Fast Planning for Dynamic Preferences

Brian D. Ziebart
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
bziebart@cs.cmu.edu

Anind K. Dey
Human-Computer Interaction Institute
Carnegie Mellon University
Pittsburgh, PA 15213
anind@cs.cmu.edu

J. Andrew Bagnell
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
dbagnell@ri.cmu.edu

Abstract

We present an algorithm that quickly finds optimal plans for unforeseen agent preferences within graph-based planning domains where actions have deterministic outcomes and action costs are linearly parameterized by preference parameters. We focus on vehicle route planning for drivers with personal trade-offs for different types of roads, and specifically on settings where these preferences are not known until planning time. We employ novel bounds (based on the triangle inequality and on the the concavity of the optimal plan cost in the space of preferences) to enable the reuse of previously computed optimal plans that are similar to the new plan preferences. The resulting lower bounds are employed to guide the search for the optimal plan up to 60 times more efficiently than previous methods.

Introduction

Planners must provide optimal plans quickly to meet the real-time demands of many planning applications. One effective approach for expeditiousness is the reuse of computations obtained from previous plans. For example, the D* algorithm (Stentz 1995) initially plans and then efficiently re-plans in dynamic environments by only recomputing portions of the search space where optimal plans are affected by action cost changes. When these cost changes are sparse, large efficiency benefits are obtained through this approach. However, if there are abundant cost changes, this approach is no better than replanning from scratch.

We focus on planning problems where the action costs are controlled by a parametric space of *preference weights* that trade-off between different cost factors. These preference weights will often differ from agent to agent, and may even change during plan execution as additional experience with the environment is obtained (Sofman *et al.* 2006), introducing potentially ubiquitous changes in action cost between *queries* (i.e., planning problems). Intuitively, if the preference weights change only slightly, then the previously optimal plan will still at least be *close* to optimal, but we desire plans with optimality guarantees.

The key insight of this paper is that previously computed optimal plans can be used to efficiently guide new optimal plan searches – even though their preference weights differ. We present an efficient algorithm based on A* search (Hart, Nilsson, & Raphael 1968). It employs previously computed optimal plans for differing preference weights to provide good lower bounds on plan costs for new preference weights. The bounds are based on the concavity of optimal plan cost in the space of preference weights. These concavity-based lower bounds guide the plan search towards the most promising plans. This guidance greatly reduces the search space and time required to obtain the optimal plan.

Our work is motivated by three applications. The primary motivation is personalized vehicle route planning, where each driver has unique preference weights (trading off travel time, distance, city street versus highway, etc.) and desires the driving route to a specified destination that is personally optimal. The second is in robotics domains where the terrain of the navigation environment is known, but the robot is unaware of its own abilities for navigating different terrains (e.g., mud, grass, hills, bushes). As the robot executes a preliminary plan, it learns its abilities in the different terrains (Sofman *et al.* 2006) and a better path can be planned with this knowledge. The last is the problems of imitation and inverse reinforcement learning (Ng & Russell 2000), where the agent's preference weights are recovered from demonstrated behavior of the agent. A number of approaches to this problem (Abbeel & Ng 2004; Ratliff, Bagnell, & Zinkevich 2006; Neu & Szepesvri 2007; Ramachandran & Amir 2007) repeatedly find optimal plans for differing preference weights. All three of these applications could benefit greatly by the improvements in speed that our approach provides.

We apply our algorithm to quickly plan personalized vehicle route recommendations across the road network of a large metropolitan city. We first consider the setting where only the preference weights and origin differ between plans, while the destination remains fixed. In this setting we find over an order of magnitude improvement compared to the previous state of the art approach. As a secondary contribution, we present a novel spatial optimal plan cost bound using the triangle inequality that expands upon an earlier approach (Goldberg & Harrelson 2005). Combining this spatial bound and our preference-based bound provides memory efficiency. We apply the resulting algorithm to the problem setting where origin, destination, and preference

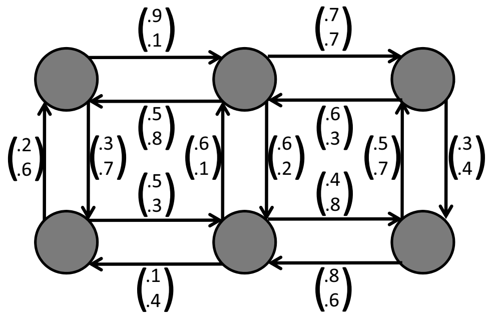


Figure 1: A parametric planning space with six states (circles) and 14 actions (directed lines). Each action has a feature vector with two features.

weights vary with each plan, and find large performance improvements.

In the remainder of the paper, we first provide some preliminary concepts and review existing work in fast graph-based planning and related problems from the computer science and operations research literature. We then present our bounds on plan cost for dynamic preference weights and describe an algorithm for efficiently computing them. For improved memory efficiency, we provide a novel spatially-based bound on optimal plan costs. Next, we illustrate the empirical benefits of our algorithm on the problem of personalized route planning. Finally, we discuss possible extensions to our work.

Preliminaries and Related Work

Many planning problems can be represented as weighted graphs, $G = (V, E)$, with each vertex ($v \in V$) representing a state (of an agent) and each edge ($e \in E$) representing an action that deterministically leads from one state to another. Edge weights, $w(e)$, are the *cost* of taking an action or, equivalently, the negative action utility or negative action reward. In this work, we focus on parameterized action costs (Equation 1), where actions are characterized by *action features*, $\mathbf{f}_e \in \Re^k$, and the cost of actions are linearly parameterized by a *preference weight*[1], $\theta \in \Re^k$, that trades off the utilities of different features. An example of this setting with two action features is shown in Figure 1.

$$w_\theta(e) = \theta^\top \mathbf{f}_e \tag{1}$$

Plans to reach a goal state (vertex v_{g}) from an initial state (vertex v_{i}) correspond to paths, $\zeta_{v_i \to v_g}$, through the graph. Plan costs are equivalent to path costs, $w_\theta(\zeta) = \sum_{e \in \zeta} w_\theta(e)$, which are the sum of the costs of actions in the path.

Research on efficiently finding the lowest cost path between two vertices in the graph for fixed θ has a long history. We denote this optimal path as $\zeta^*_{a\to b,\theta}$ and its cost as $w^*_{a\to b,\theta}$ with upper and lower bounds $w^+_{a\to b,\theta}$ and $w^-_{a\to b,\theta}$.

$$\zeta^*_{a\to b,\theta} = \operatorname*{argmin}_{\zeta_{a\to b}} w_\theta(\zeta_{a\to b})$$

$$w^*_{a\to b,\theta} = \min_{\zeta_{a\to b}} w_\theta(\zeta_{a\to b}) = w_\theta(\zeta^*_{a\to b,\theta})$$

$$w^+_{a\to b,\theta} \geq w^*_{a\to b,\theta} \geq w^-_{a\to b,\theta}$$

Dijkstra's algorithm (Dijkstra 1959) searches a tree of partial plans from the origin in the order of increasing partial plan weight until a path to the goal state is obtained, which is guaranteed to be optimal. The A* search algorithm (Hart, Nilsson, & Raphael 1968) is a *best-first search* that incorporates an underestimate of the remaining cost to the goal state. This underestimate is called an *admissible heuristic function*, $w^-_{v_x \to v_g}$. A* expands the search tree of optimal paths in the order of most promising partial plan first, which it estimates using the lower bound on that partial plan's cost to reach the goal, $w^*_{v_i\to v_x} + w^-_{v_x\to v_g}$. For spatially-embedded planning spaces, one such admissible heuristic function is obtained by multiplying the Euclidean distance between two points in the space by the smallest cost to travel a unit distance in the space. When the admissible heuristic function is *monotonic* (i.e., satisfies the triangle inequality), each node need be explored at most once and can be tracked using a closed set (Algorithm 1). The algorithm for non-monotonic admissible heuristic functions is similar, but can re-explore a vertex when a lower cost path to the vertex is found.

Algorithm 1 A* search algorithm

```
**A* search(graph G, initial state $v_i$, goal state $v_g$)**
Add partial path $\zeta = (v_i)$ to priority queue $Q$
Initialize *closed set* $C \leftarrow \{\}$
While $Q$ not empty
    Pop $\zeta_x$ (to $v_x$) with lowest $w^*_{v_i\to v_x} + w^-_{v_x\to v_g}$
    If $(v_x \in C)$ continue
    If $(v_x = v_g)$ return $\zeta_x$
    $C \leftarrow C \cup v_x$
    For each successor $v_y$ of $v_x$
        Add $\zeta_y = (\zeta_x, v_y)$ to $Q$
```

One approach for improving planning speeds is to obtain better bounds to guide A* searches. Goldberg & Harrelson (2005) employ the triangle inequality using precomputed optimal path costs to obtain tighter lower bounds for guiding vehicle route planning. One advantage of this approach is that if road costs only increase (e.g., accidents, construction, road closures), optimality is still guaranteed (Delling & Wagner 2007).

A second approach is specific to applications where the goal state is the same between queries, but some action costs change. For example, as a robot executes a plan it may sense its surroundings and refine the terrain map it uses for path planning, potentially changing the optimal plan. Intuitively, if the number of changes is small, only a limited portion of the space of optimal sub-plans may need to be updated. The D* algorithm (Stentz 1995) and various extensions (Koenig

[1]These weights may also be dynamic contextual variables of the domain (e.g., cost of gasoline) that are independent of the agent.

& Likhachev 2002; Ferguson & Stentz 2005; Likhachev *et al.* 2005) have been developed to reuse previous planning results when they are unaffected by changes in cost. These approaches have realized significant empirical performance benefits when the amount of change is limited.

A final approach is to restrict the search space during a pre-processing time, and then search only on the restricted search space. For example, large portions of the road network can be eliminated from consideration by determining all the optimal paths that pass through a particular area (Sanders & Schultes 2007). These approaches have been quite impressive in practice, reducing path planning times for continent-wide road networks to a few milliseconds. The approach has been extended to deal with changes in road segment cost (Schultes & Sanders 2007), and performs well as long as the amount of change is limited.

A similar idea can be employed in the space of preferences. In the *multicriteria shortest path* setting studied in operations research, edges have vectors of costs, but the preference weight, θ, is unknown. The problem is to find the *pareto-optimal* set of paths that contain exactly the paths that are optimal for any choice of θ. This subset of paths can then be searched more quickly than the complete class of paths. While some approaches to the *multi-criteria shortest path* problem are based on admissible heuristic functions (Refanidis & Vlahavas 2003) and A* search (Stewart & White 1991; Mandow & de-la Cruz 2005), there are a number of shortcomings to the general approach of precomputing the optimal paths for every possible preference weight. First, in general, finding the pareto-optimal set of paths is NP-hard (Hansen 1980), making it difficult to scale to large problem settings, like a large road network. Similarly, it is unclear how to obtain the subset of optimal paths efficiently for all origin states in the planning space, which is needed for vehicle route planning. Finally, unlike admissible heuristic-based search, path optimality is not guaranteed if edge costs increase.

In our problem setting, the preference weights are known at query time and the action costs are linearly parameterized by those weights. Most dynamic planning problems assume and are only effective when the number of action costs changing between queries is limited. In our setting, these action costs can all change since they are controlled by a preference weight, θ, that varies for each query. These changes in preference weight can make action costs lower, so approaches that rely on the heuristic function to remain a valid underestimate are also not applicable. Unlike the multi-criteria shortest path pre-processing approach, which finds all potentially optimal paths for any possible preference weight, our approach finds the optimal path for a particular preference weight provided at query-time and doesn't suffer from the same tractability issues.

Though the problem setting is significantly different, the concavity-based bounds we employ are similar to bounds used in Partially Observable Markov Decision Processes (POMDPs). In POMDPs, an agent's actions reveal information about hidden variables. The function of interest when planning actions in the POMDP is the optimal policy value in the space of beliefs over unobserved variables. This function is piece-wise linear and convex for finite MDPs (Kaelbling, Littman, & Cassandra 1998). The optimal policy values for a set of beliefs are computed (along with the gradient) and used to bound the optimal policy values of other portions of the belief space (Pineau, Gordon, & Thrun 2006). This approach yields solutions with bounded policy value losses and enables applicability to much larger problems than exact approaches can afford.

Preference Weight Bounds

We now derive bounds on the cost of the optimal path between two vertices (a and b), $w^*_{a\to b,\theta} = w_\theta(\zeta^*_{\theta,a\to b})$, for some preference weight, $\theta \in \Re^k$, based on a set of previous preference weights $\phi_1, \phi_2, ..., \phi_n \in \Re^k$, and their associated optimal path costs, $w^*_{a\to b,\phi_1}, w^*_{a\to b,\phi_2}, ..., w^*_{a\to b,\phi_n}$.

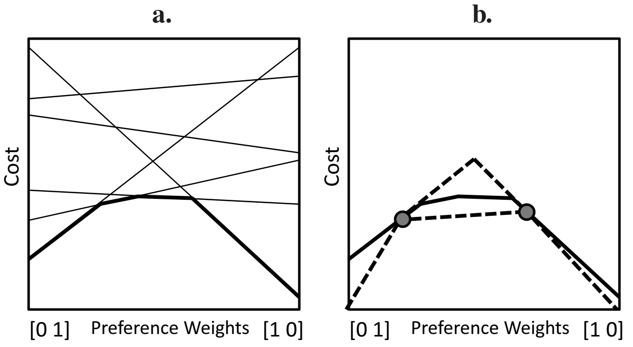


Figure 2: Six paths with costs evaluted over the preference simplex and the optimal path cost function (**a**). The optimal path cost function evaluated at two points (circles) and the resulting upper and lower bound functions (dotted lines) (**b**).

First, consider a single path, $\zeta_{a\to b}$, from vertex a to vertex b. Summing the path's action features yields a *path feature count*, $\mathbf{f}_{\zeta_{a\to b}} = \sum_{e\in\zeta_{a\to b}} \mathbf{f}_e \in \Re^k$. The cost of the path is obtained by applying a preference weight, θ, to the path feature count.

$$w_\theta(\zeta_{a\to b}) = \theta^\top \mathbf{f}_{\zeta_{a\to b}} \tag{2}$$

We restrict our consideration to preference weights and feature values that are non-negative. This constrains all path costs to be positive, and avoids the possibility of having negative cost cycles. Path optimality is invariant to positively scaling the preference weights, so without loss of generality, only the simplex of preference weights need to be considered (i.e., $\theta : \sum_i \theta_i = 1$). The cost of a single path is linear (Equation 2) over this simplex of preference weights, forming a hyperplane in the space (Figure 2a). The function of optimal path preference weights is the minimum cost of all paths from a to b evaluated at each point in the space (Figure 2a)[2].

$$w^*_{a\to b,\theta} = \min_{\zeta_{a\to b}} \theta^\top \mathbf{f}_{\zeta_{a\to b}} \tag{3}$$

[2]Our analysis and method also easily extend to paths that are not constrainted to terminate at a particular vertex.

The minimum of this set of linear functions is piecewise linear and also concave (Boyd & Vandenberghe 2004)[3]. The defining property of concave functions is that any linear interpolation between two points on the function creates a lower bound for the function. So if the optimal path costs have been obtained for two preference weights, a lower bound on any interpolation of those preference weights can be easily obtained. More generally, for $\alpha_i \geq 0$ and points ϕ_i on the $(k-1)$-simplex of preferences:

$$\sum_i \alpha_i w^*_{a\to b,\phi_i} \leq w^*_{a\to b,\sum_i \alpha_i\phi_i} \tag{4}$$

This provides a useful lower bound on $w^*_{a\to b,\theta}$ for some new preference weight θ using previously evaluted optimal path costs for *basis preference weights* ϕ_i and $\alpha_i \geq 0 : \sum_i \alpha_i\phi_i = \theta$. In Figure 2b, the lower bound is shown using two basis preference weights.

Upper bounds are simpler to obtain. Evaluating any path at θ provides an upper bound. The tightest upper bound can be obtained by taking the minimum over all those evaluations: $\min_{\phi_i} \theta^\top \mathbf{f}_{\zeta^*_{a\to b,\phi_i}}$. Figure 2b shows the upper bound obtained by "extending" the planes evaluated at each basis preference weight. If the same path is optimal in other portions of the preference weight space, the upper bound and optimal path cost functions will match in those places.

Fast Dynamic Preference Algorithms

In practice, optimizing to minimize the average query time is a trade-off between obtaining tighter bounds that guide the search (Algorithm 1) with better focus, minimizing the time to compute those bounds at query time so the overall search will be fast, and restricting the memory and time requirements for generating and storing information to compute those bounds during pre-processing.

We focus on optimizing for the problem of finding optimal routes for vehicles in the road network of a major city. At query time, we bound plan (i.e., path) costs using the costs of precomputed optimal plans for similar preference weights.

Precomputation

During precomputation, our objective is to obtain optimal path costs from each state to the goal state for varying preference values so that the bound at query time will be reasonably tight. The basic algorithm is shown below.

The first step is choosing a set of basis preference weights. For reasons we detail in the following subsection, including the principle preference vectors (e.g., $[0\ 1]^\top$ and $[1\ 0]^\top$ for 2 dimensional preferences) is useful. The remaining basis preference weights can be selected in a number of different ways. Two simple approaches are to select them randomly or systematically to evenly cover the preference space.

The second step of the algorithm is accomplished using Dijkstra's algorithm to find the shortest paths from each edge in the network to the goal state. This is repeated for

Algorithm 2 Fixed goal pre-processing algorithm

```
**Pre-process(parametric graph G, goal state $v_g$)**
1. Choose basis preference weight set $\Phi = \{\phi_1, \phi_2, ...\}$
2. $\forall_{\text{edge } x,\phi_i}$ compute and store $w^*_{x\to v_g,\phi_i}$
3. Construct searchable data structure of $\Phi$
```

each basis preference weight, ϕ_i, yielding a running time of $O(|\Phi||E|\log|V|)$ for problems with $|E|$ actions and $|V|$ states, which dominates the other precomputation steps.

In the final step of the algorithm, the algorithm constructs a *k*d-tree (Bentley 1975) that enables finding suitable basis preference weights quickly at query time. The *k*d-tree works by repeatedly splitting the k-dimensional space of preferences into a search tree that has a height that is logarithmic in the number of basis preference weights. Its usage is described in the following subsection.

Query Time

At query time, the planner is provided the origin state v_i and a preference weight (θ), and must find an optimal plan from v_i to v_g under preference weight θ. It employs A* search (Hart, Nilsson, & Raphael 1968) to explore the most promising partial paths first, using a lower bound (based on the precomputed optimal path costs) to estimate the remaining cost to the goal (Algorithm 1). The basic algorithm for computing the lower bound is shown below.

Algorithm 3 Fixed goal lower bound computation algorithm

```
**Heuristic(state $v_x$, preference weight $\theta$)**
1. If $\alpha$ uninitialized, choose $\alpha : \alpha_i \geq 0, \sum_i \alpha_i\phi_i = \theta$
2. Return $\sum_i \alpha_i w^*_{v_x\to v_g,\phi_i}$
```

The first step of the algorithm is finding basis weights, α. There are many ways to choose basis weights α so that $\sum_i \alpha_i\phi_i = \theta$. Any choice of α satisfying this equality will yield a lower bound, however the tightness of the bound will vary. The tightest bound for each vertex v_x explored in the search for the optimal path to v_g can be obtained by solving the Linear Program (LP) shown in Equation 5.

$$\max_\alpha \sum_i \alpha_i w^*_{v_x\to v_g,\phi_i} \tag{5}$$
$$\text{such that: } \sum_i \alpha_i\phi_i = \theta \text{ and } \alpha_i \geq 0$$

However, solving this LP at each step of the A* search is not efficient for our domain – the search space is minimized, but the benefits of the minimization do not outweigh the added time required to solve each LP. Instead, we find a single

[3]The minimum of a set of concave functions is concave, so our method is more generally applicable to concave cost functions.

set of basis preference weights for the entire query, rather than for each v_x, and instead of solving the LP, we use a greedy approximate algorithm that is much faster. As we show experimentally, this approximation still provides very tight bounds.

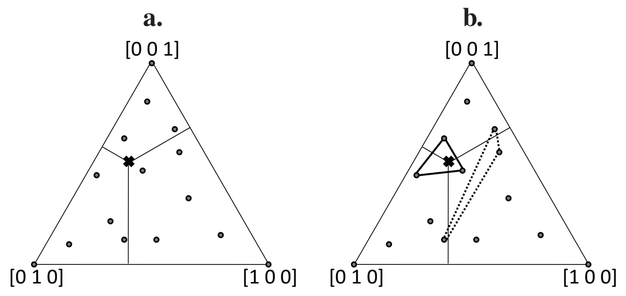


Figure 3: A preference simplex with previously evaluated preference bases (circles), a new query (x), and query-dependent subdivision of the preference space (**a**). A valid interpolation plane (solid triangle) and an invalid interpolation plane (dotted triangle) using preference bases (**b**).

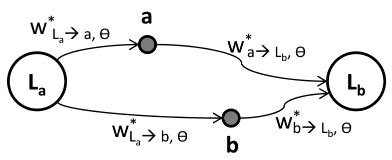
The problem characteristics of selecting basis weights are shown in Figure 3. In our approach, we divide the preference weight space into regions using the proportions of the new query preference weights as boundaries (Figure 3a). We restrict our set of basis preference weights to contain one basis weight from each of those regions. Unfortunately, this constraint alone does not yield a valid interpolation plane for our new query preference weight (i.e., one containing the new preference weights), as shown in Figure 3b. Starting with the principle preference weights (i.e., simplex corners), we iteratively find the closest basis preference weight from each region that can be included in the set while maintaining validity. We use a *k*d-tree initialized in the pre-processing algorithm to efficiently find these closest preference weights. A valid set of preference weights is guaranteed to be found by the algorithm as long as the principle preference weights are included in the basis preference weight set.

Spatial Bounds

In many domains, the goal state varies between queries. The previously described method can be applied independently for each goal state, but this approach can often lead to memory requirements that are prohibitive for large graphs. An alternative with lower memory requirements is to employ spatial bounds in addition to utility weight bounds. It's important to note that unlike Euclidean-based lower bounds, the bounds we describe do not require the planning domain to have additional geometric information available.

ALT Bounds

Goldberg & Harrelson (2005) combine A* search with Landmarks and the Triangle inequality in the ALT heuristic. A small number of landmarks (e.g., 16) are chosen in the graph. The optimal path cost between each point and each landmark in the planning graph is pre-computed. At query time, the costs of optimal paths between a landmark (L_a or L_b) and two points (a and b) are employed to compute a lower bound on the optimal path cost between those points using the triangle inequality.



$w^*_{L_a \to a, \theta}$
a
$w^*_{a \to L_b, \theta}$
L_a
L_b
$w^*_{L_a \to b, \theta}$
b
$w^*_{b \to L_b, \theta}$

Figure 4: Landmark-based bounds on the cost between a and b using costs from landmark L_a or to landmark L_b.

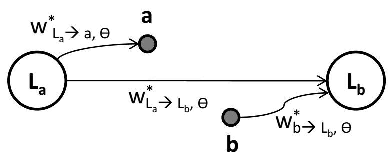
The precomputed optimal path costs are shown in Figure 4. The optimal path cost from L_a to a to b can be no lower than the optimal path cost from L_a to b, and the optimal path cost from a to b to L_b can be no lower than the optimal path cost from a to L_b.

$$\begin{aligned} w^*_{a\to b,\theta} &\geq w^*_{L_a\to b,\theta} - w^*_{L_a\to a,\theta} \\ w^*_{a\to b,\theta} &\geq w^*_{a\to L_b,\theta} - w^*_{b\to L_b,\theta} \end{aligned} \tag{6}$$

When the number of landmarks is fixed to a small constant number, the memory requirements remain linear in the size of the planning space and the approach can be applied to large networks. Goldberg & Harrelson (2005) are limited to 16 landmarks due to memory constraints and obtain faster overall query times by only considering a query-dependent subset of 3 or 4 of those landmarks.

ALT^2 Bounds

We now introduce our dual landmark-based bounding approach, which we call ALT^2 since it is a natural extension of the single landmark-based approach. Unlike ALT, which at query time selects a good subset of the 16 landmarks on which to base its bounds, ALT^2 chooses a fixed subset of landmarks at pre-processing time for each state in the planning graph. In the ALT^2 approach, optimal costs are precomputed between all pairs of landmarks and between each non-landmark and a constant-sized set of its nearby landmarks. For a fixed set of landmarks, the bounds are looser than the single landmark approach, but this approach enables a much larger number of landmarks to be incorporated (e.g., 2000 vs. 10), ultimately providing faster planning.



$w^*_{L_a \to a, \theta}$
a
L_a
L_b
$w^*_{L_a \to L_b, \theta}$
b
$w^*_{b \to L_b, \theta}$

Figure 5: Landmark-based bounds on the cost between a and b using costs from landmark L_a or to landmark L_b.

The bounds for the dual landmark-based approach are based on landmarks L_a and L_b close to points a and b. Intuitively, the bound is tight when points a and b lie on the optimal path between L_a and L_b.

$$w^*_{a\to b,\theta} \geq w^*_{L_a\to L_b,\theta} - w^*_{L_a\to a,\theta} - w^*_{b\to L_b,\theta} \quad (7)$$

If L_a and L_b are the same landmark, the single landmark bound (Equation 6) can be employed.

Consider a dual landmark bound where each of the N_{action} edges in the planning space has N_{local} local landmarks chosen from a total of N_{landmark} landmarks. The memory requirements for the dual landmark approach are: $O(N_{\text{action}}N_{\text{local}} + N^2_{\text{landmark}})$. The single landmark approach can be viewed as a special case of the dual landmark approach with $N_{\text{local}} = N_{\text{landmark}}$ that only uses the single landmark bounds of Equation 6. Its memory requirements are $O(N_{\text{action}}N_{\text{landmark}})$. If N_{local} is kept small (e.g., 1), then for the same amount of memory that the ALT approach uses for 16 landmarks in large planning graphs (e.g., $N_{\text{action}} > 1,000,000$), the ALT^2 approach can use thousands of landmarks. However, unlike the ALT bounds, the ALT^2-based bounds are non-monotonic (i.e., they may not satisfy the triangle inequality) and may require exploring a previously explored node again during the A* search.

Combining Spatial and Preference Bounds

We now present two different approaches for combining preference bounds and spatial bounds. The first uses spatial bounds within the preference bound. Note that the preference bound (Equation 4) holds when using underestimates for the cost of basis-dependent optimal paths (with appropriately chosen ϕ_i and α as described previously).

$$\sum_i \alpha_i w^-_{a\to b,\phi_i} \leq w^*_{a\to b,\theta} \quad (8)$$

Spatial bounds (Equation 6 or Equation 7) can be used for obtaining these underestimates. The combination yields our overall bounds. The combined preference-dual-landmark bound is shown in Equation 9.

$$\sum_i \alpha_i \left(w^*_{L_a\to L_b,\phi_i} - w^*_{L_a\to a,\phi_i} - w^*_{b\to L_b,\phi_i}\right) \leq w^*_{a\to b,\theta} \quad (9)$$

The second approach is to employ preference bounds within the spatial bound. The spatial bounds (Equation 6 or Equation 7) can first be relaxed using appropriate underestimates and overestimates.

$$w^*_{a\to b,\theta} \geq w^-_{L_a\to a,\theta} - w^+_{L_a\to b} \quad (10)$$

Preference bounds are then employed as those underestimates and overestimates. We show the single-landmark-preference bound (incorporating Equation 6) in Equation 11. The dual-landmark-preference bound is similarly obtained.

$$w^*_{a\to b,\theta} \geq \left(\sum_i \alpha_i w^*_{L_a\to a,\phi}\right) - \min_{\phi_i} \theta^\top \mathbf{f}_{\zeta^*_{L_a\to a,\phi_i}} \quad (11)$$

The algorithms for utilizing these bounds are very similar to Algorithm 2 and Algorithm 3. During pre-processing time, the necessary optimal path values and optimal path features are computed (e.g., landmark-based optimal paths for different preference weights). At query-time the lower bound is computed using those precomputed values.

Fast Route Planning Evaluation

We evaluate different variants of our approach and comparative baseline approaches on the problem of personalized vehicle route planning.

Experimental Setup

We evaluate our approach on the road network of Pittsburgh, Pennsylvania, which is one of the 25 largest metropolitan areas in the United States. The network contains over 300,000 nodes (road segments) and 900,000 edges (transitions between road segments). Though less compact than treating intersections as nodes and road segments as edges, this representation allows costs for different transitions at intersections. A small portion of the road network is shown in Figure 6. Note that without good search heuristics, the density of road segments in certain areas makes the search space for optimal paths very large and time-consuming to explore.

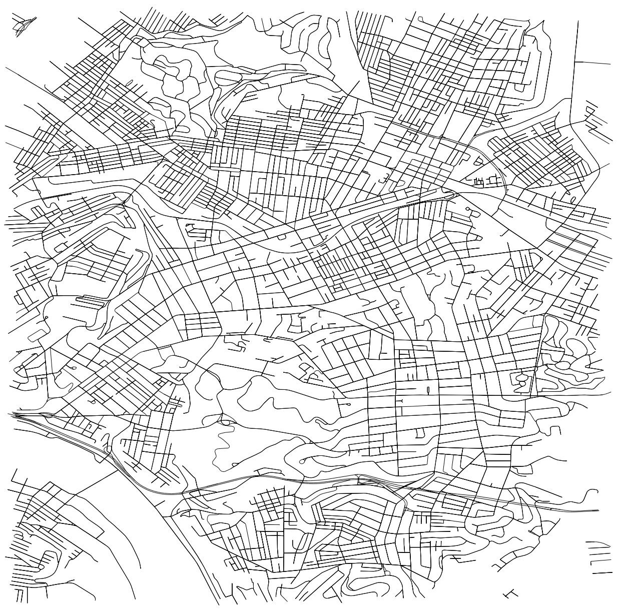


Figure 6: A small portion of the road network covering 1/256th of the total road network's area.

We employ three different cost factors that are traded off differently with each query. They are:

- Path distance
- Travel time estimate
- City street and local road distance

To allow each factor to have an equally strong influence, we scale the feature values of each factor so that the average values are equal. As can be imagined, preferentially minimizing different combinations of these factors yield vastly different routes between pairs of endpoints that can be extremely sensitive to the trade-off between cost factors.

For our experiments, the parameters of each query are chosen uniformly at random from the portion of the simplex of preferences where the travel time weight is greater than

0.5. Origins and destinations are chosen uniformly at random from the road network's road segments as well. Landmarks and basis preference weights for our experiments are chosen similarly. We plan for all pairs of 100 starting locations and 100 destinations – each with a different randomly chosen preference weight, and report averaged execution statistics.

All experiments were conducted on a single 2.33GHz Intel Xeon processor with 4MB cache size and 24GB total memory. Algorithms were implemented in C++ and compiled with g++ at optimization level 3. We used double precision (8 bytes) for all cost calculations and comparisons and a simple heap-based priority queue within our A* search.

Evaluation Metrics

Quickly providing optimal personalized routes is our primary concern. For this purpose, we measure the average optimal route planning time. We also measure machine-independent metrics that are also independent of data structure implementations. The first is the *efficiency* of the plan search, which Goldberg & Harrelson (2005) define as the ratio of path length to total nodes explored in the search. At 100% efficiency, only components of the optimal path are searched. The second metric we employ is the *tightness* of the heuristic function, which is the ratio of the optimal path cost to the heuristic function's estimate of the path cost. At a value of 100%, the heuristic function is a perfect estimate of the optimal path cost.

Fixed Destination Experiments

We first consider the scenario where the destination of queries is fixed and known by the planner during its preprocessing time, but the origin and preference weights are provided at query time.

Model	**Efficiency**	**Tightness**	**Time**
Dijkstra's	0.19%	0%	315ms
Euclidean A*	0.47%	46.2%	217ms
Preference A* (3)	22.7%	92.7%	14.9ms
Preference A* (5)	34.3%	95.0%	11.3ms
Preference A* (7)	50.0%	96.7%	8.7ms
Preference A* (10)	61.2%	97.9%	6.9ms
Preference A* (15)	68.7%	98.4%	4.2ms
Preference A* (25)	75.1%	98.7%	3.7ms

Table 1: Fixed destination evaluation results.

The preference-based bounds provide much better performance on all evaluation metrics, as shown in Table 1. The number of basis preference weights is shown in parentheses. All measures improve with the number of basis weights. In particular, the mean planning time shows significant improvement over Dijkstra's algorithm and Euclidean-based A* search, which are almost two orders of magnitude worse than the best preference-based approach.

Spatial Bounds Experiments

When the destination state is also unknown, the number of pairs of intersections in our road network (i.e., $\approx 10^{11}$) makes applying our fixed destination methods at each destination prohibitively memory and computationally expensive. Instead, we employ memory-efficient spatial bounds in addition to our preference weight bounds.

Bases, Landmarks	**Effic.** (%)	**Tight** (%)	**Time** (ms)	**Space** (MB)
5,10	2.31%	74.8%	113	240
5,20	2.62%	80.4%	96	480
5,30	2.75%	81.7%	94	720
5,40	3.46%	84.0%	89	960
10,10	2.71%	76.4%	99	480
10,20	3.84%	84.2%	74	960
10,30	4.79%	86.6%	66	1440
10,40	5.70%	88.5%	61	1920
Dijkstra's	0.19%	0%	315	0
Euclidean A*	0.47%	46.2%	217	0

Table 2: Combined preference and ALT evaluation results.

We bound the optimal plan costs using the ALT bound within the preference-based bounds (ALT version of Equation 9). Our algorithm automatically selects three landmarks to use for all estimates in a single query (Goldberg & Harrelson 2005), which provides faster planning times than any other number of landmarks. The resulting planning metrics are shown in Table 2 for different numbers of preference basis weights and landmarks. While the spatial bounds are much looser than the preference-based bounds, their combination still yields significantly better planning times than Dijkstra's algorithm and Euclidean-based A*.

Bases, Landmarks	**Effic.** (%)	**Tight** (%)	**Time** (ms)	**Space** (MB)
5,1000	1.22%	77.3%	96	65
5,2000	1.84%	82.1%	65	185
5,3000	2.30%	84.1%	54	385
5,4000	2.76%	85.0%	49	665
5,5000	3.08%	86.2%	41	1025
10,1000	1.42%	79.8%	83	130
10,2000	2.27%	84.6%	53	370
10,3000	2.91%	86.6%	42	770
10,4000	3.52%	87.6%	38	1330
10,5000	3.95%	88.6%	31	2050

Table 3: Combined preference and ALT^2 evaluation results.

In Table 3, we evaluate the combination of preference-based bounds and ALT^2 bounds (Equation 9). We employ a single local landmark, which we found provides the best average planning time. Compared to the ALT bounds, the ALT^2 bounds provide faster planning while requiring a smaller amount of space. This is despite the fact that ALT^2 bounds are not monotonic (i.e., could violate the triangle inequality), which may necessitate revisiting nodes in the A* search. However, the ALT^2 bounds do not provide corresponding performance improvements on average efficiency and tightness in comparison with the ALT bounds.

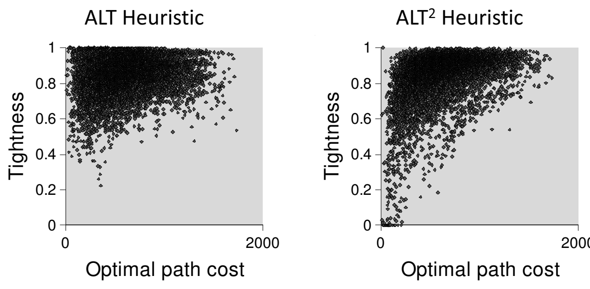


Figure 7: Scatterplots of bound tightness and optimal path cost for ALT and ALT^2.

The reason for this discrepancy is illuminated by Figure 7. The ALT^2 heuristic is generally less tight on predictions for short paths than the ALT heuristic, but better on longer paths. This translates to faster planning times for ALT^2 because the planning time cost for less tightness on shorter paths is much less severe than the additional planning time required for less tightness on longer paths.

Conclusion and Future Work

In this paper, we presented an approach for quickly finding optimal plans when costs vary according to a parametric preference weight. We employed both preference-based bounds and memory-efficient spatial-based bounds to provide good estimates of the remaining costs to reach a goal state. We applied this approach to the problem of personalized route recommendation and showed large performance improvements over previous state of the art planning approaches to the problem.

We are working to create a preference-dependent route recommendation web service powered by this approach. We are also investigating methods for automatically analyzing the range of preferences that drivers demonstrate using GPS traces of their driving patterns. This will help focus our efforts on relevant portions of the preference space. Additionally, we plan to incorporate more efficient data structures (e.g., the Fibonacci heap (Fredman & Tarjan 1987)) and additional search techniques (e.g., bi-directional techniques (Pohl 1971)) to obtain further performance improvements. Future work will also focus on incorporating both preference changes and local action cost changes into a fast planning framework. This will allow real-time factors on travel like accidents and road construction to be incorporated.

Acknowledgments

The authors thank the Quality of Life Technology Center/National Science Foundation for supporting this research under Grant No. EEEC-0540865.

References

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proc. ICML*, 1–8.

Bentley, J. L. 1975. Multidimensional binary search trees used for associative searching. *Commun. ACM* 18(9):509–517.

Boyd, S., and Vandenberghe, L. 2004. *Convex Optimization*. Cambridge University Press.

Delling, D., and Wagner, D. 2007. Landmark-based routing in dynamic graphs. In *WEA*, 52–65.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1:269–271.

Ferguson, D., and Stentz, A. T. 2005. The delayed d* algorithm for efficient path replanning. In *Proc. ICRA*, 2045 – 2050.

Fredman, M. L., and Tarjan, R. E. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM* 34(3):596–615.

Goldberg, A. V., and Harrelson, C. 2005. Computing the shortest path: A* search meets graph theory. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 156–165.

Hansen, P. 1980. Bicriterion path problems. *Multiple Criteria Decision Making: Theory and Application* 177:109–127.

Hart, P.; Nilsson, N.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 4:100–107.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artif. Intell.* 101(1-2):99–134.

Koenig, S., and Likhachev, M. 2002. Improved fast replanning for robot navigation in unknown terrain. In *Proc. ICRA*.

Likhachev, M.; Ferguson, D.; Gordon, G.; Stentz, A. T.; and Thrun, S. 2005. Anytime dynamic a*: An anytime, replanning algorithm. In *Proc. ICAPS*.

Mandow, L., and de-la Cruz, J.-L. P. 2005. A new approach to multiobjective a* search. In *Proc. IJCAI*, 218–223.

Neu, G., and Szepesvri, C. 2007. Apprenticeship learning using inverse reinforcement learning and gradient methods. In *Proc. UAI*, 295–302.

Ng, A. Y., and Russell, S. 2000. Algorithms for inverse reinforcement learning. In *Proc. ICML*, 663–670.

Pineau, J.; Gordon, G.; and Thrun, S. 2006. Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research* 27:335–380.

Pohl, I. 1971. Bi-directional search. *Machine Intelligence* 127–140.

Ramachandran, D., and Amir, E. 2007. Bayesian inverse reinforcement learning. In *Proc. IJCAI*, 2586–2591.

Ratliff, N.; Bagnell, J. A.; and Zinkevich, M. 2006. Maximum margin planning. In *Proc. ICML*, 729–736.

Refanidis, I., and Vlahavas, I. P. 2003. Multiobjective heuristic state-space planning. *Artif. Intell.* 145(1-2):1–32.

Sanders, P., and Schultes, D. 2007. Engineering fast route planning algorithms. In *WEA*, 23–36.

Schultes, D., and Sanders, P. 2007. Dynamic highway-node routing. In *WEA*, 66–79.

Sofman, B.; Ratliff, E. L.; Bagnell, J. D.; Cole, J.; Vandapel, N.; and Stentz, A. T. 2006. Improving robot navigation through self-supervised online learning. *Journal of Field Robotics* 23(12).

Stentz, A. T. 1995. The focussed d* algorithm for real-time replanning. In *Proc. IJCAI*.

Stewart, B. S., and White, III, C. C. 1991. Multiobjective a*. *J. ACM* 38(4):775–814.