Path Finding Strategies in Stochastic Networks [*]

Mohammad H. Keyhani, Mathias Schnee, and Karsten Weihe
Technische Universität Darmstadt, Algorithmik[†]

December 10, 2014

Abstract

We introduce a novel generic algorithmic problem in directed acyclic graphs, motivated by our train delay research. Roughly speaking, an arc is admissible or not subject to the value of a random variable at its tail node. The core problem is to precompute data such that a walk along admissible arcs will lead to one of the target nodes with a high probability. In the motivating application scenario, this means to meet an appointment with a high chance even if train connections are broken due to train delays.

We present an efficient dynamic-programming algorithm for the generic case. The algorithm allows us to maximize the probability of success or, alternatively, optimize other criteria subject to a guaranteed probability of success.

Moreover, we customize this algorithm to the application scenario. For this scenario, we present computational results based on real data from the national German railway company. The results demonstrate that our approach is superior to the natural approach, that is, to find a fast and convenient connection and to identify alternative routes for all tight train changes where the probability that the change breaks due to delays is not negligible.

1 Introduction

1.1 Generic problem

We consider a directed acyclic graph, where each node is associated with a discrete random variable of arbitrary kind. The random variables are not independent, but the marginal distribution of a node's random variable is a function of the marginal distributions of the random variables at the node's immediate predecessors. Each arc is *admissible* or *inadmissible*, depending on the value attained by its tail node's random variable. A path is *admissible* if all of its arcs are so; otherwise, it is *inadmissible*.

The underlying graph problem is the following: for a set of source nodes and a set of target nodes, determine an admissible path from some source node to some target node. Since the admissibility of an arc is a random variable, too, it is clearly impossible to determine the set of all admissible paths in advance.

Instead, the focus of this paper is on efficient strategies for traversing the graph from a source node towards the target nodes. A *strategy* may be viewed as a black box that delivers a particular piece of information for a given node and a given value of the node's random variable: in the positive case, the strategy returns an outgoing arc that is guaranteed to be admissible for this value of the random variable; in the negative case, the strategy reports failure. Following the strategy means to walk through the graph and always use the arc delivered by the strategy, until a target node is reached or the walk gets stuck.

[*]This work was supported by German Railways Deutsche Bahn AG (RIS).
[†]{keyhani,schnee,weihe}@cs.tu-darmstadt.de

An efficient strategy will certainly need a *set-up*, that is, a preprocessing phase, in which the outgoing arcs are computed and stored. From a technical viewpoint, this set-up is the core problem.

This generic problem has been motivated by a very specific application scenario, which we address in this paper as well. However, it is evidently of much broader practical value, because it may fit into routing and navigation scenarios of various types in public and private transport and in logistics. Another potential application scenario is planning and scheduling with alternative options at every milestone, where the next milestone is to be chosen depending on the random variable *project state*.

Overview This paper is organized as follows. In the rest of Section 1, we write about our contribution, and explain why related work is not transferable. In Section 2, we define the input for the set-up and the data to be computed in the set-up process. The set-up and the core algorithm are explained in Section 3. In Section 4, we customize the generic approach to a real-world application from train travel optimization. Our experiments and computational results on the basis of real-world data from *Deutsche Bahn*, the national German railway company, are presented in Section 5. Finally in Section 6, we conclude and present an outlook on our future work.

1.2 Our Contribution

Our contribution I: generic approach We introduce *probability of success* for each node in the graph (see Section 3). By that we mean the probability to reach one of the target nodes without getting stuck at any intermediate node – when following the strategy. For each node, a strategy has a certain *probability of success.*

Based on that notion, we present an algorithm for set-up such that the resulting strategy indeed maximizes the probabilities of success for all source nodes simultaneously. Alternatively, our algorithm is also able to optimize other criteria subject to a guaranteed probability of success. In each step of the walk, only a single look up is needed to identify the next outgoing arc to take. The complexity of our set-up is approximately linear in the number of arcs and nodes (more details can be found in Section 3.2). This output immediately solves an important related problem: given a linear ordering on the source nodes, which expresses a preference, it is then trivial to choose the most preferable source node that guarantees success with 98% probability, say.

During the walk or even before commencing the walk, new information may come in, which may result in changes of the marginal distributions. It will be obvious from our presentation that a re-run of our proposed algorithm would still be the best one can do. In particular, if there is still a way to reach the target node with 98% probability, say, the procedure will find it.

Our contribution II: real-world application The motivating application scenario is this: we have a time-expanded graph [10] representing a railway timetable. For each *departure* and *arrival* event of each train, there is a node in the graph. The arcs model traveling, waiting at stations, and changing trains. Each arrival and departure event is supposed to take place at the time found in the timetable. However, trains are probabilistically delayed, and the actual time may be later or even earlier (earlier only for arrivals). Details can be found in Section 4.1.

Connections can break at stations with planned train changes due to delays: when a traveler arrives too late at the station so that there is no sufficient time for a train change, he/she misses the connecting train. Hence, the traveler needs a strategy to decide, on the basis of the current local delay situation, whether he/she should stay in the train or leave the train at the next station to catch another train, and which one he/she should take. Probability of success comes into play when the traveler has to meet a deadline at his/her destination station, for example, to get to a business meeting in time or to catch a flight. Clearly, the traveler would like to commence the journey as late as possible, provided the deadline is met with a 98% chance, say. So, the traveler's problem is to identify the latest departure whose probability of success is 98% or more. As mentioned in Section 1.1, the latest – and therefore most preferable – departure time is trivially identified once the probability of success is known for *every* departure node at the departure station.

From a traveler's point of view An information system based on our approach may be designed to be easy to use for the traveler. Instead of a normal itinerary, the traveler receives sort of an enriched itinerary, which includes instructions in case of delays and train breaks. During the travel, the traveler can observe the current delay time of the train in which he/she is currently sitting, and follow these instructions. Fortunately, travelers are typically notified early enough by the staff (or look up internet information) about breaks of train changes. In this case, a traveler can alternatively use the instructions for that broken train change. Even better, our approach could be integrated in an app that notifies the traveler about an upcoming change of the itinerary, re-runs the algorithm, and delivers up-to-date instructions.

1.3 Related Work

Our approach is unique because of the following two characteristics: we guarantee a probability of success to arrive before a deadline, and subject to this probability, we find the latest possible departure time.

In order to achieve the above goal, we systematically incorporate changes to alternative routes, if necessary due to delays. Also in contrast to other works, we can handle a large train network with a timetable resulting in a graph with millions of nodes and arcs, very weak periodicity, and an astronomically large number of delay scenarios.

There are approaches focused on other problems like least expected travel time or earliest expected arrival time [8, 15, 3]. Besides that, Hall assumes that there is no dependency between the travel times on different arcs in the graph [8] which is not realistic in a railway network where one train can delay another one. Analogously, Dibbelt et al. use a simplified model with the assumption that all departures are on time and all random arrival times are independent [3].

Approaches based on periodicity and designed for dense, high frequency networks [8, 2] are not transformable to a railway network including (but not limited to) long-distance trains. For instance, they assume that when a bus is missed, the next one will depart after a certain waiting time.

Approaches for networks without a timetable (for example road networks) [4, 13, 16, 17, 14, 5, 15, 11, 12] are evidently not transferable to our problem: in contrast to a railway network, they can use each arc at any time, and they do not deal with the issue that arcs can get inadmissible. Moreover, the model presented by Pretolani, Nielsen, and Andersen would require a hyperarc for each possible value for the random variable of each node [15, 11, 12]. This tremendously blows up the graph size in a timetable based network.

There are also approaches which provide *robust* paths, but can only handle a limited number of delay scenarios [7, 6]. Such approaches and their solutions are typically too conservative and too expensive, and the paths are significantly longer. Moreover, they cannot guarantee an arrival before a deadline with a certain probability.

Reliability of train connections is an issue since connections can break because of delays. We have already published a method to compute and rate the reliability of train connections based on delay distributions for nodes [9]. There, we showed that such a reliability rating is quite accurate by predicting the feasibility of train connections. Our approach in this work is based on this reliability rating method (see Section 4).

2 Preliminaries

Input Consider a directed acyclic graph $G = (V, A)$. Sets $\mathit{in}(v) \subset A$ and $\mathit{out}(v) \subset A$ denote the sets of incoming and outgoing arcs of node $v \in V$. Set T denotes a finite or infinite set. Random variable X_v denotes the discrete random variable with range T, for each node $v \in V$. There is a joint probability distribution with dimension $|V|$ over all random variables X_v with $v \in V$. The marginal distributions are denoted by $\phi_v(t)$, for $t \in T$ and $v \in V$. For $t \in T$, $\phi_v(t)$ denotes the probability that X_v attains value t. Each marginal distribution ϕ_v is computed depending on the marginal distributions of the immediate predecessors of v using a function f_v: let $(u_i, v) \in \mathit{in}(v)$ denote the incoming arcs of v for $i = 1, \dots, n$ and $n = |\mathit{in}(v)|$; one has $\phi_v(t) = f_v(\phi_{u_1}, \dots, \phi_{u_n}, t)$ for each $t \in T$.

Sets $V_S \subset V$ and $V_T \subset V$ denote two non-empty disjoint sets of source nodes and target nodes. Each arc (v, w) is *admissible* or *inadmissible* depending on the value of X_v. For every value of X_v it is known whether (v, w) is admissible or not.

Data to be computed in the set-up A strategy contains the following information for each node v and each value t for X_v:

- a data field $\alpha_{v,t}$ which contains
  - either an associated outgoing arc $(v, w) \in \mathit{out}(v)$ which is *admissible* when X_v has the value t
  - or $\perp$ if the strategy has no arc selected for node v and value t;
- and a probability of success $\rho_{v,t}$.

3 Computing the Optimal Strategy

For an input as specified in Section 2, the set-up computes an optimal strategy by calculating the probability of success $\rho_{v,t}$ and, simultaneously, selecting the best arc $\alpha_{v,t}$, at each node $v \in V$ and for each $t \in T$ (see Section 3.2). The strategy is set-up backwards through the graph, beginning with the target nodes and ending at the source nodes. More explanations about backward set-up can be found in Appendix A.

3.1 Propagating Probabilities via Dynamic Programming

Briefly, in order to compute the probability of success at each node in the graph, we have to propagate the marginal distributions of the target nodes' random variables, backwards through the graph. More explanations can be found in Appendix B.

How to propagate probabilities In the following, we explain our dynamic programming approach to propagate probabilities, backwards through the graph. For an arbitrary node $w \in V$ with predecessor $v \in V$, the propagation from w to v is performed by this formula:

$$\psi_{v,a}(t) = \sum_{b \in T_w} P(X_v = a \mid X_w = b) \cdot \psi_{w,b}(t) \tag{1}$$

for each $a, t \in T$ and $T_w = \{t \in T \mid \phi_w(t) > 0\}$. For each target node $z \in V_T$, one has $\psi_{z,a}(t) = \phi_z(t)$ for $t = a$; and $\psi_{z,a}(t) = 0$ for $t \neq a$. Here, $\psi_{v,a}(t)$ denotes the probability propagated from the target nodes' random variables to node v. This is the probability that (I) X_v attains the value a, (II) there is a path to some target node exclusively over admissible arcs, (III) and the random variable of that target node attains value t. More details about the formula can be found in Appendix C.

The conditional probability in Equation 1 is calculated as follows:

$$P(X_v = a \mid X_w = b) = f_w(\phi_{u_1}, \ldots, \phi_v^a, \ldots, \phi_{u_n}, b)/\phi_w(b)$$

where $u_1, \ldots, u_n \in V$ are other predecessors of w. For v, the probability ϕ_v^a is defined as follows: $\phi_v^a(t) = \phi_v(t)$ for $t = a$; and $\phi_v^a(t) = 0$ for $t \neq a$. We need ϕ_v^a since we want to determine how the probability $\phi_v(a)$ is spread over the values in T_w for X_w. Therefore, function f_w has to compute the probabilities depending on ϕ_v^a instead of ϕ_v. The above equations require that f_w computes its result depending on ϕ_v^a such that the following condition is satisfied: $\sum_{b \in T} f_w(\phi_{u_1}, \ldots, \phi_v^a, \ldots, \phi_{u_n}, b) \leq \phi_v^a(a)$. If X_w does not depend on X_v, the probability calculated by $f_w(t)$ has to be multiplied with $\phi_v^a(t)$, for each $t \in T$. This is because we want to compute the common probability that X_v attains value a and X_w attains value b. An example for a f-function can be found in Appendix C.

Computing probability of success For node v and value a for X_v, the probability of success is computed as follows: $\rho_{v,a} = \sum_{t \in T} \psi_{v,a}(t)$. The probability of success for node v, regardless of the value for X_v, equals $\sum_{a \in T} \rho_{v,a}$.

3.2 The Set-Up: Core Algorithm

This section explains the core algorithm of the set-up which processes nodes, and propagates the probabilities over arcs, respectively. At the end of this procedure, the optimal strategy is ready.

How to process nodes For each node v and each value t for X_v, the *best* arc has to be selected and the probability of success has to be computed. At the beginning of the set-up, $\alpha_{v,a}$ is set to $\perp$ and $\rho_{v,a}$ is set to 0, for each $v \in V$ and $a \in T$. To set-up the strategy, all target nodes V_T are inserted into the queue. While there are nodes in the queue, the following iterations are performed:

1. Extract next node v from the queue. For each $a \in T$ where $\phi_v(a) > 0$ perform steps 2-4.
2. Find all outgoing arcs of v which are admissible when X_v attains value a. If there is no admissible arc, set $\alpha_{v,a} := \perp$ and $\rho_{v,a} := 0$, break the iteration, and continue with next value in T.
3. Propagate the probabilities ψ for each candidate: for each found arc (v, w) and for each $t \in T$, calculate the probability $\psi_{v,a}^w(t)$ *assuming* (v, w) is the arc selected for v and value a. (Note: we index ψ with w since (v, w) is considered as a *candidate* and is not selected for value a, definitively).
4. Compare the values $\psi_{v,a}^w(t)$ of the candidates to each other, and *select* the *best* arc (see below): let (v, z) define the best arc found; set $\psi_{v,a}(t) := \psi_{v,a}^z(t)$, set $\alpha_{v,a} := (v, z)$, and set $\rho_{v,a} := \sum_{t \in T} \psi_{v,a}(t)$.
5. If $v \notin V_S$, insert all immediate predecessors of v into the queue.

Selecting the best arc Here, we are free to choose a comparison method which is most qualified for our application. If the probability of success has to be maximized and nothing else, the best arc is the arc which maximizes $\rho_{v,a}$. In case that some target nodes are more interesting than others, or some values for the random variables of the target nodes are more interesting, the propagated probabilities $\psi_{v,a}^w(t)$ can be compared. In our evaluation in Section 5, we optimize the departure time, the duration of train connections, and the number of train changes subject to a guaranteed probability of success. Note that for our comparisons we can perform arbitrary operations on the propagated probabilities, and we are not limited to a comparison of expected arrival times, for instance.

If no admissible arc can be found for $\alpha_{v,a}$, the walk through the graph would get stuck at node v and at time a. Consequently, the probability of success at v decreases. This is the reason why probability of success can be smaller than 1.

Complexity Our approach is linear in the number of arcs and nodes. In addition, for each arc, there is the effort to propagate the probabilities to its tail node. Fortunately, the support of the computed distributions is limited in practice. In our railway scenario, we limit the support of the marginal distributions to 1000 entries, and we ignore probabilities smaller than 10^{-5}. Consequently, our approach still remains linear in the number of arcs and nodes. It may be worth noting that a large number of irrelevant nodes may be excluded from the search a priori. This modification reduces the practical run time significantly.

4 Real-World Application: Railway Networks

4.1 Finding Reliable Train Connections

We consider queries with a source and a target station, a latest arrival time t_q, and a threshold p_q for the probability of success. Using our approach, we compute for travelers strategies which guarantee an arrival at the target station not later than t_q and with a probability of at least p_q. Moreover, we find the latest possible departure time where the required probability of success is still satisfied. For this, we customize our generic approach to the railway application, and define the ingredients of the algorithm, respectively.

We have a time-expanded [10] graph $G = (V, E)$ representing the timetable. For each *departure* and *arrival* event of each train, there is a node in the graph. There are *train*, *stay-in-train*, and *train-change* edges. Each arrival and departure event is supposed to take place at the time found in the timetable. However, the actual time may be later – because of delays – or even earlier (earlier only for arrivals). In our model, for each departure and arrival event, we have a delay distribution which has for each possible point in time a probability that the train actually departs / arrives at this time. Train-change edges – connecting an arrival node of a train to a departure node of another train – can get inadmissible: this happens when the tail node of the edge is delayed so that the required change time is not satisfied (we also consider waiting time policies).

Inspired by Berger et al. [1], we have developed a method to compute delay distributions for train connections and to rate the *reliability* of the connections [9]. By reliability of a connection we mean the probability of arriving at the target without any broken train changes. The formula introduced there are used in this work as f-functions when propagating the probabilities.

4.2 Customizing Our Approach (Strategy) to Railway Networks

Our generic approach from Section 3 may be customized to the application described in Section 4.1 as follows:

- Set V_S contains all *relevant* (see below) departure nodes at the source station; set V_T contains all *relevant* arrival nodes at the target station;
- Set T contains all existing times in the timetable (one minute granularity); random variable X_v models the actual time of event/node v; function ϕ_v is the delay distribution of node v; and the definition of function f_v depends on the type of the node (see Appendix C). We refer to [9] for more details about the calculation of the delay distributions.

- Train-change edges get inadmissible when the arrival node has a delay such that the remaining time is not sufficient for a change into the departing train.

W.r.t. the query, each arrival node $v \in V$ at the target station is relevant for which one has: the p_q-quantile of ϕ_v is in time interval $[t_q - \delta, t_q]$. We introduce a left bound $t_q - \delta$ for the arrival time, since an arrival much earlier than t_q is not desired by travelers. In order to accelerate the strategy set-up, we only consider departure events within a certain time interval. This time interval is sufficiently generous and is determined depending on the arrival time interval and the expected connection length for the query. For a further acceleration, as previously mentioned, we restrict the search area to all existing paths from V_S to V_T. We determine these paths prior to the set-up.

For the described setting, we calculate the optimal strategy with the approach presented in Section 3. When comparing the outgoing arcs to select the best one, we optimize travel duration and number of train changes, subject to the guaranteed probability of success, which is defined in the query by p_q. The latest departure time is identified by selecting the departure node at the source station with the latest departure time and a probability of success not lower than p_q.

5 Evaluation

5.1 Baseline: Extending a Connection by Alternatives

In order to evaluate our strategy approach presented in Sections 3 and 4.2, we developed a baseline approach to find *connections extended by alternatives.* The algorithm to construct such an extended connection is the following:

1. Search for an attractive connection from start to target, and determine its reliability using our reliability rating method for single connections [9].

2. Detect the next train change in the connection with a reliability lower than 1. Beginning at that stop, search for a new alternative connection to the target. Then, recalculate the reliability of the train change and the whole connection regarding the new alternative.

3. Repeat step 2 until the required threshold (p_q) for the connection reliability is obtained or there is no other alternative in the graph.

The connections and the alternatives are found using our Multi-Objective Traffic Information System MOTIS [10]. Such an extended connection can be seen as a strategy, and its probability of success equals the calculated reliability rating.

This approach models the behavior of the users of train information systems who try to find highly reliable train connections. Thus far, this is even far better than anything the user can do on his/her own.

5.2 Dataset and Test Queries

We evaluated our approach to compute strategies in the railway network presented in Section 4. We compared solutions found using our strategy approach (as described in Section 4.2) to solutions found by the extended connections approach (as described in Section 5.1). Our computations were carried out on a desktop PC with Intel Xeon CPU E3-1270v2 3.50GHz and 32 GB of RAM.

Using real timetable data from German Railways, we prepared a graph for a two-day period[1]: 5-6 August 2014 with 2.2M event nodes.

We considered 5,200 different station pairs – extracted from real customer queries – as source and target stations. For each station pair, we created queries with 9 different

[1]A two-day period is needed to cover long running trains and overnight connections.

test set	total #	no ExtConn	Strategy earlier	Strategy no diff	Strategy later	later departure
all	43,890	9.95%	0.00%	70.37%	19.68%	45 min
no direct	34,762	12.17%	0.00%	65.64%	22.19%	47 min
conn. rel. < 90	15,013	18.38%	0.00%	51.42%	30.20%	49 min

Table 1: Computational study: comparison of Strategy vs. ExtConn. Column *no ExtConn* shows the percentage of queries for which the ExtConn approach could not find any solutions satisfying the deadline. The columns *earlier*, *no diff*, and *later* present the percentage of the queries for which Strategy found a solution which had in comparison to the ExtConn approach an earlier, nearly the same (delta $<$ 5 min), or a later departure time. For solutions of Strategy with a later departure time, the last column shows how much later (in average) the departure time was in comparison to the solution of ExtConn. The first two rows correspond to the evaluation of all queries in the test set. The next two rows show the evaluation of all queries for which there were no direct connection from source to target. The last two rows show the evaluation of all queries for which the connection found by the standard search algorithm of MOTIS had a reliability lower than 90%.

deadlines for the arrival time: $t_q \in \{$ 13:00, 16:00, 18:00, 20:00, 22:00, 01:00, 06:00, 08:00, 10:00 $\}$. The first five deadlines were on the first day and the last four deadlines were on the second day of the two-days period. We set δ to 100 minutes and required a probability of success p_q of 98%. W.r.t. to this setting, we created a set of 46,800 queries in total.

5.3 Results

As described in Section 4.2, for each query we computed a strategy. Each source node corresponds to the entering point in a connection from source to target with reasonable alternatives and a certain probability of success. For this evaluation, we only considered the connection beginning at the latest source node with a probability of success not lower than 98%. We denote this approach by *Strategy*.

Moreover, for each query, we found a set of extended connections as described in Section 5.1. We selected the extended connection with the latest departure time at the source station and a probability of success of at least 98%. We denote this approach by *ExtConn*.

For 4.98% of the queries, there did not exist any connection in the timetable for the considered arrival time interval. For 1.13% of the rest of the queries, there did not exist any solution to arrive before the deadline and satisfy the 98% threshold. Additionally, we sorted out 0.18% of the remaining queries, since for them a comparison between Strategy and ExtConn was not fair: ExtConn found solutions which were better than the solutions of Strategy but could not be found by Strategy due to restrictions in our implementation of the Strategy approach. In our implementation of the Strategy approach, we forbid arrivals earlier than the queried arrival interval, as explained in Section 4.2. Furthermore, we do not allow cycles through the source station. Last, when comparing arcs pairwise, as described in Section 3.2, in the case that the difference between the probabilities of success of the two compared arcs is less than 0.0001, the arc with the smaller number of train changes dominates the another one. Compared to ExtConn, this adjustment in our implementation results in finding a worse solution for just 1 query in the evaluated query set, but helps to find more comfortable train connections for many other queries. The mentioned three restrictions are not due to the Strategy approch but to our implementation. Therefore, we did not consider them in our evaluation. In Table 1, we compare Strategy with ExtConn by analyzing the remaining 43,890 queries.

approach	duration (avg)	train changes (avg)	run time (avg)
Conn	231 min	1.55	0.23 s
ExtConn	249 min	1.40	1.45 s
Strategy	246 min	1.46	20.73 s

Table 2: Travel duration, number of train changes, and run time in average.

Discussion For many queries, there is already a direct connection or a highly reliable connection, which can be found using the standard connection search algorithm of MOTIS (a multi-criteria version of the Dijkstra's algorithm without reliability optimization). In many of these cases, the Strategy and the ExtConn approach can not obtain better results. In order to demonstrate the effectiveness of the Strategy approach, we separately evaluated queries for which there is no direct connection in the timetable but a connection with at least one train change (rows 3 and 4 in Table 1) Moreover, we analyzed queries where the connection found by the standard search algorithm of MOTIS had a reliability lower than 90% (last two rows in Table 1).

In Table 1, the columns *no ExtConn* and *Strategy later* present the advantage of the Strategy approach. Considering queries for which at least one train change is necessary, ExtConn found for 12.17% of the queries no extended connection arriving before the deadline with a probability of at least 98%. For 22.19% of the queries, Strategy was able to find solutions that depart in average 47 minutes later than those found by ExtConn. Considering queries for which MOTIS found a connection with a reliability lower than 90%, the Strategy approach obtains a better result for 48.58% of the queries.

The ExtConn approach is tied to the connections delivered by the train information system. There is no guarantee that these connections have good or even any alternatives. In contrast, the Strategy approach searches specifically for solutions which satisfy the probability of success and maximize the departure time. Moreover, ExtConn searches only for alternatives which begin at stops with train changes. On the contrary, at each station and depending on the arrival time, Strategy can choose between the options "to stay in the same train" and "to change into another train". Nevertheless, as already mentioned, ExtConn is even far better than anything the user can do on his/her own.

Table 2 gives the average run time per query, and the average travel duration and the average number of train changes of the solutions found by Strategy, ExtConn, and the standard search algorithm of MOTIS. In practice, a user needs many queries to find a reliable connection with good alternatives using the ExtConn approach. In fact, this is very time consuming, even if the run time of ExtConn seems to be better than the run time of Strategy in our automatized experiments.

The values in Table 2 show that the Strategy approach has found reasonable solutions with durations and number of train changes similar to the connections found by MOTIS.

6 Conclusion and Outlook

We introduced a novel generic algorithmic problem in directed acyclic graphs, motivated by our train delay research. Moreover, we presented an algorithm to set-up strategies that selects the best arc for each node and each value for its random variable and computes the probability of success via dynamic-programming. To our knowledge, we are the first to optimize criteria such as late departure time and travel duration subject to a guaranteed probability of success in a stochastic network. We customized our algorithm to the application scenario: finding reliable connections in a railway network.

In order to evaluate the quality of the strategies, we developed another approach modeling user behavior as a baseline. We presented computational results based on real data from the national German railway company. Our study showed that our approach is able to find reasonable train connections satisfying the queried probability of success.

For up to 48.58% of the analyzed queries, we were able to find better results than the baseline approach.

Outlook As mentioned in the introduction, the approach we presented is evidently of much broader practical value, especially in the fields of transportation, logistics, and scheduling. However, this is left to future research.

We have commenced investigating speed-up techniques like parallelization to improve the run time of our approach.

In timetable information, we plan to support multi modal traffic including local public transportation and individual traffic.

References

[1] Annabell Berger, Andreas Gebhardt, Matthias Müller-Hannemann, and Martin Ostrowski. Stochastic delay prediction in large train networks. In *ATMOS*, pages 100–111, 2011.

[2] Kateřina Böhmová, Matúš Mihalák, Tobias Pröger, Rastislav Šrámek, and Peter Widmayer. Robust routing in urban public transportation: How to find reliable journeys based on past observations. In *ATMOS*, volume 33 of *OASIcs*, pages 27–41, 2013.

[3] Julian Dibbelt, Thomas Pajor, Ben Strasser, and Dorothea Wagner. Intriguingly simple and fast transit routing. In *12th International SEA*, pages 43–54, 2013.

[4] Y. Fan and Y. Nie. Optimal routing for maximizing the travel time reliability. *Networks and Spatial Economics*, 6(3-4):333–344, sep 2006.

[5] Liping Fu and L. R. Rilett. Expected shortest paths in dynamic and stochastic traffic networks. *Transportation Research Part B: Methodological*, 32(7):499–516, 1998.

[6] Marc Goerigk, Sacha Heße, Matthias Müller-Hannemann, Marie Schmidt, and Anita Schöbel. Recoverable Robust Timetable Information. In *ATMOS*, volume 33 of *OASIcs*, pages 1–14, 2013.

[7] Marc Goerigk, Martin Knoth, Matthias Müller-Hannemann, Marie Schmidt, and Anita Schöbel. The price of robustness in timetable information. In *ATMOS*, volume 20 of *OASIcs*, pages 76–87, 2011.

[8] Randolph W. Hall. The Fastest Path through a Network with Random Time-Dependent Travel Times. *Transportation Science*, 20(3):182–188, August 1986.

[9] Mohammad H. Keyhani, Mathias Schnee, Karsten Weihe, and Hans-Peter Zorn. Reliability and delay distributions of train connections. In *ATMOS*, volume 25 of *OASIcs*, pages 35–46, 2012.

[10] Matthias Müller-Hannemann and Mathias Schnee. Finding all attractive train connections by multi-criteria pareto search. In *ATMOS*, pages 246–263, 2004.

[11] L.R. Nielsen, D. Pretolani, and K.A. Andersen. k shortest paths in stochastic time-dependent networks. Technical Report WP-L-2004-05, Department of Accounting, Finance and Logistics, Aarhus School of Business, 2004.

[12] L.R. Nielsen, D. Pretolani, and K.A. Andersen. Bicriterion a priori route choice in stochastic time-dependent networks. Technical Report WP-L-2006-10, Department of Business Studies, Aarhus School of Business, 2006.

[13] Evdokia Nikolova, Matthew Brand, and David R. Karger. Optimal route planning under uncertainty. In *ICAPS*, pages 131–141, 2006.

[14] Yiyong Pan, Lu Sun, and Minli Ge. Finding reliable shortest path in stochastic time-dependent network. *Procedia - Social and Behavioral Sciences*, 96(0):451 – 460, 2013. (CICTP2013).

[15] Daniele Pretolani. A directed hypergraph model for random time dependent shortest paths. *European Journal of Operational Research*, 123:2000, 1998.

[16] S. Samaranayake, S. Blandin, and A. Bayen. A tractable class of algorithms for reliable routing in stochastic networks. *Procedia - Social and Behavioral Sciences*, 17(0):341 – 363, 2011. 19th ISTTT.

[17] Samitha Samaranayake, Sebastien Blandin, and Alexandre M. Bayen. Speedup techniques for the stochastic on-time arrival problem. In *ATMOS*, volume 25 of *OASIcs*, pages 83–96, 2012.

A Why Set-Up from Target Nodes to Source Nodes?

To decide which outgoing arc has to be selected, for each possible outgoing arc at a node, we have to know which probability of success can be obtained when selecting the arc. For each possible outgoing arc, we have to know which target nodes can be reached when selecting the arc and which probabilities the values for the random variables of theses nodes would have. Therefore, we have to know the strategy to walk from each considered successor node to the target.

Thus, we set-up the strategy beginning with the target nodes backwards through the graph, until the source nodes are reached. This can be done using a topological ordering of the nodes in the graph. The probabilities of the random variables of the target nodes are also propagated backwards through the graph by our dynamic-programming approach. This propagation allows us to compute the probability of success and to select the best outgoing arc, for each node and each value for its random variable.

B Why Propagate Probabilities?

In the following, we explain why a *propagation* is necessary in order to get the probability of success. Consider a path over nodes $u_0, \dots, u_n \in V$ and arcs $(u_i, u_{i+1}) \in A$ with $i \in 1 \dots n-1$, where each of these arcs is selected by the strategy: one has $\alpha_{u_i,a} = (u_i, u_{i+1})$ for at least one $a \in T$. To set-up the strategy, for each $t_1, t_2 \in T$, we need to compute the following probability: the probability that X_{u_0} attains value t_1, X_{u_n} attains value t_2, and all arcs (u_i, u_{i+1}) are admissible. This probability depends on the random variables of nodes $u_0, \dots, u_n$. Its computation requires a propagation of probabilities through the selected arcs w.r.t. the mentioned dependencies.

Since we set-up the strategy from target to source, we also propagate the probabilities *backwards* through the graph. After adding each new arc to the strategy during the set-up phase, the backwards propagation via dynamic programming allows a reuse of partial results from previous iterations of the set-up. So, at each step, the propagation is performed only for the new arc. In contrast to the backwards propagation, after adding each new arc to the strategy, a forwards propagation approach would require a complete repropagation through all arcs which depend on the new arc and have previously been selected by the set-up. This would entail a considerable effort, since at each iteration we would redundantly recalculate distributions for a large number of nodes. So, we decided to propagate the probabilities backwards.

In the above example, let u_n be a target node. For each $t \in T$, we propagate the probabilities $\phi_{u_n}(t)$ backwards through the path $u_0, \dots, u_n$. These propagated probabilities contribute to the probability of success ρ_{u_0,t_1} at node u_0.

C How to Propagate Probabilities (Details)

The propagated probability $\psi_{v,a}(t)$ can also be defined by introducing a new random variable:

- for $a \in T$ and $v \in V$, we define random variable $Z_{v,a}$. When the value of each random variable is revealed, if X_v has attained value a, and there is a path from v to some target node $z \in V_T$ exclusively over admissible arcs, $Z_{v,a}$ attains the actual value of X_z;

- for $v \in V$ and $a, t \in T$, function $\psi_{v,a}(t)$ denotes the probability that $Z_{v,a}$ attains value t. We denote $\psi_{v,a}(t)$ as the propagated probability of node v for values a and t.

Correctness The multiplication of the both introduced terms in Equation 1 is allowed, because they are independent. The probability $\psi_{w,b}(t)$ doesn't depend on $P(X_v = a \mid X_w = b)$. The first term only propagates the probabilities from w to v. The division in the first term makes it independent from the second term.

An example for function f In our paper [9], we introduced various formula f to compute the delay distributions. Here, we show an example. Consider a train change from train A to train B at station S. At station S, let w denote the departure node of train B, u denote the arrival node of train B, and v denote the arrival node of train A. The minimum standing time of train B at station S is denoted by ε, and the minimum time required for a train change from A to B is denoted by η. The probability that train B departs on time at time t, and a train change from A to B is possible is calculated as follows:

$$f_w(t) = P\big(X_w = t\big) = \ P\big(X_u \leq t - \varepsilon\big) \cdot P\big(X_v \leq t - \eta\big) \cdot P_{no\,Waiting}$$

The term $P_{no\,Waiting}$ denotes the probability that train B does not have to wait longer for any other train at station S. More details can be found in [9].