Multi-Modal Journey Planning in the Presence of Uncertainty

Adi Botea
IBM Research
Dublin, Ireland

Evdokia Nikolova
Dept. of Computer Science and Engineering
Texas A&M University

Michele Berlingerio
IBM Research
Dublin, Ireland

Abstract

Multi-modal journey planning, which allows multiple types of transport within a single trip, is becoming increasingly popular, due to a strong practical interest and an increasing availability of data. In real life, transport networks feature uncertainty. Yet, most approaches assume a deterministic environment, making plans more prone to failures such as major delays in the arrival. We model the scenario as a non-deterministic planning problem with continuous time and time-dependent probabilities of non-deterministic effects. We present new hardness results. We introduce a heuristic search planner, based on Weighted AO* (WAO*). The planner includes search enhancements such as sound pruning, based on state dominance, and an admissible heuristic. Focusing on plans that are robust to uncertainty, we demonstrate our ideas on data compiled from real historical data from Dublin, Ireland. Repeated calls to WAO*, with decreasing weights, have a good any-time performance. Our search enhancements play an important role in the planner's performance.

Introduction

Journey planning systems have become increasingly popular. Factors that favour the progress include the practical need among users for reliable journey plans, the ubiquity of mobile devices, and the increasing availability of relevant sensor data, such as bus GPS records. In its simplest form, journey planning is a shortest path problem: given a graph representation of a transportation network, find a shortest route between a start location and a destination. A more realistic modelling, however, such as considering uncertainty, can lead to problem variants that are computationally hard (Nikolova et al. 2006a; Nonner 2012).

Multi-modal journey planning allows combining multiple transportation means, such as bus riding and cycling, into one single trip. Most existing approaches to multi-modal journey planning operate under deterministic assumptions (Zografos and Androutsopoulos 2008; Liu and Meng 2009). However, in real life, key information needed in journey planning, such as bus arrival times, is characterised by uncertainty. Even small variations of bus arrival times can result in a missed connection, with a great negative impact on the actual arrival time at the destination. Journey plans should be able to handle such unexpected situations without compromising the plan quality (e.g., arrival time) beyond some acceptable level.

This paper investigates multi-modal journey planning in the presence of uncertainty. Our contributions are in the modelling of the problem, the hardness results, the solver, with an admissible heuristic and formal pruning rules based on state dominance, and the empirical evaluation.

The presented approach is probabilistic planning in an and/or state space with continuous time, stochastic action durations, and probabilistic action effects. Uncertainty is modelled in the estimated times of arrival (ETAs) of scheduled-transport vehicles (e.g., buses) at stops, and the duration of actions such as walking and cycling. This leads to non-determinism in the outcomes of actions such as boarding a scheduled-transport vehicle, which can either succeed or fail. The structure of the discrete and/or search space is directly impacted by the time-related probability distributions defined in the model. For example, the likelihood that a user can catch a bus depends on the user's ETA, and the bus ETA at a given bus stop.

We present an NP-hardness result, with the hardness stemming from the *dynamically changing distributions* of the travel time random variables. This complements previous results for related problems with *static* distributions (Nikolova et al. 2006a).

In multi-modal journey planning, there can be multiple quality criteria, including the travel time, the number of interchanges, and the monetary cost. In this work, we choose to optimize measures that quantify the robustness of a plan to uncertainy. To our knowledge, this is one of the first works in multi-modal journey planning addressing plan robustness.

We implemented a planner that performs heuristic search in an and/or tree, computing probabilistic plans (i.e., contingent plans where branches have probabilities). We evaluate an any-time search strategy that iteratively runs Weighted AO*, with decreasing weights. The planner uses an admissible heuristic, stored as a fast look-up table. We introduce and prove formal properties of a pruning strategy based on state dominance. Our experiments use data built from real historical GPS traces of buses across Dublin, Ireland. The results demonstrate that our approach is viable. In many cases, plans of good quality are provided fast, with our search enhancements playing an important role in the planner's speed.

Defining the Input

A multi-modal journey planing task is determined by a user query (i.e., a request to compute a journey plan), and the status of a multi-modal transport network. This section contains formal definitions of these. The transportation modes (means) that we focus on are scheduled-transport modes (with buses as a running example), bicycles available in a city's bike sharing network, and walking. Car parkings could be handled similarly to bike stations.

A *user query* is a tuple $\langle o, d, t_0, m, q\rangle$, where: o and d are the origin and the destination locations; t_0 is the start time; m specifies a subset of transportation means that should be considered; and q states maximal values (quotas) to parameters such as the *expected walking time*, the *expected cycling time*, and the *number of vehicles* (i.e., the max number of buses and bicycles that could be used in a trip).

A *multi-modal network snapshot* is a structure $\langle \mathcal{L}, \mathcal{R}, \mathcal{I}, \mathcal{W}, \mathcal{C}\rangle$ that encodes (possibly uncertain) knowledge about the current status and the predicted evolution of a transport network. $\mathcal{L}$ is a set of *locations* on the map, such as the stops for scheduled transport (e.g., bus stops) and the bike stations. In addition, given a user query, the origin and the destination are added to $\mathcal{L}$, unless already present. $\mathcal{R}$ is a set of *routes*. A route r is an ordered sequence of $n(r)$ locations, corresponding to the stops along the route. For a given route label in real life, such as "Route 39", there can be multiple route objects in set $\mathcal{R}$. For example, there is one distinct route object for each direction of travel (e.g., eastbound and westbound). Also, variations across different times of the day, if any, result in different route objects.

$\mathcal{I}$ is a collection of *trips*. Informally, each trip is one individual scheduled-transport vehicle (e.g., a bus) going along a route (e.g., the bus that starts on route 39 at 4:30pm, going westbound). Formally, a trip i is a structure $\langle r, f_{i,1}, \dots f_{i,n(r)}\rangle$, where r is the id of its route, and each $f_{i,k}$ is a probability distribution representing the estimated arrival time at the k-th stop along the route.

$\mathcal{W}$ and $\mathcal{C}$ provide walking times and cycling times, for pairs of locations, as probability distributions.

Part of the snapshot structure, such as the notions of routes and trips, resembles the GTFS format[1]. GTFS, however, handles no stochastic data and no bike-specific data.

The way the network snapshot is defined is partly justified by practical reasons. It encodes available knowledge about the current status and about the predicted evolution over a given time horizon (e.g., estimated bus arrival times). At one extreme, the snapshot can use static knowledge, such as static bus schedules, or static knowledge based on historical data. At the other extreme, increasingly available sensor data can allow to adjust the predicted bus arrival times frequently, in real time (Bouillet et al. 2011). Our network snapshot definition is suitable for both types of scenarios, allowing to adjust the approach to the level of accuracy available in the input data.

[1]https://developers.google.com/transit/gtfs/reference

And/Or State Space

The search space is an and/or state space with continuous time, stochastic action durations, and probabilistic action effects. States and transitions between states are discrete, just as in a typical and/or space. The probabilities associated with the non-deterministic effects of an action depend on stochastic time parameters, such as the estimated time of arrival (ETA) of a trip (e.g., a bus) and a user's ETA at a (bus) stop. In this section we describe the and/or state space, with a focus on how the continuous, stochastic modelling of time-related parameters affects the structure of the state space.

The core components of a state s include a position p_s, a density function t_s,[2] and a vector of numeric variables q_s. Some auxiliary state components are used for correctness (e.g., to define what types of actions apply in which states) and for pruning. In this section, we focus on the core components, and particularly the time distribution t_s.

The position $p_s \in \mathcal{L} \cup \mathcal{I}$ can be either a location or a trip id, in which case the user is aboard a trip. The time t_s is a probability distribution representing the uncertain time when the user has reached position p_s. Quotas left in this state (e.g., for walking time, cycling time, number of vehicles) are available in q_s. In the initial state, the quotas are taken from the user query. They get updated correspondingly as the search progresses along an exploration path.

Actions considered are `TakeTrip`, `GetOffTrip`, `Cycle`, and `Walk`. `TakeTrip` actions can have up to two non-deterministic outcomes (success or failure), as discussed below. Other actions always succeed. The probabilities associated with non-deterministic branches (effects of an action) are determined dynamically, when the transition is generated, based on one or two time-related probability distributions, as shown later. Waiting for a trip to arrive is implicitly considered, being the time between the user's arrival at a stop and the time of boarding a trip.

We start by describing the transitions, their probabilities and the successors generated with a `TakeTrip` action applied in a state s with the location p_s being a bus stop. Let i be a trip that stops at p_s, the k-th stop along the route. The ability to catch trip i depends on the arrival time of the user at p_s, and the arrival time of the trip. Recall that t_s, available from s, models the user's arrival time at location p_s, whereas $f_{i,k}$, available as input data from the network model, represents the arrival time of the trip. Let U be a random variable corresponding to t_s, and B a random variable corresponding to $f_{i,k}$.

Proposition 1. *Under the assumption that U and B are independent, the probability of being able to catch the trip is* $P(U < B) = \int_{-\infty}^{\infty} f_{i,k}(y) \int_{-\infty}^{y} t_s(x)dxdy$.

Proof Sketch: Fix a value y for the trip arrival time. The probability that the user makes it before y is $P(U < y) = \int_{-\infty}^{y} t_s(x)dx$. Generalizing to all values y across the range, we obtain the desired formula. □

In the general case of arbitrary probability distributions, the previous formula could be difficult to solve analytically.

[2]For clarity, we stick with the case of continuous random variables. Discrete distributions are handled very similarly.

Fortunately, Monte-Carlo simulation provides an easy solution for estimating the probability $P(U < B)$: draw a sample from U and one from B, repeat the process n times, and count how often the former is smaller.

In state s, $P(U < B) > 0$ is a precondition for the action of boarding trip i. When $0 < P(U < B) < 1$, there are two non-deterministic effects, corresponding to success and failure, each outcome having probability $P(U < B)$ and $1 - P(U < B)$ respectively. When $P(U < B) = 1$, only the successful branch is defined.

In the successor state s' along the successful branch, the time of boarding the trip is a random variable corresponding to the estimated arrival time of the trip, conditioned by the fact that the user makes it to the stop before the trip. Using a conditional probability in this case, as opposed to simply the trip's estimated arrival time, is preferable to avoid negative waiting times.

Proposition 2. *The density function $t_{s'}$, corresponding to the time of boarding the trip, is*

$$t_{s'}(y) = \frac{f_{i,k}(y) \int_{-\infty}^{y} t_s(x)dx}{\int_{-\infty}^{\infty} f_{i,k}(y) \int_{-\infty}^{y} t_s(x)dxdy}.$$

Proof. The boarding time is a random variable Z corresponding to B, conditioned by $U < B$. Applying the Bayes rule to the continuous variable B and the discrete binary variable $U < B$, we obtain: $t_{s'}(y) = f_Z(y) = \frac{P(U<B|B=y)f_{i,k}(y)}{P(U<B)} = \frac{f_{i,k}(y) \int_{-\infty}^{y} t_s(x)dx}{\int_{-\infty}^{\infty} f_{i,k}(y) \int_{-\infty}^{y} t_s(x)dxdy}$ $\square$

Corollary 1. *If $P(U < B) = 1$, then $t_{s'} = f_{i,k}$.*

That is, when the user certainly makes it to the stop before the trip (bus), then the distribution of the boarding time is the same as the distribution of the trip arrival time.

On the failed branch, the time of reaching the child state s'' is a random variable with the same distribution as t_s, corresponding to the user's time of arrival at p_s.

Obviously, the locations in the two successor states are $p_{s'} = i$ (i.e., the user is aboard the trip) and $p_{s''} = p_s$ (the user is still at the stop). Auxiliary state variables are used to forbid multiple (failed) attempts to board the same trip[3] at location p_s, either in a sequence or interleaved with failed attempts to board other trips. This is implemented in a straightforward way, in the style of a STRIPS encoding, and the exact details are beyond the scope of this paper.

When aboard a trip i, `GetOffTrip` actions are defined for all stops along the route subsequent to the boarding stop. When applying a `GetOffTrip` action, to get off at the k-th stop along the route of trip i, the time associated with the successor state s' is the same as the estimated arrival time of the trip: $t_{s'} = f_{i,k}$.

In the case of `Cycle` and `Walk` actions applied in a state s, the time of reaching the destination depends on both the starting time t_s, and the duration of the action (with a given density function g), both of which can be stochastic. Cycling (or walking) time is assumed to be independent of the starting time. Thus, the density function of reaching the destination (i.e., the time distribution of the successor state s') is the convolution of the two densities t_s and g: $t_{s'}(x) = (t_s * g)(x) = \int_{-\infty}^{\infty} t_s(y)g(x-y)dy.$

[3]Of course, taking a later trip on the same route is allowed.

As hinted earlier, distributions and probabilities derived in this section could be difficult to solve analytically. In practice, Monte Carlo simulation offers an alternative. The only requirement is the ability to sample from original distributions in the data (e.g., $f_{i,k}$, walking time distribution, cycling time distribution), which in turn leads to the ability to sample from all relevant distributions. For example, in the summation $X + Y$ of two independent variables, sample from X, sample from Y and take their sum. In a conditional distribution $(X|X < Y)$, sample from X, sample from Y, and reject those instances where the condition is invalidated.

In summary, a continuous, stochastic time modelling has a direct impact on the discrete structure of the and/or space, influencing the applicability of actions, the number of (non-deterministic) branches of an action, and their probabilities. In addition, plan quality metrics are impacted, such as the arrival time. Plan quality is discussed in the next section.

Plan Robustness to Uncertainty

As mentioned in the introduction, our optimality metrics quantify the robustness of a plan to uncertainty. Consider a contingent plan π where branches have associated probabilities, and the times associated with states are stochastic, being given as a probability distribution. Branch probabilities induce a discrete probability distribution across the pathways in the plan (i.e., each pathway π_i in the contingent plan occurs with probability p_i, where $\sum p_i = 1$).

Pathway π_i has an arrival time given by t_s, the probability distribution of its goal leaf state s. From this, we extract c_i, the maximum arrival time for a given confidence threshold. The main criterion is minimizing $c_{max} = \max_i c_i$. Ties are broken by preferring plans with a better expected arrival time, $c_{exp} = \sum_i p_i c_i$. Given two plans with costs c and c', we say that $c < c'$ iff $(c_{max} < c'_{max}) \vee (c_{max} = c'_{max} \wedge c_{exp} < c'_{exp})$ (lexicographic ordering).

Hardness Results

In this section, we develop a theoretical foundation for understanding the complexity of the multi-modal journey planning problem. To pin down causes of difficulty, we identify *simplified* problems that turn out to be computationally difficult. We contrast our worst-case cost objective with the more standard expected cost objective in terms of problem hardness. The main conclusion from this section is that when looking for a *robust* routing policy that minimizes the worst-case travel time, the computational hardness stems from the dynamically varying distributions.

Consider a graph $G = (V, E)$ with $|V| = n$ nodes, $|E| = m$ edges and a given source-destination pair of nodes S, T. The edge costs are independent random variables coming from given distributions. A routing policy specifies, for each node in the graph and for each set of random variable observations, the best successor to continue the route on. We consider two variations of the problem under the following assumptions on random variable observations:

Assumption 1. *The actual edge cost is observed once an edge is traversed.*

Assumption 2. *The actual edge cost is observed once an endpoint of the edge is reached.*

The second assumption is associated with the Canadian Traveller problem (Papadimitriou and Yannakakis 1991; Nikolova and Karger 2008), in which the goal is to find an optimal policy for reaching from the source to the destination in minimum *expected* time.

In multimodal journey planning both assumptions can arise. For example, whether a bridge is traversable or not depends on its status (open or closed), which can be observed when a user arrives at the bridge. On the other hand, riding a bus has an associated stochastic cost whose actual value can only be observed once the end-point is reached.

For a more general picture, we consider multiple combinations of Assumptions 1 and 2. In the *standard* setting, Assumption 1 holds, whereas Assumption 2 holds in the *look-ahead* setting. The *mixed* setting considers both assumptions. We now examine the computational complexity for the associated problems with the two objective functions in the different settings.

Expected cost objective

Standard setting When our goal is to reach the destination with minimum expected cost, by linearity of expectation, it suffices to replace each edge random variable cost with its expectation and run a deterministic shortest path algorithm for finding the optimal route. The expected value of the edges that have not yet been traversed does not depend on the edges traversed so far. Therefore, the offline optimal solution (optimal plan) is also an optimal policy.

Proposition 3. *The standard setting under the expected cost objective can be solved in polynomial time via a deterministic shortest path algorithm.*

Look-ahead and mixed setting The look-ahead setting with expected cost objective is also known as the Canadian traveller problem, which is known to be #P-hard (Papadimitriou and Yannakakis 1991). Since the mixed setting includes the look-ahead setting as a special case, it is at least as hard.

Worst-case cost objective

The worst-case cost objective is well-defined only if there exists a path from the source to the destination along which all edges costs have finite upper bounds. We have this assumption when talking about the worst-case cost objective. On the other hand, the expected cost can be finite even if a distribution includes infinite values, as in the case of a Normal distribution, for example.

Standard setting In the standard setting, edges traversed so far have no effect on the values of edge costs that have not yet been traversed. Therefore, we can simply replace all edge costs by their maximal values and run a deterministic shortest path algorithm with respect to these values.

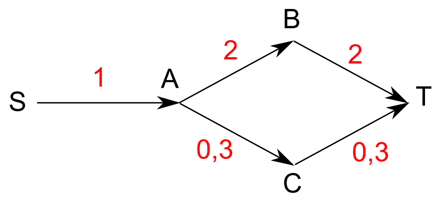

Figure 1: Worst-case cost routing changes depending on observed edge costs in the look-ahead setting, while it does not change in the standard setting.

Proposition 4. *The standard setting under the worst-case cost objective can be solved in polynomial time via a deterministic shortest path algorithm.*

Note, in particular, that just as in the standard setting under the expected cost objective, the optimal plan and the optimal policy here coincide, namely the shortest path that is computed should always be followed regardless of traversed edges and their observed values along the way.

Look-ahead and mixed setting In the worst case, we will observe the worst case edge values and will go along the shortest path with respect to these worst-case edge values.

Proposition 5. *The look-ahead setting under the worst-case cost objective can be solved in polynomial time via a deterministic shortest path algorithm with respect to edge weights equal to the maximal edge costs.*

The difference from the standard setting above is that while routing, the shortest path from the current node to the destination may change depending on the (look-ahead) observations so far. Consider the example in Figure 1. At the source node S, the path minimizing the worst-case cost is through B. However, upon reaching A and seeing that $AC = 0$, the optimal path becomes the one through C.

Finally, since a shortest path algorithm with respect to edge weights equal to the maximum edge costs yields the optimal routing strategy in both the standard and look-ahead setting, it will do so in the mixed setting, as well.

Proposition 6. *The mixed setting under the worst-case cost objective can be solved in polynomial time via a deterministic shortest path algorithm with respect to edge weights equal to the maximal edge costs.*

Time-varying distributions

In the previous sections, we assumed that the edge distributions do not change with time. We showed that in that case, worst-case cost routing can be performed in polynomial time (in contrast to expected cost routing). In practice, the distributions change with time. For example, peak and off-peak traffic distributions often differ. This changes the problem complexity considerably. In this section, we show that even in the simplest case with only two phases, the problem of just computing the maximum cost along a *given* path is computationally hard, and consequently the problem will be hard under all settings discussed above.

Theorem 1. *Given a graph with stochastic edge costs that come from static distributions before or at time t and from* *different static distributions after time t, it is NP-hard to compute the worst-case cost along a given path.*

Proof. We reduce from the subset sum problem (for a given set of integers $a_1, ..., a_n$ and a target t, decide if there exists a subset which sums to t), which is NP-hard (Garey and Johnson 1979). Without loss of generality, we assume that $a_i \neq 0$ for all $i = 1, ..., n$, since removing the zero elements has no effect on the problem. Consider a single path consisting of n edges $e_1, e_2, ..., e_n$ in that order. Suppose that before time t, the edge costs come from two-valued distributions: edge e_i takes on values $\{0, a_i\}$. After time t, all edge costs are 0. Denote the realized edge costs by d_i, $i = 1, ..., n$.

Without loss of generality, suppose that we start our journey along the path at time 0. The question is, where are we going to be at time t so as to know which distribution to use for each edge? We use the convention that if time t occurs at the start or during traversal of edge e_i, its value comes from the second distribution (namely, its value is 0) and otherwise its value comes from the first distribution.

If e_k is the last edge that follows distribution 1, namely time t occurs at the beginning or inside edge e_{k+1}, then the cost along the path can be expressed as $\sum_{i=1}^{k} d_i = \sum_{i=1}^{k} a_i x_i \leq t$, where $x_i \in \{0, 1\}$ is an indicator variable of whether $d_i = 0$ or $d_i = a_i$. Therefore, the worst-case path cost is given by the program:

$$\max_{k,x}\{\sum_{i=1}^{k} a_i x_i \mid \sum_{i=1}^{k} a_i x_i \leq t\,;\ \sum_{i=1}^{k} a_i x_i + a_{k+1} > t\,; \quad x_i \in \{0,1\} \quad \forall i = 1, ..., k.\} \tag{1}$$

The two constraint inequalities represent the fact that time t occurs at the beginning or inside of edge e_{k+1}. The maximum is taken over $k = 1, 2, ..., n$. (To be precise, when $k = n$, the second inequality is eliminated.)

Next, we will show that the value of program (1) is t if and only if there is a subset of $\{a_1, ..., a_n\}$ that sums to t.

We first show that the maximum in program (1) is attained at $k = n$, namely, it is equivalent to the following program:

$$\max_{x}\{\sum_{i=1}^{n} a_i x_i \mid \sum_{i=1}^{n} a_i x_i \leq t;\ x_i \in \{0,1\}\, \forall i = 1, ..., n.\} \tag{2}$$

Claim 1. *Programs (1) and (2) have the same optimal value.*

Proof. First, we note that the maximum value of program (1) can only increase if we remove the second constraint. On the other hand, with the second constraint removed, the maximum $(x_1, ..., x_k)$ of

$$\max_{x}\{\sum_{i=1}^{k} a_i x_i \mid \sum_{i=1}^{k} a_i x_i \leq t;\ x_i \in \{0,1\}\, \forall i = 1, ..., k.\} \tag{3}$$

extended by $x_j = 0$ for $j > k$ is a feasible solution to program (2), and thus its value is no more than the value of program (2), for all k. So, the value of program (1) is less than or equal to the value of program (2). On the other hand, the value of program (2) is less than or equal to that of program (1), since the latter is a maximum over program (2) (for $k = n$) and other programs (for $k < n$). Thus, the values of the two programs are equal. □

Claim 2. *The value of program (2) is t if and only if there is a subset of $\{a_1, ..., a_n\}$ that sums to t.*

Proof. Suppose the value of program (2) is t. Then, the objective of program (2) specifies a subset of $\{a_1, ..., a_n\}$ that sums to t. Conversely, suppose there is a subset that sums to t. Taking $x_i = 1$ if a_i is in the subset and $x_i = 0$ otherwise gives a feasible solution to the program of value t. Therefore, the maximum of the program is at least t. On the other hand, the maximum is at most t by the inequality constraint. Therefore, it is exactly t, as desired. □

To conclude, for any given set of integers $a_1, ..., a_n$, we have provided a polynomial-time transformation to the worst-case cost problem, whereby a subset of the integers sums to t if and only if the worst-case cost is t. Therefore, the worst-cost problem is NP-hard. □

Searching for a Plan

Being interested in optimizing plan robustness to uncertainty, as defined earlier, we focus on methods that eventually output optimal plans, possibly in an any-time manner. Repeated calls to Weighted AO*, with decreasing weights, provide an any-time behavior, computing increasingly better solutions. In deterministic search, a similar strategy has been used in the Lama planner (Richter and Westphal 2010) using Weighted A* (Pohl 1970) as a baseline algorithm.

In the evaluation, the and/or space is explored as a tree, since pruning rules described later in this paper work in a setting where the path to a state is relevant.

Admissible Heuristic

Given a destination location, an admissible heuristic is precomputed in low-polynomial time and stored as a look-up table. For every location (stops, bike stations, origin), the table stores an admissible estimation of the travel time from that location to the destination. The relaxation considered in the computation of the heuristic ignores waiting times, slow-downs along a route segment, any actual quotas left in a state, and the possibility that boarding a trip could fail.

The heuristic look-up table is built with a regression run of the Dijkstra algorithm, in a graph obtained as follows. Each location is a node. There are multiple kinds of edges, each corresponding to one transport mode (taking a trip, cycling, walking). Two consecutive stops along a route are connected with an edge whose cost is a lower bound for the travel time of a trip along that segment. If the shortest walking time between two given locations is smaller than a threshold T_W given as a parameter, an edge is added between the two locations. The cost is a lower bound on the walking time between the two locations. Edges corresponding to cycling are created similarly, with a different threshold parameter T_C.

Let q_w and q_c be the maximal walking time, and the maximal cycling time set in a given user query. If $T_W \geq q_w$ and $T_C \geq q_c$, then the heuristic computed as above is admissible for the instance at hand.

The computation outlined in this section is simple. Under the assumption of a deterministic environment, which is the norm with existing multi-modal journey planners, such a computation, or variations of it, could be used to provide rough deterministic journey plans, which ignore, for instance, the waiting time. In contrast, we use such information as heuristic guidance to obtain more detailed plans, with risk hedging capabilities.

Relaxing uncertainty by optimistically assuming that the best outcome will be available is relatively common. In the Canadian Traveler's Problem, optimistic roadmaps assume that all roads will always be accessible (Eyerich et al. 2010).

Pruning with State Dominance

State dominance (Horowitz and Sahni 1978) can help pruning a search space. In our domain, we define dominance as $s \prec s'$ iif $p_s = p_{s'}$, $q_s \geq q_{s'}$ component-wise, and $P(T_s \leq T_{s'}) = 1$.[4] In other words, a state dominates another if they have the same location, and the former state is better with respect to both time and relevant quotas.

State dominance is generally slower than duplicate detection, as the latter can be implemented efficiently with hash codes, such as Zobrist hashing (Zobrist 1970), and transposition tables. Thus, an efficient (i.e., fast) implementation of a full-scale dominance check, comparing every new state against previously visited states, is not trivial. Here we present an effective approach to partial dominance checking.

Given a state s, let its *subtree cost* $\mathrm{sc}(s)$ be the optimal cost of a plan rooted at s (i.e., plan to an instance where the starting time, starting location and quotas are taken from s).

Lemma 1. *Given two states, if $s \prec s'$, then $sc(s) \leq sc(s')$.*

Let $\xrightarrow{d}$ denote a deterministic transition. Similarly, $\xrightarrow{s}$ denotes a transition on a successful branch, and $\xrightarrow{f}$ a transition on an action failed branch. In particular, successful branches include deterministic branches. Symbols $\overset{d}{\rightsquigarrow}$, $\overset{s}{\rightsquigarrow}$, and $\overset{f}{\rightsquigarrow}$ denote sequences of one or more transitions of a given type.

Observation 1. *Let a be an action with two non-deterministic outcomes in a valid plan π. Removing action a from π, together with the entire subtree along the successful branch, results in a valid plan.*

Lemma 2. *Let $s \xrightarrow{s} s'$ and $s \xrightarrow{f} s''$ be two branches of an action a in an optimal plan. Then $sc(s') \leq sc(s'')$.*

The intuition for the proof is that, if $\mathrm{sc}(s') > \mathrm{sc}(s'')$, then removing action a and its subtree under the successful branch will result in a strictly better plan.

Theorem 2. *Let s_1, s_2 and s_3 be three states such that $s_1 \xrightarrow{d} s_2$ and $s_1 \overset{s}{\rightsquigarrow} s_3$. If $s_2 \prec s_3$, then assigning an ∞ score to s_3 does not impact the correctness, the completeness and the optimality of a tree search.*

Proof Sketch: It can be shown that a best solution π_3 containing $s_1 \overset{s}{\rightsquigarrow} s_3$ cannot possibly beat a best solution π_2 that contains $s_1 \xrightarrow{d} s_2$. The sequence $s_1 \overset{s}{\rightsquigarrow} s_3$ contains zero or more successful, but not deterministic branches, which leads to zero or more failed transitions branching out from pathway $s_1 \overset{s}{\rightsquigarrow} s_3$. Applying Lemma 2 recursively, for each such failed branch, starting with the branch closest to s_3, we obtain that $\mathrm{sc}(s_3) \leq c_{13}$, where c_{13} is the cost of a best plan rooted at s_1 that contains $s_1 \overset{s}{\rightsquigarrow} s_3$. From Lemma 1, $\mathrm{sc}(s_2) \leq \mathrm{sc}(s_3)$. Thus, $\mathrm{sc}(s_2) \leq \mathrm{sc}(s_3) \leq c_{13}$, which leads to $\mathrm{cost}(\pi_2) \leq \mathrm{cost}(\pi_3)$.

□

Theorem 2 allows defining the following pruning strategies. First, consider a sequence $s_1 \rightarrow s_2 \rightarrow s_3$, where the first action is boarding a trip i at the k-th stop, and the next action is getting off at the m-th stop. Let L be the set of intermediate stops from index $k+1$ to $m-1$. On any subsequent path from s_3 that contains only successful transitions, states with $p_s = l \in L$ should be considered deadends.

Secondly, consider a pathway in the search tree $s_1 \overset{s}{\rightsquigarrow} s_3$. Assume that boarding a trip i from state s_1 (action a) is possible with a deterministic transition. Assume that boarding trip i from state s_3 (action b) is possible (with a given probability of success $P > 0$). Furthermore, assume that p_{s_1} is an earlier stop than p_{s_3} along i's route. Then, there is no need to consider action b. Both rules are instantiations of the more generic claim described in Theorem 2.

Finally, consider a sequence $s_1 \overset{s}{\rightsquigarrow} s_3$ such that $p_{s_1} = p_{s_3}$. Then, s_3 can be pruned away, thus forbidding to revisit a location on a sequence of successful transitions. This rule falls slightly outside the context of Theorem 2, but its correctness can be proven in a similar manner.

Notice the importance of the conditions that the path $s_1 \overset{s}{\rightsquigarrow} s_3$ should contain only successful branches, and that $s_1 \xrightarrow{d} s_2$ should be deterministic. We illustrate this with a counter-example, showing that it can make sense to re-visit a given location s, if the path $s \rightsquigarrow s$ contains at least one failed branch. Consider that s is a stop and there is ample time to wait for the next trip b. An optimal policy could be to walk to a nearby stop and attempt to take a faster, express connection. If unable to catch the express line (*failed branch*), return to s and catch trip b.

Experiments

We implemented our planner in C++, and ran experiments on a machine with Intel Xeon CPUs at 3.47 GHz, running Red Hat Enterprise 5.8. The input data were created starting from real "static" data (location coordinates, bus routes) and real historical data (bus GPS records) from Dublin, Ireland. A total of 3617 locations considered include 3559 bus stops, 44 bike stations, and 14 points of interest. Buses in Dublin report their GPS location every 20 seconds. Based on one month of historical GPS data collected during working days, we have computed the estimated time of arrival (i.e., mean and standard deviation) for 3707 bus trips, corresponding to 202 bus routes. We have assumed a normal noise in the ETAs, bounded to a 99.7% coverage area. A similar type of noise is used for walking and cycling times, with a standard deviation of 20 seconds. When Monte Carlo simulations are used for manipulating random variables, we generate 500 samples from each distribution. There are cases when no simulation is needed. For example, given two Normal distributions $V_i \sim N(\mu_i, \sigma_i^2), i = 1, 2$, if $\mu_1 + 3\sigma_1 < \mu_2 - 3\sigma_2$,

[4] T_s is the random variable with the density function t_s.

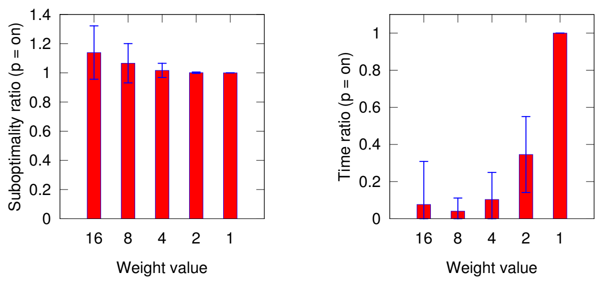

Figure 2: Any-time performance: solution quality and search effort normalised to the case $w = 1$.

the probability $P(V_1 > V_2)$ is virtually negligeable.

Six hundred user queries were created as follows. Set A, with 300 instances, is created using 100 tuples with random start and destination locations (among the 3617 locations considered), and a random starting time. The total number of vehicles (buses/bikes) planned in a trip, an important itinerary feature in travellers' perception, is varied from 2 to 4, to a total of 300 instances. Walking is set to a max of 10 minutes, and cycling to one hour. Set B, of 300 instances, is generated similarly, except that we explicitly encourage non-determinism in catching connections. The start, the target, and the starting time are chosen (randomly) under the additional conditions that catching a bus on both a first and a second leg towards the destination are uncertain. In particular, we found bus stops seeing two buses, belonging to crossing routes, whose arrival time windows overlap, leading to an uncertain connection between the two buses. Of course, this doesn't necessarily mean that a returned plan would include those particular legs.

We evaluated our planner based on WAO* with and without prunning (p = on/off), across a range of weights w. All versions include the admissible heuristic presented earlier. The heuristic is precomputed in less than 4 seconds. Most of this time is used to build the heuristic's graph, given that our implementation naively iterates through all pairs of nodes when building the walk edges. Time and memory are limited to 10 minutes and 4 GB RAM per instance.

Figure 2 summarizes the any-time performance of IWAO*, when $w \in \{16, 8, 4, 2, 1\}$. These charts are for set A with pruning on. Results for other combinations (set of instances and pruning activation) are qualitatively very similar, being omitted to save space. Plotted data correspond to the subset of instances solved with every weight value in use. To be able to aggregate the results over multiple instances, data are normalised to the case $w = 1$. In other words, given an instance ι and a weight w, the c_{max} cost of a corresponding plan is divided by the optimal c_{max} cost, obtained with $w = 1$. Similarly, timing data (right) are normalised relative to the running time of AO* on a given instance. Both mean and standard deviation values are shown. The results show a good any-time performance. Higher weight values (e.g., 4 or 8) under a given threshold (e.g., 8) allow finding good solutions quickly. If sufficient time and memory are available, an optimal solution is eventually returned with AO*. The observed any-time performance also points out

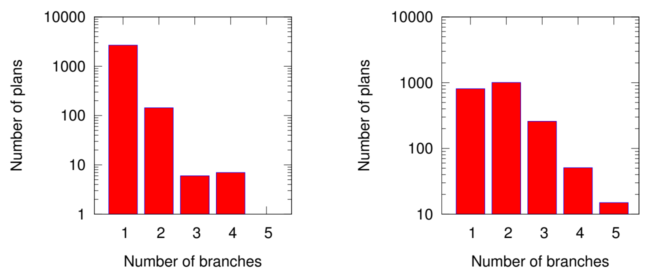

Figure 4: Nr. of branches per plan. Left: set A; right: set B.

the usefulness of the admissible heuristic. Indeed, one cannot perform weighted search unless a heuristic is available. Performing weighted search results not only in computing solutions faster, but also in more instances solved. The percentage of solved instances is 94.4 ($w = 16$), 95.5 ($w = 8$), 88.75 ($w = 4$), 51.75, ($w = 2$) and 38.5 ($w = 1$).

Figure 3 evaluates the impact of state dominance pruning. A chart shows data for the subset of instances solved by both program variants evaluated in that chart (i.e., pruning on/off). Each such instance results in a data point. Points under the diagonal line indicate a speed-up due to the pruning. As shown in the any-time performance analysis, weights such as 1 and 2 produce the high-quality plans, but they correspond to more difficult searches. Figure 3 shows that our pruning helps speeding up such difficult searches, capable of providing plans of optimal or near-optimal quality. Dominance-based pruning confirms the expectation to work in optimal searches ($w = 1$). Also, it has a consistently good behavior in a weighted, sub-optimal search, using a weight such as 2. For larger weights, pruning shows a mixed behavior which, at a closer analysis, is not entirely surprising. The concept of state dominance is closely related to solution quality, since a dominated state cannot lead to a better solution than a dominating one. On the other hand, with a large weight in use, the search aggresively prefers states evaluated as closer to the goal, and it returns a solution without proving its optimality. Thus, it is possible in principle that expanding a dominated state might lead more quickly to a more suboptimal solution.

Figure 4 shows the distribution of contingent plans according to how many pathways they contain. The data show that uncertainty in a transport network can lead to a need for more flexible, contingent plans, as opposed to sequential plans. A sample plan is illustrated in Figure 5.

Related Work

In a problem related to uni-modal transport, such as driving in a road network, Nikolova et al. (2006a) consider stochastic costs for edges, modelled with random variables. More such related models that integrate risk-aversion have been considered by (Loui 1983; Nikolova et al. 2006b; Nikolova 2010; Fan et al. 2005; Nikolova and Karger 2008). Our notion of robust plans is somewhat related to robust optimization (cf. Bertsimas et al. 2011) and percentile optimization in Markov Decision Processes (cf. Delage and Mannor 2010). Wellman et al. (1995) investigate route plan-

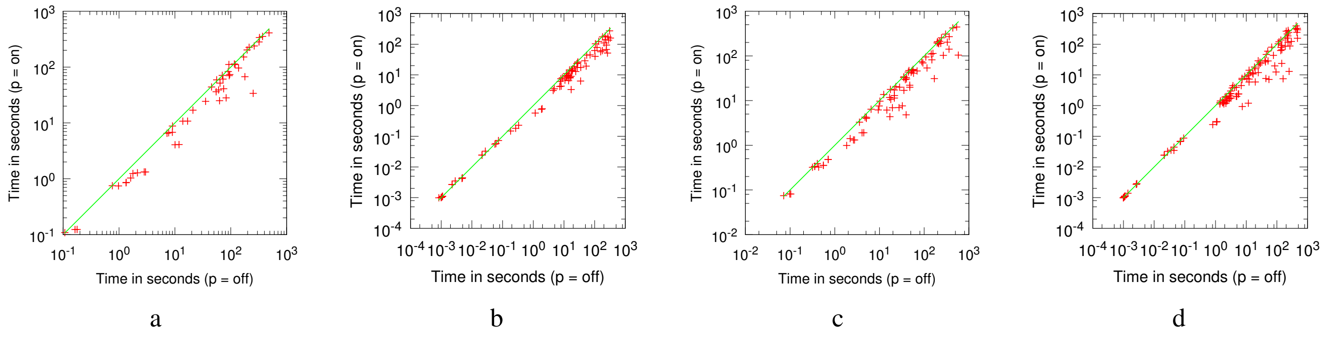

Figure 3: Impact of pruning on a logarithmic scale: a) $w = 1$, set A; b) $w = 1$, set B; c) $w = 2$, set A; d) $w = 2$, set B.

ning under time-dependent edge traversal costs. They take a dynamic programming approach, using a notion of stochastic dominance to prune away dominated partial paths.

Contingent planning (Peot and Smith 1992) is planning under uncertainty with some sensing capabilities. Uncertainty stems from a partially known initial state and/or nondeterministic action effects. Approaches to contingent planning include searching in a space of belief states (Bonet and Geffner 2000; Cimatti and Roveri 2000; Hoffmann and Brafman 2005), and compilation from a belief state space to fully observable states (Albore et al. 2009). Our planner searches in an and/or space of explicit states.

Restarting WA*, or RWA* (Richter et al. 2010), is a strategy that invokes WA* (Pohl 1970) repeatedly. RWA* reuses information across invocations of WA*, such as heuristic state evaluations, and shortest known paths to states. In our program, the heuristic is readily available in a look-up table. Furthermore, we search in a tree, where a node has a unique path from the root. Thus, in our application, it makes sense to keep the invocations of WAO* independent.

Pruning based on state dominance has been applied to problems such as robot navigation (Mills-Tettey et al. 2006), pallet loading (Bhattacharya et al. 1998) and test-pattern generation for combinational circuits (Fujino and Fujiwara 1993). For example, in robot navigation (Mills-Tettey et al. 2006), both a robot's position and its battery level are part of a state. This bears some similarity with our definition of dominance. The difference is in the "resources" considered, such as the battery level versus the (stochastic) time, and the quotas for walking, cycling and vehicle inter-changes.

A few recent works deal with problems in multidimensional networks, such as measuring structural properties (Berlingerio et al. 2011a; 2012) or analysing multidimensional hubs (Berlingerio et al. 2011b).

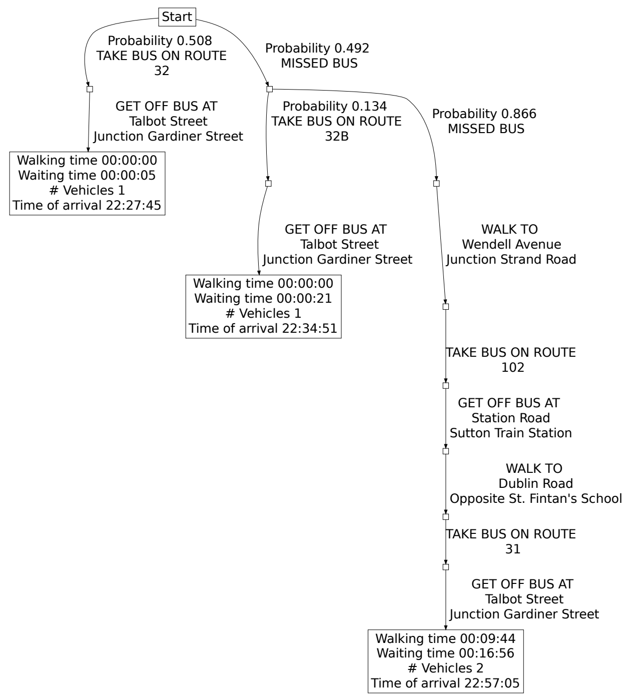

Figure 5: Sample contingent plan.

Summary and Future Work

We address multi-modal journey planning in the presence of uncertainty. We present a new hardness result, showing that, when the distributions of stochastic edge costs vary with time, minimising the maximal travel time is NP-hard. We model the problem as non-deterministic, probabilistic planning, and use a notion of plan robustness to uncertainty as a solution optimality criterion. We develop a solver using Weighted AO* as a baseline search method. Speed-up enhancements include an admissible heuristic and state-dominance pruning rules, whose soundness is formally proven. Our empirical evaluation uses Dublin's bus and shared-bicycle networks, with buses' estimated times of arrival (ETAs) compiled from actual, historical GPS traces. The results show that the approach is viable. The system has a good any-time behavior, and our enhancements play an important role in the planner's performance.

Future work includes speeding up the solver further, and adding more transport modes, such as trains and trams. We plan to connect the system to a platform that computes ETAs in real time (Bouillet et al. 2011). A simplified version of our domain could be used in the IPC probabilistic track.

References

Albore, A.; Palacios, H.; and Geffner, H. 2009. A translation-based approach to contingent planning. In Boutilier, C., ed., *IJCAI*, 1623–1628.

Berlingerio, M.; Coscia, M.; Giannotti, F.; Monreale, A.; and Pedreschi, D. 2011a. Foundations of multidimensional network analysis. In *ASONAM*, 485–489.

Berlingerio, M.; Coscia, M.; Giannotti, F.; Monreale, A.; and Pedreschi, D. 2011b. The pursuit of hubbiness: Analysis of hubs in large multidimensional networks. *J. Comput. Science* 2(3):223–237.

Berlingerio, M.; Coscia, M.; Giannotti, F.; Monreale, A.; and Pedreschi, D. 2012. Multidimensional networks: foundations of structural analysis. *World Wide Web* 1–27.

Bertsimas, D.; Brown, D. B.; and Caramanis, C. 2011. Theory and applications of robust optimization. *SIAM Rev.* 53(3):464–501.

Bhattacharya, S.; Roy, R.; and Bhattacharya, S. 1998. An exact depth-first algorithm for the pallet loading problem. *European Journal of Operational Research* 110(3):610–625.

Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *AIPS-00*, 52–61. AAAI Press.

Bouillet, E.; Gasparini, L.; and Verscheure, O. 2011. Towards a real time public transport awareness system: case study in dublin. In Candan, K. S.; Panchanathan, S.; Prabhakaran, B.; Sundaram, H.; chi Feng, W.; and Sebe, N., eds., *ACM Multimedia*, 797–798. ACM.

Cimatti, A., and Roveri, M. 2000. Conformant planning via symbolic model checking. *J. Artif. Intell. Res. (JAIR)* 13:305–338.

Delage, E., and Mannor, S. 2010. Percentile optimization for markov decision processes with parameter uncertainty. *Oper. Res.* 58(1):203–213.

Eyerich, P.; Keller, T.; and Helmert, M. 2010. High-quality policies for the canadian traveler's problem. In *AAAI*.

Fan, Y.; Kalaba, R.; and J.E. Moore, I. 2005. Arriving on time. *Journal of Optimization Theory and Applications* 127(3):497–513.

Fujino, T., and Fujiwara, H. 1993. A search space pruning method for test pattern generation using search state dominance. *Journal of Circuits, Systems and Computers* 03(04):859–875.

Garey, M., and Johnson, D. 1979. *Computers and Intractability*. New York: Freeman Press.

Hoffmann, J., and Brafman, R. 2005. Contingent planning via heuristic forward search with implicit belief states. In Biundo, S.; Myers, K.; and Rajan, K., eds., *ICAPS*, 71–80. Monterey, CA, USA: AAAI.

Horowitz, E., and Sahni, S. 1978. *Fundamentals of Computer Algorithms*. Computer Science Press.

Liu, L., and Meng, L. 2009. Algorithms of multimodal route planning based on the concept of switch point. *PFG Photogrammetrie, Fernerkundung, Geoinformation* 2009(5):431–444.

Loui, R. P. 1983. Optimal paths in graphs with stochastic or multidimensional weights. *Commun. ACM* 26(9):670–676.

Mills-Tettey, G. A.; Stentz, A.; and Dias, M. B. 2006. Dd* lite: Efficient incremental search with state dominance. In *Proceedings of AAAI*, 1032–1038. AAAI Press.

Nikolova, E., and Karger, D. 2008. Route planning under uncertainty: The Canadian Traveller problem. In *Proceedings of the Twenty-Third Conference on Artificial Intelligence (AAAI)*.

Nikolova, E.; Brand, M.; and Karger, D. R. 2006a. Optimal route planning under uncertainty. In Long, D.; Smith, S. F.; Borrajo, D.; and McCluskey, L., eds., *ICAPS*, 131–141. AAAI.

Nikolova, E.; Kelner, J. A.; Brand, M.; and Mitzenmacher, M. 2006b. Stochastic shortest paths via quasi-convex maximization. In *Lecture Notes in Computer Science 4168 (ESA 2006)*, 552–563.

Nikolova, E. 2010. Approximation algorithms for reliable stochastic combinatorial optimization. In *Proceedings of Approx/Random'10*, 338–351. Berlin, Heidelberg: Springer-Verlag.

Nonner, T. 2012. Polynomial-time approximation schemes for shortest path with alternatives. In Epstein, L., and Ferragina, P., eds., *ESA*, volume 7501 of *Lecture Notes in Computer Science*, 755–765. Springer.

Papadimitriou, C., and Yannakakis, M. 1991. Shortest paths without a map. *Theoretical Computer Science* 84:127–150.

Peot, M. A., and Smith, D. E. 1992. Conditional nonlinear planning. In *Proceedings of the first international conference on Artificial intelligence planning systems*, 189–197. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Pohl, I. 1970. Heuristic search viewed as path finding in a graph. *Artif. Intell.* 1(3):193–204.

Richter, S., and Westphal, M. 2010. The lama planner: Guiding cost-based anytime planning with landmarks. *J. Artif. Intell. Res. (JAIR)* 39:127–177.

Richter, S.; Thayer, J. T.; and Ruml, W. 2010. The joy of forgetting: Faster anytime search via restarting. In Brafman, R. I.; Geffner, H.; Hoffmann, J.; and Kautz, H. A., eds., *ICAPS*, 137–144. AAAI.

Wellman, M. P.; Larson, K.; Ford, M.; and Wurman, P. R. 1995. Path planning under time-dependent uncertainty. In *In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, 532–539. Morgan Kaufmann.

Zobrist, A. L. 1970. A new hashing method with applications for game playing. Technical report, Department of Computer Science, University of Wisconsin, Madison. Reprinted in *International Computer Chess Association Journal*, 13(2):169-173, 1990.

Zografos, K. G., and Androutsopoulos, K. N. 2008. Algorithms for itinerary planning in multimodal transportation networks. *IEEE Transactions on Intelligent Transportation Systems* 9(1):175–184.