A Sampling-based Strategy for Distributing Taxis in a Road Network for Occupancy Maximization (GIS Cup)

Kevin Buchin
k.a.buchin@tue.nl
TU Eindhoven, the Netherlands

Bram Custers*
b.a.custers@tue.nl
TU Eindhoven, the Netherlands

Irina Kostitsyna
i.kostitsyna@tue.nl
TU Eindhoven, the Netherlands

Martijn Struijs
m.a.c.struijs@tue.nl
TU Eindhoven, the Netherlands

ABSTRACT

We present a weighted sampling strategy for distributing a system of taxi agents on a road network. We consider a setting, in which each agent operates independently, following a prescribed strategy based on historical data. Furthermore, customer requests appear dynamically and are assigned to the closest unoccupied taxi agent.

We demonstrate that in this setting a simple sampling strategy based on the spatial distribution of historical data performs well in minimizing the average time that agents are unoccupied. The strategy is evaluated on taxi trip data in Manhattan and compared to various, more complex strategies.

CCS CONCEPTS

• **Information systems → Geographic information systems**;
• **Computing methodologies → Multi-agent planning**.

KEYWORDS

trajectories, taxi routing, multi-agent system, GIS Cup, dynamic scheduling, dial-a-ride

ACM Reference Format:
Kevin Buchin, Bram Custers, Irina Kostitsyna, and Martijn Struijs. 2019. A Sampling-based Strategy for Distributing Taxis in a Road Network for Occupancy Maximization (GIS Cup). In *27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '19), November 5–8, 2019, Chicago, IL, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3347146.3363348

1 INTRODUCTION

With the advent of distributed taxi services (such as Uber, Lyft, and others), and with the promising developments in the self-driving car technologies, interest in the field of multi-agent systems is increasing. An interesting feature of these ride-sharing services is that a central authority assigns customers, while the taxi drivers operate independently of each other.

The problem of assigning taxi drivers to customers has been carried out by performing fairness calculations at the central authority [3]. This approach puts all computation at the central authority, requiring it to track the locations of the drivers as well as the state of the drivers and possibly the preferences of drivers and customers.

The goal of the ACM SIGSPATIAL GIS Cup 2019[1] is to follow an alternative approach. The drivers in the system operate independently and follow a prescribed strategy for moving around the road network. The central authority only tracks the location of the drivers and whether they can pick up a new customer within a given time window. The drivers may use some historical model of the customer demand to determine their movement, but have no notion of the other drivers in the system. The goal is then to find a strategy for the drivers that maximizes the total time when the drivers are busy with customers.

Bai et al. [1] studied the problem of dynamic scheduling for independent, competing taxis. Their work focuses on assigning taxis to customers in an optimal way. Similarly, much related work focuses on pairing taxis with customers, referred to as *online taxi routing* by Bertsimas et al. [2]. Online taxi routing is a special case of the dynamic dial-a-ride problem with time windows, which in turn is a special case of the well studied dynamic vehicle routing problem (see [4, 5] for surveys).

In our setting, however, taxis and customers are simply paired based on nearest neighbors. Therefore the task is rather to *distribute taxis* on the network, in such a way that demands can be served within the time windows. On the level of individual taxis, this problem is related to recommender systems for taxis to find customers based on historical trajectories [8]. In our problem, we however do not aim to optimize for an individual taxi but for the overall system.

In this short paper, we present a strategy for the problem described above. The strategy revolves around weighted randomization of the destination of the drivers. This paper is organized as follows: In Section 2, we provide a more detailed description of the problem. In Section 3, we present our strategy and data model used. Then in Section 4, we present alternatives strategies, and compare these experimentally to our main strategy in Section 5. We then discuss variations of our data model in Section 6.

*Bram Custers is supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 628.011.00

SIGSPATIAL '19, November 5–8, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-6909-1/19/11…$15.00
https://doi.org/10.1145/3347146.3363348

[1] https://sigspatial2019.sigspatial.org/giscup2019/, last retrieved 07/09/2019. In this paper we use the date format dd/mm/yyyy.

2 PROBLEM DESCRIPTION

We now give a condensed description of the problem as formulated by the GIS Cup competition committee[2].

We are given a strongly-connected directed graph $G = (V, E)$ and a set of agents A, initially placed at random locations within G. The digraph has associated travel times $\tau(e)$ for every edge e in G. In addition, we are given a simulation period $\mathcal{T}$, starting at $t = 0$ and ending at some $t = T$. During the simulation period, an agent $a \in A$ moves around in the network according to a routing strategy $\pi : V \times \mathcal{T} \to V$. This strategy gives a destination location in the graph G based on the agent's current knowledge, which includes: (1) its location $p \in V$, (2) its state, (3) the current time $t \in \mathcal{T}$, (4) the current day of the week[3], (5) the data model (see below). The strategy is applied whenever an agent reaches a node in the graph. The movement of the agent takes into account the travel times and directions of the edges it travels on.

At unknown moments in time within the period $\mathcal{T}$, resources (customers) appear in the network. A resource has an associated source and destination locations. A central entity assigns the closest unoccupied agent to a resource. Additionally, the agent should be able to reach the resource within a fixed time limit of 10 minutes, while obeying predefined travel times on the links of the graph. If no agent can reach the resource, the resource expires. Once the assigned agent has transported the resource to the destination location, it can again be assigned to a new resource.

The agent strategy may use a historical data model to base its routing on. This data model is based on data of resource source and destination locations and associated times of past days.

We define $S(a, \pi) = \{[t_0, t_1] \mid t_0, t_1 \in \mathcal{T}\}$ for agent $a \in A$ following strategy π as the set of disjoint time intervals in $\mathcal{T}$ during which agent a is not transporting a resource. The problem is now to devise a strategy π^* for the agents such that the average *search time*, i.e., the time when the agent is unoccupied, for all agents is minimized:

$$\pi^* = \arg\min_{\pi} \frac{1}{\sum_{a \in A} |S(a, \pi)|} \sum_{a \in A} \sum_{s \in S(a, \pi)} duration(s)\,. \qquad (1)$$

During the contest the simulation is run on a random subset of days within the provided data set (refer to Section 3.2). For each of the chosen days the simulation starts one second before the first resource after 8:00, and ends after the expiration time of the last resource before 22:00.

3 MAIN ALGORITHM

3.1 Strategy: Weighted random destinations

The main challenge in developing a strategy is that each individual agent has almost no knowledge of the positions of the other agents, and has only partial knowledge about where customers may arrive. Ideally, the agents should spread in the network according to the aggregate distribution of the resources over space and time without coordinating their actions. We propose to use randomisation to resolve this issue. Concretely, in the weighted random destinations strategy (WR), each of the agents samples a random node from the graph and travels along the shortest (in travel time) path to that node. Whenever an agent reaches the node it has sampled or it has become free after being assigned a customer, it samples a new destination. The random sampling is done according to a *data model*, a discrete probability distribution that models the likelihood of a customer appearing near a node.

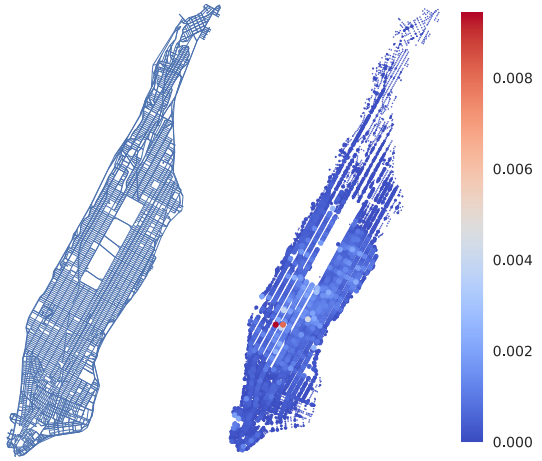

Figure 1: (Left) The Manhattan network, used in the simulation. (Right) Probability distribution for the weighted random strategy. The size of nodes is logarithmically scaled to highlight the distribution.

Since the agents all sample from the same discrete distribution, we can use the alias method to create a data structure of size linear in the number of nodes that allows sampling in constant time. We use the Java-implementation of Vose's algorithm [7] by Keith Schwarz[4] in our implementation.

3.2 Data model

The area of interest for the GIS Cup is restricted to the island of Manhattan, for which the Yellow Taxi Trip Data (YTTD) data set is available[5]. We derive the data model from the historical pickup and drop-off locations in the YTTD data set.

We simplify the graph by replacing chains of degree-2 nodes with a single edge. Thus the only nodes left in the graph correspond to proper intersections in the road network.

Next, using all resources of a given day that fit in the simulation period, we compute the weights on the nodes of the graph. For each node v we compute the total number of drop-offs d and pickups p on the edges outgoing from v, and add the value $p - \lambda d$ to the weight of v, for some parameter λ.[6] Finally, we reset the negative weights to 0. The resulting distribution is shown in Figure 1.

Rationale behind approach. Since we do not know what the initial distribution of the agents is nor the distribution for any later times, we assume no knowledge of the agent distribution at any time. The aim of the strategy is to then spread the agents according to the underlying distribution as much as possible. Parameter p is

[2] https://sigspatial2019.sigspatial.org/giscup2019/problem, last retrieved 07/09/2019.
[3] This underlines the assumption that customers probably follow a seasonal pattern.
[4] http://www.keithschwarz.com/interesting/code/?dir=alias-method, last retrieved 06/09/2019.
[5] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page, last retrieved 06/09/2019.
[6] In our experiments we settled on $\lambda = 0.2$.

weighted positively, so that the agents gravitate towards locations where customers are likely to appear. Parameter d is weighted negatively, as a location with a close-by drop-off is going to be visited by the agent delivering the corresponding customer (if the customer gets transported).

4 ALTERNATIVE STRATEGIES

We considered a number of alternative strategies. Due to the initial success of the main strategy, we investigated several other strategies with simple, often randomised, behaviour. The following five strategies work by iteratively selecting a new node in the graph and taking the shortest path to it. The main difference between them is the specifics of the node selection process.

Lévy flight strategy (LF). The Lévy flight is a type of random walk in a continuous space, with the Pareto distribution as its survivor function and is considered in foraging theory [6]. This random walk balances exploration of an unknown environment with exploitation of its resources. The Lévy flight strategy tries to apply this idea to our setting and also adds the knowledge from the data model. First, sample a value r from a Pareto distribution with scale parameter x_m and shape parameter α. Let R be the set of all nodes with travel time to the current location less than r, but more than $r \cdot f$, where f is a parameter within $(0, 1)$. The agent selects a destination from R according to the weights of the data model.

Avoidance strategy (AV). Agents sample a weighted random node only among the nodes with travel time from the current location at least r. The idea here is to avoid getting stuck in the same region, even if there are high weight nodes there.

Homebase strategy (HB). In this strategy, an agent marks its initial location as its 'home' h. When sampling a new destination, it assigns a weight of $w(v) + \zeta d(h, v)$ to each node v, where w is the weight of the node according to the data model, $d(h, v)$ is the travel time between v and the agent's home, and ζ is a parameter. The destination is sampled randomly according to these weights.

Fixed location strategy (FL). Given that the distribution of weights for the Manhattan network is skewed, a larger fraction of pickup and drop-off locations are in the area of Pennsylvania Station, we tested a strategy where all the agents concentrate at one node of the graph. As by the rules of the contest, the agents could not stay in one node, they traveled between the fixed location and a neighbor. While we did not expect this strategy to be effective on its own, it could be useful when mixed with other strategies.

Mixed strategy (MX). We have also experimented with composites of the above strategies, where each agent selects a strategy with a certain probability at the start of the simulation. In particular, we considered a mix of three strategies: the weighted random destination strategy (i.e, our main strategy), the avoidance strategy with $r = 600$, and the fixed location strategy. We denote the distribution of probabilities for selecting a strategy by p.

For a completely different approach, we explored a strategy based on machine learning, which proved to be much more complicated and time-consuming compared to our main strategy.

Reinforcement learning. The agents learn the appropriate actions to undertake, given the historical data, by applying reinforcement learning.

We take a grid over the considered region, aligned to the principal directions of the avenues and streets in the map. The state of an agent is defined as the cell that it is currently in.

The possible actions an agent can take are moving to another cell or staying in the same cell. Within the cell, an agent repeatedly picks a random destination and moves there. We restricted the movements between cells to the neighbouring cells only, including diagonal neighbours.

To learn the best action for an agent, we apply Q-learning. Let $Q : S \times A \to \mathbb{R}$ be the quality function that rates the quality of taking an action $a \in A$ given state $s \in S$. We then update this quality function using the following update rule

$$Q^{i+1}(s, a) \leftarrow (1 - \alpha)Q^i(s, a) + \alpha \left(r + \gamma \max_{a'} Q^i(a'(s), a') \right) . \quad (2)$$

Here, $a(s)$ is the state after taking action a in state s, $\alpha \in [0, 1]$ the learning factor, r the reward and $\gamma \in [0, 1]$ the discounting factor for later rewards. Initially, all $Q(s, a)$ values are set to zero. The quality function is shared among all agents.

We perform n training sessions. During training, the quality function is updated whenever an agent selects a new destination in the network. If the agent dropped off a resource before selecting a new destination, the reward is positive and defined as $1/\tau$, with τ the time since the last navigation moment. Otherwise, the reward is defined by $-1/(M - \tau)$, with $M = 1500$, chosen such that the reward is negative. When $\tau \geq M$, we use a reward of -1.

During evaluation, all agents use the same quality function Q that is now fixed.

Our initial experiments of this approach did not prove promising, and since it was substantially more complicated than the weighted sampling strategy, we chose to concentrate on the simpler randomized strategies. Still, a further investigation into the machine learning techniques in application to the studied problem is needed and we believe can lead to good results.

5 PERFORMANCE OF STRATEGIES

Despite being very simple to implement, experimental evaluation indicated that our main approach outperforms most of the alternative strategies on the given data set. We applied the strategies described in the previous section on the data set of the 01/06/2016 for the agent cardinalities $N = 5000, 7000$ and 10000 for a selection of parameter values for the different strategies. The results are shown in Table 1.

As can be seen from the table, the models all perform very similar to the weighted random destination strategy, where only the fixed location strategy is significantly worse than the other strategies. On closer inspection, the strategies seem to perform better the closer they are to the weighted random destination strategy. In particular, the mixed strategy performs best, when the first weight is set to 0.8-0.9. Therefore, only the weighted random destination strategy was used in our GIS Cup submission.

6 VARIATIONS OF DATA MODEL

With the weighted random destinations strategy, we considered different types of data models, based on the time span and number of days the aggregate the model is based on.

Strategy	Parameters	Agent cardinality		
		5000	7000	10000
WR	$\lambda = 0.2$	453	779	1436
FL	Penn. Station	486	854	1540
AV	$r = 400$	459	779	1436
	$r = 600$	458	781	1436
	$r = 800$	459	780	1436
HB	$\zeta = 1 \cdot 10^{-3}$	454	780	1436
	$\zeta = 2 \cdot 10^{-3}$	454	780	1436
	$\zeta = 1 \cdot 10^{-2}$	460	780	1436
LF	$\alpha = 2.0, f = 0.5, x_m = 200$	508	798	1435
	$\alpha = 2.0, f = 0.5, x_m = 400$	472	797	1435
	$\alpha = 2.0, f = 0.5, x_m = 600$	471	799	1435
MX	$p = (0.5, 0.5, 0)$	455	778	1436
	$p = (0.6, 0.4, 0)$	454	781	1437
	$p = (0.7, 0.3, 0)$	454	781	1437
	$p = (0.8, 0.2, 0)$	453	781	1437
	$p = (0.9, 0.1, 0)$	453	781	1437
	$p = (0, 0.5, 0.5)$	476	781	1437
	$p = (0, 0.6, 0.4)$	470	781	1437
	$p = (0, 0.7, 0.3)$	459	780	1437
	$p = (0, 0.8, 0.2)$	459	781	1437
	$p = (0, 0.9, 0.1)$	458	780	1437
	$p = (0.5, 0, 0.5)$	478	781	1437
	$p = (0.6, 0, 0.4)$	475	781	1437
	$p = (0.7, 0, 0.3)$	467	781	1437
	$p = (0.8, 0, 0.2)$	461	781	1437
	$p = (0.9, 0, 0.1)$	456	781	1437

Table 1: Average search time in seconds per strategy, displayed per selection of the associated parameters of the strategy. The strategies were tested on the data set of the 01/06/2016 for the cardinalities 5000, 7000 and 10000.

We considered the following approaches:

- take the distribution of a single day (SD)
- take the distribution of a single day, with time bins (TB)
- aggregate the distributions of weekdays (AWD)
- aggregate the distributions of weekdays, with time bins (ABWD)
- aggregate the distributions of weekend days (AED)
- aggregate the distributions of weekend days, with time bins (ABED)

We aggregate the distributions of multiple days by first constructing the probability distribution per day and/or time bin and then taking the average distribution over all considered days. For the single day and binned single day models (SD and TB), we used a different day than the tested day for testing the performance. We picked time bins of an hour in size. When selecting a new destination, the agent uses the data model that is associated with the time bin in which the agent plans its next move.

As can be seen from Table 2, the influence of taking into account the seasonality of the distributions has no significant influence. In addition, aggregation of the data over multiple days also has no significant influence.

N	SD	TB	AWD	ABWD	AED	ABED
5000	452	453	453	452	453	453
7000	779	778	779	778	779	778
10000	1436	1436	1436	1436	1436	1436

(a) Training day: 08/06/2016 (Wed). Simulation run: 01/06/2016 (Wed).

N	SD	TB	AWD	ABWD	AED	ABED
5000	497	496	497	497	497	498
7000	592	591	594	592	591	592
10000	1164	1164	1164	1164	1164	1164

(b) Training day: 26/03/2016 (Sat). Simulation run: 19/03/2016 (Sat).

Table 2: Average search times in seconds for the described models, run on two single weekdays with agent cardinalities (N) 5000, 7000 and 10000.

7 DISCUSSION

While we considered a variety of strategies, in our experiments one of the simplest strategy, the weighted random destination strategy, performed best. This may indicate that with no information about other taxis and without means to coordinate, it is difficult to do more than to aim at distributing to the expected customer distribution.

In our experiments, more elaborate data models did not (significantly) outperform a simplistic data model that only used the spatial distribution of customers from one day. This may be due to the fact that any distribution that we computed was skewed towards one geographical location (Figure 1). We expect that taking spatio-temporal variations into account would have a larger effect for less skewed distributions.

There are several approaches that we considered, but did not pursue. Firstly, one could take into account the general traffic patterns: partition the map into regions and estimate the percentage of vehicles flowing from one region to another from the historical data. Secondly, it may be advantageous to smooth the probability distribution, which would diversify the selected destination locations. This could spread the agents more over high pickup density areas. Finally, in this work we chose not to pursue the machine learning approach, which with carefully chosen model parameters can lead to efficient solutions.

REFERENCES

[1] R. Bai, J. Li, J. A. Atkin, and G. Kendall. A novel approach to independent taxi scheduling problem based on stable matching. *Journal of the Operational Research Society*, 65(10):1501–1510, 2014.

[2] D. Bertsimas, P. Jaillet, and S. Martin. Online vehicle routing: The edge of optimization in large-scale applications. *Operations Research*, 67(1):143–162, 2019.

[3] A. Glaschenko, A. Ivaschenko, G. Rzevski, and P. Skobelev. Multi-agent real time scheduling system for taxi companies. In *8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, pages 29–36, 2009.

[4] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013.

[5] U. Ritzinger, J. Puchinger, and R. F. Hartl. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research*, 54(1):215–231, 2016.

[6] G. M. Viswanathan, S. V. Buldyrev, S. Havlin, M. Da Luz, E. Raposo, and H. E. Stanley. Optimizing the success of random searches. *Nature*, 401(6756):911, 1999.

[7] M. D. Vose. A linear algorithm for generating random numbers with a given distribution. *IEEE Transactions on software engineering*, 17(9):972–975, 1991.

[8] N. J. Yuan, Y. Zheng, L. Zhang, and X. Xie. T-finder: A recommender system for finding passengers and vacant taxis. *IEEE Transactions on knowledge and data engineering*, 25(10):2390–2403, 2012.