Agent-Based Multimodal Transport Planning in Dynamic Environments

Christoph Greulich[1], Stefan Edelkamp[2], and Max Gath[2]

[1] International Graduate School for Dynamics in Logistics
[2] Institute for Artificial Intelligence
University of Bremen, Germany
{greulich,edelkamp,mgath}@tzi.de

Abstract. The development and maintenance of traffic concepts in urban districts is expensive and leads to high investments for altering transport infrastructures or for the acquisition of new resources. We present an agent-based approach for modeling, simulation, evaluation, and optimization of public transport systems by introducing a dynamic microscopic model. Actors of varying stakeholders are represented by intelligent agents. While describing the inter-agent communication and their individual behaviors, the focus is on the implementation of information systems for traveler agents as well as on the matching between open source geographic information systems, and standardized transport schedules provided by the Association of German Transport Companies. The performance, efficiency, and limitations of the system are evaluated within the public transport infrastructure of Bremen. We discuss the effects of passengers' behaviors to the entire transport network and investigate the system's flexibility as well as consequences of incidents in travel plans.

1 Introduction

Public transport networks (PTN) were introduced to urban regions in the 19th century [6] and underwent radical changes throughout history. The development of combustion engines and the discovery of electricity had huge influence on the means of public transport. Optimization of such PTNs grew into a diversified research area which ranges from developing optimization processes [14] to analysis of passenger behavior [2, 4, 7].

Applying new traffic concepts or changes to a given PTN or the infrastructure of an urban region can be very expensive. Several authors approach the evaluation of those changes by agent-based simulation of PTNs [8,10,11]. A crucial part of the simulation-based approach is multimodal route planning which considers various means of public and individual transportation. Unfortunately, multimodal route planning using public transportation (e.g., buses, trams, metros) and individual transportation (e.g., cars, bikes, or foot) requires an accurate linkage of the street map to the one for public transportation. Therefore, previous approaches have been limited to a specific area, such as Belfort [11] or Bangalore [8].

Even though publicly available and community-edited maps often contain information on stops and travel routes for public transportation, in multimodal route planning most frequently the resources of the two are diverse, especially if exact time scheduling constraints are involved. Timetables for public transportation units are, e.g., extracted from public transport company (PTC) databases and geo-located to the maps.

While this problem is conceptually easy, real-word challenges for the mapping include varying wordings of stops, and missing or reduced line information, where an line is a sequence of stops. Moreover, given geo-coordinates may not distinguish between multiple locations of stops for diverse travel directions at crossings. To resolve various namings for the geo-location and matching problem, we applied a generalization of the Needleman-Wunsch [13] algorithm for computing the alignment of genome strings. The layered algorithm is implemented in a database query language. Similarity matrices are extracted from a low-level approximate string matching, while the top-level alignment compares every two different lines passing a certain stop.

Inferred timetables are additionally fed into a public transport information system providing route information for individual passenger agents. They take the information as a guideline to plan their travel. The information system supports several implementations for shortest route queries.

The bijection between public transport and street map data combined with time-tabling information on travel and waiting times of mobile vehicles enables realistic travel simulation. Since the considered scenario provides a natural mapping of traffic stakeholder to software agents, we apply our multiagent-based simulation framework PlaSMA for evaluation. This enables fine-grained modeling of individual traffic participants on real-world transport infrastructures.

2 Simulation Model

To reach their individual goals, the agent-based simulation model contains three types of agents that interact with each other as well as with the environment. The transport infrastructure of the simulation environment is modeled as a directed graph. Nodes represent traffic junctions, dead ends, or stops of the PTN, while edges represent different types of ways, e.g., roads, motorways, and trails.

The *PublicTransportCompany-Agent* (*PTC-Agent*) receives a timetable upon creation. This timetable contains all departures and arrivals of every line of its fleet. According to the given timetable, the *PTC-Agent* creates *Vehicle-Agents* having a specific timetable for the journey from the first to the last stop of the line. In addition, the *PTC-Agent* provides a public transport information system for answering transport requests of potential passengers. To reduce communication, transportation contracts will be concluded between *Traveler-Agents* and *PTC-Agents* directly. Upon creation, the *Vehicle-Agent* drives its tour according to the timetable. *Traveler-Agents* are loaded or unloaded at stops, depending on their destination and their contract status. Finally, the *Traveler-Agent* tries to determine the most satisfying traveling option between its current position and its destination. The goal of the *Traveler-Agent* is to find the route with the earliest arrival time. If the distance between the origin and destination is above a predefined threshold, the *Traveler-Agent* avoids walking and acquires alternatives from the *PTC-Agent*. It reacts to problems (e.g., missed connections or inadmissible delays) by replanning which implies acquiring new alternative routes to reach the destination. The simulation model has already been introduced in [8].

3 Matching Data Sets

Since the simulation model is independent from the environment, the infrastructure and the simulated PTN can be exchanged easily. The underlying PlaSMA simulation platform[1] supports the import of OpenStreetMap[2] (OSM) datasets. These datasets contain PTN-related information, such as the public transport lines as well as the positions and names of their stops. Due to the fact that data is added to OSM manually, it is not guaranteed that the geographic information system is complete and accurate. Therefore, recent changes to the infrastructure or the PTN may not been transfered and stops or lines may be missing or tagged with wrong coordinates. Unfortunately, OSM does not provide information about the timetable of each line.

On the other hand, data sets of PTCs essentially consist of timetables. The Association of German Transport Companies (VDV) introduced standardized data models for the storage of PTN-related information [16]. The standard allows but not necessarily requires the storage of geographical information for each stop. Therefore, the operating data of a PTC does not provide enough information to map the stops' IDs to their geographical position which is necessary for an accurate simulation model.

Consequently, the mapping of street map data and the PTCs own timetable data is an essential problem as illustrated in Fig. 1. Names and identifiers of various stops and lines in each data set may vary significantly. Even if stops have the same names stops may form a stop area and, therefore, share the same name, e.g., two stops on opposite sides of the street. As a result, a name-to-name matching approach is not applicable. A solution is to distinguish the stops of a stop area by identifying their associated lines.

The similarity of two lines is given by the similarity of their stops and their respective indices alike. Finding pairs of two lines (one from each data set) allows to determine pairs of stops. Thus, each stop position in the street map can be mapped to the stop identifier in the PTC's data set.

3.1 The Needleman-Wunsch Algorithm

The Needleman-Wunsch algorithm [13] has originally been designed to align two nucleotide or peptide sequences but is commonly used to solve string alignment

[1] http://plasma.informatik.uni-bremen.de
[2] http://www.openstreetmap.org

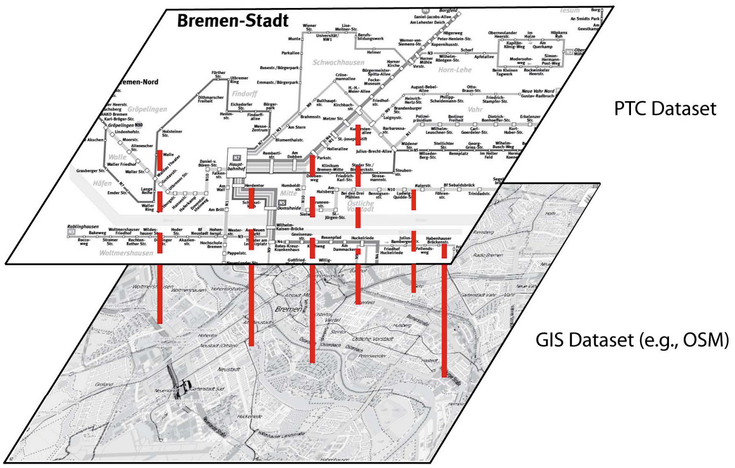


Fig. 1. Linkage of PTN information from two different data sets

problems. A similarity value $S(x, y)$ specificies the similarity score of x and y, where $x, y \in \Sigma$ and Σ is the given alphabet.

The Needleman-Wunsch-algorithm computes a matrix M with a size of $(|A|+1) \times (|B|+1)$ where A and B are two strings over Σ. The matrix contains a column for each element of a string $A = a_1 \ldots a_n$ and a row j for each element of a string $B = b_1 \ldots b_n$. Additionally, the first row and column is reserved for multiples of the gap penalty p. Although the algorithm relies on matrix recurrences it can be solved in time $O(nm)$ by dynamic programming [9]. For $M_{i,0} = p \cdot i, i \in \{1, \ldots, n\}$ and $M_{0,j} = p \cdot j, j \in \{1, \ldots, m\}$ we have

$$M_{i,j} = max \begin{pmatrix} M_{i-1,j-1} + S(a_i, b_j), \\ M_{i,j-1} + p, \\ M_{i+1,j} + p \end{pmatrix}, i, j \in \{1, \ldots, n\} \times \{1, \ldots, m\}$$

3.2 Stop and Line Matching

We apply the above algorithm to the given matching problem. Let $\mathbb{I}$ be a set of lines and $\mathcal{E}, \mathcal{F} \subset \mathbb{I}$. In order to align two lines $E \in \mathcal{E}$ and $F \in \mathcal{F}$ of two different data sets $\mathcal{E}$ and $\mathcal{F}$, the similarity matrix S is substituted by the function $s(e, f)$, which determines the similarity between two stops $e \in E$ and $f \in F$ by the comparison of their names. For solving the corresponding alignment problem to compute $s(e, f)$, we apply the trigram matching algorithm which counts the number of trigrams (sequences of 3 consecutive characters) that are included in both stop names to determine similarity [15].

Additionally, geographical information is taken into account, e.g., the distance between e and f. The value of the gap penalty p a threshold and depends on the implementation of the similarity function. The penalty score for creating a gap must be greater than the similarity score of two stops with an unlikely match.

For each line of the PTC data set, we determine the most similar line from the OSM data set. The resulting Needleman-Wunsch matrix of the best matching pair is saved. After the investigation of all line pairs, pairs of stops are computed by applying a backtracking algorithm to the Needleman-Wunsch-matrices of the line pairs. As a result, the node IDs of stops from the OSM data set are linked to the equivalent stop in the PTC's data set. Algorithms 1.1 to 1.3 show implementations of the necessary functions in pseudo code.

```
function FindMatchingLines($\mathcal{E}$,$\mathcal{F}$)
    $R \leftarrow \emptyset$
    for each $E \in \mathcal{E}$ do
        $P.Line \leftarrow E$
        $Similarity \leftarrow 0$
        for each $F \in \mathcal{F}$ do
            $M \leftarrow$ createNeedlemanWunschMatrix($E, F$)
            if $M_{|E|,|F|} > Similarity$ then
                $P.BestMatch \leftarrow F$
                $P.BestMatrix \leftarrow M$
                $Similarity \leftarrow M_{|E|,|F|}$
        $R \leftarrow R \cup \{P\}$
    return $R$
```

Algorithm 1.1. Determining best matching lines of two data sets $\mathcal{E}$ and $\mathcal{F}$

```
function createNeedlemanWunschMatrix($E$, $F$)
    $M$ is a matrix with the size of $(|E| + 1) \times (|F| + 1)$
    $p$ is the penalty for skipping an element
    for $i = 0$ to $|E|$ do
        $M_{i,0} = i \cdot p$
    for $j = 0$ to $|F|$ do
        $M_{0,j} = j \cdot p$
    for $i = 1$ to $|E|$ do
        for $j = 1$ to $|F|$ do
            $M_{i,j} \leftarrow max \begin{pmatrix} M_{i-1,j-1} + s(e_i, f_j) , \\ M_{i-1,j} + p, \\ M_{i,j-1} + p \end{pmatrix}$
    return $M$
```

Algorithm 1.2. The Needleman-Wunsch Algorithm

```
function CREATESTOPPAIRS($E$, $F$, $M$)
    $p$ is the penalty for skipping an element
    $i \leftarrow |E|$
    $j \leftarrow |F|$
    while $i > 0$ or $j > 0$ do
        if $M_{i,j} = M_{i-1,j-1} + s(e_i, f_j)$ then
            MEMORIZEPAIR($e_i$, $f_j$)
            $i \leftarrow i - 1$
            $j \leftarrow j - 1$
        else if $M_{i,j} = M_{i-1,j} + p$ then
            $i \leftarrow i - 1$
        else if $M_{i,j} = M_{i,j-1} + p$ then
            $j \leftarrow j - 1$
```

Algorithm 1.3. Backtracking for the Needleman-Wunsch Algorithm

4 Public Transport Information Systems

In order to satisfy transport requests of customers and to determine the route with the earliest arrival time, we implemented a public transport information system solving the *Earliest-Arrival-Problem* (EAP) [12] with stop-specific transfer times. In addition, the route which minimizes the number of transfers should be preferred if several routes arrive at the same time. Moreover, undesired lines have to be excluded. Next, we present three different approaches for solving the EAP. The search algorithms extend the well-known Dijkstra algorithm [5].

4.1 Time-Expanded Graph

We extend the *time-expanded graph* (TEG) approach [12]. Let S denote the set of stop areas and L denote the set of lines. A time-expanded graph includes a set of nodes N and a set of edges E. Each node $(l, s) \in (L \times S)$ represents the arrival or departure of an line $l \in L$ at a stop area $s \in S$. Edges connecting nodes represent transport routes of vehicles between stops as well as transfers of passengers at a certain stop area. The weights of edges determine the travel time. As a result, a shortest-path search is applied to satisfy a certain transport request from a start node to a goal node. To fulfill the requirements for passenger transport, the search is altered: edges which are served by undesired lines are excluded and edges that require passenger transfers are only considered if the current line does not reach the stop area within the same time. Consequently, these conditions ensure that routes with more transfers are avoided if the arriving time is equal to a route with less transfers.

4.2 Time-Dependent Graph

Within a *time-dependent graph* (TDG) [12] each stop area is represented by only one node. Two nodes are linked by exactly one edge if there is at least one

```
function FINDSHORTESTPATH($n_{source}$, $n_{sink}$, $t_0$, $I$, $X$)
    $M \leftarrow \{n_{source}\}$
    SETTIMEDISTANCE($n_{source}$, $t_0$)
    SETSUBEDGE($n_{source}$, null)
    while $M \neq \emptyset$ do
        $u \leftarrow$ GETNODEWITHMINDISTANCE(M)
        $M \leftarrow M \backslash \{u\}$
        if $u = n_{sink}$ then
            return CREATEPATH($u$)
        $e_u \leftarrow$ GETSUBEDGE($u$)
        for all $E \in$ GETEDGES($u$) do
            $v \leftarrow$ SINK($E$)
            for all $e \in E$ do
                $d_{departure} \leftarrow$ DEPARTURE($e$) - GETTIMEDISTANCE($u$)
                if  $((I = \emptyset \vee$ LINE$(e) \in I) \wedge$ LINE$(e) \notin X) \wedge$
                                  $((e_u =$ null $\wedge$ARRIVAL$(e) <$ GETTIMEDISTANCE$(v)) \vee$
                                  $(e_u \neq$ null $\wedge$ARRIVAL$(e) <$ GETTIMEDISTANCE$(v) \wedge$
                                        LINE$(e_u) \neq$ LINE$(e) \wedge$
                                        $d_{departure} \geq$ GETTRANSFERTIME$(u)) \vee$
                                  $(e_u \neq$ null $\wedge$ARRIVAL$(e) \leq$ GETTIMEDISTANCE$(v) \wedge$
                                        LINE$(e_u) =$ LINE$(e)))$  then
                    if $v = n_{source}$ then
                        SETSUBEDGE(SINK($E$), null)
                    else
                        SETSUBEDGE(SINK($E$), $e$)
                    SETTIMEDISTANCE(SINK($E$), $d$)
                    $M \leftarrow M \cup \{$SINK($E$)$\}$
    return new empty path
```

Algorithm 1.4. Implementation of the Shortest-Path Algorithm on a TDG

direct connection between each other. As a result, the search space is decreased significantly in contrast to a TEG. However, each edge contains several sub-edges with specific arrival and departure times as well as a dedicated line. The shortest-path search has to be adapted to the sub-edge graph structure. The pseudo code is shown in Algorithm 1.4.

Input parameters next to the start and goal node include the start time t_0, a set of allowed lines I, and a set of lines X which should be avoided. If I is an empty set, all lines which are not in X are considered. The shortest path search for a TDG considers only predecessor edges instead of predecessor nodes. The functions SETSUBEDGE(A, e) and GETSUBEDGE(A) save and return the best sub-edge e to the node A. The algorithm chooses the node with the shortest distance and iterates over all outgoing edges including their sub-edges. The procedure ARRIVAL(e) determines the estimated arrival time at the node that is reachable by e. In addition, the distance between nodes is computed by the travel time from node A to node B. In order to prefer routes with the earliest time of arrival, conditions for choosing the best edge are imposed.

4.3 Reduced Time-Expanded Graph

In a time-dependent graph (TDG), the graph is reduced and only the shortest-route with the earliest departure time from one node to another is considered. However, this may result in undesirable transfers between lines. Thus, we reduce the number of edges of the TEG without cutting relevant solutions. Let (v, u) denote an edge from node $v \in N$ to node $u \in N$, $s_i \in S$ the stop at node $i \in N$, and t_i the time of arrival or departure. While the classic TEG contains all edges (v, u) with $s_v = s_u \wedge t_v \leq t_u$, the reduced TEG includes only the edge (v, u) representing the next departure per line $l_u \in L$ at stop u:

$$s_v = s_u \wedge t_v = t_u \wedge (\neg \exists u' : u' \neq u \wedge s_v = s_{u'} \wedge t_v \leq t_{u'} \wedge t_{u'} < t_u \wedge l_{u'} = l_u).$$

5 Evaluation

The BSAG[3], the local PTC of Bremen (Germany), provided a set of operating data which meets the VDV-standard. The monday schedule of the PTC data set contains 116 lines. Lines may have a set of variants, e.g., for the opposite direction or for shorter/alternative routes. A lot of these variants are not accessible for passengers but merely for service maintenance and can be ignored for the purpose of simulation. Additionally, each line variation may have not one but several timetables depending on the current time of day. This results in 560 line timetables. More than 1,990 stops are served by these lines.

The PTC data set is matched against an OSM data set which contains 76 line variants and a variety of 768 stops. The OSM data set does not usually contain more than two variants of the same line (one for each direction). Unfortunately, several information in the OSM is outdated due to recent changes to infrastructure and PTN. Hence, the OSM data set contains stops that are not served anymore by any line in the PTC data set. Additionally, not all PTN information could be exported from the OSM data set. Several lines and stops are not tagged correctly or by an outdated tagging standard that is neither compatible with the new one, nor with the applied importer software.

However, 65 line variations from OSM could be successfully linked to lines from the PTC data set and 549 stops could be linked to their respective node ID. Hence, 314 of 560 line timetables can be simulated without any manual modeling. The compressed infrastructure itself is represented by a graph with 26,121 nodes and 59,950 edges.

Based on this data, a series of experiments was run to determine the capability and limitations of the simulation model and its implementation. In each experiment, we simulated a PTN with a 24 hours monday schedule. The start and destination locations of each traveler are chosen randomly. Therefore, the simulation results do not reflect real world utilization yet. In future work, varying population density and areas which are frequently visited have to be taken into account. All experiments are computed on a dedicated machine with 8 dual-core AMD processors and 64 GB of RAM.

[3] `http://www.bsag.de`

5.1 Ressource Limitations and Scalability

According to real world data, about 281,000 passengers are transported per day by the BSAG [1]. Therefore, nearly 2,000 passengers have to be created every ten minutes in order to approximate real world statistics. However, no more than about 7,000 active agents can be handled by the given machine due to a lack of memory. Another experiment with 200 passengers every ten minutes was aborted due to the same memory issues after about 90% of the simulation was already completed. Hence, a simulation of 100 passengers every ten minutes is manageable which corresponds to a scale of 1:20.

In the experiments, varying distributions (e.g. due to rush hours) have not been taken into account. It has to be considered that peak values may be much greater if a more accurate distribution is applied. Therefore, memory usage has to be reduced significantly. This may be achieved by reducing the number of agents without reducing the number of passengers.

5.2 Comparison of Information Systems

Both computing time and quality of results vary between the three different graph types due to the diverse size and complexity. Creating a complete TEG for the PTN timetable took 106.5 seconds. Furthermore, a reduced TEG was created within 10.8 seconds. The creation of a TDG took about 0.1 seconds.

The TDG always proved itself to be the fastest variant. The results of five randomly chosen transport requests are shown in Table 1. To generate more representative numbers, experiments on a scale of 1:100 have been run with each graph type. However, the TEG-experiment was aborted manually after about 36 hours and a simulation progress of less than 40%. In comparison, finishing the experiment took 14.3 hours with the reduced TEG and only 0.5 hours with the TDG. Each one of the 2,863 transport requests has been solved in 17.2 seconds with the reduced TEG and in 0.006 seconds with the TDG on average.

Table 1. Computing time of randomly chosen transport requests

Request (time 08:00)	TEG	red. TEG	TDG
Kriegerdenkmal → Hasenb. Jachthafen	146.184s	2.995s	0.076s
Hasenb. Jachthafen → Sebaldsbrück	243.908s	6.553s	0.105s
Groepelingen → Polizeipraesidium	182.154s	4.126s	0.088s
Waller Friedhof → Ricarda-Huch-Strasse	124.414s	2.307s	0.079s
Roland-Center → Julius-Brecht-Allee	62.868s	0.641s	0.064s
Average	151.905s	3.324s	0.082s

Even though the recuded TEG variant is significantly faster than the complete TEG, both variants produce identical results. Unfortunately, the solution quality varies between the TEG variants and the TDG. All variants solve the EAP correctly by computing a shortest path with the ideal arrival time. However, the

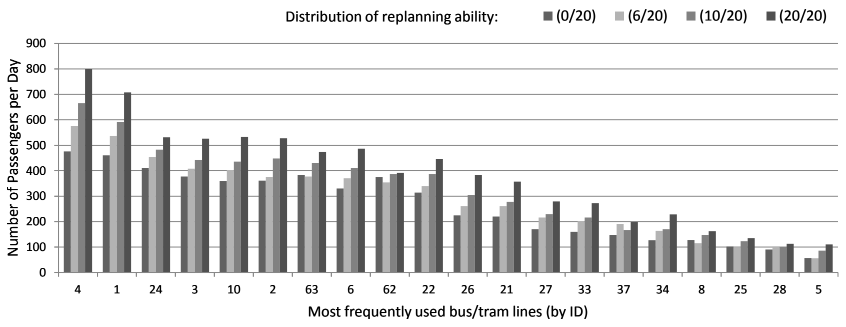
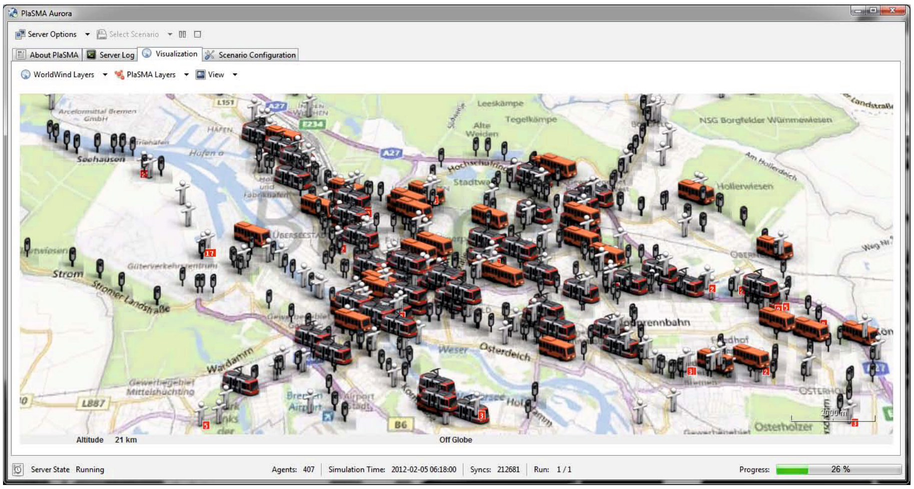


Fig. 2. Line utilization with diverse distribution of replanning ability

TDG is not capable of minimizing the number of transfers while searching for a shortest path. Consequently, passengers would not choose all of the computed routes based on a TDG because they contain unnecessary transfers a real passenger would try to avoid. From this point of view, the reduced TEG variant is the best choice for a simulation of passenger behavior, even though it is not the fastest variant.

5.3 Application: Effects of Passengers' Behaviors to the PTN

In 2013, 40% of Germans already own a smartphone [3]. Hence, two out of five Germans are able to revise their current traveling plans in case of unexpected disruptions by looking up alternative options. As a result, passengers can switch lines more often in case of delays and disruptions. This affects the utilization of the failed line and other lines alike. The goal of public transport providers



Server State Running

Agents: 407 | Simulation Time: 2012-02-05 06:18:00 | Syncs: 212681 | Run: 1 / 1

Progress: 26 %

Fig. 3. Screenshot of the running simulation software

is to anticipate the passengers' behaviors in predictable situations (e.g., road construction or public demonstrations) as well as in unexpected situations (e.g., in case of a fire or in natural disasters) to adapt timetables and schedules appropriately. The model enables strategy and scenario analysis.

Several performance indicators, such as the utilization of each line or departures and arrivals at each stop are measured by the simulation system. The share of replanning agents and the probability of line failure are given by simulation parameters. As an example, we set up a series of experiments with a high failure possibility of 33% for every vehicle and a diverse distribution of replanning ability (0%, 30%, 50%, 100%). Fig. 2 shows the utilization of the BSAG's frequently used lines according to that experiment. Though the chosen scenario setup is rather generic, it is also possible to simulate specific situations where a only certain lines fail. Therefore, the PTC can predict passenger behavior by applying the simulation model to a specific problem. Fig. 3 shows the running simulation software.

6 Conclusion and Outlook

In this paper we presented an adaptation of the Needleman-Wunsch Algorithm in order to match line information from VDV-compliant data sets to excerpts from the OSM. The approach allows the simulation model to be applied to any compatible combination of PTN and infrastructure. Furthermore, real world data sets of the PTN of Bremen (Germany) were used as an example to determine bottlenecks of a microscopic simulation of PTNs. One bottleneck is the available RAM due to the large number of more than 6,000 concurrently active agents. Another bottleneck is the graph-based approach to intermodal routing. We pinpointed by comparison that EAPs could be solved by complete TEG, reduced TEG, and TDG alike. However, while TDG is a lot faster than both TEG variants, results of the TDG are not always likely to resemble the routes a human would choose. On the other hand, the complete TEG is too slow to be applied to a simulation on larger PTN. The reduced variant of the TEG provides an acceptable tradeoff between calculation time and the quality of results.

In order to handle a more fine-grained scale of the simulation and larger PTNs, we are interested in increasing the efficiency of the information system (e.g., by adapting a recently developed K-shortest-paths search for TDGs [17] or by grouping passengers which have similar or identical interests and, therefore, can be represented by one single agent). Further research will additionally focus on the integration of traffic simulation within our framework as well as on modeling and simulation of unexpected changes to the infrastructure.

Acknowledgement. The presented research was partially funded by the German Research Foundation (DFG) within the Collaborative Research Centre 637 Autonomous Cooperating Logistic Processes (SFB 637) and the project Autonomous Courier and Express Services (HE 989/14-1) at the University of Bremen, Germany. We thank the BSAG for providing real-world timetable data.

References

1. Bremer Straßenbahn AG: Blickpunkte. Geschäftsbericht 2011 (2012)
2. Buehler, R.: Determinants of transport mode choice: a comparison of Germany and the USA. Journal of Transport Geography 19(4), 644–657 (2011)
3. Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V. (BITKOM): Auch Ältere steigen auf Smartphones um. 40 Prozent aller Deutschen haben ein Smartphone (2013)
4. Collins, C.M., Chambers, S.M.: Psychological and situational influences on commuter-transport-mode choice. Env. and Beh. 37(5), 640–661 (2005)
5. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271 (1959)
6. Divall, C., Schmucki, B.: Introduction: Technology (sub)urban development and the social construction of urban transport. In: Suburbanizing The Masses. Public Transport and Urban Development in Historical Perspective. Ashgate (2003)
7. de Donnea, F.: Consumer behaviour, transport mode choice and value of time: Some micro-economic models. Regional and Urban Economics 1(4), 55–382 (1972)
8. Greulich, C., Edelkamp, S., Gath, M., Warden, T., Humann, M., Herzog, O., Sitharam, T.: Enhanced shortest path computation for multiagent-based intermodal transport planning in dynamic environments. In: ICAART (2), pp. 324–329 (2013)
9. Gusfield, D.: Algorithms on Strings, Trees and Sequences. Cambridge University Press (1997)
10. Klügl, F., Rindsfüser, G.: Agent-based route (and mode) choice simulation in real-world networks. In: WI-IAT, vol. 2, pp. 22–29 (2011)
11. Meignan, D., Simonin, O., Koukam, A.: Multiagent approach for simulation and evaluation of urban bus networks. In: AAMAS, pp. 50–56 (2006)
12. Müller-Hannemann, M., Schulz, F., Wagner, D., Zaroliagis, C.: Timetable information: Models and algorithms. In: Geraets, F., Kroon, L.G., Schoebel, A., Wagner, D., Zaroliagis, C.D. (eds.) Railway Optimization 2004. LNCS, vol. 4359, pp. 67–90. Springer, Heidelberg (2007)
13. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology 48(3), 443–453 (1970)
14. van Nes, R., Hamerslag, R., Immers, B.H.: Design of public transport networks. Transportation Research Record 1202, 74–83 (1988)
15. Tardelli, A.O., Anção, M.S., Packer, A.L., Sigulem, D.: An implementation of the trigram phrase matching method for text similarity problems. In: Bos, L., Laxminarayan, S., Marsh, A. (eds.) Medical and Care Compunetics 1, 1st edn., vol. 103. IOS Press (2004)
16. Verband Deutscher Verkehrsunternehmen: ÖPNV-Datenmodell 5.0. "Schnittstellen-Initiative". VDV Standardschnittstelle. Liniennetz/Fahrplan. Version: 1.4 (2008)
17. Yang, Y., Wang, S., Hu, X., Li, J., Xu, B.: A modified k-shortest paths algorithm for solving the earliest arrival problem on the time-dependent model of transportation systems. In: International MultiConference of Engineers and Computer Scientists, pp. 1562–1567 (2012)