Supply-Demand Ratio and On-Demand Spatial Service Brokers: A Summary of Results

Reem Y. Ali*, Emre Eftelioglu*, Shashi Shekhar*, Shounak Athavale[§], Eric Marsman[#]

*Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN

[§]Ford Motor Company, Palo Alto, CA

[#]Ford Motor Company, Dearborn, MI

{alireem,emre,shekhar}@cs.umn.edu, {sathaval,emarsman}@ford.com

ABSTRACT

This paper investigates an on-demand spatial service broker for suggesting service provider propositions and the corresponding estimated waiting times to mobile consumers while meeting the consumer's maximum travel distance and waiting time constraints. The goal of the broker is to maximize the number of matched requests while also keeping the "eco-system" functioning by engaging many service providers and balancing their assigned requests to provide them with incentives to stay in the system. This problem is important because of its many related societal applications in the on-demand and sharing economy (e.g. on-demand ride hailing services, on-demand food delivery, etc). Challenges of this problem include the need to satisfy many conflicting requirements for the broker, consumers and service providers and the high computational complexity for a large number of consumers and service providers. Related work in spatial crowdsourcing and ridesharing has mainly focused on maximizing the number of matched requests and minimizing travel cost, but did not consider the importance of engaging more service providers and balancing their assignments, which could become a priority when the available supply exceeds the demand. In this work, we propose a new category of service provider centric heuristics for meeting these conflicting requirements. We evaluated our algorithms using synthetic datasets with real-world characteristics. Experimental results show that our proposed heuristics can achieve a larger number of matched requests when supply and demand are balanced. They also engage a larger number of service providers with a more balanced provider assignment when the available supply greatly exceeds demand.

CCS Concepts

•Information systems → Location based services; Geographic information systems;

Keywords

supply-demand matching; on-demand services; on-demand spatial service broker

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

IWCTS'16, October 31-November 03 2016, Burlingame, CA, USA

© 2016 ACM. ISBN 978-1-4503-4577-4/16/10...\$15.00

DOI: <http://dx.doi.org/10.1145/3003965.3003974>

1. INTRODUCTION

The increasing proliferation of mobile technologies such as smart phones has led to non-traditional business models and the appearance of new marketplaces as evidenced by the emerging on-demand and sharing economy. Today, the on-demand economy attracts millions of consumers annually and over \$50 billion in spending [5]. Success stories include many on-demand ride hailing services; food delivery services such as Instacart [8]; and other types of on-demand services such as bike rental (e.g. Spinlister [13]), home services (e.g. TaskRabbit [15]) and beauty services (e.g. StyleBee [14]), etc. These new marketplaces benefit consumers by increasing their access to services while reducing investment costs in resources and infrastructure (e.g. road and parking space) and other societal costs (e.g. greenhouse gas emissions, fuel consumption). These benefits result from the collaborative consumption nature of these systems as well as the centralized management of resources which allows meeting larger demand from consumers through efficient management of the available supply.

In this paper we investigate an on-demand spatial service broker for identifying commerce opportunities between mobile consumers on the road and registered service providers (e.g. restaurants, grocery stores, hair salons, etc). Given a set of service providers defined by their locations and service rates, a set of dynamically arriving consumer service requests and a number of required propositions K , the on-demand spatial service broker matches each consumer request to K service provider propositions and presents the corresponding estimated waiting time for each proposition. In this context, a spatial service refers to a category of services where the locations of consumers and service providers are critical in the matching process since consumers need to drive or walk to get the service. The service provider propositions assigned to a consumer must meet consumer constraints such as maximum acceptable travel distance and maximum time before service, while not violating the providers' service rate constraints. The goal of the broker is to maximize the number of matched requests. In addition, the broker has to keep the "eco-system" functioning by engaging as many service providers as possible and balancing the matched requests among them. For instance, in an on-demand ride hailing service, service providers (i.e., drivers) that are consistently not assigned any rides may eventually decide to leave the system or switch to another broker resulting in overall service degradation. The importance of this balance increases and becomes a high priority during periods of unbalanced supply and demand, particularly when

the available supply greatly exceeds demand and engaging many service providers becomes more challenging.

Challenges: Designing an on-demand spatial service broker is challenging for the following reasons: First, the broker needs to satisfy many conflicting requirements. For instance, the broker aims to maximize the number of matched requests for maximizing its profit while simultaneously keeping the eco-system functioning by balancing the assignments among service providers. Moreover, the broker needs to satisfy the conflicting requirements of consumers (e.g. minimize their travel costs and wait times) and service providers (e.g. maximize the number of each provider’s assigned requests). Second, the relationship between available supply and demand in on-demand business models, as depicted by the supply-demand ratio, exhibits spatio-temporal heterogeneity. Hence, a matching strategy that works well for a given time and/or location may not work as well for other times or locations with different supply-demand ratios. Third, given a number of consumer requests and a list of candidate propositions that satisfy the constraints of each request, finding the set of K -propositions that maximize the number of matched requests is an NP-hard problem.

Related Work and their Limitations: The related work for this problem falls into two categories. The first category is spatial crowdsourcing [11, 10, 6, 16]. In a spatial crowdsourcing system with a central server, the server dynamically receives information about available tasks as well as requests from workers who are ready to work. Each worker specifies his constraints such as the region in which he can accept tasks and the maximum number of tasks he is willing to perform. The server then assigns the tasks to the workers with the goal of maximizing the number of assigned tasks. In addition, the server tries to minimize the distance traveled by the workers by giving a higher priority to task-worker assignments with the “Least Travel Cost”. In [11], another idea was proposed for prioritizing the tasks to be assigned, namely, the “Least Location Entropy Priority”. In this method, the entropy of each task location is computed based on the number of worker visits to that location. Tasks with lower entropy indicate fewer consumer visits (i.e. tasks in worker-sparse areas) and are given higher priority since tasks in areas with higher worker densities are more likely to be assigned in the future. This allows the broker to maximize the number of tasks assigned in the future when more workers arrive. The second category of related work is online ridesharing systems [7, 4, 12, 2] in which trip requests are dynamically matched to vehicles while satisfying the waiting and service time constraints of the passenger and the maximum detour distance specified by the drivers. In these systems, an incoming trip request is matched to the vehicle that adds the “Least Travel Cost”.

The related work discussed above has mainly focused on maximizing the number of matched requests (or tasks) and minimizing the total travel time. These works do not consider the need to keep the eco-system functioning by maximizing the number of engaged service providers (aka workers in spatial crowdsourcing systems or drivers in ridesharing systems) and balancing their assigned requests. In our work, we propose several service provider-centric heuristics and we show that the proposed heuristics can engage more service providers and achieve a better balance for the assigned requests among them.

Contributions: This paper makes the following contribu-

tions: (1) We formally define the problem of On-demand Spatial Service Propositions. (2) We propose a new category of service provider-centric heuristics for increasing the number of engaged service providers while meeting the conflicting requirements of the broker, consumers and service providers. (3) We evaluated our heuristics using synthetic datasets with real-world characteristics. Results show that our proposed heuristics can achieve a larger number of matched requests for balanced supply and demand scenarios and a larger number of matched service providers with a more balanced provider assignment particularly when the available supply exceeds demand.

Scope: This paper focuses on the problem of designing an on-demand spatial service broker that matches incoming consumer requests with service provider propositions while satisfying consumer and supply constraints and keeping the eco-system alive. However, learning consumer preferences for different service providers (e.g. preferred stores, meals, or cuisines) and incorporating them into the matching algorithm is considered outside the scope of this paper. We also assume that the broker and service providers are driven by the number of matched service requests (i.e. transactions) which is a proxy for their profit and that profits are fixed for all service requests. Additionally, this paper does not model the temporal evolution in the behavior of consumers and service providers which may be critical to the robustness and survivability of the eco-system.

2. PROBLEM DEFINITION

The problem of On-demand Spatial Service Propositions (OSSP) can be expressed as follows:

Given:

1. A set of service providers. Each provider p is defined using its location latitude and longitude coordinates (p_{lat} , p_{long}) and a time series representing the service rate per hour of this provider for a typical day.
2. A set R of consumer requests arriving dynamically. Each request $r_i \in R$ specifies the consumer current location l_i , maximum acceptable travel distance d_i , and maximum acceptable waiting time before service w_i .
3. A number of required propositions K
4. A timeout interval length $t_{timeout}$

Find: K service provider propositions and the corresponding estimated waiting times for each $r_i \in R$

Objective: Maximize the number of matched requests

Constraints:

1. Service provider propositions matched to a consumer request should satisfy the consumer’s maximum acceptable waiting time before service constraint and the maximum acceptable travel distance constraint.
2. Propositions from each service provider should not violate the provider’s service rate.

In this problem, for each consumer request, the broker finds K service propositions which satisfy the consumer maximum travel distance and waiting time constraints and returns them to the consumer with their corresponding estimated waiting times. The estimated waiting time includes both the time spent driving and the waiting time at the service provider. If no K propositions satisfying the consumer constraints were found, the consumer request is not matched. The fourth input is a timeout interval length $t_{timeout}$. After a consumer receives K service provider propositions, he/she needs to select a proposition within $t_{timeout}$

Figure 1: Example input for the OSSP problem

Table 1: Possible outputs for problem in Figure 1

Possible output	# matched requests	# matched providers	Aggregate travel cost
t=0: (C1, P1) t=1: (C2, P1)	2	1	4 miles
t=0: (C1, P2) t=1: (C2, P2)	2	1	6 miles
t=0: (C1, P1) t=1: (C2, P2)	2	2	5 miles

to guarantee the estimated waiting times presented. While matching consumer requests, the broker has to also respect the service rate constraint for each service provider.

Other considerations: The main objective of the broker is to maximize the number of matched requests in order to increase the number of transactions and maximize profits. In addition, the broker has two secondary objectives: First, the broker aims to match as many service providers as possible and to balance their assigned requests as evenly as possible. This objective ties to the application domain requirement of keeping the “eco-system” functioning by incentivizing providers to stay in the system. Second, responses to consumer requests need to be provided in real-time due to the nature of on-demand services.

Problem Example: Figure 1 shows an example input for the OSSP problem. The weights shown on the edges of the road network refer to the travel distance (in miles) along the edge. Two service providers P_1 and P_2 are located at nodes C and D respectively. Each service provider is assumed to have a service rate of 12 requests/hr for the current hour, which corresponds to a mean service time of 5 min. For simplicity, the number of required propositions (K) is set to 1 and the timeout interval length is also set to 1 min. At the first time instant ($t=0$), a consumer C_1 , located at node A , submits a request to the broker to receive service propositions. At $t=1$, consumer C_2 submits a request at node F . The maximum acceptable travel distance and maximum acceptable waiting time before service constraints for each consumer request are 3 miles and 10 min respectively.

Since the mean service time per request for both providers is 5 min and each consumer can wait up to 10 min, we can see that both service providers meet the maximum acceptable waiting time constraints for both consumers. Similarly, P_1 is located at a distance of 2 miles from each consumer, while P_2 is located at a distance of 3 miles. Therefore, both providers also satisfy the maximum travel distance constraints of both C_1 and C_2 . Table 1 shows three possible output solutions for this input. In the first row, consumer C_1 is assigned to provider P_1 at time $t=0$. At $t=1$, C_2 arrives and is also assigned to P_1 . This solution results in a total of two matched requests, with only one service provider being matched. The total travel cost is 4 miles. The second row shows another possible solution where both consumers are assigned to the second service provider, resulting in a similar number of matched requests and matched service providers, but with

a higher travel cost (i.e., 6). Finally, the third row shows a solution where C_1 is assigned to P_1 . Then at $t=1$, C_2 is assigned to P_2 . This solution results in both providers being matched with a total travel cost of 5.

The first row solution is what we get when we apply the “Least Travel Cost” heuristic to this problem, since each consumer is assigned to the nearest service provider that satisfies his constraints. Applying the “Least Location Entropy Priority” strategy means that consumers in provider-sparse areas are given higher priorities. For this example, each consumer has two service providers, P_1 and P_2 , in his neighborhood since both providers fall within the consumer’s maximum acceptable travel distance. Hence, the location entropy of each consumer is equal to $Entropy(loc_{C_i}) = -\sum_{P_i \in P_{C_i}} p_i(P_i) \log p_i(P_i)$, where P_{C_i} is the set of providers in the neighborhood of C_i and $p_i(P_i) = \frac{1}{|P_{C_i}|}$. Hence, both consumers have a location entropy equal to $-\left[\frac{1}{2} \log_2\left(\frac{1}{2}\right) + \frac{1}{2} \log_2\left(\frac{1}{2}\right)\right] = 1$. However, since for this example, only one consumer is available at every time instant, this entropy will not be used for prioritizing consumers and thus each consumer can end up being assigned to any of the two providers. Consequently, for this example, the least location entropy priority can lead to any of the three outputs presented in Table 1. Thus, we can see that both the “Least Travel Cost” and “Least Location Entropy Priority” discussed in related work are consumer or broker-centric; they do not aim to increase provider engagement or balance providers’ assignments.

For a set of consumer requests arriving at time t , and a set of service providers satisfying the travel distance and waiting time constraints of these consumers, identifying the set of K propositions for each consumer that maximizes the number of matched consumers is a challenging problem since a broker could receive a large number of requests with a large number of candidate providers for each request. In [3], we showed this $OSSP_t$ problem (i.e. the OSSP problem for every time instant t) to be NP-hard by reduction from the maximum 3-dimensional matching problem. The proof is eliminated here due to space limitations.

3. PROPOSED APPROACH

Since solving the On-demand Spatial Service Propositions problem at every time instant is NP-hard, as shown in [3], we propose several heuristics (presented next in this section) for efficiently matching consumer requests to service providers in real-time. Each heuristic is applied using a greedy approach whose outline is shown in Algorithm 1. The algorithm starts by finding a set of candidate propositions for each available consumer request (lines 1 to 4). To efficiently identify the candidates for a consumer, we use a spatial grid index for the locations of all service providers. Using this index, we retrieve all grid cells within the maximum acceptable travel distance of a consumer. We then run Dijkstra’s shortest path algorithm for one time to calculate the distances from this consumer to all the retrieved service providers. If a provider’s next available service time satisfies the consumer’s maximum acceptable waiting, the provider is added as a candidate proposition for that consumer. In lines 5 to 10 each candidate proposition gets a score that is calculated based on one of the proposed heuristics. Then, a priority queue stores the candidate propositions in the order of their scores. Finally, lines 11 to 18 iterate through each

Algorithm 1 Consumer-Provider Matching Algorithm

Input:

- (1) A set of consumer requests (*arrivalEvents*)
- (2) List of service providers
- (3) Matching heuristic, e.g. LAF, LCF

Output:

A set of matched propositions for each consumer

Algorithm:

```
1: for each consumer  $c$  in arrivalEvents do
2:    $nearProviders \leftarrow$  Find providers satisfying consumer's  $d_c$ 
3:    $candidatePropositions[c] \leftarrow$  Find providers in
    $nearProviders$  satisfying consumer's  $w_c$ 
4: end for
5: for each consumer  $c$  in arrivalEvents do
6:   for each proposition in  $candidatePropositions[c]$  do
7:      $proposition.score \leftarrow$  Calculate score based on input
       heuristic
8:      $priorityQueue[c] \leftarrow proposition$ 
9:   end for
10: end for
11: for each consumer  $c$  in arrivalEvents do
12:    $matchedPropositions[c] \leftarrow$  Find K propositions in
    $priorityQueue[c]$  that still satisfy consumer's  $w_c$ , otherwise
   return an empty set.
13:   if  $matchedPropositions[c]$  not empty then
14:     for each proposition prop in  $matchedPropositions[c]$  do
15:        $prop.provider.nextAvailableServiceTime =$ 
        $prop.provider.nextAvailableServiceTime +$ 
        $prop.provider.serviceTime$ 
16:     end for
17:   end if
18: end for
19: return  $matchedPropositions$ 
```

consumer request, and select the first K propositions from its priority queue that satisfy his maximum acceptable waiting time constraint. The waiting time constraint is rechecked at line 12 since the next available service time of a provider in a candidate proposition may violate the consumer's wait time constraint if that provider has been already matched to another consumer. Once a consumer is matched to K propositions, the next available service time for the providers of these propositions is updated based on the providers' current service rates to reserve the service time. The algorithm then returns the set of matched propositions for all matched consumers.

Next, we describe our two proposed service provider-centric heuristics, namely, the "Least Accepted First" and "Least Appearance as Candidate First" heuristics:

Least Accepted First (LAF): This heuristic aims to increase the number of service providers matched by the broker. This is achieved by giving higher priority to the service providers that have received the fewest number of acceptances by consumers so far. Hence, each proposition from a provider is assigned a score equal to the number of accepted requests from that provider. Then, when matching a consumer request, candidate propositions from this consumer are sorted in increasing order of their assigned scores, allowing service providers who have fewer prior acceptances to be matched first. For instance, consider the input shown in Figure 1, where each of the providers P_1 and P_2 has a service rate of 12 requests/hr (i.e., 5 min/request) and each consumer has a maximum acceptable travel distance and waiting time of 3 miles and 10 min respectively. At $t=0$, consumer C_1 can be matched to either P_1 or P_2 since both satisfy his travel distance and waiting time constraints. Suppose C_1 is matched to P_1 . C_1 will need to accept the proposition before its timeout at $t=1$. Since K is set to 1, P_1 is the only proposed service provider to C_1 and thus we can simply assume that C_1 accepts this service provider propo-

sition. At $t=1$, the request of C_2 arrives and again both P_1 and P_2 are considered as candidate propositions. However, since P_1 has already been accepted by C_1 , its score is set to 1, while P_2 has never been accepted before, making its score equal to zero. Thus P_2 will be given a higher priority and C_2 will be matched to P_2 . Therefore, using LAF, we get the solution shown in the last row of Table 1, which has the largest number of matched providers.

Least Appearance As Candidate First (LCF): This heuristic also aims to increase the number of matched service providers. However, instead of prioritizing service providers with the fewest number of previous acceptances, the LCF heuristic gives a higher priority to providers that have been least considered as a candidate proposition by consumers. A service provider is considered as a candidate proposition if it lies within the maximum acceptable travel distance of a consumer and its next available service time satisfies the consumer's maximum acceptable waiting time before service. Hence, the intuition behind LCF is to favor service providers in regions with fewer originating requests (since these providers have a smaller probability of being matched to consumers) and to also favor service providers with longer service times since such providers are more likely to have difficulty meeting the maximum acceptable waiting time constraints of many consumers and thus have a smaller probability of being matched. Hence, when using LCF, each service provider proposition is assigned a score equal to the number of appearances of this service provider as a candidate proposition. Then, propositions are sorted in increasing order of their scores, allowing service providers with fewest appearances as a candidate to be matched first.

4. EXPERIMENTAL EVALUATION

The goal of our experiments was to evaluate the performance of the proposed heuristics compared to the related work under different variations of the supply-demand ratio. We tested four heuristics: (a) Least Travel Cost (LTC), (b) Least Location Entropy Priority (LLEP), (c) Least Accepted First (LAF), and (d) Least Appearance As Candidate First (LCF). A grid index was used to divide the spatial region into grid cells for calculating the location entropy. Consumers in the same grid cell of a provider were considered to be in the neighborhood of that provider.

4.1 Discrete-Event Simulation Framework

We simulate the interactions between arriving consumer requests and the on-demand spatial service broker using a discrete-event simulation framework [9]. The simulation starts by initializing a simulation clock. A priority queue stores the simulation events in the order of their arrival times. All consumer arrival events are generated and stored in the queue. Each event carries the arrival time of the consumer's request, maximum acceptable travel distance and maximum acceptable time before service. The queue also carries another type of event, namely, proposition acceptance events when a user accepts a service provider proposition from among the propositions suggested by the broker. The simulation loop then starts and continues until the queue is empty. For each iteration, the simulation dequeues all proposition acceptance and consumer arrival events with arrival times before or at the current clock reading. Proposition acceptance events are handled first. For each of these events, the next available service time of all

the consumer’s unaccepted prepositions is updated to release the service time units reserved for that consumer. Then, the heuristic-based greedy algorithm (presented in Section 3) is called for matching the available consumer requests to service providers. For every matched request, a corresponding acceptance event is then added into the queue to simulate a consumer accepting a suggested proposition. The time of the event is set to a randomly selected time between 0 and $t_{timeout}$ and the accepted proposition is also randomly chosen from the K propositions matched to that consumer. Finally, the simulation performance metrics are updated and the clock is advanced to the next time instant.

4.2 Experimental Design and Dataset

We generated synthetic datasets with real-world characteristics as captured by real service provider locations and the use of real-world population data in the city of Minneapolis, MN. Population data was used for generating the origin locations of consumer’s requests. The dataset consisted of 120 restaurants (i.e. service providers) in the city of Minneapolis.

The following procedure was used by the simulator to generate the supply and demand: First, providers’ service rates were generated using a random number that was uniformly distributed over the range $[r_{min}, r_{max}]$. Then, given a value for the supply-demand ratio sdr to simulate, we generated a number of consumer requests per hour such that $\frac{\text{total supply per hour (i.e. } \sum_{i=1}^{|P|} r_i)}{\text{total number of requests in that hour}} = sdr$, where $|P|$ is the set of all service providers. To select the origin locations of the requests, the number of requests generated at each node in the network was proportional to the ratio of that node population to the total city population. Experiments simulated 10 hours of operation assuming 5 lunch hours and 5 dinner hours for each restaurant. The locations of the requests during the lunch hours were generated in proportion to the day population of the nodes, while the locations of the dinner requests were proportional to the night population. Consumers’ maximum acceptable travel distances and maximum acceptable waiting times before service were also generated using random numbers that were uniformly distributed over the ranges $[d_{min}, d_{max}]$ and $[w_{min}, w_{max}]$ respectively. The arrival time of each request generated in a given hour was also uniformly distributed over that hour duration. The experiments were performed using the road network provided by the Minnesota Department of Transportation [1], and U.S. Census population data.

The default parameter values were as follows: $K = 3$, $t_{timeout} = 2$ min, $r_{min} = 5$ requests/hr, $r_{max} = 15$ requests/hr, $d_{min} = 4000$ m and $d_{max} = 12000$ m, $w_{min} = 5$ min, $w_{max} = 15$ min and $l_G = 100$ m, unless stated otherwise. Experiments were run on a machine with an Intel Xeon Quad Core 3.00 GHz processor and 64 GB RAM.

4.3 Experimental Results

To evaluate the effect of the supply-demand ratio on the performance of the different heuristics we ran our simulation for the sdr values of 0.1 (supply $\ll$ demand), 1 (supply $\approx$ demand), 10 (supply $\gg$ demand) and 50 (supply extremely exceeds demand). For each value, the generated supply was fixed and the number of generated requests varied as shown in Table 2.

Percentage of matched requests: Figure 2a shows the effect of sdr on the percentage of matched requests. At small

Table 2: Synthetic dataset details. Total supply is fixed across all sdr values.

Supply-demand ratio	No. of consumer requests	No. service providers	Avg. No. of requests per provider
0.1	121600	120	1013.3
1	12160	120	101.3
10	1220	120	10.2
50	240	120	2

sdr value (i.e. 0.1), most of the requests could not be matched since the demand is much larger than the available supply. As sdr increases the percentage of matched requests increases. For balanced supply and demand ($sdr=1$), the LAF and LCF heuristics outperform all other heuristics. The reason is that by engaging more service providers these heuristics are able to make the most efficient use of the available supply and thus can serve a larger number of requests. Although at $sdr=1$, all requests can be theoretically matched, if consumers provide small acceptable travel distance and waiting time as constraints, the broker might still not be able to match their requests. Below LAF and LCF, we can see the broker-centric LLEP heuristic since its focus was to maximize the future match size. For larger sdr values, all heuristics had a high and comparable performance since supply greatly exceeds demand and many requests can be easily matched.

Percentage of matched providers: Figure 2b shows the effect of sdr on the percentage of matched providers. It can be seen that as supply exceeds demand (i.e. larger sdr values), the percentage of matched providers decreases for almost all heuristics except for LAF, which is able to engage 100% of the service providers even with a demand that is much lower than the available supply. However, due to the small available demand, these matched providers will only be using a small service capacity and may thus reduce their labor cost (e.g. by using a smaller number of waiters).

Average no. of assigned requests per provider: Figure 2c shows the effect of sdr on the average number of assigned requests per provider. As sdr increases, a fixed supply is maintained but the number of generated consumer requests decreases. Thus, the average number of assigned requests per provider also decreases. At balanced supply and demand, we notice that LAF and LCF assign a larger number of requests per provider since they aim to balance provider selection based on providers’ previous number of requests. Another observation is that at $sdr = 0.1$, the average number of assigned requests per provider for all heuristics is much lower than the average number shown in Table 2 for $sdr = 0.1$. The reason is that at $sdr = 0.1$ the demand greatly exceeds the supply and thus many requests have not been matched. However, as sdr increases, most of the demand is being met and thus the average number of assigned requests per provider for larger sdr values exceeds the average values in Table 2. The reason is that, for this experiment, a single request is matched with $K=3$ providers rather than a single provider which increases the average number of matched requests per provider.

Standard deviation of assigned requests per provider: Figure 2d shows the effect of sdr on the standard deviation of the assigned requests per provider. At $sdr = 0.1$, all heuristics had a large and comparable standard deviation value due to the large demand which can be concentrated in more populated regions resulting in some providers being assigned much more often than other providers. As

Figure 2: Effect of supply-demand ratio sdr (by fixing supply and varying demand). The number of total consumer requests (i.e. demand) corresponding to each sdr value is shown in Table 2.

sdr increases, we notice a consistently smaller standard deviation with LAF than with other heuristics due to its focus on balancing assignments between different providers. At $sdr = 1$, LCF also achieves a better balance compared to the other heuristics since it gives higher priority to service providers that have been least considered as candidate matches. However, when supply greatly exceeds demand (i.e. $sdr = 10$ and $sdr = 50$), we can see that LCF does not achieve a good balance as compared to the other heuristics since with a small number of requests (i.e. small demand) many providers may have been considered as candidates before but still have never been matched due to the limited demand. Hence, those providers will always be missed by LCF when matching consumer requests. We can also see that as sdr increases and consequently the number of requests decreases (assuming fixed supply), the standard deviation generally decreases. This is because with a small number of requests, only a few requests, if any, are assigned to each provider and thus only a small difference can occur between these small number of assigned requests.

5. CONCLUSION AND FUTURE WORK

This work explored the problem of On-demand Spatial Service Propositions (OSSP) where an on-demand spatial service broker aims to maximize the number of matched consumer requests while keeping the eco-system functioning by engaging a large number of service providers and balancing their consumer assignments. We proposed a heuristic-based greedy matching approach with a new category of service provider-centric heuristics for increasing the number of matched service providers while meeting the conflicting requirements of the broker, consumers and service providers. We evaluated our approach using synthetic datasets with real-world characteristics. Experimental results show that our “Least Accepted First” and “Least Candidate First” heuristics achieve the largest number of matched consumer requests when the available supply and demand are balanced. For scenarios where the available supply exceeds demand, the “Least Accepted First” heuristic results in the largest number of matched service providers and achieves the best balance in the number of assigned requests per provider, thus allowing the eco-system to stay alive during periods of high supply-demand ratio. In the future, we plan to extend OSSP to allow mobile service providers and incorporate consumer ratings into the matching process. We also plan to explore how the broker can adapt to spatio-temporal variations in market conditions and the evolving behavior of

consumers and service providers. Finally, we plan to model the variation in profit across different types of consumer requests.

6. ACKNOWLEDGMENTS

This material is based upon work supported by FORD University Research Program (URP), the National Science Foundation under Grant No. 1029711, IIS-1320580, 0940818 and IIS-1218168, the USDOD under Grant No. HM1582-08-1-0017 and HM0210-13-1-0005, the OVRP U-Spatial and Minnesota Supercomputing Institute (MSI) at the University of Minnesota (www.msi.umn.edu).

7. REFERENCES

- [1] Minnesota mndot interactive gis basemap. <http://www.dot.state.mn.us/maps/gdma/gis-data.html>. Minnesota Department of Transportation, 2016.
- [2] N. A. Agatz et al. Dynamic ride-sharing: A simulation study in metro atlanta. *Transportation Research Part B: Methodological*, 45(9):1450–1464, 2011.
- [3] R. Y. Ali et al. Supply-Demand Ratio and On-Demand Spatial Service Brokers. Technical report, University of Minnesota, Department of Computer Science and Engineering, 09 2016.
- [4] B. Cici et al. Designing an on-line ride-sharing system. In *Proc. of the 23rd SIGSPATIAL Intl. Conf. on Advances in Geographic Information Systems*, page 60. ACM, 2015.
- [5] C. Colby and K. Bell. Harvard Business Review. The On-Demand Economy Is Growing, and Not Just for the Young and Wealthy. <http://goo.gl/LyOIHI>, Apr. 2016.
- [6] D. Deng et al. Task matching and scheduling for multiple workers in spatial crowdsourcing. In *Proc. of the 23rd SIGSPATIAL Intl. Conf. on Advances in Geographic Information Systems*, page 21. ACM, 2015.
- [7] Y. Huang et al. Large scale real-time ridesharing with service guarantee on road networks. *Proc. of the VLDB Endowment*, 7(14):2017–2028, 2014.
- [8] Instacart. <https://www.instacart.com/>.
- [9] R. K. Jain. *Art of Computer Systems Performance Analysis: Techniques for Experimental Design Measurements, Simulation, and Modeling*. John Wiley, 1991.
- [10] L. Kazemi et al. Geotrucrowd: trustworthy query answering with spatial crowdsourcing. In *Proc. of the 21st SIGSPATIAL Intl. Conf. on Advances in Geographic Information Systems*, pages 314–323. ACM, 2013.
- [11] L. Kazemi and C. Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *Proc. of the 20th SIGSPATIAL Intl. Conference on Advances in Geographic Information Systems*, pages 189–198. ACM, 2012.
- [12] M. Ota et al. A scalable approach for data-driven taxi ride-sharing simulation. In *IEEE Intl. Conf. on Big Data*, pages 888–897. IEEE, 2015.
- [13] Spinlister. <https://www.spinlister.com/>.
- [14] StyleBee. <https://www.stylebee.com/>.
- [15] TaskRabbit. <https://www.taskrabbit.com/>.
- [16] H. To et al. A server-assigned spatial crowdsourcing framework. *ACM Trans. on Spatial Algorithms and Systems*, 1(1):2, 2015.