Distance-Aware Competitive Spatiotemporal Searching Using Spatiotemporal Resource Matrix Factorization (GIS Cup)

Joon-Seok Kim
George Mason University
jkim258@gmu.edu

Dieter Pfoser
George Mason University
dpfoser@gmu.edu

Andreas Züfle
George Mason University
azufle@gmu.edu

ABSTRACT

Congested traffic wastes billions of liters of fuel and is a significant contributor to Green House Gas (GHG) emissions. Although convenient, ride sharing services such as Uber and Lyft are becoming a significant contributor to these emissions not only because of added traffic but by spending time on the road while waiting for passengers. To help improve the impact of ride sharing, we propose an algorithm to optimize the efficiency of drivers searching for customers. In our model, the main goal is to direct drivers represented as idle agents, i.e., not currently assigned a customer or resource, to locations where we predict new resources to appear. Our approach uses non-negative matrix factorization (NMF) to model and predict the spatio-temporal distributions of resources. To choose destinations for idle agents, we employ a greedy heuristic that strikes a balance between distance greed, i.e., to avoid long trips without resources and resource greed, i.e., to move to a location where resources are expected to appear following the NMF model. To ensure that agents do not oversupply areas for which resources are predicted and under supply other areas, we randomize the destinations of agents using the predicted resource distribution within the local neighborhood of an agent. Our experimental evaluation shows that our approach reduces the search time of agents and the wait time of resources using real-world data from Manhattan, New York, USA.

CCS CONCEPTS

• **Information systems → Geographic information systems**; • **Computing methodologies → Non-negative matrix factorization**; **Agent / discrete models**.

KEYWORDS

Non-negative matrix factorization, simulation, discrete event simulation, spatio-temporal search, distance-aware search

ACM Reference Format:
Joon-Seok Kim, Dieter Pfoser, and Andreas Züfle. 2019. Distance-Aware Competitive Spatiotemporal Searching Using Spatiotemporal Resource Matrix Factorization (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/10.1145/3347146.3363350

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

ACM ISBN 978-1-4503-6909-1/19/11…$15.00
https://doi.org/10.1145/3347146.3363350

1 INTRODUCTION

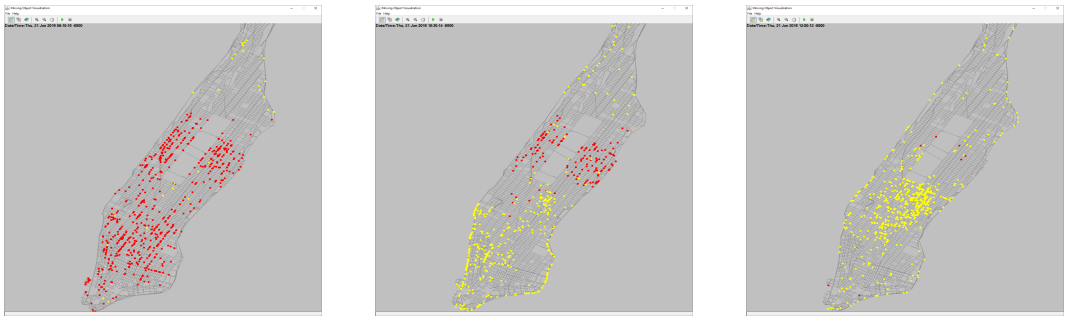
In the United States alone, drivers spend 6.9 billion of man-hours stuck in traffic each year [5] leading to a annual waste of more than 11 billion liters of fuel [3]. The transportation sector alone contributes 29% of the total US Green House Gas emissions, of which in turn 59% are are contributed by passenger vehicles [4]. Adding to this balance is the increasing popularity of ride-sharing services such as Uber and Lyft. In the future, a fleet of autonomously driving ride-sharing vehicles will worsen this situation as predicted by Bryan Mistele, founder and CEO of INRIX in a SIGSPATIAL 2017 keynote [1]. Ride-sharing and autonomous vehicles will not only transport passengers but also incur idle trips to pick up and search for passengers.

The aim of this work is to help mitigate this emerging problem by providing smart strategies and heuristics for drivers (denoted as *agents*) to smartly traverse the network while searching for customers (denoted as *resources*) to minimize the average time of agents without passengers (denoted as *search time*). We leverage available resource data (origin and destination) by employing a non-negative matrix factorization approach to model and predict the distribution of resources over space and time. Given this prediction model, we describe our algorithm to guide agents searching for resources. Our algorithm combines the following properties:

- **Generalization:** Assuming that resources vary between days, the algorithm should adapt to random deviations in resource distributions, rather than overfitting to the resource distribution in the training data set;
- **Balancing Utility and Distance:** Agents should find a balance between exploration (moving to more useful locations) and exploitation (remaining in the current location). The algorithm must find a balance between these two aspects;
- **Fairness:** Without allowing agents to communicate with each other, the algorithm should evade flocking towards the same destination, to avoid oversupplying some regions while causing starvation in others;
- **Timing:** Agents should move towards regions where there is a maximal chance of finding a resource at the time of arrival.

The outline of this papers is as follows. We first motivate our approach by visually exploring a simulation of agents in New York City using real resource data as described in Section 2. Section 3.1 describes our non-negative matrix factorization approach to model and predict resource demand over space and time. Using this model, the search algorithm for agents is given in Section 3. Finally, our experimental evaluation in Section 4 shows empirically that our algorithm reduces the average search time of agents and the average wait time of resources compared to baseline solutions.

[1] https://sigspatial2017.sigspatial.org/keynotes/#bryan



(a) 9:10am: Undersupply (b) 10:30am: Bad Agent Distribution (c) 12:00pm: Oversupply

Figure 1: Several screenshots from the simulator (red dot: available resource, yellow dot: searching agent)

2 VISUAL DATA ANALYSIS

At the outset of our work, we had to understand the problems and challenges that lead to a high average search time of agents. We leverage an open-source simulation framework called COMpetitive SEarching Testbed (COMSET)[2]. A simulation in this context consists of four types of entities: a road network, mobile agents (i.e., taxicabs), and stationary resources (i.e., customers), and an assignment authority. The road network is from Manhattan, New York City, USA. All agents are introduced at the beginning of a simulation, each located at a random location on the road network. Resources are streamed into the simulation, each having a pick-up location and a destination and using New York City Trip Record Data [1]. Each resource has a maximum life time (MLT) starting from its introduction; if the MLT is reached, the resource will be automatically removed from the system, an outcome that we call resource expiration. The assignment authority matches new resources to their nearest agent if any agent exists that is able to reach the resource before it expires. If no agent can reach the resource, the resource will wait and may eventually expire should no agent become available nearby. An assigned agent will travel (on a shortest path) to the resource and take the resource (on a shortest path) to their destination. We use a simple heuristic that makes agents travel to a random destination until they are assigned to a resource. We added a visualization layer to COMSET to show a road network as well as available resources and searching agents. Figure 1 shows searching agents as yellow dots, while available resources are shown as red dots. A video that demonstrates our visualization is found at *http://giscup19.joonseok.org/* and is summarized in the following.

During the rush hour peak, we observe an under supply as shown in Figure 1a, which shows the simulation at 9:10am on January 21st, 2016. At this time, resources become available at such high frequency, that agents cannot possibly keep up anymore. During this time, agents that finish a ride are immediately assigned to their next resource. In this case, we observe many available resources near central park, while only a few resources are available on the northern and southern side of Manhattan. Only very few agents find themselves searching in a resource-desert with no available resource in range. However, keep in mind that some of these agents have not been taken to this desert by their choice, but rather, were taken there by a destination of another trip. This yields a situation of "anywhere else is better than here" where the baseline that chooses a random destination is almost guaranteed to take them out of the desert. Thus, in this case, very little is to be gained in terms of reducing search time.

At 10:30am of the same day, we observe in Figure 1b that resources are available near Central Park. This implies that there is no empty agent available to reach the resource within its expiration time. At the same time, we also see that agents are searching in other parts of the network. This is an interesting, because in this case it is possible to achieve a potential improvement! If some of the searching agents were close to the available resources, they could be assigned and would not have to search any longer. Thus, to make an improvement, searching agents need to be led to areas where resources are expected to become available, rather sending them to random destinations.

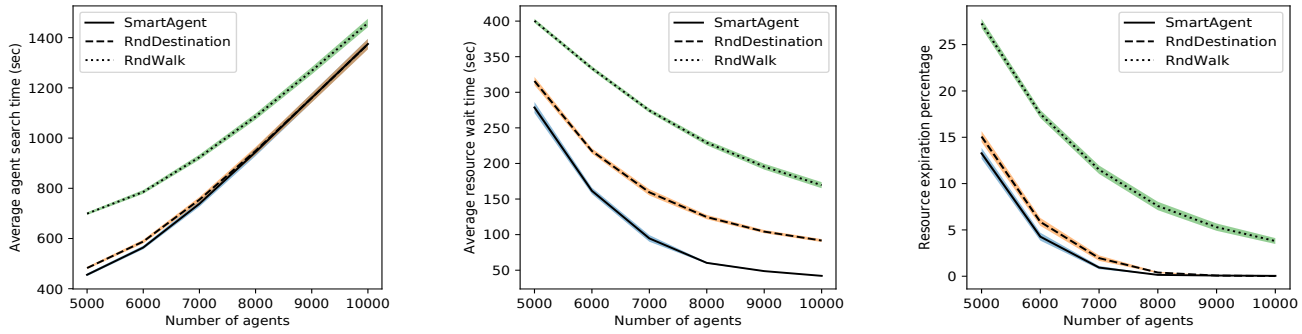
The third case that can be observed in this simulation is oversupply, as shown in Figure 1c during noon on the same day. During this time, resources become so scarce that it is simply impossible to assign each agent to a resource. During these times, we cannot further reduce the search time of agents, as a resource can only be assigned to a single agent.

To summarize the lessons learned from data visualization, our only hope of reducing the search time is to improve the cases when there are searching agents in some parts of the network, while there are resources available in other parts of the network. Towards this goal, we need to predict these areas of high resource availability, and we have to direct searching agents to find these resources quickly. In our video, you can further see that our proposed algorithm is seemingly perfectly able to supply all resources with agents. In the video, you can see that during the afternoon rush at around 5:50pm, the network is nearly empty, as all resources and agents are assigned to each other. Agents move smartly to locations where they will be needed. We note that the bad agent distribution in the morning (at around) 10:30am using our approach is an artifact resulting from the initial distribution of agents to uniform random locations.

3 ALGORITHM DESCRIPTION

Resource prediction is at the core of our algorithm. We then leverage this prediction to guide searching agents to minimize their search time.

[2]https://github.com/Chessnl/COMSET-GISCUP



(a) Average Agent Search Time (b) Average Resource Wait Time (c) Resource Expiration Rate

Figure 2: Experimental Comparison to Baseline Solutions

3.1 Spatio-Temporal Resource Prediction

Given a large set of resources, each annotated with an origin location, a destination location, and a time-stamp, we first aggregate all information in a spatio-temporal resource origin matrix M. Each cell M_{ij} of M corresponds to the number of resources that became available in a spatial region i during time interval j. We define this matrix formally as follows:

Definition 3.1 (Spatio-Temporal Resource Origin Matrix). Let $\mathcal{D}$ be a training set of resources such that each resource $r_i \in \mathcal{D}$ is a triple (o_i, d_i, t_i), where o_i is the origin of the resource, d_i is the destination of the resource, and t_i is the time at which this resource appears. Further, let $\mathcal{S} = \{S_1, ..., S_{|\mathcal{S}|}\}$ be a set of spatial regions, and let $\mathcal{T} = \{T_1, ..., T_{|\mathcal{T}|}\}$ be a set of temporal intervals spanning all days of $\mathcal{D}$. Then, we define the following $|\mathcal{S}| \times |\mathcal{T}|$ matrix as follows:

$$m_{ij} = |\{r_k \in \mathcal{D} | o_k \in S_i \wedge t_k \in T_j\}|$$

To divide our dataset into spatial regions, we treat each individual road segment as a spatial region and we split time into 20-minute intervals. Thus, we define $\mathcal{S}$ as the set of road segments, and $\mathcal{T}$ as the set of 20-minute intervals of the dataset. Next, we factorize M using canonical decomposition [2] into two matrices A and B, where A is a $|\mathcal{S}| \times k$ matrix that describes each region in $\mathcal{S}$ by k latent features, and B is a $k \times |\mathcal{T}|$ matrix that describes each time interval by a set of k latent features. Here, k is a parameter that allows us to specify the level of detail of the model. This parameter is selected empirically (for this data set) as described in Section 4.

Expansion of matrices A and B yields the estimated matrix $\hat{M} = A \cdot B$ that we use for prediction of future resources as follows. For a new day at a time t, for which resources have been observed only up to t, we find the most similar day s (using cosine distance) among all days in the training set using the time observed so far. Then we look up the column $\hat{t}$ of $\hat{M}$ that corresponds to time t on day s. The following subsection shows how we use this estimation, and the subsequent columns (which corresponds to the resources estimations of the 20-minute intervals after $\hat{t}$) for prediction.

We note that our model does not utilize the destination locations of resources, as we see no way to leverage the destination distribution to reduce search times. While it is possible that search times can be reduced by having agents reject resources that lead them to remote places, the simulated setting that we aim to optimize does not allow agents to reject resources.

3.2 Agent Guidance

The predicted spatio-temporal resource distribution is used to guide searching agents to minimize their search time. In a nutshell, we first leverage the prediction matrix $\hat{M}$ to draw random candidate resources predicted to appear within a time horizon Δ. To ensure that agents prefer closer locations, we weigh the resulting candidate resources by their distance to the agent. Then, we draw resource from this weighted distribution as destination for the agent.

More formally, let t denote the current time, and let $\hat{M}$ be the prediction matrix that predicts the number of resources for each road segment at future time intervals as described in Section 3.1. Thus, a cell $\hat{m}_{ij}$ contains the predicted number of resources at road segment S_i at time T_j. Given current location of an agent l, time horizon Δ, sampling size N, and distance decay factor γ, we determine the destination of a searching agent as follows. First, we draw a sample $\mathcal{S}$ of N candidate resources from the distribution of resources estimated within the next Δ minutes:

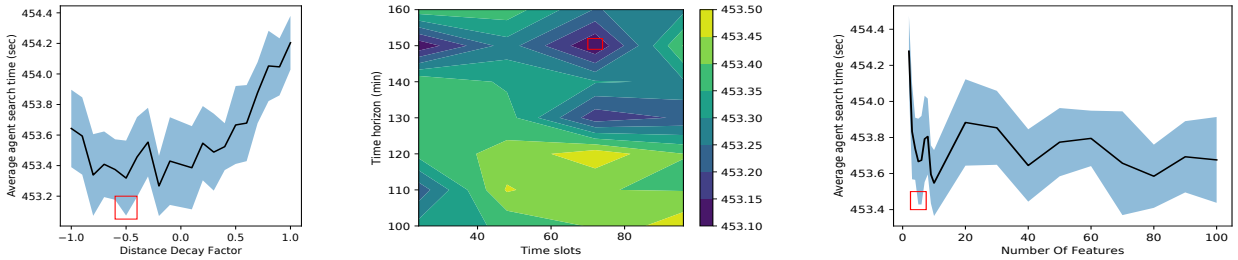
$$\mathcal{S} = Sample_N(\hat{m}_{\cdot,[t,t+\Delta]}),$$

where $\hat{m}_{\cdot,[t,t+\Delta]}$ is set of resource predictions of all locations in the network at any times between the current time t and the time horizon $t + \Delta$. The function $Sample_N(\hat{m}_{\cdot,[t,t+\Delta]})$ draws N random sample from the sampling space $\hat{m}_{\cdot,[t,t+\Delta]}$, where each element $\hat{m}_{i,j} \in \hat{m}_{\cdot,[t,t+\Delta]}$ of the sampling space has a probability of $\frac{m_{i,j}}{\sum \hat{m}_{\cdot,[t,t+\Delta]}}$ of being drawn.

The resulting set $\mathcal{S}$ contains a set of N candidate locations chosen randomly from the distribution of resources predicted in the next Δ minutes. From this set $\mathcal{S}$ we draw a destination for a searching agent weighted by the distance to the agent's current location l:

$$\mathcal{D} = Sample_1([(s, dist(l, s)^\gamma) | s \in \mathcal{S}]),$$

where $dist(l, s)$ is the network distance between the agent's current location l and a candidate resource $s \in \mathcal{S}$, γ is a parameter that controls the agent's preference to visit nearby locations, $[(s, dist(l, s)^\gamma) | s \in \mathcal{S}]$ is a list (note that this list is not a set, as it may contain duplicates) of pairs of candidate resources in $\mathcal{S}$ and their corresponding distance values taken to the power of γ, and the function $Sample_1$ is the same function that randomly draws a weighted sample from $\mathcal{S}$ weighted by their. That is, each sample $s \in \mathcal{S}$ is chosen with a probability of $\frac{dist(l,s)^\gamma}{\sum_{s' \in \mathcal{S}} dist(l,s')^\gamma}$. The location of this resource D is chosen as destination of the searching agent.



(a) Distance Decay Factor γ (b) Time horizon and granularity of temporal slot (c) Number of NMF features (1-100)

Figure 3: Average Agent Search Time when Varying Algorithm Parameters (red box: the selected parameters)

4 EXPERIMENTAL EVALUATION

Using the simulated scenario described in Section 2, our goal is to leverage our algorithm described in Section 3 to "control the mind of agents" when they are not currently on a trip to minimize the average search time of agents. Per default, our simulation uses 5000 agents, a sample size of $N = 5$, a time horizon of $\Delta = 140$ minutes, and a number of latent features $k = 6$. We compare our approach to two baselines. The *Random Walk* baseline lets agents cruise in random directions. Thus, whenever a random walk agent reaches an intersection, they will continue into a direction chosen uniformly at random. The *Random Destination* baseline chooses a random destination uniformly random from all network nodes.

The results of comparing our approach, denoted as SmartAgent, to other methods are shown in Figure 2. First, we note that *the random walk yields, by a margin, the worst results.* This is attributed to not being able to supply areas of high demand due to a high chance of keeping an agent in the vicinity of their most recent drop-off. We further observe that the *random destination approach yields much more promising results.* The reason is that a random destination is likely to lead an agent through the center, which is where most resources appear in this simulation. We further observe in Figure 2a that, at first glance, our SmartAgent approach appears to yield only a marginal improvement over the random destination approach. However, as we have observed in Section 2, during most times of the simulation no improvement is possible, as the system is either oversupplied with agents (assigning all resources immediately), or undersupplied (having no agents searching). Thus, the improvement observed in Figure 2a is attributed only to short periods where agents have to compete for resources, leading to a very noticeable improvement during these times. This improvement becomes evident in Figure 2b showing that *the average wait time of resources is decreased drastically by our SmartAgent approach.* The reason is that our agents are able to anticipate where new resources will appear, thus traveling to these regions ahead of time. Figure 2c further shows that our approach is able to prevent the expiration of resources with a much smaller number of agents, thus allowing to achieve the perfect quality of service for resources at a drastically reduced cost in terms of agents paid and fuel wasted. Next, we explore the effect of the various parameters used by our approach shown in Figure 3.

Distance Decay Factor γ Parameter γ controls how agents prefer close to far-away predicted resources. Figure 3a shows the average search time (and 95% confidence intervals) of 100 simulation runs (with different random seeds) for $-1 \leq \gamma \leq 1$ in 0.1 increments. We first observe that positive values of γ cause more distant places to be weighted higher, thus sending agents to distant resources incurring large search times. We also note that $\gamma < -0.8$ causes agents to become too attracted to their nearest candidate resource, thus getting stuck in their current locations. Empirically, we found that a value of $\gamma = -0.5$, thus using the inverse square root to weigh distances, yields good results.

Time Horizon Δ and Temporal Resolution Figure 3b shows how the average search time of agents is affected by (a) a different time horizons Δ and (b) different temporal resolution. Therefore, the x-axis of Figure 3b denotes the number of equal-duration time slots used per day, whereas the $y - axis$ varies the time horizon Δ used to predict future resource availability. For brevity, Figure 3b shows only a subarea of this experiment that has the least average search time. This result justifies our choice of having a temporal resolution of 72 intervals per day, which equals 20 minutes per interval and using a time horizon Δ of 140 minutes.

Number of Latent Features Finally, Figure 3c shows the average agent search time as the number of latent features k is varied. We observe that, in general, a larger value of k is beneficial. However, we note that in our experiments, any day used for testing is also used for training the prediction matrix $\hat{M}$ (a setting that was permitted by the ACM SIGSPATIAL GIS Cup 2019). Therefore, we note that high values of k may overfit to the training data, and thus, not generalize to new days. To avoid this overfitting, we chose $k = 6$ as our default value. However, experiments to investigate this overfitting effect are the subject of future work.

ACKNOWLEDGMENT

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under cooperative agreement No.HR00111820005 and the National Science Foundation Grant CCF-1637541. The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

REFERENCES

[1] New york city taxi and limousine commission, trip record data. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page, Accessed Sep 17, 2019.
[2] A. Cichocki and A.-H. Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Trans. Fund. Elect. Comm. Comp. Sci.*, 92(3):708–721, 2009.
[3] G. Cookson. INRIX global traffic scorecard (2017). INRIX Research, February, (published February 2018).
[4] D. Moran, K. Kanemoto, M. Jiborn, R. Wood, J. Többen, and K. C. Seto. Carbon footprints of 13 000 cities. *Environmental Research Letters*, 13(6):064041, jun 2018.
[5] D. Schrank, B. Eisele, T. Lomax, and J. Bak. Urban Mobility Scorecard. The Texas A&M Transportation Institute and INRIX, 2015.