Landmark Selection Strategies for Path Execution

Xiaotie Deng, Evangelos Milios, Andy Mirzaian
Department of Computer Science
York University
North York, Ontario, Canada M3J 1P3
{deng,eem,andy}@cs.yorku.ca

March 17, 1997

Abstract

A mobile robot often executes a planned path by measuring its position relative to visible landmarks at known positions and then using this information to estimate its own absolute position. The minimum number of landmarks k required for self-location depends on the types of measurements the robot can perform, such as visual angles using a video camera on a pan-tilt unit, distances using a laser range finder, or absolute orientation using a compass. The problem we address is how to find which k landmarks the robot should detect and track over which segments of a path, so that the cost of sensing (detection and tracking) is minimized. We assume that there are more landmarks visible from each point of the robot's path than the minimum necessary, and that their positions are known relative to the path. We present several formulations of this problem in graph-theoretic terms, with different amounts of flexibility, generality and complexity, which can be solved by known graph-theoretic algorithms. We present uniform-cost algorithms (all landmarks have equal cost of detection and tracking), and weighted-cost algorithms (each landmark has different cost). The complexity of these algorithms is low-order polynomial in the number of landmarks k that must be simultaneously tracked at each point of the robot's path. We also present an algorithm that can incorporate not only the sensing cost for each landmark, but the suitability of the landmark configurations relative to the robot. The resulting complexity is, in this case, exponential in the number of landmarks k tracked at each point of the robot path. The algorithm is still practical, since k is typically a fixed small integer.

1 Introduction

Previous research in mobile robotics has typically addressed the following problems:

- path planning in the case of a perfectly known environment, perfect odometry information, and perfect sensing. The optimality criterion in this class of problems has been the minimization of energy required for path traversal between a start and end position specified by their coordinates, while obstacles are being avoided [17].

- path execution in the case of a known real environment. The focus here has typically been the sensing required to accurately execute a preplanned path, and the main problem addressed is how to match sensed data (vision, sonar, laser, infrared) against map information [3, 12, 7, 1, 15, 19, 4].

- exploration of an unknown world with perfect range sensing and odometry information [20].

- exploration of an unknown world with noisy range sensing [16, 6], and chapter 3 of vol. 1 and chapter 4 of vol. 2 of [12].

A mobile robot typically operates in a partially known environment. A map of the permanent features of its environment is available to the robot, but several additional temporary or time-varying features may also exist along with the permanent features. Navigation of the robot, i.e. execution of a preplanned path, is a nontrivial problem for the following reasons. First, the robot always executes the commanded motion with a small error (due to physical realities like wheel slippage). Second, odometry sensors, which measure the actually executed motions, also have a small error themselves. As a result, for each step the robot makes along its preplanned path, the error between the robot's knowledge of its position and the actual position increases with the number of steps taken.

The only way to remedy the above problem is to register the robot's position with the real world often enough to prevent the position error of the robot from becoming unacceptably high. This introduces the issue of inaccuracies in sensor measurements (visual or range). It is, therefore, useful to incorporate in the off-line path planning not only the computation of the robot's path so that it avoids obstacles and minimizes a distance measure, but also the planning of the sensory operations that must be carried out to maintain the robot on its course. It is this second problem that we address in this paper, assuming that the geometric path has already been computed.

An important way of observing the real world for position estimation is by focusing on landmarks. A landmark is a localized physical feature that the robot can sense and use for the estimate of its own position relative to a map that contains the landmarks' absolute positions. The robot can measure relative visual angles between landmarks. An absolute visual angle is the angle between the forward direction of the robot and the direction from the robot to the landmark. A relative visual angle between two landmarks is the difference of their absolute visual angles. It should be pointed out that the forward direction of the robot is known within some error margin, just as the absolute position of the robot, and therefore the self-location process must determine it, as well. The important question of how the geometric configuration of the landmarks and the robot affects the relation between angular error (in measuring relative visual angles between landmarks) and localization error was addressed in [22, 23]. The question of selecting suitable landmarks for navigation is addressed in [13], and of associating sensed features with map landmarks is addressed in [14].

Recent research [18] has addressed the problem of path planning in the presence of motion and odometry error, and assuming perfect sensing in specified circular neighbourhoods of landmarks. That work has shown that polynomial algorithms are possible for this problem, provided that the directional error in the robot motion is bounded. It is also suggested that "inverse" problems are also of importance, where the issue is where to place landmarks to facilitate path planning.

In this paper, we assume a known world populated with distinct landmarks that are fixed in space and can be sensed by the robot if the line of sight between the robot and the landmark is uninterrupted. The problem we address is landmark selection: which landmarks the robot should detect and track at different parts of a given path, so as to minimize the total cost of detecting and tracking landmarks. The number of distinctly recognizable landmarks k required for position estimation is an input parameter, and it depends on the type of sensor the robot is using. Different types of sensing include:

- visual angle sensing alone. In this case, sighting of a minimum of three distinctly recognizable landmarks is required for unconstrained position estimation.

- visual angle sensing plus range sensing, or visual angle plus compass. In this case, sighting of a minimum of two distinctly recognizable landmarks is required.

- visual angle sensing plus range sensing plus compass. In this case, a single distinctly recognizable landmark is sufficient for self-location.

In general, sighting of more landmarks than the minimum required in each case leads to an overdetermined problem. Use of statistical techniques (e.g. least squares minimization) to find the robot location that is most compatible with the observed data leads to more robust self-location. In this paper we assume that detecting and tracking landmarks is costly, so minimizing this cost is worthwhile.

The sensing cost of the path is in general the sum of the computational cost of acquiring new landmarks and the cost of tracking already acquired landmarks.

In this paper we address two versions of the landmark selection problem:

- the uniform cost version. Here we assume that the cost of acquiring a new landmark is equal to one, and the cost of tracking a landmark negligible.

- the weighted cost version. Here each landmark is associated with a different cost of acquisition and tracking.

2 Definitions

In this section, we briefly review world models for robot navigation, and we present a formal model of the landmark selection problem. We formally define the world model for our discussion and mathematically state our problem of guiding the navigation of a robot. Figure 1 shows a set of polygonal obstacles and a planned path.

* The world is described as a 2– or 3–dimensional space with opaque obstacles.

* An obstacle–free route P is chosen from the robot's origin to the goal, both specified as points in the world coordinate system.

* A set of potential landmarks $L = \{L1, L2, \cdots, Ln\}$ are specified in the world. The positions of the landmarks are known in the world coordinate system. In Figure 1, landmarks $L1, L2, ..., L9$ are shown. Landmarks can be anywhere in the world.

* The robot sensors can effectively detect these landmarks somewhere along the route P. A maximal interval along which a landmark can be detected is called a *window interval* for the landmark. We denote the landmark corresponding to a window interval e by $l(e)$. In general, a single landmark may incur several intervals in the route P, resulting from obstacles blocking the robot's sensor along the route. We can visualize a landmark as a light source, illuminating one or more window intervals along the robot's path.

* Without loss of generality, we assume that, along the route P, all the window intervals have different endpoints. We assign a linear order to the endpoints of the window intervals on the

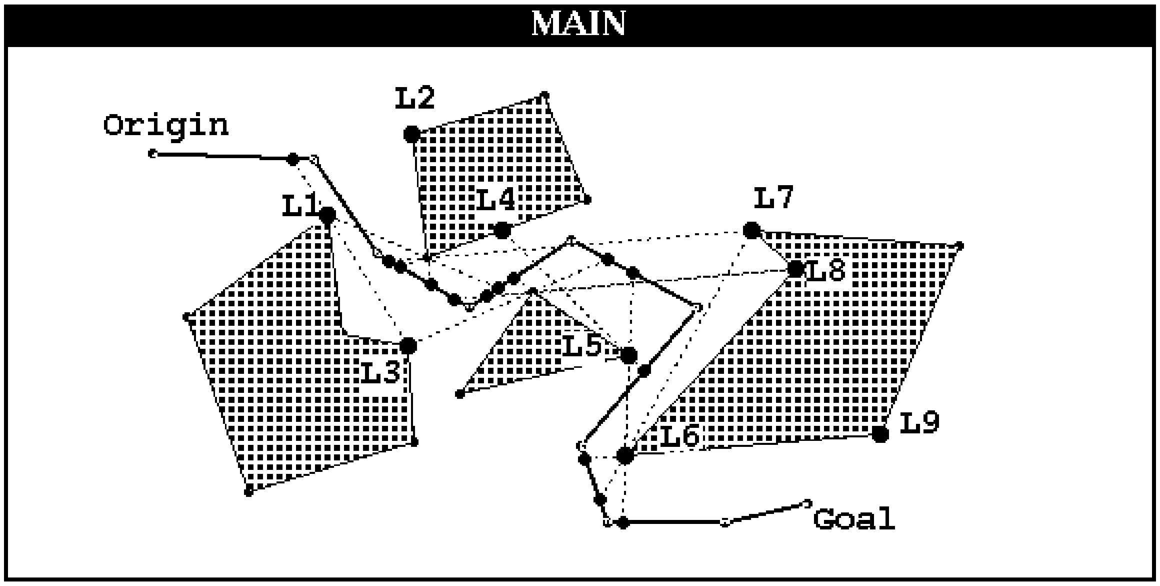


Figure 1: A typical geometry is shown, including the robot path, shown as a solid piecewise linear curve, four opaque polygonal obstacles, and nine landmarks. A window interval endpoint along the path is defined by a straight line, shown as dotted here, that starts at a landmark, and is tangent to an obstacle. The endpoint is the intersection of that line with the robot path, provided that the line does not intersect an obstacle before intersecting with the path. Endpoints are indexed by their distance from the origin of the path. Two successive endpoints due to the same landmark define a window interval. Landmarks visible from the origin or the goal define window intervals that have the origin or the goal as one of their endpoints. A single landmark typically defines several window intervals. Intuitively, the window intervals correspond to the segments of the path that are lit by each landmark, considering it as a light source. All the points of the path that do not belong to any of the window intervals defined by a landmark are in the shadow of the landmark.

route P. For each window interval e, let $left(e)$ and $right(e)$ be the point on the route P closest to the origin and the goal respectively.

* Depending on the type of sensor and the required level of robustness, the robot needs to maintain k landmarks at every point along its planned route. Such a set of landmarks over a segment of the robot path is called a *guiding set of landmarks* for that segment. We want to find guiding sets of landmarks with the minimum cost over the complete path of the robot.

We call this the *landmark selection problem* (LSP for short). As noted above, in this paper we consider different formulations of the problem of minimizing the computational cost of establishing and tracking landmarks during navigation. The simplest formulation assumes that the cost of detecting a landmark is one and the cost of maintaining it is zero, and therefore the problem reduces to the minimization of the total number of landmarks that must be detected during the navigation of the given path. We present an algorithm for this case in Section 3. A more general formulation assumes that each landmark has its own cost of detection and tracking. This is the *weighted version*. The simplest case, in which a single landmark suffices, corresponds to the practical situation when a robot is equipped with a laser range finder on a pan/tilt unit and a compass. We present an algorithm for this case in Section 4. In Section 5 we present a shortest path algorithm for the general k-landmark weighted version. The algorithm represents explicitly landmark groups that are tracked together, therefore the size of the formulation is exponential in k. It has the advantage, though, that the usefulness of different landmark groups over the same path stretch can be taken into account. In Section 6, we obtain a min-cost max-flow formulation for solving the general weighted version, with $k \geq 1$, leading to a polynomial algorithm in k. This method, however, cannot take into account the usefulness of landmarks as groups. Finally, in Section 7 we describe an implementation of the k-landmark algorithms and an example of their results on a typical two-dimensional problem involving polygonal obstacles.

3 Algorithms for the uniform version of the LSP

In this section, we discuss the *uniform* version of the landmark selection problem. This formulation assumes that the cost of detecting a landmark is one and the cost of maintaining it is zero, and therefore the problem reduces to minimizing the total number of landmarks that must be detected during the navigation of the given path. The uniform version can be formulated as a shortest path set problem in interval graphs [25] as follows:

* We construct a directed graph G. Its vertex (node) set V contains all the window intervals, also the origin and the goal. Its edge set contains directed pairs (u, v) such that window intervals u and v overlap on the route P. Edge (u, v) corresponds to a switch from landmark $l(u)$ to $l(v)$. The weight of the edge (u, v) is the computational cost for establishing the landmark v.

* If the robot needs to maintain k landmarks in its navigation, our problem is to find k vertex–disjoint paths from the origin to the goal such that the total number of vertices is minimized.

This shortest k vertex–disjoint path problem in directed graphs can be solved in polynomial time using a min–cost max–flow problem formulation ([5, 8]). However, for the uniform version, we have a much simpler solution to be introduced next. Since the corresponding graphs are interval graphs, we have much simpler polynomial algorithms. We now present a greedy algorithm that operates on a general interval graph, and in Appendix B a more efficient algorithm for the case of a polygonal world, in which polygon vertices serve as landmarks.

The idea of this algorithm is to replace the landmark that goes out of sight first with another landmark that will be visible the longest.

Algorithm **GREEDY**

0. Construct the interval graph, and on each node n_i, we call the right endpoint of the corresponding window interval n_i, $right(n_i)$, the value of the node and denote it by $v(n_i)$. Choose k nodes adjacent to the origin which have the k–highest values, $v(n_1) \leq v(n_2) \leq \cdots \leq v(n_k)$.

1. Suppose, the current landmark set is $v(n_1) \leq v(n_2) \leq \cdots \leq v(n_i)$ with $i \geq k$. The last k landmarks are the ones currently tracked by the robot, as they are the ones that will go out of sight last. Landmark $v(n_{i-k+1})$ is the one that will go out of sight first and will have to be replaced.

2. If the value is $v(n_{i-k+1}) < v(goal)$, then the goal has not been reached yet. Among all the nodes which are adjacent to n_{i-k+1}, choose the one with the highest value, i.e. the landmark that will go out of sight last. Insert it in the list and rename the nodes so that they are ordered.

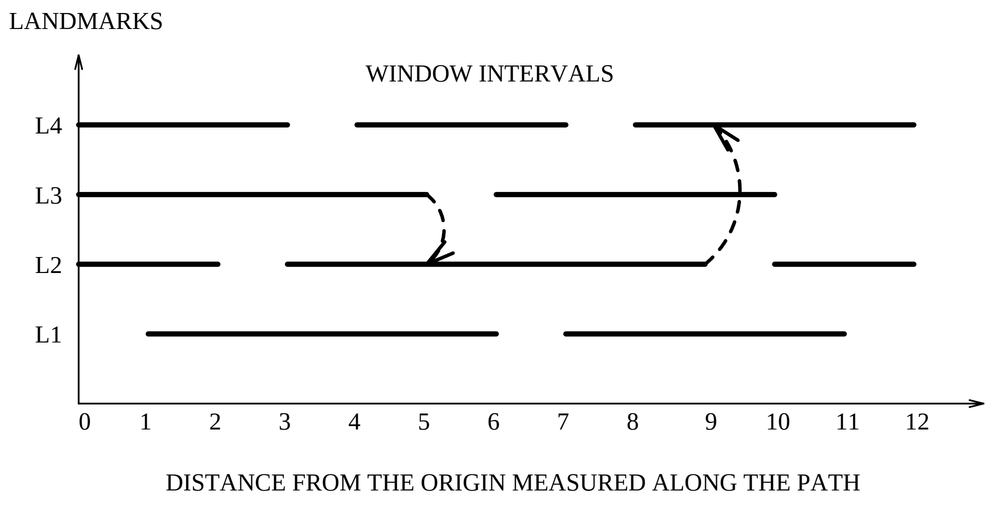


Figure 2: Window intervals due to four landmarks are shown as horizontal line segments. Distance along the horizontal in this diagram corresponds to distance from the origin along the path in a geometric representation. Window intervals at the same height are due to the same landmark. The optimal solution computed by the GREEDY algorithm for the case k=1 is also shown: the robot starts by tracking landmark L3, then switches to L2 at point x5, and finally switches to L4 at point x9.

3. If $v(n_j) \geq v(goal)$ for all $j : i+1-k < j \leq i+1$, then terminate and return with the node set $\{n_1, n_2, \cdots, n_{i+1}\}$.

4. $i \leftarrow i+1$. Go to Step 1.

For the example in Fig 2, we sort the endpoints as x_0, x_1, $\cdots$, x_{12}. We designate a window inteval by (x_i, x_j). Notice that in this example, some endpoints of the intervals coincide. When $k = 1$, applying GREEDY, we obtain the solution: $[(x_0, x_5), (x_3, x_9), (x_8, x_{12})]$. This solution implies that the robot will track landmark L_3 from the origin of the path till point 5 on the path, then it will switch to landmark L_2 till point 9, and finally to L_4 till the goal (point 12) is reached.

In Appendix A we prove the following theorem:

Theorem 3.1 *GREEDY gives the optimal solution for the uniform version of the LSP.*

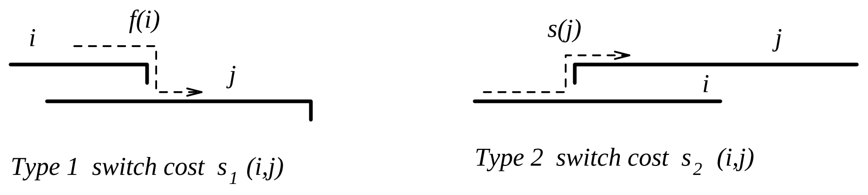
Thus, in general, we can first obtain all the window intervals of landmarks on the planned route and then apply GREEDY to obtain an optimal solution. This algorithm covers very general situations. It allows different types of sensors, allows different range of sensors and landmarks, and different locations of the landmarks. Thus, for this approach, the time complexity of LSP will be the sum of the time to construct the window intervals along the planned route, the time to sort the end points of the intervals, and the time for the GREEDY algorithm to be applied to the interval graph. The total number of window intervals may not be linear in the input size since a single landmark may project $O(n)$ intervals in the planned route, where n is the number of obstacles. However it is polynomial in the input size. Thus, this approach gives a polynomial time algorithm for the LSP. In Appendix B, we simplify the solution for the case of 2-dimensional polygon interior world.

4 A shortest-path algorithm for the single-landmark nonuniform version

In this section we consider the single-landmark nonuniform case: we want to find the least costly landmark selection schedule, given that the necessary preprocessing is done and we have the list of intervals along the designated path with their endpoints sorted along the path, and the associated sensing cost for detecting and tracking each landmark. This case is practical for a robot equipped with a compass and an orientable range-measuring device, such as a laser range finder on a pan unit. Once the robot has detected a known landmark, it can measure the bearing with respect to north for a single landmark (i.e. the angle defined from the north direction to the direction from the robot to the landmark), plus the distance to the landmark. This is sufficient information for the robot to locate itself on a map containing the landmark.

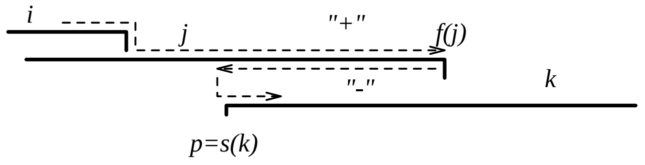
The case $k = 1$ turns out to be a shortest path problem as follows. Consider the ordered list of landmarks, and their associated intervals, that appear on the optimal schedule. As the robot moves along the path, from time to time it will switch from the current landmark to the next one on the schedule. There are two possible types of landmark switches as shown in Figure 3(a).

A *type 1*, or *end-to-middle* type switch is when the robot has reached the finishing end-point $f(i)$ of the current landmark i, and switches to landmark j which is the next landmark on the schedule. We associate a cost $s_1(i, j)$ to this type of switch. A *type 2*, or *middle-to-end* type switch is when the robot has to deactivate the current landmark i and switch to the beginning $s(j)$ of the next landmark on the schedule. We associate a cost $s_2(i, j)$ with this type of switch. (It is not hard to see that without loss of generality we can ignore any "middle-to-middle" switch, since there always is an optimal schedule in which there is no occurrence of such a switch.) A third cost

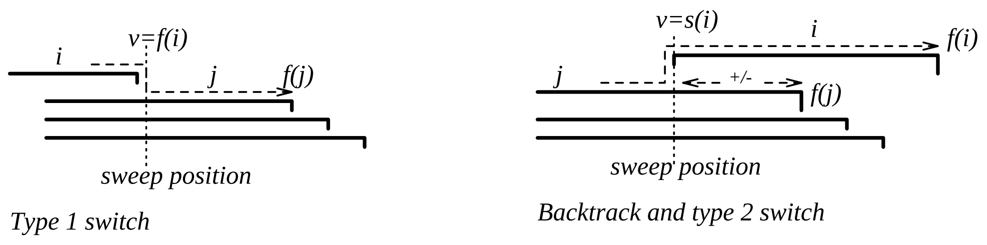


i
f(i)
j
s(j)
j
i
Type 1 switch cost s_1 (i,j)
Type 2 switch cost s_2 (i,j)

(a) Type 1 and 2 switches.



(b) The backtracking trick.



(c) Updating the costs during the sweep.

Figure 3: A shortest path algorithm for the single-landmark nonuniform version.

is the cost of maintaining the tracking of the current landmark i, and we assume this is a linear cost, that is, it is $d \cdot c(i)$ where d is the distance traveled while tracking landmark i, and $c(i)$ is the unit distance cost.

Figure 3(b) shows that we can consider a middle-to-end switch from landmark j to landmark k at position $p = s(k)$ in a cost-equivalent form of first moving to $f(j)$, then subtracting the cost for returning back to position p, then making the switch to landmark k.

With this *backtracking* trick we need to consider only the minimal cost schedules from the starting point s to only end-points of the intervals.

This brings us to the following algorithm.

Let $\mathit{cost}(v)$ denote the minimum cost of a schedule from the start s to the interval finishing end-point v. Our algorithm will make one sweep over the path from s to t and maintain tentative values for $\mathit{cost}(v)$ for each interval finishing end-point v with the following invariant: for each interval finishing end-point v, $\mathit{cost}(v)$ is the minimum cost schedule for going from s to v without passing through end-points not yet swept. Initially $\mathit{cost}(s) \leftarrow 0$, and $\mathit{cost}(v) \leftarrow \infty$ for all (other) endpoints v.

Let us call an interval *active* if it includes the current sweep position. Suppose the next endpoint to be swept is v. There are two cases to consider depending on whether v is a starting or a finishing end-point of an interval. Figure 3(c) shows what needs to be done in these cases.

First consider a sweep event of type $v = f(i)$. Consider all the active intervals other than i. For each such interval j set:

$$\begin{aligned} &\mathit{cost}(f(j)) \leftarrow \\ &\qquad \mathit{min}(\quad \mathit{cost}(f(j)), \\ &\qquad\qquad \mathit{cost}(f(i)) + s_1(i,j) \; + \; c(j) \cdot d(f(i), f(j))). \end{aligned}$$

where $d(p,q)$ is the distance along the path from p to q. The above cost update corresponds to making a type 1 switch from i to j and then moving all the way to the finishing end-point of interval j. The total time required for this event is proportional to the number of active intervals, i.e., intervals that include v. This is clearly $O(m)$ time in the worst-case, where m is the number of landmark window intervals over the path.

Now consider a sweep of type $v = s(i)$. For each active interval j (other than i) consider the backtracking trick for making a type 2 switch from i to j. That is, set:

$$\begin{aligned}
&cost(f(i)) \leftarrow \\
&\qquad min(\quad cost(f(i)), \\
&\qquad\qquad cost(f(j)) - \quad d(s(i), f(j)) \cdot c(j) + \\
&\qquad\qquad\qquad s_2(j,i) + d(s(i), f(i)) \cdot c(i)).
\end{aligned}$$

The total time for processing such an event is again at most $O(m)$.

Finally, when the sweep terminates, the minimum cost of a schedule to go from s to t would be the minimum of $cost(f(j)) - d(t, f(j)) \cdot c(j)$, where the minimum is taken over all intervals j that include position t.

¿From the above discussion we conclude the following theorem:

Theorem 4.1 *There is an algorithm that computes the optimum weighted landmark scheduling problem for the case $k = 1$ and takes a total of $O(m^2)$ time and $O(m)$ space, where m is the number of landmark intervals on the path.*

5 A shortest-path algorithm for the weighted version

In practice, it is desirable to address the general case of k landmarks being required for position estimation at all times. This is necessary if the robot has no compass, therefore requiring more landmarks for self-location (e.g. three, if only relative bearings can be measured). More landmarks than the minimum required give rise to an overdetermined self-location problem, which can be solved with least-squares techniques. Furthermore, some sets of landmarks may not be very useful for the position estimation problem. For example, if the number of landmarks required $k = 3$, and only bearing to landmarks is measured, it is not very useful if the three landmarks are cocircular with the robot, since the two circles intersecting at the robot position now collapse into a single circle, making the robot position indeterminate. In general, whether a set of k landmarks is useful for self-location depends on the position of the landmarks relative to the current position of the robot. As the robot moves along its path, their usefulness may also change.

To address this problem, we propose the following formulation. Assume that window intervals have been defined along the path, as before. We sort the endpoints of these intervals in increasing order:

$$x_0 < x_1 < x_2 < ... < x_m \quad (1)$$

It does not matter if accidental alignment occurs and two endpoints coincide: for defining the window intervals, we consider only one of the coincident endpoints. We now focus on a primitive interval $[x_r, x_{r+1}]$, defined by two successive endpoints. Assume that the robot can sense M_r landmarks $\{l_{r_1}, l_{r_2}, ..., l_{r_{M_r}}\}$ while within that interval. There are: $\left(\begin{array}{c} M_r \\ k \end{array} \right)$ possible selections of k landmarks out of M_r. For each such selection $\{l_{s_1}, l_{s_2}, ...l_{s_k}\}$, we define a single undesirability value $U(V_s)$, which is inversely proportional to the usefulness of the selection for navigation over the interval $[x_r, x_{r+1}]$. In [23] a measure for the usefulness of 3 landmarks is developed. We note that switching between landmarks is a discrete process, whereas the undesirability measure varies continuously as the robot moves. This can be addressed as follows:

1. For each selection of k landmarks, compute the continuous undesirability value as a function of the distance of the robot from the origin along its path.
2. Insert the points, where the undesirability value is equal to the threshold, into the window interval endpoint sequence $x_1, x_2, ..., x_m$ defined previously.
3. Landmark selections with undesirability value below the threshold are excluded from further consideration, unless no other selections exist with undesirability value above the threshold over the same window interval.

The result of the above preprocessing is a sequence of window intervals, whose endpoints are defined either by landmark visibility discontinuities or by points where the undesirability value of visible landmark selections exceeds a given threshold. A set of landmark selections that are acceptable for locating the robot is associated with each window interval. Which ones will be chosen is found by the following algorithm.

1. Sort all endpoints of the window intervals in increasing order: $x_0, x_1 < x_2 < ... < x_m$.
2. For each primitive interval $[x_i, x_{i+1}]$ of the endpoint sequence, do the following:
   (a) Determine all possible selections of k landmarks that are visible from within that interval. Define a graph vertex V_s for each one of them.

(b) For the interval $[x_{i-1}, x_i]$, determine all landmark selections that differ by at most one landmark from vertex V_s. Connect each such landmark selection V_{s-1} with vertex V_s via a directed edge. Assign a length to that edge equal to the sum of the cost of tracking the landmarks of V_{s-1} over the interval $[x_{i-1}, x_i]$, the average undesirability $U(V_s)$ over the same interval, and the cost of detection of the landmark that is new in V_s.

3. Define an initial vertex V_0 connected to all landmark selections over interval $[x_0, x_1]$. The length of an edge emanating from V_0 and ending at a vertex V_s defined over the interval $[x_0, x_1]$ is the sum of the cost of detection of all landmarks in V_s and the undesirability $U(V_s)$.

4. Define a goal vertex V_g to which all landmark selections over interval $[x_{m-1}, x_m]$ are connected. The length of an edge emanating from a vertex V_s defined over the interval $[x_{m-1}, x_m]$ and ending at V_g is the sum of the cost of tracking the landmarks in V_s over interval $[x_{m-1}, x_m]$.

5. The problem of landmark selection is then reduced to the problem of finding the shortest path from V_0 to V_g.

The number of nodes of the graph due to a single primitive interval depends both on the number of landmarks k required for self-location and the total number of landmarks M_r visible from a given location.

Figure 4 shows part of the graph for this formulation.

6 A min–cost max–flow algorithm for the weighted version

In this section, we present a min–cost max–flow problem formulation for the *weighted version* of the LSP. Since we know that the min–cost max–flow problem can be solved in polynomial time ([5, 8]), it follows that LSP can be solved within time polynomial both in the number of window intervals for the LSP and in the number of landmarks k required for robot self-location.

Theorem 6.1 *Suppose the number of window intervals along the planned route is polynomial in the input size of the LSP. The weighted version of the LSP can then be solved in polynomial time.*

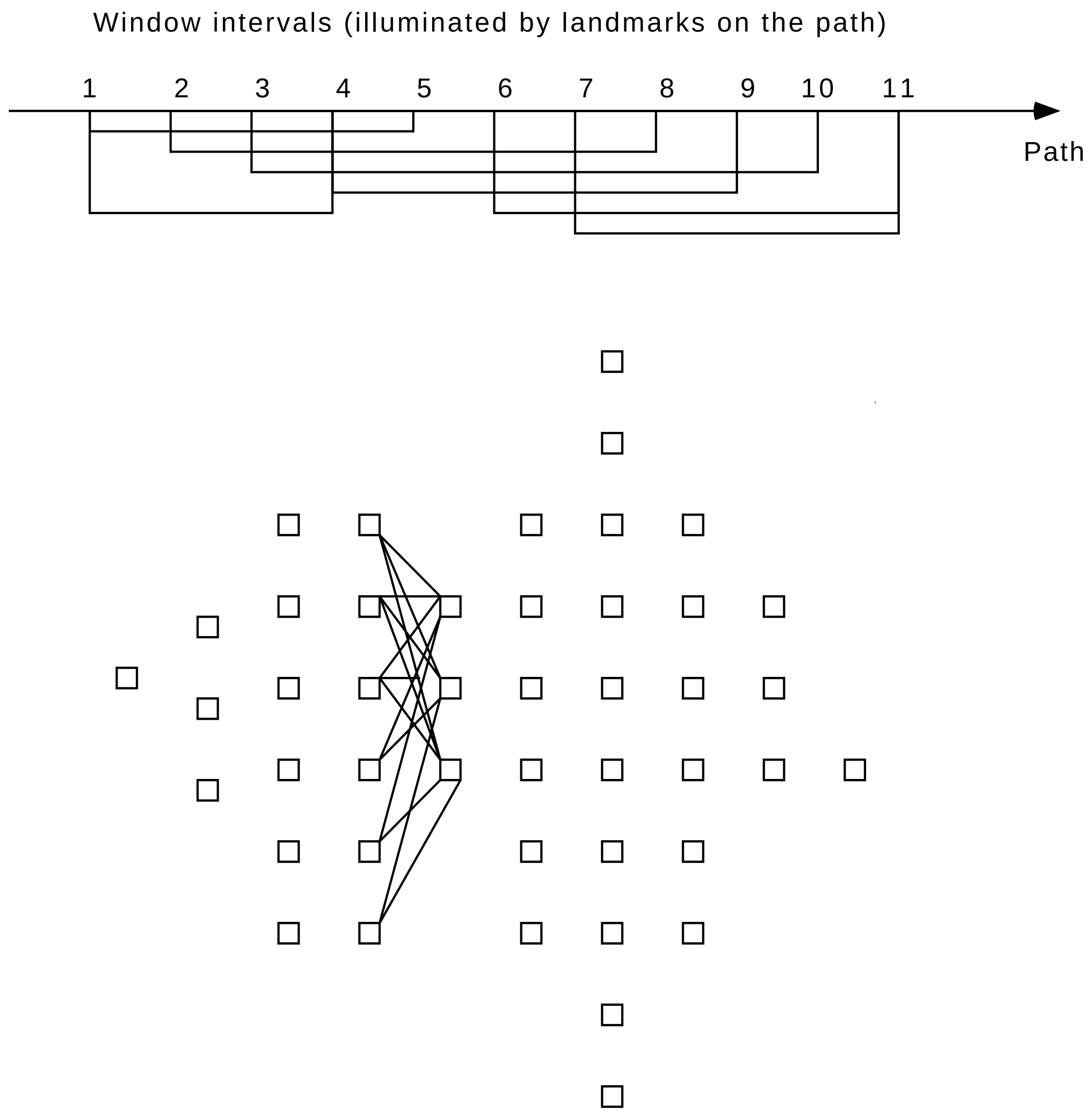


Figure 4: The figure shows part of the graph for the shortest-path formulation of the landmark selection problem. All the nodes corresponding to a primitive interval are vertically aligned with it. The edges are directed from left to right.

In this formulation, first we sort all the endpoints of the intervals in increasing order: $x_1 < x_2 < \cdots < x_m$, where x_1 is the origin and x_m is the goal. For a window interval $e = [x_i, x_j]$ with $i < j$, we construct $j - i + 1$ nodes $v_{e,i}, v_{e,i+1}, \cdots, v_{e,j}$ and an edge of capacity one from $v_{e,k}$ to $v_{e,k+1}$ for all $k : i \leq k < j$. The cost of the edge $[v_{e,k}, v_{e,k+1}]$ is the same as the computational cost of *visually tracking* the landmark $l(e)$ from x_k to x_{k+1} (Figure 5).

We call this subgraph corresponding to a window interval the *interval subpath*. To build the network, we first build interval subpaths for all the window intervals. Then, we need to add more edges between interval subpaths corresponding to intersecting window intervals. These edges represent the case when the robot switches from a landmark it has been tracking so far to a new landmark. Therefore, the cost of these edges is equal to the cost of *visually detecting or acquiring* the new landmark.

Let $e = [x_i, x_j]$ and $f = [x_k, x_l]$ be two intersecting window intervals with $i \leq k \leq j \leq l$ (the case when one interval is contained in another is treated similarly). We establish a directed edge from $v_{e,t}$ to $v_{f,t}$ with capacity one for all $t : k \leq t \leq j$. Its cost will be the computational cost to detect or acquire the landmark $l(f)$. This edge corresponds to the robot switching from landmark $l(e)$ to landmark $l(f)$ at time t. In the most general case, we also establish a directed edge from $v_{f,t}$ to $v_{e,t}$ with capacity one for all $t : k < t < j$ and the cost being the computational cost to establish the landmark $l(e)$. This edge corresponds to the robot switching from landmark $l(f)$ to landmark $l(e)$ at time t (note that with these edges the graph ceases being acyclic).

In practice, the robot will rarely decide to switch from landmark $l(e)$ to landmark $l(f)$ at a time $t \neq j$. It will also rarely decide to switch from a later landmark $l(f)$ back to an earlier one $l(e)$, unless the costs of tracking and detecting landmarks vary with the relative position of the robot with respect to the landmark. In Figure 5 we show part of the flow graph in the simplest case of not allowing switches either to earlier landmarks or halfway through an interval.

The above directed graph can now form the basis for a min-cost max-flow formulation of the landmark selection problem. Suppose we need d landmarks at all times along the path. We visualize the d landmarks at each primitive window interval as determined by a "flow" of value d from a source node to a sink node through the above graph. If a horizontal graph edge has non-zero flow, i.e. flow equal to 1, since the capacity of the edges is equal to 1, this implies that the associated landmark is being retained and tracked over that primitive interval. If a vertical graph edge has non-zero flow, i.e. flow equal to 1, this implies that a landmark switch is taking place. "Pushing" these flows through these edges incurs a cost equal to the tracking or detection cost respectively. To complete the construction of the graph, on which the min-cost max-flow computation is defined, we must add four more nodes to the graph (Figure 5).

- A single node x_0 connected with a directed edge of capacity 1 to each node x_{1r} corresponding to an interval of the form $[x_1, x_r]$. The cost of that edge is equal to the cost of acquiring the landmark corresponding to the node x_{1r}.
- A *source node* S connected with a single directed edge of capacity d and cost 0 to node x_0. The flow of size d originates at this node, and then it propagates through the network. No cost is incurred by this flow, as it does not correspond to an acquisition or tracking of a landmark, but merely to initialize the problem.
- A node x_{m+1} connected with a directed edge of capacity 1 from each node corresponding to an interval of the form $[x_r, x_m]$. The cost of that edge is zero (this asymmetry with x_1 is a result of the fact that we choose to incorporate the cost of landmark acquisition into an edge that is directed towards the node associated with such acquisition).
- A *sink node* T connected with a single directed edge of capacity d and cost 0 from node x_{m+1}. The sink node serves to collect all the flow that originated at the source node.

We have now completed the construction of a network. It is straightforward to show that by definition the min–cost max–flow solution of this network corresponds to an optimal solution to the Landmark Selection Problem. $\square$

7 Implementation

The shortest path algorithm and the min-cost flow algorithm for the nonuniform case have been implemented, together with a program that generates intervals given a path, a list of polygonal objects and a list of landmarks. The shortest path computation uses the Moore-Bellman-d'Esopo algorithm as described in [24]. The min-cost flow algorithm used is the Busacker-Gowen algorithm [24]. In the formulation of the min-cost flow problem we omitted the detection edges that are directed upward in figure 5, or, more precisely, the edges that involve a switch to a landmark that was observable earlier than the landmark being abandoned. These edges will rarely be traversed in practice, since typically the cost of tracking is much smaller than the cost of detection.

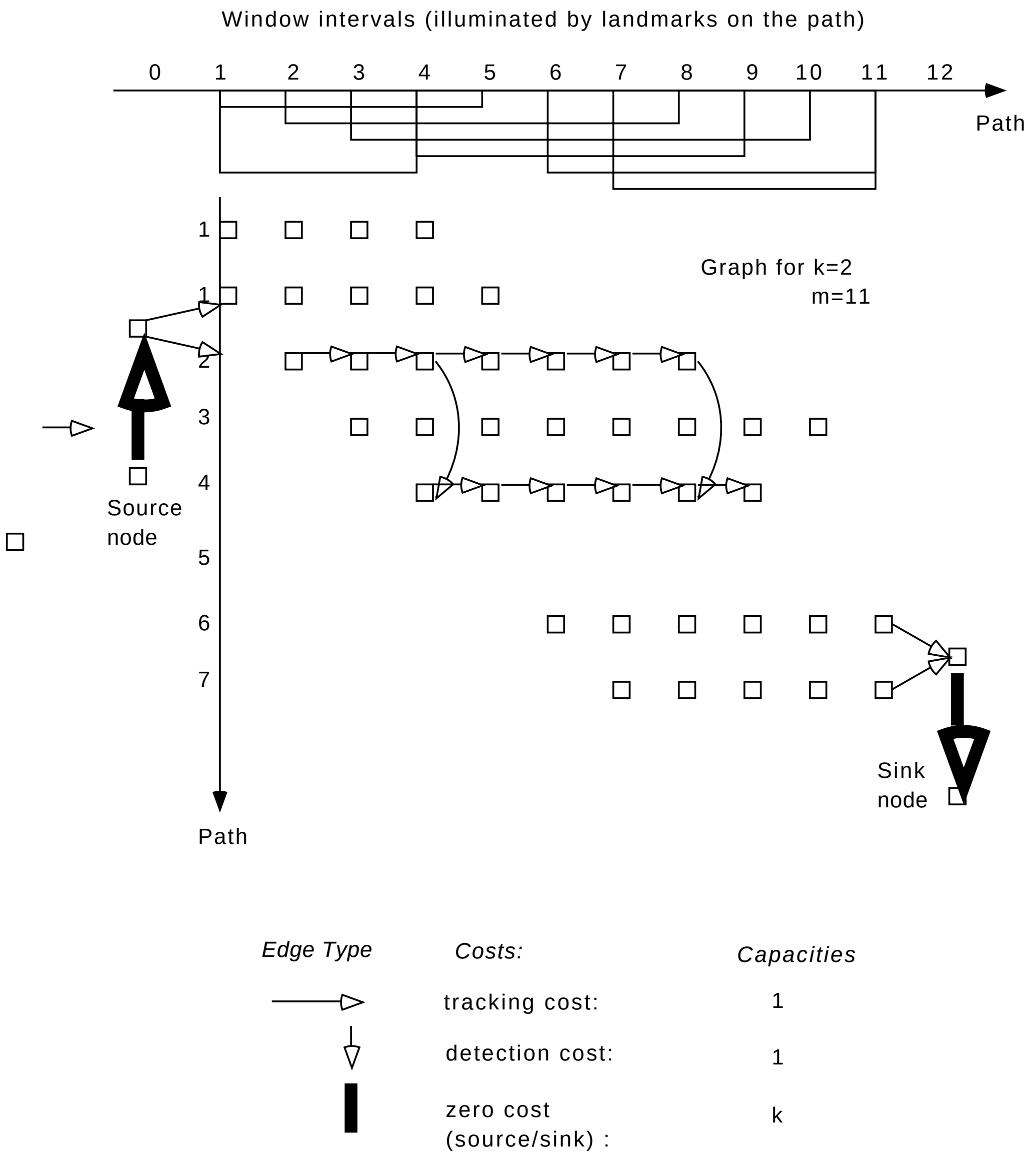


Figure 5: The directed graph that reduces the weighted landmark selection problem to a min–cost–max–flow problem. The path has been stretched to a horizontal straight line. We show all the nodes of the graph. To avoid cluttering the display, we only show the edges associated with the source and the sink node and with the window intervals starting at positions 2 and 4 of the path. Many more horizontal and vertical edges are present in the complete network. Each horizontal sequence of nodes is connected by horizontal edges, and is aligned with the window interval it corresponds to. Flow of size 1 passing through that sequence corresponds to continued tracking of the associated landmark. A vertical edge corresponds to a switch between two landmarks, and its cost is the acquisition cost of the new landmark.

Landmark	x	y	detection-cost	tracking cost
L1	166	188	10.0	1.0
L2	210	230	8.0	0.8
L3	208	120	12.0	1.2
L4	256	179	14.0	1.4
L5	323	115	9.0	0.9
L6	321	63	7.0	0.7
L7	387	180	6.0	0.6
L8	410	160	15.0	1.5
L9	454	74	5.0	0.5

Figure 6: A list of landmarks with their location and properties

Figure 1 shows a typical geometry including several landmarks, polygonal obstacles and a given robot path. Interval endpoints correspond to occlusions. If we view landmarks as light sources and assume that obstacles are opaque, interval endpoints are the boundaries of the shadows cast on the path by the objects, if illuminated by one light source at a time. It should be pointed out that, although the example shown in figure 1 is two-dimensional, the algorithms we presented are applicable to the general three-dimensional case of a point robot moving along a 3D curve among 3D obstacles observing point landmarks.

Figure 6 shows the list of landmarks, together with their (x,y) coordinates and their detection and tracking costs. Figure 7 shows the landmark detection and tracking schedule, as computed by both the shortest path and the min-cost flow algorithms on the landmarks of the previous figure and with $k = 2$. In both cases we assume that the tracking cost is constant, independent of the distance over which the landmark must be tracked, and it is incurred every time an endpoint is passed by the robot. Since the two algorithms solve the same optimization problem, they come up with identical results.

The program was implemented in Allegro Common LISP and the Common LISP Object Standard (CLOS).

8 Discussion

This paper describes a class of algorithms for establishing an execution plan for a given path in a known environment. The execution plan consists of sets of landmarks that are visible from

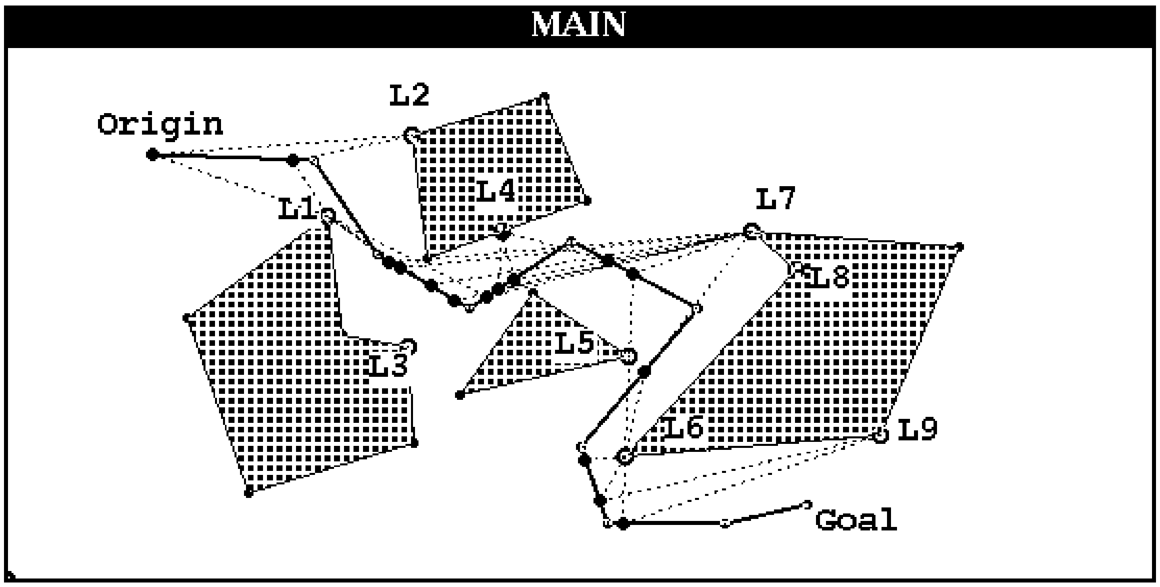


Figure 7: The landmark detection and tracking schedule. The endpoints are indexed sequentially in order of their distance from the start of the path. The figure shows the result graphically according to the following convention. Each endpoint is connected by dotted lines to the landmarks that must be tracked over the primitive interval starting at the endpoint. Primitive intervals are defined by consecutive endpoints. The following figure shows the same result in text form. One notes that both algorithms avoid the "costly" landmarks, and try to minimize the number of landmark switches.

```
From      To          Track
endpoint  endpoint    Landmarks

 0         1          (L1 L2)
 1         2          (L1 L2)
 2         3          (L1 L7)
 3         4          (L1 L7)
 4         5          (L1 L7)
 5         6          (L1 L7)
 6         7          (L1 L7)
 7         8          (L4 L7)
 8         9          (L4 L7)
 9        10          (L4 L7)
10        11          (L6 L7)
11        12          (L6 L7)
12        13          (L6 L9)
13        14          (L6 L9)
14        15          (L9 L6)
```

Figure 8: The result of the previous figure in text form.

and must be detected and tracked at different segments of the given path. Not all landmarks visible from a given segment of the path are included in the landmark set associated with that segment. Landmark sets and segments are defined so as to minimize the effort of detecting and tracking landmarks. The question of how to estimate the costs of detection and tracking of a given landmark is not addressed in this paper. We propose determining these costs experimentally for specific landmarks, and image processing hardware and software. The long-term objective is to use these algorithms on the mobile robot ARK ([21]). The mobile platform is a Cybermotion model, equipped with a novel video/laser sensor on a pan-tilt unit, capable of computing distances to visually detected landmarks to a maximum distance of about $100m$ with a maximum distance error of $5cm$ and a laser beam width of $5mrad$.

Acknowledgements. The research described in this paper is funded by research grants from the Natural Sciences and Engineering Research Council of Canada. The ARK Project receives its funding from PRECARN Associates Inc., the Department of Industry, Science and Technology, the National Research Council of Canada, the Ontario Technology Fund. Ontario Hydro, and Atomic Energy of Canada Limited. An earlier version of this paper was presented at the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS), Yokohama, Japan.

References

[1] R.A. Brooks. *Visual Map Making for a Mobile Robot*, volume 13. Readings in Computer Vision, edited by M. A. Fischler and O. Firschein, Morgan Kaufmann Pub., Inc., Los Altos, California, 1987.

[2] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete and Computational Geometry*, 6:485–524, 1991.

[3] I. Cox and G. Wilfong (eds.). *Autonomous Robot Vehicles.* Springer Verlag, New York, 1990.

[4] I. Masaki (ed.). *Vision-based Vehicle Guidance.* Springer-Verlag, New York, 1992.

[5] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J.ACM*, 19:248–264, 1972.

[6] A. Elfes. Sonar-based real-world mapping and navigation. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 233–249. Springer Verlag, 1990.

[7] C. Fennema, A. Hanson, E. Riseman, J.R. Beveridge, and R. Kumar. Model-directed mobile robot navigation. *IEEE Transactions on Systems, Man and Cybernetics*, 20(6), Nov.-Dec. 1990.

[8] L.R. Ford and D.R. Fulkerson. *Flows in Networks.* Princeton University Press, Princeton, NJ, 1962.

[9] M.R. Garey, D.S. Johnson, F.D. Preparata, and R.E. Tarjan. Triangulation of a simple polygon. *Inform. Process. Lett.*, pages 175–180, 1978.

[10] L.J. Guibas, J. Hershberger, D. Leven, M.Sharir, and R.E. Tarjan. Linear time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–234, 1987.

[11] John Hershberger and Jack Snoeyink. Computing minimum length paths of a given homotopy class. In *Proc. 2nd WADS Springer-Verlag LNCS 519*, volume 2, pages 331–342, 1991.

[12] S. Iyengar and A. Elfes. *Autonomous Mobile Robots, Vols. 1 and 2.* IEEE Computer Society Press, Los Alamitos, CA, 1991.

[13] T. Kanbara, J. Miura, and Y. Shirai. Selection of efficient landmarks for an autonomous vehicle. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1332–1338, 1993.

[14] E. Krotkov. Mobile robot localization using a single image. In *IEEE Int. Conf. on Robotics and Automation, Scottsdale, AZ*, pages 978–983, May 1989.

[15] B. Kuipers and Y. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8:47–63, 1991.

[16] B. Kuipers and Y-T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. In W. Van de Welde, editor, *Toward Learning Robots*, pages 47–64. MIT Press, 1993.

[17] Jean-Claude Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, 1991.

[18] A. Lazanas and J-C. Latombe. Landmark-based robot navigation. Technical Report STAN-CS-92-1428, Dept. of Computer Science, Stanford University, May 1992.

[19] T.S. Levitt and D. T. Lawton. Qualitative navigation for mobile robots. *Artificial Intelligence*, 44:305–360, 1990.

[20] V. Lumelsky. A comparative study of the path length performance of maze-searching and robot motion algorithms. *IEEE Transactions on Robotics and Automation*, 7(1), February 1991.

[21] S. Nickerson, D. Long, M. Jenkin, E. Milios, B. Down, P. Jasiobedzki, A. Jepson, D. Terzopoulos, J. Tsotsos, D. Wilkes, N. Bains, and K. Tran. ARK: Autonomous navigation of a mobile robot in a known environment. In *International conference on Intelligent Autonomous Systems, IAS-3, Feb. 15-19, Carnegie-Mellon University, Pittsburgh, PA*, 1993.

[22] K. Sutherland and W. Thompson. Inexact navigation. In *IEEE Int. Conference on Robotics and Automation, Atlanta, GA*, pages 1–7, 1993.

[23] K. Sutherland and W. Thompson. Localizing in unstructured environments: Dealing with the errors. *IEEE Transactions on Robotics and Aytomation*, 10(6):740–754, December 1994.

[24] M. Syslo, N. Deo, and J. Kowalik. *Discrete Optimization Algorithms.* Prentice–Hall, Inc., New Jersey, 1983.

[25] R. Wilson and J. Watkins. *Graphs: An Introductory Approach.* John Wiley and Sons, Inc., New York, 1990.

A Proof of the optimality of the GREEDY algorithm

Consider k disjoint paths as k intervals $[0, L]$, that must be covered using window intervals representing visible portions of landmarks.

The main idea of the greedy algorithm is to cover the intervals $[0, L]$ from left to right and, with a fixed number of window intervals, cover $[0, t]$, $t \leq L$, k times for t as large as possible, and then cover $[t, s]$, $k-1$ times for s $(t \leq s \leq L)$ as large as possible, and so on.

More formally, let $\{[0, p_i] : 1 \leq i \leq k\}$ a set of k disjoint paths with m nodes (m corresponds to the total number of landmarks to be accessed). Without loss of generality, let $p_1 \leq p_2 \leq \cdots \leq p_k$ (rename them if necessary.)

The GREEDY algorithm maximizes the vector $< p_1, p_2, \cdots, p_k >$ in a lexicographically maximum sense among all such collections of paths with m nodes. That is, with cost m, we want to maximize the distance the robot can move while accessing at least k landmarks.

Consider a partially constructed S, it does not have k internally disjoint path from start point to the goal yet. Define $range(S)$ to mean the maximum value such that there are k paths cover the interval $[0, range(S)]$. Under this condition, after $range(S)$, we want to maximize the distance the robot can move while accessing $k-1$ landmarks, and so on.

When the robot has to move further than p_1, we pick a node corresponding to an interval starting before the end of p_1 (to replace the interval ending at p_1 as the new landmark from p_1 on) and extending as far as possible. Because of the choice for p_2, this new added landmark allows us to move the farthest with cost $m+1$ while accessing to k landmarks all the way.

This heuristic turns out to be optimal because of the special type of interval graph we deal with. The formal proof is as follows.

Let S be the solution obtained by Algorithm GREEDY discussion in Section 3. Thus S consists of k internally disjoint paths S_i, where:

$$S_i = [origin = s_{(i,0)},\ s_{(i,1)},\ \cdots,\ s_{(i,p_i)} = goal], \quad i = 1, 2, \cdots, k.$$

We prove that, for any set of k internally disjoint paths from the origin to the goal, S has the minimum size. Suppose this is not true. We choose the set T

$$T_i = [origin = t_{(i,0)},\ t_{(i,1)},\ \cdots,\ t_{(i,q_i)} = goal], \quad i = 1, 2, \cdots, k,$$

of k internally disjoint paths from the origin to the goal that has the following properties:

- T has the minimum size.
- T has the maximum h, where h be the first time, according to the order nodes in S are chosen, that a node $s_{(i,j)}$ is selected to add to S with $s_{(i,j)} \neq t_{(i,j)}$. For example, suppose $s_{(1,1)}$ is the longest interval intersecting the starting point 0 but $t_{(1,1)}$ is not the same as $s_{(1,1)}$. Then h=1. The GREEDY algorithm will choose intervals to put into S in a certain order. Following that order, consider the first chosen, the second chosen, and ..., $i-th$ chosen interval. If the $i-th$ chosen interval $s_{(x,y)}$ is not the same as $t_{(x,y)}$ but all those chosen before match, we let $h=i$.

The following lemma is useful.

Lemma A.1 *For $j \geq 1$, $t_{(i,j)}$ is adjacent to $t_{(i,j-1)}$ but $t_{(i,j)}$ is not adjacent to $t_{(i,l)}$ for $l < j-1$, (we use the convention that $t_{(i,-1)} = 0$.)*

Proof of Lemma. If $t_{(i,j)}$ is adjacent to $t_{(i,l)}$ for some $l < j-1$, then we can remove $t_{(i,j-1)}$ and T_i is still a path from the origin to the goal by the interval graph properties.

With this lemma, we are ready to complete our proof by induction.

- Base Case $h=1$. We can permute T_i, $1 \leq i \leq k$, such that $t_{(1,0)} > t_{(2,0)} > \cdots > t_{(k,0)}$. If $s_{(1,0)}$ is still not the same as $t_{(1,0)}$, then $s_{(1,0)} > t_{(1,0)}$ by GREEDY algorithm. We can simply replace $t_{(1,0)}$ by $s_{(1,0)}$. To do this, we need to make sure that $s_{(1,0)}$ is not in $\cup_{i=1}^{k} T_i$, which is guaranteed by Lemma A.1.
- Recursive case. Let

  $$S'_l = [origin = t_{(l,0)},\ t_{(l,1)},\ \cdots,\ t_{(l,p'_l)}], \quad l = 1, 2, \cdots, k,$$

  be the partial path already constructed before $s_{(i,j)}$ is chosen. By the greedy algorithm the value of $t_{(i,p'_i)}$ is the minimum among $\{t_{(l,p'_l)} : 1 \leq l \leq k\}$. Let

  $$T'_l = [t_{(l,p'_{l+1})},\ t_{(l,p'_{l+1})},\ \cdots,\ t_{(l,q_l)}], \quad l = 1, 2, \cdots, k.$$

  Therefore, $T_l = S'_l \cup T'_l$, $1 \leq l \leq k$. If $s_{(i,j)}$ is not any of $t_{(l,p'_{l+1})}$, we can simply replace $t_{(i,j)}$ by $s_{(i,j)}$ in T_i. T_i will still be a path since the interval for $s_{(i,j)}$ extends to the right more than $t_{(i,j)}$ does by the greedy algorithm. If $s_{(i,j)}$ is one of $t_{(l,p'_{l+1})}$, say, $t_{(i',p'_{i'+1})} = s_{i,j}$, then we can set $T_i = S'_i \cup T'_{i'}$ and $T_{i'} = S'_{i'} \cup T'_i$ and keep T_l the same for $l \neq i$, $l \neq i'$, and $1 \leq l \leq k$. After this transformation, T_i and $T_{i'}$ are still paths since the rightmost point of S'_i is no bigger than the rightmost point of $S'_{i'}$. $\square$

B A Fast Algorithm for the case of a 2D Polygon Interior World

We now restrict the robot's world to be the interior of a two-dimensional polygon of size n. We assume that vertices are the possible landmarks. In a real-world scenario, the vertices could correspond to vertical edges, detected by a vision or a range sensor. As we noted above, the minimum number of landmarks needed to guide robot navigation is different for different technologies used. First we show that our discussion can be restricted to the case $k \leq 3$, where k is the number of landmarks visible from any path location.

Lemma B.1 *At any point we can see at least three vertices and there are polygons with positions from which we cannot see more than three vertices.*

Proof: Consider a triangulation of the polygon world. Each point is in one of the triangles. The three vertices of the triangle are visible from that point. □

Suppose that the planned route R is polygonal and consists of m segments. We should show later that the restriction of the line being polygonal can be removed. One difficulty in obtaining an efficient algorithm is that one vertex may cast many (up to $O(m)$) window intervals in the planned route R. This would lead to an interval graph of nm nodes and $(nm)^2$ edges, which would lead to an algorithm of time complexity at least $(nm)^2$. We improve over this coarse result by applying GREEDY at the same time that we construct the interval graph. First, we construct the visibility graph of the polygon. From the origin, find all the visible vertices. By GREEDY, we want to find the vertex which covers the longest window interval along the route from the origin. In general, at each point we need to replace an old landmark, we find all the vertices visible from this point. Then, we choose the vertex which covers the longest window interval along the route from this point on. Each such operation takes time mn. Therefore, we have an algorithm of time $mn|S|$, where S is the solution and $|S| = \sum_{i=1}^{k}(1 + p_i)$ using the notation of Appendix A. Consider a triangulation of the polygon Δ. Within a triangle, there are three vertices which cover all the parts of the planned route inside the triangle. Let's cut the planned route R into pieces at intersection points with edges in the triangulation Δ. Let $p_\Delta(R)$ denote the total number of pieces of R. Let $p(R) = \max_{all\Delta} p_\Delta(R)$. Thus, we have $|S| \leq kp(R)$, for $k \leq 3$. Therefore, the time complexity of the algorithm is $O(mn|p(R)|)$. For shortest paths between two points, it is not difficult to see that $p(R) = O(n)$. Hence we conclude:

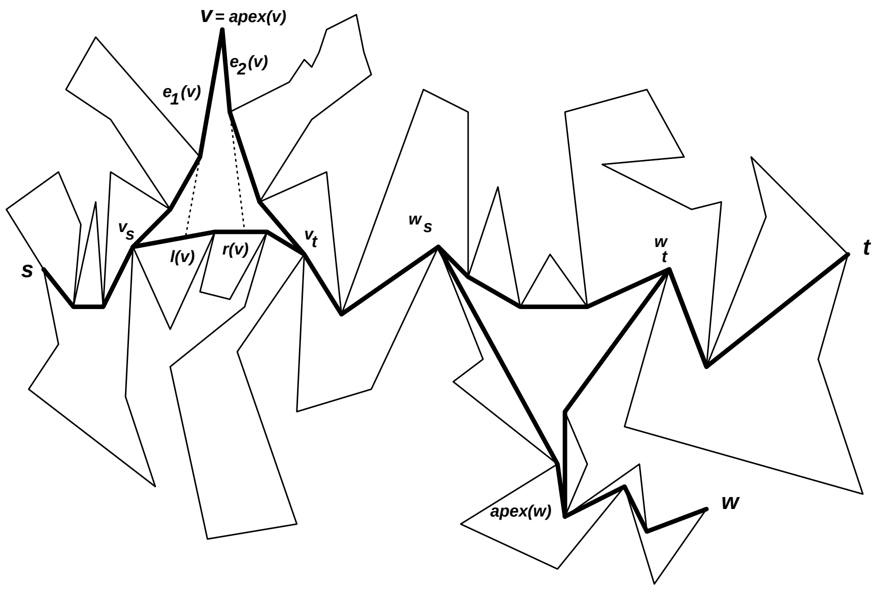


Figure 9: The window interval due to a landmark.

Lemma B.2 *There is an algorithm of time complexity $O(mn^2)$ for the uniform cost version of the landmark selection problem in two dimensions.*

However, for shortest paths between two points in a simple polygon, we can improve the result even further. An important observation is the following:

Lemma B.3 *Let $a = x_0 x_1 \cdots x_m = b$ be a shortest path between a and b inside a simple polygon. A point x visible from both a and b is visible from all the intermediate point x_i, $0 < i < m$.* $\square$

Let two vertices s and t be designated as the starting point and the target point of our robot, respectively. The robot will move from s to t along a *designated path* inside P. We assume the vertices of P are the landmarks, and a landmark v is visible from the current position p of the robot, if the line segment $\overline{vp}$ does not intersect the exterior of P. Let $\mathcal{L}(p)$ denote the set of landmarks scheduled for point p on the designated path. The greedy strategy mentioned earlier, suggests that the new visible landmark must be chosen from the landmarks not in $\mathcal{L}$ but visible from point p, so that it remains visible the longest along the designated path.

For each pair u, v of vertices of P, let $SP(u, v)$ denote the shortest path inside P from u to v. Here we assume the designated $st-$path is $SP(s, t)$. By Lemma B.3, any landmark can see only a connected subpath of $SP(s, t)$. Figure 9 characterizes this situation. Consider an arbitrary vertex

v of P. Let v_s be the vertex on the intersection of $SP(s,t)$ and $SP(s,v)$ closest to v. Similarly, let v_t denote the vertex on the intersection of $SP(t,s) = SP(s,t)$ and $SP(t,v)$ closes to v. The path $SP(v_s, v_t)$ is a subpath of $SP(s,t)$ from v_s to v_t and is convex towards v. Let $apex(v)$ denote the vertex common to both $SP(v, v_s)$ and $SP(v, v_t)$ farthest from v. Then, the paths $SP(apex(v), v_s)$ and $SP(apex(v), v_t)$ are also convex inward as suggested by Figure 9. We say a vertex v is a *useful* landmark if v can see a non-empty open interval of $SP(s,t)$. Otherwise, v is *useless.* A vertex v is useful if and only if $apex(v) = v$. In Figure 9 vertex v is useful, but vertex w is not. All vertices on $SP(s,t)$ are useful.

Let $l(v)$ and $r(v)$ denote the points (not necessarily vertices of P) of $SP(s,t)$ visible from v that are, respectively, closest and farthest from s. Then, $SP(l(v), r(v))$ is entirely visible from v and is a subpath of $SP(v_s, v_t)$, unless v is on $SP(s,t)$. Suppose v is a useful landmark not on $SP(s,t)$. Let $e_1(v)$ be the edge of $SP(v, v_s)$ incident to v and $e_2(v)$ be the edge of $SP(v, v_t)$ incident to v. Then, $l(v)$ is the intersection of $SP(s,t)$ with the extension of $e_1(v)$, and $r(v)$ is the intersection of $SP(s,t)$ with the extension of $e_2(v)$. (See Figure 9.) If v is on $SP(s,t)$, then $l(v)$ and $r(v)$ are the two vertices adjacent to v along $SP(s,t)$.

Let $T(x)$ denote the single source shortest path tree inside polygon P from the source x to each vertex of P. For any given vertex x of P, $T(x)$ can be computed in $O(n)$ time [10, 2, 11]. (For a much simpler, but $O(n \log n)$ time polygon triangulation algorithm, see [9].) Our algorithm to compute the intervals $(l(v), r(v))$ for each vertex v is as follows. We first compute the two single source shortest path trees $T(s)$ and $T(t)$. The shortest path $SP(s,t)$ is a path in both $T(s)$ and $T(t)$. By traversing these two trees, we can determine the set of useful vertices, and for each such vertex v, we can determine (v_s, v_t), and $(e_1(v), e_2(v))$. These can be done in a total of $O(n)$ time. We also store the vertices of $SP(s,t)$ in sorted order from s to t in an array so that we can perform binary search on some of its convex subchains. Now, for each useful vertex v, we perform a binary search on the convex subchain (v_s, v_t) of $SP(s,t)$ to determine the unique intersection points of the chain with the extensions of $e_1(v)$ and $e_2(v)$. These intersection points are $l(v)$ and $r(v)$ as desired, and can be computed in $O(\log n)$ time for each vertex v. We conclude:

Lemma B.4 *The endpoints of window intervals $(l(v), r(v))$, for all useful vertices v of P, in sorted order along $SP(s,t)$ can be computed in $O(n \log n)$ time and $O(n)$ space.*

Theorem B.5 *There is an algorithm of time complexity $O(n \log n)$ for the uniform case of the landmark selection problem for executing a shortest path in a simple polygon.*