Sensor-Based Planning with the Freespace Assumption[*]

Sven Koenig Yury Smirnov
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213-3891
{skoenig, smir}@cs.cmu.edu

Abstract

A popular technique for getting to a goal location in unknown terrain is planning with the freespace assumption. The robot assumes that the terrain is clear unless it knows otherwise. It always plans a shortest path to the goal location and re-plans whenever it detects an obstacle that blocks its path or, more generally, when it detects that its current path is no longer optimal. It has been unknown whether this sensor-based planning approach is worst-case optimal, given the lack of initial knowledge about the terrain. We demonstrate that planning with the freespace assumption can make good performance guarantees on some restricted graph topologies (such as grids) but is not worst-case optimal in general. For situations in which its performance guarantee is insufficient, we also describe an algorithm, called Basic-VECA, that exhibits good average-case performance and provides performance guarantees that are optimal up to a constant (user-defined) factor.

1 Introduction

In this paper, we analyze a navigation problem in which a robot has to navigate from a given start location to a given goal location in unknown terrain. This path planning problem is complicated by the fact that the sensors on-board a robot can typically sense the environment only near its current position, and thus the robot has to interleave planning with movement to be able sense to its environment. As the robot moves, it acquires more knowledge about the terrain and consequently reduces its uncertainty about the environment, which also reduces the number of obstacle configurations that the planner has to consider. Thus, sensing during plan execution and using the acquired knowledge for re-planning, often called "sensor-based planning" [1], makes the planning problem tractable. A popular technique for sensor-based planning is planning with the freespace assumption [2] [8] [11] [14]: The robot assumes that the terrain is clear unless it knows otherwise. It always plans a shortest path from its current location to the goal location. When it detects an obstacle that blocks its path or, more generally, when it detects that its current path is no longer optimal, it re-plans a shortest path from its current location to the goal location using its knowledge about all obstacles encountered so far. Planning with the freespace assumption has been used on both grids and visibility graphs, but, different from other sensor-based planning algorithms [6], it was unknown which performance guarantees it provides. While we care mostly about its average-case performance, it is also important that it provides a good performance guarantee because only then one can be certain that it performs well in all environments. We show that planning with the freespace assumption provides good performance guarantees on some restricted graph topologies (such as grids) but is not worst-case optimal in general. For situations in which its performance guarantee is insufficient, we also describe an algorithm, called Basic-VECA, that exhibits good average-case performance and provides performance guarantees that are optimal up to a constant (user-defined) factor.

2 The Navigation Problem

We formalize the sensor-based planning problem as follows: A robot is given an undirected, finite graph ("map") with positive edge ("street") lengths and vertices ("junctions") that are either blocked or unblocked (by, say, "construction sites"). The status of the vertices does not change over time. One unblocked vertex is labeled the starting vertex; one vertex is labeled the goal vertex. The robot can always move from its current vertex to any unblocked neighboring vertex. Initially, it does not know the status of any vertex, but it always senses the status of its neighboring ver-

[*]Thanks to Illah Nourbakhsh and Anthony Stentz for providing detailed comments on our ideas. Thanks to Leonard Schulman for detecting a problem with one of our previous examples used to demonstrate that planning with the freespace assumption is not worst-case optimal. Thanks also to Howie Choset, Geoffrey Gordon, Andrew Moore, Reid Simmons, and Manuela Veloso for helpful discussions. This research is sponsored in part by the Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, and ARPA under grant number F33615-93-1-1330, and NSF under grant number IRI-9502548. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations or the U.S. government.

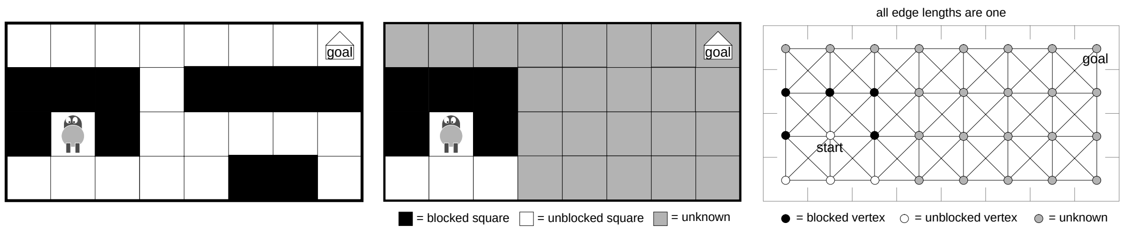

Figure 1: Outdoor Navigation Problem

tices. The robot is started at the starting vertex and has to either move to the goal vertex or recognize that this is impossible. In the literature, this problem has been studied in the context of actual robot navigation problems. For example, the purpose of NAVLAB II, Carnegie Mellon's unmanned robot HMMWV, is to reach specified coordinates in unmapped static terrains [13]. To do so, the vehicle discretizes the unknown area into a coarse-resolution map of square cells. Each cell is either traversable or untraversable. The vehicle always occupies exactly one cell and can move in all eight compass directions to traversable adjacent cells. Its sensors always detect which of its eight adjacent cells are traversable, and integrate all new information into the map. Figure 1 (center) shows the initial knowledge that NAVLAB II has when it operates in the (unrealistic, but simple) terrain of Figure 1 (left). Figure 1 (right) shows the corresponding traversability graph.[1] Other robot navigation problems have been modeled in similar ways, for example the indoor navigation problem described in [8].

3 The Freespace Assumption

Planning with the freespace assumption is a greedy way for solving sensor-based planning problems: The robot always makes the optimistic assumption that vertices are unblocked if it does not know their status. It uses this assumption to plan a shortest potentially traversable path (a path that does not contain vertices that are known to be blocked) from its current vertex to the goal vertex and traverses it until it learns about a blocked vertex on the path. At this point, it repeats the procedure, taking into account its knowledge about which vertices are blocked. If it reaches the goal vertex, its stops and reports success. If, at any point in time, it fails to find a traversable path from its current vertex to the goal vertex, it stops and reports that the goal vertex cannot be reached.

Theorem 1 *Planning with the freespace assumption is correct.*

Proof: Every time the robot cannot follow a planned path, it has learned about at least one additional blocked vertex. There are only a finite number of them, implying that planning with the freespace assumption terminates in finite time. Planning with the freespace assumption reports success only if it is at the goal vertex and has thus solved the sensor-based planning problem. It reports failure only if no traversable path from its current vertex to the goal vertex exists. Since there is a traversable path from its current vertex to the starting vertex, there is no traversable path from the starting vertex to the goal vertex either. Consequently, reaching the goal vertex is impossible in this case. ■

Planning with the freespace assumption has been used on actual robots. For example, it has been applied to the outdoor [13] and indoor [8] navigation problems mentioned above. Planning with the freespace assumption has several advantages: It is easy to implement. Its computations can be done efficiently. (The most time-consuming step is to re-calculate a shortest path after new knowledge about obstacles has been acquired and the Dynamic A* algorithm [10] does this without unnecessary re-calculations.) It immediately takes advantage of newly acquired knowledge and always uses all of its knowledge. Without any changes to the algorithm, it is able to use prior knowledge about blocked vertices. It learns an optimal trajectory over multiple trials with the same starting and goal vertices, since the freespace assumption encourages the exploration of vertices with unknown status. Finally, when it is used in conjunction with grids (as in the outdoor navigation problem), it does not need to make assumptions about the shapes of obstacles.

[1]We assume, for clarity purposes, that the robot is omnidirectional, point-sized, equipped with only a radial short-distance sensor, and capable of error-free motion. It is possible to drop these assumptions, but the example and proofs would get more complicated. Also, NABLAB II does not only distinguish between traversable and untraversable cells, but differentiates more fine-grained by assigning traversal costs to them. Re-planning occurs whenever a new traversal cost is assigned to a cell. Our description is a special case of this approach that distinguishes only two traversal costs, one of which is infinite. This does not affect our conclusions, because a sensor-based planning algorithm that is not worst-case optimal for a special case is not worst-case optimal in general either.

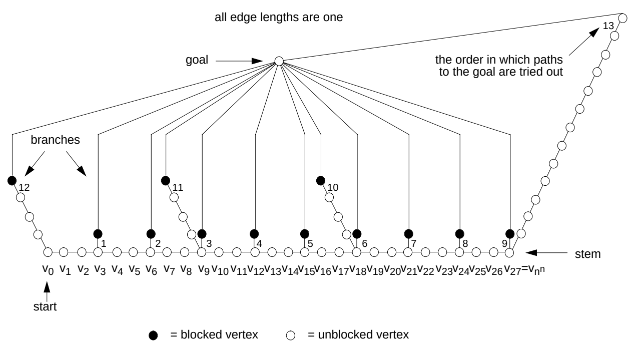
all edge lengths are one
goal
the order in which paths to the goal are tried out
13
branches
12
11
10
1
2
3
4
5
6
7
8
9
stem
v0 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15 v16 v17 v18 v19 v20 v21 v22 v23 v24 v25 v26 v27=vnn
start
● = blocked vertex
○ = unblocked vertex

Figure 2: Graph G_1 for $n = 3$

4 Performance Guarantees

We measure the performance of sensor-based planning algorithms using the total distance that the robot has to travel before it reaches the goal vertex or discovers that this is impossible. This is reasonable because robots move so slowly that the total problem solving time is completely dominated by the travel distance. The performance guarantee of an algorithm is optimal according to the standard notion of optimality used in computer science theory iff, for any problem size, the worst-case performance of the algorithm over all problems of this size (here: sensor-based planning problems on graphs with the same weight) is no worse than the worst-case performance of any other algorithm over the same problems. To derive lower and upper bounds on the performance guarantee of planning with the freespace assumption in the worst case (its performance guarantee) we need the following notation: $weight(G)$ denotes the weight of graph $G = (V, E)$ (the sum of the lengths of all its edges), and $dist_G(v_1, v_2)$ denotes the length of a shortest path between $v_1 \in V$ and $v_2 \in V$ in G.

4.1 Lower Performance Bounds

Lower bounds on the performance guarantee of planning with the freespace assumption can be established by example. Our graph topology is planar since maps often have this property. It makes Basic-VECA traverse the same path repeatedly forward and backward, and this travel distance is large compared to the lengths of the edges that are necessary to mislead planning with the freespace assumption.[2]

Theorem 2 *The performance guarantee of planning with the freespace assumption on graphs $G = (V, E)$ is $\Omega(\frac{\log |V|}{\log \log |V|}\, weight(G))$.*

[2]We have assumed that the robot is only able to sense the status of its neighboring vertices. The example can easily be adapted to sensors with larger lookaheads, say of x vertices, by replacing each edge with x consecutive edges that are connected via $x - 1$ unblocked intermediate vertices.

Proof: Graph $G_1 = (V_1, E_1)$ (Figure 2) consists of a stem with several branches that connect the stem to the goal vertex. All edge lengths are one. The stem has length n^n for some integer $n \geq 1$ and consists of the vertices $v_0, v_1, \ldots, v_{n^n}$. For each integer i with $1 \leq i \leq n$ there are n^{n-i} branches of length $1 + \sum_{j=0}^{i-1} n^j$ each. These branches attach to the stem at the vertices $v_{j\, n^i}$ for integers j; if i is even, then $0 \leq j \leq n^{n-i} - 1$, otherwise $1 \leq j \leq n^{n-i}$. All vertices not directly connected to the goal vertex are unblocked, and so are the goal vertex and the vertex on the longest branch that is directly connected to the goal vertex. All other vertices are blocked. v_0 is the starting vertex. Planning with the freespace assumption can behave as follows: It starts at v_0 and traverses the whole stem, trying to use the branches of length 2 to get to the goal vertex, only to discover that they are untraversable. It then switches directions and travels along the whole stem in the opposite direction, this time trying to use the branches of length $n+2$ to get to the goal vertex, and so forth, switching directions repeatedly. It succeeds when it finally attempts to use the longest branch. To summarize, the edges connected to the goal vertex are tried out in the order indicated in Figure 2. The total travel distance is $\Omega(n^{n+1})$ since the stem of length n^n is traversed n times. $n = \Omega(\frac{\log |V_1|}{\log \log |V_1|})$ since $weight(G_1) = \Theta(|V_1|)$ and $weight(G_1) = |E_1| = \Theta(n^n)$ [4]. Put together, it follows that the total travel distance is $\Omega(n^{n+1}) = \Omega(n\, weight(G_1)) = \Omega(\frac{\log |V_1|}{\log \log |V_1|}\, weight(G_1))$. ■

4.2 Upper Performance Bounds

To derive upper bounds on the performance guarantee of planning with the freespace assumption, we first prove properties of the "multiple shortest path algorithm." This algorithm is given an undirected, finite graph $G = (V, E)$ with positive edge lengths (all vertices are unblocked) and a sequence $[w_i]_{i=0}^n$ of distinct vertices $w_i \in V$. When it is started at w_0, it moves on a shortest path to w_1, then moves on a shortest path to w_2, and so forth until it reaches w_n and stops. Any path that can result from this behavior is called a $([w_i]_{i=0}^n, G)$ path. We utilize the following simple properties of the multiple shortest path algorithm:

Theorem 3 *Any $([w_i]_{i=0}^n, G)$ path on any tree $G = (V, E)$ with unblocked vertices contains any edge $e \in E$ at most $\min(2|V_1|, 2|V_2|)$ times, where $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are the two disconnected graph components that are obtained from G by removing e.*

Proof: Without loss of generality, assume that $|V_1| \leq |V_2|$. The multiple shortest path algorithm traverses e towards G_1 only when it takes a shortest path from a vertex w_i in G_2 to a vertex w_{i+1} in G_1. Since the graph is a tree, the traversals of e alternate directions. If $w_0 \in V_1$, then the

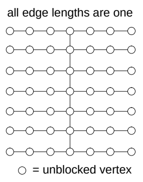

Figure 3: Spanning Tree

edge is traversed at most $|V_1|-1$ times towards G_1 and $|V_1|$ times towards G_2. If $w_0 \in V_2$, then the edge is traversed at most $|V_1|$ times towards G_1 and $|V_1|$ times towards G_2. ■

Theorem 4 *Let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be any two graphs with unblocked vertices and $dist_{G_1}(v, v') \le dist_{G_2}(v, v')$ for all vertices $v, v' \in V$. Then, the length of any $([w_i]_{i=0}^n, G_1)$ path is at most as long as the length of any $([w_i]_{i=0}^n, G_2)$ path.*

Proof: Since $dist_{G_1}(v, v') \le dist_{G_2}(v, v')$ for all vertices $v, v' \in V$, the length of any shortest path from vertex w_i to vertex w_{i+1} in G_1 is at most as long as the length of any shortest path between the same two vertices in G_2. ■

The relationship between the multiple shortest path algorithm and planning with the freespace assumption is the following:

Theorem 5 *A path of planning with the freespace assumption on graph $G = (V, E)$ is a $([w_i]_{i=0}^n, G')$ path on graph G' that is obtained from G by removing all blocked vertices and all edges that are incident to them. w_0 is the starting vertex, w_i for $i = 1 \dots n-1$ are the vertices at which planning with the freespace assumption successfully re-planned paths, and w_n is the vertex at which it reported success or failure.*

Proof: The vertices w_i are pairwise distinct, since planning with the freespace assumption only re-plans paths at vertices at which it has never been before. This is the case because it only re-plans when it realizes that its current path is untraversable and it would have known that the path was untraversable had it been at that vertex before. It moves on a shortest path from w_i to w_{i+1}, because the path it takes is a subpath of a shortest potentially traversable path to the goal. ■

Theorem 5 implies that the performance guarantee of planning with the freespace assumption is no worse than $|V|\, weight(G)$ on any graph $G = (V, E)$. This follows by summing the values of Theorem 3 over all edges, since $\min(2|V_1|, 2|V_2|) \le V$. It is currently unknown whether this bound is tight in general, but for some graph topologies one can easily obtain tighter bounds. The following theorem holds for all supergraphs of the spanning tree shown in Figure 3, such as grids or environments similar to that shown in Figure 1:

Theorem 6 *The performance guarantee of planning with the freespace assumption on graphs $G = (V, E)$ is $O(|weight(G)|^{3/2})$ if the graph topology shown in Figure 3 is a spanning tree of the graph $G' = (V', E')$ that is obtained from G by removing all blocked vertices and all edges that are incident to them.*

Proof: According to Theorem 4, for each sequence $[w_i]_{i=0}^n$ of vertices, the travel distance of the multiple shortest path algorithm on graph G' is at most as large as that on its spanning tree. Thus, the maximum travel distance on graph G' over all sequences $[w_i]_{i=0}^n$ is at most as large as that on the spanning tree. An upper bound on the maximum travel distance on the spanning tree is obtained by summing the values of Theorem 3 over all of its edges. The resulting bound is $O(|V|^{3/2}) = O(|weight(G)|^{3/2})$. According to Theorem 5, the maximum travel distance on graph G' over all sequences $[w_i]_{i=0}^n$ is an upper bound on the performance guarantee of planning with the freespace assumption on graph G, and the theorem follows. ■

It is currently unknown whether the upper bound of Theorem 6 is tight and thus whether planning with the freespace assumption provides an optimal performance guarantee for graph topologies such as grids.

5 Better Performance Guarantees

Sensor-based planning problems can be solved with a variety of graph search algorithms. A complete and-or graph search always provides the worst-case optimal solution, but is intractable [7]. Agent-centered search algorithms [3], such as planning with the freespace assumption, make the problem tractable by interleaving planning and execution. So far, we have shown that the performance guarantee of planning with the freespace assumption is, at best, superlinear in the weight of the graph (even for planar graphs) except for some restricted graph topologies. Other sensor-based planning algorithms guarantee a performance that is always linear in the weight of the graph. Chronological backtracking is an example: The robot always moves from its current vertex to a neighboring unblocked vertex that has not yet been visited; if such a vertex does not exist, it leaves the current vertex along the edge with which it was entered for the first time. If it reaches the goal vertex, its stops and reports success. If, at any point in time, it is at the starting vertex and has already traversed all of the edges that leave the starting vertex and lead to unblocked vertices, it stops and reports that the goal vertex cannot be reached. Chronological backtracking solves every sensor-based planning problem with at most two traversals of every edge and consequently provides a performance guarantee of twice

Given a graph $G = (V, E)$, Basic-VECA uses the following variables for all $v, w \in V$ with $(v, w) \in E$: $count(v, w)$ keeps track of how often the edge has been traversed from v to w, $reserve(v, w)$ is the reserved VECA cost for its traversal from v to w, and $cost(v, w)$ is the actual VECA cost for its traversal from v to w. Basic-VECA always makes the optimistic assumption that vertices are unblocked if it does not know their status. Consequently, we call a path potentially traversable if it does not contain vertices that are known to be blocked.

1. Set $count(v, w) := cost(v, w) := 0$ for all $(v, w) \in E$. Set $i := 0$ and s to the starting vertex.
2. If s is the goal vertex, then stop and report success.
3. If no potentially traversable path with finite actual VECA cost from s to the goal vertex exists, stop and report that the goal vertex cannot be reached. Otherwise, consider all traversable paths that start at s, end in an untraversed edge, and contain only traversed edges in between. (At least one such path exists. All these paths are guaranteed to contain only unblocked vertices, including the endpoints.) Select a path with minimal actual VECA cost from these paths. (This path has finite actual VECA cost.) Break ties by selecting a path with the smallest value, where the value of a path is the length of the shortest potentially traversable path from s to the goal vertex that contains the path as prefix. (At least one such path exists).
4. Repeat the following steps until the selected path has been traversed:
   (a) Let $(s, w) \in E$ be the next edge on the path.
   (b) Traverse the edge by moving to vertex w.
   (c) Set $count(s, w) := count(s, w) + 1$.
   (d) If $count(s, w) + count(w, s) = 1$, then set $i := i + 1$ and afterwards $reserve(s, w) := reserve(w, s) := 2^{-i}$.
   (e) If $count(s, w) + count(w, s) \geq k$ and $cost(s, w) = 0$, then set $cost(s, w) := cost(w, s) := reserve(s, w)$.
   (f) If $count(s, w) + count(w, s) > k$, then set $cost(s, w) := \infty$.
   (g) Set $s := w$.
5. Go to step 2.

Our implementation of Basic-VECA uses Dijkstra's algorithm in conjunction with priority lists on lexicographic path orderings to avoid exponentially decreasing VECA costs.

Figure 4: Basic-VECA

the weight of the graph. No sensor-based planning algorithm can provide a better performance guarantee. On the other hand, planning with the freespace assumption exhibits a much better average-case performance than chronological backtracking in typical robot navigation domains since chronological backtracking does not focus its search towards the goal vertex.[3] In the following, we describe a sensor-based planning algorithm that combines the advantages of these two algorithms.

The Basic Variable Edge Cost Algorithm (Basic-VECA, Figure 4) uses planning with the freespace assumption as a subroutine (tie breaker) in Step 3. It monitors the behavior of planning with the freespace assumption until it discovers that it performs unsystematically on some part of the graph, as indicated by the kth traversal of an edge (k is a parameter of Basic-VECA). It then discourages planning with the freespace assumption from traversing this edge again unless it is absolutely necessary, and thus switches gradually from focusing its search towards the goal vertex to exploration. Basic-VECA achieves this by increasing the VECA cost of the edge (which is different from its length) when it traverses the edge for the kth time. Initially, all VECA costs are zero and thus no restrictions are placed on planning with the freespace assumption. Thus, Basic-VECA behaves exactly like planning with the freespace assumption until some edge has been traversed k times, and Basic-VECA with $k = \infty$ always behaves identically to planning with the freespace assumption. Basic-VECA with $k = 0$, on the other hand, behaves identically to chronological backtracking except that it skips cycles that consist exclusively of backtracking moves when it has to backtrack multiple times. Values of k between zero and infinity produce behaviors that mediate between planning with the freespace assumption and chronological backtracking.

Theorem 7 *Basic-VECA is correct for all parameter values $k = 0, 2, 4, \ldots$ no matter how ties are broken in Step 3. Its performance guarantee on graphs $G = (V, E)$ is at most $(k+2)\,weight(G)$, i.e. $O(weight(G))$ for fixed k.*

The proof is given in [5]. The parameter k can be used to trade-off average-case and worst-case performance. The average-case performance of Basic-VECA is similar to that of planning with the freespace assumption for sufficiently large values of k; the larger the value of k is, the longer both algorithms behave the same. On the other hand, the performance guarantee of Basic-VECA is the better, the smaller the value of k is. A small value of k, however, also causes Basic-VECA to deviate earlier from planning with the freespace assumption, which can force it in some situations to explore parts of the graph unnecessarily and increase its total travel distance.

To summarize, in all environments in which planning with the freespace assumption performs well (examples are grids), Basic-VECA behaves similarly and thus has a good average-case performance. For instance, the scatter plot in Figure 5 compares the average-case performances of Basic-VECA (with $k = 2$) and depth-first search (an efficient version of chronological backtracking) in randomly generated acyclic mazes of size 64×64. The figure shows that Basic-VECA consis-

[3]Experimental results for planning with the freespace assumption are reported in [10]. Its average-case performance is very good, although recent experimental evidence [12] suggests that, at least in some domains, one can improve the total travel distance slightly by assuming that the status of a vertex is similar to the status of its neighbors instead of assuming that it is unblocked. Another advantage of planning with the freespace assumption over chronological backtracking, that we do not address in this paper, is that it is also correct on infinite graphs.

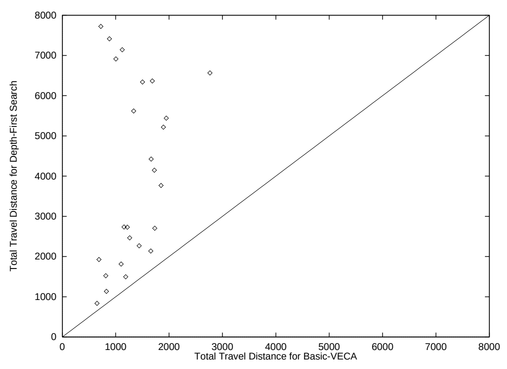

Figure 5: Basic-VECA versus Depth-First Search

tently outperformed chronological backtracking and resulted in travel distances that were about 40 percent shorter on average. On the other hand, in all environments in which planning with the freespace assumption does not perform well (examples are Figures 2 and ??), Basic-VECA acts as a "safety net" and its performance guarantee results in a much better average-case performance than that of planning with the freespace assumption. At the same time, Basic-VECA does not add much computational overhead to planning with the freespace assumption. Since Theorem 7 does not depend on which algorithm is used to break ties in Step 3, Basic-VECA is a general technique for combining *any* heuristic navigation algorithm (not just planning with the freespace assumption) with chronological backtracking. This combination inherits the efficiency of the heuristic navigation algorithm and its performance guarantee is optimal up to a constant (user-defined) factor. A more complete description of Basic-VECA and empirical results for different values of k are contained in [9].

6 Conclusion

In this paper, we have studied a popular approach to goal-directed navigation in unknown environments. Planning with the freespace assumption, a sensor-based planning approach, always plans a shortest path to the goal, assuming that the terrain is clear unless it knows otherwise. We showed that its performance guarantee is not optimal. For situations in which its performance guarantee is insufficient, we described an algorithm, called Basic-VECA, that uses planning with the freespace assumption as a subroutine. It exhibits good average-case performance and provides performance guarantees that are optimal up to a constant (user-defined) factor.

References

[1] H. Choset and J. Burdick. Sensor based planning and nonsmooth analysis. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3034–3041, 1994.

[2] G. Foux, M. Heymann, and A. Bruckstein. Two-dimensional robot navigation among unknown stationary polygonal obstacles. *IEEE Transactions on Robotics and Automation*, 9(1):96–102, 1993.

[3] S. Koenig. Agent-centered search: Situated search with small look-ahead. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1996.

[4] S. Koenig and Y. Smirnov. Graph learning with a nearest neighbor approach. In *Proceedings of the Conference on Computational Learning Theory (COLT)*, pages 19–28, 1996.

[5] S. Koenig and Y. Smirnov. Assumptive planning and generalized depth-first search. Technical Report CMU-CS-97-110, School of Computer Science, Carnegie Mellon University, 1997.

[6] V. Lumelsky. Algorithmic and complexity issues of robot motion in an uncertain environment. *Journal of Complexity*, 3:146–182, 1987.

[7] I. Nourbakhsh. *Interleaving Planning and Execution*. PhD thesis, Department of Computer Science, Stanford University, 1996.

[8] I. Nourbakhsh and M. Genesereth. Assumptive planning and execution: a simple, working robot architecture. *Autonomous Robots*, 3(1):49–67, 1996.

[9] Y. Smirnov, S. Koenig, M.M. Veloso, and R.G. Simmons. Efficient goal-directed exploration. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 292–297, 1996.

[10] A. Stentz. The focussed D* algorithm for real-time replanning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1652–1658, 1995.

[11] A. Stentz. Optimal and efficient path planning for unknown and dynamic environments. *International Journal of Robotics and Automation*, 10(3):89–100, 1995.

[12] A. Stentz. Map-based strategies for robot navigation in unknown environments. In *AAAI Spring Symposium on Planning with Incomplete Information for Robot Problems*, pages 110–116, 1996.

[13] A. Stentz and M. Hebert. A complete navigation system for goal acquisition in unknown environments. *Autonomous Robots*, 2(2):127–145, 1995.

[14] A. Zelinsky. A mobile robot exploration algorithm. *IEEE Transactions on Robotics and Automation*, 8(6):707–717, 1992.