Enhanced Shortest Path Computation for Multiagent-based Intermodal Transport Planning in Dynamic Environments

Christoph Greulich[1], Stefan Edelkamp[1], Max Gath[1], Tobias Warden[1], Malte Humann[1], Otthein Herzog[1] and T. G. Sitharam[2]

[1]*Center for Computing and Communication Technologies, University of Bremen, Germany*

[2]*Center for Infrastructure, Sustainable Transportation and Urban Planning, Indian Institute of Science, Bangalore, India*

{*greulich,edelkamp,mgath,warden,humi,herzog*}*@tzi.de,sitharam@civil.iisc.ernet.in*

Keywords: Multiagent-based Simulation

Abstract: This paper addresses improved urban mobility using multiagent simulation. We provide a description of the agent model and the routing infrastructure as a step towards a rich model of the interactions that happen in intermodal transport planning tasks. The multiagent model is generic in the sense that different public and individual transport agents and transportation agencies can be added and parameterized on-the-fly. It integrates planning with execution. We show that a sequence of calls to Dijkstra's single-source shortest-paths algorithm is crucial for planning and provide an efficient memory-less implementation with radix heaps in order to make this application feasible with respect to scalability. As a case study, we implement a scenario for Bangalore (India), starting on a higher level of abstraction and drilling down to a running program.

1 INTRODUCTION

Under the umbrella term of *smart mobility*, the development of forward-looking traffic concepts for fast-growing metropolitan areas has attracted considerable interest of regional authorities and transport planners. In this context, optimized utilization of existing and, potentially, planned traffic infrastructure, using new mobility concepts has come in sharp focus.

Applying new traffic concepts to the real world can be very expensive, especially when the given infrastructure has to be altered or new ressources have to be acquired. Therefore, *multiagent-based simulation (MABS)* can be used to procure well-founded assessments of the impact of potential changes before actual deployment. Agent-based traffic simulation is an active research area (Chen and Cheng, 2010). Various simulators are available, e.g., MATSIM and SUMO.

Similar to Klügl and Rinsfüser (2011), we do not concentrate on traffic flow but on complex multimodal transportation tasks. However, they provide a rather abstract model that only contains dedicated route selection agents. Our model is also related to the one introduced by Meignan, Simonin, and Koukam (2006) but is far more dependent on inter-agent communication. We assume that information about possible travelling options can not be obtained directly but must be requested from the respective transportation service.

Our main contributions are:

- *push-button infrastructure map import* from the OpenStreetMap database yielding simulation results based on *real-world transportation data*.
- *dynamic creation of agents* in a multimodal route planning environment and scalable integration in a modern MABS system.
- a *generic model* for interleaved travel planning and plan execution which includes replanning on failure. While planning, the agents consider both public and individual transportation options.
- *enhanced shortest path planning* in optimal linear time with off-line space overhead of two pointers per node and no additional space allocated during the search.

Section 2 introduces MABS. Section 3 discusses the proposed simulation model with a focus on the implemented multiagent system. Section 4 provides a time and space optimal implementation of Dijksta's algorithm. Section 5 specifies the agent configuration used in our scenario and report on the feasibility assessment of our implementation with results on simulation efficiency on the traffic infrastructure of Bangalore.

2 AGENT-BASED SIMULATION

Any traffic simulation model can be classified by its level of detail. While macroscopic models only consider aggregated information, microscopic models are based on detailed modeling of individual entities (Hogendorn and Bovy, 2001).

Multiagent systems provide the capability to implement microscopic models. Even though the definition of an agent is widely discussed, the most important characteristic of an agent is its autonomy (Macal and North, 2010): Every agent is a unique entity and gains information by sensing its environment or by social interaction with other agents. In addition, an agent acts and reacts upon its own decisions which are based on the individual goals of the agent and its knowledge of the environment.

A MABS system can combine distributed discrete-event or time-stepped simulation with decision-making encapsulated in agents as separate and concurrent logical processes. In classical simulation systems, the involved logical processes as well as interaction links have to be known in advance and must not change during simulation. This is not the case in MABS systems, as each agent may interact with all other agents. Agents may join or leave simulation during execution, e.g., depending on a stochastic simulation model.

We use our event-driven MABS system which has been designed to solve and evaluate scenarios in the logistics domain. The system is based on the Java Agent DEvelopment framework (Bellifemine, Caire, and Greenwood, 2007) and adds the functionality of a discrete time simulation as well as conservative synchronization with time model adequacy, causality and reproducibility (Gehrke, Schuldt, and Werner, 2008). In general, *our system architecture* has the following components: World model, physical objects, infrastructure and agents including behaviour definitions.

The (physical) *simulation world model* of a scenario can be modeled as a graph so that the infrastructure can be mapped accordingly. Graph nodes represent, e.g., traffic junctions. Graph edges represent roads, rails, waterways, etc. In order to model real transport processes, we extended our system to simulate scenarios within real traffic infrastructures that are imported from OpenStreetMap databases. Agents act either as artificial autonomous decision makers on behalf of their associated entity within this model or as background services, e.g., for creating additional agents or providing information.

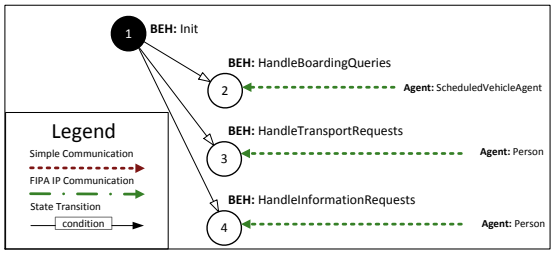

Figure 1: *Scheduled Transportation Company Agent*.

3 SIMULATION MODEL

We distinguish between simple behaviors (like *Init* or *HandleInformationRequests* or *Driving*) and complex behaviors (like *TransportPassengers*) that itself consist of an arrangement of behavioral states (see Fig. 1 - 3). Complex behaviors are implemented in terms of Finite-State-Machine (FSM) behaviors. FSM behaviors are labeled transition systems with a starting state (indicated with an incoming arc), one or more terminal states (encircled node), and several conditional state transitions. The behaviors also include communication arcs (dashed) that show whether or not a behavior communicates with another agent. Additionally, we show which agent poses shortest path queries (SPS).

The *Scheduled Transportation Company Agent* is an implementation of a transportation company that provides an information infrastructure for answering initial transport queries (*HandleInformationRequests*), sets-up and maintains a list of persons to be picked up at stops for a specific vehicle (*HandleBoardingQueries*), and receives and answers transport requests of persons at stops (*HandleTransportRequest*).

The corresponding agent model is shown in Fig. 1. After registration with the system wide directory facilitator and acquisition of time-table information from a configuration file, the agent is responsible for invoking time-dependent initialization of *Scheduled Vehicle Agent* instances that represent the means of transport. The sub-behaviors are spawned, and run in parallel.

On its initialization, a *Scheduled Vehicle Agent* receives a schedule from the agency that it has to follow on a daily basis. The time table information contains the arrival and departing time of each stop in the tour. If a vehicle is running late it usually tries to catch up with its schedule, reducing waiting times. The Scheduled Vehicle Agent's model is shown in Fig. 2. The FSM behavior for scheduled driving mainly implements a cycle of boarding, debarking and moving. Furthermore, it uses an interaction protocol to communicate with the transportation company agent to receive new instructions.

In addition, we implemented a *Autonomous Transportation Vehicle Agent* which basically combines the

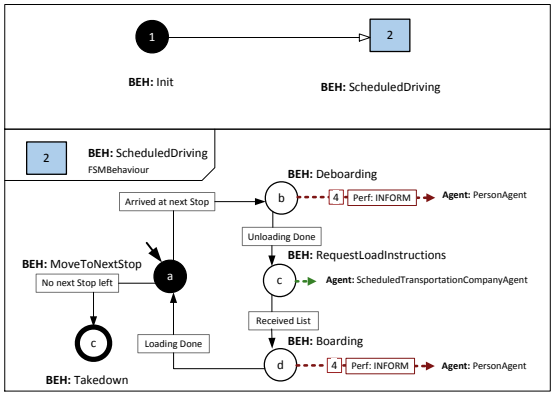

Figure 2: *Scheduled Vehicle Agent*.

functionality of a scheduled transportation agent and a vehicle. Instead of running on schedule, the autonomous vehicle agent picks up a person on demand, just like a taxi would. The answer of the autonomous transportation vehicle agent to a transport query (a, b) is a single transport option which covers the whole route from a to b since the autonomous vehicle is not bound to scheduled routes.

The *Person Agent* is a complex agent which communicates with transportation agencies and vehicles. It plans and executes travel routes, either by using public transport passively or by walking actively. The implemented model is shown in Fig. 3. We see a hierarchy of complex FSM behaviors. It shows that a person operates in a loop of planning and executing plans.

For the generation of plans (*Planning* behavior) shortest paths have to be generated for walking, as well as transportation agencies have to be contacted for options. While the start and goal locations are assumed to be on an arbitrary node in the map, not all transportation requests can be satisfied by bus, so that planning includes to pad vehicle usage with walks to or from the stops of the vehicles. Once the plan is fixed, it goes to execution (*ExecutePlan* behavior), where each step of the plan will be executed sequentially, either as an active (walking) or passive transport. We see that passive transport has to be monitored and can fail (e.g., by a timeout on the waiting time), so that the planning behavior has to be reinvoked on termination.

The *Birth-Giver Agent* initializes person agents with several parameters like the start location, a certain budget in time and cost, as well as a target location. All values are random numbers, drawn according to a given probability distribution. Optionally, the start and end location can be specified manually by determining a fixed node. The realistic modeling of the random process is crucial for the applicability of the simulation outcome. The better the model the better its prediction. These data might be indirectly inferred by information on where people live and where they work, or by monitoring their current use of vehicles. In reality, the choice of a transport mode depends on several details, such as spatial distance or social-economic variables (Buehler, 2011). In the end, a rather complex probability distribution for transport requests, humans and their queries should be derived.

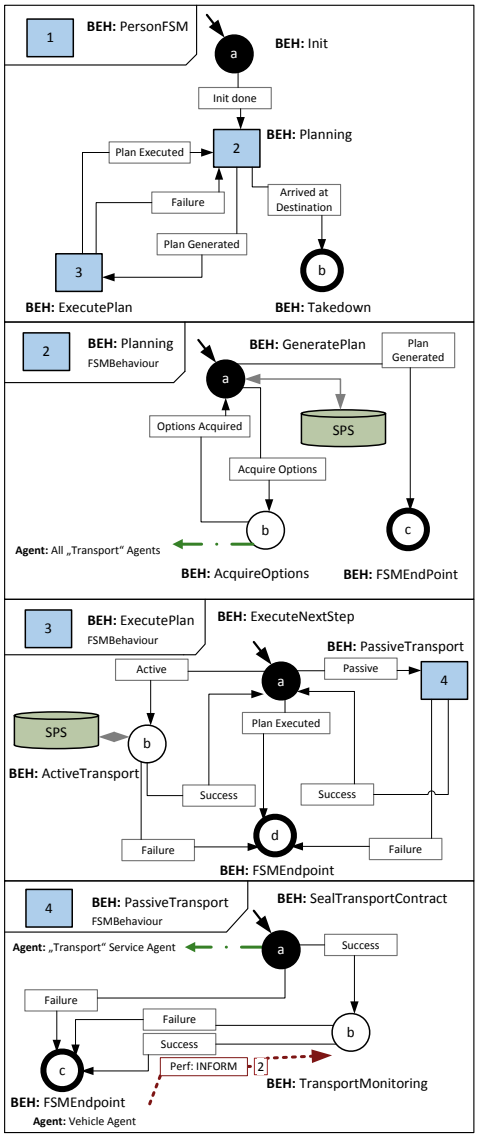

Figure 3: *Person Agent*.

4 SHORTEST PATH SEARCH

Single-source shortest paths search with Dijkstra's or the A* algorithm (Dijkstra, 1959) was already present in our MABS system, but showed performance prob-

lems and slowed down the overall MABS process considerably. Therefore we engineered the implementation by using a memory-efficient joint representation of graph and radix heap nodes. A *joint node representation* includes a state label (unlabeled, labeled or scanned), a linked list of edges, the element for storing the distances, the radix bucket a node is stored in as well as two pointers (pred, succ) for linking the elements in the radix heap. An edge is a pair of a successor node ID and according weight (cost/distance). We observed that the joint node representation is more crucial to the performance of the search than the proper choice of the data structure. One reason is to avoid memory allocation, another is that efforts for maintaining handles to the nodes can be avoided.

We also employed a key-based priority queue, exploiting the maximum weight C of all edges. A *radix heap* (Ahuja, Mehlhorn, Orlin, and Tarjan, 1990) maintains an array of $\lceil \lg(C+1) \rceil + 1$ buckets of sizes 1, 1, 2, 4, 8, 16, etc. Elements in the buckets are doubly-linked. More precisely, we maintain buckets $b[0..B]$ and bounds $u[0..B+1]$ with $B = \lceil \log(C+1) \rceil + 1$ and $u[B+1] = \infty$. The invariants of the algorithms are: 1) all keys in $b[i]$ are in $[u[i], u[i+1]]$, 2) $u[1] = u[0] + 1$, and 3) for all $i \in \{1, \ldots, B-1\}$ we have $0 \leq u[i+1] - u[i] \leq 2^{i-1}$.

The main difference to one-level buckets (Dial, 1969) is to use buckets of exponentially increasing sizes. Therefore, only $O(\lg C)$ buckets are needed. If edge weights are integers or floating point numbers $O(\lg C)$ can be interpreted as a constant independent from the number of nodes n and edges m.

For shortest path search, in the presence of a lower bound heuristic function h, A* (Hart, Nilsson, and Raphael, 1968) can be applied. Without reopening, A* resorts to a variant of Dijkstra's algorithm with $f(s) = h(s)$ for start node s and new weight $w'(u,v) = w(u,v) - h(u) + h(v)$ for all edges (u,v).

Theorem 1 (Time Optimality Shortest Paths Exploration)**.** *Given that the edge weights are computer words (64-bit integer or floating-point numbers) and provided a matching number representation for storing the accumulated distances at each node, our implementation of the Single-Source Shortest Paths Algorithm of Dijkstra (or A* with consistent heuristic) has optimal linear time complexity.*

Proof. The radix heap assumes that all edge costs in the graph are integers bounded above by C. The result is that Dijkstra's algorithm can be implemented with a time complexity of $O(m + n \lg C)$, where n is the number of nodes and m is the number of edges. Given that the logarithm of a 64-bit integer is bounded by a constant $\lg C = 64$, the running time on a modern computer is linear $O(m+n)$. If edge weights are doubles, $\lg C = \lg(1.79769 \cdot 10^{308}) = O(1)$ □

Furthermore, the radix heap is significantly simpler to implement compared to the Fibonacci heap and similar data structures. The radix heap achieves this time complexity by taking advantage of the property that shortest path distances fall into a finite range during the computation shortest paths by Dijkstra's algorithm.

The practical savings were considerable. Erstwhile unanalyzable systems turned out to be quickly analyzable, and even a full Dijkstra exploration was much faster than the original implementation of A*. In smaller graphs (200,000 nodes, 2,000,000 edges), generating the graph turned out to be more complex (4,443 ms) than actually searching it (1,668 ms). But even larger graphs (1,000,000 nodes, 10,000,000 edges) could be generated (30,574 ms) and searched (34,631 ms) in adequate time.

Recall, that shortest path queries are frequently posed by different agents such as persons and vehicles; buses in a dynamic world may have to recompute shortest paths from every bus station to the next (based on dynamic changes to the road network due to traffic jams). Frequent shortest path queries are also needed for preprocessing the graph in order to solve so-called *vehicle routing problems*. The decision making then relies on efficient solutions to the *Traveling Salesman Problem* (TSP), a touchstone for many general approaches in combinatorial optimization: Genetic algorithms, simulated annealing, tabu search, ant system, just to name a few. The problem is strong NP-hard and difficult to approximate unless the triangular inequality holds (Christofides, 1976; Arora, 1998). In an application of a forwarding agency, the TSPs are generated via shortest paths reductions of route networks. Each order to be served corresponds to one city in the TSP.

On the first glance, in case a better shortest path search performance is needed, A* is an obvious alternative to Dijkstra's algorithm. For a consistent heuristic evaluation function, it is optimal efficient and (up to tie-breaking) A* will expand the minimum number of nodes. For a heuristic that is strictly more informed than the trivial 0-heuristic applied in Dijkstra's algorithms, it is guaranteed to expand less nodes. However, at this level of speed per node, the number of expanded nodes is not the only key performance measure. We experienced that the time needed for computing the lower bound heuristic during shortest path search can negatively influence the overall performance. The computation of shortest paths of several hundred TSP matrices based on the distance calculations according to the Haversine formula resulted in A* requiring 330s, while Dijkstra's algorithm only required 274s. The

net computing time for computing all heuristic values took 207s, so that despite the savings in the number of node in A*, the heuristic itself is too complex. Hence, computationally more simpler heuristics are needed to beat Dijkstra's algorithm. Devising an efficiently computable consistent lower bound heuristic is non-trivial, as approximations of the heuristic are often inconsistent and thus sacrifies optimality, or call more involved data structures.

As an alternative improvment to the search, bidirectional shortest paths search can be executed. Moreover, there are many speed-up techniques that preprocess the graph for a much better search time (Bast, Funke, Sanders, and Schultes, 2007). However, such off-line improvements hardly translate to a dynamic scenario.

5 EXPERIMENTAL SETUP

In the context of a study on opportunities for novel last-mile connectivity, in a preliminary case study we looked at a MABS model for the Bangalore urban region, based on a user needs study (Akhilesh, Sitharam, Goswami, and Manjula, 2012) and the 2010 *Traffic Management Plans for Major Towns in Bangalore Metropolitan Region* for the Bangalore Metropolitan Region Development Authority conducted by Wilbur-Smith Associates.

To perform the simulation we extracted route data from the OpenStreetMap. As Bangalore was not predefined as a coherent district, we defined a bounding box and included also streets that cross the boundaries instead of clipping them exactly at the border of the bounding box. The road infrastructure contains 49,399 nodes and 134,222 edges and includes the International Airport of Bangalore (BIAL).

The bus routes and associated information on travel fares as well as fleet sizes is supplied by the Bangalore Metropolitan Transport Corporation (BMTC) authorities. They refer to real-life data. We decided to focus on the last-mile connectivity of passengers at BIAL. The distribution of passenger transports measured for a particular week shows that there are about 6,000 persons who use the bus lines on a day. In comparison there are about 10,000 persons taking a taxi (trips per day). Information on real-life bus schedules as well as bus stops is added manually. The simulated scenario models 8 bus lines, whose first or last stop is the BIAL.

The implementation of agents for the initial study is fully operational. In our scenario, a first birth-giver agent is responsible for creating person agents that act for persons located in the inner city district. The desire of this persons is to arrive the airport as soon as possible. Person agents that are created by a second

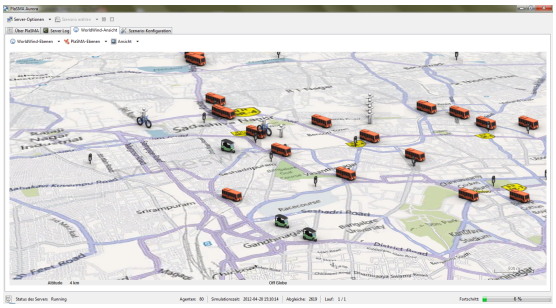

Figure 4: Running simulation with busses, persons, rickshaws and bikes.

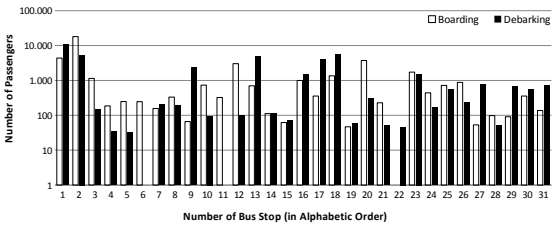

Figure 5: Simulation results. The open bars show the number of boarding and the closed bars the number of debarking passengers at each bus stop within an whole week. The bus stops are numbered consecutively to their official name in alphabetic order.

birth-giver agent represent individuals whose desire it is to get from the airport to the inner city district. Finally, a third birth-giver agent generates a fixed number of rickshaws which start at the airport and also drive to the inner city district.

To generate the start and end location randomly, we implemented a random walk strategy starting at a bus stop to generate requests at certain nodes with an arbitrary distance. Therefore we ensure that transport requests are not far from the encoded bus stations. Simulating dynamically changing traffic conditions is the subject of further research. As a result, edge following is mainly determined by the speed of the human or the vehicle as well as the type of the road.

Moreover, we added bikes, so that persons have the flexibility to decide how to get to the bus stop or home: if the distance is close enough (smaller than some predefined threshold) they walk, otherwise they ride by bike (see Fig. 4). The maximum walking speed of a person is limited to 5 km/h while the maximum speed of a bike is limited to 30 km/h. We simulated a time span of a whole week.

For our sample scenario we are interested in the number of passengers at each bus stop (see Fig. 5). No. 2 is the BIAL and, therefore, either the place of arrival or departure of each person. Low traffic is primarily caused by a high density of bus stops within a certain district. In total we simulated more than 45,000

agents with up to 1,100 agents running concurrently.

Performance indicators recorded in different simulations can easily be combined (e.g., averaged values). Besides the number of passengers that board or debark a bus, there are a lot of other interesting performance indicators for the simulation like the time spend by persons to reach their destination, the budget needed for the travel, or a combination of both. Experiments have shown that the bottleneck of the computations is computing several shortest paths on the routing graph by each person agent. Buses do not change their routes if there is no occurrence of unexpected events. As a consequence, the computation time of the buses are minimized by the use of an cache.

6 CONCLUSION AND OUTLOOK

This paper reported on the current status of a fine-grained MABS model for urban mobility in Bangalore. The physical simulation world model builds upon detailed OpenStreetMap data for the basic traffic infrastructure. It allows the mapping of the extensive network of bus lines operated by the BMTC. The generation of individuals that utilize modeled transport modalities can be configured according to actual distributions. Besides agent models for transport operators such as the BMTC and independent operators (using, for instance, taxis and rickshaws), a particular focus was put on the rational modeling of transport customers. These are equipped with capabilities for interleaved planning and execution of intermodal transport schedules. Experiments have shown that our system handles a hand-modeled excerpt of BMTC bus lines connecting airport and city center with a realistic number of more than 6,000 passengers per day in a time frame which make extensive experimentation practical in spite of complex planning and route-finding calculations performed by the simulation actors. The challenges to be addressed in ongoing multi-disciplinary research on smart mobility in metropolitan areas are manifold. The results of this research (e.g. optimized utilization of existing traffic systems or new emergency strategies) can be compared and evaluated by using a MABS model like the one we introduced in this paper.

ACKNOWLEDGEMENTS

The presented research was partially funded by the BMBF within the acatech project GRIP IT (Förderkennzeichen 01/S09048) and the German Research Foundation (DFG) within the Collaborative Research Centre 637 "Autonomous Cooperating Logistic Processes: A Paradigm Shift and its Limitations" (SFB 637) at the University of Bremen, Germany.

REFERENCES

Ahuja, R. K., Mehlhorn, K., Orlin, J. B., and Tarjan, R. E. (1990). Faster Algorithms for the Shortest Path Problem. *Journal of the ACM*, 37(2):213–223.

Akhilesh, K. B., Sitharam, T. G., Goswami, M., and Manjula, D. (2012). User Needs Study: Living Lab on Bangalore Mobility and ICT Research for Smart City Solutions. Technical report, CiSTUP.

Arora, S. (1998). Polynomial time approximation schemes for euclidian traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782.

Bast, H., Funke, S., Sanders, P., and Schultes, D. (2007). Fast Routing in Road Networks with Transit Nodes. *Science*, 316(5824):566–566.

Bellifemine, F., Caire, G., and Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE*, volume 5. Wiley.

Buehler, R. (2011). Determinants of transport mode choice: a comparison of germany and the usa. *Journal of Transport Geography*, 19(4):644 – 657.

Chen, B. and Cheng, H. H. (2010). A review of the applications of agent technology in traffic and transportation systems. *IEEE Transactions On Intelligent Transportation Systems*, 11(2):485 –497.

Christofides, N. (1976). Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University.

Dial, R. B. (1969). Shortest-path forest with topological ordering. *Comm. of the ACM*, 12(11):632–633.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

Gehrke, J. D., Schuldt, A., and Werner, S. (2008). Quality Criteria for Multiagent-Based Simulations with Conservative Synchronisation. In *13th ASIM Dedicated Conference on Simulation in Production and Logistics (ASIM)*, pages 177–186.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, SSC-4(2).

Hoogendoorn, S. P. and Bovy, P. H. L. (2001). State-of-the-art of vehicular traffic flow modelling. *Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 215(4):283–303.

Klügl, F. and Rindsfüser, G. (2011). Agent-based route (and mode) choice simulation in real-world networks. In *IAT*, pages 22–29.

Macal, C. M. and North, M. J. (2010). Tutorial on agent-based modelling and simulation. *Operational Research Society*, 4:151 – 152.

Meignan, D., Simonin, O., and Koukam, A. (2006). Multiagent approach for simulation and evaluation of urban bus networks. In *AAMAS'06*, pages 50–56.