accuracy, disambiguation cost, and obstacle density, as well as convergence and threshold-adjusted ACS(k) results. We conclude in Section 6 with a discussion of the main findings and directions for future work.

2. The SOS problem

2.1. Continuous SOS problem

The continuous SOS problem is formally defined as follows: Given a bounded obstacle field $\Omega \subset \mathbb{R}^2$, false obstacles (*e.g.* clutter such as rocks, debris, fake mines) are assumed to be located at points, $\mathcal{X}_F$, which is assumed to follow a spatial point process F_F. Then, an *obstacle placing agent* (OPA) places true obstacles at the points, $\mathcal{X}_T$, assumed to follow another spatial point process F_T, possibly different from F_F. Each obstacle is represented with a disk D_x centered at x with a fixed radius $r > 0$ and false obstacles are centered at the points in $\mathcal{X}_F$ and

true obstacles are centered at the points in $\mathcal{X}_T$. A *navigating agent* (NAVA) wishes to traverse from a given starting point, $s \in \Omega$, to a target point, $t \in \Omega$, and is equipped with a sensor that assigns random probabilities $p_{_F} : \mathcal{X}_F \to [0,1]$ and $p_{_T} : \mathcal{X}_T \to [0,1]$ prior to NAVA's traversal. When observing a realization of these processes, NAVA only sees $\mathcal{X} := \mathcal{X}_F \cup \mathcal{X}_T$ as obstacles, some of which may be true. Let $p(x)$ be the probability that $x \in \mathcal{X}_T$, *i.e.*, x is a true obstacle for all $x \in \mathcal{X}$. Then NAVA's detector assigns $p(x) = p_{_T}$ for $x \in \mathcal{X}_T$ and $p(x) = p_{_F}$ for $x \in \mathcal{X}_F$. For every x in $\mathcal{X}$, when the navigation route intersects the boundary of the disk, ∂D_x (*i.e.*, when NAVA runs into an obstacle), it can choose to disambiguate the true status of x, determining whether x is in $\mathcal{X}_T$ or not. This decision incurs an additional cost $c > 0$ for disambiguation to the route's length. NAVA is allowed to traverse through disks that have been disambiguated to be false, but it must circumvent disks that remain ambiguous or those clarified as true obstacles. The NAVA seeks to traverse a continuous (s,t) route without hitting any true obstacle at shortest achievable arc length (*i.e.*, in shortest traversal length). The influence of terrain is ignored, as it is assumed to be homogeneous, and the NAVA's traversal speed is assumed constant in this setting. We further suppose that NAVA has a dynamic disambiguation capability. How the NAVA should route the continuous (s,t) traversal curve – and where and when the disambiguations should be performed – in order to minimize the expected length (including the disambiguation cost) of this curve is called the *continuous SOS problem* [19]. Conversely, the challenge of strategically positioning obstacles to lengthen the NAVA's travel distance (or cost) in the SOS scenario, called the *optimal obstacle placement (OOP) with disambiguations* problem, was proposed by Aksakalli and Ceyhan [4].

2.2. Discretized SOS problem

To enhance computational efficiency, we utilize a discrete approximation of the continuous SOS setting by using the 8-adjacency integer lattice (possibly after scaling the traversal region, which is also assumed to be rectangular), as per the methodology outlined in Aksakalli *et al.* [5]. This discretization is represented as a graph G, with vertices corresponding to integer pairs i, j satisfying $i = 1, 2, \ldots, i_{\max}$ and $j = 1, 2, \ldots, j_{\max}$, where $i_{\max}$ and $j_{\max}$ are prespecified integers. The navigation field Ω is partitioned by an $i_{\max} \times j_{\max}$ grid of unit squares, whose vertices form the graph's vertices. The edges in G reflect the grid's vertex adjacency, including edges connecting adjacent vertices horizontally and vertically with unit length, and diagonally with length $\sqrt{2}$, for all $i, j = 0, 1, \ldots, 99$. Additional unit-length corner edges connect vertices at $(100, j)$ and $(100, j+1)$, and $(i, 100)$ and $(i+1, 100)$.

In this model, one vertex on one edge of Ω is designated as the starting point s, and another vertex at the opposite edge of Ω as the target t. The NAVA's objective is to navigate from s to t in G, using edges that avoid intersecting any true or ambiguous obstacles. Disambiguation of any intersected ambiguous obstacle can occur at the edge endpoints outside the obstacle. The aim is to develop an algorithm for NAVA that minimizes the expected traversal length (or cost), leveraging disambiguation abilities and obstacle location information. This setup, called the *discretized SOS problem (D-SOSP)*, represents a specific instance of the CTP.

2.3. SOS problem formulation

A *walk* in a graph $G = (V, E)$ is a finite sequence $\omega = v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k$ such that: $v_i \in V$ for all $i = 0, 2, \ldots, k$, and $e_i = \{v_{i-1}, v_i\}$ (in an undirected graph) or $e_i = (v_{i-1}, v_i)$ (in a directed graph) for all $i = 1, 2, \ldots, k$. Here, v_0 is the *start vertex* and v_k is the *end vertex*, and k is the *length* of the walk, indicating the number of edges in the sequence. Notice that walks allow for the repetition of vertices and edges, *i.e.*, it is possible for $v_i = v_j$ for some $i \neq j$ and $e_i = e_j$ for some $i \neq j$. On the other hand, a walk where all edges are distinct is called a *path* [22]. Let $V(\omega)$ be the set of vertices v_i and $E(\omega)$ be the set of edges e_i in walk ω, respectively.

In the discretized SOS problem, NAVA is restricted to move along the edges of the spatial grid G, with the possibility of revisiting an edge due to edge costs or an edge intersecting a true obstacle. Consequently, the traversal from $v_0 = s$ to $v_k = t$ on the graph is more accurately described as a *walk* rather than a *path*, in accordance with graph theory terminology. NAVA's trajectory in practice is a continuous route, and the set of

edges traversed by NAVA approximates this route. Note also that a cycle (*i.e.*, a closed walk with at least one edge and no repeated vertices or edges other than the starting and ending vertex) is also possible in our setting.

Let $\Omega \subset \mathbb{R}^2$ denote the traversal region and let $W \subset \Omega$ be the obstacle placement window. Let $\mathcal{X} = \mathcal{X}_T \cup \mathcal{X}_F$ denote the (random) set of obstacle centers, where $\mathcal{X}_T$ and $\mathcal{X}_F$ represent the sets of true and false obstacles, respectively, and $|\mathcal{X}| = n$ is fixed.

For a given realization of $\mathcal{X}$, let $\omega = \omega(s, t; \mathcal{X})$ denote the walk generated by a specified navigation algorithm from s to t, and let $C(\omega, \mathcal{X})$ denote the corresponding traversal cost (including disambiguation costs). The objective of the navigation agent (NAVA) is to choose a navigation policy (equivalently, an induced walk mapping $\omega(\cdot)$) that minimizes the expected traversal cost:

$$\min_{\omega(\cdot)} \mathbb{E}\big[C\big(\omega(s,t;\mathcal{X}),\mathcal{X}\big)\big] \text{ subject to } \mathcal{X} \subset W,\ E\big(\omega(s,t;\mathcal{X})\big) \cap \mathcal{X}_T = \emptyset,\ |\mathcal{X}| = n, \tag{1}$$

where the expectation is taken with respect to the randomness in obstacle placement and sensor-generated probability marks. Note that the constraint $W \subset \Omega$ ensures that all obstacles are placed within the designated window (i.e. $\mathcal{X} \cap W^c = \emptyset$) and while $E\big(\omega(s,t;\mathcal{X})\big) \cap \mathcal{X}_T = \emptyset$ enforces that the realized traversal avoids all true obstacles. The condition $|\mathcal{X}| = n$ fixes the total number of obstacles.

Equation (1) may therefore be interpreted as the optimization problem of minimizing the expected traversal cost with respect to the obstacle configuration, subject to feasibility constraints. In contrast, the navigation algorithms studied in this article aim to approximate the inner minimization over admissible (s, t) walks for a given realization of X.

We emphasize that all navigation strategies studied in this paper are heuristic in nature. Except in special graph structures (*e.g.*, disjoint parallel paths), global optimality guarantees for the SOS/CTP problem remain open problems. Our goal is not to resolve this long-standing theoretical challenge, but rather to understand how different penalty growth mechanisms affect practical traversal performance under uncertainty. The proposed methods should therefore be viewed as structured heuristic improvements within the established paradigm.

3. Methodological framework: penalty-based navigation

This section establishes the general navigation framework within which all penalty functions considered in this paper operate. Specific penalty structures are introduced in Section 4.

In the discretized SOS problem, the graph G is defined as in Section 2.2, with its vertices in V and edges in E. Consider $D = D_x$, a disk centered at x in $\mathcal{X} = \mathcal{X}_F \cup \mathcal{X}_T$, with a specified radius $r > 0$ and $|\mathcal{X}| = n$. The objective is to determine, at each planning stage, a minimum-risk planned path from the current vertex to t under the chosen edge-weighting rule, while the realized end-to-end traversal may be an executed walk whose cost equals the Euclidean length actually traversed plus any disambiguation costs incurred during execution.

A navigation algorithm initially assigns a weight function $w(e)$ to each edge e, which is additive in the Euclidean length $\ell(e)$ of e, and a penalty function $\mathcal{P}(c, p_i)$ depending on disambiguation cost c and probability of an obstacle i being true p_i if the edge intersects the obstacle.

Remark 3.1 (Optimal path *versus* executed walk)**.** As noted in Section 2.3, the realized trajectory of NAVA on the grid may be a walk rather than a path, because replanning after disambiguation can cause previously traversed vertices or edges to be revisited. The shortest-path routines used within the navigation algorithms, however, are designed to compute an optimal path rather than an optimal walk. Indeed, for any walk between two vertices in a weighted graph, there exists a path between the same vertices with no greater total length. Thus, at each replanning stage the algorithm selects a minimum-risk path from the current vertex to the target, while the concatenation of these successive planned paths over the full execution may produce an observed walk ω_{obs}. In this sense, the algorithms compute tentative optimal paths locally, whereas the realized end-to-end traversal need not itself be a path.

3.1. Navigation algorithm and adapted A^* algorithm

We will employ the navigation algorithm which also relies on an adaptation of the commonly-used A^* algorithm for finding the optimal path on G. Both algorithms are adapted for our setting from Ye and Priebe [26] where they are stated for navigation on directed graphs.

3.1.1. Navigation algorithm

The navigation algorithm repeatedly computes a shortest (v, t) route under the current edge weights, advances along it, and performs disambiguations only when required by the planned route.

In Algorithm 1, p_i denotes the initial probability mark that obstacle D_i is true, and p_i^+ denotes the updated value after disambiguation (so $p_i^+ \in \{0, 1\}$). The variable v is the current vertex (*i.e.*, NAVA's current location). For an edge $e = uv$, we write $I_e := \{\, i : e \text{ intersects } D_i \,\}$ for the set of obstacle indices whose disks intersect e, and we write $I_d := \{\, i : D_i \text{ has not been disambiguated} \,\}$ for the set of indices of obstacles that remain ambiguous. The weight function w is the current edge-weight function induced by the chosen penalty rule and the current collection of probability marks $\{p_i\}$.

Algorithm structure.

Initialization. The algorithm assigns probability marks $\{p_i\}$ to obstacles, constructs the corresponding edge weights w, and sets the current vertex to $v = s$.
Path finding and movement. Given the current vertex v, the algorithm computes a shortest path from v to t under the current weights w (using the shortest-path routine adopted in our implementation) and begins to traverse this path. It continues until either reaching the target t or encountering an edge $e = vv'$ whose traversal would intersect at least one ambiguous obstacle, *i.e.*, $I_e \cap I_d \neq \emptyset$.
Disambiguation and update. When such an edge is encountered, NAVA disambiguates all obstacles D_i with indices $i \in I_e \cap I_d$ at the current vertex v, updates their marks to p_i^+, updates the edge weights w accordingly, and then repeats the path-finding step from the updated state.

Algorithm 1. Navigation algorithm for the SOS problem.

Input: Graph $G = (V, E)$, starting vertex s, target vertex t, initial probabilities p_i for obstacles $i \in \{1, 2, \ldots, m\}$.
Output: Shortest path from s to t, adjusted for obstacle probabilities.

```
Initialize G, assign p_i for obstacles, set current vertex v = s.
while v ≠ t do
    Use A* for the shortest path v to t in G, with edge weights w.
    if edge e = vv' intersects disk D_i with p_i > 0 then
        Disambiguate at v for obstacles D_i intersecting e where i ∈ I_e ∩ I_d.
        Update p_i to p_i^+ for all obstacles according to the outcome of the disambiguation.
        Update the weights w in G based on new p_i.
    else
        Move along edge e to v' and set current vertex v = v'.
    end if
end while
```

3.1.2. Adapted A^* algorithm

We employ a standard A^* shortest-path procedure [14, 21] with dynamically updated edge weights reflecting the probabilistic penalty structure. We omit standard details of A^* and focus only on the modifications induced by the uncertainty-adapted weight function $w(\cdot)$ used in our navigation framework. Algorithm 2 summarizes the resulting implementation.

Algorithm 2. A^* Pathfinding algorithm with uncertainty adaptation.

Input: Graph $G = (V, E)$, heuristic function h, starting vertex s, target vertex t, probabilistic weight function w.
Output: Minimum cost path from s to t.

Initialize $V_{\text{pend}} = \{s\}$, $d(s) = 0$, $V_{\text{exp}} = \emptyset$.
while $V_{\text{pend}} \neq \emptyset$ and $t \notin V_{\text{exp}}$ **do**
  Choose u from V_{pend} as $u = \arg\min_{v \in V_{\text{pend}}}(d(v) + h(v, t))$.
  Move u from V_{pend} to V_{exp}.
  if $u = t$ **then**
    return path from s to t.
  end if
  for each v adjacent to u not in V_{exp} **do**
    Calculate $d_{\text{tent}} = d(u) + w(uv)$.
    if $v \notin V_{\text{pend}}$ or $d(v) > d_{\text{tent}}$ **then**
      Update $d(v) = d_{\text{tent}}$ and set $pred(v) = u$.
      If $v \notin V_{\text{pend}}$, then $V_{\text{pend}} = V_{\text{pend}} \cup \{v\}$.
    end if
  end for
end while

3.2. NAVA's estimated (weighted) cost and traversal length

We first note that the actual traversal length incurred for NAVA and estimated risk at each reset stage of the navigation algorithm are different random variables. We provide some stochastic ordering and limiting results for the navigation algorithms and their estimated risks and total traversal lengths.

Recall that NAVA's objective is to navigate from a start point s to a target point t along an (s, t) route within the discretized grid, equivalently, via an (s, t) walk in graph G. Each edge in $E(G)$ bears a weight corresponding to its Euclidean length and the disambiguation cost arising from any intersecting obstacle disk. The starting point s is chosen to be located at the center of one boundary edge of the (rectangular) domain Ω, and the target t is chosen to be at the midpoint of the opposite boundary edge of Ω.

Although it is possible for NAVA's route to be a walk on G, the navigation algorithms work with paths to find the optimal route (see Rem. 3.1). Suppose that all (s, t)-paths have M distinct Euclidean lengths (excluding loops), with the ith length occurring N_i times. An (s, t)-path can then be denoted by $\pi_{ij} = \pi_{ij}(s, t)$ for $j = 1, 2, \ldots, N_i$ and $i = 1, 2, \ldots, M$, yielding a total of

$$K = \sum_{i=1}^{M} N_i$$

distinct (s, t)-paths on the standard 8-adjacency integer grid. Let

$$L_{ij} = L(\pi_{ij})$$

represent the Euclidean length of path π_{ij}, and assume, without loss of generality, that

$$L_{1j} = m < L_{2j} < \cdots < L_{Mj}.$$

Note that for any fixed i, all paths in the ith length class have the same Euclidean length, that is,

$$L(\pi_{ij}) = L(\pi_{ij'}) \qquad \text{for } j, j' = 1, 2, \ldots, N_i.$$

Additionally, we assume that $r > 1/2$, guaranteeing that an obstacle intersects grid edges with probability 1. The disambiguation cost $c(x) = c > 0$ is assumed to be fixed for all obstacles, and the distribution of the sensor probability, $p(x)$, is assumed to be continuous.

Throughout the paper we distinguish three related objects.

Edge weight. For each grid edge $e \in E(G)$, the navigation rule assigns a weight $w(e)$ equal to the Euclidean edge length plus a penalty term whenever e intersects an (as-yet ambiguous) obstacle; the penalty depends on the disambiguation cost c and the sensor mark $p(\cdot)$ through the chosen penalty function $P(c,p)$.
Planned/estimated risk. For any candidate (s,t)-path π, the planned (pre-traversal) risk is $R(\pi) := \sum_{e\in E(\pi)} w(e)$. The algorithm selects the minimum-risk path $\pi^* = \arg\min_\pi R(\pi)$ at each replanning step.
Realized traversal cost. The realized cost incurred by the executed walk ω_{obs} is $C(\omega_{\text{obs}}, \mathcal{X}) = L(\omega_{\text{obs}}) + c\, n_{\text{dis}}$, *i.e.*, Euclidean length actually traversed plus the disambiguation costs actually paid. In general, $C(\omega_{\text{obs}}, \mathcal{X})$ and $R(\pi)$ are different random variables.

We now formalize these definitions and relate the planned risk $R(\pi)$ to the realized cost $C(\omega_{\text{obs}}, \mathcal{X})$ under the navigation-and-disambiguation procedure.

The navigation algorithms we consider in this article incorporate the disambiguation cost c and the probability mark $p(x)$ when an edge intersects an obstacle x, and assign the weight $w(e) := w(e, c, p, \mathcal{X})$ to any edge $e \in E(\pi(s,t))$ as

$$w(e) = \ell(e) + \frac{1}{2}\sum_{x\in\mathcal{X}} \mathbf{1}\{D_r(x) \cap e \neq \emptyset\}\, \mathcal{P}(c, p(x)), \tag{2}$$

where $\ell(e)$ is the Euclidean length of edge e (equal to 1 or $\sqrt{2}$), $\mathbf{1}\{\cdot\}$ is the indicator function, and $\mathcal{P}(c,p)$ is the penalty function used by NAVA to incorporate the disambiguation cost and the probabilistic obstacle mark. For example, under the RD algorithm,

$$\mathcal{P}(c,p) = \mathcal{P}_{\text{RD}}(c,p) = \frac{c}{1-p}\cdot$$

Thus, when $p(x) = 0$, the edge weight is $\ell(e) + c/2$, whereas as $p(x) \to 1$ the penalty diverges. Furthermore, for $p(x) \in \{0,1\}$, NAVA has exact information about the obstacle status, so disambiguation is unnecessary. Adding these weights for the edges of the path $\pi(s,t)$ yields the total cost assigned to path $\pi(s,t)$ by NAVA

$$\mathcal{R}(\pi) := \mathcal{R}(\pi, \mathcal{X}, \mathcal{P}) = \sum_{e\in E(\pi(s,t))} w(e). \tag{3}$$

The quantity $\mathcal{R}(\pi)$ is the planned (pre-traversal) risk assigned by NAVA to path π, and is the objective minimized at each replanning step. Equivalently,

$$\mathcal{R}(\pi) = \underbrace{\sum_{e\in E(\pi)} \ell(e)}_{\mathcal{L}(\pi)} + \sum_{\substack{x\in\mathcal{X}, e\in E(\pi):\\ D_r(x)\cap e\neq\emptyset}} \mathcal{P}(c, p(x)). \tag{4}$$

For a candidate path π_{ij}, we write $\mathcal{R}(\pi_{ij})$ for its estimated risk under the current sensor information, and define

$$\pi^* := \arg\min_{\pi_{ij}} \mathcal{R}(\pi_{ij}), \qquad \mathcal{R}^* := \mathcal{R}(\pi^*).$$

Thus, π^* is the minimum-risk path planned by NAVA under the current edge weights.

When there is no obstacle, *i.e.*, $n = 0$, the quantity $\mathcal{R}(\pi)$ is not a random quantity. Because, $\mathcal{R}(\pi) = \mathcal{L}(\pi)$ holds true in this case, since $w(e) = \ell(e)$ for every edge e in graph G. Hence, NAVA opts for the path with the minimum Euclidean length, $\arg\min_{\pi(s,t)} \mathcal{R}(\pi)$, typically a direct linear route connecting s and t. On the other hand, the presence of obstacles, envisioned as disks with non-zero radii, renders $\mathcal{R}(\pi)$ a random variable. This randomness stems from the stochastic distribution of obstacles $\mathcal{X}$ and the probabilistic marks of these obstacles by NAVA's sensor, typically modeled using a Beta distribution. Under these circumstances, NAVA, positioned at a vertex $v \in V(G)$ in the current step selects the path $\pi(v,t)$ with the lowest risk $\mathcal{R}(v,t)$, where $\mathcal{R}(v,t)$

represents the risk of traversing from vertex v to t. The navigation algorithm aids in identifying the path(s) of minimum risk, employing Algorithm 1 for weight consideration in graph G.

Let ω_{obs} be the observed walk traversed by NAVA from s to t, and let $\mathcal{C}(\omega_{\text{obs}}, \mathcal{X})$ be the realized traversal cost along ω_{obs}. Then

$$\mathcal{C}(\omega_{\text{obs}}, \mathcal{X}) = \sum_{e \in E(\omega_{\text{obs}})} \ell(e) + c \times n_{\text{dis}} = \mathcal{L}(\omega_{\text{obs}}) + c \times n_{\text{dis}}. \tag{5}$$

Here n_{dis} denotes the number of disambiguations performed during execution. In general, the realized traversal cost $\mathcal{C}(\omega_{\text{obs}}, \mathcal{X})$ and the planned risk $\mathcal{R}(\pi)$ are different random variables: $\mathcal{C}(\omega_{\text{obs}}, \mathcal{X})$ depends on the actual executed walk and the disambiguation outcomes, whereas $\mathcal{R}(\pi)$ is the pre-traversal weighted objective used to select a candidate path.

In settings with no obstacles, by design, we have $\mathcal{C}(\omega_{\text{obs}}, \mathcal{X} = \emptyset) = \min_{i,j} \mathcal{L}_{ij} = \min_{\pi_{ij}} \mathcal{L}(\pi_{ij})$ with ω_{obs} being the minimum length path from s to t, *i.e.*, $\omega_{\text{obs}} = \arg\min_{\pi_{ij}} \mathcal{L}(\pi_{ij}) = \pi^*$. When encountering exclusively false obstacles, *i.e.*, $n = n_F > 0$, NAVA opts for the path with the minimum estimated risk $\mathcal{R}^*$ (*i.e.*, the minimum cost walk is the same as the minimum risk path). Specifically, $\omega_{\text{obs}} = \pi^*$ and so

$$\mathcal{C}_F := \mathcal{C}(\pi^*, \mathcal{X}_F) = \sum_{e \in E(\pi^*)} w(e, 0, c, \mathcal{X}_F) = \ell^*_{ij} + c \times n^*_{\text{dis}} \tag{6}$$

where $n^*_{\text{dis}} = \sum_{x \in \mathcal{X}_F} \mathbf{1}_{D_r(x) \cap \pi^* \neq \emptyset}$ is the number of disambiguations performed by NAVA along π^* and ℓ^*_{ij} is the total (Euclidean) length of the edges in $E(\pi^*)$. However, the actual traversal cost is different from $\mathcal{R}^*$ as $\mathcal{C}_F \leq \mathcal{R}^*$ holds true with probability 1, provided that $\mathcal{P}(p, c) > c$ for $0 < p < 1$ (which happens, *e.g.*, for the $\mathcal{P}_{\text{RD}}$).

3.3. Decision-theoretic view of realized traversal cost

The navigation procedure in Algorithm 1 induces a sequence of replanning steps and local disambiguation decisions, resulting in an executed (observed) walk ω_{obs} from s to t. The realized traversal cost is

$$\mathcal{C}(\omega_{\text{obs}}, \mathcal{X}) = \mathcal{L}(\omega_{\text{obs}}) + c\, n_{\text{dis}}, \tag{7}$$

i.e., the total Euclidean length actually traversed plus c times the number of disambiguations performed. This realized cost differs from the planned (pre-traversal) risk $R(\pi)$ used for shortest-path selection in Section 3.2, because replanning is triggered by disambiguation outcomes and obstacle realizations.

For completeness, Appendix A provides a detailed decision-theoretic/action-sequence representation of ω_{obs} and $\mathcal{C}(\omega_{\text{obs}}, \mathcal{X})$ that makes the sequential structure explicit.

4. Penalty structures and parametric unification

Building on the general navigation framework introduced in Section 3, we now compare classical penalty functions and introduce new additive structures. We then show how the ACS(k) family provides a parametric unification of these heuristics.

4.1. Existing penalty functions for navigation algorithms

We first describe reset disambiguation, simulated risk, and distance to termination algorithms. In the *reset disambiguation* (RD) protocol of Aksakalli *et al.* [5], the weight function w_{RD} is $w(e)$, with the penalty function being

$$\mathcal{P}_{\text{RD}}(c, p) = \frac{c}{1 - p}.$$

NAVA identifies the shortest s, t path in G using these weights, via the navigation algorithm (see Algorithm 1) with obstacle D_i is disambiguated to be true, then p_i is set to 1, the weights are updated, and consequently

all edges intersecting D_i are removed (having infinite cost, they have no chance of being selected by the A^* algorithm). Basic limiting behavior and monotonicity properties are summarized in Table 1.

In another SOS navigation algorithm called the *simulated risk* (SR) algorithm [15], the penalty function is defined as:

$$\mathcal{P}_{\mathrm{SR}}(\alpha, p) = -\alpha \log(1-p),$$

where α represents a constant tuning parameter. A limitation of the SR algorithm is the necessity to adjust the tuning parameter α for optimizing performance and another is that it does not incorporate the disambiguation cost c (unless α is taken to be a function of c). The work by Aksakalli *et al.* [5], through a series of simulated experiments, demonstrates that algorithms employing the penalty function $\mathcal{P}_{\mathrm{RD}}$ outperform those using $\mathcal{P}_{\mathrm{SR}}$ for most α values. Its main structural properties are therefore better understood through the limiting and monotonicity behavior summarized in Table 1.

The *distance to termination* (DT) algorithm, introduced by Aksakalli and Ari [3], has demonstrated superior performance over previous algorithms in various scenarios. This algorithm is based on the principle that an effective penalty function should monotonically increase with both the disambiguation cost c and the probability p of an obstacle being true [27]. In line with this, Aksakalli and Ari [3] tested numerous penalty functions with additive cost terms that adhere to this principle. Their exploration resulted in the formulation of the following penalty function for the DT algorithm:

$$\mathcal{P}_{\mathrm{DT}}(c, p, d_t) = c + \left(\frac{1-p}{d_t}\right)^{\log(1-p)},$$

where d_t denotes the Euclidean distance from the center of disk D to the target point t. For the DT penalty, we write

$$d_t(x) := \|x - t\|$$

for the Euclidean distance from the center of obstacle x to the target t. When needed for comparison, we also write

$$d_s(x) := \|x - s\|$$

for the corresponding distance from the obstacle center to the source s. When c and p are held constant, as NAVA gets closer to the target t, DT algorithm tends to disambiguate more obstacles, otherwise prefer not to disambiguate (*i.e.*, becomes more risk-averse). A distinctive feature of $\mathcal{P}_{\mathrm{DT}}(c, p, d_t)$ is its rapid exponential growth with increasing values of p, setting it apart from other algorithms. The DT algorithm tends to minimize risk, preferring to disambiguate as few obstacles as possible. This concept echoes the ideas in Bnaya *et al.* [9,10], relating to the CTP, where sensing corresponds to the disambiguation option in the discretized SOS problem. They explored various sensing costs including zero cost, infinite cost, constant cost, and a cost proportional to the distance from the current position to the starting point s.

In the traditional CTP, the zero-disambiguation path from s to t is often deemed too costly (like opting for a helicopter over driving), leading the NAVA to avoid this zero-disambiguation path unless all other paths are blocked. However, in the discretized SOS context, given the study window of $[0, 100] \times [0, 100]$ and obstacle distribution within $[10, 90] \times [10, 90]$, it is assumed that there's always a navigable path around the obstacles from s to t [4]. The DT algorithm, due to the characteristics of its penalty function, is inclined towards avoiding risks, often forgoing the opportunity to disambiguate obstacles, and defaulting to the zero-disambiguation path as a viable and often chosen option.

Table 1 summarizes the basic limiting behavior and monotonicity of the penalty functions used in this paper.

4.2. New penalty functions for the navigation algorithms

We propose new penalty functions noting a limitation in the RD algorithm. When the disambiguation cost is negligible or zero, its effectiveness diminishes. This is because the computation of $w_{\mathrm{RD}}(c, p)$ neglects the impact of the probabilistic information when disambiguation cost is close to zero (as disambiguation cost is multiplicative

TABLE 1. Limiting behavior and monotonicity of penalty functions. Here $c > 0$ is the disambiguation cost, $p \in (0,1)$ is the sensor probability mark, $dt > 0$ is distance-to-target (DT), and $\alpha > 0$, $k \geq 0$ are tuning parameters (SR and ACS(k), respectively). In the limit columns the subscript of the penalty term $P_\cdot$ correspond to the row labels, *e.g.*, $\lim_{p\to 0} P_{\mathrm{RD}} = c$.

Penalty	$\lim_{p\to 0} P_\cdot$	$\lim_{p\to 1} P_\cdot$	$\lim_{c\to 0} P_\cdot$	Monotonicity
RD	c	∞	0	increasing in c and p
SR	0	∞	n/a	increasing in α and p
DT	$c+1$	∞	$\left(\frac{1-p}{d_0}\right)^{\log(1-p)}$	increasing in c and p; decreases as $d_0 \downarrow$
AP	c	∞	$\frac{5p}{1-p^{1-p}}$	increasing in c and p
ACS(k)	$c+1$	∞ ($k>0$)	$(1-p)^{-k}$	increasing in c and p (for $k>0$), and in k

in the penalty function). Thus, with RD algorithm, NAVA will disambiguate any obstacle regardless of the sensor information leading to suboptimal outcomes, potentially trapping NAVA in localized problem areas. This issue is highlighted in the findings of Aksakalli and Ceyhan [4], where they observed an increased mean traversal length of NAVA in environments with uniformly distributed true obstacles within a V-shaped sub-region, amidst uniformly distributed false obstacles in the traversal region. The mean traversal cost was considerably higher compared to other obstacle configurations. To mitigate this, we suggest the disambiguation cost be an additive component in the penalty function. Along this line, we introduce the following penalty function:

$$\mathcal{P}_{\mathrm{AP}}(c,p) = c + \frac{5p}{1-p^{1-p}}.$$

We refer to the corresponding navigation rule as the AP algorithm, reflecting its additive penalty in disambiguation cost and sensor probability. Even when the disambiguation cost is 0, AP algorithm puts a positive risk to an obstacle (which is larger for obstacles with larger probability marks). Thus, AP has a similar shortcoming (of being extremely risk-averse) as the DT algorithm above. From our simulation studies (to be elaborated in the subsequent sections), we see that the AP algorithm outperforms the RD algorithm in terms of achieving a shorter mean traversal length under various settings.

We visualize the growth of the classical penalty functions $\mathcal{P}_{\mathrm{RD}}$, $\mathcal{P}_{\mathrm{AP}}$, and $\mathcal{P}_{\mathrm{DT}}$ as functions of the probability mark p and the disambiguation cost c in Figure 1. Panel (a) plots the penalties *versus* p (with other parameters fixed): all three functions are increasing in p and diverge as $p \to 1$, reflecting increasing aversion to edges that intersect obstacles that are likely to be true. The separation of the curves highlights distinct risk-growth profiles: $\mathcal{P}_{\mathrm{RD}}$ increases more moderately with p, whereas $\mathcal{P}_{\mathrm{AP}}$ and, especially, $\mathcal{P}_{\mathrm{DT}}$ increase more sharply, implying earlier switching to detours for comparable cost settings. Panels (b)–(d) plot the penalties *versus* the disambiguation cost c at fixed probability levels $p \in \{0.25, 0.50, 0.75\}$. For $p = 0.25$, the penalties increase approximately linearly with c and remain in a comparable numerical range over $c \in [0, 20]$, indicating that when blockage is relatively unlikely, differences among rules are driven mainly by how they weight disambiguation cost *versus* probability. At $p = 0.50$, the same monotone trend holds but the spread between curves increases, reflecting a stronger interaction between cost and uncertainty: for the same increase in c, more risk-averse penalties inflate the effective edge weight more and therefore favor detours earlier. At $p = 0.75$, the penalties become substantially larger (most notably for $\mathcal{P}_{\mathrm{DT}}$) and grow rapidly with c, indicating that when an obstacle is already quite likely to be true, even moderate disambiguation costs can make crossing-and-disambiguating unattractive and push the planner strongly toward avoidance routes.

For comparative analysis, we also consider a '*benchmark protocol*' where NAVA has prior knowledge of location and status of all obstacles. NAVA starts its journey assigning a probability of 1 to true obstacles

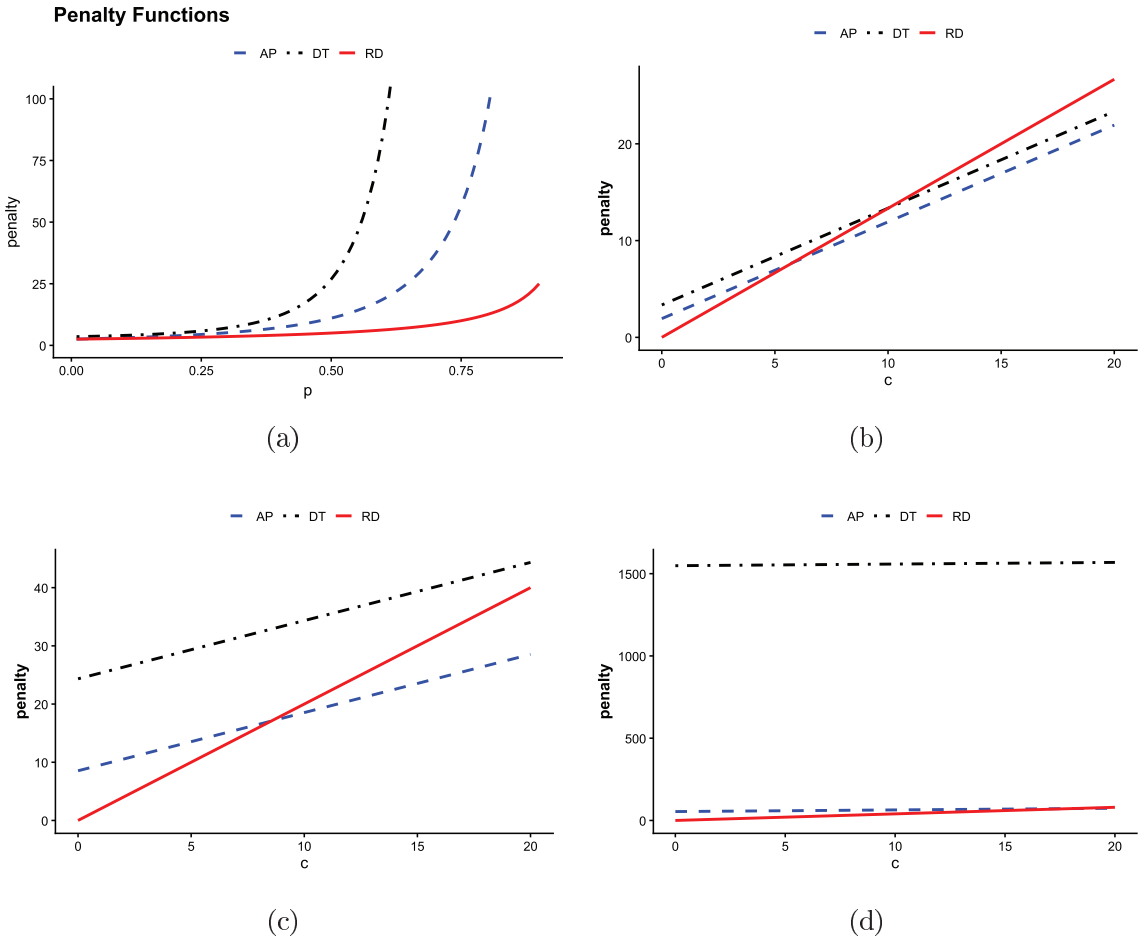


FIGURE 1. Comparison of the penalty functions $\mathcal{P}_{\mathrm{RD}}$, $\mathcal{P}_{\mathrm{AP}}$ and $\mathcal{P}_{\mathrm{DT}}$, with $c = 5$ and $d_0 = 50$. Panel (a): penalty *versus* p. Panels (b)–(d): penalty *versus* $c \in (0, 20)$ for fixed $p = 0.25, 0.50, 0.75$, respectively. For better visualization of $\mathcal{P}_{\mathrm{RD}}$ and $\mathcal{P}_{\mathrm{AP}}$ in panel (a), the y-axis limit is restricted to $(0, 100)$. Note also that y-axes are differently scaled.

and 0 to false ones, and utilizes the steps from the RD algorithm for calculating traversal length. In cases of false obstacles, NAVA crosses them, but still disambiguates them with disambiguation cost being added into the total traversal cost. With $p = 1$ for true obstacles, any edge intersecting a true obstacle will have infinite weight, so, NAVA will not choose a path containing such an edge, and will not be disambiguating any true obstacle. Thus, in the benchmark protocol, NAVA avoids paths that cross true obstacles. This benchmark algorithm offers the shortest possible traversal length attainable by any navigation heuristic approach as in Algorithm 1. Additionally, we define the *zero-disambiguation navigation algorithm*, denoted as ZD, where no disambiguation occurs in determining the shortest s, t path. This algorithm completely avoids the risk by not engaging in any disambiguation during the traversal.

4.2.1. A new penalty function in navigation algorithms

The RD and DT penalties differ primarily in whether disambiguation cost enters multiplicatively or additively. This observation motivates the introduction of a more flexible additive-parametric family. In the RD algorithm,

the penalty function is multiplicative in c and (a function of) p, causing extreme values of p significantly reducing the impact of the disambiguation cost c, and vice versa. This suggests the potential benefit of penalty functions that are additive in c and (some function of) p, *i.e.* $c + f(p)$, as in the AP and DT algorithms. Along this line, we propose a new penalty function that incorporates aspects of the previous frameworks and is additive in c and p as follows:

$$\mathcal{P}_{\mathrm{ACS}}(c, p, k) = c + (1 - p)^{-k},$$

where $k \geq 0$. This formulation makes the penalty function polynomial in $1/(1-p)$, making it flexible and facilitating its analysis and understanding. We refer to the corresponding navigation rule as the ACS(k) algorithm, reflecting its additive penalty in disambiguation cost and sensor probability. Even when the disambiguation cost is 0, ACS(k) algorithm puts a positive risk to an obstacle (which is larger for obstacles with larger probability marks), having a similar shortcoming as the DT and AP algorithms above.

Our goals are two-fold in introducing this penalty function: Firstly, we aim to incorporate this function within greedy navigation algorithms (as in Algorithm 1), determining the optimal k values for various scenarios (with varying sensor accuracy (*i.e.*, varying distributions for p) and disambiguation cost). Secondly, we want to identify a value for k for which $\mathcal{P}_{\mathrm{ACS}}$ can effectively approximate existing penalty functions over a range of probabilities $[0, p_{\max}]$, with appropriately chosen $p_{\max} < 1$ to avoid the asymptotic increase in penalty as p tends to 1. In approximating the penalty functions with $\mathcal{P}_{\mathrm{ACS}}(c, p, k)$, we employ L^1, L^2, and L^∞ norms. That is, we find

$$\arg\min_{k \geq 0} \left\| \mathcal{P}_{\mathrm{RD}}(c, p) - \mathcal{P}_{\mathrm{ACS}}(c, p, k) \right\|_q,$$

where the L^q norm is $\|f(x)\|_q = \left(\int |f(x)|^q \, \mathrm{d}x \right)^{1/q}$, for $q = 1, 2$ and the integration is taken over $p \in (0, p_{\max})$ and the L^∞ norm is $\|f(x)\|_\infty = \max |f(x)|$ (*i.e.*, the maximum absolute difference).

To complement the penalty-shape comparisons in Figure 1, Figure 2 visualizes the induced routing regimes in a simple toy environment. Across all penalties, the region in which the planned route crosses the obstacle contracts as either p or c increases, reflecting a shift toward more conservative routing when the obstacle is more likely to be true or when disambiguation becomes more costly. The location and shape of the decision boundary differ across penalties, highlighting their distinct risk-growth behavior. In particular, RD remains comparatively willing to cross for small c, whereas AP and DT switch to detouring at smaller p values once c increases. The ACS(k) panels further show that increasing k shifts the boundary systematically toward detouring, providing an interpretable bridge between the classical penalties and the proposed parametric family.

Figure 3 complements the decision maps by showing representative planned routes for selected parameter pairs. These examples make the threshold behavior visually concrete: depending on the penalty rule and the (c, p) combination, the preferred route either continues through the obstacle or switches to a detour around it. Taken together with Figure 2, these snapshots show how changes in disambiguation cost, obstacle probability, and the ACS(k) parameter translate into qualitatively different route choices.

We first provide a result on existence of an optimal k to approximate the RD, DT, and AP penalty functions with $\mathcal{P}_{\mathrm{ACS}}(c, p, k)$.

Proposition 4.1. *For any given values of $c > 0$ and $0 < p_{\max} < 1$, there exist an optimal value of $k > 0$ denoted as $k^*_{\mathrm{RD}}(c, q)$ such that $\mathcal{P}_{\mathrm{RD}}(c, p)$, can be optimally approximated (with respect to the L^q norm for any $0 < q \leq \infty$) by $\mathcal{P}_{\mathrm{ACS}}(c, p, k)$ with $k = k^*_{\mathrm{RD}}(c, q)$ over the interval $(0, p_{\max})$. That is, the L^q norm of difference between $\mathcal{P}_{\mathrm{ACS}}(c, p, k)$ and $\mathcal{P}_{\mathrm{RD}}(c, p)$ is minimized when $k = k^*_{\mathrm{RD}}(c, q)$. A similar result holds for the penalty functions $\mathcal{P}_{\mathrm{DT}}(c, p, d_t)$, and $\mathcal{P}_{\mathrm{AP}}(c, p)$ with optimal $k^*_{\mathrm{DT}}(c, q, d_t)$ and $k^*_{\mathrm{AP}}(c, q)$ values, respectively.*

Proof. Here the optimization is minimization of L^q norm of the difference $\mathcal{P}_{\mathrm{ACS}}(c, p, k) - \mathcal{P}_{\mathrm{RD}}(c, p)$ in k (which is the objective function) for given values of c, p and q. First we prove that the objective function is well-defined and continuous in k. For any (bounded) constants $c > 0$ and $k > 0$, the penalty functions $\mathcal{P}_{\mathrm{ACS}}(c, p, k)$ and $\mathcal{P}_{\mathrm{RD}}(c, p)$ are continuous and bounded in p for $p \in [0, p_{\max}]$. Consequently, the difference between $\mathcal{P}_{\mathrm{ACS}}(c, p, k)$ and $\mathcal{P}_{\mathrm{RD}}(c, p)$ is also continuous and bounded within this interval. The region of integration (*i.e.*, the domain of

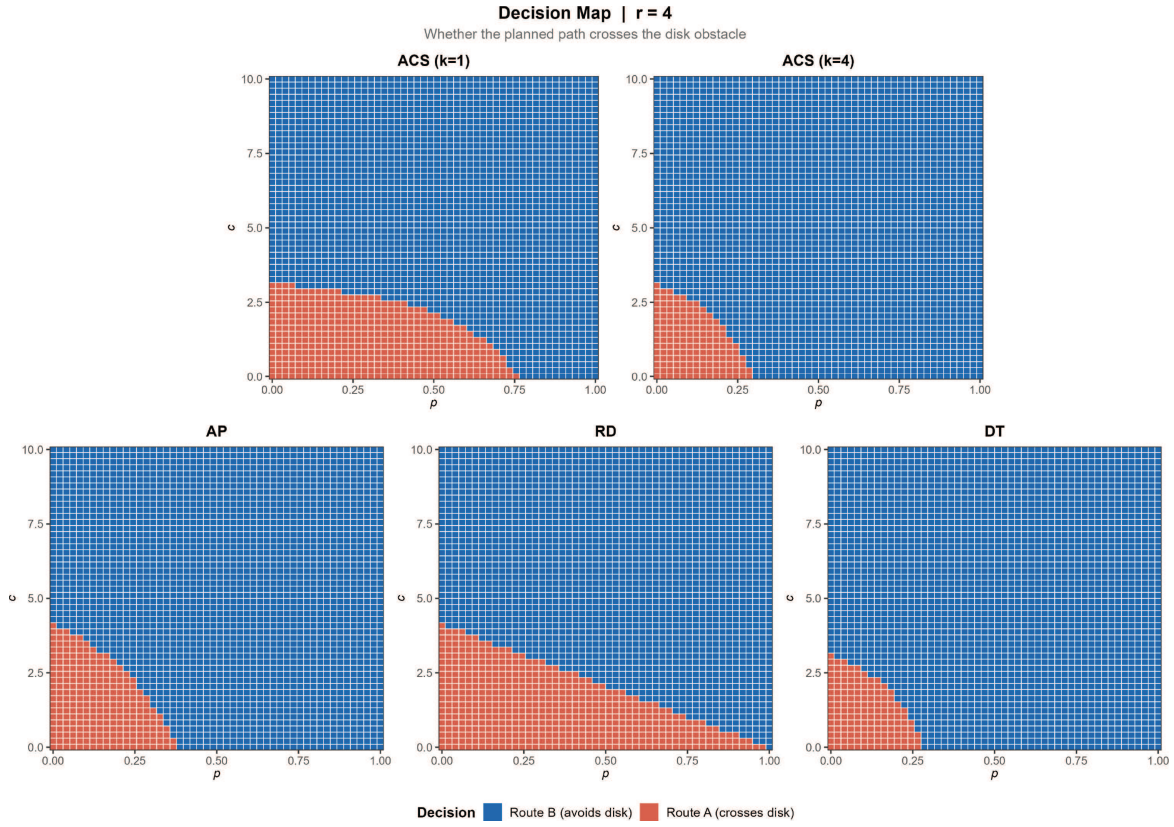


FIGURE 2. Decision map for a single disk obstacle (radius $r = 4$) illustrating how penalty growth in (p, c) affects planned route choice. For each algorithm (RD, AP, DT, and ACS(k)), we evaluate a 50×50 grid of (p, c) values with $p \in [0, 1]$ and $c \in [0, 10]$ on a fixed toy environment and compute the planned shortest path using Dijkstra with penalty-weighted edges. Colors indicate whether the planned path crosses the obstacle or detours around it; the boundary between regions reflects the algorithm's threshold for choosing detour *versus* crossing.

p, $[0, p_{\max}]$) is compact, so the objective function is well-defined and finite and is continuous in k. The domain of the objective function is $k \geq 0$, and the penalty $\mathcal{P}_{\mathrm{ACS}}(c, p, k)$ tends to infinity as $k \to \infty$, since $1 - p < 0$. Thus, the objective function also tends to infinity, making it coercive. Since the L^q norm is bounded below by 0, for $\mathcal{P}_{\mathrm{RD}}$, there exists an optimal $k = k^*_{\mathrm{RD}}(c, q)$ that minimizes the L^q norm over the interval $(0, p_{\max})$. The proof for the penalty functions $\mathcal{P}_{\mathrm{DT}}$ and $\mathcal{P}_{\mathrm{AP}}$ follow similarly, with optimal k values being, $k^*_{\mathrm{DT}}(c, q, d_t)$ and $k^*_{\mathrm{AP}}(c, q)$, respectively. □

Note that the square of the L^2 norm is differentiable with respect to k, so differential calculus can be used to find the minimizer of this norm.

In our analysis, we fix the disambiguation cost as $c = 5$, noting that varying c would influence the k values for the optimal approximation. For finding the optimal k value with the L^1 norm, we use the "L-BFGS-B" optimization method (default in the R function `optim`) which stands for Limited-memory Broyden–Fletcher–Goldfarb–Shanno with Bounds [13], and it is particularly suited for optimization problems with large numbers of variables and with constraints on the parameter values. The L-BFGS-B method is a variation of the L-BFGS algorithm, which itself is an improvement of the BFGS algorithm. The BFGS algorithm is a quasi-Newton

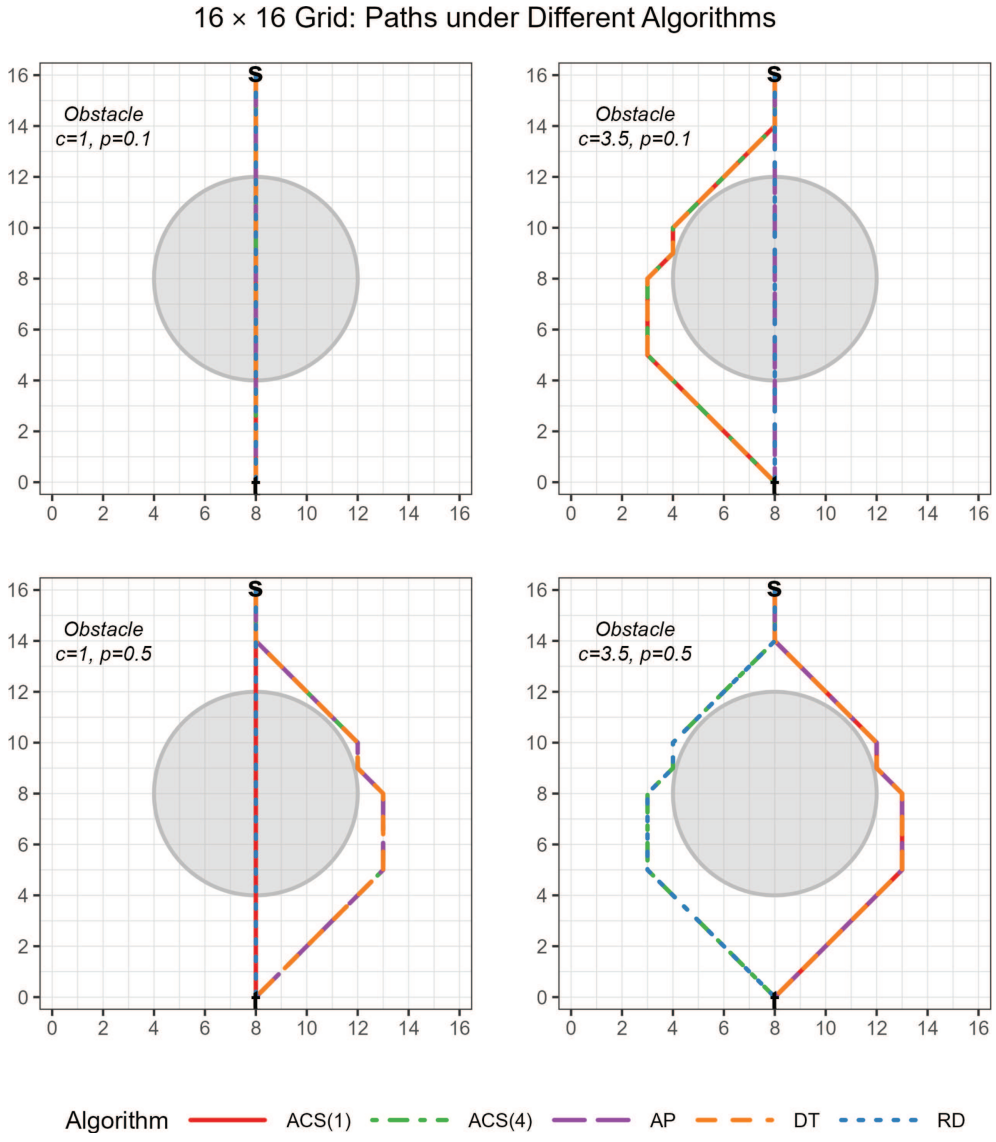


FIGURE 3. Representative planned routes in the toy grid environment. We plot the shortest paths from $s = (8, 16)$ to $t = (8, 0)$ on a 16×6 grid graph with a single disk obstacle of radius $r = 4$ centered at $(8, 8)$, under five penalty rules (ACS($k = 1$), ACS($k = 4$), AP, DT, and RD) for four parameter pairs (c, p). For $(c, p) = (1, 0.1)$ all methods select the crossing route. For $(3.5, 0.1)$, ACS($k = 1$), ACS($k = 4$), and DT detour, while AP and RD still cross. For $(1, 0.5)$, ACS($k = 4$), AP, and DT detour, while ACS($k = 1$) and RD cross. For $(3.5, 0.5)$ all methods detour. These snapshots provide concrete examples of the detour/crossing regimes summarized by the decision map in Figure 2.

method for unconstrained nonlinear optimization problems [18]. For finding the optimal k with the L^2 and L^∞ norms, we use differential calculus using the functions from the symbolic software Maple.

For the DT penalty, the original distance term is naturally interpreted as

$$d_t(x) := \|x - t\|,$$

TABLE 2. Optimal k values with respect to L^1, L^2, and L^∞ norms for RD, AP, and DT algorithms.

Algorithm	L^1 Norm	L^2 Norm	L^∞ Norm
RD	1.774566	1.737021	1.696283
AP	2.699773	2.677187	2.655342
DT	4.979976	5.029031	5.073148

the Euclidean distance from the center of obstacle x to the target t. Likewise, one may define

$$d_s(x) := \|x - s\|.$$

Under the approximately symmetric simulation setting considered here, our numerical results indicate that replacing the obstacle-specific target-distance term by a comparable source-distance term has little practical effect on the approximation. This motivates the use of a fixed surrogate distance parameter

$$d_0 := \frac{d(s,t)}{2},$$

where $d(s,t) = \|s-t\|$ is the Euclidean distance between the source and target. In our simulation configuration, $d(s,t) = 100$, so that $d_0 = 50$. Accordingly, in the approximation analysis and plots for the DT penalty, we use

$$\mathcal{P}_{\mathrm{DT}}(c,p,d_0) \quad \text{with} \quad d_0 = 50.$$

We choose $p_{\max} = 0.9$ for $\mathcal{P}_{\mathrm{RD}}$ and $\mathcal{P}_{\mathrm{AP}}$, and $p_{\max} = 0.7$ for $\mathcal{P}_{\mathrm{DT}}$. These intervals are selected for approximation as they effectively reflect the dynamics of the penalty functions. For p larger than these values, all penalty functions become extremely large due to their exponential growth in p. In our comparison, $\mathcal{P}_{\mathrm{RD}}(c,p)$ shows a slower rate of increase compared to $\mathcal{P}_{\mathrm{DT}}(c,p,d_t)$, which increases more rapidly with respect to p. For instance, within the range of $[0, 0.9]$, the maximum value of $\mathcal{P}_{\mathrm{RD}}(c,p)$ remains below 100, whereas $\mathcal{P}_{\mathrm{AP}}(c,p)$ reaches approximately 450 in the same interval. The $\mathcal{P}_{\mathrm{DT}}(c,p,d_t)$ function attains this 450 threshold within the narrower interval of $[0, 0.7]$ and continues to grow exponentially too large between 0.7 and 0.9. Based on these observations, we propose that the functions $\mathcal{P}_{\mathrm{RD}}(c,p)$, $\mathcal{P}_{\mathrm{AP}}(c,p)$, and $\mathcal{P}_{\mathrm{DT}}(c,p,d_t)$ can be effectively approximated by the function $\mathcal{P}_{\mathrm{ACS}}(c,p,k)$, with an appropriately chosen $k > 0$. Table 2 summarizes the optimal k values obtained for a fixed c value of 5, for different algorithms and norms. We also plot the penalty functions $\mathcal{P}_{\mathrm{RD}}$ and $\mathcal{P}_{\mathrm{ACS}}(c,p,k)$ with optimal k values for $p \in (0, 0.9)$ in Figure 4 where we observe that the optimal k for L^1 norm yields the closest approximation to $\mathcal{P}_{\mathrm{RD}}(c,p)$. Thus, in our subsequent approximations, we use the L^1 norm henceforth. Furthermore, we plot the penalty functions $\mathcal{P}_{\mathrm{AP}}$ and $\mathcal{P}_{\mathrm{ACS}}(c,p,k)$ with optimal k values for $p \in (0, 0.9)$ and $\mathcal{P}_{\mathrm{DT}}$ and $\mathcal{P}_{\mathrm{ACS}}(c,p,k)$ with optimal k values for $p \in (0, 0.7)$ in Figure 5, and observe the close match between the penalty functions.

For a given constant c, let

$$k^*_{\mathrm{RD}}(c) := k^*_{\mathrm{RD}}(c, q = 1)$$

denote the minimizer of the L^1 norm of the difference between $\mathcal{P}_{\mathrm{RD}}(c,p)$ and $\mathcal{P}_{\mathrm{ACS}}(c,p,k)$ over $p \in (0, 0.9)$. That is,

$$k^*_{\mathrm{RD}}(c) = \arg\min_{k \geq 0} \|\mathcal{P}_{\mathrm{RD}}(c,p) - \mathcal{P}_{\mathrm{ACS}}(c,p,k)\|_1 .$$

This function $k^*_{\mathrm{RD}}(c)$ varies with the parameter c and is bounded for c varying within a compact interval, see also Figure 6. Notably, for $c = 5$ and when considering a positive real k, we have found that $k^*_{\mathrm{RD}}(c) \approx 1.77$, as shown in Table 2. This result is consistent with the trends observed in Figure 6. We define $k^*_{\mathrm{AP}}(c)$ and, analogously, the corresponding optimal ACS-approximation parameter for the DT penalty in the same manner.

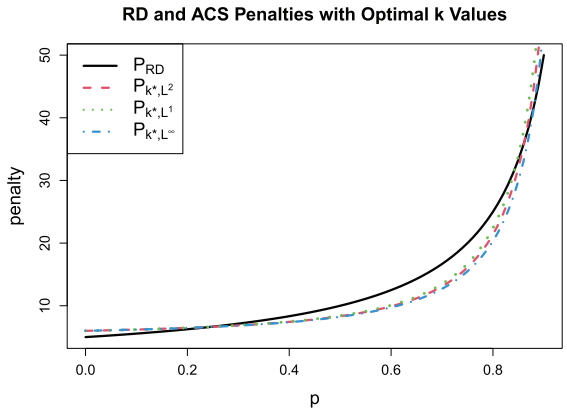
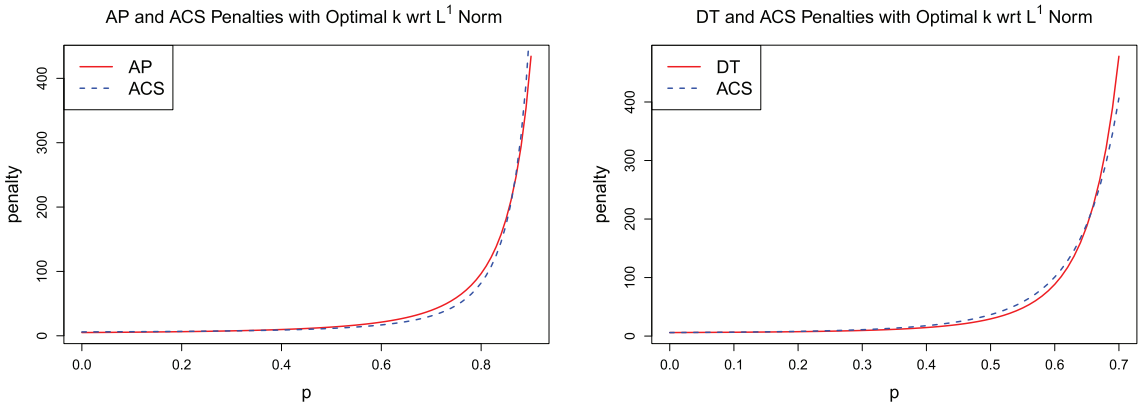


FIGURE 4. Comparison of penalty functions $\mathcal{P}_{\mathrm{RD}}$ and $\mathcal{P}_{\mathrm{ACS}}(c,p,k)$ with optimal k values for $p \in (0, 0.9)$. The penalty function $\mathcal{P}_{\mathrm{ACS}}(c,p,k^*_{\mathrm{RD}})$ with respect to L^2 norm is denoted as P_{k^*,L^2} for brevity, and likewise for P_{k^*,L^1}, and P_{k^*,L^∞}.



(a) (b)

FIGURE 5. Comparison of penalty function values $\mathcal{P}_{\mathrm{ACS}}(c,p,k)$ with optimal k value with respect to L^1 norm and (a) $\mathcal{P}_{\mathrm{AP}}$ for $p \in (0, 0.9)$ and (b) $\mathcal{P}_{\mathrm{DT}}$ in for $p \in (0, 0.7)$. Note that both vertical and horizontal values are differently scaled.

In Figure 6 (left), we also observe that as disambiguation cost increases, the optimal k value $k^*_{\mathrm{RD}}(c)$ also tends to increase. The reason for this is that the difference in the penalty functions is

$$\mathcal{P}_{\mathrm{RD}}(c,p) - \mathcal{P}_{\mathrm{ACS}}(c,p,k) = \frac{c}{1-p} - c - \frac{1}{(1-p)^k} = \frac{cp}{1-p} - \frac{1}{(1-p)^k},$$

which tends to get larger when c increases, and to minimize the absolute difference, k needs to increase as well. Hence the optimal k also tends to increase. A similar argument holds for $k^*_{\mathrm{DT}}(d_t)$ having an increasing trend in d_t as observed in Figure 6 (right).

4.2.2. Comparison of ACS(k) algorithms with other navigation algorithms

We perform Monte Carlo simulations to compare the ACS(k) algorithms with other navigation algorithms for various values of k. In the discretization, we set $i_{\max} = j_{\max}$ to 100, and define the corresponding graph as

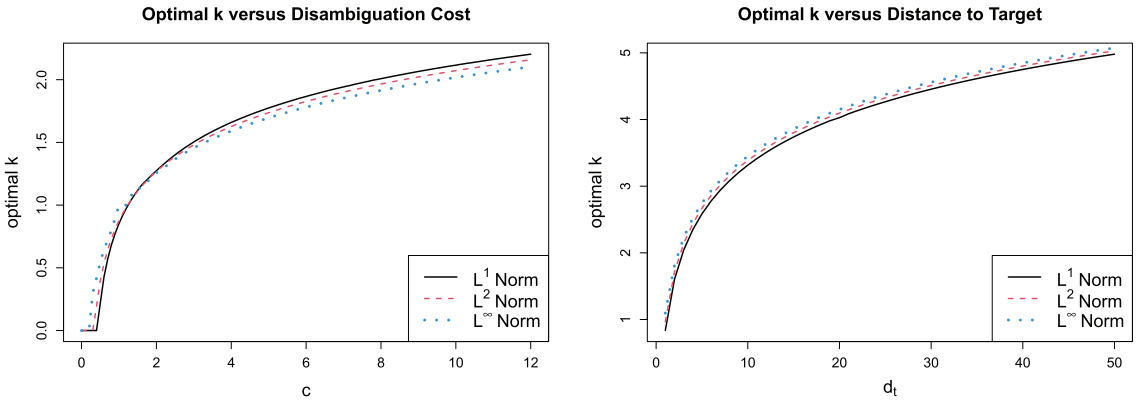


FIGURE 6. Optimal k values $k^*_{\mathrm{RD}}(c)$ *versus* disambiguation cost c (*left*) and optimal k values $k^*_{\mathrm{DT}}(d_0)$ *versus* the surrogate DT distance parameter d_0 (*right*), for L^q-norm differences ($q = 1, 2, \infty$).

$G = (V, E)$ based on the grid with unit squares. The initial vertex s is designated at coordinates $s = (50, 100)$, situated along the upper boundary of the grid. Conversely, the target vertex t is allocated at coordinates $t = (50, 1)$ on the lower boundary, directly aligned with s. This configuration is reasonable, particularly in the context of employing the grid to represent a segment of a coastline. Within this framework, the point s represents the position of offensive forces, whereas t denotes the position of defensive forces. An implementation of the simulation environment generators and all navigation algorithms used in our experiments is available in Zhou [28].

The obstacle (disk) radius is set to $r = 4.5$ units, with a constant disambiguation cost $c = 5$. We specify the distributions for $p_{_F}$ and $p_{_T}$ as $F_F = \mathrm{Beta}(4 - \lambda, 4 + \lambda)$ and $F_T = \mathrm{Beta}(4 + \lambda, 4 - \lambda)$, respectively, for some $\lambda \in [0, 4)$. To assess the performance of NAVA's sensor under various scenarios, we conduct simulations over a range of λ values. Setting $\lambda = 0$ implies a lack of precision in the sensor, equivalent to NAVA operating with a blind sensor. In this situation, the probability of correctly identifying an obstacle is $1/2$ (since $P(p_{_T} > 1/2) = P(p_{_F} < 1/2) = 1/2$ when $\lambda = 0$), leading to random decision-making as in flipping a coin. On the other hand, as $\lambda \to 4$, sensor tends to have perfect accuracy, *i.e.*, $\lim_{\lambda\to 4} P(p_{_T} > 1/2) = \lim_{\lambda\to 4} P(p_{_F} < 1/2) = 1$ (in fact $\lim_{\lambda\to 4} P(p_{_T} > 1-\varepsilon) = \lim_{\lambda\to 4} P(p_{_F} < \varepsilon) = 1$ for any $\varepsilon > 0$) enabling NAVA to distinguish true obstacles with (almost) complete certainty. For our simulations, we adopt $\lambda = 2$ as a balanced choice for the sensors specificity and sensitivity, as recommended by Aksakalli and Ari [3], Aksakalli and Ceyhan [4], Ye *et al.* [27], and Priebe *et al.* [20]. The centers of disk-shaped obstacles, $x \in \mathcal{X} = \mathcal{X}_F \cup \mathcal{X}_T$, are uniformly distributed over the region $[10, 90] \times [10, 90]$ to guarantee the presence of a feasible (albeit long) path between s and t. The numbers of false obstacles and true obstacles are set to $n_F = 25, 50, 75, 100, 125$ and $n_T = 25, 50, 75, 100, 125$, respectively, following the setting outlined in Witherspoon *et al.* [24] and Priebe *et al.* [20].

Our extensive simulation results with mixed obstacles (not presented) indicate that the ACS(k) algorithms demonstrate a close approximation to the RD, AP, and DT algorithms for corresponding optimal values of k. In fact, the percentage error, defined as

$$\text{percentage error} = \left| \frac{\mathcal{C}_{\mathrm{ACS}(k)} - \mathcal{C}_{\mathrm{RD}}}{\mathcal{C}_{\mathrm{RD}}} \right| \times 100.$$

is found to be less than 2% in our experiments. This is because for $p \in (0, 1/2]$, the corresponding error functions are pretty close, and for larger p, each penalty functions motivates choosing paths with minimum number of true-obstacle disambiguations. These results extend to the cases of false-obstacle-only and true-obstacle-only as well.

We next prove that the traversal cost of NAVA using RD, AP, and DT algorithms can be closely approximated by those from ACS(k) algorithms for corresponding optimal values of k.

Proposition 4.2. *For fixed $c > 0$ and $\lambda \in (0, 4)$, $d_0 > 0$, and $n < \infty$, let $\mathcal{C}_{\mathrm{RD}}$, $\mathcal{C}_{\mathrm{AP}}$, and $\mathcal{C}_{\mathrm{DT}}$ represent the traversal costs when NAVA uses the algorithms RD, AP, and DT, respectively. Additionally, let $\mathcal{C}_{\mathrm{ACS}(k)}$ be traversal costs obtained from the ACS(k) algorithms for $k \geq 0$. Then, there exists optimal $k = \tilde{k}$ values that minimizes the absolute difference between the expected traversal costs: $|\mathbf{E}[\mathcal{C}_{\mathrm{RD}}] - \mathbf{E}[\mathcal{C}_{\mathrm{ACS}(k)}]|$, $|\mathbf{E}[\mathcal{C}_{\mathrm{AP}}] - \mathbf{E}[\mathcal{C}_{\mathrm{ACS}(k)}]|$, and $|\mathbf{E}[\mathcal{C}_{\mathrm{DT}}] - \mathbf{E}[\mathcal{C}_{\mathrm{ACS}(k)}]|$.*

Proof. The random variable $\mathcal{C}_{\mathrm{RD}}$ is a theoretically well-defined discrete random variable since the number of possible walks (hance paths) from s to t and number of possible edge intersections with a finite number of obstacles are finite (although combinatorially explosive), making the number of possible traversal costs to be finite for given c. The pmf (and hence cdf) of these costs depend on c and distribution of p (*i.e.*, λ) through the penalty function $\mathcal{P}_{\mathrm{RD}}$. Then it trivially follows that the expected value $\mathbf{E}[\mathcal{C}_{\mathrm{RD}}]$ is well-defined and exists (and is constant for given c and λ). The same hold for $\mathcal{C}_{\mathrm{ACS}(k)}$ which also depends on $k > 0$. Furthermore, expected value of $\mathcal{C}_{\mathrm{ACS}(k)}$ is continuous in k since small changes in k imply small changes in the probabilities in its pmf and hence in its expected value. So, $|\mathbf{E}[\mathcal{C}_{\mathrm{RD}}] - \mathbf{E}[\mathcal{C}_{\mathrm{ACS}(k)}]|$ is a continuous function of k, denote it as $f_{\mathrm{RD}}(k)$. Since $f_{\mathrm{RD}}(k)$ is continuous and bounded for $k \in [0, \infty)$, and finite as k approaches 0 and as $k \to \infty$, we apply the transformation $y = \exp(-k)$ to map the domain of k onto $y \in (0, 1]$ and let $g(y) = f_{\mathrm{RD}}(-\log(y))$. Notice that the transformation $y = \exp(-k)$ is a bijective mapping from $k \in (0, \infty)$ to $y \in (0, 1)$. The function $g(y) = f_{\mathrm{RD}}(-\log(y))$ is now defined on the interval $(0, 1]$ which is a bounded interval and $g(y)$ can be extended continuously to the closed interval $[0, 1]$ and remains bounded as y approaches 0, then $g(y)$ is continuous on a compact set $[0, 1]$. So, by the Extreme Value Theorem, $g(y)$ attains its minimum at some point $y_0 \in (0, 1)$, then the corresponding $k_0 = -\log(y_0) \in (0, \infty)$ is the point at which $f_{\mathrm{RD}}(k)$ attains its minimum in its original domain. The proofs for $|\mathbf{E}[\mathcal{C}_{\mathrm{AP}}] - \mathbf{E}[\mathcal{C}_{\mathrm{ACS}(k)}]|$, and $|\mathbf{E}[\mathcal{C}_{\mathrm{DT}}] - \mathbf{E}[\mathcal{C}_{\mathrm{ACS}(k)}]|$ follow similarly. □

Proposition 4.2 essentially asserts the existence of a minimizer in an optimization problem, and implies that the traversal costs computed by NAVA using the algorithms RD, AP, and DT can be closely approximated by the traversal costs obtained from the corresponding ACS(k) algorithms for some optimal value of k which depends on which property of the traversal cost is optimized.

The above results similarly extends for the distributions of the traversal costs.

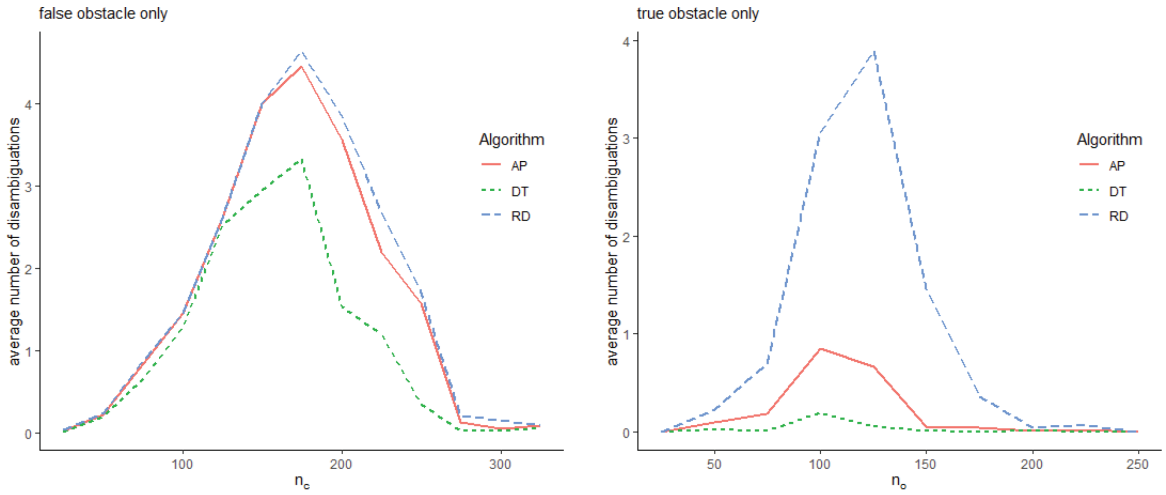
Corollary 4.1. *Under the conditions in Proposition 4.2, there exists optimal $k = \hat{k}$ values that minimize the norms between the cdfs of $\mathcal{C}_{\mathrm{RD}}$, $\mathcal{C}_{\mathrm{AP}}$, and $\mathcal{C}_{\mathrm{DT}}$ and the cdf of $\mathcal{C}_{\mathrm{ACS}(k)}$ with respect to L^q norm for all $q > 0$.*

The proof is similar to the proof of Proposition 4.2. Each of the results of Proposition 4.2 and Corollary 4.1 essentially assert the existence of a minimizer in an optimization problem, and implies that the traversal costs computed by NAVA using the algorithms RD, AP, and DT can be closely approximated by the traversal costs obtained from the corresponding ACS(k) algorithms for some optimal value of k which depends on which aspect of the traversal cost is optimized. However, for a given algorithm, say RD algorithm, the optimal k values in Propositions 4.1 and 4.2 and Corollary 4.1 are not necessarily same, as highlighted in the below remark.

Remark 4.1. For a fixed disambiguation cost $c > 0$ and $\lambda \in (0, 4)$, and $d_t > 0$ we just showed that:

$$\mathcal{C}_{\mathrm{RD}} \approx \mathcal{C}_{\mathrm{ACS}(\tilde{k}_{\mathrm{RD}})}, \quad \mathcal{C}_{\mathrm{AP}} \approx \mathcal{C}_{\mathrm{ACS}(\tilde{k}_{\mathrm{AP}})}, \quad \text{and } \mathcal{C}_{\mathrm{DT}} \approx \mathcal{C}_{\mathrm{ACS}(\tilde{k}_{\mathrm{DT}})}$$

in expectation (and in distribution with $\tilde{k}$'s replaced with $\hat{k}$'s). In other words, the traversal costs computed by NAVA using the algorithms RD, AP, and DT can be closely approximated in mean (and in distribution) by the traversal costs obtained from the corresponding ACS(k) algorithms for the specified values of $\tilde{k}$. However these optimal $\tilde{k}$ values may not necessarily be the optimal values in Proposition 4.1 nor in Corollary 4.1. For example $\tilde{k}_{\mathrm{RD}}$ is not necessarily equal to $k^*_{\mathrm{RD}}(c, p)$ value in Proposition 4.1, or $\hat{k}_{\mathrm{RD}}(c, p)$ value in Corollary 4.1.



(a) (b)

FIGURE 7. Comparison of the average number of disambiguations by RD, AP, and DT algorithms with $c = 5$, $d_t = 50$, and $\lambda = 2$ in the (a) false-obstacle-only case with $n_F = 50, 75, \ldots, 300$, and (b) true-obstacle-only case with $n_T = 50, 75, \ldots, 300$. Note that vertical axes are differently scaled.

The DT algorithm tends to avoid disambiguations when encountering true obstacles, while RD and AP algorithms are more prone to disambiguate obstacles. We illustrate this difference by examining the total number of disambiguations for each algorithm. In the false-obstacle-only case, RD and AP exhibit similar average disambiguation counts (see Fig. 7a), while DT's count decreases with an increasing number of false obstacles. In true-obstacle-only settings (Fig. 7b), RD's average disambiguation count initially increases, reaches a peak, and then decreases as the number of true obstacles grows. In contrast, AP maintains a consistently low average count, and DT's average is nearly zero, reflecting its preference for routes with fewer or no obstacles (so as to disambiguate as few obstacles as possible).

5. Empirical evaluation and comparative analysis

We begin with a summary plot, Figure 8, that provides a consolidated comparison of mean traversal cost across the principal penalty-based heuristics under three key operating factors: sensor accuracy λ, disambiguation cost c, and obstacle density n_F or n_T. Panels (a)–(b) show that traversal cost generally decreases as λ increases in both false-only and true-only environments, reflecting improved sensing and hence more effective routing decisions. The curves also converge as $\lambda \to 4$, where the sensor becomes nearly perfect and all methods behave similarly. Panels (c)–(d) illustrate the effect of disambiguation cost. In the false-obstacle-only case, the mean traversal cost increases with c for all methods, which is consistent with more frequent payment of disambiguation costs when obstacles are traversable. In the true-obstacle-only case, increasing c tends to reduce the total cost for several penalties by encouraging earlier detours and fewer disambiguations, thereby producing clearer separation between more conservative and less conservative rules. Panels (e)–(f) summarize the role of obstacle density. Increasing n_F produces a monotone increase in traversal cost, since a larger number of false obstacles creates more opportunities for unnecessary disambiguation. In contrast, increasing n_T yields a non-monotone pattern with a peak at intermediate densities, consistent with greater rerouting pressure when true obstacles are common but not yet so dense that detours become uniformly dominant. Finally, across panels, the ACS$(\cdot)$ curves closely

track their corresponding classical baselines (RD, AP, and DT), providing end-to-end visual confirmation that the norm-based penalty approximations yield comparable traversal performance under these operating regimes.

The remainder of Section 5 provides the supporting detail. Section 5.1 describes the experimental design and baseline setting; Sections 5.2–5.4 then report targeted sensitivity analyses with respect to (i) sensor accuracy (parameterized by λ through the distributions of probability marks), (ii) disambiguation cost c, and (iii) obstacle density (via n_F and n_T). Section 5.5 summarizes the main empirical takeaways and connects them back to the structural properties of the penalty functions introduced in Sections 3 and 4.

We now evaluate the navigation heuristics introduced in Sections 3 and 4 through Monte Carlo simulation. The goal is not to establish optimality, but to understand how structural differences in penalty growth influence observed traversal cost under varying sensor and obstacle regimes.

In the previous section, we have examined how well the ACS(k) algorithms approximate the other penalty functions when disambiguation cost $c = 5$ and sensor accuracy parameter $\lambda = 2$. An important subsequent inquiry is to explore how the performance of navigation algorithms depend on the sensor's detection parameter λ, the disambiguation cost c and the number of obstacles in the traversal window.

5.1. Performance of NAVA *versus* sensor accuracy

Recall that before NAVA begins its traversal, its sensor assigns probabilities of non-traversability (*i.e.*, obstacles being true) to disk-shaped obstacles using its sensor. These probabilities, $p_{_F}$ and $p_{_T}$, are modeled with distribution functions F_F and F_T, respectively, so that $p_{_F} \leq_{st} p_{_T}$ and $P(p_{_F} \leq 1/2) \geq P(p_{_F} \geq 1/2)$ and $P(p_{_T} \leq 1/2) \leq P(p_{_T} \geq 1/2)$. Usually, F_F and F_T are taken to be Beta$(4-\lambda, 4+\lambda)$ and Beta$(4+\lambda, 4-\lambda)$ distributions, respectively, for the same $\lambda \in [0, 4)$ which satisfy these conditions for $\lambda \in (0, 4)$. See Section 4.2.2 for the limiting behaviour of F_F and F_T for extreme values of $\lambda \in (0, 4)$.

We first provide the definition of stochastic ordering for completeness. If X and Y are random variables defined on the same sample space Ω, then X is *stochastically smaller* than Y (denoted as $X \leq_{st} Y$) if $F_X(\omega) \geq F_Y(\omega)$ for all $\omega \in \Omega$. Then we present a stochastic ordering result for the estimated risk of NAVA when ACS(k) algorithm is employed.

Proposition 5.1. *Let $\mathcal{R}_{\text{ACS}}(k, \lambda, \mathcal{X})$ denote the estimated traversal risk computed by the ACS(k) algorithm before its traversal with the sensor accuracy parameter $\lambda \in (0, 4)$ and the obstacle set $\mathcal{X}$ as described above. Then, for any disambiguation cost $c > 0$, we have the following results:*

(a) *For fixed $k \geq 0$ and for any $0 \leq \lambda' < \lambda \leq 4$ (i.e., when the precision of the sensor increases), we have $\mathcal{R}_{\text{ACS}}(k, \lambda, \mathcal{X}_F) \leq_{st} \mathcal{R}_{\text{ACS}}(k, \lambda', \mathcal{X}_F)$ and $\mathcal{R}_{\text{ACS}}(k, \lambda', \mathcal{X}_T) \leq_{st} \mathcal{R}_{\text{ACS}}(k, \lambda, \mathcal{X}_T)$.*

(b) *For fixed $\lambda \in [0, 4]$ and for any $k > k' \geq 0$, we have $\mathcal{R}_{\text{ACS}}(k, \lambda, \mathcal{X}) \geq \mathcal{R}_{\text{ACS}}(k', \lambda, \mathcal{X})$ with probability 1,*

(c) *For fixed $\lambda \in [0, 4]$, $k \geq 0$, and fixed number of obstacles $n > 0$, we have $\mathcal{R}_{\text{ACS}}(k, \lambda, \mathcal{X}_F) \leq_{st} \mathcal{R}_{\text{ACS}}(k, \lambda, \mathcal{X}_T)$*

Sketch of Proof. Let $k \geq 0$ be given. (a) Suppose $\mathcal{X} = \mathcal{X}_F$ (*i.e.*, false-obstacle-only case). For any false obstacle D, let $p_{_F}(\lambda)$ be its probability mark from the Beta$(4-\lambda, 4+\lambda)$ distribution. From properties of Beta distribution, we have $p_{_F}(\lambda) \leq_{st} p_{_F}(\lambda')$ for $0 \leq \lambda' < \lambda \leq 4$. Thus, for fixed $k \geq 0$, $c + \frac{1}{(1-p_{_F}(\lambda))^k} \leq_{st} c + \frac{1}{(1-p_{_F}(\lambda'))^k}$. Since this ordering holds for each edge and risk of the path is sum of these penalty terms and Euclidean distances, we have the desired result $\mathcal{R}_{\text{ACS}}(k, \lambda, \mathcal{X}_F) \leq_{st} \mathcal{R}_{\text{ACS}}(k, \lambda', \mathcal{X}_F)$.

Similarly, when $\mathcal{X} = \mathcal{X}_T$ (*i.e.*, true-obstacle-only case), let $p_{_T}(\lambda)$ be its probability mark from the Beta$(4+\lambda, 4-\lambda)$ distribution. From properties of Beta distribution, we have $p_{_T}(\lambda') \leq_{st} p_{_T}(\lambda)$ for $0 \leq \lambda' < \lambda \leq 4$. As in the false-obstacle-only case, we have $\mathcal{R}_{\text{ACS}}(k, \lambda', \mathcal{X}_T) \leq_{st} \mathcal{R}_{\text{ACS}}(k, \lambda, \mathcal{X}_T)$.

(b) For any fixed λ (constant sensor accuracy), as k increases, so does the penalty function $\mathcal{P}_{\text{ACS}}(c, p, k)$. That is, for $k > k'$, we have $\mathcal{P}_{\text{ACS}}(c, p, k) > \mathcal{P}_{\text{ACS}}(c, p, k')$ with probability 1, which implies their sums with the added associated Euclidean lengths satisfy the same inequality. Hence, the result follows. Notice here that this result also implies the stochastic ordering $\mathcal{R}_{\text{ACS}}(k', \lambda, \mathcal{X}) \leq_{st} \mathcal{R}_{\text{ACS}}(k, \lambda, \mathcal{X})$.

(c) This case is similar to part (a). For a false obstacle, $p_{_F}(\lambda)$ is from the Beta$(4-\lambda, 4+\lambda)$ distribution and for true obstacles $p_{_T}(\lambda)$ is from the Beta$(4+\lambda, 4-\lambda)$ distribution. From properties of the Beta distribution, we have $p_{_F}(\lambda) \leq_{st} p_{_T}(\lambda)$. Hence, the desired result follows as in part (a). □

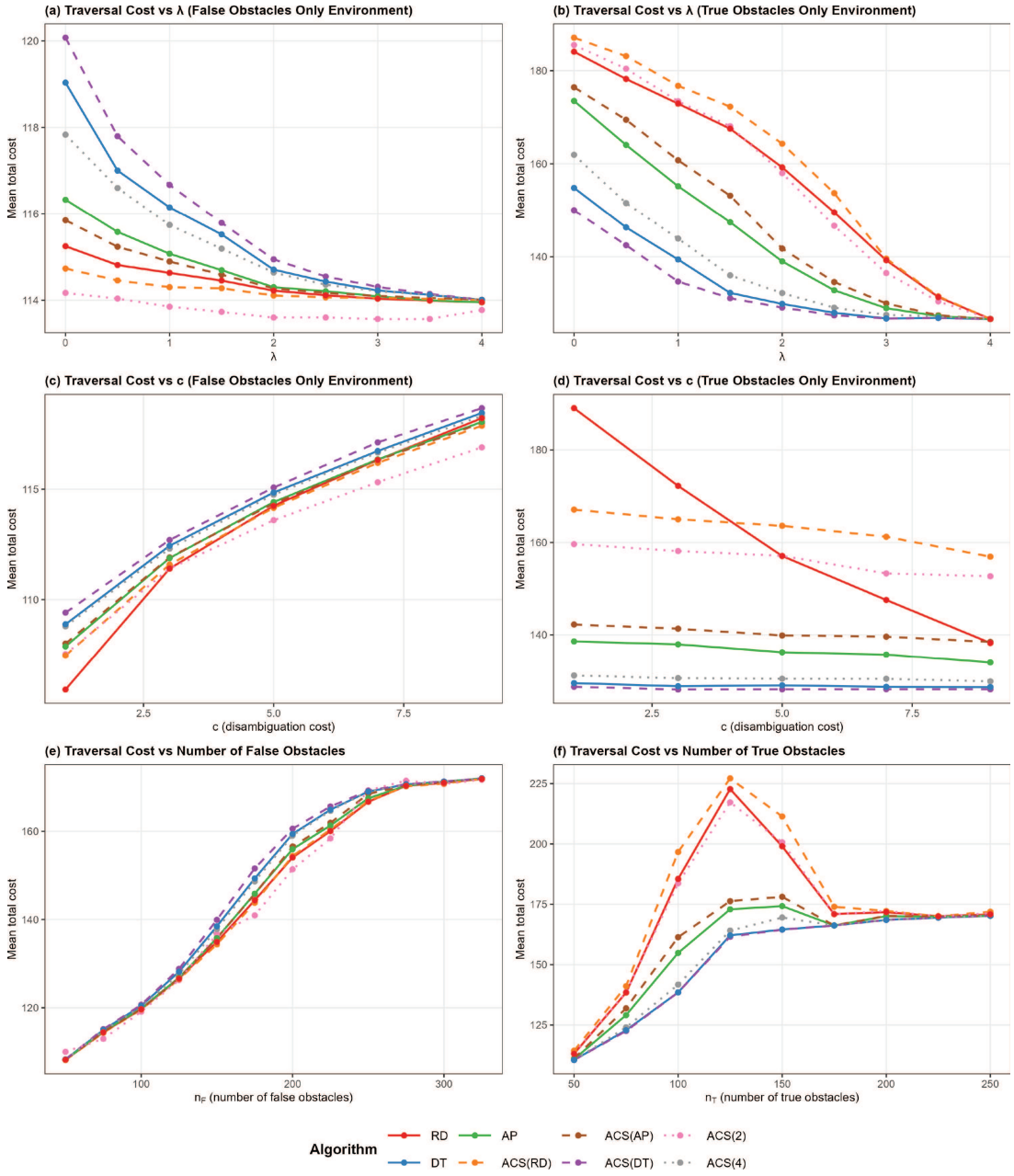


FIGURE 8. Consolidated performance comparison across eight algorithms (RD, AP, DT, ACS(k^*_{RD}), ACS(k^*_{AP}), ACS(k^*_{DT}), ACS(2), and ACS(4)). Results are shown separately for false-obstacle-only and true-obstacle-only environments. Panels (a–b) report mean total traversal cost *versus* sensor parameter λ (nine values on $(0, 4)$); panels (c–d) report mean total traversal cost *versus* disambiguation cost $c \in \{1, 3, 5, 7, 9\}$; and panels (e–f) report mean total traversal cost *versus* obstacle count (false obstacles n_F from 50 to 325 in increments of 25; true obstacles n_T from 50 to 250 in increments of 25).

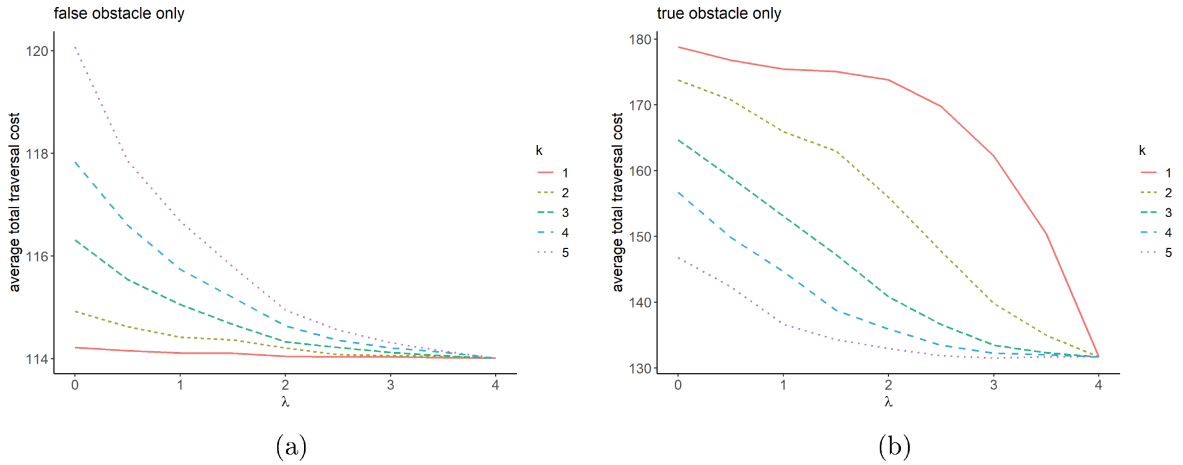


FIGURE 9. Mean traversal lengths (averaged over the obstacle number levels) for ACS(k) algorithm with $k = 1 - 5$ in the (a) false-obstacle-only case with $n_F = 50, 75, 100, 125$, (b) true-obstacle-only case with $n_T = 50, 75, 100, 125$ in the traversal window where obstacles are uniformly distributed in the traversal window. Note that vertical axes are differently scaled.

We compare the mean traversal lengths (averaged over obstacle number levels) for ACS(k) algorithm with $k = 1, 2, \ldots, 5$ and $c = 5$ in the false-obstacle-only case with $n_F = 50, 75, 100, 125$, and true-obstacle-only case with $n_T = 50, 75, 100, 125$ where obstacles are uniformly distributed in the traversal window. We present the results in Figure 9, where we observe that as the sensor's precision (λ) increases, the mean traversal length (averaged over the obstacle number levels) decreases across each ACS(k) algorithm (for $k = 1 - 5$), eventually converging towards a benchmark value, but with different trends and at different rates of convergence. In the false-obstacle-only case, at each λ value, as k increases, the (estimated) mean traversal cost tends to increase as seen in Figure 9a. Conversely, in the true-obstacle-only case, the trend is in the opposite direction: at each λ value, as k increases, the (estimated) mean traversal cost tend to decrease as seen in Figure 9b. The value of k seems to have the reverse effect on the traversal length for true and false obstacles.

As seen in Proposition 5.1, with increasing k, the estimated risk values tend to increase, causing NAVA to choose risk-averse (*i.e.*, more obstacle-free) paths. But in scenarios with false obstacles, this choice tend to lead to longer or costlier routes due to a preference for safer paths (although there are shorter cost traversable paths available). Conversely, with true obstacles, this choice tend to result in shorter traversals as it aims to avoid increased risk from true obstacle intersections (hence blockages), notably with higher probability marks affecting the risk increase. This result suggests that for lower p markings, NAVA should disambiguate more and pass through the false obstacles and with higher p markings, NAVA should avoid the obstacles and choose to circumvent the true obstacles.

The above simulation results are suggestive for the selection of the most effective ACS(k) algorithm for a given λ value when either $\mathcal{X} = \mathcal{X}_F$ or $\mathcal{X} = \mathcal{X}_T$. However, the optimal choice of k in an ACS(k) algorithm remains unresolved in cases where both true and false obstacles are available (*e.g.*, uniformly distributed) within the traversal environment.

Figure 10 shows settings with a total of $n_F + n_T$ obstacles uniformly distributed, where n_F and n_T range from 25 to 100. In these settings, ACS(k) with $k = 4, 5$ are the most effective algorithms overall (with ACS($k = 5$) being slightly better, especially for smaller λ values). We note that the preferred ACS(k) algorithm can vary based on the proportion of true and false obstacles. To illustrate this, we consider cases where $n_F + n_T = 100$, achievable with combinations like $(n_F, n_T) = (25, 75), (50, 50), (75, 25)$. We present the case for $(n_F, n_T) = (25, 75)$ in Figure 11a, where we see that when true obstacles are much larger than false obstacles, larger k

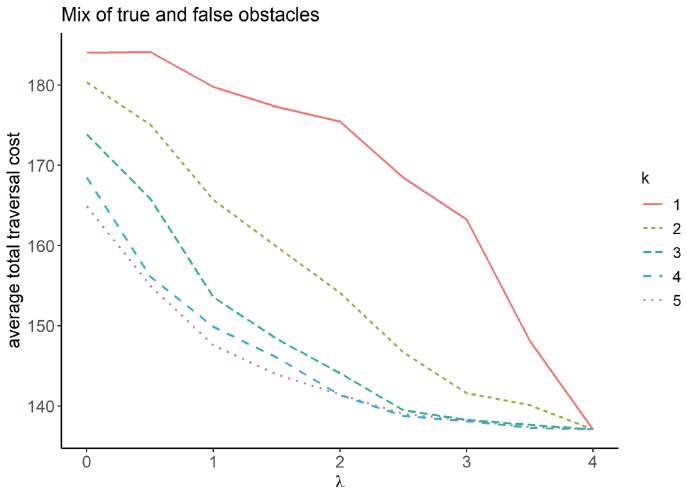


FIGURE 10. Mean traversal costs with ACS(k) algorithm when there are a total of $n_F + n_T$ obstacles uniformly distributed in the traversal window with $n_F = 25, 50, 75, 100$ and $n_T = 25, 50, 75, 100$.

values are better and performance gets better with increasing k with ACS(5) being the optimal choice. In Figure 9b, we present the evenly split case of $(n_F, n_T) = (50, 50)$ shown in Figure 11b. Notice that ACS(k) with $k = 3, 4, 5$ show very similar performance with $k = 4$ being the optimal for most λ values. Figure 11c presents the case of $(n_F, n_T) = (75, 25)$, and indicates that the performance order of ACS(k) depends on λ. In particular, for smaller λ, performance gets better with decreasing k, and for moderate λ, $k \approx 3$ seems to outperform others, and for larger λ, the performance is very similar for all k values. Hence, in uniformly-distributed-obstacle cases, the choice of the most suitable algorithm depends on the number of true obstacles (n_T), the number of false obstacles (n_F), and the sensor parameter λ. When there are more true obstacles in the traversal region, penalizing the obstacle disambiguations gets more important, so larger k is suggested. However, there is no clear message when there are more false obstacles in the environment. Furthermore, Figure 10 also indicates that as λ increases (*i.e.*, the precision of the sensor gets better), the performance of ACS(k) improves for each k value.

When the obstacles are uniformly distributed, it is proposed that for a given set of parameters (n_T, n_F, c, λ), the optimal ACS(k) algorithm can be selected by adjusting the penalty function $\mathcal{P}_{\text{ACS}}(c, p, k)$, where k is determined based on $n_T, n_F, c,$, and λ. An alternative method involves considering the ratio $\rho = n_T/n_F$. Based on our results, the suggested optimal choice of k for ACS(k) algorithm is as follows:

With n_T, n_F sufficiently larger than 0,

- if $\rho \leq 1$, it is advisable to opt for either k around 2 or 3.
- For $\rho > 1$, the preferable choices are $k \geq 4$, *i.e.* k around 4 or 5.

In the extreme scenario where $\min(n_T, n_F) \approx 0$,

- when $n_T \approx 0$ but n_F is sufficiently larger than 0, smaller k values (*i.e.*, $k \lesssim 1$) are recommended.
- Conversely, if $n_F \approx 0$ but n_T is sufficiently larger than 0, choosing k around 4 or 5 is the most reasonable approach.

Essentially, when obstacles are from any distribution, we suggest larger values of k when there are more true obstacles and smaller values of k when there are more false obstacles. However, the particular values of k will depend on the spatial distribution of the obstacles (which can also be determined empirically by Monte Carlo simulations). Furthermore, the true value of ρ is not known *a priori*, but can be estimated by the probability marks assigned by NAVA's sensor.

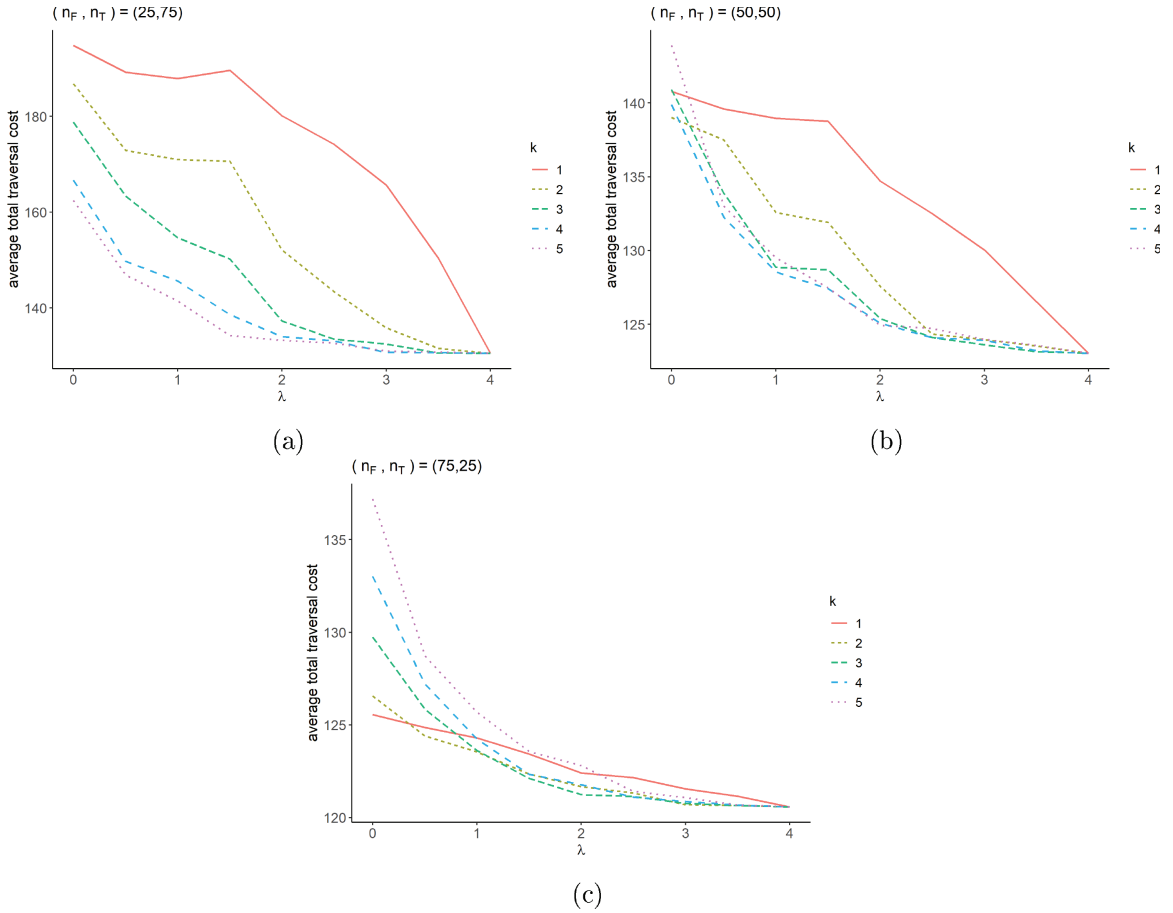


FIGURE 11. Mean traversal costs estimated by algorithms ACS(k): (a) for $(n_F, n_T) = (25, 75)$, (b) for $(n_F, n_T) = (50, 50)$, (c) and for $(n_F, n_T) = (75, 25)$. Notice that vertical axes are differently scaled.

5.2. Performance of NAVA *versus* disambiguation cost

In Section 5.1, the performance of the ACS(k) algorithms was analyzed with a fixed disambiguation cost $c = 5$. A relevant inquiry pertains to the determination of the optimal value of k within the ACS(k) family when the cost associated with disambiguation varies, while keeping the sensor parameter λ constant. We set $\lambda = 2$ and vary the disambiguation cost across $c = 1, 3, 5, 7, 9$ units. Consistent with previous simulation settings, the obstacles will be uniformly distributed within the navigation window. Figure 12 shows that ACS(3), ACS(4), and ACS(5) are considerably more effective than their counterparts. However, as indicated in Figure 11, this observation may not extend for different combinations of true and false obstacles. Along this line, we examine simulation settings where $n_F + n_T = 100$, achievable with combinations like $(n_F, n_T) = (25, 75), (50, 50), (75, 25)$.

Figures 13a and 13b demonstrate that ACS(k) with $k \geq 4$ generally outperforms the other algorithms. We also notice that ACS(k) with $k = 4, 5$ perform similarly. The trends in mean traversal costs are similar to the ones in Section 5.1 for unevenly split types of obstacles. In particular, for $(n_F, n_T) = (25, 75)$ case, ACS(k) with $k = 4, 5$ are the most effective algorithms overall, with ACS($k = 5$) being slightly better, especially for smaller disambiguation cost c values. For the evenly split case of $(n_F, n_T) = (50, 50)$, ACS(k) with $k = 3, 4, 5$ performs similarly, with $k = 4$ being optimal for most c values. On the other hand, for $(n_F, n_T) =$

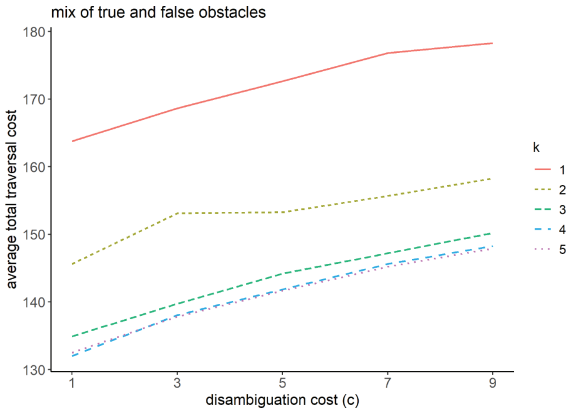


FIGURE 12. Mean traversal costs estimated by the algorithms ACS(k) when there are $n_F + n_T$ obstacles which were uniformly generated with $n_F, n_T \in \{25, 50, 75, 100\}$.

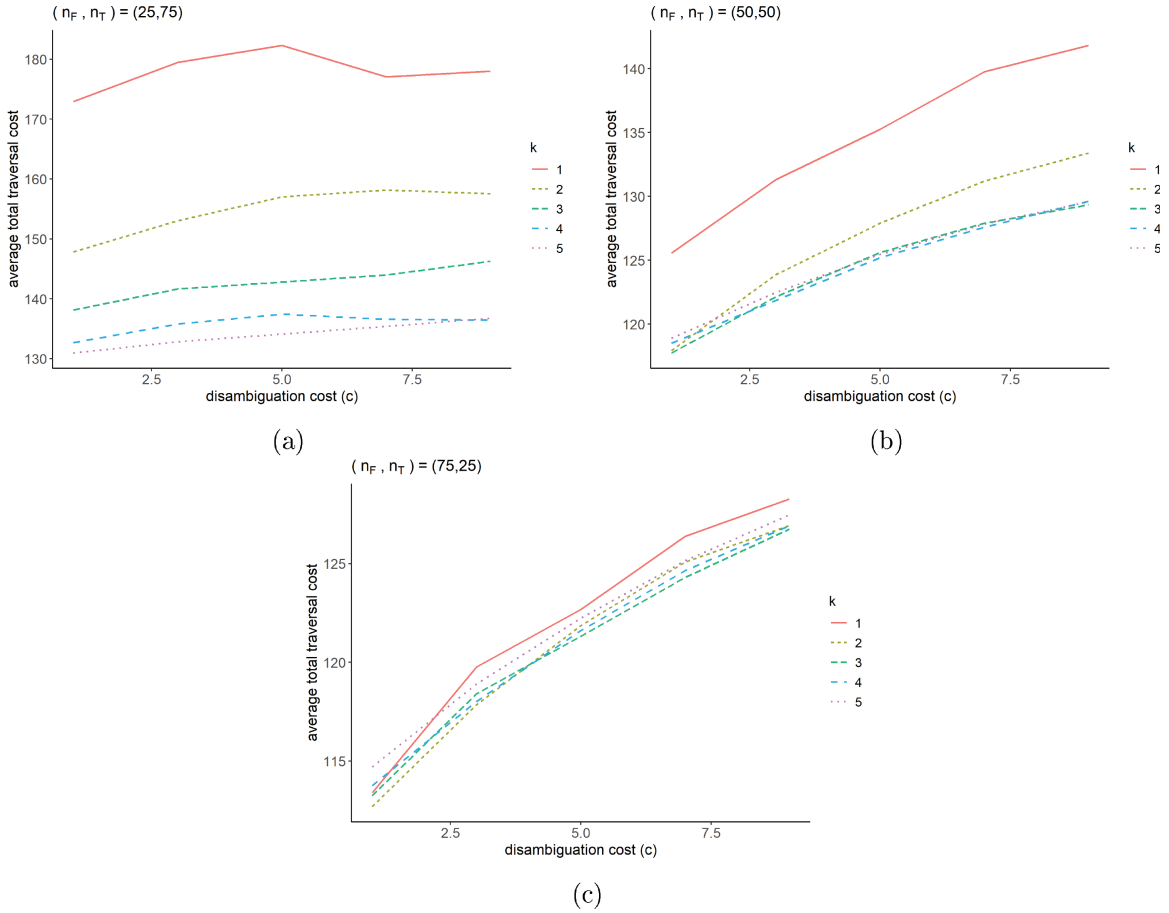
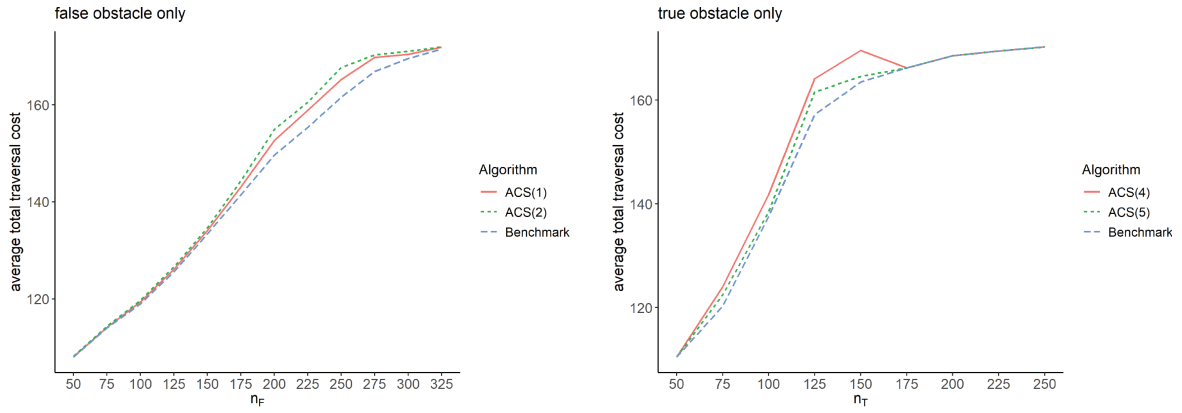


FIGURE 13. Comparison of mean traversal costs estimated by the algorithms ACS(k): (a) for $(n_F, n_T) = (25, 75)$, (b) for $(n_F, n_T) = (50, 50)$, (c) and for $(n_F, n_T) = (75, 25)$. Notice that vertical axes are differently scaled.



(a) (b)

FIGURE 14. (a) Mean traversal costs by benchmark, ACS(1), ACS(2) for varying n_F with $n_F = 50, 75, \ldots, 325$. (b) Lengths by benchmark, ACS(4), ACS(5) against n_T with $n_T = 50, 75, \ldots, 250$. Notice that horizontal axes are differently scaled.

$(75, 25)$ case, moderate k values (*i.e.*, 2,3,4) show very similar performance. Hence, in uniformly-distributed-obstacle cases, the choice of the most suitable algorithm depends on the number of true obstacles (n_T), false obstacles (n_F), and disambiguation cost c. Furthermore, it is noted that as disambiguation cost c increases, the mean traversal cost also tends to increase. It is also conceivable that for higher values of c, the mean traversal cost would approach that of the zero-disambiguation path. This would occur because as c increases, the total cost of disambiguating obstacles becomes prohibitive, causing all ACS(k) algorithms to increasingly avoid disambiguation when encountering obstacles during NAVA's traversal.

5.3. Performance of NAVA *versus* number of obstacles

In this section, we explore how the ACS(k) algorithm, with $k \geq 0$, performs as the number of obstacles changes. We have seen in the simulation results in Sections 5.1 and 5.2 that the mean traversal costs tended to be higher when there are more of true obstacles in the traversal window. In settings with only false obstacles, based on insights from previous sections, ACS(1) is recommended to achieve the shortest s, t path. Conversely, in environments with true obstacles only, algorithms in the ACS(k) series with higher k values tend to be more effective. In this section, we choose $\lambda = 2$ and $c = 5$ and present only the results in the true-obstacle-only and false-obstacle-only cases in Figure 14. In the false-obstacle-only case, we present the mean traversal costs for the benchmark, and ACS(k) with $k = 1, 2$, and in the true-obstacle-case, for the benchmark, and ACS(k) with $k = 4, 5$. The reason for these choices is that for the minimum cost s, t path, the most effective ACS(k) algorithm in environments with only false obstacles employs a smaller k, while in scenarios with only true obstacles, a larger k is preferable. Notice that the corresponding ACS(k) algorithms yield mean traversal costs very similar to that of the benchmark algorithm. Specifically, if $\mathcal{X} = \mathcal{X}_F$ (false-obstacle-only), choosing $k \approx 0$ approximates the benchmark value for the mean traversal cost. Conversely, if $\mathcal{X} = \mathcal{X}_T$ (true-obstacle-only), selecting $k = 5$ or higher approximates the benchmark value for the mean traversal cost.

5.4. Convergence results for the ACS(k) algorithm

In Figures 9–11, we observe a consistent trend in which the mean traversal cost decreases as λ approaches 4, that is, as NAVA's sensor becomes effectively perfect in distinguishing true and false obstacles. In this regime, the benchmark setting is recovered, since the false-obstacle marks converge to 0 and the true-obstacle marks converge to 1.

For the convergence argument below, it is convenient to introduce a modified version of the ACS(k) penalty:

$$\mathcal{P}'_{\mathrm{ACS}}(c,p,k) := \mathcal{P}_{\mathrm{ACS}}(c,p,k) - 1 = c - 1 + (1-p)^{-k}.$$

We refer to the corresponding navigation rule as the modified ACS(k) algorithm and denote its estimated path risk and realized traversal cost by

$$\mathcal{R}'_{\mathrm{ACS}}(k,\lambda,\pi) \quad \text{and} \quad \mathcal{C}'_{\mathrm{ACS}}(k,\lambda),$$

respectively. Thus, throughout this subsection, the prime indicates the same modification: replacement of $\mathcal{P}_{\mathrm{ACS}}$ by $\mathcal{P}'_{\mathrm{ACS}}$ in the corresponding risk and cost quantities.

We also extend the path-cost convention by setting

$$\mathcal{C}(\pi,\mathcal{X}) = \infty \qquad \text{if } \pi \cap \mathcal{X}_T \neq \emptyset,$$

while noting that NAVA never selects an infinite-cost path by construction, so the realized traversal cost of NAVA remains finite.

Theorem 5.1. *Conditional on a realization of $\mathcal{X}$ with any combination of true and false obstacles (including true-obstacle-only or false-obstacle-only cases),*

(i) *For any fixed $k > 0$, the estimated risk of a path π under the modified ACS(k) algorithm converges in probability to the total cost along π as $\lambda \to 4$, that is,*

$$\mathcal{R}'_{\mathrm{ACS}}(k,\lambda,\pi) \xrightarrow{p} \mathcal{R}'_{\mathrm{ACS}}(k,4,\pi) = \mathcal{C}(\pi,X).$$

(ii) *If the path π is obstacle-free, that is, $\pi \cap X = \emptyset$, then the estimated risk under the ACS(k) algorithm is simply the Euclidean length of π, regardless of k and λ:*

$$\mathcal{R}_{\mathrm{ACS}}(k,\lambda,\pi) = L(\pi).$$

If instead $\pi \cap X \neq \emptyset$, then

$$\mathcal{R}_{\mathrm{ACS}}(k,\lambda,\pi) \xrightarrow{p} \infty \qquad \text{as } k \to \infty.$$

Proof. First we note that by properties of Beta distribution, for $X \sim \mathrm{Beta}(a,b)$, $X \xrightarrow{p} 0$ as $a \to 0$ for any $b > 0$ and $X \xrightarrow{p} 1$ as $b \to 0$ for any $a > 0$.

Recall from equation (4) that the estimated risk can be expressed as

$$\mathcal{R}_{\mathrm{ACS}'}(k,\lambda,\pi) = \mathcal{L}(\pi) + \sum_{\substack{x\in\mathcal{X}, e\in E(\pi):\\ D_r(x)\cap e\neq\emptyset}} \mathcal{P}_{\mathrm{ACS}'}(c,p(x),k).$$

(i) In this case, as $\lambda \to 4$ (for fixed $k > 0$), for an obstacle $x \in \mathcal{X}_F$, $p(x) \to 0$ in probability, which implies $\mathcal{P}_{\mathrm{ACS}'}(c,p(x),k) \to c$ in probability (by continuous mapping theorem). Similarly, for an obstacle $x \in \mathcal{X}_T$, as $\lambda \to 4$ (for fixed $k > 0$), $p(x) \to 1$ in probability, which implies $\mathcal{P}_{\mathrm{ACS}'}(c,p(x),k) \to \infty$ in probability. Therefore, for any path π, the estimated risk satisfies

$$\mathcal{R}_{\mathrm{ACS}'}(k,\lambda,\pi) \xrightarrow{p} \mathcal{C}(\pi,\mathcal{X})$$

where

$$\mathcal{C}(\pi,\mathcal{X}) = \begin{cases} \mathcal{L}(\pi) + c \times n_{\mathrm{dis}}, & \text{if } \pi \cap \mathcal{X}_T = \emptyset \\ \infty, & \text{otherwise} \end{cases}$$

where $n_{\mathrm{dis}} = |\{x \in \mathcal{X}_F, e \in E(\pi) : D_r(x) \cap e \neq \emptyset\}|$. Notice also that $\mathcal{C}(\pi,\mathcal{X}) = \mathcal{L}(\pi)$ if $\pi \cap \mathcal{X} = \emptyset$. The convergence $\mathcal{R}_{\mathrm{ACS}'}(k,\lambda,\pi) \xrightarrow{p} \mathcal{R}_{\mathrm{ACS}'}(k,\lambda=4,\pi)$ follows from continuity in λ.

(ii) In this case, for any path $\pi \cap \mathcal{X} \neq \emptyset$, as $k \to \infty$, $\mathcal{R}_{\mathrm{ACS}}(k,\lambda,\pi) \xrightarrow{p} \infty$ because $\mathcal{P}_{\mathrm{ACS}'}(c,p(x),k) \xrightarrow{p} \infty$ for any $p \in (0,1)$. And for an obstacle-free path, $\mathcal{R}_{\mathrm{ACS}}(k,\lambda,\pi) = \mathcal{L}(\pi)$ vacuously follows from the definition. □

Let C_B denote the traversal cost of NAVA under the benchmark algorithm, and let $C'_{\mathrm{ACS}}(k,\lambda)$ denote the traversal cost under the modified ACS(k) algorithm defined above. Then we have:

Theorem 5.2. *Conditional on a realization of $\mathcal{X}$ with any combination of true and false obstacles (including the true-obstacle-only and false-obstacle-only cases), for any fixed $k>0$,*

$$\mathcal{C}'_{\mathrm{ACS}}(k,\lambda) \xrightarrow{p} C_B \qquad \text{as } \lambda \to 4.$$

Proof. Recall that as $\lambda \to 4$, the sensor accuracy gets to be perfect (in probability), *i.e.*, $p_{_T} \to 1$ and $p_{_F} \to 0$ in probability. In particular, for the general case of $\mathcal{X} = \mathcal{X}_F \cup \mathcal{X}_T$, for $x \in \mathcal{X}_F$, we have $p = p_{_F} \xrightarrow{p} 0$, which implies that $\mathcal{P}_{\mathrm{ACS}'}(c,p,k) \xrightarrow{p} c+1-1 = c$ and for $x \in \mathcal{X}_T$, we have $p = p_{_T} \xrightarrow{p} 1$, which implies that $\mathcal{P}_{\mathrm{ACS}'}(c,p,k) \xrightarrow{p} \infty$. Therefore, as $\lambda \to 4$, for a given configuration of $\mathcal{X}$, the traversal cost of NAVA converges in probability to that under the benchmark algorithm. Furthermore, as $\lambda \to 4$, the estimated risk becomes same as the traversal cost of NAVA under the benchmark algorithm (since it only disambiguates false obstacles with cost c and avoids disambiguation of true obstacles). Thus, the limiting behaviour of NAVA is as in the benchmark setting with perfect knowledge of the false obstacles. □

In Theorem 5.2, the results will follow for ACS(k) (*i.e.* with no modification in the penalty function $\mathcal{P}_{\mathrm{ACS}}(c,p,k)$) when $\mathcal{X} = \mathcal{X}_T$, since in this case, as $\lambda \to 4$ (with any given $k>0$) $p = p_{_T} \xrightarrow{p} 1$ implying $\mathcal{P}_{\mathrm{ACS}'}(c,p,k) \xrightarrow{p} \infty$.

Combining our previous results and findings, we have the following corollary. In the statements below, primed quantities refer to the modified ACS(k) penalty $\mathcal{P}'_{\mathrm{ACS}}$, whereas unprimed quantities refer to the original ACS(k) penalty $\mathcal{P}_{\mathrm{ACS}}$.

Corollary 5.1. *For fixed $c>0$ and $\lambda \in [0,4]$,*

1. *when $\mathcal{X} = \mathcal{X}_F$, as $k \to 0$, $\mathcal{C}_{\mathrm{ACS}'}(k,\lambda) \xrightarrow{p} \mathcal{C}_B \overset{wp1}{=} \mathcal{C}_{\mathrm{ACS}'}(k=0,\lambda)$.*
2. *As $k \to \infty$, $\mathcal{C}_{\mathrm{ACS}}(k,\lambda) \xrightarrow{p} \mathcal{L}(\pi)$. In particular, when $\mathcal{X} = \mathcal{X}_T$, the limit is $\mathcal{L}(\pi) = \mathcal{C}_B$.*

Proof. 1. In computing the benchmark value, NAVA knows the location and status of all obstacles. In this scenario, the use of the sensor is unnecessary, and the probability of an obstacle being true is irrelevant. As $k \to 0$ in the penalty function $\mathcal{P}_{\mathrm{ACS}}(c,p,k)$, the influence of the probability of an obstacle being true gets smaller. Furthermore, when $\mathcal{X} = \mathcal{X}_F$, the estimated risk will converge in probability to the incurring traversal cost of each path when $k \to 0$. Thus, $\mathcal{C}_{\mathrm{ACS}'}(k,\lambda) \xrightarrow{p} \mathcal{C}_B$ as $k \to 0$ with any $\lambda \in [0,4]$.

The probability of an obstacle being true is irrelevant when $k=0$ in the penalty function $\mathcal{P}_{\mathrm{ACS}'}(c,p,k)$. Furthermore, when $\mathcal{X} = \mathcal{X}_F$, the estimated risk and the incurring traversal cost of each path will be equal when $k=0$. Thus, the traversal cost computed by ACS'(0) is the same as that computed by the benchmark algorithm, leading to the equality $\mathcal{C}_{\mathrm{ACS}'}(k,\lambda) = \mathcal{C}_B$ with probability 1 for $k=0$ with any $\lambda \in [0,4]$.

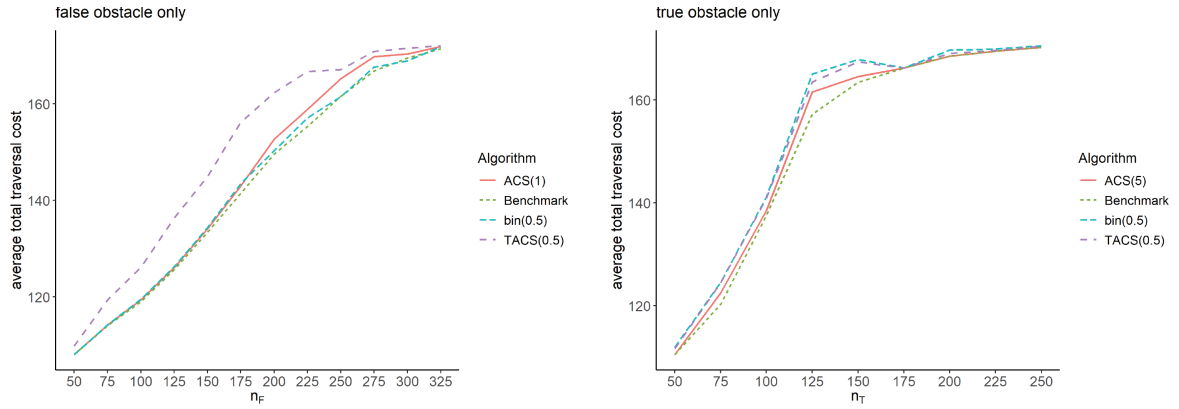
2. By Theorem 5.1 (ii), $\pi \cap \mathcal{X} \neq \emptyset$, $\mathcal{R}_{\mathrm{ACS}}(k,\lambda,\pi) \xrightarrow{p} \infty$ as $k \to \infty$. So, NAVA will tend to choose the obstacle-free path with minimum (Euclidean) length (since only the obstacle-free paths will have finite risk). But such a path will incur cost as the total length of the path. Hence, the result follows.

Furthermore, when $\mathcal{X} = \mathcal{X}_T$, the benchmark path for NAVA would be the minimum (Euclidean) length obstacle-free path. The estimated risk will be finite and converge in probability to the incurring traversal cost of each path when $k \to \infty$. Consequently, $\mathcal{C}_{\mathrm{ACS}}(k,\lambda) \xrightarrow{p} \mathcal{L}(\pi) = \mathcal{C}_B$ as $k \to \infty$. □

The above observations suggest a relationship between k and path selection efficiency, highlighting a trade-off between risk aversion and traversal cost in different obstacle environments, provided in the below conjecture.

Conjecture: Let $\mathcal{C}_{\mathrm{ACS}}(k,\lambda)$ denote the traversal length under the ACS(k) algorithm with the sensor accuracy parameter λ for the given problem setting as described in Section 4.2.2. Then, we have the following:

(a) For a fixed $k \geq 0$ and for any $0 \leq \lambda' < \lambda \leq 4$, we have $\mathbf{E}[\mathcal{C}_{\mathrm{ACS}}(k,\lambda)] \leq \mathbf{E}[\mathcal{C}_{\mathrm{ACS}}(k,\lambda')]$.
(b) For a fixed $\lambda \in [0,4]$ and for any $k > k' \geq 0$, we have $\mathbf{E}[\mathcal{C}_{\mathrm{ACS}}(k,\lambda)] \geq \mathbf{E}[\mathcal{C}_{\mathrm{ACS}}(k',\lambda]$ when $\mathcal{X} = \mathcal{X}_F$.
(c) For a fixed $\lambda \in [0,4]$ and for any $k > k' \geq 0$, we have $\mathbf{E}[\mathcal{C}_{\mathrm{ACS}}(k,\lambda)] \leq \mathbf{E}[\mathcal{C}_{\mathrm{ACS}}(k',\lambda]$ when $\mathcal{X} = \mathcal{X}_T$.



(a) (b)

FIGURE 15. (a) Mean traversal cost *versus* $n_F \in \{50, 75, \ldots, 325\}$ for benchmark, bin($\alpha = 0.5$), ACS(1), TACS($\alpha = 0.5$) algorithms. (b) Mean traversal cost *versus* $n_T \in \{50, 75, \ldots, 250\}$ for benchmark, bin($\alpha = 0.5$), ACS(5), TACS($\alpha = 0.5$) algorithms. Note that horizontal axes are differently scaled.

5.5. Threshold adjusted ACS(k) algorithms

While we have yet to develop an efficient algorithm for environments containing both true and false obstacles, the results in Theorem 5.1 suggest the creation of a new algorithm, called TACS(α) (for ACS(k) algorithm being chosen based on a *threshold* probability α). Here, $\alpha \in (0, 1)$ acts as the probability threshold determining whether an obstacle should be disambiguated. For a given problem instance, TACS(α) selects the k value in the penalty function $\mathcal{P}_{\text{ACS}}(c, p, k)$ based on the α parameter. The penalty function used by TACS(α), denoted as $\mathcal{P}_{\text{TACS}}(c, p, \alpha)$, is defined as follows:

$$\mathcal{P}_{\text{TACS}}(c, p, \alpha) = \begin{cases} \mathcal{P}_{\text{ACS}}(c, p, k = 0) = c + 1 & \text{if } p < \alpha, \\ \mathcal{P}_{\text{ACS}}(c, p, k) \text{ with } k = -\frac{\log(z_0 - c)}{\log(1-\alpha)} & \text{if } p \geq \alpha, \end{cases}$$

where z_0 represents the traversal cost of the zero-disambiguation path, as computed by the ZD algorithm, from s to t.

Alternatively, a binary threshold based on assigning obstacles as true or false based on sensor probabilities can also be proposed as:

$$\mathcal{P}_{\text{bin}}(c, p) = \begin{cases} c & \text{if } p < \alpha, \\ \infty & \text{if } p \geq \alpha. \end{cases}$$

We call the corresponding algorithm as bin(α). A natural choice of α is 0.5.

Given that the centers of obstacles are uniformly distributed within the region $[10, 90] \times [10, 90]$ and the radius r is set to 4.5, the area $[5, 95] \times [5, 95]$ becomes densely populated as the number of obstacles increases. To facilitate simpler calculations in simulations, it is practical to approximate the value of z_0 as the Euclidean distance following the path $s \to (95, 95) \to (95, 5) \to t$, which results in 180.45 units. For instance, with a disambiguation cost of $c = 5$ and a threshold $\alpha = 0.5$, we calculate k to be 7.5. NAVA, only informed of the probabilities of obstacles being true, can optimally utilize the threshold $\alpha = 0.5$. Altering this threshold, either by decreasing α or increasing α, could potentially lead to the loss of information about true or false obstacles, and hence deteriorating the algorithm's performance. Furthermore, α can be specified based on the sensor's ability to detect true or false obstacles more correctly.

We present the mean traversal cost for the benchmark, bin($\alpha = 0.5$), ACS(1), TACS($\alpha = 0.5$) algorithms in the false-obstacle-only case with $n_F \in \{50, 75, \ldots, 325\}$ in Figure 15a and for the benchmark, bin($\alpha = 0.5$), ACS(5), TACS($\alpha = 0.5$) algorithms in the true-obstacle-only case with $n_T \in \{50, 75, \ldots, 250\}$ in Figure 15b. Notice that in the false-obstacle-only case, TACS(0.5) algorithm performs the worst, TACS($\alpha = 0.5$) algorithm performs and bin(0.5) algorithm is closest to the benchmark. On the other hand, in the true-obstacle-only case, bin(0.5) algorithm is the worst performer and ACS(5) algorithm is closest to the benchmark. The performance of TACS gets better compared to bin algorithm in mixed obstacle environments with both true and false obstacles (not presented), we don't use bin algorithm in our later analysis. Nonetheless, the mean traversal cost calculated by TACS($\alpha = 0.5$) still does not align well with the benchmark value. The main drawback of the penalty function $\mathcal{P}_{\text{TACS}}(5, p, 0.5)$ is its strict distinction between true and false obstacles, due to the specified threshold $\alpha = 0.5$.

To overcome this shortcoming, we enhance the approach by incorporating a more refined partitioning of the probability values. We subdivide the interval $[0, 1]$ into m equal segments of $1/m$ units each. Define a vector

$$\mathbf{I} = (I_0, I_1, \ldots, I_m) = (0, 1/m, 2/m, \ldots, (m-1)/m, 1)$$

to represent these interval endpoints. Similarly, for a given positive integer κ, construct the vector

$$2\kappa\,\mathbf{I} = (0, 2\kappa/m, 4\kappa/m, \ldots, 2\kappa(m-1)/m, 2\kappa).$$

Considering the probabilities $p_1, p_2, \ldots, p_n$ of disk-shaped obstacles being true obstacles, let $\bar{p}$ denote their mean, calculated as

$$\bar{p} = \frac{1}{n}\sum_{j=0}^{n} p_j.$$

Let $i^* = \max\{i : I_i \le \bar{p}\}$ (*i.e.*, the maximum index of $\mathbf{I}$ with $I_i \le \bar{p}$). Then, we modify the ith entry in vector $2\kappa\,\mathbf{I}$ as follows:

$$\begin{array}{ll} 2\kappa\, I_i + (i - \frac{n}{2} - 1)\frac{2\kappa}{m} & \text{if } i \le i^*, \\ 2\kappa\, I_i + (i - \frac{n}{2} + 1)\frac{2\kappa}{m} & \text{if } i > i^*. \end{array}$$

If any entry in the modified version of $2\kappa\,\mathbf{I}$ becomes negative or exceeds 2κ due to this operation, set it to 0 or 2κ, respectively. That is, we obtain the vector $\mathbf{K} = (K_0, K_1, \ldots, K_m)$ with

$$K_i = \max(\min(2\kappa\, I_i, 2\kappa), 0) \quad \text{for } i = 1, 2, \ldots, m. \tag{8}$$

Finally, we define the new penalty function as:

$$\mathcal{P}_{\text{MTACS}}(c, p, \kappa, m) = \mathcal{P}_{\text{ACS}}(c, p, k = K_i) \quad \text{if } I_i \le p < I_{i+1}, \text{ for } i = 1, 2, \ldots, m$$

and refer to the corresponding algorithm as MTACS(κ) (for *multiple threshold* values for the probability p in the ACS(k) algorithm to choose the exponent k). We first note that our simulations (not presented) demonstrate that the restriction of k to the interval $(0, 2\kappa)$ in equation (8) is not strictly necessary, as the unrestricted versions exhibit similar performance to the restricted ones. Nonetheless, we maintain the restriction for convenience and consistency in our calculations.

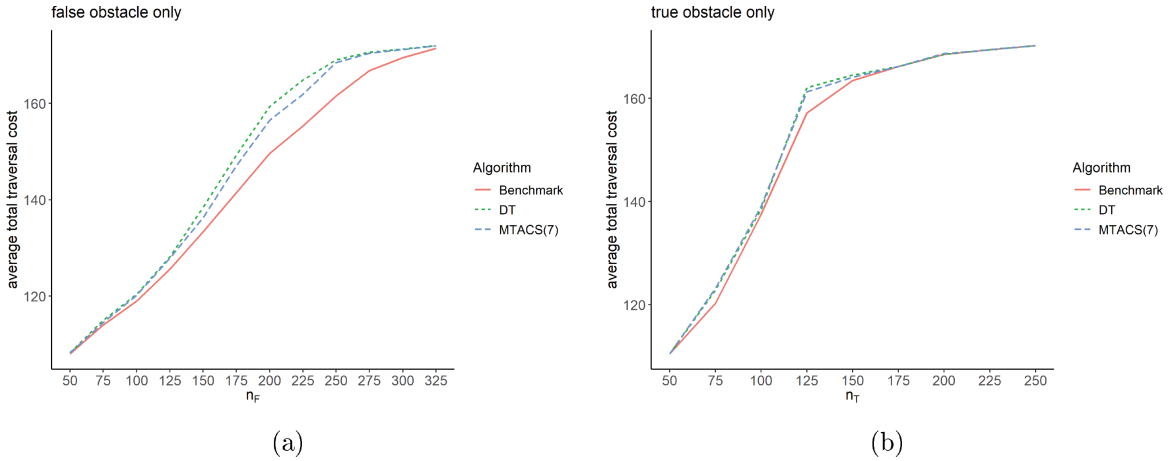
In Figure 16, we observe that the new MTACS(κ) algorithm with $k' = 7$ is better than the DT algorithm (*i.e.*, closer to the benchmark values) when there are either only false obstacles or only true obstacles. Figure 16 demonstrates that the MTACS(κ) algorithm, particularly with $\kappa = 7$, outperforms the DT algorithm in scenarios featuring exclusively true or false obstacles. In such specific cases, the benchmark value is better approximated on average by the traversal cost calculated using the $\mathcal{P}_{\text{MTACS}}(c, p, \kappa = 7)$ penalty function compared to DT algorithm.

However, in Figure 16 we observe that MTACS(7) is only slightly better than the DT algorithm. To conduct a thorough analysis of the new MTACS(κ) algorithm, we examine its performance across various combinations

Algorithm 3. The penalty function $\mathcal{P}_{\mathrm{MTACS}(\kappa)}$.

Input: $\mathcal{X} = \mathcal{X}_T \cup \mathcal{X}_F$, c: disambiguation cost, κ: range parameter for k, m: number of subintervals to partition $(0, 1)$
Output: Penalty function $\mathcal{P}_{\mathrm{MTACS}}(c, p, \kappa, m)$

$\bar{p} = \frac{1}{n}\sum_{j=1}^{n} p_j$
$\mathbf{I} := (I_0, I_1, \ldots, I_m) = \left(0, \frac{1}{m}, \ldots, 1\right)$,
$\mathbf{K} := (K_0, K_1, \ldots, K_m) = \left(0, \frac{2\kappa}{m}, \ldots, 2\kappa\right)$
$i^* = \max\{i : I_i \le \bar{p}\}$
for $i = 0$ to m **do**
  $k' \leftarrow K_i + (i - \frac{n}{2} - \mathbb{I}(i \le i^*) + \mathbb{I}(i > i^*))\frac{2\kappa}{m}$
  $k' \leftarrow \max(\min(k', 2\kappa), 0)$
  Update K_i as $K_i \leftarrow k'$
end for
Define $\mathcal{P}_{\mathrm{MTACS}}(c, p, \kappa, m)$ based on updated K for each $p \in \{p_j, j = 1, 2, \ldots, n\}$
i.e., $\mathcal{P}_{\mathrm{MTACS}}(c, p_j, \kappa) \leftarrow \mathcal{P}_{\mathrm{ACS}}(c, p_j, k = K_i)$ for $p_j \in [I_i, I_{i+1})$

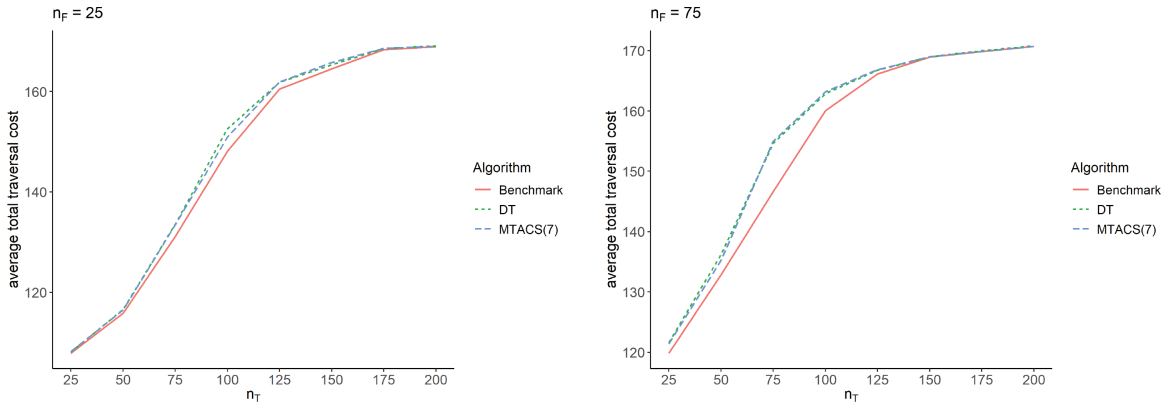


(a) (b)

FIGURE 16. (a) Mean traversal cost *versus* n_F estimated by the benchmark, DT, MTACS(7) (with $m = 20$) algorithms with $n_F \in \{50, 75, \ldots, 325\}$. (b) Mean traversal cost *versus* n_T computed by the benchmark, DT, MTACS(7) algorithms with $n_T \in \{50, 75, \ldots, 250\}$. Notice that horizontal axes are differently scaled.

of (n_F, n_T) pairs. Figure 17 shows that, on average, MTACS(κ) with $\kappa = 7$ outperforms the DT algorithm. For instance, as depicted in Figure 17a, for the pair $(n_F, n_T) = (25, 100)$, the mean traversal cost achieved by MTACS(7) is shorter than that of the DT algorithm.

The observation that the MTACS(κ) algorithm doesn't converge to the benchmark value motivates further refinement of the algorithm. Rather than splitting the interval $[0, 1]$ into $m = 20$ bins as previously done, we now divide it into $m = 100$ equal segments. For our experiments, we test $\kappa = 2$, 4, 6, 8, 10 as detailed in Algorithm 3. For each κ value, we run 100 Monte Carlo simulations for the (n_F, n_T) combinations. We average over these 100 replicates for each (n_F, n_T) pair, and the κ value leading to the minimum average total traversal cost is selected as the best κ. Thus, different (n_F, n_T) pairs may have different best κ values. To further facilitate the comparison, we also obtain the confidence interval of the total traversal cost associated with the selected κ value for each (n_F, n_T) pair. The results presented in Figure 18 reveal that the mean traversal cost calculated by this improved MTACS(κ) algorithm aligns closely with the benchmark value, since at almost all (n_F, n_T) combinations mean traversal by MTACS algorithm is not significantly different from that of the benchmark algorithm. This result is suggestive that the choice of κ (and hence indirectly the choice of k) should



(a) (b)

FIGURE 17. Comparison of mean traversal costs estimated by the algorithms benchmark, DT, MTACS(7) (with $m = 20$), (a) for $n_F = 25$ with $n_T \in \{25, 50, \ldots, 200\}$, (b) and for $n_F = 75$ with $n_T \in \{25, \ldots, 200\}$.

incorporate the probability marks of the obstacles. Identifying the most effective κ value for achieving an even more precise benchmark approximation is an area for future research.

Remark 5.3. In any obstacle combination setting, the benchmark value can often be closely approximated by the ACS(k) algorithm also, provided an appropriately chosen (optimal) value of k is chosen. If the number of false and true obstacles are known, together with the sensor accuracy λ, one can perform simulations for a grid of k values (say ranging from 0 to 10 across with 50 or 100 Monte Carlo replications). Then for the particular combination of (n_F, n_T), the most effective $\mathcal{P}_{\mathrm{ACS}}(c, p, k)$ penalty function to approximate the mean traversal cost of the benchmark algorithm can be estimated. In some instances, the results (not presented) closely matched the mean benchmark values, with the largest deviation observed showed an average heuristic traversal cost with a percentage error under 1.5%. However, all this hinges on the possibly unrealistic assumption of knowing the actual numbers of true and false obstacles, together with λ value which is usually unavailable *a priori*.

6. Discussion and conclusions

The current greedy algorithms for NAVA utilize a heuristic incorporating penalty functions for disambiguation of the obstacles. We proposed new penalty function families, with a goal to enhance and approximate the currently used penalty functions, and hence study and understand them better. Specifically, for any given disambiguation cost c, the penalty functions of the RD, AP, and DT algorithms – $\mathcal{P}_{\mathrm{RD}}(c, p)$, $\mathcal{P}_{\mathrm{AP}}(c, p)$, and $\mathcal{P}_{\mathrm{DT}}(c, p, d_t)$ – can be effectively approximated by the penalty function $\mathcal{P}_{\mathrm{ACS}}(c, p, k)$ with respective optimal k values (see Tab. 2). Moreover, the ACS(k) algorithm which computes heuristic traversal costs based on $\mathcal{P}_{\mathrm{ACS}}(c, p, k)$ penalty function, can approximate the mean traversal costs realized by RD, AP, and DT algorithms, although optimal k for the penalty approximation and traversal cost approximation may not be identical (see Prop. 4.2 and Rem. 4.1). As k increases beyond 5, the penalty function $\mathcal{P}_{\mathrm{ACS}}(c, p, k)$ tends to assign higher edge weights, leading ACS(k) to adopt a more risk-averse approach, minimizing the number of disambiguations during NAVA's traversal. This characteristic aligns with the behavior of the DT algorithm, which, unlike RD and AP, predominantly avoids disambiguations when encountering true obstacles, thereby favoring a zero-disambiguation path or paths with minimal disambiguations, as illustrated in Figure 7. Our Monte Carlo simulations suggest that in special circumstances such as the true-obstacle-only or false-obstacle-only cases the benchmark value (computed by the algorithm B) can be approximated by choosing the larger or smaller k values, respectively.

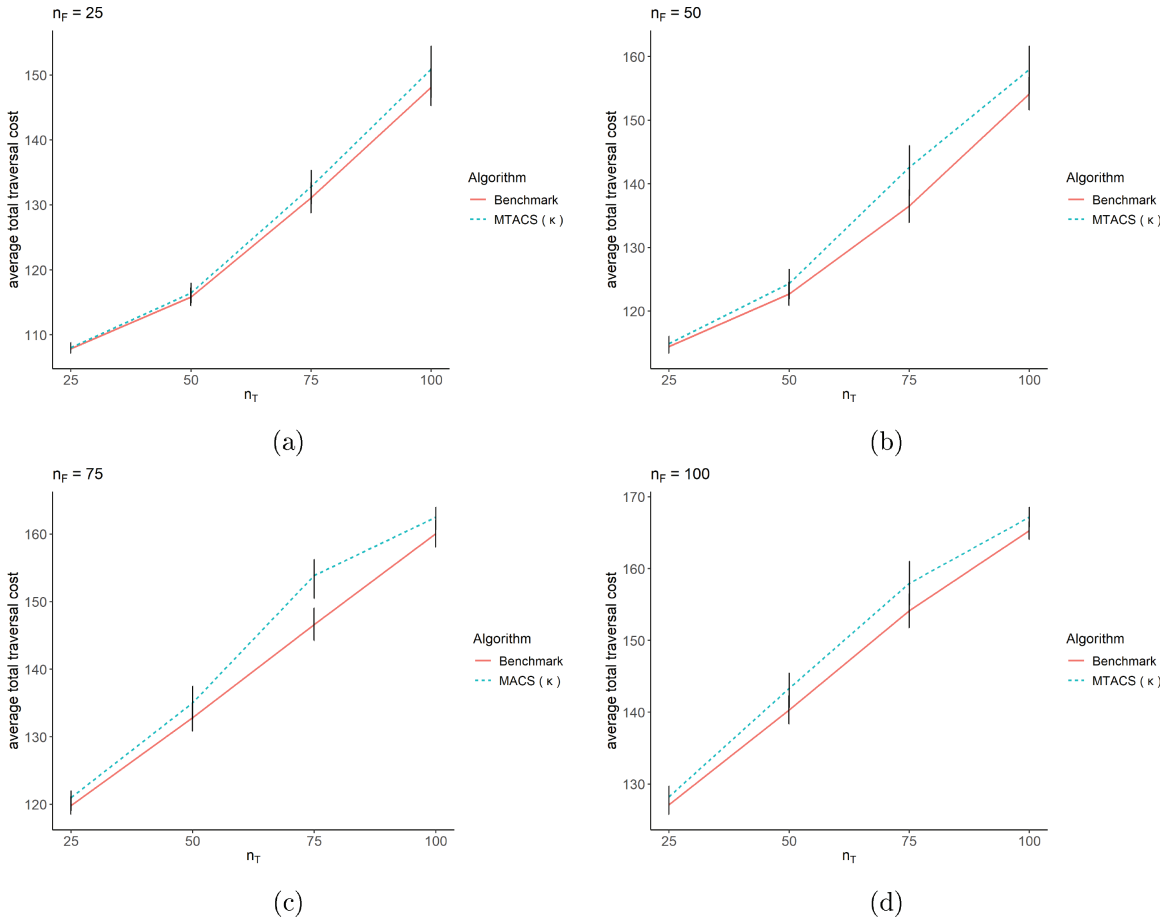


FIGURE 18. Comparison of mean traversal costs estimated by the benchmark and MTACS(κ) algorithms (with $m = 100$) for (a) $n_F = 25$, (b) $n_F = 50$, (c) $n_F = 75$, and (d) $n_F = 100$ with $n_T = 25, 50, 75, 100$. The 95% confidence intervals for the benchmark and MTACS mean traversal estimates are also provided. Optimal κ values are used for each (n_F, n_T) pair (as described in the text).

The performance evaluation in this paper relies primarily on Monte Carlo simulation. This is inherent to the SOS/CTP setting: outside of special graph topologies, global optimality characterizations remain unavailable, and traversal cost depends on coupled stochastic elements including obstacle placement, sensor marks, and dynamic rerouting decisions. In such settings, analytical comparison of heuristic policies is generally intractable. Our conclusions should therefore be interpreted as empirical but structurally motivated. The simulation study is designed to isolate the effect of penalty growth behavior under controlled variations of sensor accuracy, obstacle density, and disambiguation cost. While numerical performance levels depend on parameter choices, several qualitative observations remain robust across regimes: (i) additive penalty structures reduce over-disambiguation when c is small, (ii) steeper penalty growth favors conservative routing under high sensor uncertainty, and (iii) the ACS(k) family interpolates smoothly between risk-seeking and risk-averse behaviors. We emphasize that the proposed algorithms remain heuristic, and simulation results do not constitute optimality guarantees. Rather, they provide comparative guidance on structural penalty design under uncertainty. We also note that the relative

ordering of algorithms is stable across moderate perturbations of the Beta sensor parameters and obstacle counts, indicating that the observed performance differences are not driven solely by fine parameter tuning.

We investigate the dependence of NAVA's traversal performance on the sensor accuracy, cost of disambiguation, and also the number of obstacles. Based on our stochastic ordering results in Proposition 5.1, we see that the estimated traversal risk computed by the ACS(k) algorithm $\mathcal{R}_{\mathrm{ACS}}(k, \lambda, \mathcal{X})$ tends to be smaller for higher accuracy of NAVA's sensor (*i.e.*, as λ increases), and as k increases (*i.e.*, as the penalty for disambiguation increases), the estimated risk tends to be larger. The disambiguation cost (c) significantly influences the mean traversal cost derived from various heuristic algorithms. It's noted that an increase in c leads to an increase in the mean traversal cost, continuing until the point where the algorithm opts for a zero-disambiguation path, avoiding disambiguations entirely. With higher disambiguation costs, NAVA tends to perform fewer disambiguations, which result in longer traversal paths and also the disambiguations incur higher costs, increasing the total traversal cost. A similar trend occurs when the number of obstacles gets larger. We also provide convergence results as sensor accuracy tends to be perfect (*i.e.*, as $\lambda \to 4$) and as $k \to \infty$.

To get closer to the benchmark algorithm's performance (which is the ideal in our setting), we consider various adaptations of the ACS(k) algorithm. The first adaptation we consider is a disambiguation strategy based on a threshold parameter α. Here, disambiguation of a disk-shaped obstacle is performed if its probability of being a true obstacle p is less than α; otherwise, the obstacle is bypassed without risk. Utilizing this concept, we defined the penalty function $\mathcal{P}_{\mathrm{TACS}}(c, p, \alpha)$ and demonstrated performance of TACS($\alpha = 0.5$) algorithm (using the $\mathcal{P}_{\mathrm{TACS}}(5, p, 0.5)$ penalty function) depends on the composition of the obstacles (with more true obstacles, we observe shorter traversal costs, and with more false obstacles, the mean traversal cost tends to get higher compared to ACS(k) algorithm. A key limitation of $\mathcal{P}_{\mathrm{TACS}}(5, p, 0.5)$ is its rigid differentiation between true and false obstacles based on the set threshold $\alpha = 0.5$. In the specific problem context discussed in Section 4.2.2, altering the threshold α – either by decreasing or increasing it – could result in the loss of important information about either true or false obstacles. Such changes have the potential to negatively impact the algorithm's stability and performance. To address this limitation, we introduce a finer partitioning of the probability marks p and use different k values for each partition grid for p. This led to the creation of a new penalty function $\mathcal{P}_{\mathrm{MTACS}}(c, p, \kappa)$ and the corresponding MTACS(κ) algorithm (see Algorithm 3). Our findings demonstrate that, in many instances, the benchmark value can be approximated very closely by employing the MTACS(κ) algorithm with suitable choices of κ (see Fig. 18).

Based on our simulations, we suggest the use of ACS(k) algorithm if the number of false and true obstacles and the obstacle patterns are known. Because in this case, one can perform pilot Monte Carlo simulations to determine the optimal k for this setting. On the other hand, if the number of false and true obstacles and the obstacle patterns are not known (which is more realistic), one can employ the MTACS(κ) algorithm with $m = 100$.

In our future research, we aim to improve our algorithms or introduce new algorithms to achieve even more accurate approximations of the benchmark value. These heuristic algorithms, which have been developed and examined within the context of our problem setting, possess broader applicability. In particular, they can be applied to settings including, but not limited to, the Canadian traveler's problem [19], the discrete SOS problem [2], the discretized OPD [4], the repeated-task CTP [11], and the CTP-Tree [16]. Additionally, we intend to explore the field of competitive analysis of heuristic algorithms, where the ratio of heuristic performance to worst-case performance is investigated [12]. Previous works, such as those by Westphal [23] and Xu *et al.* [25], have delved into competitive analyses related to our problem. We plan to conduct similar analyses for existing algorithms like RD, AP, and DT, as well as for our heuristic algorithms ACS(k) and MTACS(κ). This will further our understanding of their competitiveness and performance in various scenarios.

Funding

Most of the Monte Carlo simulations presented in this article were executed at Easley HPC Laboratory of Auburn University. LZ and EC were supported by Office of Naval Research Grant N00014-22-1-2572 and EC were supported by NSF Award # 2319157.

Data availability statement

The R code used to generate the synthetic obstacle environments and to implement the penalty-based navigation algorithms (RD, AP, DT, ACS(k), TACS, MTACS), together with scripts supporting the Monte Carlo experiments reported in this paper, is publicly available at the GitHub repository Zhou [28].

References

[1] V. Aksakalli, The BAO* algorithm for stochastic shortest path problem with dynamic learning, in Proceedings of IEEE Conference on Decision and Control, New Orleans, LA (2007).

[2] V. Aksakalli, *Protocols for stochastic shortest path problems with dynamic learning*. Ph.D. thesis, The Johns Hopkins University (2007).

[3] V. Aksakalli and I. Ari, Penalty-based algorithms for stochastic obstacle scene problem. *INFORMS J. Comput.* **26** (2013) 370–384.

[4] V. Aksakalli and E. Ceyhan, Optimal obstacle placement with disambiguations. *Ann. Appl. Stat.* **6** (2012) 1730–1774.

[5] V. Aksakalli, D.E. Fishkind, C.E. Priebe and X. Ye, The reset disambiguation policy for navigating stochastic obstacle fields. *Nav. Res. Logist.* **58** (2011) 389–399.

[6] V. Aksakalli, O.F. Sahin and I. Ari, An AO* based exact algorithm for the Canadian traveler problem. *INFORMS J. Comput.* **28** (2016) 96–111.

[7] A.F. Alkaya, V. Aksakalli and C.E. Priebe, A penalty search algorithm for the obstacle neutralization problem. *Comput. Oper. Res.* **53** (2015) 165–175.

[8] A. Bar-Noy and B. Schieber, The Canadian traveller problem, in Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA (1991) 261– 270.

[9] Z. Bnaya, A. Felner, E. Shimony, G.A. Kaminka and E. Merdler, A fresh look at sensor-based navigation: navigation with sensing costs, in AAAI 2008 Workshop on Search in Artificial Intelligence and Robotics, Chicago, IL (2008) 11–17.

[10] Z. Bnaya, A. Felner and S.E. Shimony, Canadian traveler problem with remote sensing, in Proceedings of the International Joint Conference on Artificial Intelligence, Pasadena, CA (2009) 437–442.

[11] Z. Bnaya, A. Felner, D. Fried, O. Maksin and S.E. Shimony, Repeated-task Canadian traveler problem. *AI Commun.* **28** (2015) 453–477.

[12] A. Borodin and R. El-Yaniv, Online Computation and Competitive Analysis. Cambridge University Press, New York (1998).

[13] R.H. Byrd, P. Lu, J. Nocedal and C. Zhu, A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.* **16** (1995) 1190–1208.

[14] E.W. Dijkstra, A note on two problems in connexion with graphs. *Numer. Math.* **1** (1959) 269–271.

[15] D.E. Fishkind, C.E. Priebe, K. Giles, L.N. Smith and V. Aksakalli, Disambiguation protocols based on risk simulation. *IEEE Trans. Syst. Man Cybern. Syst.* **37** (2007) 814–823.

[16] D. Fried, *Theoretical aspects of the generalized Canadian traveler problem*. Ph.D. thesis, Ben-Gurion University of the Negev, Beersheba, Israel (2013).

[17] E. Nikolova and D.R. Karger, Route planning under uncertainty: the Canadian traveller problem, in The 23rd AAAI Conference on Artificial Intelligence, Chicago, IL (2008).

[18] J. Nocedal and S.J. Wright, Numerical Optimization, 2nd edition. Springer, Berlin (2006).

[19] C.H. Papadimitriou and M. Yannakakis, Shortest paths without a map. *Theor. Comput. Sci.* **84** (1991) 127–150.

[20] C.E. Priebe, D.E. Fishkind, L. Abrams and C.D. Piatko, Random disambiguation paths for traversing a mapped hazard field. *Nav. Res. Logist.* **52** (2005) 285–292.

[21] S.J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th edition. Pearson, Boston (2020).

[22] D. West, Introduction to Graph Theory, 2nd edition. Prentice Hall, NJ (2001).

[23] S. Westphal, A note on the k-Canadian traveller problem. *Inform. Process. Lett.* **106** (2008) 87–89.

[24] N. Witherspoon, J. Holloway, K. Davis, R. Miller and A. Dubey, The Coastal battlefield reconnaissance and analysis (COBRA) program for minefield detection, in Proceedings of SPIE: Detection Technologies for Mines and Minelike Targets, Vol. 2496 (1995) 500–508.

[25] Y. Xu, M. Hu, B. Su, B. Zhu and Z. Zhu, The Canadian traveler's problem and its competitive analysis. *J. Comb. Optim.* **18** (2009) 195–205.

[26] X. Ye and C.E. Priebe, A graph-search based navigation algorithm for traversing a potentially hazardous area with disambiguation. *Int. J. Oper. Res. Inf. Syst.* **1** (2010) 14–27.
[27] X. Ye, D.E. Fishkind, L. Abrams and C.E. Priebe, Sensor information monotonicity in disambiguation protocols. *J. Oper. Res. Soc.* **62** (2011) 142–151.
[28] L. Zhou, SOS-pathfinding: R code for penalty-based navigation algorithms in the discretized stochastic obstacle scene (SOS) problem (2026). https://github.com/Li-Z27/sos-pathfinding, Accessed 2026-03-26.

Appendix A. Decision-theoretic formulation of the executed walk

In this section, we adopt a decision-theoretic approach to express NAVA's traversal cost as opposed to the ones in equations (5) and (6). This approach involves making sequential decisions (actions) based on the current understanding of the environment (obstacle disambiguation and traversal choices) and adapting with new information acquired during navigation.

Let $\Pi(v,t)$ denote the sets of all possible paths from a vertex v to the target t,

$$\Pi(v,t) = \{\pi(v,t) : \text{ a path from } v \text{ to } t \}.$$

Define the optimal path (*i.e.*, the path with the minimum estimated risk) from v to t as

$$\pi^*(v,t) = \arg\min_{\pi\in\Pi(v,t)} \mathcal{R}(\pi).$$

However, as described before, the realized or observed walk, denoted $\omega_{\text{obs}}(v,t)$, might differ from the optimal estimated path. Let the first edge in this walk be represented by $e_1(\omega)$.

The traversal process can be more practically described through a sequence of actions, $\mathcal{A}$, as follows:

$$\mathcal{A} = (a_1, a_2, \ldots, a_n), \text{ with } a_i \in \{\text{traversal, disambiguation}\},$$

where each a_i is the action taken by NAVA at step i. We can further refine the disambiguation action as disambiguation of an obstacle to be true or false, *i.e.*,

$$a_i = \begin{cases} e, & \text{if the action is traversal over edge } e \\ d^T, & \text{if the action is disambiguation of an obstacle to be true} \\ d^F, & \text{if the action is disambiguation of an obstacle to be false.} \end{cases}$$

The sequence of edges in the observed walk $\omega_{\text{obs}}(v,t)$ is given by $(e_1, e_2, \ldots, e_{k_{\text{obs}}-1})$, corresponding to the vertex sequence $(v_1 = s, v_2, \ldots, v_{k_{\text{obs}}} = t)$ such that $e_i = v_i v_{i+1}$ for $i = 1, 2, \ldots, k_{\text{obs}} - 1$. Thus, $k_{\text{obs}} = \sum_{i=1}^{n} I(a_i = e)$.

Here, each traversal action occurs over an edge $e = uv$ and each disambiguation occurs at a vertex u if an edge v is adjacent to u and v the first edge in the optimal path $\pi^*(u,t)$ (as found by the A^* navigation algorithm). Thus, each action is associated with an edge and we can associate an edge sequence, denoted $\mathcal{E}$, to the action sequence as:

$$\mathcal{A} = (a_1, a_2, \ldots, a_n) \quad \text{and} \quad \mathcal{E} = (e_1^a, e_2^a, \ldots, e_n^a).$$

So,

if $a_i = e$, then $e_i^a = e_j$ for some $j = 1, 2, \ldots, k_{\text{obs}} - 1$,
if $a_i = d^F$, then $e_i^a = e_{i+1}^a = e_j$ for some $j = 1, 2, \ldots, k_{\text{obs}} - 1$, and
if $a_i = d^T$, then $e_i^a \neq e_j$ for all $j = 1, 2, \ldots, k_{\text{obs}} - 1$.

Sequences of indices for traversal and disambiguation actions are defined as $I_e = \{i : a_i = e\}$, $I_d = \{i : a_i = \text{disambiguation}\}$, $I_d^F = \{i : a_i = d^F\}$, and $I_d^T = \{i : a_i = d^T\}$.

Then the sequence of edges constituting the observed walk from vertex v to the target t is

$$\omega_{\text{obs}}(v, t) = (e_1, e_2, \ldots, e_{k_{\text{obs}}-1}) = \left(e^a_{i_{(1)}}, e^a_{i_{(2)}}, \ldots, e^a_{i_{(k_{\text{obs}}-1)}}\right)$$

where $e^a_{i_{(j)}}$ is the j-th entry (or index) in I_e, identified through the observed action sequence $\mathcal{A}_{\text{obs}}$. Because of the way the navigation algorithms work,

$$e_i = e_1(\pi^*_{no}(v_i, t))$$

that is, e_i is the first edge on a path starting at v_i ending at t with the minimum estimated risk and not crossing a previously disambiguated true obstacle. More precisely,

$$\pi^*_{no}(v_i, t) = \arg\min_{\pi \in \Pi_{no}(v_i, t)} \mathcal{R}(\pi)$$

where

$$\Pi_{no}(v_i, t) = \{\pi \in \Pi(v_i, t) : \ E(\pi) \cap E_i^T = \emptyset\}$$

denotes the set of all paths from v_i to t that do not intersect with any edge identified as part of E_i^T, where $E_i^T = \{e_j^a : a_j = d^T, \ j = 1, 2, \ldots, i', \text{ and } e_{i'}^a = e_i\}$ denotes the set of edges that have been identified through disambiguation actions as intersecting true obstacles up to the i-th step or action. Letting $\mathcal{C}(\mathcal{A}_{\text{obs}}, \mathcal{X})$ be the cost associated with the action sequence $\mathcal{A}_{\text{obs}}$. Then

$$\begin{aligned}\mathcal{C}(\mathcal{A}_{\text{obs}}, \mathcal{X}) &= \sum_{i=1}^{n} \sum_{j=1}^{k_{\text{obs}}-1} \ell(e_j) \ I(a_i = e) + \sum_{i=1}^{n} c \times I(a_i = d) \\ &= \sum_{j=1}^{k_{\text{obs}}-1} \ell(e_j) + \sum_{i=1}^{n} c \times I(a_i = d) = \sum_{e \in E(\omega_{\text{obs}})} \ell(e) + c \times |I_d|,\end{aligned}$$

where $|I_d|$ stands for length of (the index vector) I_d. Note that for $i \in I_d^T$, $e_1(\pi^*(v_{i^-}, t)) = e_i^a$ and $e_i^a \cap \mathcal{X}_T \neq \emptyset$, where i^- is the rank (*i.e.*, order index) of $\max\{j : a_j = e_j, \ j = 1, 2, \ldots, i\}$ in I_e. We also observe that

$$\{e_i^a : \ i \in I_d^F\} \subseteq \{e_i^a : \ i \in I_e\} \ \text{ and } \ \{e_i^a : \ i \in I_d^T\} \cap \{e_i^a : \ i \in I_e\} = \emptyset.$$

In the true-obstacle-only case (*i.e.*, $\mathcal{X} = \mathcal{X}_T$), we have $I_d^F = \emptyset$, so $a_i \in \{e, d^T\}$ and $I_d = I_d^T$. Also, we have $e_i = e_1(\pi^*_{no}(v_i, t))$ as above and

$$\mathcal{C}(\mathcal{A}_{\text{obs}}, \mathcal{X}_T) = \sum_{e \in E(\omega_{\text{obs}})} \ell(e) + c \times \left|I_d^T\right|.$$

The above formulation simplifies when there are only false obstacles, or no obstacles in the traversal window.

False-obstacle-only case: $\mathcal{X} = \mathcal{X}_F$

In this case, $I_d^T = \emptyset$, so $a_i \in \{e, d^F\}$ and $I_d = I_d^F$. Also, we have $e_i = e_1(\pi^*(v_i, t))$ leading to $\omega_{\text{obs}} = \pi^*$. Then,

$$\begin{aligned}\mathcal{C}(\mathcal{A}_{\text{obs}}, \mathcal{X}_F) &= \sum_{j=1}^{k_{\text{obs}}-1} \ell(e_j) + \sum_{i=1}^{n} c \times I(a_i = d^F) = \sum_{j=1}^{k_{\text{obs}}-1} \ell(e_j) + \sum_{j=1}^{k_{\text{obs}}-1} c \times I(e_j \cap \mathcal{X}_F \neq \emptyset) \\ &= \sum_{j=1}^{k_{\text{obs}}-1} (\ell(e_j) + c \times I(e_j \cap \mathcal{X}_F \neq \emptyset)) = \sum_{e \in E(\pi^*)} \ell(e) + c \times \left|I_d^F\right|.\end{aligned}$$

Observe that $\mathcal{C}(\mathcal{A}_{\text{obs}}, \mathcal{X}_F) = \mathcal{C}_F$ (see Eq. 6).

No obstacle case: $\mathcal{X} = \emptyset$

In this case, $I_d = \emptyset$, so $a_i = e$ for all i and $e_i = e_1(\pi^*(v_i, t))$ where $\pi^*(v_i, t)$ is the path of minimum Euclidean length from v_i to t. This implies $\omega_{\text{obs}} = \pi^*$ which is the minimum length path from s to t Thus,

$$\mathcal{C}(\mathcal{A}_{\text{obs}}, \mathcal{X} = \emptyset) = \sum_{j=1}^{k_{\text{obs}}-1} \ell(e_j).$$

Here, we see that $\mathcal{C}(\mathcal{A}_{\text{obs}}, \mathcal{X} = \emptyset) = \ell^*_{ij}$ which is the sum of lengths of edges in π^* (see Eq. 6).