TR 16-033

Supply-Demand Ratio and On-Demand Spatial Service Brokers

Reem Y. Ali, Emre Eftelioglu, Shashi Shekhar, Shounak Athavale, Eric Marsman

September 8, 2016

Supply-Demand Ratio and On-Demand Spatial Service Brokers

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. In addition, the broker has to keep the "eco-system" functioning not only by meeting consumer requirements, but also 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 in addition to the problem's computational complexity which is shown to be NP-hard. Related work has mainly focused on maximizing the number of matched requests (or tasks) 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 highly exceeds the demand. In this work, we propose several matching heuristics for meeting these conflicting requirements, including a new category of service provider centric heuristics. We employed a discrete-event simulation framework and evaluated our algorithms using synthetic datasets with real-world characteristics. Experimental results show that the proposed heuristics can help engage more service providers and balance their assignments while achieving a similar or better number of matched requests. We also show that the matching heuristics have different dominance zones that vary with the supply-demand ratio and that a supply-demand ratio aware broker is needed to select the best matching policy.

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 [6]. Success stories in the on-demand marketplace include many on-demand ride hailing services; food delivery services such as Instacart [13]; and other types of on-demand services such as bike rental (e.g. Spinlister [18]), home services (e.g. TaskRabbit [20]) and beauty services (e.g. StyleBee [19]), etc. The benefits of these new marketplaces include the ability to improve consumer welfare by increasing access to services while reducing investments in resources and infrastructure (e.g. road and parking space) and reducing 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 supply rates, a set of dynamically arriving consumer service requests and a number of required propositions K, the on-demand spatial service broker is responsible for matching each consumer request to K service provider propositions and presenting 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 service providers supply rate constraints. The goal of the broker is to maximize the number of matched requests. In addition, the broker has to keep the 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, drivers (aka service providers) that are consistently not assigned any rides may eventually decide to leave the system or switch to another on-demand service 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 highly exceeds the demand and thus 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. minimizing their travel costs and waiting times) and service providers (e.g. maximizing the number of their assigned requests). Second, the relationship between the 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 as shown in Section 2.2.

Related Work and their Limitations: The related work for this problem falls into two categories. The first category is spatial crowdsourcing [16, 15, 7, 21]. 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 [16], another idea was proposed for prioritizing the tasks to be assigned, namely, the "Least Location Entropy Priority". In this method, a higher priority is given to tasks in worker-sparse areas 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 [12, 5, 17, 2, 8] 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 for keeping 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, in addition to other broker and consumer-centric heuristics, and we show that the best performing heuristic varies with the variation of the supply-demand ratio.

Contributions: This paper makes the following contributions:

1. We formally define the problem of On-demand Spatial Service Propositions.
2. We propose several matching heuristics for meeting the conflicting requirements of the broker, consumers and service providers including a new category of service provider-centric policies for increasing the number of engaged service providers.
3. We employ a discrete-event simulation framework for modeling the interactions between the consumers and the broker and provide an experimental evaluation using synthetic datasets with real-world characteristics. Our results show that our proposed heuristics can result in 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 the available demand. We also show that the performance and dominance zones of the different matching heuristics vary with the supply-demand ratio and that a supply-demand ratio-aware broker is needed to select the best matching policy.

Scope and Outline: 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.

The rest of the paper is organized as follows: Section 2 presents the formal definition for the On-demand Spatial Service Propositions problem and an analysis of the problem complexity. Section 3 describes our simulation framework and presents our proposed heuristics using a greedy matching approach. The experimental evaluation is presented in Section 4. Section 5 discusses other related work and alternative approaches. Finally, Section 6 concludes the paper and discusses future work.

2. PROBLEM DEFINITION

In this section we present a formal definition for the On-demand Spatial Service Propositions problem followed by an analysis for the problem complexity.

2.1 Problem Statement

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, direction of motion θ_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 ac-

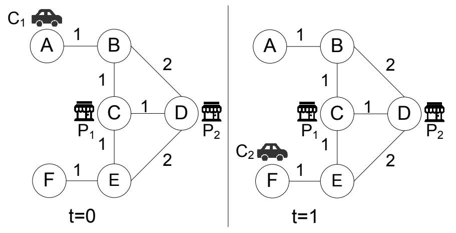


Figure 1: Example input for the OSSP problem

ceptable travel distance constraint along his direction of motion.
2. Propositions from each service provider should not violate the provider's supply 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 in 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. Each consumer also specifies a direction of motion $\theta \in \{N, S, E, W, NE, NW, SE, SW\}$ and the broker only proposes service providers that are along the consumer's direction of motion (i.e. θ) or within a 45° from θ in both directions. 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 $t_{timeout}$ to guarantee the estimated waiting times presented. While matching consumer requests, the broker has to also respect the supply 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 maximize its profit by increasing the number of transactions. In addition, the broker has other 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 and moving east towards node B, submits a request to the broker to receive service propositions. At t=1, consumer C_2 submits a request at node F while moving east towards node E. 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 given 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 while only one service provider is 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 another possible 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.

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

By applying the "Least Travel Cost" heuristic to this problem, we get the solution in this first row, 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. Hence, the location entropy of each consumer is equal to $Entropy(l_{Ci}) = -\sum_{P_i \in P_{C_i}} p_l(P_i) log\ p_l(P_i)$, where P_{C_i} is set of providers in the neighborhood of C_i and $p_l(P_i) = \frac{1}{|P_{C_i}|}$. Hence, both consumers have a location entropy equal to $-[\frac{1}{2}log(\frac{1}{2}) + \frac{1}{2}log(\frac{1}{2})] = 0.3$. 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. We may also consider applying the "Least Location Entropy Priority" for prioritizing the service providers when assigning them to each consumer. In that case, providers in consumer-sparse areas are given higher priority and are assigned first. For this example, at t=0 both service providers will have only one consumer visit in their neighborhood so far (i.e. C_1). Therefore, both P_1 and P_2 will have the same location entropy ($= -\frac{1}{1}log(\frac{1}{1}) = 0$) and either of them can be assigned to C_1. Similarly, when consumer C_2 submits a request at t=1, both P_1 and P_2 will have the same number of consumer prior visits (=2) and their location entropies will be equal to $-[\frac{1}{2}log(\frac{1}{2})+\frac{1}{2}log(\frac{1}{2})] = 0.3$ so either of them can be assigned to C_2. Consequently, for this example, the least location entropy priority, when applied for prioritizing either consumers or producers, 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 more consumer or broker-centric, but do not aim to increase provider engagement or balance providers' assignments.

Table 2: Potential K-propositions for 3 consumers with 2 service providers and K=2

Consumer	Potential K-propositions
C_1	$\{(C_1, <P_1P_2>), (C_1, < P_1P_3 >), (C_1, < P_2P_3>)\}$
C_2	$\{(C_1, < P_1P_2 >), (C_1, < P_1P_3 >), (C_1, < P_2P_3>)\}$
C_3	$\{(C_1, < P_1P_2 >), (C_1, < P_1P_3 >), (C_1, < P_2P_3>)\}$

2.2 Complexity Analysis

This section presents a proof of the NP-hardness of the OSSP problem. Consider an example with 3 consumer requests arriving at time t=0, and three service providers P_1, P_2, P_2 whose supply capacity satisfies the maximum acceptable travel distance and maximum acceptable waiting time before service for all three consumers. Let the number of required propositions per consumer (K) equal 2. Table 2 shows the potential K-propositions for each consumer.

Identifying the subset of K-propositions from Table 2 that maximizes the number of matched consumers is a challenging problem since an on-demand service broker could receive a large number of requests with a large number of candidate providers for each request. Therefore, we now prove that this $OSSP_t$ problem (i.e. the OSSP problem for every time instant t) is NP-hard by reduction from the maximum 3-dimensional matching problem. Our proof follows the proof of NP-hardness for the spatial crowdsourcing problem presented in [15].

Definition 1. **Maximum 3-dimensional Matching Problem** Let X, Y, and Z be finite, disjoint sets, and let T be a subset of X × Y × Z. That is, T consists of triples (x, y, z) such that x ∈ X, y ∈ Y, and z ∈ Z. Now M ⊆ T is a 3-dimensional matching if for any two distinct triples (x1, y1, z1) ∈ M and (x2, y2, z2) ∈ M, we have x1 ≠ x2, y1 ≠ y2, and z1 ≠ z2. The maximum 3-dimensional matching problem is to find a 3-dimensional matching M ⊆ T that maximizes the number of triples in M (i.e. $\mid M \mid$).

Now we define a special problem instance of $OSSP_t$, namely $OSSP_{t,1}$, where we assume that every service provider can only be assigned to one consumer at time t. We start by proving that $OSSP_{t,1}$ is an NP-hard problem.

Lemma 1. The $OSSP_{t,1}$ problem is NP-hard.

Proof. To prove this lemma, we first provide a reduction of the Maximum 3-dimensional matching problem to the $OSSP_{t,1}$ problem in polynomial time. That is, given an instance I_M of the maximum 3-dimensional matching problem, we show that there is an instance I_O of the $OSSP_{t,1}$ problem where the solution to I_O can be converted into a solution to I_M in polynomial time. Now consider an instance I_M where each of the sets X, Y and Z have n elements. Let T be a subset of X x Y x Z. Then, the solution to I_M involves finding a set M ⊆ T such that $\mid M \mid$ is maximized. Next, we describe a mapping from I_M to I_O. To achieve this mapping, for every element in X, we create a consumer request and for every element in Y and Z, we create a service provider such that we now have n consumers and $2n$ service providers. Every consumer has a set of potential K-propositions M_i where $M = \cup_{i=1}^{|C|} M_i$ and every potential K-proposition in $\cup_{i=1}^{|C|} M_i$ will be in the form $(C_x, < P_y P_z >)$, where 0 $<x \leq$ n, 0 $<y \leq$ n and n $<z \leq$ 2n. To solve I_O, we need to find the set A ⊆ M such that A is the largest 3-dimensional matching (i.e. a set of maximum cardinality in which no two K-propositions contradict). This means that for every 2 K-propositions in A, say $(C_{x_1}, < P_{y_1} P_{z_1} >)$ and $(C_{x_2}, < P_{y_2} P_{z_2} >)$, $C_{x_1} \neq C_{x_2}$, $P_{y_1} \neq P_{y_2}$ and $P_{z_1} \neq P_{z_2}$. Thus, if A is the solution to I_O, then the solution to I_M is the set M with maximum cardinality. $\square$

Lemma 2. The $OSSP_t$ problem is NP-hard.

Proof. By restriction from $OSSP_{t,1}$: since the $OSSP_{t,1}$ is a special problem instance of $OSSP_t$ and is proved to be an NP-hard problem by Lemma 1, therefore, the $OSSP_t$ problem is NP-hard. $\square$

3. PROPOSED APPROACH

In this section we describe the discrete-event simulation framework used in our approach. We then present a heuristic-based greedy matching algorithm with several proposed heuristics for matching consumer requests while meeting the requirements of the broker, consumers and service providers.

3.1 Discrete-Event Simulation Framework

In order to simulate the interactions between arriving consumer requests and the on-demand spatial service broker, we employ a discrete-event simulation framework [14]. The framework uses the following four components: consumer requests, service providers, the matching algorithm run by the broker and the road network used during matching. Algorithm 1 shows the simulation details. The algorithm starts by loading and initializing the list of providers and creating the road network graph. Line 3 creates a priority queue for storing simulation events where events with earlier arrival times are pushed to the front of the queue. In this line, all consumer arrival events are being generated and stored in the queue. Each event carries the arrival time of the consumer's request, direction of motion, maximum acceptable travel distance and maximum acceptable time before service. This 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. Lines 5 to 24 show the simulation loop which continues until the queue is empty. The simulation starts by dequeuing all proposition acceptance and consumer arrival events with arrival times before or at the current clock (lines 6 and 7 respectively.) Proposition acceptance events are handled in lines 8 to 14. For each event, the next available service time of all the consumer's unaccepted propositions is updated to release the service time units reserved for that consumer. Line 15 then calls a heuristic-based greedy algorithm (presented in Section 3.2) for matching the available consumer requests to service providers. Then, for every matched request, a corresponding acceptance event is added into the queue (lines 16 to 21). 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 for that consumer. Finally, the simulation performance metrics are updated and the clock is advanced to the next time instant.

3.2 Heuristic-based Greedy Matching

Since solving the On-demand Spatial Service Propositions problem is NP-hard, as shown in Section 2.2, we propose several heuristics for efficiently matching consumer requests to

Algorithm 1 Simulation Algorithm

```
1: $providersList \leftarrow$ Initialize list of service providers
2: $roadNetworkGraph \leftarrow$ Initialize edges and vertices of road network
3: Queue $queue \leftarrow$ Insert consumer request arrival events
4: $clock \leftarrow 0$
5: while $queue$ not empty do
6:     $acceptanceEvents \leftarrow$ Get all proposition acceptance events with an event arrival time $\leq$ clock
7:     $arrivalEvents \leftarrow$ Get all consumer arrival events with an event arrival time $\leq$ clock
8:     for each event $e$ in $acceptanceEvents$ do
9:         for each proposition in $e.propositions$ do
10:            if $proposition \neq e.acceptedProposition$ then
11:                $proposition.provider.nextAvailableServiceTime \leftarrow proposition.provider.nextAvailableServiceTime - proposition.provider.serviceTime$
12:            end if
13:        end for
14:    end for
15:    $matchedProp \leftarrow$ HeuristicBasedGreedyMatching($clock$,$arrivalEvents$,$roadNetworkGraph$)
16:    for each consumer proposition set $match$ in $matchedProp$ do
17:        $acceptedProposition \leftarrow$ randomly select a proposition from $match.propositions$
18:        $timeBeforeAcceptance \leftarrow$ randomly generate time between 0 and $t_{timeout}$
19:        Create an acceptance event e with arrival time = $clock + timeBeforeAcceptance$ and storing $acceptedProposition$, and $match.propositions$
20:        queue.enqueue(e)
21:    end for
22:    Update simulation statistics
23:    $clock \leftarrow clock + 1$
24: end while
25: Output Simulation Statistics
```

service providers in real-time. Each heuristic is applied using a greedy approach whose outline is shown in Algorithm 2. 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 each 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 use a single run of Dijkstra's shortest path algorithm 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 (presented next in this section). Then, a priority queue stores the candidate propositions in the order of their scores. Finally, lines 11 to 18 iterate through each 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 waiting 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 service providers of these propositions is updated based on the providers' current service rates. The algorithm then returns the set of matched propositions for all matched consumers.

Next, we present our proposed matching heuristics which can be classified into three categories: service provider-centric, broker-centric and consumer centric-heuristics.

Algorithm 2 Heuristic-based Greedy Matching Algorithm

```
1: for each consumer $c$ in $arrivalEvents$ do
2:     $nearProviders \leftarrow$ Find providers satisfying consumer's $d_c$
3:     $candidtatePropositions[c] \leftarrow$ 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 $candidtatePropositions[c]$ do
7:         $proposition.score \leftarrow$ Calculate score based on selected 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 $matchedPropositions$
```

3.2.1 Service Provider-centric 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 had a service rate of 12 requests/hr (i.e. 5 min/request) and each consumer had 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 any of P_1 and P_2 since both satisfy his travel distance and waiting time constraints. Suppose C_1 was 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 proposition. At t=1, the request of C_2 arrives and again both P_1 and P_1 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 service 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 may have greater 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.

3.2.2 *Broker-centric Heuristics:*

Least Service Time First (LSTF): Unlike service provider-centric heuristics, LSTF is a broker-centric heuristic that aims to help the broker maximize the number of matched consumers by selecting service provider propositions at the current time instant in a way that increases the probability of matching more consumers in the future. To achieve this, LSTF gives a higher priority to propositions from providers with shorter service times. The intuition behind LSTF is that by matching service providers with shorter service times first, the broker is able to use the least number of service time units to match the current consumer request, thereby, leaving more service capacity in the system for matching future requests. Thus, in LSTF the current service time of each service provider is used as a score for all propositions from the provider. Propositions from different providers are sorted in increasing order of their scores, allowing propositions with shorter service times to be selected for matching first.

3.2.3 *Consumer-centric Heuristics:*

Most Dominant First (MDF): The heuristics discussed above focus on maximizing the number of matched providers and matched consumers while satisfying the consumer's maximum acceptable travel distance and waiting time constraints. However, consumers may also prefer service provider propositions that minimize their travel distance and waiting time before service. Thus, the HDF heuristic aims to simultaneously minimize consumers' travel distances and waiting times. To achieve this we define a dominance relationship between service provider propositions for a given consumer as follows:

Definition 2. **Proposition Dominance Relationship** Let $r1$= $(P_1, t_{w,1}, d_1)$ and $r2$= $(P_2, t_{w,2}, d_2, w_2)$ be two service provider propositions for a given consumer C, where P_i denotes the service provider proposed in proposition r_i, $t_{w,i}$ denotes the estimated waiting time for consumer C for provider P_i, and d_i denotes the travel distance between C and P_i. Now, proposition r_1 dominates proposition r_2 if and only if $t_1 \leq t_2$ and $d_1 < d_2$ or $t_1 < t_2$ and $d_1 \leq d_2$.

Based on Definition 2, the MDF heuristic compares all candidate propositions for a given consumer and identifies the dominant proposition in every pair of candidate propositions. MDF then assigns a score to each proposition equal to the number of candidate propositions it dominates. Finally, candidate propositions are stored in a decreasing order of their scores and propositions with higher scores are matched first.

4. EXPERIMENTAL EVALUATION

The goal of our experiments was to perform: (a) a comparative analysis to evaluate how the performance of the proposed heuristics compares to the related work, and (b) a sensitivity analysis to evaluate the effect of the different parameters on the performance of the proposed heuristics. The following candidate heuristics were included in our analysis:

- Least Accepted First (LAF)
- Least Appearance As Candidate First (LCF)
- Least Service Time First (LSTF)
- Most Dominant First (MDF)

We also included the following related work strategies:

- Least Travel Cost (LTC)
- Consumer Least Location Entropy Priority (CLLEP): prioritizes consumers in provider-sparse areas.
- Provider Least Location Entropy Priority (PLLEP): prioritizes providers in consumer-sparse areas.

A grid index was used to divide the spatial region into grid cells for calculating the location entropy in both CLLEP and PLLEP. In CLLEP, consumers in the same grid cell of a provider were considered to be in the neighborhood of that provider. Similarly, providers in the same grid cell of a consumer were considered to be in the neighborhood of that consumer.

We analyzed the performance of the different heuristics by varying and observing the effect of the following workload parameters: the supply-demand ratio *sdr*, the number of required propositions K, the timeout interval length $t_{timeout}$, the minimum and maximum values for the provider service rate (r_{min} and r_{max} respectively), the minimum and maximum values for the maximum acceptable travel distance constraint (d_{min} and d_{max} respectively), the minimum and maximum values for the maximum acceptable waiting time before service constraint (w_{min} and w_{max} respectively), and the grid cell length l_G.

4.1 Experimental Design and Dataset

The experimental design is illustrated in Figure 2. 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 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 $\mid P \mid$ 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,

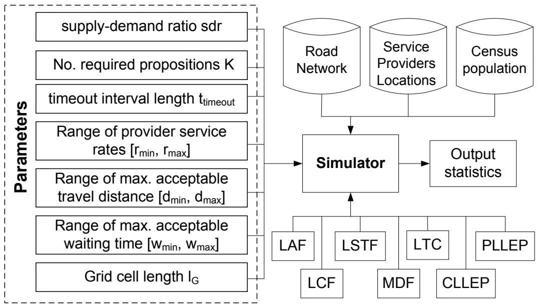

Parameters
supply-demand ratio sdr
No. required propositions K
timeout interval length $t_{timeout}$
Range of provider service rates [r_{min}, r_{max}]
Range of max. acceptable travel distance [d_{min}, d_{max}]
Range of max. acceptable waiting time [w_{min}, w_{max}]
Grid cell length l_G
Road Network
Service Providers Locations
Census population
Simulator
Output statistics
LAF
LSTF
LTC
PLLEP
LCF
MDF
CLLEP

Figure 2: Experimental Design

maximum acceptable waiting times before service and direction of motion were also generated using random numbers that were uniformly distributed over the ranges [d_{min}, d_{max}], [w_{min}, w_{max}] and from $\{N, S, E, W, NE, NW, SE, SW\}$ 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 set as follows: sdr = 1, K = 3, $t_{timeout}$ = 2 min, r_{min} = 5 requests/hr, r_{max} = 15 requests/hr, d_{min} = 4000 meters and d_{max} = 12000 meters, w_{min} = 5 min, w_{max} = 15 min and l_G = 100 meters, unless stated otherwise. We generated the following performance metrics as the output of our simulation: the percentage of matched requests, the percentage of matched service providers, the average and standard deviation of the number of assigned requests per service provider, total execution time, average query response time, average distance traveled per consumer, and average waiting time before service per consumer. Algorithms were implemented using the Java programming language. All experiments were run on a machine with an Intel Xeon Quad Core 3.00 GHz processor and 64 GB RAM.

4.2 Experimental Results

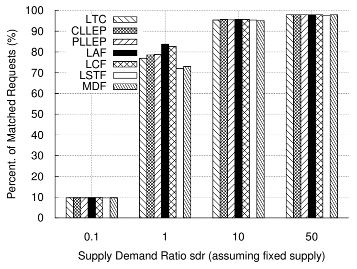
Effect of supply-demand ratio (sdr): To evaluate the effect of the supply-demand ratio on the performance of the different heuristics we ran our simulation for the sdr values of 0.1 (supply << demand), 1 (supply ≈ demand), 10 (supply >> demand) and 50 (supply extremely exceeds demand). For each value, the generated supply is fixed and the number of generated requests varies as shown in the table in Figure 3i.

Percentage of matched requests: Figure 3a shows the effect of sdr on the percentage of matched requests. At small 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 with sdr=1, all requests could have been 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 CLLEP and PLLEP heuristics since their focus was to maximize the future match size. However, LSTF did not achieve a similar percentage of matched requests. The reason is that, unlike CLLEP and PLLEP, LSTF does not account for the spatial distribution of requests. Hence, if most requests originate in regions with small-service-time-providers, the service units saved in other providers might not be utilized. For larger sdr values, all heuristics had a high and comparable performance since supply highly exceeds demand and many requests can be easily matched.

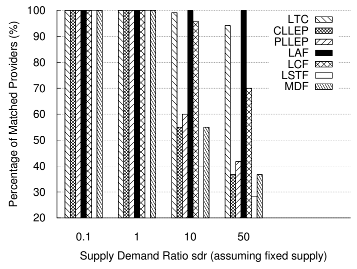
Percentage of matched providers: Figure 3b shows the effect of sdr on the percentage of matched providers. It can be seen that as supply exceeds the demand (i.e. larger sdr values), the percentage of matched providers decreases for almost all heuristics except for the LAF heuristic which is able to engage 100% of the service providers even with a demand that is much lower than the available supply. However, due to the small available demand, each of these matched providers will only be using a small service capacity and thus these providers may also reduce their labor cost (e.g. by using a smaller number of waiters).

Average No. of assigned requests per provider: Figure 3c 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 assigns a larger number of requests per provider since they aim to balance the selection of the providers based on their 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 3i for $sdr = 0.1$ (i.e. =1013.3). The reason is that at $sdr = 0.1$ the demand highly 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 3i. The reasons for exceeding the values in the table 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.

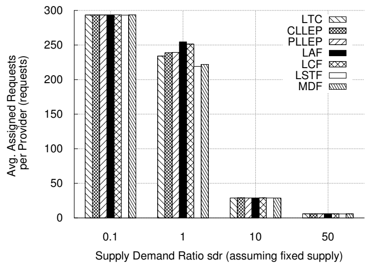
STDEV of assigned requests per provider: Figure 3d shows the effect of sdr on the standard deviation of the assigned requests per provider. At $sdr = 0.1$, all heuristics had a large and comparable standard deviation value due to the large demand which can be concentrated in more populated regions resulting in some providers being assigned much more than other providers. As sdr increases, we notice the LAF consistently results in a smaller standard deviation than other heuristics due to its focus on balancing assignments between different providers. At $sdr = 1$, LCF also achieves a better balance compared to other heuristics since it gives higher priority to service providers that have been least considered as candidate matches allowing a balanced distribution of requests among service providers. However, when supply highly exceeds demand (i.e. $sdr = 10$ and $sdr = 50$), we can see that the LCF does not achieve a good balance as compared to 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 the LCF when matching consumer requests. We can also see that as sdr in-



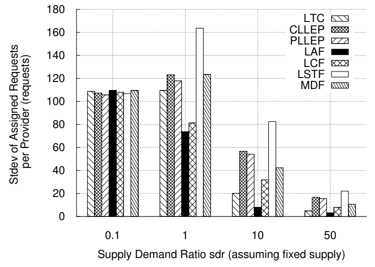
(a) Percentage of matched requests



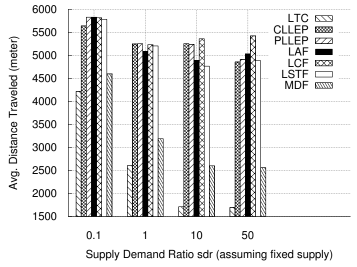
(b) Percentage of matched service providers



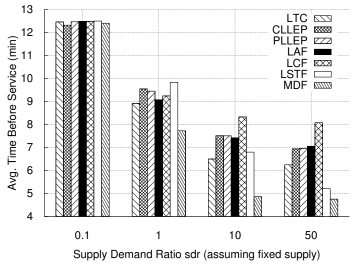
(c) Avg. No. assigned requests per provider



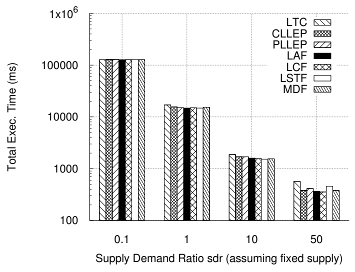
(d) STDEV of No. assigned requests per provider



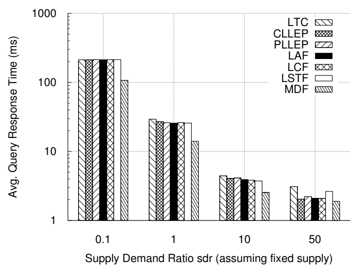
(e) Average distance traveled



(f) Average waiting time before service



(g) Total execution time



(h) Average query response time

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

(i) Synthetic dataset details. Total supply is fixed across all *sdr* values.

Figure 3: 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 3i.

creases 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.

Average distance traveled and waiting time: Figure 3e and Figure 3f show the effect of *sdr* on the average traveled distance and waiting time before service per consumer respectively. From both figures, we can observe that the average distance traveled and average waiting time decrease with the increase in *sdr*. This reason is that with larger *sdr* values, the supply is higher than the demand resulting in many possible service options available for each consumer request which in turn minimizes the needed travel distance and waiting time for each consumer. Figure 3e shows that LTC achieves the least average travel distance followed by MDF (with a difference up to 0.89 km) since both are consumer-centric heuristics that focus on minimizing consumer incurred costs. On the other hand, Figure 3f shows that MDF achieves the least average waiting time per consumer. LTC comes next in the average waiting time (with a difference up to 1.6 min from LTC) except at very large *sdr* values (i.e. 50) where LSTF performs better than LTC due to selecting the least-service-time-providers from the many available providers. From both Figures 3f and 3e we can see that MDF achieves a good balance between the average waiting time and average distance traveled per consumer.

Total execution time and average query response time: Figure 3g shows the total execution time as *sdr* increases. Since the number of generated requests decreases with the increase in *sdr* (as depicted in the table in Figure 3i), the total execution time also decreases, linearly, with the decrease in the number of requests. Figure 3h shows effect on the average query response time per consumer. Again, for a larger number of requests (i.e. at $sdr = 0.1$), at each clock, a large group of requests is being processed simultaneously which results in a larger response time per request/query. As the number of requests decreases at larger *sdr* values, the response time decreases since only a small number of requests is handled per unit time.

Effect of number of required propositions (K): Figure 4 shows the effect of the number of required propositions K on the percentage of matched requests. As K increases,

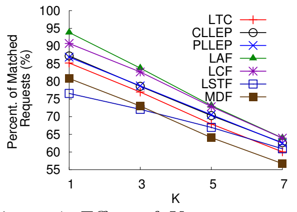


Figure 4: Effect of K (best viewed in color)

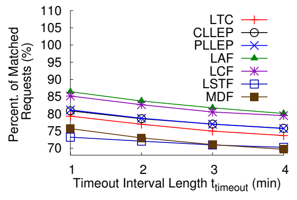


Figure 5: Effect of $t_{timeout}$ (best viewed in color)

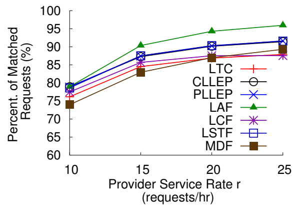


Figure 6: Effect of service rate r (best viewed in color)

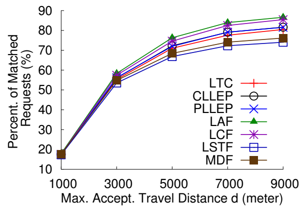


Figure 7: Effect of max. travel distance d (best viewed in color)

the percentage of matched requests decreases. This is because for each group of consumer requests being processed at time t, each request has to be matched to a larger number of service providers with the service time being reserved at all the matched providers and thus a smaller number of requests can be matched as K increases since more service capacity is being reserved per request. We can also see that LAF consistently has a higher percentage of matched consumers followed by LCF, which agrees with the results from Figure 3a.

Effect of timeout interval length ($t_{timeout}$): Figure 5 shows the effect of the timeout interval length on the percentage of matched requests. We observe that as $t_{timeout}$ increases, the percentage of matched requests decreases. This is because the service time for each matched consumer in its K proposed providers remains reserved until the consumer accepts one of the propositions. The time taken to accept a proposition can extend up to $t_{timeout}$ and hence as $t_{timeout}$ increases, the longer it takes for the reserved capacity to be assigned to another consumer, thus reducing the number of matched requests. We also observe that LAF performs best, followed by LCF.

Effect of providers service rate (r): In this experiment, all service providers were assigned an equal service rate which varied from 10 to 25 requests per hour. As shown in Figure 6, as the service rates increase, more requests can be handled per unit time and thus the percentage of matched requests also increases.

Effect of maximum acceptable travel distance (d): In this experiment, all consumers were assigned an equal max acceptable travel distance which was varied as shown in Figure 7. The increase in d results in an increase in the percentage of matched requests. This is because as d increases, each consumer has a higher probability of being matched to more (further) service providers and thus increasing the percentage of matched requests. We can also notice that the rate of increase in the percentage of matched requests slows for larger d values since the maximum acceptable waiting time places another constraint that limits the number of service providers that can be matched to a given request.

Effect of maximum acceptable waiting time before service (w): Figure 8 shows that longer acceptable waiting time constraints (w) increases the percentage of matched requests. The rate of increase slows for larger values of w since the maximum acceptable travel distance constraint places another limit on the service providers that can be matched to a given consumer even when having a large w constraint. In addition, we have also observed that increasing w (or d) results in a larger percentage of matched providers.

Effect of grid cell length (l_G): Figure 9 shows that increasing the grid cell length decreases the total execution time. This is because when querying the grid index for finding all service providers within a given maximum distance from a consumer's location, a smaller number of grid cells is being retrieved. However, the rate of decrease eventually slows. This occurs because larger retrieved grid cells may contain more service providers outside the max distance specified in the query. We also evaluated the effect of the grid cell length on the performance of the CLLEP and PLLEP heuristics since l_G is used to define the neighborhood size in these heuristics. However, we observed no significant change in the percentage of matched requests (plot eliminated for space limitations).

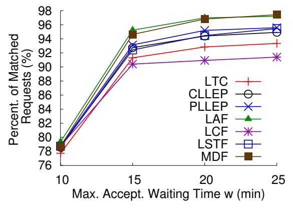


Figure 8: Effect of max. waiting time w (best viewed in color)

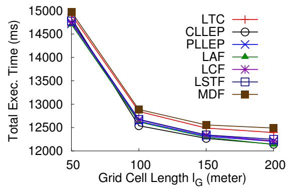


Figure 9: Effect of grid cell length l_G (best viewed in color)

5. DISCUSSION

Other Related Work: Some other studies in the literature have addressed the spatial crowdsourcing problem with different objective functions. In [4], the problem of reliable diversity-based spatial crowdsourcing was studied. In this problem, time constrained spatial tasks are assigned to workers such that the completion reliability and the spatial/temporal diversities of these spatial tasks are maximized. For instance, for a task such as taking videos or photos of a landmark, it is desirable to have views for the landmark from diverse directions and at different times of the day. In [10], workers are assumed to be in moving vehicles whose trajectories are known in advance. The goal is to recruit a subset of the vehicles according to a given limited budget such that the spatial and temporal coverage of the vehicles for the study region are maximized. While these studies fall under the spatial crowdsourcing category, their objective is maximizing the spatial and temporal diversity of the recruited workers or tasks rather than maximizing the number of assigned tasks.

Consumer and Service Provider Constraints: In our On-demand Service Propositions problem definition, every consumer specifies a number of constraints including a maximum acceptable travel distance and a direction of motion. However, different alternatives for consumer constraints can also be used without affecting the computational structure of the problem. For instance, the maximum acceptable travel distance could be replaced by a maximum acceptable travel time cost. Similarly, instead of proposing service providers along the consumer's direction of motion, consumers may alternatively provide a maximum acceptable detour if they are willing to accept service propositions in the opposite direction of their motion within a given detour distance. In addition, service providers may also provide weekly rather than daily service rates since service rates over the weekends may vary from those during weekdays.

Choice of Discrete-event Simulation: In this paper, we employed a discrete-event simulation framework to model the interactions between consumers and the broker and evaluate the performance of the different matching heuristics. Alternative approaches in the literature have used mathematical formulas and queuing theory for modeling supply and demand [3, 9, 11]. However, these methods usually pose restrictive assumptions on the parameters distribution and/or the network topology for developing closed forms. Optimization methods (e.g. integer programming [12]) have also been used; however, they do not scale to large datasets and thus would violate the real-time on-demand requirement of this problem. In our work, we simulated different supply and demand scenarios using synthetic datasets generated with an attempt to approximate real-world characteristics. We could not perform real-world experiments since we did not have access to real datasets which are usually collected as proprietary data.

6. 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 several proposed heuristics for meeting the conflicting requirements of the broker, consumers and service providers, including a new category of service provider-centric heuristics for increasing the number of matched service providers. We evaluated our approach using synthetic datasets with real-world characteristics and a discrete-event simulation framework. Experimental results show that our "Least Accepted First" and "Least Candidate First" heuristics result in the largest number of matched consumer requests when the available supply and demand are balanced. For scenarios were 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. We also proposed a consumer-centric heuristic, "Most Dominant First", that simultaneously minimizes the consumer's travel distance and waiting times. Overall, our results show that the dominance zones of different matching heuristics vary with the supply-demand ratio and thus a supply-demand ratio aware broker is needed for choosing the best matching policy. In the future, we plan to extend OSSP to allow mobile service providers and incorporate consumer ratings in the matching process. We also plan to explore a supply-demand ratio aware broker that adapts to spatio-temporal variations in market conditions and the evolution of the behavior of consumers and service providers. Finally, we plan to model the variation in profit across different types of consumer requests.

7. ACKNOWLEDGMENTS

This material is based upon work supported by FORD University Research Program (URP), the National Science Foundation under Grant No. IIS-1320580, and Minnesota Supercomputing Institute (MSI) at the University of Minnesota (www.msi.umn.edu).

8. 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] S. Benjaafar et al. Peer-to-Peer Product Sharing: Implications for Ownership, Usage and Social Welfare in the Sharing Economy. *Social Science Research Network Working Paper Series*, Oct. 2015.

[4] P. Cheng et al. Reliable diversity-based spatial crowdsourcing by moving workers. *Proc. of the VLDB Endowment*, 8(10):1022–1033, 2015.

[5] 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.

[6] C. Colby and K. Bell. Harvard business review. the on-demand economy is growing, and not just for the young and wealthy. `https://hbr.org/2016/04/the-on-demand-economy-is-growing-and-not-just-for-the-youn`

[7] 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.

[8] P. M. d'Orey and M. Ferreira. Can ride-sharing become attractive? a case study of taxi-sharing employing a simulation modelling approach. *IET Intelligent Transport Systems*, 9(2):210–220, 2014.

[9] S. P. Fraiberger and A. Sundararajan. Peer-to-Peer Rental Markets in the Sharing Economy. *Social Science Research Network Working Paper Series*, Mar. 2015.

[10] Z. He et al. High quality participant recruitment in vehicle-based crowdsourcing using predictable mobility. In *IEEE Conference on Computer Communications*, pages 2542–2550. IEEE, 2015.

[11] M. Hu and Y. Zhou. Dynamic Type Matching. *Social Science Research Network Working Paper Series*, Apr. 2015.

[12] 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.

[13] Instacart. `https://www.instacart.com/`.

[14] R. K. Jain. *Art of Computer Systems Performance Analysis: Techniques for Experimental Design Measurements, Simulation, and Modeling*. John Wiley, 1991.

[15] L. Kazemi et al. Geotrucrowd: trustworthy query answering with spatial crowdsourcing. In *Proc. of the 21st ACM SIGSPATIAL Intl. Conf. on Advances in Geographic Information Systems*, pages 314–323. ACM, 2013.

[16] L. Kazemi and C. Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *Proc. of the 20th Intl. Conf. on Advances in Geographic Information Systems*, pages 189–198. ACM, 2012.

[17] 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.

[18] Spinlister. `https://www.spinlister.com/`.

[19] StyleBee. `https://www.stylebee.com/`.

[20] TaskRabbit. `https://www.taskrabbit.com/`.

[21] H. To et al. A server-assigned spatial crowdsourcing framework. *ACM Trans. on Spatial Algorithms and Systems*, 1(1):2, 2015.