Giving Advice to People in Path Selection Problems

Amos Azaria, Zinovi Rabinovich, Sarit Kraus, Claudia V. Goldman, Omer Tsimhoni
Computer Science, Bar-Ilan University
azariaa1@mail.biu.ac.il, {zinovi,sarit}@cs.biu.ac.il
General Motors Advanced Technical Center - Israel
{claudia.goldman, omer.tsimhoni}@gm.com

Abstract

We present a novel computational method for advice-generation in path selection problems which are difficult for people to solve. The advisor agent's interests may conflict with the interests of the people who receive the advice. Such optimization settings arise in many human-computer applications in which agents and people are self-interested but also share certain goals, such as automatic route-selection systems that also reason about environmental costs. This paper presents an agent that clusters people into one of several types, based on how their path selection behavior adheres to the paths preferred by the agent and are not necessarily preferred by the people. It predicts the likelihood that people deviate from these suggested paths and uses a decision theoretic approach to suggest modified paths to people that will maximize the agent's expected benefit. This technique was evaluated empirically in an extensive study involving hundreds of human subjects solving the path selection problem in mazes. Results showed that the agent was able to outperform alternative methods that solely considered the benefit to the agent or the person, or did not provide any advice.

Introduction

This paper considers the problem of path selection in human-computer applications in which people and computers are self-interested, but also have shared goals. For example, consider an automatic system for suggesting commuting routes to a human driver. Both participants in this setting share the goal of getting the driver from home to work and back. However, each participant also has its own incentives. The driver wishes to choose the route that minimizes the commuting time, while the computer may prefer taking a longer route that emits fewer pollutants, or does not pass near schools and playgrounds.

The paper proposes a novel model, User Modeling for Path Advice (UMPA), for computer-generated advice for such route-selection problems which explicitly reasons about the extent to which people deviate from a given path. This information is then used to suggest modified paths which are beneficial to the computer agent while still likely to be considered by people. It focuses on route-selection problems that are too large for people to solve optimally, and in which the agents' utilities may significantly differ from those of people. Our study includes a 2-participant task setting for choosing a path on a large colored grid that is analogous to the route-selection problem. The person's sole incentive is to choose the shortest path, while the agent's incentives also include the number of color changes in the path. Choosing a path on the grid corresponds, for example, to selecting a commuting route between home and work. The colors on the grid represent constraints, such as environmental and social considerations. Switching between colors on the path represent the violation of one of these constraints. The person's preferences solely consider the length of the route, while the agent's preferences take into account both the length of the route as well as the number of constraint violations.

Our approach defined a deviation (or "cut") from a suggested path as an alternative segment for connecting two local points in the original path. A cut may improve the path from the user point of view by shortening it, but may decrease the benefit of the agent. For any such suggested path, UMPA generates the set of all cuts, corresponding to modified, shorter paths that are more likely to be accepted by people. We used a standard feature-based supervised learning approach to estimate the likelihood that people follow these modified paths when suggested by computer agents. We present an agent-design that clusters people's route selection behavior into several types: those that are likely to follow suggested paths that are more beneficial for the agent, those that are likely to deviate and follow modified paths, and those that are likely to ignore suggested paths. It uses a decision theoretic approach to suggest modified paths that maximize the agent's utility given its model of people's path selection behavior.

We evaluated this agent in an extensive empirical study of more than 700 human subjects solving the path selection problem in 4 different mazes. The results showed that an agent using our approach was able to outperform agents using alternative approaches for suggesting paths that solely consider the agent's utility, the person's utility, or do not provide advice. In addition, people were satisfied by UMPA advice.

Related Work

Route selection (or path selection) has become one of the most prominent applications of computer assisted guidance (see a survey at (Hipp *et al.* 2010)). In fact, route guidance systems using GPS have become pervasive over the years, thanks to the significant research effort in addressing both the cognitive limitations and the range of individual preferences of human users (e.g. (Duckham and Kulik 2003; Park *et al.* 2007)).

Many of the challenges in the development of route guidance systems stem from the high variance across individuals of the evaluation and the acceptance of route advice. This variance makes it important to tailor route advice and guidance to a specific user. To this end, a wide range of machine learning techniques are used to capture and utilize user routing preferences (e.g. (Park *et al.* 2007)). We, on the other hand, use a machine learning approach to model user's attitudes towards advice which will increase the agent's benefit. There was some work on driver acceptance of unreliable route guidance information (Hanowski *et al.* 1994). Antos and Pfeffer (Antos and Pfeffer 2009) designed a cooperative agent that uses graphical models to generate arguments between human decision makers and computer agents in incomplete information settings. They use a qualitative approach that does not model the extent to which people deviate from computer generated advice. Other works have demonstrated a human tendency to accept advice given by an adversary in games (Kuang *et al.* 2007) and some theoretical analysis suggests this behavior to be rational (Rayo and Segal 2010). To some extent, these results were used in the framework of large population traffic manipulation (either by explicitly changing the network topology or by providing traffic information, e.g. (Hui *et al.* 2009; Chorus *et al.* 2006)). However, to the best of our knowledge, we are the first to study the combination of human choice manipulation and the personal route selection problem in a given network.

Modeling The Path Selection Problem

To allow a formal discussion of the path selection problem, we employ a maze model. We assume that a user has to solve the shortest path problem within a rectangular maze either by constructing a path or by augmenting a path suggestion. More formally, we define a *maze* M as a grid of size $n \times m$ with one vertex marked as the source S, and another vertex as the target T. Each vertex v is associated with a label $c(v)$, that we will refer to as the *color* of v. We will denote the white color or label number 0 as an *obstacle*. $x(v)$ and $y(v)$ denote the horizontal and the vertical grid coordinates of the vertex v, respectively. We assume that the user can move along the grid edges in the four standard directions: up, down, left or right. A sequence of vertexes that does not include an obstacle and can be traversed moving the four standard directions is a *valid path*. Formally, a *valid path* π of length $l(\pi)$ (or simply l), is defined by a set of vertexes: $v^1, v^2 ... v^l$ so that $\forall 1 \leq i \leq l, c(v^i) \neq 0$ and $\forall 1 \leq i < l, (|x(v^i) - x(v^{i+1})| = 1 \wedge y(v^i) - y(v^{i+1}) = 0)$ or $(|y(v^i) - y(v^{i+1})| = 1 \wedge x(v^i) - x(v^{i+1}) = 0)$. In the remainder of the paper, to distinguish between vertexes of different paths, we will denote them by the path's name with a superscript: e.g. vertexes of a path π will be denoted by $\pi^1, ..., \pi^l$. A valid path will be called a *full path* if $\pi^1 = S$ and $\pi^l = T$, i.e. it begins at the start and ends at the target node, thus solving the maze.

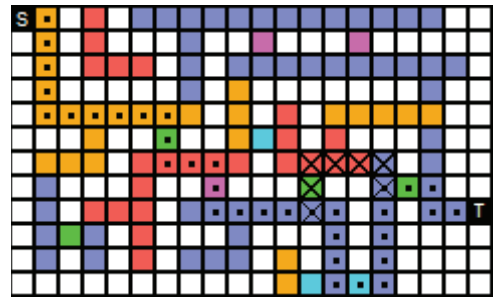

Figure 1: Path selection problem visualized in a small maze

We model the *path selection problem* as a user's task to find the shortest full path through the maze. Formally, we assume that user's cost of a path is equal to its length, i.e. π is $u_u(\pi) = l(\pi)$. In contrast, the advisor agent's cost depends on both the length of the path and the number of color switching along the path. Formally, given a color switching cost W, the agent cost u_a of a full path π is given by: $u_a(\pi, W) = l(\pi) + W \cdot \sum_{1 \leq i < l} \mathbf{1}\{c(\pi^i) \neq c(\pi^{i+1})\}$. We term a full path which minimizes u_a the *agent path*, and a full path which minimizes u_u is termed the *shortest path*. Notice, that there are multiple valid paths through a maze, and it is possible that there is more than one full path as well. We therefore define the average cost of the agent with respect to a multi-set $\Omega = \{\pi_1, \pi_2, ..., \pi_k\}$ of valid paths as $u_a(\Omega) = \frac{1}{k} \sum_{0 \leq i \leq k} u_a(\pi_i)$. Such a multi-set may be formed, for instance, by an experimental data set, where different users have selected different paths π_i. Furthermore, we do not assume that a user is capable of finding the shortest path, and allow the set Ω to include any full paths.

Now, in addition to the maze grid, its color labeling, and the start and finish nodes, we also allow for a secondary labeling of a particular full path through the maze. This labeling represents the advised path of the agent to the user. We assume that the user is aware of this labeling prior to solving the path selection problem. In fact, the advised path is part of the input to the path selection problem. When a user is given a maze (with or without an advised path) his/her goal is to solve the maze by finding the shortest full path from S to T. However, due to the complexity of the maze, finding said shortest path may not be trivial or clear from looking at the maze during the limited amount of time given to the user. Therefore, the user may find it beneficial to take some advice provided to him in order to choose his route. In turn, the *best advised path problem* is: given a maze M, find a full path to be given as advice so that the set of likely user solutions, Ω, will minimize the average agent cost, $u_a(\Omega)$.

Figure 1 visualizes the formal setting in a small maze. In the figure, obstacles are represented by the white color, while the start and the end nodes are black. In turn, the

dotted nodes represent the advised path, while the crossed nodes are a valid (partial) path selected by the user.

UMPA approach

Given a maze M, we employ a three-stage process to solve the best advised path problem: (i) clustering users into one of three types depending on the extent to which their path selection behavior adheres to suggested paths that may be more beneficial to the agent than to themselves, then predicting the likelihood that a user will belong to one of these three clusters;(ii) predicting the likelihood that people deviate from a suggested path; (iii) generating the advised path using a decision theoretic approach which utilizes the prediction from the first two stages in order to compute the expected benefit of the agent from a given path.

We assume the availability of training data for the prediction stages. Consider a set $\mathcal{M}$ of mazes where $M \notin \mathcal{M}$. UMPA is given a training set, Ψ, of tuples (M', π, τ, α) and (M', τ), where: $M' \in \mathcal{M}$ is a maze; π is an advised path through the maze; α is a binary variable indicating whether the user considers π as a good solution or not (α equal 1 or 0 respectively); and τ is the solution selected by a human user, who was presented with M' and π.

Modeling diversity in people's reactions

UMPA clusters the users into three types: *Advice followers*, *Advice ignorers* and *Baseline users*. It labels the user of each example of Ψ with one of the types and denotes the labeled set of examples Ψ^l.

Advice followers are users who follow the advised path without any modification regardless of the quality of the advice and even when they think that the advice is not a good solution. We can only be sure a subject is an advice follower if he claims the advice wasn't good but followed it exactly; therefore, if he was given any other advice (regardless of how good it was from his point of view) he is anticipated to follow it exactly as well. However, if he claimed the advice was good, he might have just taken the advice because that it seemed good to him. The user of an example $(M', \pi, \tau, \alpha) \in \Psi$ is labeled as *Advice followers* if $\tau = \pi$ and ,$\alpha = 0$.

However, most users would at least attempt to improve upon the advised path, or simply ignore it entirely. In order to characterize these users we will introduce the concept of a *cut* and a *baseline solution*.

Given two vertices ,π^i and $\pi^{i'}$, of an advised path π, any path τ between these two vertices (that otherwise does not intersect with π) is termed a *cut*. Although there may be an exponential number of cuts, certain human cognitive tendencies (see e.g. (Duckham and Kulik 2003; Richter and Duckham 2008)) allow us to bound the maximal cut length. All users who deviated from the advised path only by taking cuts were considered as users who used the advice as a *baseline solution*, and therefore termed *Baseline users*.

More formally, given a valid path π we define a cut τ of length l to be a valid path such that $\exists i, \tau^1 = \pi^i$ and $\exists i' > i, \tau^l = \pi^{i'}$ and $\forall 1 < i'' < l, \nexists j, \pi^{i''} = \pi^j$. The sequence of $\pi^i, ..., \pi^l$ will be called the original segment of cut τ and will be denoted by $o(\tau)$. Figure 1 shows an example for a cut marked by crossed nodes. In turn, using the definition of a cut we define a *baseline (solution)* as follows. Let $L_1 \in \mathbb{N}$ and $L_2 \in \mathbb{R}^+$. Let M be a maze with a marked advised path π. A solution path η uses the advice π as a baseline subject to parameters L_1 and L_2 if there are cuts of π, $\tau_1, ..., \tau_k$, with their original segments $o(\tau_1), ..., o(\tau_k)$ respectively, such that $\forall i, l(\tau_i) < L_1$ and $l(\tau_i) < L_2 \cdot l(o(\tau_i))$ and there are sequences (possibly empty) $\sigma_1, ..., \sigma_{k+1}$, such that $\pi = \sigma_1, o(\tau_1), ...\sigma_k, o(\tau_k), \sigma_{k+1}$ and $\eta = \sigma_1, \tau_1, \sigma_2, \tau_2 ...\sigma_k, \tau_k, \sigma_{k+1}$. The user of an example $(M', \pi, \tau, \alpha) \in \Psi$ is labeled a *Baseline user* if τ is a baseline solution of M' with respect to π.

Finally, we define the *Advice ignorers* as all users who are neither *Baseline users* nor *Advice followers* and the relevant examples of Ψ were labeled accordingly. It is important to understand that being an advice follower does not depend on the specific maze and advice. However, deciding whether to ignore an advice or use it as a baseline depends on the specific maze and advice.

Estimating advice cost

Given a maze M and a possible advice π, UMPA tries to estimate using Ψ^l the expected utility for the agent by presenting users with π.

The first step is to identify the multi-set $\Omega(\pi)$ – the multi-set of solutions produced by users in response to the advised path π - and estimate $u_a(\Omega(\pi))$.

Notice that it is relatively easy to calculate the contribution of *Advice followers*.These are users that, independent of the maze or the particulates of the advised path π, always fully comply with π. Therefore, their contribution to $u_a(\Omega)$ will always be $l(\pi)$ multiplied by the ratio of *Advice followers*.

However, the contribution of the *Advice ignorers* and the *Baseline users*) is more complex. To estimate it, we need to understand how frequently the available cuts at a node π^i of π are used.

Estimating the Probability of a User Taking a Cut Given a possible advice π, UMPA estimates the probability of a user taking a specific cut τ at a given vertex π^i. We denote this probability as $p(M, \pi, \pi^i, \tau)$ and use $p(\tau)$ when the other parameters are clear from the context. UMPA assumes that the function p is a linear combination of three cut features: *cut benefit*, *cut orientation*, and *cut greediness* (see e.g. (Hochmair and Karlsson 2005)).

Cut Benefit measures the relative reduction in steps between the cut and the original path segment. Formally, $\frac{l(\tau)-l(o(\tau))}{l(\tau)}$.

Cut Orientation captures the tendency of human users to continue with a straight line motion, and its value depends on whether the cut or the original segment conformed to this tendency. The reference motion is the edge between the cut divergence node π^i and its predecessor in the advised path π^{i-1}. If the cut deviates from the advice by remaining in the same direction as the edge (π^{i-1}, π^i) we say that the cut has positive $+1$ orientation. Otherwise, if the original path segment (π^i, π^{i+1}) is similarly directed to (π^{i-1}, π^i), we

say that the cut has negative -1 orientation. Otherwise, the cut's orientation is 0 (neutral). For example in Figure 1 the value of the orientation of the cut marked by x's is -1 since the advised path continues straight and the cut turns left.

Cut Greediness measures how greedily the path attempts to move towards the target node's horizontal and vertical coordinates. We express this by assigning a positive unit value to a move that closes the gap from the target node, and a negative (-2) value to a move that increases the gap.[1] The total path value is calculated as a discounted sum of movements along the path. We omit rigorous the formal definition of greediness due to space limitations.

Given that there is a very large number of cuts, it is almost impossible to collect enough examples in Ψ to learn the weights of p's features directly. Therefore, this estimation process was divided into two steps. First, UMPA tried to estimate the probability, $r(M, \pi, \pi^i, \tau)$, that a cut τ will be taken by a user at vertex π^i, assuming that τ is the only possible cut at π^i. It was assumed that r is a linear combination of the three cut features described above, similar to p. To compute the weight of r's features, UMPA created a training set of the form $(M', \pi, \pi^i, \tau, prop(\pi^i)$ where τ is a cut of π that starts at π^i and is the cut that was taken at π^i by the highest number of users according to Ψ. $prop(\pi^i)$ is the proportion of users that visited π^i and deviated there taking any cut. Using these examples, the weights were estimated using linear regression.

Next, r after normalization is used to compute p. For any π^i, it was assumed (based on the way r was learned) that the probability to deviate in π^i across all cuts is equal to the highest r value of a cut starting at π^i. This probability is distributed across all possible cuts, starting at π^i, proportional to their r value. We omit the algorithm due to space restrictions.

Estimating the Utility Contribution of Baseline Users Having the estimated probability for each cut, an estimation for the agent utility of the *baseline users*, denoted as $b(\pi)$, can be calculated using the following algorithm:

Input: A maze, with an advised path π.
Output: $EstCost$ – estimated contribution to agent cost

$EstCost \leftarrow u_a(\pi)$.
$vec \in \mathbf{R}^{l(\pi)} \leftarrow \vec{0}$. $vec(0) = 1$.
for each $i < l(\pi)$ **do**
  for each cut τ s.t. $\tau^1 = \pi^i$ **do**
    {**Predict the fraction of users who take the cut**}
    $a(\tau) \leftarrow (1 + \sum_{j<i} vec[j]) \cdot p(\tau)$
    $EstCost \leftarrow EstCost + (u_a(\tau) - u_a(o(\tau))) \cdot a(\tau)$.
    {**Update mass at cut entry point.**}
    $vec[i] \leftarrow vec[i] - a(\tau)$
    {**Update the cut exit point**}
    $vec[j|\pi^j = \tau^{l(\tau)}] \leftarrow vec[j] + a(\tau)$
return $EstCost$.

Intuitively, the algorithm's basic assumption is that the set of users forms a continuous unit mass. The algorithm then traces the flow of this unit of mass along different cuts that diverge (or converge) at vertices along the advised path.

[1]This assignment's rational follows Tversky and Kahneman's results that showed losses to be weighed twice as strongly as gains.

In more detail, the algorithm begins by stating that, even if all users are *Advice followers*, their contribution will be at least $u_a(\pi)$ and initializes the utility estimate by this value. Then the vector vec, which lists for each vertex along the path π the proportion of people who have reached it, is initialized by placing proportion 1 (all people) at the start node and zero (no people) at all other nodes along the path. The algorithm then systematically propagates the mass of people along the path and the path's cuts. At every node along the path, the mass of people split – some continue along the path to the next node along π, while others take one of the available cuts. Specifically, they split proportionately to the probability of users to adopt a particular path segment. Those who choose a cut τ are advanced and added to the mass of people who reach the end point of that cut.

A straightforward implementation of the algorithm will yield complexity of $O(\#cuts \cdot l(\pi))$. However, by saving $\sum_{j<i} vec[j]$ in each iteration the algorithm complexity can be reduced to $O(\#cuts + l(\pi))$.

Assessing Proportions of User Types Based on the literature on the subject (see e.g. (Hochmair and Karlsson 2005)), we presume that the proportion of *Baseline users* for a given advice π is strongly characterized by the overall greediness value of π, denoted $g(\pi)$. Given Ψ^l UMPA generates a set of tuples $\pi', g(\pi'), prop(\pi')$ where $prop(\pi')$ are the proportion of users in Ψ^l that received the advice π' and are labeled as *Baseline users*. Denote by $AvgGV$ ($StdGV$) the average (standard deviation) of the $g(\pi')$s and by $AvgBU$ ($StdBU$) the average (standard deviation) of $prop(\pi')$s. Finally, we estimate the proportion of *Baseline users* to be: $p_b(\pi) = \frac{g(\pi) - AvgGV}{StdGV} \cdot StdBU + AvgBU$,

As we have noted before, the *Advice follower* users have followed the advised path even if they did not evaluate it as a good path, which allowed us to assume that the proportion of *Advice follower* users is constant across all advised paths. We have extracted this proportion from Ψ^l, and in the following we denote it by E. The remaining proportion of users $1 - E - p_b(\pi)$ are assumed to be the *Advice ignorer* users. This latter set of users deviates from the advised path so much that it is possible to assume that they would have selected the same path with or without any advice given. Let $\Omega_\emptyset = \{\tau | (M, \tau) \in \Psi\}$, i.e the set of paths in Ψ selected by users who did not receive any advice. We assume that the contribution of *Advice ignorer* users to the estimated expected agent cost is $N = u_a(\Omega_\emptyset)$.

Given the above proportion and utility contribution estimates, we can compose the final *heuristic* estimate of the advised path cost $e(\pi)$, that is the expected agent cost across all human generated path solutions in response to π:

$$e(\pi) = p_b(\pi) \cdot b(\pi) + (1 - E - p_b(\pi))N + E \cdot u_a(\pi)$$

Generating Advice

Given a maze M we view the maze as a graph, and use the A^* search algorithm to find a path π from the start node S to the target node T with the minimal expected cost $e(\pi)$. We do not use node recognition, therefore the maze is really spread out as a tree. When given a vertex v in the tree, there is a unique path from S to v, denote this path as θ. The cost

function for node v is $e(\theta)$ and we use the minimal agent cost of traveling between v to T as the heuristic function. The minimal agent cost to travel from each vertex to T can be efficiently calculated for all vertexes using the Dijkstra's algorithm starting at T.

To limit the manipulation effect of UMPA, the search considers only paths with cuts where the agent does not gain from the user's taking them. That is, the agent prefers the user to take the advised path and does not benefit from his deviation. Formally, UMPA considers only paths such that for any suffix $\tau = \pi^i \cdots \pi^{l(\pi)}, i \geq 1$ holds $e(\tau) \geq u_a(\tau)$. If A^* stops with a path that does not satisfy the condition above it will be rejected, and A^* will be forced to continue the search.

Experimental Evaluation

We have developed an online system that allows people to solve path selection problems in a maze. It can be accessed via http://cupkey.com/selfmazeplayer.swf. The maze design was chosen to remove from the experiments all effects of familiarity with the navigation network. We also chose a maze design that does not directly resemble a street map in order to avoid the effects of city navigation experience. Furthermore, every human subject was presented with a single instance of the problem in order to exclude effects of learning or trust. All of our experiments were run using the Amazon's Mechanical Turk service (Amazon Mechanical Turk 2010). Participation in our study consisted of 701 subjects from the USA: 383 females and 298 males. The subjects' ages ranged from 18 to 72, with a mean of 37.

Experimental Setup

Each experiment consisted of a single-colored maze panel similar to the one depicted in Figure 1. A single panel was shown to each participant. In some panels an advised path was also presented (as described below). Subjects were informed that the advised path was calculated *both* to reduce its length *and* the number of color switches, although they were not given color switching cost W.

The subject's task was to select the shortest path through the Maze. As a stimulus all subjects were guaranteed a small monetary bonus inversely proportionate to the length of the path they have selected. A set of questions was presented both prior to and following the task execution. The questions prior to the task execution were designed to verify understanding of the task. The post-task questions were designed to assess the general attitude towards computer advice and the subjective evaluation of the advised path quality.

We have used four distinct mazes, M_1,...,M_4, all with a grid size of 80×40. Their complexity was chosen such that users would find it difficult to compute the shortest path in the limited time allotted for the task. In all the experiments we have used the color switch weight of $W = 15$.

For each maze M_i, $i = 1, .., 4$, we used the other mazes for the training data, i.e., $\mathcal{M}_i = \{M_1, ..., M_4\} \setminus \{M_i\}$ and collected the examples of Ψ_i in $\mathcal{M}_i$. That is, UMPA generated the advice for M_i after performing its learning process over Ψ_i, i.e., the examples taken from the other mazes. Finally, we have used the following settings of UMPA parameters: the length of a cut was bound to $L_1 = 40$; a cut's potential increase in length to $L_2 = 20\%$ of the corresponding original segment; finally, the discount factor in the greediness feature calculation was set to $\delta = 0.95$.

Before starting the UMPA evaluation we first had to determine the basic algorithm with which to compare it. There two extreme cases. In one extreme we can look for advice that will be generated only from the point of view of the user, i.e., the one that minimizes the length of the path. In the other extreme, there is the path that is generated only from the point of view of the agent, i.e., the one that minimizes the agent's cost u_a. While we expect that the acceptance of the first advice by users will be high, the number of *Advice Ignorers* will be small, and the probability of deviation also will be small, this path may yield high cost to the agent. On the other hand, when providing the second advice we run the risk that most of the users will ignore it, while the ones that will accept it will yield the highest agent benefits. Finally, we should consider not giving advice at all. The implementation of the three options is straightforward and we name them as follows:

- *No advice (silent)*, where no advice was present in the maze panel,
- *Shortest*, i.e. the shortest path through the maze was presented as the advised path,
- *Agent path*, i.e. an optimal path with respect to u_a was presented as an advised path.

After identifying the basic algorithm we considered the following questions:

- What is the effectiveness of UMPA estimation methods?
- Is UMPA significantly better than the basic algorithm?
- If UMPA is better than the basic algorithm from the agent's point of view, will it, in return, decrease the users' benefits and satisfaction? Or does UMPA yield mutually beneficial results compared to the basic algorithm?

Experimental Results

Comparisons between advice type characteristics were performed using Analyses of Variance (ANOVA). ANOVA is a method of analysis used to determine the level of statistical significance when dealing with more than two groups.

Basic approaches We calculated the effects of Silent, Shortest and Agent advice on the average of agent cost across selected paths by users given the advice in our experiments. The corresponding three columns on the left of Table 1 summarize the results (the lower the better). For all maze panels Agent advised paths have resulted in significantly (with p-value $p < 0.001$) lower cost than the Shortest advice and the Silent advice.

We have also studied the statistics of the advice effect on the user costs (see two right most column of Table 1). As expected, the users costs from the Shortest advice were significantly lower than in the Agent and Silent advice cases. However, we wanted to check whether giving advice that

Table 1: Average Costs of Basic Advice

Maze	Average Agent Cost			Average User Cost		
	Agent	Shortest	Silent	Agent	Shortest	Silent
M_1	580.9	620.5	643.8	163.5	142.6	161.5
M_2	475.9	544.9	515.7	146	133.4	137.6
M_3	454.1	517.4	557.3	133.5	120.3	133.1
M_4	495.8	555.4	522.1	135.4	127.1	138.8
Mean	501.68	559.55	559.73	144.6	130.85	142.75

Table 2: Comparative Average Costs for UMPA

Maze	Average Agent Cost		Average User Cost	
	UMPA	Agent	UMPA	Agent
M_1	531.8	580.9	156.5	163.5
M_2	475.9	475.9	146	146
M_3	477.7	454.1	132.7	133.5
M_4	454.4	495.8	134.1	135.4
Mean	484.95	501.68	142.33	144.6

is the best for the agent decreases the user benefits compared to not giving any advice at all. The results were mixed and no significant difference was found between Agent and Silent. That is, while Agent advice improved significantly the agent's utility, it did not harm significantly the user's utility. So we concluded that UMPA should be compared to Agent advice.

UMPA Evaluation The first step in our UMPA evaluation was to verify the effectiveness in computing $p(M, \pi, \pi^i, \tau)$. We found a high correlation (0.77) between the *p* of a cut and the fraction of users who took it when reaching the cut's divergence node. High correlation (0.7) was also found between the fraction of *Baseline users* given advice π and $p_b(\pi)$. Finally, we obtained a high correlation (0.76) between the estimated value of advice $e(\pi)$ and the average value of the actual selected path in response to advice π in our experiments. This is significant since the correlation between the agent cost of π itself and the empirical average of the selected path was only 0.06.

We then compared UMPA and Agent with respect to the average of agent utility across selected paths by users given the UMPA advice and the Agent advice in our experiments. Consider the two corresponding columns on the left of Table 2 (the lower the better). On average, UMPA outperformed Agent, resulting in significant lower costs ($p < 0.05$).

We also compared the effect of UMPA and Agent advice on the user cost (see the two right most columns of Table 2). To our surprise, the average results of the users that were given UMPA advice were significantly better (lower cost) than the users that were given the Agent advice ($p < 0.05$). In other words, when comparing the UMPA and Agent-advised path generation techniques, both the average utility of the agent and the average utility of human users would increase significantly when using UMPA compared to the Agent advice. So UMPA manipulative advice is *mutually* beneficial as compared to Agent advice.

Finally, we considered the subjective view of the users on the advised path. After finishing the task we presented them with the following two questions: (i) "How good was the advice given to you by the system?" and (ii) "How much did you trust the advice given to you by the system?" The possible answers were on a scale of 1-5, where 5 indicates highest satisfaction and 1 the lowest satisfaction.

The average answers to question (i) were: UMPA – 3.29 and Agent – 3.05, i.e., UMPA advice was considered significantly better than Agent advice ($p < 0.05$). Similarly with respect to trust. The average answers to question (ii) were UMPA — 3.23 and Agent — 2.92, i.e., users trusted UMPA advice significantly more than Agent advice ($p < 0.05$).

Conclusions and Future Work

This paper presents an innovative computational model for advice generation in human-computer settings where agents are essentially self-interested, but share some common goals with the human. To assess the potential effectiveness of our approach we performed an extensive set of path selection experiments in mazes. Results showed that the agent was able to outperform alternative methods that solely considered the agent's or the person's benefit, or did not provide any advice.

In future work, we will extend this approach to settings in which people and computers interacted repeatedly, requiring the agent to reason about the effects of its current advice on people's future behavior. We also mean to adapt our model to real road maps. We expect that the only significant change will be a different greediness heuristic. Naturally, the cost of color change will need to be replaced by a more practical measure, such as switching of transportation mode or road type (speed limit, interstate, etc.). Long term research should consider other self-interested recommendation systems such as travel agents.

Acknowledgments

We acknowledge General Motors for supporting this research.

References

Amazon Mechanical Turk services. http://www.mturk.com/, 2010.

D. Antos and A. Pfeffer. Using reasoning patterns to help humans solve complex games. In *IJCAI*, pages 33–39, 2009.

C. G. Chorus, E. J. Molin, and B. van Wee. Travel information as an instrument to change car drivers travel choices. *EJTIR*, 6(4):335–364, 2006.

M. Duckham and L. Kulik. "Simplest" paths: Automated route selection for navigation. In *LNCS*, volume 2825, pages 169–185, 2003.

R. J. Hanowski, S. C. Kantowitz, and B. H. Kantowitz. Driver acceptance of unreliable route guidance information. In *Proc. of the Human Factors and Ergonomics Society*, pages 1062–1066, 1994.

M. Hipp, F. Schaub, F. Kargl, and M. Weber. Interaction weaknesses of personal navigation devices. In *Proc of AutomotiveUI)*, 2010.

H. H. Hochmair and V. Karlsson. Investigation of preference between the least-angle strategy and the initial segment strategy for route selection in unknown environments. In *Spatial Cognition IV*, volume 3343 of *LNCS*, pages 79–97, 2005.

S. K. Hui, P. S. Fader, and E. T. Bradlow. Path data in marketing. *Marketing Science*, 28(2):320–335, 2009.

X. J. Kuang, R. A. Weber, and J. Dana. How effective is advice from interested parties? *J. of Economic Behavior and Organization*, 62(4):591–604, 2007.

K. Park, M. Bell, I. Kaparias, and K. Bogenberger. Learning user preferences of route choice behaviour for adaptive route guidance. *IET Intelligent Transport Systems*, 1(2):159–166, 2007.

L. Rayo and I. Segal. Optimal information disclosure. *J. of Political Economy*, 2010.

K.-F. Richter and M. Duckham. Simplest instructions: Finding easy-to-describe routes for navigation. In *Proc. of the Geographic Information Sience*, volume 5266 of *LNCS*, pages 274–289, 2008.