A Graph Model for Taxi Ride Sharing Supported by Graph Databases

Dietrich Steinmetz[1], Felix Merz[1], Hui Ma[2], and Sven Hartmann[1(✉)]

[1] Clausthal University of Technology, Clausthal-Zellerfeld, Germany
{dietrich.steinmetz,felix.merz,sven.hartmann}@tu-clausthal.de
[2] Victoria University of Wellington, Wellington, New Zealand
hui.ma@ecs.vuw.ac.nz

Abstract. The emergence of more complex, data-intensive applications motivates a high demand of effective data modeling for graph databases to support efficient query answering. In this paper, we develop an intuitive graph data model for dynamic taxi ride sharing. We argue that our proposed data model meets the data needs imposed by three fundamental tasks associated with taxi ride sharing. An experiment consisting of a taxi ride sharing simulation with real-world data demonstrates the effectiveness of our modelling approach.

Keywords: Graph database · Data modelling · Ride sharing

1 Introduction

With the increasing number of complex, data-intensive problems emerged, not only are data sets getting bigger, but also data is getting more and more connected. For example, cyber-traffic analysis is a domain where the size and interconnectivity of data is massively increasing due to the still rising usage of the Internet [6]. A typical example of such complex, data-intensive problems is the Dynamic Taxi Ridesharing Problem (DTRP) [8]. This problem aims to find taxi routes and allocate passengers to taxis with the objectives of maximizing the number of serviced passengers and minimizing the operating cost and passenger inconvenience [1]. Due to the current rise of companies like Uber and Lyft and the possible utilization in autonomous driving it is quite popular. The DTRP is NP-hard [20], so solving it is computationally challenging. This problem also attracts attention because historic taxi trip records, e.g., from New York City (NYC) are openly available, which can be used to generate problem instances for experiments. Besides NYC [8,16,19] data from other cities like Shanghai and Beijing are frequently used for research [4,5,24].

To efficiently answer queries against large interconnected datasets, data should be organized carefully so that it does not become a bottleneck for applications. In particular, we need not only effectively store data but also consider the relationships among data and how this affects the performance of queries.

A. H. F. Laender et al. (Eds.): ER 2019, LNCS 11788, pp. 108–116, 2019.
https://doi.org/10.1007/978-3-030-33223-5_10

While relational databases are still the most common database technology for data-intensive storage and retrieval applications, they are not very efficient for queries of interconnected data due to the expensive joins [10]. To efficiently answer structural queries for complex, data-intensive problems, graph databases are a better choice since they provide native support not only for data but also for relationships between data [18,23]. Graph databases consist of nodes and relationships where nodes represent objects and relationships represent relations between objects [17, Sect. 1]. With them the retrieval of related objects or entire paths is often surprisingly efficient. This makes graph databases attractive for the DTRP where a lot of path calculations are needed to compute solutions.

For interconnected data there are many ways to store them in a graph database. To make best use of the capabilities of graph databases, data should be organized in a way that important queries can be performed efficiently. However, since the appearance of graph databases in the 1980s there has been far less research on conceptual modeling for them than for the relational databases [2]. The intuitive way of data modeling is to identify relevant concepts in the application domain and to abstract them as nodes and relationships [17, Sect. 3]. The following objectives will be achieved:

- To provide proper support of the fundamental tasks (e.g., finding taxi routes, allocating travelers to taxis) of the DTRP by a graph database. Based on the requirements we propose an intuitive model for the graph database.
- To evaluate our modeling approach we conduct a theoretical analysis of the data needs of the DTRP that are met by our proposed graph model as well as an experiment that explores the travel request satisfaction rate for different numbers of taxis. For that we model real-world datasets of the DTRP according to our proposed approach and store them in a graph database. For our prototype implementation, we use Neo4j to store and retrieve the data of the DTRP since it is currently the most popular graph database system [18,21], and road network data can be easily imported from Open Street Map (OSM) [22].

Organization. This paper is organized as follows. In Sect. 2 we briefly discuss related work and outline the DTRP and its subproblems (allocating travelers to taxis, sequencing the taxi schedule) to understand the requirements. In Sect. 3 we propose an intuitive graph model for the DTRP. In Sect. 4 we report on the experiment that we have conducted. In Sect. 5 we give conclusions and an outlook on future work.

2 Background

2.1 Data Modeling for Graph Databases

A graph model is a data model for graph databases and refers here to the labeled property graph model presented in [17, Sect. 3]. Data modeling for graph

databases has not been researched as thoroughly as for relational databases. Primarily an intuitive modeling approach is chosen, because the data often already exists in a graph-like structure in cases where a graph database is used. Neo4j lists multiple examples in their GraphGists list [12]. Some more sophisticated examples for the most common use cases of Neo4j are given at [11]. Intuitive graph modeling is also used in the literature in areas like cyber-traffic analysis [6], healthcare [14] and biology [3,7].

2.2 Traveler-Taxi Allocation and Taxi Schedule Sequencing

Our work is motivated by the DTRP, cf. [8,9,20]. In this problem, a set of taxis is running in a road network to serve customers, that is, to pick them up from their location and to drop them off at another location. Customers can share a taxi to save costs. Taxis have a limited seat capacity. To a certain extend taxis can make detours but taxi drivers need to account for the interests of other passengers. The objective of the DTRP is to achieve a high travel request satisfaction rate while minimizing the total travel distance (or cost) of taxis. The DTRP is an online problem since travel requests are coming on the fly and taxis need to be scheduled in real-time. The information on travel requests is unknown until the request is received.

NP-hard problems like the DTRP are particularly challenging, since problems in this class are suspected to have no polynomial-time algorithms. Therefore, heuristics are widely used to ensure scalability. The DTRP is a scheduling problem where taxis are resources and travel requests are tasks. Scheduling problems are very popular in many application domains, and often tackled by decomposing them into an allocation problem and a sequencing problem [15, Chap. 1].

Therefore, the DTRP is often treated as a composition of two subproblems: the traveler-taxi allocation problem and the taxi schedule sequencing problem. When a travel request is received, the goal is to allocate it to a taxi that is close enough to pick up the traveler while satisfying the constraints of the request as well as the constraints due to the seat capacity of the taxi and the requirements of other travelers who are already on board of the taxi. Once a taxi has a new request allocated to it, the schedule of this taxi has to be reorganized to account for the potential detour and waiting time. Traveler-taxi allocation and taxi schedule sequencing are not independent subproblems, since finding the best candidate taxi for a request depends on how that request affects the taxi route.

2.3 Requirements for Our Graph Model

To solve the DTRP efficiently, we aim to design a graph model for it. A review of the state-of-the-art literature on the DTRP resulted in the following set of important tasks that should be supported by our graph model, cf. [8,9,20]:

Task 1 *Retrieve the minimum travel time between the pickup and dropoff location for a specified travel request.*

Task 2 *Retrieve suitable taxis that can reach the pickup location of a request in a specified timeframe.*

Task 3 *Retrieve the remaining capacity and the remaining slack time at a specified point in the taxi schedule.*

These tasks are fundamental for the traveler-taxi allocation and the taxi schedule sequencing. The minimum travel time of a request is the basis of calculation of the maximum detour time for this request. Moreover, based on the time of a request, the minimum travel time and the maximum detour time it is possible to compute the latest arrival time of a request. This is crucial in order to decide for a candidate taxi whether it can arrive in time at the pickup location of a travel request. Finding suitable taxis for a request is the central aim for the traveler-taxi allocation. The maximum slack time of involved trips and the remaining capacity of a taxi are used when checking if a request can be inserted into a taxi schedule. Among the candidate taxis the best one will be selected, i.e., the one that causes the least increase of the overall travel distance or cost.

3 An Intuitive Graph Model for the DTRP

Based on the requirements discussed above we will now design a graph model for the DTRP that can meet the data needs of the three important tasks.

For the DTRP the following real-world entities are relevant: travel requests, taxis and a road network. We regard a *road network* as a directed graph $G = (V, E)$ where V is a set of road points and E a set of road segments. The road points are used to model intersections, terminal nodes and other points of interest, in particular potential pickup and dropoff locations of passengers. The road segments are used to model roads or part of roads. In our graph model, road points $v \in V$ are represented by nodes with label *RoadPoint*. For each road point we store the properties latitude and longitude. Road segments $e \in E$ are represented by relationships with type ROAD_SEGMENT between road points. For each road segment we store the property travel time.

Travel requests $r \in R$ are represented as nodes with label *TravelRequest*. Requests come from potential passengers with a desired pickup and dropoff location. For each request we store the properties datetime, passenger count and maximum slack time. Furthermore, each request is linked to two road points through two relationships with types IS_PICKED_UP_ AT and IS_DROPPED_OFF_AT for the pickup and dropoff location, respectively.

Taxis $h \in T$ are represented by nodes with the label *TaxiShift*. We regard a taxi as a shift of a taxi driver.[1] For each taxi we store the properties passenger capacity, shift start and shift end. We model the schedule of a taxi h as a set S_h of taxi states. Taxi states $\sigma_h \in S_h$ are represented by nodes with label *TaxiState*. Each taxi state is linked to a road point through a relationship with type IS_LOCATED_AT. The next taxi state $next_{S_h}(\sigma_h) \in S_h$ and the previous taxi state $prev_{S_h}(\sigma_h) \in S_h$ are linked through relationships with type IS_BEFORE.

[1] For simplicity, we assume in this work that each taxi has just one taxi shift.

Furthermore, there are relationships with type IS_SCHEDULED_BY between a taxi shift h and each of its taxi states $\sigma_h \in S_h$.

We regard a taxi state σ_h as a stay of taxi h at the road point v_{σ_h}. For each taxi state we store the properties number of passengers n_{σ_h}, period start t_{s,σ_h} and period end t_{e,σ_h}. They need to satisfy the constraint that the period end of a taxi state differs from the period start of the next taxi state by the travel time between their respective road points. Furthermore, for every taxi state we store a property s_{σ_h} whether a pickup or dropoff is happening. A *taxi stop* is a taxi state with a pickup or dropoff of some passenger. This causes a certain delay of γ called the change time.[2] Taxi stops have higher priority than other taxi states since they have to be passed while other taxi states connecting the stops can be replaced by different routes. To optionally skip the states there is an additional relationship with type IS_BEFORE_STOP at each stop connecting it to the next stop $next_{s,S_h}(\sigma_h)$ and previous stop $prev_{s,S_h}(\sigma_h)$ if existent.

Trips are represented by nodes with label *Trip*. For each trip we store the property remaining slack time. Once a request is accepted, it results in a trip of the traveler. Each trip is linked to a request through a relationship with type IS_INITIALIZED_BY. We regard the trip schedule as a subset of the schedule of its assigned taxi. Hence, there are relationships with type IS_SCHEDULED_BY between a trip and each taxi state that it shares with its assigned taxi.

After the definition of the nodes and relationships we can now assemble them in our intuitive graph model for the DTRP shown in Fig. 1.[3]

Proposition 1. *Using our intuitive graph model in Fig. 1, it is possible to meet all data needs of Tasks 1, 2 and 3*

Sketch of Proof. We will demonstrate that based on our intuitive graph model it is possible to solve the three important tasks.

For Task 1 we want to retrieve minimum travel time between the pickup and dropoff location of a request. To compute the minimum travel time between two road points $v_1, v_2 \in V$ we find the path P with the lowest total travel time $\omega(P)$ in the road network. We refer to this path P as the shortest path[4] $p(v_1, v_2)$. It can be computed using a shortest path algorithm like Dijkstra's or the A^*-heuristic.

For Task 2 we want to retrieve suitable taxis that are close to the pickup location of a request. We can use Dijkstra's algorithm with a maximum path weight to find the schedule states close to the pickup location of the request.

For Task 3 we want to retrieve the remaining capacity and the remaining slack time of a taxi in a given taxi state. The remaining seat capacity for a taxi

[2] This change time is not considered in some publications on the DTRP even though it has severe implications on ride sharing efficiency, since picking up passengers causes a schedule delay even if the pickup location is on the taxi route.

[3] For a better overview, we show the graph model with its nodes and relationships, but do not visualize the properties stored for nodes and relationships.

[4] In the literature this term is often used based on travel distance. Road segments, however, can have different travel speeds which leads to the invalidity of the triangle inequality on the road network. The path with the lowest total travel distance between two locations might not necessarily be the shortest path between them.

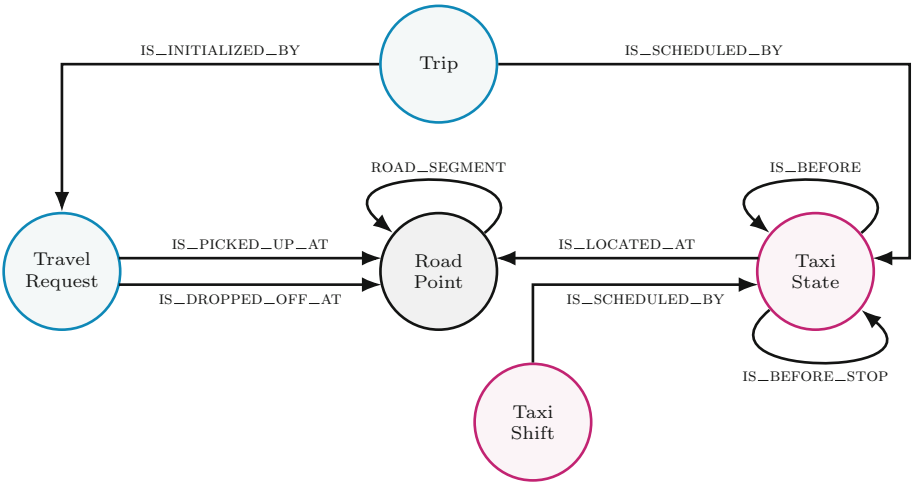

Fig. 1. Our intuitive graph model for the DTRP.

state can be computed from the total capacity (stored as a property of the taxi node) and the current passenger number (stored as an aggregated property of the taxi state node). The remaining slack time can be computed by inspecting the current taxi state and all future taxi stops, and finding the minimal value of the remaining slack times of the trips connected to them (stored as an aggregated property of the trip node).

4 Experimental Evaluation

To evaluate our modeling approach we have implemented our proposed graph model using Neo4j. In addition, we adapted the taxi ride sharing algorithms from [8,9,20] and implemented them as a plugin for Neo4j. Our experiment was based on real-world data of NYC utilizing OSM data and historic taxi trip data from NYC [13]. The imported road network consisted of 605,828 road points and 694,102 road segments, increasing to 927,621 road points and 1,931,503 road segments after data preprocessing, which included data cleaning and integration.

For our experiment we used data for the week from January 4 to 10, 2016 involving 319,081 travel requests after preprocessing 328,643 taxi trips. After the experiment the results were verified against our proposed graph model and the time and seat capacity constraints, to check for the correctness of the implemented algorithms.

Figure 2 shows the satisfaction rate of travel requests given by the number of trips that are shared or completed without sharing and the travel requests that are rejected on January 4, 2016 for 250, 500 and 1000 taxis. We observe that significantly less travel requests can be handled when using 250 taxis compared to 500 taxis, while 1000 taxis yield no significant improvement compared to 500 taxis.

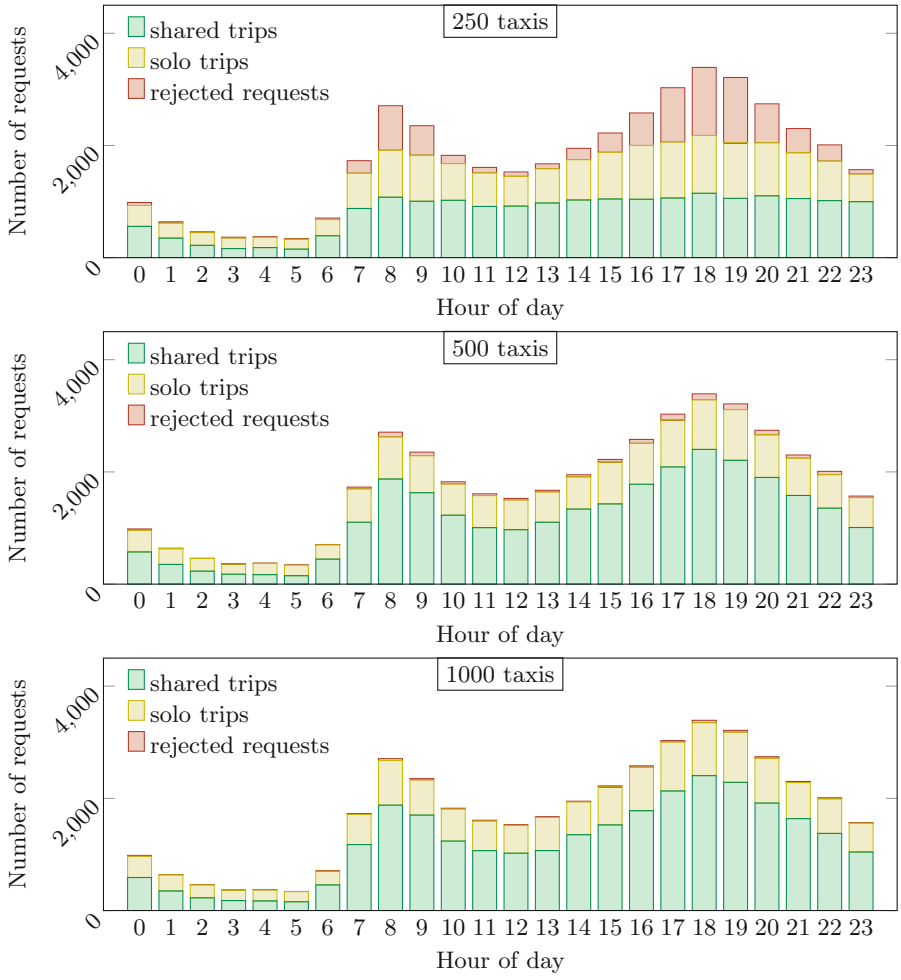

Fig. 2. Plots of the cumulative number of shared trips, solo trips and rejected requests for January 4, 2016 with 250, 500 and 1000 taxis, respectively.

5 Conclusion and Future Work

In this paper, we have proposed a labeled graph property model for the DTRP. Based on a review of state-of-the-art solutions for the DTRP we identified fundamental tasks for solving the problem and developed an intuitive graph model for the DTRP. We then verified that our proposed graph model has the capability to satisfy the requirements imposed by the fundamental tasks. In addition, we provided a prototype impelmentation of our graph data model and the respective taxi ride sharing algorithms, which we then utilised for a taxi ride sharing simulation with real-world data.

For the future, we plan to conduct further experiments to explore the performance and scalability of our approach. Moreover, we will investigate possible design alternatives for our intuitive graph model in order to further improve the support provided by the graph database backend for dynamic taxi ride sharing.

References

1. Agatz, N., Erera, A., Savelsbergh, M., Wang, X.: Optimization for dynamic ride-sharing: a review. Eur. J. Oper. Res. **223**, 295–303 (2012)
2. Angles, R., Gutierrez, C.: Survey of graph database models. ACM Comp. Surv. **40**, 1 (2008)
3. Graves, M., Bergeman, E.R., Lawrence, C.B.: Graph database systems. IEEE Eng. Med. Biol. Mag. **14**, 737–745 (1995)
4. Hou, Y., et al.: Towards efficient vacant taxis cruising guidance. In: IEEE GLOBECOM, pp. 54–59 (2013)
5. Huang, Y., Bastani, F., Jin, R., Wang, X.S.: Large scale real-time ridesharing with service guarantee on road networks. PVLDB **7**(14), 2017–2028 (2014)
6. Joslyn, C., Choudhury, S., Haglin, D., Howe, B., Nickless, B., Olsen, B.: Massive scale cyber traffic analysis: a driver for graph database research. In: International Workshop Graph Data Management Experiences and Systems, p. 3. ACM (2013)
7. Lysenko, A., Roznovăţ, I.A., Saqi, M., Mazein, A., Rawlings, C.J., Auffray, C.: Representing and querying disease networks using graph databases. BioData Min. **9**(1), 23 (2016)
8. Ma, S., Zheng, Y., Wolfson, O.: T-share: a large-scale dynamic taxi ridesharing service. In: IEEE ICDE, pp. 410–421 (2013)
9. Ma, S., Zheng, Y., Wolfson, O., et al.: Real-time city-scale taxi ridesharing. TKDE **27**, 1782–1795 (2015)
10. Mishra, P., Eich, M.H.: Join processing in relational databases. ACM Comp. Surv. **24**, 63–113 (1992)
11. Neo4j: Graph database use cases. https://neo4j.com/use-cases/
12. Neo4j: Neo4j GraphGists. https://neo4j.com/graphgists/
13. NYC Taxi & limousine commission: trip record data. http://www.nyc.gov/html/tlc/html/about/triprecorddata.shtml
14. Park, Y., Shankar, M., Park, B.H., Ghosh, J.: Graph databases for large-scale healthcare systems. In: IEEE ICDE Workshops, pp. 12–19 (2014)
15. Pinedo, M.L.: Scheduling: Theory, Algorithms, and Systems. Springer, Heidelberg (2016)
16. Qian, X., Zhang, W., Ukkusuri, S.V., Yang, C.: Optimal assignment and incentive design in the taxi group ride problem. Trans. Res. B: Meth. **103**, 208–226 (2017)
17. Robinson, I., Webber, J., Eifrem, E.: Graph Databases. O'Reilly, Sebastopol (2013)
18. Sahu, S., Mhedhbi, A., Salihoglu, S., Lin, J., Özsu, M.T.: The ubiquity of large graphs and surprising challenges of graph processing. PVLDB **11**, 420–431 (2017)
19. Santi, P., Resta, G., Szell, M., Sobolevsky, S., Strogatz, S.H., Ratti, C.: Quantifying the benefits of vehicle pooling with shareability networks. Proc. Nat. Acad. Sci. **111**, 13290–13294 (2014)
20. Santos, D.O., Xavier, E.C.: Dynamic taxi and ridesharing: A framework and heuristics for the optimization problem. In: IJCAI, vol. 13, pp. 2885–2891 (2013)
21. solidIT: DB-engines ranking - popularity ranking of graph DBMS. https://db-engines.com/en/ranking/graph+dbms

22. Steinmetz, D., Dyballa, D., Ma, H., Hartmann, S.: Using a conceptual model to transform road networks from OpenStreetMap to a graph database. In: Trujillo, J.C., et al. (eds.) ER 2018. LNCS, vol. 11157, pp. 301–315. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00847-5_22
23. Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y., Wilkins, D.: A comparison of a graph database and a relational database: a data provenance perspective. In: ACM Southeast Conference, p. 42 (2010)
24. Yuan, N.J., Zheng, Y., Zhang, L., Xie, X.: T-Finder: a recommender system for finding passengers and vacant taxis. IEEE TKDE **25**, 2390–2403 (2013)