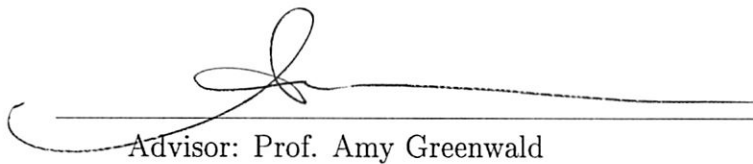


Exploration of the Routing Task Domain

by  
Christian Matthew Leroy

Submitted in partial fulfillment of the requirements for the Degree of Master  
of Science in the Department of Computer Science at Brown University

 5/14/03  
Advisor: Prof. Amy Greenwald Date

May 2003

# 1 Abstract

In this work, we provide the framework and justification for a route solving system that maximizes the utility of the routes it returns across a population of users. We show by comparing the properties of the extreme solutions of routing tasks that multiple routes exist which different users, or the same user at different times, may consider to be the best solution to a given task. We present a method for exploring the space in between these extremes based on building a distribution of multiattribute utility functions and solving for the set of optimal routes with respect to that distribution. We characterize this distribution as representing either a population of users with different preferences, or the different sets of preferences a single user may have at different times. We demonstrate that the choices we make in building this distribution allow us to control the type of solution routes that we discover. We also give two algorithms for reducing the size of the set of solution routes to a level appropriate for output to the user.

# 2 Introduction

Commercial websites providing driving directions have existed for a number of years and have become a popular tool for many web-users. In essence, they answer the question “how can I get from here to there?” Typically, interaction involves the entering of a start and end location by the user, followed by the return of a route as either a map or step-by-step directions by the system. While the responses to this question are often sufficient, those provided by a system that answers the question “what is the best way to get from here to there?” would be preferable.

This question is inherently subjective; which route is the “best” depends on the user’s preferences. As such, for a system to find the “best” route, it becomes necessary to model user preferences in some way. The decision-theoretic approach is to identify the important attributes of routes, for example their lengths and complexities, and to formalize the tradeoffs between these attributes. This is done through a multiattribute utility function taking the form of a weighted linear combination of single attribute value functions [4]. Most often, the value functions are thought to map attributes of differing scales into some normal utility space, and a unit weight vector to encode the tradeoffs between these mapped values. A set of tradeoffs such as,

“I am willing to go an extra mile to save two minutes of travel time”, might be captured as a higher weight value associated with the time attribute than the distance attribute.

For a problem as complex as determining the “best” route for a given task, it is unrealistic to assume that a single utility function, expressible as one set of tradeoffs, can express the preferences of a user at all times. Factors otherwise outside the bounds of a route-solving system can influence what type of route a user is looking for at any given time. A user’s mood, the condition of her car, or whether her mother-in-law got in early at the airport, may all influence her preferences at the time of her request, but it is unreasonable to assume that any system could model these directly in a single utility function. We address this by solving her task using multiple utility functions, each expressing different tradeoffs between the route’s attributes, in the hope that her current preferences will be captured by one of these and reflected in our solution. This method can be likened to modeling all a user’s possible preferences simultaneously. Understood in another way, we forgo modeling individuals, and instead use our distribution of functions to model an entire population of users, with the implication being that a user with preferences that differ between interactions with our system is for all practical purposes indistinguishable from multiple users with different fixed preferences.

Solving a single routing problem with a set of utility functions results in a set of solutions. This set is the collection of the optimal routes of each sampled member of our modeled population. If our system treated this route set as its final output, the user would be faced with the additional task of selecting a final route from it. Depending on the number of utility functions we use, this solution set can be quite large, and even for small sets, it may be undesirable to present it whole to the user. This might be the case when some or even all members of this solution set are exceedingly similar (although just different enough to register some difference in utility during generation). In such cases, if we choose to present only a single representative of all the similar routes, the savings in the user’s time during the final selection process may be worth a small loss in utility across our population. We impose the restriction that, although we still output a set of routes, there is some small limit as to the number of routes we will allow our system to return. Our system therefore must address the additional concern of determining a subset of the initial solutions that is as small and diverse as possible, while maintaining a high utility across our entire population.

In this work, we limit ourselves to the study of sets of utility functions produced by uniformly sampling the space of our users’ preferences. If we sample with sufficient frequency, our set of utility functions should capture all possible user preferences. We liken this to an exhaustive search for all, or nearly all, potential “best” solutions to a given task. In future work, we plan on studying preference distributions obtained from observed user-data: from offline sources such as surveys, or online methods such as prompting users for their preferred routes and using these to solve for their implicit preferences. We may also explore passive methods of collecting information, such as cataloging system interactions for continuous refinement of our model of our user population. The realistic distributions we could obtain from these methods would limit our search space, prune our initial solution set of routes that no real person would prefer, and result in a faster system with potentially better output route sets.

Formally, this work addresses the following problem: given a distribution of multiattribute utility functions, a routing task, and a constraint on the size of the output set, how can we find a set of solutions to the input task, within our size constraint, that maximizes the utility of the function set as a whole. Our approach is to solve directly for the optimal route for each function using Dijkstra’s algorithm, then to perform a number of selection algorithms to reduce the size of the output set. First though, we explore the properties of the solution space of a given task in two experiments. In the first, we establish by exploring their borders, that the solution spaces for most tasks are nontrivial. We then discuss the details of using a distribution of utility functions to explore the space’s interior. Our second experiment presents the results of this exploration pertaining both to the effectiveness of our user modeling system, and to the observed characteristics of the space. Finally, we present two algorithms that, given a set of routes and the utility functions that generated them, determine a subset of routes within a given size constraint to return to the user.

### 3 The Routing Task Domain

We consider a *route* to be a series of connected, non-branching segments with a well-defined start and end location. For driving directions, segments may correspond to roads or pieces of roads, and the start and end locations could be addresses or landmarks. A *routing task* is any problem, as specified



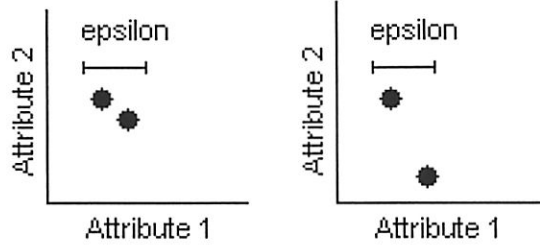


Figure 1: Effectively trivial solution spaces and  $\epsilon$ -dominance

by a start and end location, to which a route is a solution. We describe routes in terms of 4-dimensions: *length*, *time*, *complexity* and *scenicness*, while the segments of which they are made have a length, time, street name and an indicator of whether or not they are scenic. While routes can be described by many characteristics that are collective combinations of the characteristics of their segments, they may also have global properties that their individual segments do not. For example, while the length of a route is simply the sum of the lengths of its segments, the complexity of a route is the number of consecutive segments it contains that differ in name, plus one. This corresponds to the number of turns from one street to another in the route, ensures that all routes have a complexity of at least one, and does not depend on any notion of the complexity of an individual segment, which we do not define. The time of a route is the sum of the times of its segments. The scenicness of a route is the sum of the lengths of all a route's scenic segments.

Many of our analyses and algorithms are geometric in design. We will often speak of a route, via its description, as a point in  $n$ -dimensional space and operate on it as such. While we have limited our description of a route to four dimensions, it is easy to imagine other potentially useful characteristics. We have therefore avoided making any assumptions that would prevent expanding the dimensionality of our route description. Additionally, all our considerations of and operations on routes employ this description, and thus we consider two routes with the same description to be identical. We may eventually consider the real-world geometric differences between routes as a further source of characterization, but we leave that for future work.

We call the set of possible preferable solutions to a task, the *solution space* of that task. We say the “preferable” solutions, because we do not include a route in a task’s solution space if it is dominated by any other route that solves the task. A route is said to *dominate* another if it is the same or better in every dimension of its description. This can be expressed as an assumption that “all things being equal, a user will prefer a route that is shorter, faster, simpler, or more scenic”. To say that a route is preferable means that some user, with a particular set of preferences, could think that the route was the “best” solution to a particular task.

We call a solution space trivial if it is made up of a single route. If a task’s solution space is trivial, some solution route exists that dominates all other solving routes. At times, we will describe a task’s solution space as being “effectively” trivial. In this case, a single route exists that  $\epsilon$ -dominates all other solving routes. We say a route  $\epsilon$ -dominates another route if adjustments of its description components by no more than some small factor  $\epsilon$  allow it to dominate that route. Figure 1 illustrates two cases of  $\epsilon$  dominance in effectively trivial solution spaces. The dots are routes plotted according to their descriptions in 2-dimensional attribute space. On the left, a task has two preferable solutions that are both  $\epsilon$ -dominant by nature of their similarity. On the right, the lower route  $\epsilon$ -dominates the upper as it is much better in attribute two and only slightly worse in attribute one.

Because the solution space is made up of routes, entities that either exist or do not as a result of real-world street geometry, we call it discrete. We shall see the implications of this property in later sections, the most important being that although there may be “room” in a task’s solution space for a preferable route, a range of description values that would be dominated by no other solution routes, there is no guarantee that a route exists to fill it.

## 4 The Solution Space of Routing Tasks

We demonstrate in this section that the solution spaces of many routing tasks are not trivial by showing that the spaces’ borders, as defined by the tasks’ extreme solutions, are sufficiently separated in some and usually many dimensions. This guarantees that at least a small diverse set of solution routes exists, and implies one potentially much larger. This implication, confirmed in our exploration of a task’s non-extreme, or interior, solutions, tells us that there are hybrid routes to be found by our distribution of functions that are

```

LENGTHTHRESHOLD(shortest, fastest, simplest, threshold)
1  lthresh  $\leftarrow (1.0 + \text{threshold}) * \text{shortest.length}$ 
2  if max(fastest.length, simplest.length) > lthresh
3    then true
4    else false

```

Figure 2: Simple percentage length thresholding symmilarity metric

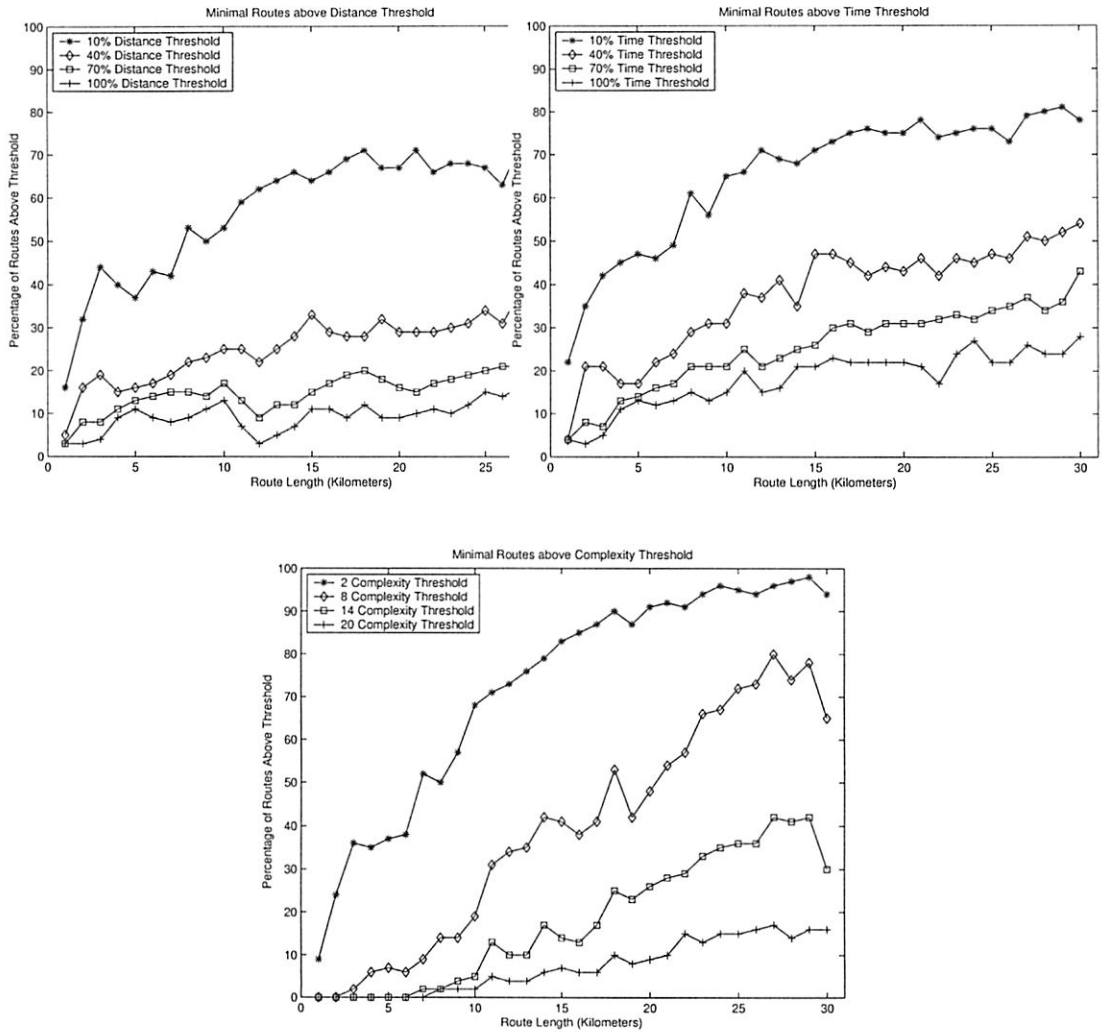


Figure 3: Percentage of tasks above one-dimensional thresholds

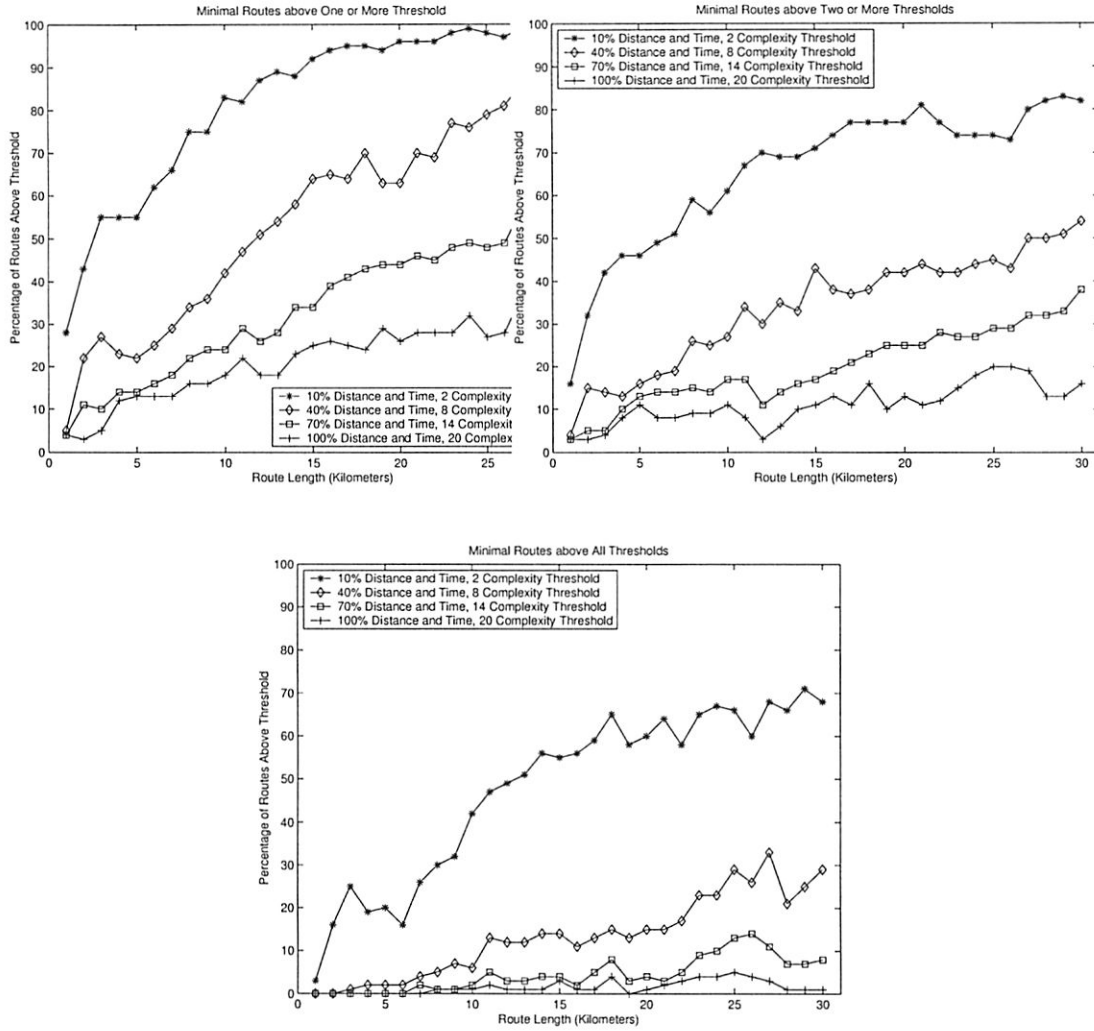


Figure 4: Percentage of tasks above multi-dimensional thresholds. Left: At least one threshold. Right: At least two thresholds. Bottom: At least three thresholds

Route	Length	Time	Complexity
Shortest	10	20	20
Fastest	20	10	20
Simplest	20	20	10
Best	10	10	10
Worst	20	20	20

Table 1: Sample descriptions of extreme routes and the bounds they form

desirable in various degrees to the separate members of our user population. In addition, we show that we can detect when a task’s solution space is so small as to limit either the size or the diversity of the task’s output set. In such cases, we can skip our usual method of exploration, and output an optimal solution route immediately.

## 4.1 The Boundaries of the Solution Space

Initially, we attempted to derive some easily obtained measure for the combined size and diversity, or “interestingness”, of a task’s solution space, and we centered our search around the properties of the task extreme solutions – the shortest, fastest, and simplest routes that solve it. As scenicness is unique in our route description as an attribute one might wish to maximize, no extreme solution exists for it, as for any task this would be a route that included all the scenic segments in our database and the segments necessary to connect them; taking the user through Yellowstone National Park, for example, on their way from New York to Boston.

These routes are useful not only because they are easily computed, well defined points of reference for any task, but also because they intuitively bound the task’s solution space. As minima they give the lower bound for length, time, and complexity for any solving route. They also offer a sort of upper bound, because we have limited our solution space to preferable routes. We can express these bounds in two route descriptions, each composed of components borrowed from the extreme solutions. The first of these captures our lower bound:

$$best = \langle shortest.length, fastest.time, simplest.complexity \rangle$$

Any route with this description dominates, and so is preferable to, all other solutions to the task, so we call it the “best” solution. This captures our lower

bound on the values of each of our length, time, and complexity attributes. We capture our upper bound in similar fashion:

$$\begin{aligned} \text{worst.length} &= \text{worst}(\text{fastest.length}, \text{simplest.length}) \\ \text{worst.time} &= \text{worst}(\text{shortest.time}, \text{simplest.time}) \\ \text{worst.complexity} &= \text{worst}(\text{shortest.complexity}, \text{fastest.complexity}) \end{aligned}$$

We call this the “worst” route, as it is by construction dominated by all of our extreme solutions, so neither it nor any route it dominates could be a member of the task’s solution space. Unlike our best route, however, this route’s description constrains rather than bounds the length, time, and complexity of potential solutions.

To illustrate how these bound can be used to characterize the routes in a task’s solution space, imagine some route  $R$  in the solution space of the task with the extreme solutions described in Table 1. We know from the task’s best route that the complexity of  $R$  cannot be less than 10. Although we can make no such claim as to the maximum  $R$ ’s complexity can be, if we find that it is greater than or equal to 20, the complexity of the task’s worst route, we can make some claims as to the values its other attributes must have by nature of it being in the task’s solution space. In this case,  $R$ ’s length and time attributes must both be at least 10 and below 20. By the task’s best route,  $R$  must have a length of at least 10, and if it has a length of 20 or more, it will be dominated by the fastest route. Similarly,  $R$  must have a time of at least 10, and if it has a time of 20 or more, it will be dominated by the shortest route. We have found these constraints are useful for understanding the limits of a task’s solution space, and note that although we formulate our bounds as route descriptions, no routes with these descriptions are guaranteed, or even likely, to exist.

Our motivation for determining a task’s “interestingness” stemmed from a desire to be able to identify the amount of effort we should exert in exploring its solution space, so we could tune our system parameters accordingly, on a per task basis. We developed the notion of the *spread* of task  $t$  defined as follows:

$$\begin{aligned} \text{spread}(t) &= ||x_1(t) - x_2(t)|| + ||x_1(t) - x_3(t)|| + ||x_2(t) - x_3(t)|| \\ x_1(t) &= \langle 1.0, \frac{\text{shortest}(t).time}{\text{fastest}(t).time}, \frac{\text{shortest}(t).complexity}{\text{simplest}(t).complexity} \rangle \end{aligned}$$

$$x_2(t) = \left\langle \frac{fastest(t).length}{shortest(t).length}, 1.0, \frac{fastest(t).complexity}{simplest(t).complexity} \right\rangle$$

$$x_3(t) = \left\langle \frac{simplest(t).length}{shortest(t).length}, \frac{simplest(t).time}{fastest(t).time}, 1.0 \right\rangle$$

This is the Euclidean distance between the three extreme solutions in a normalized attribute 3-space. We had hoped that a higher spread value, indicating increased separation between a task’s extreme solutions, would imply a solution set for the task of increased size and diversity. Unfortunately, this did not bear out as a result of the discrete nature of a task’s solution space, and led us to abandon any attempt to gauge the “interestingness” of a task’s solution space in a similar fashion. We shall argue this point in the next section.

Discarding the notion of spread, we reduced our search from the “interestingness” of a task’s entire solution space to just that of its borders. As mentioned above, these borders are defined by a task’s extreme solutions which are well defined and easily computed. Since there are always the same number of extreme solutions, three per task by our description, their level of “interestingness” is gauged solely by how much they differ. To measure this, we computed the extreme solutions for one-hundred randomly generated tasks for each of thirty one-kilometer bins, from one to two kilometers to thirty to thirty-one kilometers. Each task then had its extreme solutions compared using a number of simple thresholding similarity metrics. For example, to find the number of tasks with at least a ten percent difference in the length of their extreme solutions, we would compare 110% of the length of each task’s best route to the length of the task’s worst route. This algorithm is presented in Figure 2. We employed four threshold levels each in the length, time, and complexity dimensions. For length and time these took the form of percentage thresholds as described above. Our complexity measure differs significantly in scale from that of both our length and time dimensions, so we compared the complexities of our extreme solutions using difference thresholds. We compared the difference between the complexities of the best and worst routes with fixed threshold values. We treat the thresholding methods as effectively equivalent, and prefer them to spread as they are easier to interpret and allow consideration of differences in individual dimensions.

Results of this experiment are shown in Figure 3 for the single-dimensional thresholds. In each graph, the x-axis displays the task bins arranged by

length from shortest to longest. The y-axes display the percentage of tasks that registered above threshold, and the four lines represent the various threshold levels. We combined the one-dimensional comparison results to get the percentage of tasks above any one-, two- and three-dimensional thresholds in the left, right and bottom graphs of Figure 4. Again, the x-axes display the bins and the lines represent differing thresholds. In the left graph, the y-axis shows the percentage of tasks registering above at least one of the three single-dimensional thresholds, while in the right graph, the y-axis shows the percentage of tasks registering above two of three thresholds. The bottom graph shows the percentage of tasks above all three thresholds.

The graphs in Figure 3 show that for tasks of sufficient length, somewhere between five and ten kilometers, one can expect a majority of tasks to have extreme solutions that differ by some margin in each dimension: a ten-percent difference in length or time, or a difference of two in complexity. These are results for differences of a single dimension, but the right graph in Figure 4 shows that a majority of these tasks are also above threshold in at least two-dimensions. The bottom graph in Figure 4 shows that although fewer tasks have extreme solutions above a three dimensional threshold, many still do. While the results in Figure 3 are useful for understanding the relationship between a task's length and the diversity of its extreme solutions in a particular dimension, they and the left graph in Figure 4 are insufficient to guarantee that the tasks registering above the single-dimensional thresholds have nontrivial solution spaces. The extreme solution in these thresholds' dimensions may still  $\epsilon$  dominate all other solving routes.

Figure 4 allows us to make some initial claims as to the "interestingness" of a task's solution space. The multidimensional differences shown there, in at least two dimensions on the right and three dimensions on the bottom, guarantee that more than one preferable solution exists for those tasks above the thresholds. At the least, the differing extreme solutions themselves are members of the task's solution set, as each is preferable to the other for some set of preferences. They also allow for the existence of hybrid routes in the space between them, which we call the interior of the task's solution space. Some of these hybrid routes could also be members of the task's solution set: "best" for some set of user-preferences.

In general, for both single and multidimensional thresholds, longer tasks have a larger chance of being above threshold, and have a larger chance of being above larger thresholds. This is shown by the positive slope of all the threshold lines in each of our graphs, and means that a greater portion of



longer tasks have the *potential* to have “interesting” interiors and these interiors are *potentially* more “interesting” than those of shorter tasks. We stress the potential nature of this “interestingness” because the discrete nature of the interior of a task’s solution space makes it impossible to infer from its borders alone how many hybrid solutions may exist for that task or how diverse those hybrids may be. At the same time, a task registering above a higher threshold demonstrates more room in its solution space allowing for both a greater density and diversity of hybrid solutions.

Although we cannot infer the “interestingness” of the interior of a task’s solution space from its extreme solutions, we can at least determine from them if it is uninteresting. We can characterize tasks whose minimal solutions are not above some low multidimensional threshold as having an effectively trivial solution spaces, as in this case we know that at least one of the task’s extreme solutions is  $\epsilon$ -dominant, with an  $\epsilon$  defined by our threshold. Since our solution space is trivial, the dominant route is optimal for all users, and should be returned immediately. Because we can easily compute when a task’s extreme solutions are below some multidimensional threshold, we can optimize our system to notice task’s with trivial solution spaces, returning their optimal extreme solution to the user. Since each of the figures shows that some tasks of all lengths register below even the lowest multidimensional threshold levels, this offers a valuable optimization for our system.

## 4.2 The Interior of the Solution Space

We have shown that there can be multiple non-dominated solutions to a routing task, and that each of these could be considered the “best” route for that task by some user. We therefore should take into account what our users’ preferences are when determining which routes to supply them. While we believe that a user has a certain set of preferences during any particular interaction with our system, we think that it is unrealistic to expect that any system can faithfully model these preferences. Our chief concern is the high degree of mutability of a user’s preferences between interactions with a route-solving system due to considerations no system could, or should be expected to, model. For example, during one interaction, a user may have found out her mother-in-law’s flight got in early at the airport, and so wants the fastest possible route, whereas when driving in unfamiliar territory, she usually prefers a combination of short, and simple routes. Such circumstances prohibit attempts to learn a single set of preferences per user, even over

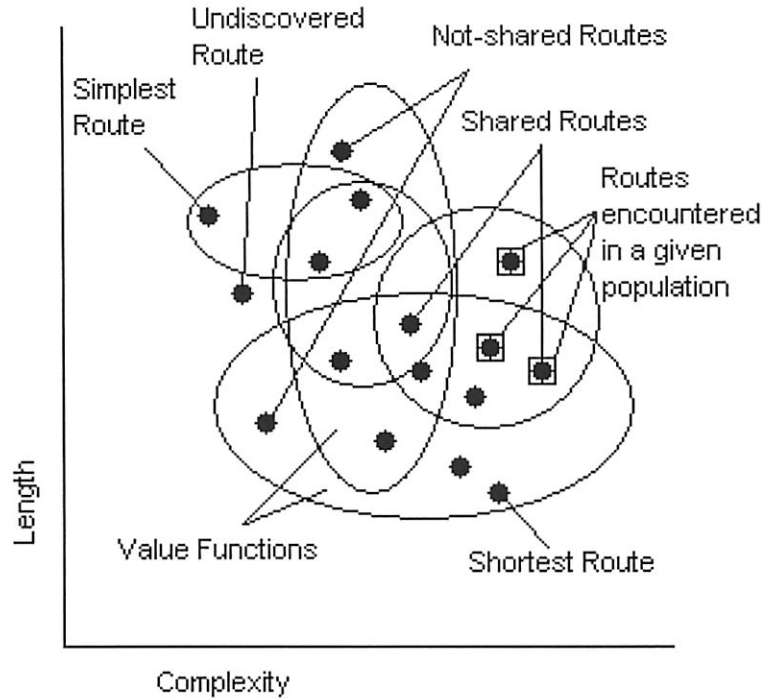


Figure 5: The solution space and utility functions

multiple interactions, and reduce a system to trying to guess what these preferences may be at any given time.

To address this shortcoming of any modeling technique, we aim to satisfy not a single set of preferences but a distribution of them. One way of interpreting this is that our system does not model individual users, but populations of them, and seeks to optimize its performance across that population. Our system views a user whose preferences change between interactions as essentially two separate members of our population. Because we are now attempting to satisfy many, rather than a single, sets of preferences, and we have seen that a given task may have many preferable solutions, our system should at times output multiple routes. An output set allows us to better satisfy a larger portion of our preference distribution, and if it is small, does not place too large a burden on the user who should be able to quickly identify

Function 1	$cw * cr + tw * tr + lw * lr$
Function 2	$cw * cr + tw * tr + lw * lr^2$
Function 3	$cw * cr + tw * tr^2 + lw * lr$
Function 4	$cw * cr^2 + tw * tr + lw * lr$
Function 5	$cw * cr + tw * tr^2 + lw * lr^2$
Function 6	$cw * cr^2 + tw * tr + lw * lr^2$
Function 7	$cw * cr^2 + tw * tr^2 + lw * lr$
Function 8	$cw * cr^2 + tw * tr^2 + lw * lr^2$
Function 9	$cw * \sqrt{cr} + tw * tr + lw * lr$
Function 10	$cw * \sqrt{cr} + tw * tr^2 + lw * lr^2$

Table 2: Utility functions

which single route is currently best for them.

We now need a formal model for users' preferences, a method for exploring the interior of a task's solution space in search of potential output routes, and a way of determining which of these routes to actually provide the user. We shall address the first two questions here, and the last in the following section. In doing so, we show that between tasks, even those characterized by similarly spaced extreme solutions, the properties of their solution spaces' interiors can vary widely as a result of their discrete nature. We show that decisions we make in how to model users' preferences can affect the routes found in our exploration of a task's solution space, and we describe the interplay between the task's and our model's influence on our final view of a task's solution space. We begin with a discussion of our exploration method.

Having formulated boundaries on which routes may comprise a task's solution space, it is possible to solve for all interior solutions to a given task then meet the additional criteria that none dominates any other. If we solved for all such routes, we would have fully explored the solution space of the given task, borders and interior, and could proceed to determine which routes to output for the user's consideration. Unfortunately, this requires considering a potentially large number of routes that are not in the task's solution space, and especially for long tasks, this can become computationally undesirable. Another concern is the nature of our upper bound which provides multidimensional constraints on our attributes rather than strict single attribute bounds (as our lower bound does). As we increase the dimensionality of our route description these constraints increase in dimensionality as well, making them inherently weaker, so we will end up considering even more

routes at an even higher computational cost. Instead, we incorporate our model of the users’ preferences directly into our task-solving algorithm, and limit ourselves to a direct search for the routes that optimally satisfy these preferences.

We model our population of preferences as a set of multiattribute utility functions, each expressing a single user’s tradeoffs between the values of the independent attributes of a potential solution. For  $n$  attributes, each of the functions can be written:

$$U(x_1, x_2, \dots, x_n) = \sum_{i=1}^n \lambda_i v_i(x_i)$$

where  $x_i$  is the  $i^{th}$  attribute value,  $v_i$  is the  $i^{th}$  value function, mapping  $x_i$  into some utility space, and  $\lambda_i$  is the weight associated with the  $i^{th}$  attribute, expressing the relative value placed on differences between attributes. A route whose description evaluates to an optimal function value, for our functions the minimum, is the “best” route for the set of preferences captured by that function. We use functions in this form as the criteria optimized by a Dijkstra’s algorithm [5] based search which we run once for each of our functions. The routes returned by this search are guaranteed to be both in the task’s solution space and optimal in terms of the preferences of some member of our modeled population.

Figure 5 demonstrates the relationship between value functions and the exploration of a task’s solution space. The dots in the picture represent hybrid solutions to a task that exist to be discovered, projected into a two-dimensional attribute space.<sup>1</sup> The circular and elliptical areas in the picture represent possible mappings between attribute and utility spaces by different value functions. Routes inside a value function’s area have attribute values that map to utilities that can be combined with weights to optimize the overall utility function. In this picture the value functions vary considerably in the portions of the attributes space they consider, and therefore the routes that utility functions using them can find optimal. The areas covered by two value functions can overlap, and there may be routes in areas covered

---

<sup>1</sup>We project the routes in Figure 5 because of the difficulty of displaying higher-dimensional points. One implication of the projection is that some of the routes in the Figure appear to be dominated by other plotted routes. For the purposes of this discussion we overlook this, and justify this by saying that in non-projected attribute space, all of these routes are non-dominated.

by no value function we can easily express. Within a fixed value function, the weights used further determine which routes are discovered. A thorough exploration of the weight space may capture all possible routes within a value function’s consideration, but it is possible to ignore certain routes by never solving with the weights necessary to find them. The choice of value functions and weights used offers a two-tiered level of control of the exploration of a task’s solution space that we shall show can have a significant effect on the type of routes returned.

Table 2 presents the ten utility functions with which we have experimented that differ in their value function mappings between attribute and utility space. Each of these functions captures a distribution of preferences that can be sampled by using it with specific set of weights. We experiment with ten functions to validate our distribution based exploration model and to show that one can control what types of routes are explored by modifying these functions. We construct our weight sets by randomly sampling a uniform distribution of all possible weights to make our search for hybrid routes within a given value function’s area as complete as possible. Ultimately, both the distribution of our weights and our choice of value functions will be informed by some external authority on the reality of people’s preferences. This may take the form of a survey for the weight distribution or a psychologist’s description for the value functions.

Each of the functions in Table 2 combines a set of attribute weights, the length weight,  $lw$ , time weight,  $tw$ , and complexity weight,  $cw$ , with a set of value functions. All of our value functions operate on some ratio between a route’s attribute and the attribute of task’s corresponding extreme solution. For example, the length ratio,  $lr$ , of a route is its length divided by the length of the same task’s shortest route. We define the time ratio,  $tr$ , and complexity ratio,  $cr$  in similar fashion. Some of our value functions raise this ratio to a power, others take its square root. Since our utility functions are minimized while solving Dijkstra’s algorithm, growing ratios, signifying routes with attributes increasingly distant from the task’s minimum value in that attribute, can be seen as incurring penalties to their overall utility. These penalties are adjusted according to the power of the current value function and the magnitude of the current attribute weight. Larger weight values and powers for an attribute indicate a desire to produce solutions with values closer to a task’s minimum value in that attribute. Reduced weights and roots show a tolerance for exploration of solutions with larger attribute values. For example, utility function 10 in Table 2 expresses a general willingness to

consider increased complexities (the complexity ratio is rooted) but quickly penalizes any increase in time or length (these attributes have their ratios squared). When paired with the appropriate weights, however, this could still express a desire for the least complex route, a weight vector with a complexity weight of 1.0 for example, some compromise between the fastest and shortest, time and distance weights of 0.5, or any number of other preferences.

Not shown in the Table 2 is that all ten utility functions discount the penalties incurred by scenic portions of a route according to the scenic weight,  $sw$ . The discounting works as follows: if a segment in a route is scenic, the contribution to the route’s utility of that segment’s local attributes is multiplied by  $(1.0 - sw)$ . This method was chosen because, as we have already pointed out, there is no extreme solution corresponding to a route’s scenicness attribute, and thus no point of reference exists for us to build a scenicness ratio as we do with our other attributes. Currently the utility from a route’s global attributes is not discounted according to scenicness as a result of the difficulty in formulating the impact that a single scenic segment has on these attributes. We plan on addressing this issue in future work.

Our second experiment exercises our exploration method while investigating the solution spaces of some of the tasks generated in our first experiment. We constructed a weight set by randomly sampling a uniform distribution of all possible weights one-hundred times, and used it to sample the ten distributions of users’ preferences captured by the utility functions in Table 2. We solved all of the five kilometer and twenty-three kilometer tasks generated in our first experiment using the sets of multiattribute utility functions produced by this sampling. We then examined the resulting routes and computed some intuitive measures describing the tasks’ solution spaces and the effectiveness of our method in exploring them.

The first of these measures, the number of unique routes produced for each task by each distribution, is presented in Table 3. Table 3 shows the average number of unique output routes for all one-hundred five kilometer and twenty-three kilometer tasks as well as for two subsets of the twenty-three kilometer tasks. The first subset is made up of all of the twenty-three kilometer tasks that registered above a small three dimensional threshold: 10% in length and time and 2 in complexity. The second subset is made up of all of the twenty-three kilometer tasks that registered above a larger three dimensional threshold: 50% in length and time and 10 in complexity.

Our second and third measures are the number of routes shared and not-shared between differing distributions for a given task. We call a route

Distribution	5 km	23 km	23 km low subset	23 km high subset
Distribution 1	3.04	9.17	11.09	13.00
Distribution 2	3.10	9.13	11.06	12.71
Distribution 3	3.10	9.50	11.66	13.76
Distribution 4	2.99	8.38	11.18	11.29
Distribution 5	3.17	9.26	11.38	13.41
Distribution 6	3.00	8.43	10.35	12.12
Distribution 7	3.01	8.10	9.89	10.82
Distribution 8	3.01	8.26	10.14	11.82
Distribution 9	3.21	10.51	12.82	15.53
Distribution 10	3.24	10.97	13.40	16.47
Aggregate	3.65	16.60	20.91	25.35

Table 3: Average number of unique routes by distribution

shared between two distributions if both consider the route as optimal for some set of preferences they capture. We express this measure in Table 4 as an average percentage across all tasks and distributions, which reflects the portion of a distribution’s output routes for a given task that it may expect to share with any one of the others. The actual routes shared differ from distribution to distribution, however, so we also note the average number of routes for each task that each distribution produces that are produced by no other distribution. We call this measure the number of routes that are not-shared.

In Table 5 we have selected two particular twenty-three kilometer tasks from our second thresholded subset, and listed our specific unique and non-shared results to illustrate a few points. First, tasks, even those with similarly spaced extreme solutions, can vary widely in the density of their solutions spaces. We call the number of routes populating a task’s solution space its *density*, and although we have not measured it exactly, we work under the assumption that the number of unique routes discovered by the aggregate of our distributions provides a good approximation. The aggregate of our ten distributions is made up of the combined optimal routes for all the samples of all our distributions, and so explores all portions of a task’s solution space covered by any of our value functions with any member of our weight set. Even if we have not found all possible solutions, the two tasks in the figure have very different densities: 11 and 54.

Table 5 also gives us anecdotal evidence as to the performance of our



Distribution	5 km S	23 km S	5 km NS	23 km NS
Distribution 1	97.83	85.29	0.01	0.07
Distribution 2	97.19	84.48	0.01	0.19
Distribution 3	97.33	83.40	0.03	0.31
Distribution 4	97.72	85.58	0.02	0.36
Distribution 5	96.18	84.18	0.00	0.12
Distribution 6	98.46	87.97	0.02	0.11
Distribution 7	97.80	87.70	0.00	0.22
Distribution 8	98.53	90.29	0.00	0.13
Distribution 9	94.87	77.78	0.04	0.72
Distribution 10	93.81	71.21	0.07	1.53
Sum			0.20	3.76

Table 4: Average percentage of shared(S) and number of nonshared(NS) routes by distribution

exploration method. If we look at the number of unique and non-shared routes produced for tasks one and two, we see how our system handles a rather sparse and a much denser task respectively. For the sparse task one, we see from the lack of non-shared routes and the nearly identical number of unique routes found by each distribution, that the choice of value functions is not terribly important when a task’s solution space is so lightly populated. Tables 3 and 4 show that this is nearly always the case for five kilometer tasks. There is no discernible difference in the performance of any of our ten distributions: each finds three preferable routes on average, and very rarely do any discover a solution not shared with the other distributions. We can say with some confidence then that it does not matter what value functions we choose for such tasks, and that a weight set of one-hundred weights clearly samples our distributions more than is necessary. By contrast, Table 5’s dense task two shows that for some tasks, searches with each distribution can produce a large number of routes, and that many of these may be limited to that distribution. The average number of unique routes produced per distribution is 22.7. Of the 54 unique routes found by the aggregate of our distributions, 35% were found by just one of our distributions. The other 65% were shared between the different distributions. These three qualities tell us that for this and similar tasks, the choice of value functions is meaningful, and allows us to express a nice measure of control over how the solution space



Distribution	Task 1 U	Task 2 U	Task 1 NS	Task 2 NS
Distribution 1	9	26	0	0
Distribution 2	9	19	0	2
Distribution 3	9	21	0	3
Distribution 4	8	20	0	0
Distribution 5	9	25	0	0
Distribution 6	9	15	0	1
Distribution 7	8	16	0	2
Distribution 8	9	24	0	1
Distribution 9	10	31	0	3
Distribution 10	10	30	0	7
Aggregate/Sum	11	54	0	19

Table 5: Unique(U) and non-shared(NS) routes by distribution for three highly thresholded 23km tasks

is explored. We can choose our value functions based on the characteristics of the non-shared routes they lead to without sacrificing a large base of common routes that are valued by users in each of our modeled populations.

We shall close this section with a discussion of our reasons for abandoning our attempts to measure the “interestingness” of a task from the characteristics of its extreme solutions. Primarily, this was because many tasks whose extreme solutions differed greatly, as measured by registering above some high multidimensional threshold or having a large spread, were more similar to the first task in Table 5 than the second. Tasks with high spread values were too often less “interesting” than those with lower spreads because of the discrete nature of a task’s solution space. That is, the increased “room” for hybrid routes between a task’s extreme solutions is not necessarily populated because of their dependence on the real-world existence of streets and roads. This meant that while a task’s spread in general correlated with the density of a task’s solutions, the correlation was not strong enough to make spread a good predictor of density. Therefore, as described above, we limit our use of the separation of a task’s extreme solutions to noticing when a task’s solution space is trivial, and if so returning an optimal solution immediately.

$U$	$S_1$	$S_2$	$S_3$
$D_1$	5	100	35
$D_2$	19	5	20
$D_3$	6	15	5

Table 6: Sample utilities of three routes by three functions

## 5 Determining Interesting Output Routes

We have shown that for many routing tasks we can produce a set of routes whose members are each optimal for some set of user preferences. If this set of routes is small, say three or four routes, it is reasonable to supply it to the user as is and let her determine which route they prefer to take. If the set is not small, however, and the user has to take the time to consider a large number of output routes, our system has placed an undesirable burden on the user (and one that no other current routing system places on them). We therefore are faced, for some tasks, with the additional problem of determining a good subset of routes, within some size constraint, to return from the task’s solution space. We present two algorithms, which share a common preprocessing step, that solve this problem and differ in runtimes and their notion of what constitutes a “good” subset.

Our first attempt to describe what makes a “good” subset of routes to output centered on the notion of ensuring the diversity of the routes in this subset. We postulated that having the routes in our subset be as different from each other as possible increased our chances of having some “good” route for each member of our population. For example, if we have three very different routes in our subset and some user does not like two of them, she may like the third by virtue of its difference from the first two. We note that diversity in our output set is a desirable quality even if a user does not know her preferences, as it simplifies the decision process of the user when selecting a final route from our set. Because the routes are diverse the user never has to make a difficult comparison between routes that are too similar: the difference between the routes can be thought highlight the user’s preferences.

Another concern is that we never want to return to the user two routes that are very similar to each other. If the differences between two routes’ descriptions are slight enough, we recall our notion of  $\epsilon$ -dominance, there is no reason to return them both to the user. If we do, we are either wasting

valuable space in our return subset on practically duplicate routes, or should be returning a single route in the first place. We therefore propose a preprocessing step to be used before any subset selection algorithm to remove all  $\epsilon$ -dominated routes from consideration. This works as follows: each route in our solution space is considered in turn, and compared to each other route in the solution space for  $\epsilon$ -dominance. If one route  $\epsilon$ -dominates the other, the dominated route is removed from further consideration during the determination of our output. If two routes are found to  $\epsilon$ -dominate each other, we remove the route with the lower average utility with respect to all members of our population. The choice of  $\epsilon$  depends on how strict a notion of similarity one wishes to enforce, and we leave the problem of picking good values to future work.

After eliminating all  $\epsilon$ -dominated routes from our solution set, we are guaranteed that any routes selected to be returned to the user have descriptions different by our factor  $\epsilon$ . This ensures a certain level of diversity in any output set formed of the remaining routes, and has the secondary effect of reducing the number of routes we need to consider in our subset selection algorithms. We abandon any further direct attempt to increase the diversity of the routes in our output set, instead focusing on optimizing two different definitions of the utility of a set of our routes with respect to our distribution of preferences. These two definitions form the basis of our two set-selection algorithms. We shift our focus, from diversity to utility, because the utility of a set is the direct measure of its usefulness to our population of user, and we have already guaranteed the set's diversity.

Our first algorithm begins with a simple notion of what make a "good" subset. For a subset of size  $n$ , we will simply take the  $n$  routes whose summed utilities with respect to all the members of our population are optimal. For us, lower utilities are better, so we are try to compute the following:

$$\min_{S \subseteq R} \text{AverageUtility}(D, S) = \sum_{i=1}^{|D|} \sum_{j=1}^{|S|} U(D_i, S_j)$$

where  $D$  is the set of samples from our distribution of utility functions, and  $D_i$  is the  $i^{\text{th}}$  sample. The set  $S$  is our subset of solution routes selected from the routes in our solution space  $R$ ,  $S_j$  is the  $j^{\text{th}}$  route from this subset, and  $U(D_i, S_j)$  is the utility of route  $S_j$  according to function  $D_i$ .

A subset  $S$  of size one that minimizes this quantity will be a route with minimal utility, on average, with respect to all samples in our population.

Each additional route added to the subset while minimizing this quantity will have property that, of all routes not currently in our subset, it has the lowest average utility with respect to our population. This means that we are effectively finding a single cluster of solution routes that in utility space is centered around the route with the lowest average utility. These routes are easy to find through an exhaustive search of the potential solution routes, which takes  $O(|D||S|)$  time. As we have already stated that there is some small upper bound on the size of our subsets, this is effectively linear in the number of samples we take from our distribution of functions.

It is important to note that routes with similar average utilities do not necessarily have similar descriptions. Of course they may, but observe the following example. If we have two utility functions sampled from our distribution  $D_1$  and  $D_2$ , and two routes  $S_1$  and  $S_2$ , if  $D_1(S_1) = D_2(S_2) = 10$  and  $D_1(S_2) = D_2(S_1) = 20$ ,  $S_1$  and  $S_2$  have the same average utility but must have very different route description to vary in utility so much between  $D_1$  and  $D_2$ . Optimizing for average utility then allows us to cheaply compute a subset whose members each have the best chance possible to satisfy the samples of our population, and no limit is imposed on the diversity of this subset.

Our second algorithm is inspired by the observation that although we return a subset of routes, a user selects only one of them as the final solution to their query. If we assume the user will pick the route from our subset with the minimum utility with respect to her exact preferences at the time, another measure of how “good” our subset is would be the following sum:

$$MinimumUtility(D, S) = \sum_{i=1}^{|D|} \min\{U(D_i, S_j) \mid j = 1, \dots, |S|\}$$

If we minimize this quantity we optimize the average utility, with respect to all members of our population, of only the best route in our subset for each sample in our population.

For a subset of size one this will select the same route as our AverageUtility algorithm, but when we solve for subsets of size two or more, the algorithms may diverge. Consider the example in Table 6. As before  $D_1$ ,  $D_2$ , and  $D_3$  are utility functions sampled from our distribution, whose utilities for routes  $S_1$ ,  $S_2$  and  $S_3$  from our solution space are given in the Table 6. If we select a subset of size two according to our AverageUtility algorithm, we will choose routes  $S_1$  and  $S_3$  by virtue of their average utilities of 10 and

20 being lower than the 40 of  $S_2$ . Our MinimumUtility algorithm, however, will select the subset of  $S_1$  and  $S_2$ , 19, over  $S_1$  and  $S_3$ , 29, and  $S_2$  and  $S_3$ , 45. This is because  $S_1$  has an optimal utility for  $D_1$  and a near optimal for  $D_3$ , and  $S_2$  is optimal for  $D_2$ . It is interesting to note that in this example, our MinimumUtility algorithm is free to work around the very poor utility of  $S_2$  to  $D_1$  and still include  $S_2$  in an output subset, while this essentially eliminates it from consideration by our AverageUtility algorithm.

An algorithm that exhaustively searches for the optimal subset with respect to our MinimumUtility measure is much slower than the one we describe to optimize for AverageUtility. This is because unlike for AverageUtility, a size  $n$  subset does not necessarily contain the size  $n - 1$  subset and an additional route we can find directly. An exhaustive search would therefore require enumerating all subsets of which there are at worst  $O(\binom{|D|}{|S|})$ . Since for distributions that result in large solution sets this quickly becomes infeasible, even for small subsets, we use a simple hill-climbing algorithm instead of an exhaustive search. This means our return subset is not guaranteed to be globally optimal in our MinimumUtility measure, but repeated random restarts run for a moderate number of iterations have been observed to have a stable output which we take to be optimal. The running time of our hill-climbing algorithm is  $O(\text{iterations} * \text{restarts})$ .

## 6 Related Work

Current internet mapping sites such as MapQuest (<http://www.mapquest.com>) and MapPoint (<http://mappoint.msn.com>) offer route solving systems that optimize for either distance driven or driving time. MapQuest can also be instructed to, if possible, avoid toll roads and ferries. The focus of recent improvements to these sites seems to be on developing more intuitive and useful methods of route presentation: maps that can be zoomed or rotated and feature business or geographic landmarks.

Other route solving systems have been developed to return route solutions other than the fastest or shortest. Rogers *et. al* [2] do so by modeling the preferences of individual users using a single multiattribute utility function per user. This function takes the form of a weighted linear combination of four route attributes: length, traversal time, number of intersections, and number of turns. The vector of weights associated with a user encapsulates their tradeoffs between these attributes' values, and is said to express their

preferences. Dijkstra’s algorithm is then used to solve for the optimal route in terms of this utility function. This is similar to what our system does for each sample of our distribution of utility functions, and does not require a final set selection algorithm as we do.

They use simple machine learning techniques to improve their view of a user’s preferences, encapsulated in a weight vector, across multiple system interactions. For every task provided by the user, the system provided two routes in response: the optimal and one produced by using a slight variation of the user’s weights. If the user select the first route, the weight vector stays the same. If they choose the second, their weights are adjusted according to the perceptron rule to reflect the system’s new view of the user’s preferences. Further work focused on improving the interface for interaction between the system and a user [3], to make it usable while driving, and the application of more powerful learning algorithms such a support vector machines. [1]

Haigh *et. al* implemented a system that identifies good routes without modeling users. [6] They use case-based reasoning to find routes that take advantage knowledge gained in solving previous tasks. This knowledge takes the form of a database of cases, previous route solutions, or pieces of them, deemed appropriate for reuse, that is created and maintained over a period of many system interactions. A detailed, domain-specific geometric similarity metric is used to piece together previous cases to form a plan for a more detailed solution to the current task. This plan is used to direct the search for an actual route using a modified version of Dijkstra’s shortest path algorithm. Since plan generation occurs in the space of cases, which is much smaller than the segment space, and it allows for a severely limited search of the segment space, their method provides a significant computational benefit. Also, the cases can be assigned a “goodness” which effects their selection for reuse, and this can even be parameterized by some factor such a time. This allows the system to represent things such as traffic, where portions of a road are less desirable at certain times of the day, and other concerns of a dynamic nature.

## 7 Conclusions and Future Work

In this work, we have presented our method for finding a good *set* of routes to return to a user for a given routing task. We have shown by comparing the properties of the extreme solutions of routing tasks that multiple routes



exist which different users, or the same user at different times, may consider to be the best solution to a given task based on their current preferences. We have presented a method for exploring the space in between these extremes based on building a distribution of utility functions and solving for the set of optimal routes over that distribution. We have characterized this distribution as simultaneously representing either a population of users with different preferences, or the different sets of preferences a single user may have at different times. We have demonstrated that the choices we make in building this distribution, the value functions we select and weights which act as our sampling points, allow us to control the type of solution routes that we discover. Using this method, we can create a set of solution routes for a task whose members are each optimal for some members of our population. Finally, we have given two algorithms for reducing the size of this set to a level appropriate for output to the user, and provided guarantees that this subset is diverse. Taken as a whole, we have provided the justification for and framework of a route solving system that maximizes the utility of the routes it returns across a population of users.

Our future work includes improvements to the system model described here, as well as additions to our problem domain that require new methods of exploration. Incremental improvements include adding attributes to our route and segment descriptions and employing a more detailed notion of route complexity and scenicness. As a start we would like to handle one-way streets, speed-limits, tolls, and stop signs and stoplights. Larger goals include incorporating dynamic and temporal information, such as traffic conditions, into our knowledge base. Also in this category, we would like to include alternative methods of transportation that may operate under some uncertainty, particularly trains and buses, and accurately reflect their operational stochasticity.

For each of these changes, we will need to adjust our template for the utility functions that make up our distribution. We not only want our new utility functions to take into account any new route attributes we may introduce, we would also like them to accurately capture the real preferences of our users. While in this work we have experimented with a number of value mappings and a large evenly-distributed weight set to explore the effects of using different distributions, we would like the utility functions we use in future work to be informed by some authority on the actual preferences of our user population. We may obtain this knowledge through psychological experiments, surveys, or a search of the related literature. This includes not

only value function mappings for our route attributes but the weights we employ to sample our distribution.

Finally, we see from our own and the related work that there are ample opportunities to apply machine learning techniques to improve a system's output quality, execution time, or both. This could take the form of adjusting the number or placement of our distribution samples, encoded as our weight set, based on observed user interaction and feedback. Also if adding temporal concerns such as train and bus schedule significantly increases the complexity of our problem space, we might also apply some hybrid of our current system with the case-based techniques employed by Haigh *et. al* to offset this.

The routing task domain is an interesting problem space in which we have applied the concept of modeling users' preferences using a distribution of multiattribute utility functions. This concept could be applied in similar domains where many preferable solutions exist for a user's problem, and some system must decide which solutions to provide. Planning and recommender systems such as automated travel agents or activity planners fit this description, and we can imagine many of an emerging class of autonomous web-agents that do as well.

## References

- [1] Claude-Nicolas Flechter and Seth Rogers. Learning subjective functions with wide margins. *AAAI Spring Symposium on Adaptive User Interfaces*, pages 106–113, June 2000.
- [2] Seth Rogers Claude-Nicolas Flechter and Pat Langley. An adaptive interactive agent for route advice. In Oren Etzioni, Jörg P. Müller, and Jeffrey M. Bradshaw, editors, *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 198–205, Seattle, WA, USA, 1999. ACM Press.
- [3] Seth Rogers Claude-Nicolas Flechter and Cynthia Thompson. Adaptive user interfaces for automotive environments. *IEEE Intelligent Vehicle Symposium*, October 2000.
- [4] Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives*. John Wiley and Sons, New York, 1976.



- [5] Thomas H. Corman Charles E. Leiserson and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.
- [6] Karen Zita Haigh Jonathan Richard Shewchuk and Manuela Veloso. Exploiting domain geometry in analogical route planning. *Journal of Experimental and Theoretical AI* 9, pages 509–541, 1997.