

8-2018

Online spatio-temporal matching in stochastic and dynamic domains

Meghna LOWALEKAR

Singapore Management University, meghnal.2015@phdis.smu.edu.sg

Pradeep VARAKANTHAM


Singapore Management University, pradeepv@smu.edu.sg

Patrick JAILLET

Massachusetts Institute of Technology

DOI: <https://doi.org/10.1016/j.artint.2018.04.005>

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

 Part of the [Artificial Intelligence and Robotics Commons](#), [Numerical Analysis and Scientific Computing Commons](#), and the [Theory and Algorithms Commons](#)

Citation

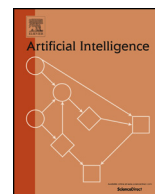
LOWALEKAR, Meghna; VARAKANTHAM, Pradeep; and JAILLET, Patrick. Online spatio-temporal matching in stochastic and dynamic domains. (2018). *Artificial Intelligence*. 261, 71-112. Research Collection School Of Information Systems.

Available at: https://ink.library.smu.edu.sg/sis_research/4329

This Journal Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email libIR@smu.edu.sg.

Contents lists available at [ScienceDirect](https://www.sciencedirect.com)

Artificial Intelligence

www.elsevier.com/locate/artint


Online spatio-temporal matching in stochastic and dynamic domains [☆]


 Meghna Lowalekar ^{a,*}, Pradeep Varakantham ^a, Patrick Jaillet ^b
^a School of Information Systems, Singapore Management University, Singapore

^b Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, United States of America

ARTICLE INFO

Article history:

Received 27 January 2017

Received in revised form 21 March 2018

Accepted 26 April 2018

Available online 5 May 2018

Keywords:

Online matching

Online linear programming

Stochastic optimization

MDPs

ABSTRACT

Online spatio-temporal matching of servers/services to customers is a problem that arises at a large scale in many domains associated with shared transportation (e.g., taxis, ride sharing, super shuttles, etc.) and delivery services (e.g., food, equipment, clothing, home fuel, etc.). A key characteristic of these problems is that the matching of servers/services to customers in one stage has a direct impact on the matching in the next stage. For instance, it is efficient for taxis to pick up customers closer to the drop off point of the customer from the first stage of matching. Traditionally, greedy/myopic approaches have been adopted to address such large scale online matching problems. While they provide solutions in a scalable manner, due to their myopic nature, the quality of matching obtained can be improved significantly (demonstrated in our experimental results). In this paper, we present a multi-stage stochastic optimization formulation to consider potential future demand scenarios (obtained from past data). We then provide an enhancement to solve large scale problems more effectively and efficiently online. We also provide the worst-case theoretical bounds on the performance of different approaches. Finally, we demonstrate the significant improvement provided by our techniques over myopic approaches and two other multi-stage approaches from literature (Approximate Dynamic Programming and Hybrid Multi-Stage Stochastic optimization formulation) on three real world taxi data sets.

© 2018 Elsevier B.V. All rights reserved.

1. Introduction

In many spatio-temporal problems associated with shared transportation [2–4] which includes taxis, ride sharing, super shuttles, etc., customers have to be assigned to servers (e.g., taxis, shuttles) in an online fashion to optimize the revenue or quality of service. Also in case of emergency services [5] such as ambulances, fire trucks, etc. and delivery services [6,7]

[☆] This paper is an extension of our earlier paper at AAAI 2016 [1]. We have extended the paper in the following ways: (a) The earlier paper focused on an approach that considers only two stages. We have now extended it to consider multiple stages; (b) We have added theoretical results on the hardness of offline and online spatio-temporal matching problems; (c) We have proved *a priori* bounds for the algorithms in multi-stage matching problems; (d) We have extended our experimental results to consider our updated approach and also generated results on synthetic data sets to better understand where our approach works well and where it does not work well; (e) We have also provided the results on a new dataset (from New York) that is publicly available and also compare against two other multi-stage approaches from literature (Approximate Dynamic Programming and Hybrid Multi-Stage Stochastic optimization formulation).

* Corresponding author.

E-mail addresses: meghna.2015@phdis.smu.edu.sg (M. Lowalekar), pradeepv@smu.edu.sg (P. Varakantham), jaillet@mit.edu (P. Jaillet).

involving pickup and delivery of food, equipment, clothing, home fuel etc., effective online matching of customers to servers is of great value.

The wide usage of applications such as Uber, Lyft, etc. is a testament to the importance of doing matching well and in quick time. In these applications, a set of available taxis are matched to the customers that are looking for taxis. A key observation that is not exploited in such applications and in most existing literature is that after taking the matched customer to their destination location, the taxi will be best served in next stage of assignment to customers typically around the destination location. In this paper, we are specifically interested in considering such dependence (e.g., assignment for a taxi in the second stage is dependent on the assignment in the first stage) in matching across multiple stages. Specifically, we focus on online matching in domains where:

- Customer demand is uncertain and time dependent, with data available about past customer demand;
- Problems are at a societal scale (with thousands of customers and servers) with a need to make online decisions; and
- There is a need or an opportunity to optimize revenue or quality of service (e.g., time to pick up customers or time to delivery);

Given the challenging nature of the problems (stochasticity, dynamism, societal scale, online, multi-stage), most existing work on relevant problems (described in Section 2) has focused on myopic algorithms [8,9] like greedy and randomized ranking. While these approaches have good competitive ratios in case of online bipartite matching (single stage), they have obvious inefficiencies in handling multi-stage problems due to their myopic nature. To address these, there has been research on multi-stage stochastic models and Approximate Dynamic Programming approaches that consider expected demand for the future stages. However, these approaches have been limited to small scale problems and in restricted settings [10–13,7,14,15].

In this paper, we address the key limitations of previous work by providing the following contributions:

- We first formalize the offline and online spatio-temporal matching problems. The offline problem considers that the customer requests at every decision epoch are known beforehand. The online problem on the other hand considers that only the current decision epoch customer requests are known and the future decision epoch requests arrive according to some partially known (through past demand scenarios) distribution.
- We then provide a two-stage stochastic optimization formulation, which considers samples of future customer demand (typically obtained from historical data) for finding the assignment of servers to customers. We also provide a multi-stage extension.
- Given the large scale of problems of interest with thousands of servers and customers, we provide a decomposition of the above formulation to improve parallelism in handling future demand.
- We provide the theoretical results on competitive ratios for myopic and two-stage algorithms for online spatio-temporal matching problems.
- Apart from considering randomly generated problems, we have also evaluated our approach on datasets of three major taxi companies. We compare against myopic algorithms (typically employed by standard taxi matching applications) such as greedy and one-stage bipartite optimal assignment. In addition, we also compare against the Approximate Dynamic Programming (ADP) approach that has been successfully applied in many resource allocation problem [11–13] and Hybrid Multi-Stage Stochastic optimization (HSS) approach used for the truckload assignment problem [10]. We show that our multi-stage formulation can be solved in times that are competitive to these approaches, while providing significantly better solutions.

We provide a detailed description of the related work, along with a background of existing methods in Section 2. We provide a model and a linear optimization formulation to represent the offline spatio-temporal matching problem in Section 3. We then describe the online variant along with a linear optimization model in Section 4. A decomposition method to improve the scalability in solving the optimization model is also presented in the same section. In Section 5, we describe the key assumptions and how they can be relaxed. Finally, in Section 6, we provide a detailed experimental analysis of our methods against existing benchmarks on three real world taxi data sets and specially created synthetic scenarios. In the appendix, we provide detailed proofs for all the propositions described in Sections 3 and 4. In the appendix, we also provide the details of Approximate Dynamic Programming (ADP) and Hybrid Multi-Stage Stochastic optimization (HSS) approaches used in experimental comparison.

2. Related work

We now describe the multiple threads of research in online sequential decision making that are of relevance to this paper. Online algorithms [16] typically consider requests that are revealed incrementally over time and algorithms have to make decisions based on the requests that are revealed. The key threads of research that are of relevance are: online matching, online multi vehicle pickup and delivery, online MDPs and online stochastic optimization algorithms.

2.1. Online matching

A matching, M in a graph $G(V, E)$ is defined as the set of edges $M \subset E$ such that for every $v \in V$ there is at most one edge in M incident on v . In classical online bipartite matching problem [8], one side of the vertices is known and the other

side of the vertices arrive online. This is formally defined as a graph $G(U, V, E)$ where vertices in the set U are known and vertices in the set V appear online. The goal is to maximize the size of the matching M .

A simple generalization of the online bipartite matching is the weighted case where vertices or edges have weights associated with them and the goal is to maximize the total weight of the matching. This problem has applications in online advertising employed by Yahoo, Google etc. Specifically in such applications, the goal is to optimize the allocation of a fixed advertising space to incoming advertisers who typically arrive at different times.

We describe the commonly used approaches for solving online bipartite matching problems. In the experimental section, we compare our proposed approach with the following approaches.

1. **Greedy:** Greedy algorithm matches the incoming vertex with the best available choice. In case of weighted models, it matches with the maximum weighted vertex or edge. Greedy algorithm is shown to have a competitive ratio¹ of $\frac{1}{2}$ [8].
2. **Randomized greedy:** The randomized greedy algorithm perturbs the value of matching by multiplying it by a random number w between 0 and 1. It then greedily matches the servers to customer by using perturbed value. Goel et al. [9] show that the randomized greedy algorithm achieves a ratio of $1 - \frac{1}{e}$ for the vertex weighted bipartite matching.

In classical online bipartite matching, vertices appear one by one. In our model, discussed in the introduction, a group of requests (possibly more than one) arrive simultaneously at each decision epoch, therefore we can apply standard bipartite matching algorithm on the currently available partial graph. We call this One-Stage Algorithm as it only considers requests available at one stage (the current decision epoch). One-Stage Optimal finds the maximum weighted bipartite matching between available vertices at every decision epoch by solving a linear program. Although most work in online bipartite matching addresses the problem where one side is fixed [17–19], results in [20] show that greedy algorithm achieves a competitive ratio of $1/2$ when both sides of vertices appear online. Recently in Wang et al. [21] an algorithm based on water-filling algorithm has been proposed which achieves a competitive ratio of 0.526 in case both sides of vertices appear online.

While there are similarities, there are multiple differences in our work from research in online matching:

- We consider a multi-stage problem, where there are multiple connected rounds of bipartite matching. Therefore, unlike in online bipartite matching that assigns one service to only one customer, in this paper, we match one service to multiple customers (with one customer at any specific point) over time. A recent work by Dickerson et al. [22] proposes a new model for Online Matching with (offline) Reusable resources in which resources on one side are reusable, i.e., resources are matched multiple times over time but their model assumes that each resource has a fixed position and comes back to its original position before it can be matched again. This assumption is not valid for the problems of interest in this paper where position of resource (server) depends on the previous match (assignment).
- In addition, the spatio-temporal aspect of requests (not present in traditional online matching problems) adds further computational complexity to the matching problem.
- Finally, in terms of approaches, unlike work in online matching which has primarily considered myopic approaches, we pursue multi-stage approaches that consider potential future requests.

2.2. Online multi vehicle pickup and delivery problem

Online Multi Vehicle Pickup and Delivery problems typically represent problems where there are multi-capacity servers that transport multiple goods/loads from their origins to destinations. When servers are used to move people instead of goods, the problem is referred to as dial-a-ride problem [23–25] and when all the origins or all the destinations are located at a depot, the problem is referred to as vehicle routing problem [7].

The general representation of the dial-a-ride problems is ideally suited to represent problems faced by companies such as super shuttle (transports people from an airport to different locations in the city), uber pooling (transports customers from near by start locations to near by destination locations). These problems are hard to solve and the traditional approaches for these problems can solve only very small instances of 96 requests and 8 vehicles [26].

The integer programming formulation, without any spatial or temporal aggregation [26], is difficult to solve and is not scalable to large scale problems and online decision making even for unit capacity. So in the online setting, the problems are solved by either using the heuristics [27–29] or by optimizing at each timestep using only current set of available requests [6,30]. Even after considering discretization of timesteps, exact formulation of this problem involves product of variables. Even after linearizing the product terms, the formulation (shown in Appendix A) is a mixed integer program that is not scalable.

Therefore existing work [10,31,12] and even our paper considers matching supply and demand at the level of zones, cities or areas (abstraction of locations) and not at the level of individual locations. That is to say, all demand and/or supply of a specific type within an area are deemed equivalent to improve scalability.

¹ Competitive ratio of an algorithm is defined as the ratio of the worst case solution obtained by the algorithm and the optimal solution.

There are multiple key differences in assumptions/constraints made in existing work on truckload assignment problem (and other similar ones) as compared to the problem of interest in this paper (e.g. taxi matching). These differences in assumptions are significant as the computational complexity class changes due to these assumptions/constraints.

- **Zone assignment constraints:** In the existing models used for truckload assignment problem, at any timestep, truck can only be assigned to load in the same zone.² This is a strong assumption which makes the formulation a pure network flow problem that can be solved in polynomial time. The complexity in truckload assignment comes due to other generalizations (such as returning trucks to their base locations, each truck serving multiple loads etc.). On the other hand, due to requirement of smaller duration (about 5 minutes) for picking up customers, zones cannot be very large and a server from a neighboring zone can be assigned to the customer in a zone (if it is the nearest server). These assumptions/constraints of our problem result in NP-hardness and thereby increase the complexity.
- **Time to compute dispatch strategies:** Since truckload tasks typically take anywhere from 1 day to 4 days, the time available to compute a dispatch strategy is in the order of tens of minutes or even an hour. In the problem of interest of this paper, decisions on dispatch have to be decided in real time (in less than a minute) and the number of servers is in the order of thousands and demand is in the order of at least 300–400 customers per minute.
- **Representation of current and future assignments in formulation:** The differences in the assumptions/constraints also introduce a key change in the formulations for modeling the problems. Unlike our approach, the existing work on truckload (and other similar domains) employ pure network flow formulations (i.e., only assigning demand from the same zone as supply) for both current and future time steps, or only for the future time step [10]. This is the reason for better approximations provided by our formulation, as shown in the non-trivial improvements in results (on an average 9%) provided by our approach over Hybrid Multi-Stage Stochastic optimization (HSS) [10] on three different taxi data sets.
- **Linear or piecewise linear approximation of value function:** Existing work on the Approximate Dynamic Programming in solving fleet optimization MDPs [31,12,32] approximates the future value using the linear or piece-wise linear functions. While this seems to have provided good results in the truck fleet optimization problems, it is not a good approximation in the taxi fleet optimization. The value function approximation considered for fleet management problem is separable over zones but as in our case servers can be assigned from the nearby regions, the value of having one extra server in a zone will depend on the number of servers present in the nearby zones. Experimentally, our approaches provide on an average 9% improvement over ADP as well.

2.3. Online MDPs

Another relevant thread of research is Online Markov Decision Processes (MDPs). In the online setting, rewards and transition functions of MDP for future timesteps are unknown. To learn these unknown reward and transition functions, following two cases are considered.

- The more popular sub-thread has focused on the cases where the reward and the transition function are assumed to be stochastically stationary and the instances of the reward and the transition function are revealed depending on the action. In this case, Reinforcement Learning [33] has presented numerous techniques for learning the policies, which are guaranteed to maximize the expected value.
- In the second sub-thread [34–36] the revealed reward and transition functions are adversarial to the executed action.

While it is possible to represent our problem as an online MDP, the number of states and actions are exponential in the number of agents. Since the number of agents is in the thousands, it is even difficult to specify the model. For the large scale MDPs, existing works have used state aggregation [37] and Approximate Dynamic Programming methods [13] to compute policies. As described in Section 2.2, Approximate Dynamic Programming methods have been widely applied for the resource allocation problems. We used linear and piecewise linear value function approximation which are shown to work well for truckload assignment and fleet management problem [32,12]. We provide a comparison with this approach in our experiment section. In summary, following are the differentiating factors of the work presented in this paper:

- While it is possible to represent our problem as an online MDP, the number of states and actions are exponential in the number of agents. We provide an experimental comparison of our model with the Approximate Dynamic Programming approach which is generally used to solve the large scale MDPs.
- We neither associate adversarial behavior with the nature nor do we assume stationarity with respect to reward and transition functions. Instead, we assume that the behavior is similar to what has been observed in the training demand settings.

Given that both Online MDPs and our work are focused on online sequential decision making under uncertainty, existing work in online MDPs can benefit by adapting the following two key ideas mentioned in this paper in order to scale to problems with multiple agents:

² Some works [32] consider in the evaluation (not in the formulation) that the loads (tasks/requests) if not served will be available at the future timesteps. But, in the cases where waiting time is few minutes, i.e., demands are available only for one timestep and disappear if not assigned (for example in taxi case), using existing formulations, requests can not be served from the neighboring zones.

- Exploiting anonymity (lack of identity) and homogeneity of agents to address the exponential complexity associated with increasing the number of agents.
- Exploiting decomposability across multiple samples of the future state space evolution.

2.4. Online stochastic optimization

Stochastic programming is used to model the optimization problems that involve uncertainty. These models take advantage of the fact that the probability distributions which represent the data are known or can be estimated [38]. To represent the future uncertainty, multiple samples (scenarios) from the known or estimated probability distribution are considered.

The complexity of these multi-stage stochastic programs increases with increasing the number of stages and sample scenarios [39]. Online anticipatory algorithms [40–43] are generally used to solve large scale stochastic integer programs when the set of feasible decisions at each stage is finite [40]. The assumption is that there exists an offline deterministic algorithm for the application. Online anticipatory algorithms relax the non anticipativity constraints in the stochastic program and make decisions online at a time t in three steps:

- Sample the distribution to obtain a subset of future scenarios.
- Optimize each scenario for each possible decision.
- Select the best decision over all scenarios.

The above algorithm requires evaluating each decision for each possible sample which is computationally expensive. Hence, in general, the following two approximations are used.

- *Consensus*: Instead of optimizing each sample for each decision, consensus optimizes each sample once. Only decisions which are optimal for a sample receive positive score, other decisions are given zero score. The decision having highest score is executed at time t .
- *Regrets*: Regrets algorithm assumes the availability of an application specific regret function which gives fast approximation on the regret value of any decision. Now similar to consensus, regrets algorithm optimizes each sample once but unlike consensus, it assigns score to all decision using the application specific regret function.

Online Anticipatory algorithms have been used in the applications like online vehicle routing, packet scheduling, reservation systems but these applications typically have a small set of feasible decisions at each stage (50–100 requests) [41,44,40].

Unfortunately, in our case, the number of feasible decisions are in millions (tens of thousands of servers and hundreds/thousands of requests). Furthermore, if we are to optimize for each sample separately (using the optimization problem presented in Section 3), we can only handle very few samples within the online time constraints. Thus, the key differentiating factor of this paper is in providing approaches (relaxation and Benders Decomposition on top of the linear optimization) that improve scalability considerably to handle millions of decisions. Though Benders Decomposition is typically used to solve the large scale stochastic optimization problems [45–47], the novelty is to use this for our updated formulation to obtain competitive results.

3. Offline spatio-temporal matching (OSTM)

We first provide an intuitive description of the OSTM problem. We assume that the time is divided into discrete blocks of duration Δ minutes. Given a set of customer requests available at each decision epoch and the initial location of servers (e.g., taxis), the goal of the OSTM is to find a matching between servers and customer requests such that the objective function (e.g., revenue, number of requests) of the matching is optimized. The key constraint is that the requests should be assigned to a server at the decision epoch at which they become available and the server should start serving the request on assignment. On serving a customer request the server moves to the destination location of the customer request. Therefore, the location and availability of servers at each decision epoch depends on the matching performed at previous decision epochs. Formally, an OSTM problem is described using the following tuple:

$$\langle \mathcal{Z}, \mathcal{D}, \mathcal{N}, \mathcal{C}, T, f, g, M \rangle$$

- \mathcal{Z} is the set of all zones. A zone groups all nearby locations into one abstract entity. For instance, in the context of taxis,³ we represent each zone as an area where the distance between any two points in that area is 0.5 km.
- \mathcal{D}^t represents the demand or the set of customer requests for servers at decision epoch t . Each element $j \in \mathcal{D}^t$ is characterized by a tuple $\langle o_j, d_j, R_j^t \rangle$ where $o_j, d_j \in \mathcal{Z}$ denote the origin and destination zone of the requests and R_j^t denotes the number of requests having origin at zone o_j and destination at zone d_j at decision epoch t . In the context of taxis, this would correspond to the set of customer requests that travel between certain starting and ending zones.

³ We include more details on zone creation in Section 6.1.1.

Table 1
OSTM formulation.

OSTM($M, \mathcal{Z}, \mathcal{D}, \mathcal{N}, \mathcal{C}, f, g, T, \delta$):

$$\max \sum_{t=1}^M \sum_{i \in \mathcal{Z}} \sum_{\substack{j \in \mathcal{D}^t, \\ o_j \in g(i,t)}} C_{i,o_j,d_j}^t * x_{ij}^t \quad (2)$$

$$\text{s.t.} \quad \sum_{\substack{j \in \mathcal{D}^1, \\ o_j \in g(i,1)}} x_{ij}^1 \leq \mathcal{N}_i \quad \forall i \quad (3)$$

$$\sum_{i \in f(o_j,t)} x_{ij}^t \leq R_j^t \quad \forall t, j \in \mathcal{D}^t \quad (4)$$

$$\sum_{\substack{j \in \mathcal{D}^t, \\ o_j \in g(i,t)}} x_{ij}^t \leq \mathcal{N}_i - \sum_{t'=1}^{t-1} \sum_{\substack{j \in \mathcal{D}^{t'}, \\ o_j \in g(i,t')}} x_{ij}^{t'} + \sum_{t'=1}^{t-1} \sum_{t''=t'+1}^t \sum_{j \in \mathcal{D}^{t'}, i' \in f(o_j,t')} \sum_{d_j=i} \delta_{i'j}^{t',t''} * x_{i'j}^{t'} \quad \forall i, t > 1 \quad (5)$$

$$x_{ij}^t \text{ integer} \quad \forall i, j, t \quad (6)$$

- \mathcal{N}_i indicates the initial number of servers in zone $i \in \mathcal{Z}$.
- \mathcal{C} represents the objective (e.g., revenue, number of requests served, negative of waiting time), with C_{i,o_j,d_j}^t denoting the objective value obtained by matching a server in zone i to a single customer request with origin and destination zones given by (o_j, d_j) at decision epoch t .
- $T(z, z', t)$ gives the time taken by a server to move from zone z to z' at decision epoch t .
- $f(z, t)$ gives the list of zones for servers that can be assigned to a request originating in zone z at decision epoch t . Depending on the quality of service (QoS) constraints in the domain, these can be appropriately defined. We can define f as a function returning the set of zones, which can be reached from z within a duration τ at decision epoch t , i.e., $z' \in f(z, t)$ if and only if $T(z', z, t) \leq \tau$. This ensures that waiting time for any request is less than τ . Such QoS constraints can be employed to derive f as well as g , that is described next.
- $g(z, t)$ provides the list of originating zones for requests that can be assigned to a server in zone z at decision epoch t , i.e., $z' \in g(z, t) \implies z \in f(z', t)$.
- M is the total number of decision epochs.

Our goal is to identify the assignment of requests to servers so as to maximize the sum of objective value over M decision epochs. This is subject to considering valid matching and the number of requests at each decision epoch.

3.1. Optimization formulation for OSTM problems

We provide the integer linear optimization formulation for OSTM problems in Table 1. Intuitively, the goal of the formulation is to set values for the assignment (of servers to demand) variables while ensuring the flow of servers between zones and the corresponding assignment to demand are valid. Depending on whether the Markovian property for resource allocation is satisfied, the complexity of this formulation can either be polynomial time or NP-Hard. We provide a detailed discussion on this aspect in Section 3.1.1 and an example to describe symbols of the formulation in Section 3.1.2.

The complexity of the formulation is determined by the assumptions on the underlying binary constants $\delta_{ij}^{t,t'}, 1 \leq t \leq t' \leq M$ (refer to Section 3.1.1) and hence these binary constants are defined first. $\delta_{ij}^{t,t'}$ indicates whether zone i server assigned to a single request belonging to the element j of \mathcal{D}^t at decision epoch t completes its trip exactly at decision epoch t' . The formal definition is as follows:

$$\delta_{ij}^{t,t'} = \begin{cases} 1 & \text{if } t + \lfloor \frac{T(i,o_j,t) + T(o_j,d_j,t)}{\Delta} \rfloor + 1 = t'. \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The integer variable x_{ij}^t denotes the number of zone i servers assigned to the element $j \in \mathcal{D}^t$ at decision epoch t . The number of servers available in any zone, at any decision epoch depends on assignments at previous decision epochs. Constraints (3) and (5) ensure that at any decision epoch t , the number of servers assigned from a zone is less than the number of available servers in the zone at decision epoch t . Constraint (4) ensures that for each element in \mathcal{D}^t , the number of requests assigned to servers is less than the number of available requests between origin and destination zone pairs corresponding to element j of \mathcal{D}^t .

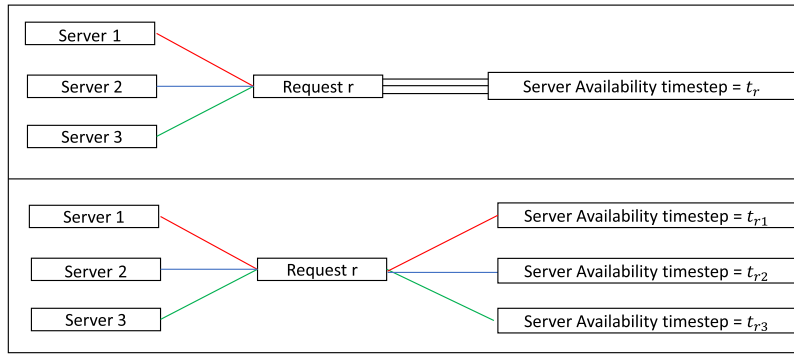


Fig. 1. Cases for server assignment.

3.1.1. Discussion on $\delta_{ij}^{t,t'}$

We can observe in equation (1) that the values of binary constants $\delta_{ij}^{t,t'}$ depends on the total time taken by a zone i server to reach destination zone of element j of D^t . Therefore, as shown in the Fig. 1, we have two cases

- The time at which server becomes available again after serving a request is independent of the assigned server.
- The time at which server becomes available again after serving a request is dependent of the assigned server. For example, suppose server A assigned to a request r at $t = 1$ becomes available again for assignment at $t = 4$. Now if request r is served by another server B at $t = 1$, then it is possible that B will still be serving the request r at $t = 4$ and becomes available again at $t = 6$. One of the cases where this is possible is when the time taken by different servers to reach pickup location of request is different. In the above example A reaches pickup location of r at $t = 2$ and B reaches pickup location of r at $t = 4$.

Case 1 is similar to the Markov property defined in the context of resource allocation problem⁴ and makes the problem polynomial time solvable as the formulation is equivalent to a minimum cost network flow problem. Formally stating the above discussion, for the special case when $\delta_{ij}^{t,t'} = \delta_{i'j}^{t,t'}, \forall j, t, t', i, i'$ (i.e., a request completion decision epoch is same irrespective of the server assigned to it), we show the reduction of the OSTM to a min cost flow problem with integer capacities (Proposition 1). As the min cost flow problem is polynomial time solvable, the OSTM is also polynomial time solvable for this special case.

Proposition 1. *If $\forall j, i, i', t, t' \delta_{ij}^{t,t'} = \delta_{i'j}^{t,t'}$, then the OSTM is reducible to a min cost flow problem.*

Proof. See Appendix D.1. □

For case 2 (i.e., for at least one setting of j, i, i', t, t' if $\delta_{ij}^{t,t'} \neq \delta_{i'j}^{t,t'}$), we show that the OSTM is NP-hard by reducing the well known 3-SAT problem to it (Proposition 2).

Proposition 2. *If $\exists j, i, i', t, t'$ s.t. $\delta_{ij}^{t,t'} \neq \delta_{i'j}^{t,t'}$, then the OSTM is NP-hard.*

Proof. See Appendix D.2. □

Relaxation of optimization formulation As shown in Proposition 1, if special condition on δ values holds, the problem is reducible to min cost flow. Therefore, we will get the integer optimal solution even if we relax the integrality constraints in the integer program for the OSTM.

For the general case, we do not have any theoretical guarantees on the linear relaxation of the integer program. But in our experiments, we observed that the linear relaxation of the OSTM integer program is tight and the difference between the optimal value of the integer program and the relaxed linear program is always less than 1%. Therefore, the objective value obtained on solving the relaxation of the OSTM, provides a tight upper bound on the value of the optimal solution. In addition, a major part of the solution has integer values. A simple heuristic of taking the integer part of the solution

⁴ Please refer to chapter 13 of the book by Powell [13], which states that attributes of a transition resulting from a decision acting on a resource with attribute r is independent of r (i.e., the transition is memoryless). In the context of resource allocation problem, it is mentioned that when system satisfies the Markov property, then a network formulation is obtained otherwise the formulation is not integral but there is no theoretical proof provided. We also obtain a similar result and provide theoretical proofs for both cases. In [13] authors suggest to use aggregation over attributes to restore the Markov property (i.e., network structure) but as we show in our experimental results, this does not give good results in our case.

satisfying all constraints provides a solution which is nearly 95% of the optimal integer solution. In Section 4, we also provide a heuristic to extract an integral solution from the LP relaxation solution.

3.1.2. Example

In this section, we show an example that explains the different parameters and elements of OSTM model.

Example 1. We consider a 3 zone problem and show request assignment over 3 decision epochs (i.e., $M = 3$) in Fig. 2. The following table provides a mapping between formal notation and the visual depiction of the OSTM problem.

Shape	Meaning	Notation	In the shape
Circle	Zone	i	Zone number
Black hexagon	Initial servers	\mathcal{N}_i	Number of servers associated with the zone next to the hexagon at the decision epoch 1
Black diamond	Request	j	Request index for the rectangle below the diamond
Black rectangle	Demand	$\langle o_j, d_j, R_j^t \rangle$	\langle source zone, destination zone, # of requests from source to destination \rangle
Black line	Assignment	x_{ij}^t	Line between zone i and demand j at decision epoch t indicates the possible assignment of zone i server to the demand j .
Green numbers	Revenue	C_{i,o_j,d_j}^t	Revenue obtained on performing the assignment indicated by the line below the number.
Black dashed line	Binary constants	$\delta_{ij}^{t,t'}$	Dashed line from the line between zone i and demand j at decision epoch t to the destination zone d_j at the decision epoch t' indicates that $\delta_{ij}^{t,t'} = 1$. Absence of the dashed line indicates $\delta_{ij}^{t,t'} = 0$.

For ease of explanation, we assume that the travel times between zones are same at all decision epochs, so we drop index t from the definition of T , f and g . Travel time between each pair of zone is mentioned at the bottom of the Fig. 2. This represents the set T . The value of τ and Δ is taken as 5 (min). As mentioned in the model, the sets f and g are populated based on the travel time and τ values. Therefore, $f(2)$ will contain all zones which are reachable from 1 in less than 5 minutes, $f(2) = 1, 2, 3$. Similarly $g(2)$ contains all zones from which it takes less than 5 minutes to reach zone 2, therefore, $g(2) = 1, 2$. All other values are populated in the same way.

The server availability at decision epochs 2 and 3 will depend on the assignments at the decision epoch 1. The server, if assigned, will be available in the destination zone of the requests at the decision epoch where the dashed line from the corresponding assignment line ends. Unassigned servers will be available in the same zone at next decision epoch.

We can also observe from this example, the importance of considering multi-stage matching. For this, we compare the revenue earned by two kinds of decisions

- 1. Single-stage decision, i.e., decisions which do not consider future requests:** In this case, at decision epoch 1 zone 1 server will be assigned to the demand element 2 and zone 2 server will be assigned to the demand element 1, i.e., $x_{12}^1 = 1, x_{21}^1 = 1$ and rest all variables as 0. This will give a total revenue of $20 + 14$. As none of the servers will be available at decision epoch 2, no other requests can be served.
- 2. Multi-stage decision, i.e., decisions which consider future requests:** In this case, at decision epoch 1, zone 1 server will be assigned to the demand element 1 and zone 2 server will not serve any request, i.e., $x_{11}^1 = 1$ and rest all variables for decision epoch 1 are set to 0. At decision epoch 2 both servers will be available in the zone 2 and can serve both requests, i.e., $x_{21}^2 = 1$ and $x_{22}^2 = 1$. This will give a total revenue of $15 + 15 + 10 = 40$.

We have presented the model for the case where the exact demand information is available for all decision epochs. In the next section, we extend this model to consider uncertain customer demand which is revealed in an online fashion.

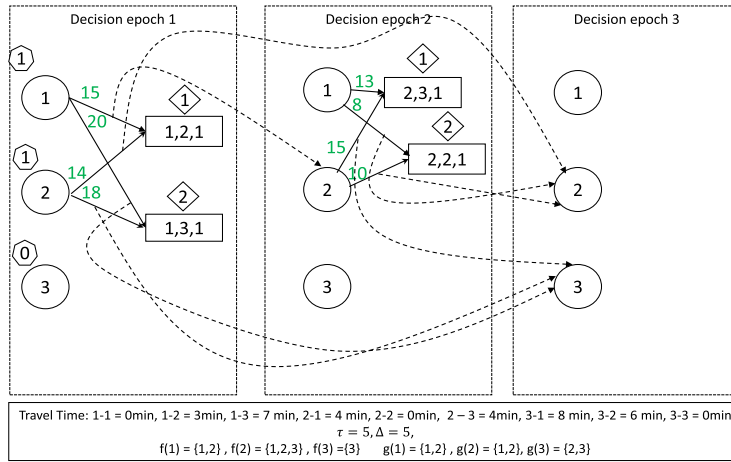


Fig. 2. OSTM example. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

4. Online spatio-temporal matching (OLSTM)

In the **OnLine Spatio Temporal Matching (OLSTM)** problem, requests for all the decision epoch are not known in advance, but instead requests for each decision epoch are only revealed at that decision epoch. The goal is to make matching decisions at each decision epoch so as to optimize the overall objective. Considering potential future scenarios can help in making better matching decisions [10,15,32]. Therefore, we use multiple potential scenarios of demand requests at future decision epochs. One way of obtaining the potential scenarios is to consider the scenarios observed in the past data.

4.1. Problem

An OLSTM problem is described using the following tuple:

$$\langle \mathcal{Z}, \mathcal{D}, \mathcal{N}, \mathcal{C}, \xi^D, \xi^S, f, g, T \rangle$$

We only define, the new elements (i.e., the scenario related information) beyond those defined for the OSTM problem.⁵

- ξ^D is the set of customer request samples for the future decision epoch, where $\xi_t^{D,k}, t > 1$, represents the set of customer requests at $t - 1$ decision epochs in the future in sample k . Each element $j \in \xi_t^{D,k}$ is characterized by a tuple $\langle o_j, d_j, R_j^{t,k} \rangle$ where $o_j, d_j \in \mathcal{Z}$ denote the origin and destination zone of requests and $R_j^{t,k}$ denotes the number of requests having origin at zone o_j and destination at zone d_j at decision epoch t for sample k .
- $\xi^{S,t}$ is the set of servers that are not available at the current decision epoch but will become available at decision epoch t ,⁶ where $\xi_i^{S,t}$ represents the number of servers that will become available in zone i .

Similar to the binary constants δ defined in OSTM model, we will have binary constants $\delta_{ij}^{t,t',k} (t > 1)$, which are set to 1 if zone i server assigned to a single request belonging to the element j of $\xi^{D,k,t}$ at decision epoch t completes its trip exactly at decision epoch t' .

For ease of explanation, we first describe the formulation for a two-stage model, where we only consider requests at the next decision epoch in making matching decisions at the current decision epoch.

Our goal is to identify the assignment of requests to the servers so as to maximize the sum of objective values for the current decision epoch and the expected objective value for the next decision epoch. An integer linear optimization formulation is provided in the Table 2 for computing the best match at the current decision epoch while considering multiple samples of potential requests in the next decision epoch. We refer to this as **TSS()**. Integer variables x_{ij}^1 denote the number of zone i servers assigned to element $j \in \mathcal{D}^1$. Similarly, integer variables $x_{ij}^{2,k}$ denote the number of zone i servers assigned to the element $j \in \xi_2^{D,k}$.

⁵ For ease of notation, in the formulation, we use decision epoch 1 to denote the current decision epoch. Therefore, decision epoch t corresponds to the current decision epoch + $(t - 1)$.

⁶ This could be due to assignments made at previous decision epochs where servers are assigned to requests which take multiple decision epochs to complete.

Table 2
TSS optimization formulation.

TSS($\mathcal{Z}, \mathcal{D}, \mathcal{N}, \mathcal{C}, \xi^D, \xi^S, f, g, T, \delta$):

$$\max \sum_{i \in \mathcal{Z}} \sum_{\substack{j \in \mathcal{D}^1 \\ o_j \in g(i,1)}} C_{i,o_j,d_j}^1 * x_{ij}^1 + \frac{1}{|\xi^D|} \sum_{k \leq |\xi^D|} \sum_{i \in \mathcal{Z}} \sum_{\substack{j \in \xi_2^{D,k} \\ o_j \in g(i,2)}} C_{i,o_j,d_j}^2 * x_{ij}^{2,k} \quad (7)$$

$$\text{s.t.} \quad \sum_{\substack{j \in \mathcal{D}^1 \\ o_j \in g(i,1)}} x_{ij}^1 \leq \mathcal{N}_i \quad \forall i \in \mathcal{Z} \quad (8)$$

$$\sum_{i \in f(o_j,1)} x_{ij}^1 \leq R_j^1 \quad \forall j \in \mathcal{D}^1 \quad (9)$$

$$\sum_{\substack{j \in \xi_2^{D,k} \\ o_j \in g(i,2)}} x_{ij}^{2,k} \leq \mathcal{N}_i - \sum_{j \in \mathcal{D}^1} x_{ij}^1 + \xi_i^{S,2} + \sum_{j \in \mathcal{D}^1, d_j=i} \sum_{i' \in f(o_j,1)} \delta_{i'j}^{1,2} * x_{i'j}^1 \quad \forall i \in \mathcal{Z}, k \leq |\xi^D| \quad (10)$$

$$\sum_{i \in f(o_j,2)} x_{ij}^{2,k} \leq R_j^{2,k} \quad \forall k \leq |\xi^D|, j \in \xi_2^{D,k} \quad (11)$$

$$x_{ij}^1 \quad \text{integer} \quad \forall i \in \mathcal{Z}, j \in \mathcal{D}^1 \quad (12)$$

$$x_{ij}^{2,k} \quad \text{integer} \quad \forall i \in \mathcal{Z}, \forall k \leq |\xi^D|, j \in \xi_2^{D,k} \quad (13)$$

While the first component of the objective value corresponds to the current decision epoch, the second component computes the expected value associated with the future requests (provided in ξ^D). Constraints (8) and (10) ensure that at any decision epoch, the number of assigned servers from the zone i is less than the number of available servers. In Constraint (10) the number of servers available at decision epoch “2” in zone i is calculated by considering the remaining servers in zone i after doing assignments for decision epoch “1”. Constraints (9) and (11) ensure that at any decision epoch, between any zone pair, the number of requests assigned to servers is less than the number of available requests between the origin and destination zone pair. For a single sample, the above integer program is equivalent to an OSTM integer program with $M = 2$. Therefore, as shown in Proposition 1, if $\delta_{i'j}^{1,2} = \delta_{ij}^{1,2} \quad \forall i, i', j$, the relaxation of TSS, for a single sample, will have an integer optimal solution.

For a general case with multiple samples, we show in Proposition 3 that **TSS** is NP-hard.

Proposition 3. Solving **TSS** for more than one sample is an NP-hard problem irrespective of δ values.

Proof. See Appendix D.3. \square

Relaxation of the TSS optimization formulation

For a general case with multiple samples, the linear relaxation of the **TSS** integer program can yield fractional solutions. But in our experiments on synthetic domains and two real world datasets, we observe that the linear relaxation of **TSS** is tight and the difference between the optimal value of the integer program and the relaxed linear program is always less than 1%. We also observe that most of the time, the solution is integral and even if the solution is not integral, major part of the solution has integer values. Therefore, our approach is to solve the relaxed version of the problem and in case the solution is not integral, we round it to an integer solution as described below.

While converting a fractional solution to an integer, only the parts of the solution that are fractional are modified. From the fractional part, variables x_{ij}^1 are rounded in such a way that the number of servers assigned from each zone at the first decision epoch remain close to the fractional optimal solution. This ensures that the servers which were left unassigned by the **TSS**, remain unassigned and the assignments at the second stage are least affected. We denote F_i as $\lceil \sum_j x_{ij}^1 - \sum_j \lfloor x_{ij}^1 \rfloor \rceil$, i.e., the sum of fractional assigned servers from zone i rounded up to the nearest integer. We denote G_j as $\lceil \sum_i x_{ij}^1 - \sum_i \lfloor x_{ij}^1 \rfloor \rceil$, i.e., the number of fractionally assigned requests rounded up to the nearest integer. We further divide the set G_j into two parts with the set G_{1j} containing requests which complete on or before decision epoch 2 and the set G_{2j} containing remaining requests. We greedily assign the servers available in $F_i, \forall i$ to the requests available in the set $G_j, \forall j$ in two rounds. In the first round, we only consider the requests in the set G_{1j} , i.e., we give priority to the requests which can be completed by decision epoch 2. If after first round $\exists i, F_i > 0$, we greedily assign them to the requests in G_{2j} .

4.1.1. Example

In this section, we show an example that explains the different parameters and elements of the OLSTM model.

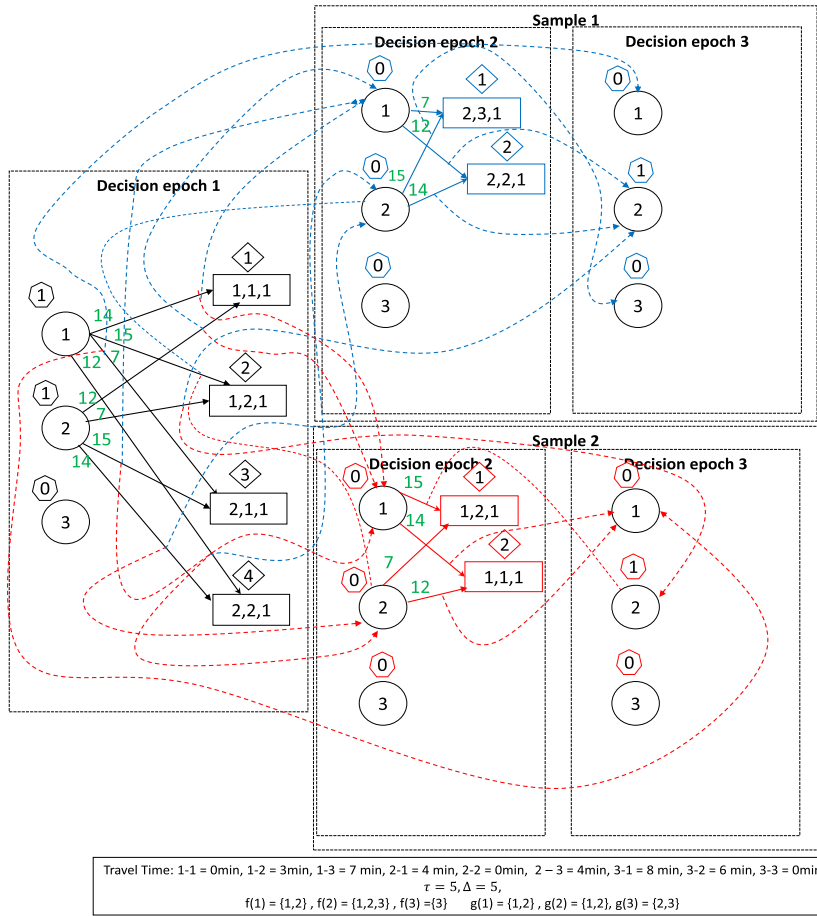


Fig. 3. OLSTM example. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Example 2. Similar to OSTM, we consider a 3 zone problem as shown in the Fig. 3. The following table provides a mapping between the formal notation and the visual depiction of the OLSTM problem. We only define the new elements as compared to the OSTM example.

Shape	Meaning	Notation	In the shape
Colored hexagon	Servers	$\xi_i^{S,t}$	Servers which become available in the zone next to the colored hexagon at decision epoch t due to the completion of previously assigned requests. The value will be same for all the samples.
Colored rectangle	Sampled demand	$\langle o_j, d_j, R_j^{t,k} \rangle$	\langle source zone, destination zone, # of requests from the source to the destination in the sample \rangle
Colored diamond	Sampled request	j	Request index for the colored rectangle below the colored diamond
Colored line	Assignment in samples	$x_{ij}^{t,k}$	Line between zone i and demand element j in sample k at decision epoch t indicates possible assignment of the zone i server to the demand element j in sample k .
Colored dashed line	Binary constants for sample	$\delta_{ij}^{t,t',k}$	Colored dashed line from the line between zone i and the demand element j in the sample k at decision epoch t to the destination zone d_j at decision epoch t' indicates that $\delta_{ij}^{t,t',k} = 1$. Absence of the dashed line indicates $\delta_{ij}^{t,t',k} = 0$.

Table 3
Master formulation.

Master:

$$\max \sum_{i \in \mathcal{Z}} \sum_{\substack{j \in \mathcal{D}^1, \\ o_j \in g(i,1)}} C_{i,o_j,d_j}^1 * x_{ij}^1 + \frac{1}{|\xi^{\mathcal{D}^1}|} \sum_{k \leq |\xi^{\mathcal{D}^1}|} \mathcal{Q}(\{x_{ij}^1\}_{i \in \mathcal{Z}, (j \in \mathcal{D}^1, o_j \in g(i,1))}, k) \quad (14)$$

$$\text{s.t.} \sum_{\substack{j \in \mathcal{D}^1, \\ o_j \in g(i,1)}} x_{ij}^1 \leq \mathcal{N}_i \quad \forall i \in \mathcal{Z} \quad (15)$$

$$\sum_{i \in f(o_j,1)} x_{ij}^1 \leq R_j^1 \quad \forall j \in \mathcal{D}^1 \quad (16)$$

For decision epoch 1, the assignment decisions are independent of samples, but server which will become available after serving the request, will be available for all samples. Therefore, in the graphical representation, we represent it by creating copies of first stage δ constants for each sample and represent them with colored dash lines even for first decision epoch. All these copies will have identical values. In the formulation, this is shown by having $\delta_{ij}^{1,t'}$ and x_{ij}^1 in constraint (10).

The value of τ and Δ is taken as 5. Travel time T and sets f and g are populated as explained in the OSTM example.

We can also observe from this example, that the optimal decision for individual samples will not remain optimal when all the samples are considered together. We show the decisions and revenue computation in following three cases:

1. **Only sample 1 is considered:** If only sample 1 is considered, it will be beneficial to move both servers to zone 2 at decision epoch 2. Therefore, the optimal decision will be, $x_{12}^1 = 1, x_{24}^1 = 1, x_{21}^{2,1} = 1, x_{22}^{2,1} = 1$ and rest all variables as 0. This will give a total revenue of $15 + 14 + 15 + 14 = 58$.
2. **Only sample 2 is considered:** Similar to above case, if only sample 2 is considered, the optimal decision will be $x_{11}^1 = 1, x_{23}^1 = 1, x_{11}^{2,1} = 1, x_{11}^{2,1} = 1$ and rest all variables as 0. This will give a total revenue of $15 + 14 + 15 + 14 = 58$.
3. **Both samples are considered:** On the other hand if both samples are considered, the optimal decision will be $x_{12}^1 = 1, x_{21}^1 = 1, x_{ij}^{2,1} = 1, x_{ij}^{2,1} = 1$ and rest all variables as 0. This will give a total revenue of $14 + 14 + \frac{((15+12)+(15+12))}{2} = 55$. In this case, individual sample's optimal decision will give a revenue of $15 + 14 + \frac{((15+14)+(7+12))}{2} = 53$.

4.2. Benders decomposition

Given the scale of the problems of interest in this paper (i.e., thousands of taxis serving thousands of customers spread across hundreds of zones), we reduce the complexity associated with increasing the number of samples by exploiting the following observation:

Observation 1. In TSS, once the assignment at the first decision epoch, $\{x_{ij}^1\}$ is given, the optimization models for computing the assignment at the second decision epoch, $\{x_{ij}^{2,k}\}$ for each of the samples k , are independent of each other.

We exploit Observation 1 by using the Benders Decomposition [48] method, a master slave decomposition technique where **the Master Problem** is responsible for obtaining the solutions for the “difficult” variables; and **the Slave problem(s)** is (are) responsible for finding the solutions to other variables, given a fixed assignment of values to “difficult” variables (from the master). The Slave problem(s) also generate Benders cuts, which are added to the master problem and the master problem is solved with these cuts to obtain an improved solution. This process continues until no more cuts can be added to the master problem.

Based on Observation 1, $\{x_{ij}^1\}$ are the difficult variables as they impact the values assigned to all the other variables. Therefore, the master is responsible for obtaining the assignments for the $\{x_{ij}^1\}$ variables and the slave(s) are responsible for obtaining the assignments to the $\{x_{ij}^{2,k}\}$. For the master (Table 3), in the optimization provided in TSS, we replace the part of the objective dealing with future variables, $\{x_{ij}^{2,k}\}$ by the recourse function $\mathcal{Q}(\{x_{ij}^1\}_{i \in \mathcal{Z}, (j \in \mathcal{D}^1, o_j \in g(i,1))}, k)$ which becomes the objective function in the slave problems. The recourse function $\mathcal{Q}()$ needs to be computed for each value of x_{ij}^1 . In the slaves (Table 4), we consider the fixed values of x_{ij}^1 and to avoid confusion, we refer to them using the capital letter notation, X_{ij}^1 .

Table 4
Slave formulation.

Slave $\mathcal{Q}(\{X_{ij}^1\}_{i \in \mathcal{Z}, j \in \mathcal{D}^1}, k)$:

$$\max \sum_{i \in \mathcal{Z}} \sum_{\substack{j \in \xi_2^{D,k}, \\ o_j \in g(i,2)}} C_{i,o_j,d_j}^2 * x_{ij}^{2,k} \tag{17}$$

$$\sum_{\substack{j \in \xi_2^{D,k}, \\ o_j \in g(i,2)}} x_{ij}^{2,k} \leq \mathcal{N}_i - \sum_{\substack{j \in \mathcal{D}^1, \\ o_j \in g(i,1)}} X_{ij}^1 + \xi_i^{S,2} + \sum_{\substack{j \in \mathcal{D}^1, \\ d_j=i}} \sum_{i' \in f(o_j,1)} \delta_{i'j}^{1,2} * X_{i'j}^1 \quad \forall i \in \mathcal{Z} \tag{18}$$

$$\sum_{i \in f(o_j,2)} x_{ij}^{2,k} \leq R_j^{2,k} \quad \forall j \in \xi_2^{D,k} \tag{19}$$

Table 5
Dual slave formulation.

Dual Slave $(\{X_{ij}^1\}_{i \in \mathcal{Z}, j \in \mathcal{D}^1}, k)$:

$$\min \sum_{i \in \mathcal{Z}} \alpha_i^k * (\mathcal{N}_i - \sum_{\substack{j \in \mathcal{D}^1, \\ o_j \in g(i,1)}} X_{ij}^1 + \xi_i^{S,2} + \sum_{\substack{j \in \mathcal{D}^1, \\ d_j=i}} \sum_{i' \in f(o_j,1)} \delta_{i'j}^{1,2} * X_{i'j}^1) + \sum_{j \in \xi_2^{D,k}} \beta_j^k * R_j^{2,k} \tag{20}$$

$$\text{s.t. } \alpha_i^k + \beta_j^k - C_{i,o_j,d_j}^2 \geq 0 \quad \forall i \in \mathcal{Z}, j \in \xi_2^{D,k} \tag{21}$$

$$\alpha_i^k \geq 0 \quad \forall i \in \mathcal{Z} \tag{22}$$

$$\beta_j^k \geq 0 \quad \forall j \in \xi_2^{D,k} \tag{23}$$

The dual⁷ of the primal slave problems are provided in Table 5, where α variables are the dual variables corresponding to the constraints (18) and β variables are the dual variables corresponding to the constraints (19).

The *weak duality theorem* [49] states that the solution to a maximization primal problem is always less than or equal to the solution of the corresponding dual problem. Therefore, using the concept of weak duality we can say that by taking the dual of the slave problems, we can find an upper bound on the value of the recourse function ($\mathcal{Q}()$) (objective of primal slave problem), in terms of the master problem variables x_{ij}^1 . These can then be added as optimality cuts to the master problem [47] for generating better first stage assignments.⁸ Let θ^k be the approximation of $\mathcal{Q}()$ function then the master problem with optimality cuts is provided in the Table 6. It should be noted that we are using x_{ij}^1 variables in the “master with optimality cuts” and not the fixed values, X_{ij}^1 . In each iteration we solve the master problem and the computed x_{ij}^1 variable values are passed to the dual slave problems. After solving the dual slave problems, optimality cuts are generated. If the current values of $\theta^k(\forall k)$ satisfy the optimality cut conditions then we have obtained an optimal solution, else cuts are added to the master problem and the master problem is solved again. As we can see in the “Dual Slave” linear programs, the slave problems are independent of each other and are only connected by the choice of the master variables (“difficult” variables). Therefore, once the master variables are fixed, the slave problems can be solved in a parallel fashion.

⁷ The idea of taking a dual of a linear program is an important concept in linear programming. The linear program is typically called the ‘primal’ linear program. For each linear program, there is an associated linear program called its ‘dual’. The dual of a linear program is obtained by creating one variable for each constraint of the primal, and having one constraint for each variable of the primal. The maximization problem is changed to minimization and the roles of the coefficients of the objective function and of the right-hand sides of the inequalities are switched. In addition, transpose of the matrix of coefficients of the left-hand side of the inequalities is taken.

That is to say for a primal problem,

$$\max_x C^T x \quad \text{s.t.} \quad Ax = B$$

we have the associated dual given by

$$\max_y B^T y \quad \text{s.t.} \quad A^T y \geq C.$$

⁸ As the slave problems are always feasible for any value of the master variables we only need to add optimality cuts to the master problem.

Table 6

Master formulation with optimality cuts.

Master Formulation with Optimality Cuts:

$$\max \sum_{i \in \mathcal{Z}} \sum_{\substack{j \in \mathcal{D}^1 \\ o_j \in g(i,1)}} C_{i,o_j,d_j}^1 * x_{ij}^1 + \frac{1}{|\xi^{\mathcal{D}^1}|} \sum_{k \leq |\xi^{\mathcal{D}^1}|} \theta^k \quad (24)$$

$$\text{s.t. } \theta^k \leq \sum_{i \in \mathcal{Z}} \alpha_i^k * (\mathcal{N}_i - \sum_{\substack{j \in \mathcal{D}^2 \\ o_j \in g(i,1)}} x_{ij}^1 + \xi_i^{S,2} + \sum_{\substack{j \in \mathcal{D}^1 \\ d_j=i}} \sum_{i' \in f(o_j,1)} \delta_{i'j}^{1,2} * x_{i'j}^1) + \sum_{j \in \xi_2^{D,k}} \beta_j^k * R_j^{2,k} \quad (25)$$

$$\sum_{\substack{j \in \mathcal{D}^1 \\ o_j \in g(i,1)}} x_{ij}^1 \leq \mathcal{N}_i \quad \forall i \in \mathcal{Z} \quad (26)$$

$$\sum_{i \in f(o_j,1)} x_{ij}^1 \leq R_j^1 \quad \forall j \in \mathcal{D}^1 \quad (27)$$

4.3. Complexity analysis

The complexity of a linear program depends on the number of variables and constraints in the formulation. The maximum number of variables in the TSS formulation are $|\mathcal{Z}| \cdot |\mathcal{D}^1| + \sum_k |\mathcal{Z}| \cdot |\xi_2^{D,k}|$. This maximum value will be obtained when any request can be served using server from any zone. But in practice depending on the value of τ , f and g function will restrict the number of variables. For $\tau = 0$, request can only be served using the server present at its origin zone, therefore, the number of variables are $|\mathcal{D}^1| + \sum_k |\xi_2^{D,k}|$. As the value of τ increases, the number of variables increases. The number of constraints are $|\mathcal{Z}| + |\mathcal{D}^1| + |\xi^{\mathcal{D}^1}| \cdot |\mathcal{Z}| + \sum_k \xi_2^{D,k}$.

Therefore, on increasing the number of zones and the number of samples, both variables and constraints increase. The size of \mathcal{D}^1 and $\xi_2^{D,k}$ increases with the value of Δ . This is because Δ is the duration between 2 decision epochs, so when Δ is small, the number of requests between two decision epochs will be less. Therefore, on increasing the value of Δ , the number of variables and constraints both increase.

In addition to τ , the travel time between zones (T) also affect the number of variables. For a constant τ , if travel time between zones increases, there will be less zones from where servers can be used to serve any requests. Hence the number of variables decrease on increasing travel time keeping the value τ constant.

4.4. Competitive ratio

Competitive analysis is typically employed for evaluating the quality of online algorithms. The metric used is called as the competitive ratio [16] and it compares a solution produced by an online algorithm with the best possible solution. Specifically, the competitive ratio of an online algorithm is defined as the worst-case ratio between the objective of the solution found by the algorithm and the objective of an optimal solution, which assumes all uncertain information is known beforehand. For deterministic input model and maximization problems, an online algorithm is called c -competitive if for any instance of the problem

$$\frac{ALG(I)}{OPT(I)} \geq c \quad \forall I \quad (28)$$

where $ALG(I)$ is the value of any given online algorithm and $OPT(I)$ is the value of the offline optimal algorithm for instance I . Equivalently, this can be written as

$$c = \inf_I \frac{ALG(I)}{OPT(I)} \quad (29)$$

In stochastic environments, we calculate empirical competitive ratio, c_μ^D that is defined as

$$c_\mu^D = E_D \left[\frac{ALG(I)}{OPT(I)} \right] \quad (30)$$

For stochastic input models where multiple distributions of uncertainty are provided, we can also calculate **expected competitive ratio** (Please refer Section 2.2.1 in [18]). It is computed in expectation over the randomness in the input:

$$c_\mu = \inf_D E_D \left[\frac{ALG(I)}{OPT(I)} \right] \quad (31)$$

where D is the distribution over instances I from which input is drawn and expected competitive ratio is the expectation over the ratio achieved by the algorithm and the optimum for that distribution.

Proposition 4. In OLSTM without sample information and adversarial behavior from environment,⁹ when maximizing the number of requests satisfied for a fixed number of servers N , the competitive ratio, c for any deterministic b -stage algorithm (i.e., with information available up to the b th decision epoch) in a M -decision epoch ($M \geq b$) problem is

$$c \leq \frac{1}{M - b + 1}$$

Proof. See Appendix D.4. \square

Proposition 5. In OLSTM with sample information and stochastic behavior from environment according to the samples, when maximizing the number of requests satisfied, the expected competitive ratio, c_μ , of the TSS algorithm is

$$c_\mu \leq \frac{3}{4 * (M - 1)} + \frac{3}{4 * M}$$

where M is the number of decision epochs ($M \geq 3$).

Proof. See Appendix D.5. \square

The expected competitive ratio value is typically overly pessimistic as it measures the worst case. However, the algorithms can have good average case performance, c_μ^D as demonstrated in our experimental results. We will compute this for our formulations and other one-stage approaches, by comparing the solution values obtained with the M -stage optimal offline solution. We solve the OSTM integer program defined in Section 3 to compute the M -stage optimal offline solution.

5. Relaxing the assumptions

In our optimization models for the OSTM and OLSTM, we make a few assumptions. In this section, we describe the assumptions and provide the ways of relaxing them:

1. The time is divided into discrete blocks (each with duration Δ) and we consider only the next decision epoch in TSS. Such an approach is limiting as different requests have different travel time. We relax this assumption by providing a multi-stage model, which allows for smaller values of Δ and also considers multiple discrete blocks at once. More details about the multi-stage model is provided in the Section 5.1.
2. The server moves from one place to another if and only if it is assigned to a request. It is fairly trivial to relax this assumption by introducing the notion of dummy requests. We provide a detailed description in the Section 5.2.
3. Customers are impatient and if they are not assigned a server in one decision epoch, they will not wait for the next decision epoch for a server to be assigned. However, we can easily relax this assumption by representing a customer staying across multiple decision epochs as a customer who leaves and arrives as a new customer at the next decision epoch.
4. The server will start moving as soon as it is assigned to a request. The time taken to compute the assignment is ignored. Therefore, a server assigned to a request at decision epoch t will start moving towards request at the time corresponding to decision epoch t . This is a reasonable assumption, as we are able to obtain solutions in less than a minute (as shown in our experiments).

5.1. Multi-stage model

In TSS, we only consider the future requests for the next decision epoch. Given that different requests may have different travel time, we extend the TSS model to consider the samples for multiple decision epochs to improve the performance for small Δ values. The extended model is referred to as MSS.

The optimization model for MSS is shown in the Table 7. A parameter Q is added to the formulation which denotes the look-ahead timesteps, i.e., Q denotes the number of decision epochs for which future demand samples are considered. The number of variables and constraints in the formulation increase with the value of Q . Therefore, the complexity of MSS formulation increases as the value of Q increases.

The variable x_{ij}^1 denotes the number of zone i servers assigned to the j th element of D^1 at the current decision epoch. The variable $x_{ij}^{t,k}$, $t > 1$ denotes the number of zone i servers assigned to element j of $\xi_t^{D,k}$.

Constraints (33) and (35) ensure that at any decision epoch, the number of servers assigned from zone i is less than the number of available servers. Constraints (34) and (36) ensure that at any decision epoch, between any zone pair, the

⁹ Environment generates demand that creates worst case performance for algorithms decisions.

Table 7
MSS.

MSS($\mathcal{Z}, \mathcal{D}, \mathcal{N}, C, \xi^D, \xi^S, f, g, T, \delta, Q$):

$$\max \sum_{i \in \mathcal{Z}} \sum_{\substack{j \in \mathcal{D}^1 \\ o_j \in g(i,1)}} C_{i,o_j,d_j}^1 * x_{ij}^1 + \frac{1}{|\xi^D|} \sum_{k \leq |\xi^D|} \sum_{t=2}^{Q+1} \sum_{i \in \mathcal{Z}} \sum_{\substack{j \in \xi_t^{D,k} \\ o_j \in g(i,t)}} C_{i,o_j,d_j}^t * x_{ij}^{t,k} \quad (32)$$

$$\text{s.t.} \quad \sum_{\substack{j \in \mathcal{D}^1 \\ o_j \in g(i,1)}} x_{ij}^1 \leq \mathcal{N}_i \quad \forall i \in \mathcal{Z} \quad (33)$$

$$\sum_{i \in f(o_j,1)} x_{ij}^1 \leq R_j^1 \quad \forall j \in \mathcal{D}^1 \quad (34)$$

$$\begin{aligned} \sum_{\substack{j \in \xi_t^{D,k} \\ o_j \in g(i,t)}} x_{ij}^{t,k} \leq \mathcal{N}_i - \sum_{\substack{j \in \mathcal{D}^1 \\ o_j \in g(i,1)}} x_{ij}^1 - \sum_{t'=2}^{t-1} \sum_{\substack{j \in \xi_{t'}^{D,k} \\ o_j \in g(i,t')}} x_{ij}^{t',k} + \xi_i^{S,t} \\ + \sum_{t'=2}^t \sum_{\substack{j \in \mathcal{D}^1, \\ d_j=i}} \sum_{i' \in f(o_j,1)} \delta_{i'j}^{1,t'} * x_{i'j}^1 + \sum_{t'=2}^{t-1} \sum_{d_j=i}^t \sum_{j \in \xi_{t'}^{D,k}, i' \in f(o_j,t')} \delta_{i'j}^{t',t''} * x_{i'j}^{t',k} \quad \forall i \in \mathcal{Z}, k \leq |\xi^D|, \forall t > 1 \end{aligned} \quad (35)$$

$$\sum_{i \in f(o_j,t)} x_{ij}^{t,k} \leq R_j^{t,k} \quad \forall k \leq |\xi^D|, j \in \xi_t^{D,k}, \forall t > 1 \quad (36)$$

$$x_{ij}^1 \quad \text{integer} \quad \forall i \in \mathcal{Z}, j \in \mathcal{D}^1 \quad (37)$$

$$x_{ij}^{t,k} \quad \text{integer} \quad \forall i \in \mathcal{Z}, \forall k \leq |\xi^D|, j \in \xi_t^{D,k}, \forall t > 1 \quad (38)$$

number of requests assigned to servers is less than the number of available requests between the origin and destination zone pair.

Given the similarity in the formulations, we can again employ Benders Decomposition to reduce the complexity with increasing the number of samples. Difficult variables will still be the stage 1 variables, i.e., x_{ij}^1 and all other variables, $x_{ij}^{t,k}$ ($t > 1$) would be the slave variables with a slave for each sample of customer requests.

5.2. Dummy requests

We can remove the assumption that servers will only move when they are assigned to a request by introducing dummy requests. Dummy requests have zero revenue and have a destination in a given zone. We introduce u_{ij}^1 as an integer variable denoting the number of zone i servers assigned to move to zone j at the current decision epoch, i.e., the number of zone i servers assigned to dummy requests with destination in zone j at the current decision epoch. Similarly, $u_{ij}^{t,k}$ denote the number of zone i servers assigned to move to zone j in sample k at decision epoch t , $t > 1$. We modify the **MSS** formulation to include these variables. There will be a cost associated with the movement of server to serve dummy requests which is included in the objective function. The modified **MSS** formulation is shown in Table 8.¹⁰ $\text{Cost}_{i,j}^t$ denotes the cost of moving a server from zone i to zone j at decision epoch t .

6. Experiments

In this section, we will compare the performance of seven approaches Multi-Stage Stochastic optimization (**MSS**)¹¹ and Benders Decomposition (BD), Greedy (GD) algorithm, Randomized Greedy Algorithm (RGD), One-Stage optimization (OS), Approximate Dynamic Programming formulation (ADP) and Hybrid Multi-Stage Stochastic optimization approach (HSS) used in [10]. We employ **MSS**($\Delta = x, Q = y$) and BD($\Delta = x, Q = y$) to refer to our approaches when the time interval Δ is set to x and the look ahead duration Q is set to y . We compare different approaches with respect to (i) Revenue earned by taxis; (ii) Number of requests satisfied; and (iii) Run-time to compute the assignment.

Table 9 provides the outline for this section. We will show two main results that demonstrate the significant utility of our approaches:

¹⁰ We abuse the notation a bit and also use $\delta_{ij}^{t,t'}$ as a binary constant which is 1 if server starting at decision epoch t from zone j reaches zone i exactly at decision epoch t' .

¹¹ **MSS** is a generalization of **TSS**, so we only focus on **MSS**. **TSS** is equivalent to **MSS** for $Q = 1$.

Table 8
MSS formulation with Dummy requests.

MSS($\mathcal{Z}, \mathcal{D}, \mathcal{N}, \mathcal{C}, \xi^D, \xi^S, f, g, T, \delta, Q$):

$$\begin{aligned} \max \quad & \sum_{i \in \mathcal{Z}} \left(\sum_{\substack{j \in \mathcal{D}^1 \\ o_j \in g(i,1)}} C_{i,o_j,d_j}^1 * x_{ij}^1 - \sum_{j' \in \mathcal{Z}} Cost_{ij'}^1 * u_{ij'}^1 \right) \\ & + \frac{1}{|\xi^D|} \sum_{k \leq |\xi^D|} \sum_{t=2}^{Q+1} \sum_{i \in \mathcal{Z}} \left(\sum_{\substack{j \in \xi_t^{D,k} \\ o_j \in g(i,t)}} C_{i,o_j,d_j}^t * x_{ij}^{t,k} - \sum_{j' \in \mathcal{Z}} Cost_{ij'}^t * u_{ij'}^{t,k} \right) \end{aligned} \quad (39)$$

$$\text{s.t.} \quad \sum_{\substack{j \in \mathcal{D}^1 \\ o_j \in g(i,1)}} x_{ij}^1 + \sum_{j' \in \mathcal{Z}} u_{ij'}^1 \leq \mathcal{N}_i \quad \forall i \in \mathcal{Z} \quad (40)$$

$$\sum_{i \in f(o_j,1)} x_{ij}^1 \leq R_j^1 \quad \forall j \in \mathcal{D}^1 \quad (41)$$

$$\begin{aligned} \sum_{\substack{j \in \xi_t^{D,k} \\ o_j \in g(i,t)}} x_{ij}^{t,k} + \sum_{j' \in \mathcal{Z}} u_{ij'}^{t,k} \leq \mathcal{N}_i - \sum_{j' \in \mathcal{Z}} u_{ij'}^1 - \sum_{\substack{j \in \mathcal{D}^1 \\ o_j \in g(i,1)}} x_{ij}^1 - \sum_{t'=2}^{t-1} \left(\sum_{\substack{j \in \xi_{t'}^{D,k} \\ o_j \in g(i,t')}} x_{ij}^{t',k} + \sum_{j' \in \mathcal{Z}} u_{ij'}^{t',k} \right) + \xi_i^{S,t} \\ + \sum_{t''=2}^t \sum_{\substack{j \in \mathcal{D}^1 \\ d_j=i}} \sum_{i' \in f(o_j,1)} \delta_{ij'}^{1,t''} * x_{i'j}^1 + \sum_{t''=2}^{t-1} \sum_{t''=t'+1}^t \sum_{\substack{j \in \xi_{t''}^{D,k} \\ d_j=i}} \delta_{ij'}^{t',t'',k} * x_{i'j}^{t',k} \end{aligned}$$

$$+ \sum_{t''=2}^t \sum_{j' \in \mathcal{Z}} \delta_{ij'}^{1,t''} * u_{ij'}^1 + \sum_{t''=2}^{t-1} \sum_{t''=t'+1}^t \sum_{j' \in \mathcal{Z}} \delta_{ij'}^{t',t'',k} * u_{ij'}^{t',k} \quad \forall i \in \mathcal{Z}, k \leq |\xi^D|, \forall t > 1 \quad (42)$$

$$\sum_{i \in f(o_j,t)} x_{ij}^{t,k} \leq R_j^{t,k} \quad \forall k \leq |\xi^D|, j \in \xi_t^{D,k}, \forall t > 1 \quad (43)$$

$$x_{ij}^1, u_{ij'}^1 \quad \text{integer} \quad \forall i \in \mathcal{Z}, j \in \mathcal{D}^1, j' \in \mathcal{Z} \quad (44)$$

$$x_{ij}^{t,k}, u_{ij'}^{t,k} \quad \text{integer} \quad \forall i \in \mathcal{Z}, \forall k \leq |\xi^D|, j \in \xi_t^{D,k}, j' \in \mathcal{Z}, \forall t > 1 \quad (45)$$

Table 9
Experiment section outline.

Section	Description	Key content
6.1	Datasets	Details on the datasets and different data fields used from the datasets.
6.2	Experimental settings	Details on the different inputs, parameters and evaluation settings used.
6.1.1	Zone creation	Describes the process of creating multiple zones for any dataset.
6.3	Main results	Describes our key results and shows the comparison of seven algorithms at different times on the three metrics of revenue, number of requests and run-time.
6.4	Justification for values of parameter settings	Justifies the parameters used in Section 6.3 by comparing the performance of our algorithms for different parameter values.
6.5	Synthetic scenarios where MSS and BD do not improve performance	Describes the specially created scenarios where our algorithms do not provide good results.

- **MSS** and BD consistently outperform myopic (GD, RGD and OS) and multi-stage approaches (ADP and HSS). While the improvement varies, on an average (across data sets and other settings) there was a 20% improvement over GD, RGD and OS and 9% improvement over HSS and ADP.
- BD provides the best trade off between run-time and solution quality. It can solve the problems in quickly while achieving roughly the same solution quality as **MSS**.

6.1. Datasets

We conducted our experiments by taking the demand distribution from three real world datasets of major taxi companies. The first dataset is the publicly available New York Yellow Taxi Dataset [50], henceforth referred to as NYDataset. The names of the other two real world datasets can not be revealed due to confidentiality agreements. They are referred to as

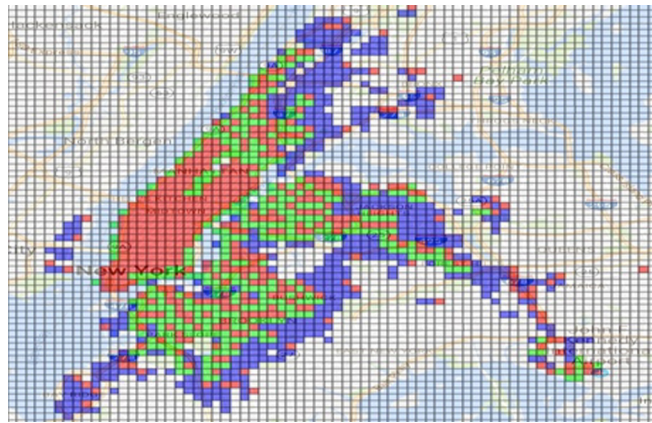


Fig. 4. Zone creation. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Dataset1 and Dataset2. These datasets contain the data of past customer requests for taxis at different times of the day and for different days of the week. From these datasets, we take the following fields:

- Pickup and Drop off Location (Latitude and Longitude coordinates): These locations are mapped to the zones as mentioned in the Section 6.1.1.
- Pickup Time: This time is converted to the appropriate timestep based on the value of Δ . The pickup time $hh:mm$ is mapped to the timestep/decision epoch $\lceil \frac{hh*60+mm}{\Delta} \rceil$. For example, for $\Delta = 5$, 08:04 AM is mapped to the timestep $\lceil \frac{8*60+4}{5} \rceil = 97$, i.e., all the requests between time 08:00 AM and 08:05 AM are mapped to the time 08:05 AM.

The distance of a trip and the revenue earned from a trip are computed using the underlying model typically employed by the taxi companies (an initial base fare + a quantum that accrues with the distance traveled/time taken). We describe it in Section 6.2.

Since our approaches perform well on these three real world data sets, we also provide the specially created scenarios where our approaches may not work as well. The details of these specially created synthetic scenarios are described in Section 6.5.

In the next subsection, we describe the zone creation process using the publicly available NYDataset. The zone creation process is important because we perform matching at the level of zones. The zone level matching ensures scalability while providing a good approximation.

6.1.1. Zone creation

We employ past data of pickups and drop-offs to divide a city into zones. Using past data helps in providing the right tradeoff between having few zones and the error introduced due to consideration of zones. There are four key steps to the zone creation process for all three data sets:

- We divide the entire city area into small grids ($0.5 \text{ km} \times 0.5 \text{ km}$).
- We consider the minimum number of grids that make up most of the pickups and drop-offs ($> 90\%$). These are the first set of zones. This will ensure that the error is negligible for most of the requests.
- We then add a minimum number of zones, so that 9% of the remaining pickups and drop-offs are within $1 \text{ km} \times 1 \text{ km}$ of a zone.
- We add a minimum number of zones, so that the remaining 1% of the pickups and drop-offs are within $2 \text{ km} \times 2 \text{ km}$ of a zone.

Fig. 4 provides the spatial configuration of zones in case of New York city. All the red grids represent zones. Green grids represent the areas that are within $1 \text{ km} \times 1 \text{ km}$ of a zone and account for 9% of the pickups/drop-offs. Blue grids represent the areas that are within $2 \text{ km} \times 2 \text{ km}$ of a zone and account for 1% of past pickups/drop-offs. We used the trip data for 6 months from Jan 2016–June 2016 (total 136830072 pickups/drop-offs) to create these zones. To assign zone to any new location, we check the distance of the location from the centers of these created zones and assign the zone with minimum distance. The percentage values can be changed depending on the need and dataset,

6.2. Experimental settings

There are three different categories of settings that have an

1. **Inputs provided to all algorithms:** These include:

- τ : It represents the maximum time within which the taxi should reach the pickup point. We take the value of τ as 5 minutes, i.e., at any decision epoch a taxi can be assigned to a request if and only if the time taken by the taxi to reach the request origin zone is less than or equal to 5 minutes.
- Revenue model: We employed the following model for experimental evaluation that is a well accepted standard ([51]). The model calculates revenue as follows:

$$C_{i,o_j,d_j}^t = \mathbf{B} + \mathbf{R} * \text{dist}(o_j, d_j) - \mathbf{P} * (\text{dist}(i, o_j) + \text{dist}(o_j, d_j)) \quad \forall t \quad (46)$$

where $\text{dist}(o_j, d_j)$ is the distance between the centers of zones o_j and d_j . For different cities, the values of \mathbf{B} , \mathbf{R} and \mathbf{P} are different.¹²

- Travel time between zones: For the experimental results, the time taken to travel between zones¹³ is also same irrespective of the time of day and is calculated as

$$T(z, z', t) = \frac{\text{dist}(z, z')}{v_{\text{taxi}}} \quad \forall t$$

where v_{taxi} is the speed of taxi which is taken as 40 km per hour.

- Number of taxis (\mathcal{N}_i^1): The number of taxis used is dependent on the fleet size of the taxi company. At the start of the experiment taxis in different zones are distributed either uniformly if taxi locations are not observed (NYDataset and Dataset1) or as observed in the data (Dataset2). Based on the assignment obtained by algorithms at any decision epoch availability of taxis at the next decision epoch is updated. In the results section, we vary the number of taxis to show the performance of all algorithms for different number of taxis.
- Number of zones ($|\mathcal{Z}|$): We described the zone creation process in the Section 6.1.1. Using our zone creation process we obtained 483 zones for the 2 confidential datasets and 436 zones for the NYDataset. For a different city and dataset, the number of zones will be different.

2. Parameters of algorithms: The parameters required by our algorithms (MSS and BD) are:

- Decision epoch length (Δ): This parameter determines how often, the algorithm should be executed and assignment decisions are made. We identify the right value of Δ through experiments as described in results section.
- Look ahead timesteps (Q): The value of Q is taken such that $\Delta * (Q + 1) = 30$ minutes, i.e., if Δ is 1, Q is taken as 29 and if Δ is 15, Q is taken as 1. We choose a fix value of 30 minutes because more than 90% of the requests complete within 30 minutes with average travel time between 5–10 minutes. So 30 minutes provides a good look ahead. But this value can be changed depending on the dataset.
- Number of samples (ξ^D): While computing an assignment at decision epoch t , our approaches (MSS and BD) and existing multi-stage approaches (HSS and ADP) require samples of customer requests at decision epochs $t + 1, t + 2, \dots, t + Q$ from past data (at the same decision epoch on a weekday/weekend depending on whether the decision epochs are on a weekday/weekend). We identify the right value for the number of samples through experiments as described in results section.

3. Evaluation settings: For each algorithm, once the assignment is computed at each decision epoch, we evaluate the assignment on the realized requests (which are samples from the past data that are not considered while computing the assignment). The objective for all the algorithms is to maximize revenue. To provide the right trade-off between run-time and solution quality, we considered 3 iterations in BD. The evaluation settings include the following :

- Number of decision epochs (M): Typically transitions in the taxi demand happen approximately every 3 hours (morning and evening peak hours, lunch), so we choose the value of 2.5 hours, i.e., $M = \frac{2.5 * 60}{\Delta}$. However, we did try other values for M (lower and higher than 2.5 hrs) and the results were similar.
- Number of days and the time of evaluation: We performed experiments with requests at various times of the day, 8:00 AM, 3:00 PM, 6:00 PM and on different days. We evaluated the approaches by running them on 45 different days and taking the average values over 45 days for Dataset1 and Dataset2. For NYDataset, we evaluated the approaches on 15 weekdays between 4th April 2016–22nd April 2016.

We conducted experiments with all the combinations of settings and inputs mentioned in this section. To avoid repeating similar results over and over again, we provide the representative results.

6.3. Main results

In this section, we compare the revenue and the number of requests served by all seven algorithms. We also compare the run-time of all the approaches by using different number of taxis.

¹² For dataset1, dataset2, we use 3.8, 0.5 and 0.1 for \mathbf{B} , \mathbf{R} and \mathbf{P} respectively. For NYDataset, we use 2.5, 2.5, 0.1 for \mathbf{B} , \mathbf{R} and \mathbf{P} respectively.

¹³ We can update it to more realistic values from data without impacting the overall results.

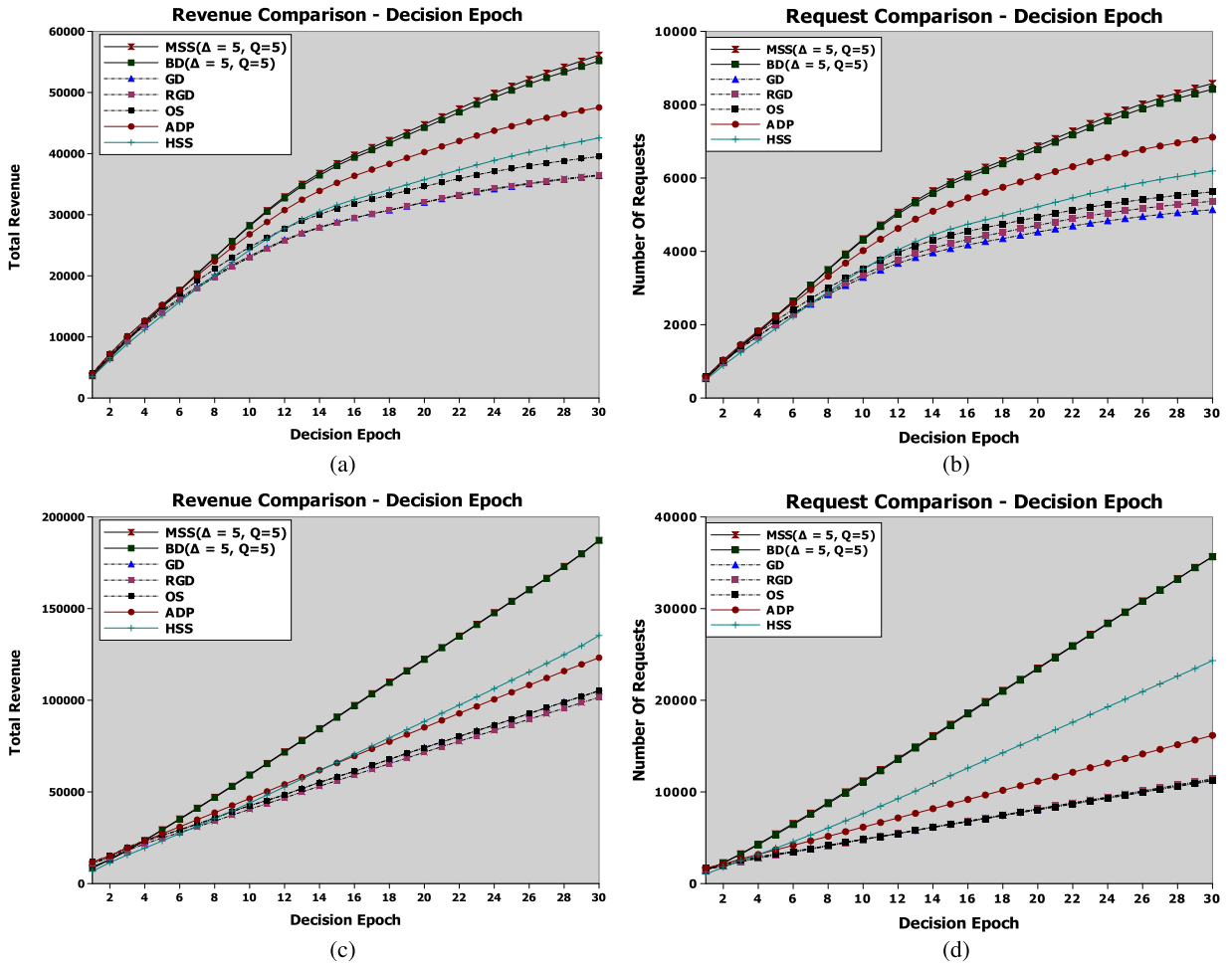


Fig. 5. Comparison of revenue earned and number of requests served using different algorithms at each decision epoch. $|\mathcal{Z}| = 483$, $|\xi^D| = 10$, $\mathcal{N}_t^1 = 2000$ (a), (b) Dataset1, (c), (d) Dataset2.

We conducted experiments by considering the demand distributions from all the datasets. We choose the best configuration for parameters of all algorithms (justification provided in the next section). All approaches are executed for $\Delta = 5$. For our approaches, we use 10 samples and the look ahead timesteps (Q) as 5. For HSS and ADP as well, we use the number of samples as 10. For HSS similar to MSS, we use the look ahead timesteps as 5. For ADP, we use the look ahead timesteps as 10. ADP algorithm as described in the Appendix C involves solving an optimization problem for each sample and for each time step in the planning horizon. As opposed to other multi-stage approaches, ADP algorithm considers one sample at a time, therefore, we use a higher value of Q for ADP. But it is not possible to run ADP algorithm for more than 10 time steps for sufficient number of samples in real time. Moreover, we checked on few instances by using all the $M(30)$ time steps as planning horizon and did not see any significant improvement in the performance of ADP algorithm. Therefore, we report the results for ADP algorithm with a fixed value of Q as 10.

We first compare the revenue obtained by all the algorithms at each time step for 2000 taxis starting from 8AM. We have similar results while starting from 3:00 PM and 6:00 PM. Figs. 5 and 6 show the comparison of the total revenue obtained and the total number of requests served by different algorithms at each decision epoch for all the datasets. Here are the key observations:

- For the first decision epoch, the value obtained by the myopic approaches is higher than MSS and BD. This is because MSS and BD take sub-optimal decisions at the current decision epoch to obtain high revenue over future decision epochs. After initial decision epochs, MSS and BD starts outperforming other approaches.
- With respect to the revenue, on NYDataset, MSS and BD provided close to \$37 improvement per taxi as compared to the myopic approaches and close to \$22 improvement per taxi as compared to ADP and HSS. For Dataset2, MSS and BD provided nearly \$41 improvement per taxi as compared to the myopic approaches and \$26 improvement as compared to ADP and HSS. Similarly, on Dataset1, we get \$8 improvement as compared to the myopic approaches and \$4 improvement as compared to ADP and HSS.

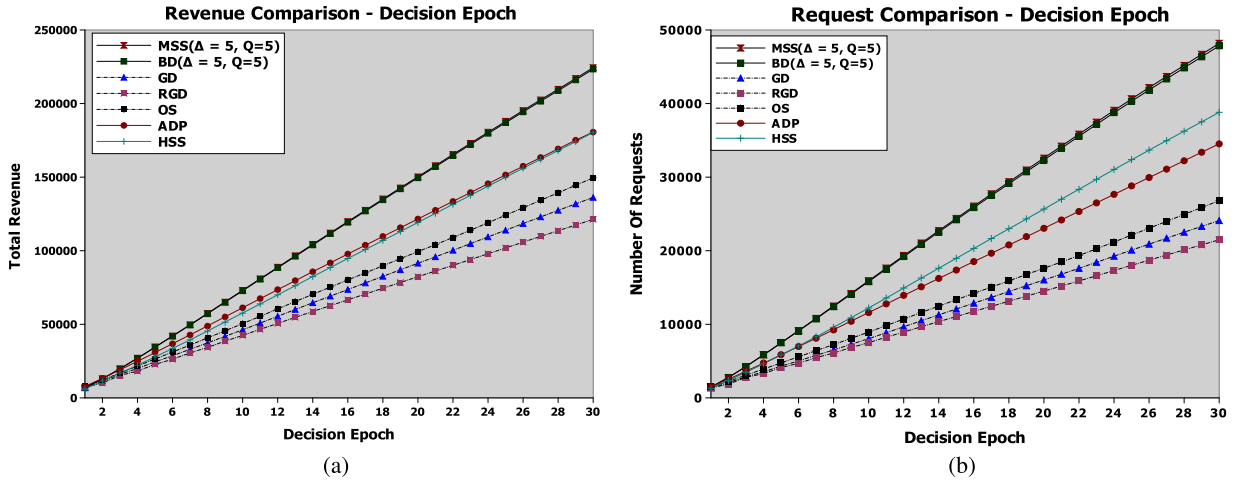


Fig. 6. Comparison of revenue earned and number of requests served using different algorithms at each decision epoch. $|\mathcal{Z}| = 436$, $|\xi^D| = 10$, $\mathcal{N}_i^1 = 2000$ (a), (b) NYDataset.

- With respect to the number of requests served, on NYDataset, **MSS** and **BD** serve 21 381 additional requests as compared to the myopic approaches and they serve 9378 additional requests as compared to **ADP** and **HSS**. For Dataset2, **MSS** and **BD** serve 24 000 additional requests as compared to myopic approaches and it serves additional 11 000 requests as compared to **ADP** and **HSS**. Similarly, on Dataset1, we serve 3000 additional requests as compared to the myopic approaches and 1400 additional requests as compared to **ADP** and **HSS**. This is a significant result, because in most cities at rush hours, the number of taxis is almost always lower than the actual demand available.
- On NYDataset, **ADP** and **HSS** have identical results (requests served by **HSS** is more but revenue is almost the same). On Dataset2, **HSS** slightly outperforms **ADP** and on Dataset1 **ADP** outperforms **HSS**. Dataset1 has higher variance in requests as compared to the other 2 datasets at 8AM. Therefore, for NYDataset and Dataset2, **ADP** gets almost similar samples in each iteration but with Dataset1 it gets different samples which could be the reason for better performance of **ADP** on Dataset1.

The reason for **MSS** and **BD** providing better results as compared to **HSS** is that **HSS** works with larger regions at future stages, so **MSS** and **BD** have better approximation of future as compared to **HSS**. One of the reasons for **ADP** not performing as well as **MSS** is that the decision epoch at which the request completes depends on the assigned server (as described in the Section 3) which breaks the Markov property. In addition, as described in Appendix C, possible assignment of the taxis from nearby regions makes the value function non separable across zones, but **ADP** approximation assumes this separation. Due to these reasons, the value function approximation updated using the dual variables of a relaxed linear program is not accurate.

The value function approximation used for **ADP** is linear. With piece-wise linear value function approximation, the time taken to run each optimization increases, so it is not possible to perform significant number of iterations using piece-wise linear value function approximation. The results obtained using the piece-wise linear value function approximation with smaller number of taxis are similar to the linear value function approximation. Therefore, we use linear value function approximation for reporting the results.

Peak and non-peak time: To evaluate the sensitivity of performance of algorithms with respect to the time of the day, we now compare the competitive ratio of algorithms in the morning (8:00AM), evening (06:00PM) and afternoon (03:00PM) for all datasets. For NYDataset, as shown in the next section, the distribution of requests remains almost same throughout the day and the number of requests were low only at the night time. Therefore, for NYDataset, we also compare the competitive ratio at 12AM.

For Dataset1, the average number of requests at each decision epoch is 700 at 08:00AM, 200 at 03:00PM and 350 at 06:00PM. We can observe in the Fig. 7 that at 03:00PM, when there are more taxis as compared to the number of requests, the gap between the competitive ratio of **MSS**, **BD** and myopic approaches is less than 10% but at 08:00AM and 06:00PM, on an average the gap between the competitive ratio of **MSS**, **BD** and myopic approaches is more than 25%, the gap between **MSS**, **BD** and **ADP** is nearly 12% and the gap with **HSS** is nearly 19%.

Similar results are observed on Dataset2. For Dataset2, the number of available taxis is taken as observed in the data. The average number of requests at each decision epoch is 2000 at 08:00AM, 1700 at 03:00PM and 1800 at 06:00PM. We can observe that at 3PM the gap between the competitive ratio of **MSS**, **BD** and other approaches is less than 1% due to

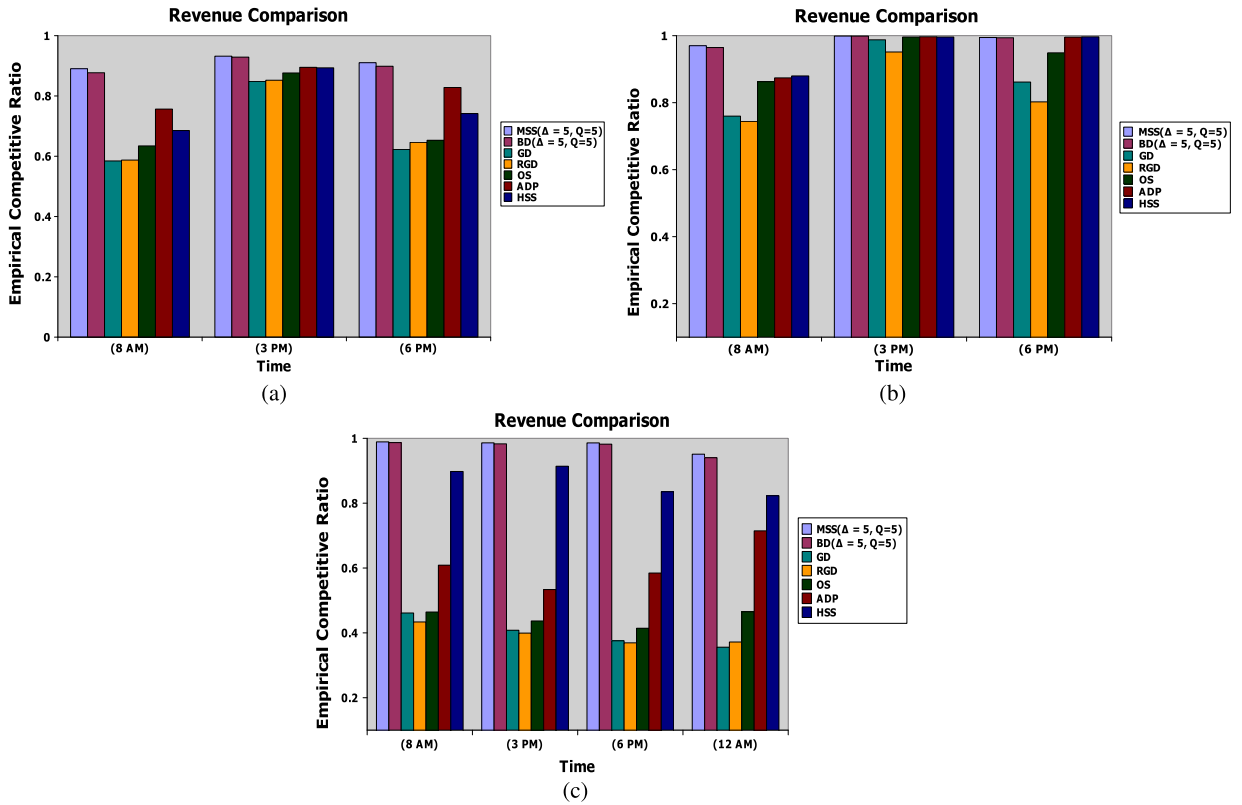


Fig. 7. Comparison of empirical competitive ratio of algorithms during peak and non peak time, $|\xi^D| = 10$, (a) Dataset1 $\mathcal{N}_i^1 = 2000$, (b) Dataset2 $\mathcal{N}_i^1 = \text{As observed in data}$, (c) NYDataset $\mathcal{N}_i^1 = 1000$.

fewer requests and more available taxis (nearly 8000 taxis at 3PM as compared to 5500 taxis at 08:00AM). At 08:00AM the gap is nearly 10% from all the approaches.

For NYDataset, the average number of requests at each decision epoch is 1941.8 at 08:00AM, 1726.1 at 03:00PM, 2407.617 at 06:00PM and 712.88 at 12:00AM. Fig. 7 shows the comparison of the competitive ratio of algorithms at different times. In NYdataset, we did not observe the competitive ratio of myopic algorithm improve with fixed number of taxis at different time but the competitive ratio of ADP improves at 12:00AM. This improvement is more likely due to variance in the samples provided. As shown in the Section 6.4, NYDataset has low mean and high variance in the number of requests at night time. The gap between the competitive ratio of **MSS**, **BD** and myopic approaches is more than 40% at all times. The maximum gap between **MSS**, **BD** and **ADP** is 40% and minimum gap is 20%. The maximum gap between **MSS**, **BD** and **HSS** is 12% and minimum gap is 7%.

Number of taxis: Finally, we also compare all the algorithms by experimenting with different number of taxis (Figs. 8 and 9). We observe that as the number of taxis increase, the gap between **MSS**, **BD** and other approaches decreases. This is because when more taxis are available, taxis will be free even after executing assignments at the current decision epoch, so future demands can be met irrespective of the current assignment. Furthermore, when there are significantly more taxis than the demand, sophisticated matching approaches are not required.

We also compare the run-time of algorithms with different number of taxis. Fig. 10 shows the run-time of **MSS** and **BD** with different number of taxis. We observed that when the number of taxis are much less or taxis are available in excess, run-time of **MSS** is lower as compared to the case when the number of taxis is comparable to the available requests. For Dataset1, **MSS** has maximum run-time at 2000 taxis and for the other 2 datasets it has maximum run-time for 5000 taxis. Though run-time of **MSS** changes with the number of taxis, run-time of **BD(P)** remains almost same irrespective of the number of taxis.¹⁴ Moreover we conducted our experiments on academic systems, on commercial servers this will take much less time.

¹⁴ Since, the reported run-time for **BD** is based on sequential solving of slaves, the actual run-time when slaves are run in parallel on multiple cores is lower. We show the expected time which will be taken by **BD**, if slaves are solved in parallel (denoted by **BD(P)** in the figure). We calculated this time by considering the time taken to solve slaves in parallel as the maximum time taken by any slave in each iteration.

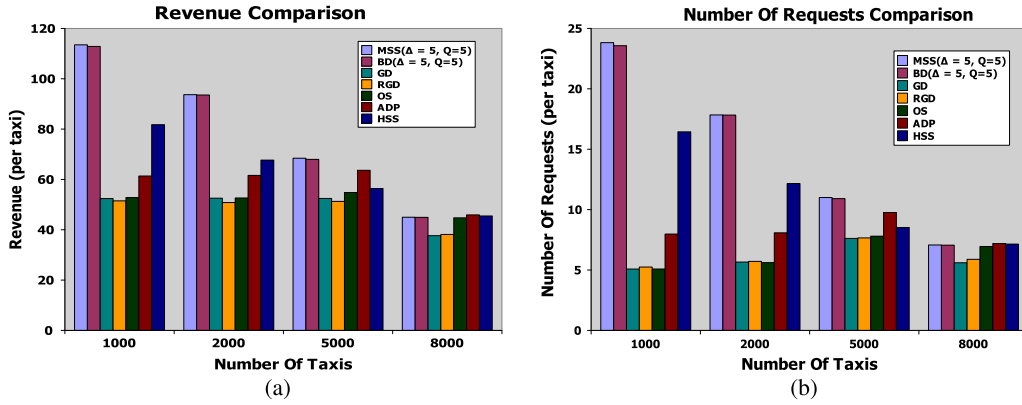


Fig. 8. Comparison of the revenue earned and the number of requests served using different algorithms. NYDataset: In (a) and (b) $|\mathcal{Z}| = 436$, $|\xi^D| = 10$.

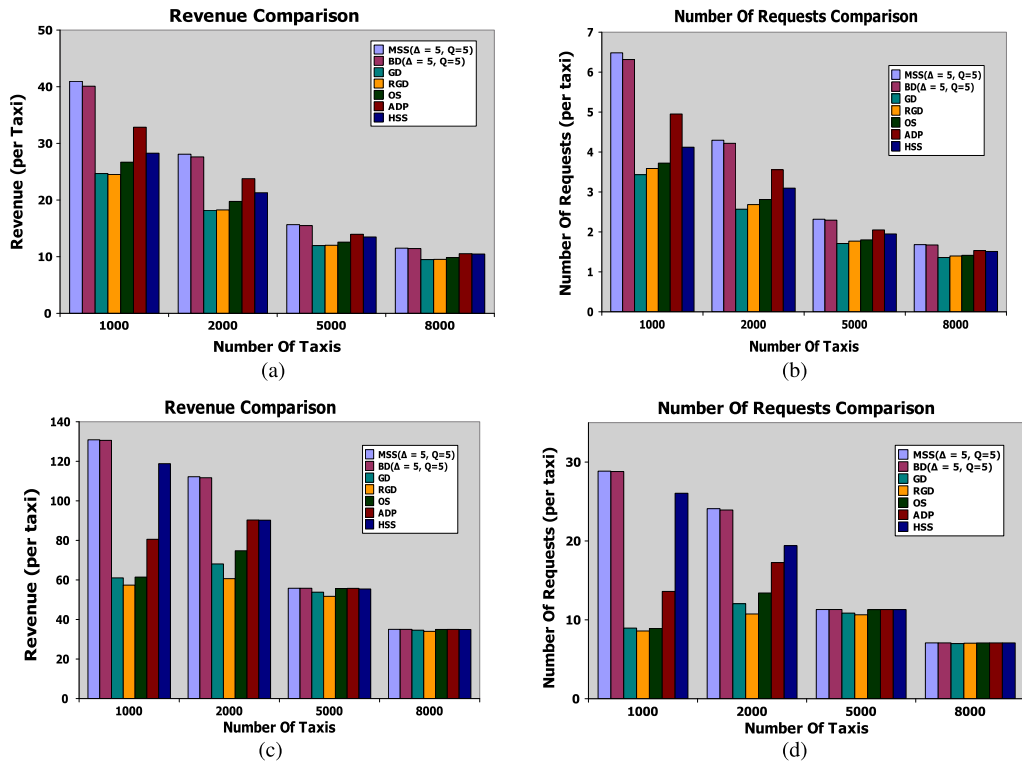


Fig. 9. Comparison of the revenue earned and the number of requests served using different algorithms. (a), (b) Dataset1 and (c), (d) Dataset2. In (a), (b), (c) and (d) $|\mathcal{Z}| = 483$, $|\xi^D| = 10$.

6.4. Justification for values of parameter settings

In this section, we show the reason for using the Δ value as 5 and the number of samples as 10 in the previous section. **Decision epoch length (Δ):** We identify the appropriate value of Δ which provides the right trade-off between solution quality and run-time. The solution quality increases with decreasing the value of Δ . This is because a server is assigned to only one request in each decision epoch (Section 5). Therefore, for large Δ values, even if a server can serve more than one request in Δ duration, it will be serving only one request. To observe the change in solution quality on decreasing the value of Δ , we compare the objective value of OSTM over 2.5 hours for different value of Δ on both datasets.

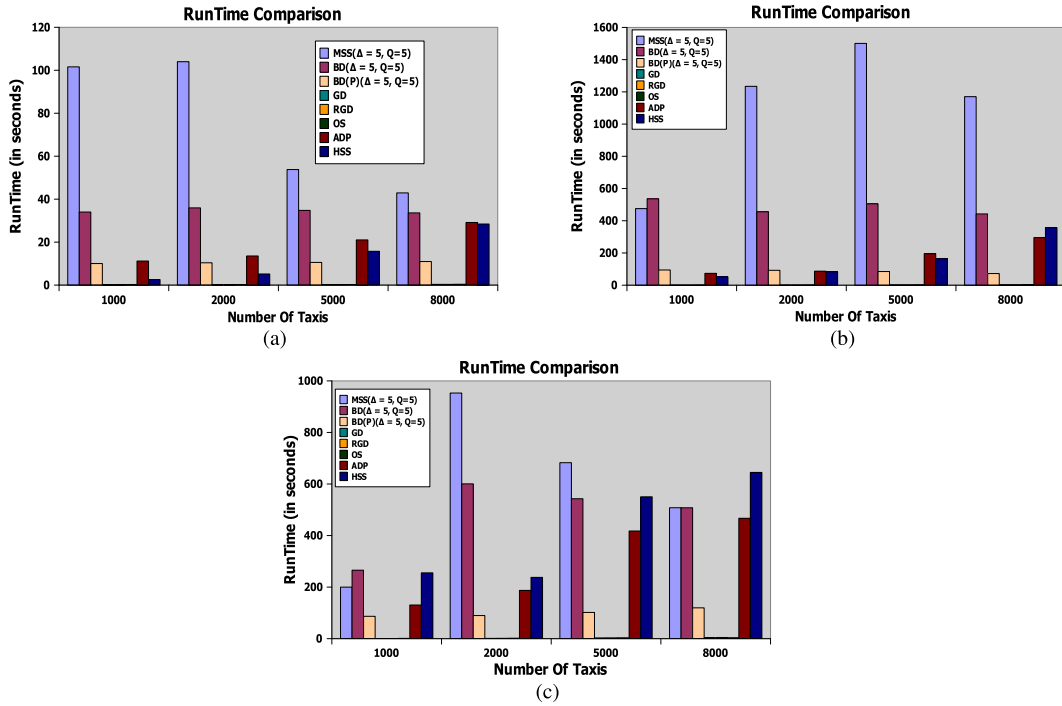


Fig. 10. Run time comparison of our algorithms for different number of taxis (a) $|\mathcal{Z}| = 483, |\xi^D| = 10$ Dataset1, (b) $|\mathcal{Z}| = 483, |\xi^D| = 10$ Dataset2, (c) $|\mathcal{Z}| = 436, |\xi^D| = 10$ NYDataset.

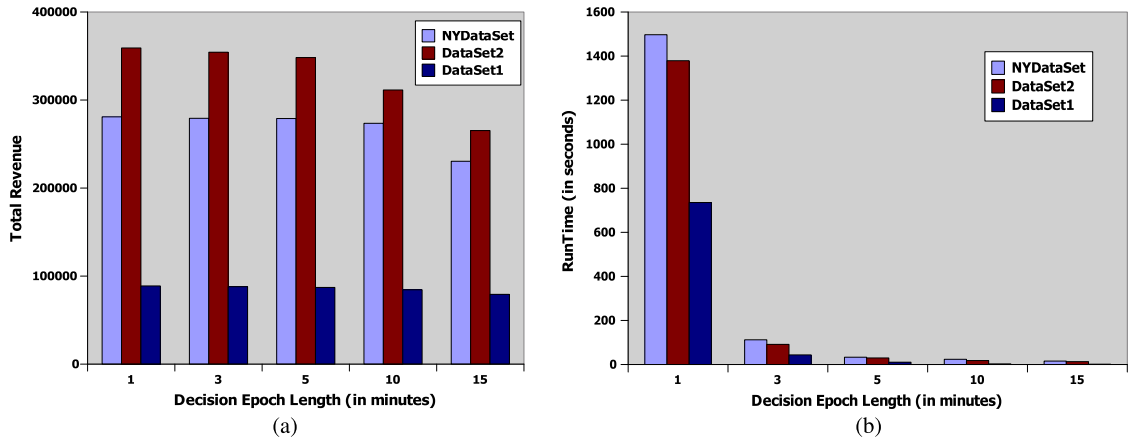


Fig. 11. Revenue and the run-time comparison of the Offline algorithm on different datasets for different decision epoch length.

Fig. 11 shows the comparison of total revenue earned and run-time of OSTM for different Δ values (in minutes) on all the datasets.¹⁵

As mentioned in the Section 6.2, the value of Q is taken such that $\Delta * (Q + 1) = 30$ minutes, i.e., if Δ is 1, Q is taken as 29 and if Δ is 15, Q is taken as 1.

Fig. 11a shows the total revenue obtained on executing OSTM. We observe that on all the datasets, there is a major increase in revenue on decreasing the value of Δ from 15 to 10 and 10 to 5. But the increase in revenue on decreasing the value of Δ from 5 to 3 and 3 to 1 is comparatively less. Fig. 11b shows the maximum time taken to compute a single assignment with 1 sample of future demand. We observe that the time taken to compute an assignment, drastically increases on decreasing the value of Δ from 3 to 1. This is due to the large increase in the number of constraints in the

¹⁵ The average number of requests in Dataset 2 and NYDataset is more than twice the average number of requests in Dataset1 (nearly 60000 requests in Dataset2 and NYDataset as compared to 30000 requests in Dataset1). Therefore, there is a significant difference between total revenue earned and time taken to compute an assignment.

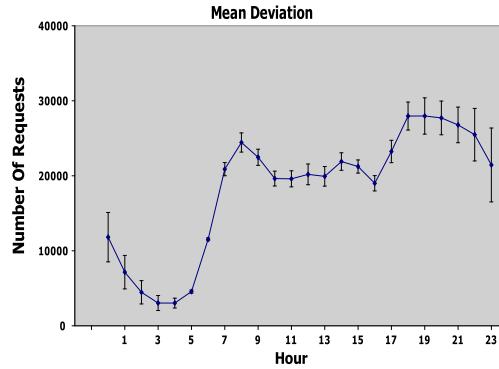


Fig. 12. Mean and deviation in the number of requests available at each hour for NYDataset.

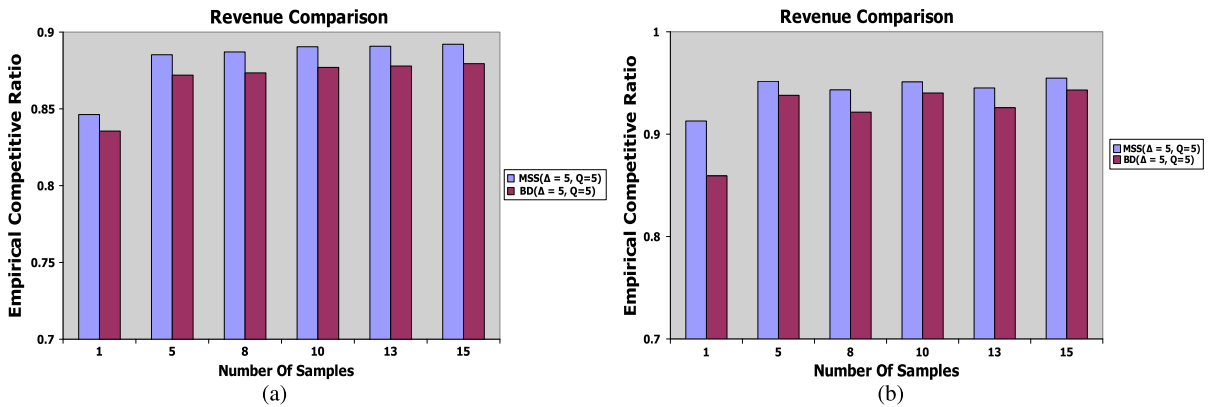


Fig. 13. Comparison of empirical competitive ratio of algorithms for different number of samples. (a) Dataset1: $|\mathcal{Z}| = 483, \sum_{i \in \mathcal{Z}} \mathcal{N}_i^1 = 2000$, (b) NYDataset : $|\mathcal{Z}| = 436, \sum_{i \in \mathcal{Z}} \mathcal{N}_i^1 = 1000$.

optimization formulation. Even for $\Delta = 3$, the time taken to compute a single assignment with one sample is 100 seconds on Dataset2.

As the time taken is high for $\Delta < 5$ and quality of solution at $\Delta = 5$ is comparable to the solution quality at $\Delta = 1$, we take the value of Δ as 5 for our experiments.

Number of samples ($|\xi^D|$): Our next set of experiments measure the performance of our approaches with respect to change in the number of samples. Fig. 12 shows the mean and deviation in the number of requests available at each hour over 15 weekdays in NYDataset. From the figure, we can observe that there is a high variance in the number of requests at night between 10:00PM to 02:00AM, but during the day, the variance is low. Therefore, during the day, even with a single sample, we can obtain good quality results. Due to this reason, to observe the effect of using different number of samples for NYDataset, we provide the results at 12:00AM (as there is low mean and high variance at 12:00AM). On the other hand, Dataset1 has high variance in the number of requests in the morning at 8:00AM on weekday (minimum: 6000, maximum: 60000 over 2.5 hours). Therefore, for Dataset1, we provide the results at 08:00AM. The average number of requests at each decision epoch is 700 for Dataset1 and 712.88 for NYDataset. The total number of taxis is set to 1000 for NYDataset and 2000 for Dataset1.

Fig. 13 provides the results for the average value of empirical competitive ratio for Dataset1 and NYDataset. In Fig. 13a and 13b, X-axis denotes the number of samples considered while computing the assignment and Y-axis denotes the empirical competitive ratio (from Section 4.4). The key observations are as follows:

- (1) The most significant result is that **MSS** and **BD** are able to achieve more than 88% of the optimal revenue with 10 samples on Dataset1 and more than 95% on NYDataset. Even though both datasets have high variance in the number of requests, Dataset1 has higher variance. Moreover, Dataset1 has more requests within the same pair of zones which makes a single zone more prominent than the others. So in case of Dataset1, the important zones for different samples are different and can be completely different from the test day. On the other hand, requests are more distributed in NYDataset, therefore, NYDataset has higher empirical competitive ratio as compared to the Dataset1.
- (2) Even with 1 sample, on an average **MSS** and **BD** obtain 84% of the optimal revenue on the Dataset1 and more than 85% of the optimal revenue on NYDataset.

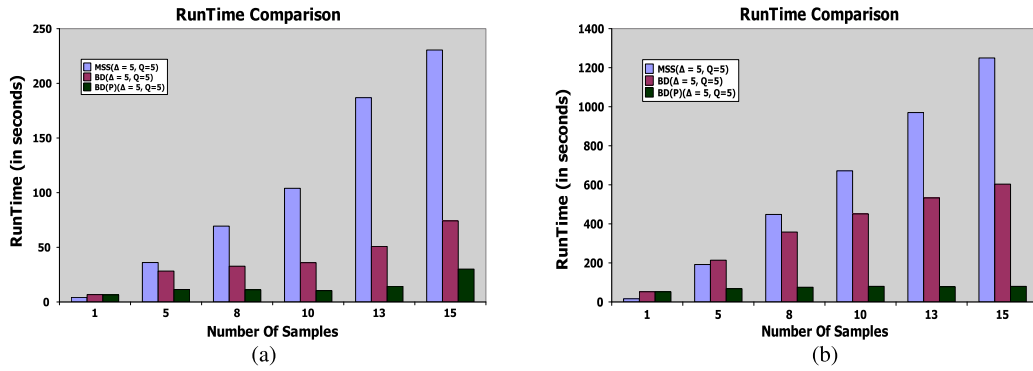


Fig. 14. Comparison of run time of algorithms for different number of samples. (a) Dataset1: $|\mathcal{Z}| = 483$, $\sum_{i \in \mathcal{Z}} \mathcal{N}_i^1 = 2000$, (b) NYDataset : $|\mathcal{Z}| = 436$, $\sum_{i \in \mathcal{Z}} \mathcal{N}_i^1 = 1000$.

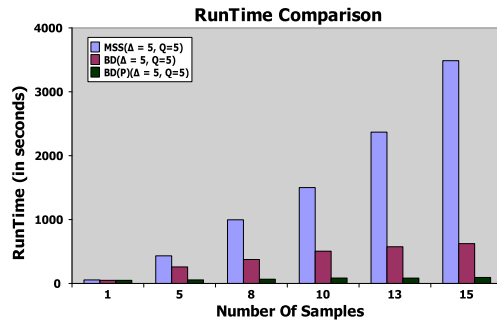


Fig. 15. (a) Dataset2: $|\mathcal{Z}| = 483$, $\sum_{i \in \mathcal{Z}} \mathcal{N}_i^1 = 5000$.

- (3) With respect to the revenue, on Dataset1, on an average, **MSS** provides 1.5% additional improvement over **BD**, while on NYDataset, with higher number of samples, **MSS** provides 1% improvement over **BD**.
- (4) **BD** provides the right trade-off between run-time and solution quality. While **MSS** provides high quality solutions (especially for Dataset1), it can take significantly more time (up to 1500 seconds when making online decisions).
- (5) **BD(P)** on an average takes 24 seconds with 10 samples to compute an assignment for Dataset1 and 83 seconds with 10 Samples for NYDataset.

On the other hand, there is very low variance over the requests on different days in Dataset2 (minimum:56000, maximum:60000 over 2.5 hours). Therefore, the empirical competitive ratio is more than 95% even on taking only one sample and does not improve much on adding more samples. The run-time results on Dataset2 are similar to NYDataset. NYDataset and Dataset2 take more time to compute an assignment as compared to Dataset1 because as discussed before Dataset1 has more requests between same zone pairs while requests are more distributed in other 2 datasets. Therefore, the number of zone pairs which have requests between them is much lower for Dataset1. Due to this, the overall size of linear program is larger for other two datasets resulting in higher run-time. (See Figs. 14 and 15.)

On all the datasets, **MSS** and **BD** provide high value of competitive ratio with very few samples. One of the major reasons for this is that the samples need not be exactly similar to the test data to provide a gain in the competitive ratio. As mentioned before, a taxi can be assigned to a request if it can reach the pickup location of request within 5 (τ) minutes and the cost of traveling to reach the request pickup location is negligible as compared to the revenue obtained. Therefore, unless samples are drastically different, they will contribute a lot in increasing the competitive ratio.

6.5. Synthetic scenarios where **MSS** and **BD** do not improve performance

To better understand the settings where our approaches work well and where it does not, we also performed experiments on the synthetic datasets. We investigate the following cases, where our approaches can potentially yield bad results as compared to the myopic approaches:

1. Requests between zones generated using uniform, binomial and Poisson distributions: For this setting, we obtained similar results (as on the real world data sets) with **MSS** and **BD** outperforming myopic approaches. On synthetic data sets with uniform distribution, **MSS** and **BD** do not get the advantage of making taxis available in the locations with more demand (based on samples) as requests are available in all the zones. The reason behind **MSS** and **BD** still outperforming myopic approaches in this scenario is the revenue model which rewards shorter trips over longer trips.

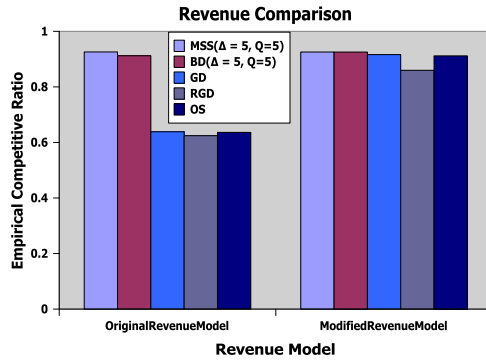


Fig. 16. Revenue model with no bias for short trips: $|\mathcal{Z}| = 50$, $\sum_{i \in \mathcal{Z}} \mathcal{N}_i^1 = 2000$, $\frac{1}{30} \sum_{t=1}^{30} \mathcal{D}^t = 2507$.

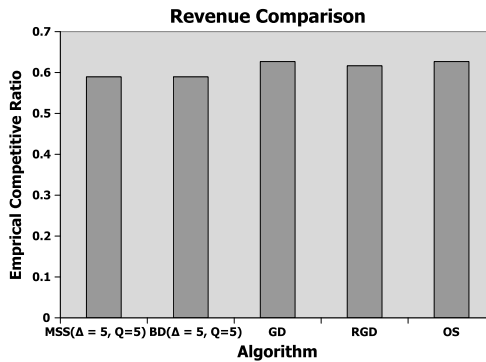


Fig. 17. High proportion of long distance trips with far apart zones: $|\mathcal{Z}| = 5$, $\sum_{i \in \mathcal{Z}} \mathcal{N}_i^1 = 10$, $\sum_{t=1}^{30} \mathcal{D}^t = 21$.

2. Revenue model with no bias for short trips (over long trips): As myopic approaches consider the revenue for only the current decision epoch, they will assign taxis to a trip having long distance. On the other hand, **MSS** and **BD** will prefer serving multiple short distance trips. When we change the revenue model in equation (46) such that the revenue is directly proportional to the distance, **MSS** and **BD** do not get the advantage by serving multiple short distance trips. On synthetic dataset with uniform distribution of requests and the modified revenue model, all algorithms have comparable value of the competitive ratio, as shown in Fig. 16.

3. High proportion of long distance trips with far apart zones: To simulate this scenario, we had to create a specific setting that is hard to replicate in real world data sets. We took 5 zones and no request has an origin and destination in the same zone. The time taken to travel between zones is taken such that the taxis need to be in exactly the same zone to serve the requests. In the 5 zones considered, we take the minimum time taken to travel between any pair of zones as 15 minutes and maximum time as 40 minutes. Out of 20 possible (5*4) zone pair combinations, time taken to travel between 6 zone pairs is taken as more than 30 minutes (revenue 12.5), between 6 other zone pairs as 15 minutes (revenue 7.2), 4 other zone pair as 20 minutes (revenue 8.2) and between remaining 4 zone pairs as 25 minutes (revenue 10.5). Requests are generated such that the requests with travel time 15, 20 and 25 minutes are not available at all decision epochs but the requests having travel time 30 minutes or more are available at all decision epochs. The value of τ and Δ is 5 minutes and the value of Q is 5, i.e., $\Delta * (Q + 1) = 30$ minutes.

In these settings, **MSS** and **BD** can perform arbitrarily bad as compared to the myopic approaches. The intuition is that **MSS** and **BD** can keep on taking sub-optimal decision at the current decision epoch with an expectation of getting higher revenue at the next decision epochs but on reaching the next decision epoch it again takes a sub-optimal decision.

The competitive ratio of **MSS** and **BD** in this setting is constantly less than the myopic approaches even after providing multiple samples as shown in Fig. 17. The reason is as **MSS** and **BD** are only looking at the next 30 minutes, they decide to first serve a request with 15 minute travel time followed by a request having travel time 30 minutes or more. But on reaching the next decision epoch it ignores the request of higher travel time even if there are no requests with 15-minute travel time available. **MSS** and **BD** find it a better option to be idle for a couple of decision epochs in order to wait for a 15 minute request so that it can serve that and follow it with a request having travel time more than 30 minutes. This keeps on happening till the last decision epoch. As a result, **MSS** and **BD** remain idle, waiting for a short trip and the myopic approaches keep on serving the requests, resulting in higher revenue for them.

7. Conclusion

In this paper, we have presented a multi-stage stochastic formulation for online assignment of servers to customers and a Benders Decomposition of the formulation to deal with large numbers of future scenarios. We provide crucial theoretical results that indicate the complexity of the problem and the worst case performance obtained by greedy and multi-stage approaches. In addition, we have compared the assignment computed by our algorithms against multiple existing myopic and multi-stage algorithms (including an approach for solving MDPs online). We found that our algorithms are able to provide a significant gain in performance over all the current best approaches on three different taxi data sets. We also created synthetic scenarios and demonstrated certain revenue models and zone configurations (which typically do not occur in real taxi domains) where our approaches do not provide improvements.

In the future, we would like to extend this work to support queuing of requests and multi-capacity servers to address the general pickup and delivery problems with time windows. We are also exploring how anonymity and homogeneity (key ideas that are exploited in this paper) can be used to improve the performance of online MMDPs.

Acknowledgements

This work was supported by the Singapore National Research Foundation through the Singapore-MIT Alliance for Research and Technology (SMART) Centre for Future Urban Mobility (FM) and National Research Foundation, Prime Minister's Office, Singapore under its International Research Centres in Singapore Funding Initiative. The authors thank the anonymous referees and the editor for their valuable comments and suggestions.

Appendix A. Two-stage formulation of assignment problem

In this section, we provide the two-stage formulation for the pure assignment problem in the Table A.10. In this formulation, the locations are not grouped in the zones and each server and request is considered individually. Time is discretized into timesteps of duration Δ . We provide the formulation for the offline problem. The formulation does not provide a solution within 5 minutes even for 20 servers.

We use \mathcal{V} to denote the set of servers and \mathcal{W}^t to denote the set of requests at timestep t . The set \mathcal{W}^t corresponds to the element D^t in our OSTM model as both represent the demand. But as each element of the set D^t corresponds to multiple requests and set \mathcal{W}^t corresponds to individual requests, for each element in D^t , there will be multiple elements in the set \mathcal{W}^t .

x_{vw}^t denotes that the server v is assigned to the request w at time t . b_{vw}^t denotes the waiting time of the request w when it is served using server v at timestep t . a_v^t is the binary variable which is set to 1 if the server v is available at timestep t . C_{vw}^t provides the net revenue obtained on assigning the server v to the request w at time t . The revenue for the timestep $t + 1$ depends on the updated location of the server as shown in the constraint A.8. Constraint (A.2) ensures that the server is assigned to atmost one request at each timestep and constraint (A.3) ensures that the request is assigned to atmost one server. Constraint (A.4) ensures that the waiting time of any request is less than τ . Constraint (A.5) computes the waiting time of request depending on the updated position of server. Constraint (A.7) checks the availability time of server depending on its updated position.

Table A.10

Two-stage formulation for assignment problem.

$$\max \sum_v \sum_w \sum_{t=1}^2 C_{vw}^t * x_{vw}^t \quad (\text{A.1})$$

$$\sum_v x_{vw}^t \leq 1 \quad \forall w, t \quad (\text{A.2})$$

$$\sum_w x_{vw}^t \leq 1 \quad \forall v, t \quad (\text{A.3})$$

$$\sum_v b_{vw}^t * x_{vw}^t \leq \tau \quad \forall w, t \quad (\text{A.4})$$

$$b_{vw}^{t+1} = (1 - \sum_{w'} x_{vw'}^t)[T_{vw}(v, w, t + 1)] + (x_{vw'}^t)[T_{ww}(w', t, w, t + 1)] \quad \forall v, w \quad (\text{A.5})$$

$$x_{vw}^t \leq a_v^t \quad \forall v, w, t \quad (\text{A.6})$$

$$a_v^{t+1} = 1 \text{ if } \sum_w x_{vw}^t [t + T_{vw}(v, w, t)] + T_w(w, t) + (1 - \sum_w x_{vw}^t) * a_v^t \leq t + 1 \quad \forall v \quad (\text{A.7})$$

$$C_{vw}^{t+1} = (1 - \sum_{w'} x_{vw'}^t)[rev(w, t + 1) - c_{vw}(v, w, t + 1)] + (x_{vw'}^t)[rev(w, t + 1) - c_{ww}(w', t, w, t + 1)] \quad (\text{A.8})$$

$T_{vw}(v, w, t)$ denotes the time taken to travel from server's initial location at t to the pick up location of request w at timestep t . $T_{ww'}(w', t, w, t + 1)$ denotes the time taken to travel from the drop location of request w' at t to the pickup location of request w at $t + 1$. $T_w(w, t)$ denotes the travel time of request w at timestep t .

Similarly $c_{vw}(v, w, t)$ denotes the cost to travel from server's initial location at t to the pick up location of the request w at timestep t . $c_{ww'}(w', t, w, t + 1)$ denotes the cost to travel from the drop location of the request w' at t to the pickup location of the request w at $t + 1$. $rev(w, t)$ denotes the revenue of request w at time t .

Extending this formulation to multiple timesteps further increases the complexity. Linearizing the product of b_{vw}^{t+1} and x_{vw}^{t+1} we can write, (M is a large constant)

$$z_{vw}^{t+1} = b_{vw}^{t+1} * x_{vw}^{t+1} \quad (\text{A.9})$$

$$z_{vw}^{t+1} \leq M * x_{vw}^{t+1} \quad (\text{A.10})$$

$$z_{vw}^{t+1} \leq b_{vw}^{t+1} \quad (\text{A.11})$$

$$z_{vw}^{t+1} \leq b_{vw}^{t+1} - (1 - x_{vw}^{t+1}) * M \quad (\text{A.12})$$

Similarly, we need to linearize the product of C_{vw}^{t+1} and x_{vw}^{t+1} and the constraint (A.7).

Appendix B. Hybrid multi-stage stochastic optimization (HSS)

In this section, we provide the details and the formulation of the model used by Powell et al. [10]. This formulation solves the assignment problem at first stage and a pure network flow approximation at the future stages. As for the truckload problem, the typical value of time step is one day, therefore, even after having an assignment problem at first stage, this formulation will be a pure network for the truckload assignment using a single sample of future demand. As described in Sections 2.2 and 3, in our case, due to smaller timesteps and specifically as the completion timestep of the customer request depends on the server assigned, this formulation will not be a pure network. But using this formulation for our case, allows us to measure the impact of using more accurate information for future demands. As opposed to the formulation and experimental evaluation in [10], we use multiple samples of future demand for this approach.

Table B.11 presents the Hybrid Multi-Stage Stochastic optimization formulation for our problem. As *HSS* considers pure assignment problem at the current timestep, we use \mathcal{V} to denote the set of servers and \mathcal{W} to denote the set of requests at current timestep. The set \mathcal{W} in this model and D^1 in our model both represent current demand but as each element of D^1 corresponds to multiple requests and as the set \mathcal{W} corresponds to individual requests, for each element in D^1 , there will be multiple elements in the set \mathcal{W} . x_{vw} denotes that a server v is assigned to the request w . We set x_{vw} to 0 if the server v can not reach the pickup zone of the request w within τ minutes. u_v^1 denotes that the server v is held at its initial zone.

Table B.11
HSS.

HSS($\mathcal{V}, \mathcal{Z}, \mathcal{W}, C, \xi^D, \xi^S, f, g, T, \delta, Q$):

$$\max \sum_{v \in \mathcal{V}} \sum_{w \in \mathcal{W}} C_{o_v, o_w, d_w}^1 * x_{vw}^1 + \frac{1}{|\xi^D|} \sum_{k \leq |\xi^D|} \sum_{t=2}^{Q+1} \sum_{j \in \xi_t^{D,k}} C_{o_j, o_j, d_j}^t * x_{o_j, d_j}^{t,k} \quad (\text{A.13})$$

$$\text{s.t.} \sum_{w \in \mathcal{W}} x_{vw}^1 + u_v^1 = 1 \quad \forall v \in \mathcal{V} \quad (\text{A.14})$$

$$\sum_{v \in \mathcal{V}} x_{vw}^1 \leq 1 \quad \forall w \in \mathcal{W} \quad (\text{A.15})$$

$$z_{o_j, d_j}^{t,k} \leq R_j^{t,k} \quad \forall k \leq |\xi^D|, j \in \xi_t^{D,k}, \forall t > 1 \quad (\text{A.16})$$

$$y_i^{t,k} = y_i^{t-1,k} + \sum_{t'=2}^{t-1} \sum_{j \in \xi_{t'}^{D,k}, d_j=i} z_{o_j, i}^{t',k} * \delta_{o_j, i}^{t',t} + \xi_i^{S,t} + \sum_{v \in \mathcal{V}} \sum_{\substack{w \in \mathcal{W}, \\ d_w=i}} x_{vw}^1 * \delta_{o_v, i}^{1,t} - \sum_{\substack{j \in \xi_t^{D,k}, \\ o_j=i}} z_{i, d_j}^{t,k} \quad \forall i \in \mathcal{Z}, k \leq |\xi^D|, \forall t > 1 \quad (\text{A.17})$$

$$y_i^{1,k} = \sum_{v \in \mathcal{V}, o_v=i} u_v^1 + \sum_{t'=2}^{t-1} \sum_{j \in \xi_{t'}^{D,k}, d_j=i} z_{o_j, i}^{t',k} * \delta_{o_j, i}^{t',1} + \xi_i^{S,1} - \sum_{\substack{j \in \xi_t^{D,k}, \\ o_j=i}} z_{i, d_j}^{1,k} \quad \forall i \in \mathcal{Z}, k \leq |\xi^D| \quad (\text{A.18})$$

$$0 \leq x_{vw}^1 \leq 1 \quad \forall v \in \mathcal{V}, w \in \mathcal{W} \quad (\text{A.19})$$

$$0 \leq u_v^1 \leq 1 \quad \forall v \in \mathcal{V} \quad (\text{A.20})$$

$$z_{o_j, d_j}^{t,k} \quad \text{integer} \quad \forall k \leq |\xi^D|, j \in \xi_t^{D,k}, \forall t > 1 \quad (\text{A.21})$$

$$y_i^{t,k} \quad \text{integer} \quad \forall i \in \mathcal{Z}, \forall k \leq |\xi^D|, \forall t > 1 \quad (\text{A.22})$$

$z_{o_j, d_j}^{t,k}$ denotes the number of servers moving from the zone o_j to the zone d_j at time t in the sample k . $y_i^{t,k}$ denotes the number of servers held at the zone i at timestep t in sample k . We abuse the notation a bit and also use $\delta_{ij}^{t,t'}$ as a binary constant which is 1 if the server starting at the decision epoch t from the zone i reaches zone j exactly at the decision epoch t' . We use o_v to denote the initial zone of server v and o_w and d_w to denote the origin and destination zones of the request j .

Constraint (A.14) ensures that either the server is assigned to atmost one request or it is held at its initial position. Constraint (A.15) ensures that a request is assigned to utmost one server. Constraint (A.16) ensures that for any sample at any timestep, the number of servers moving between zones is equal to the number of available requests between those pair of zones. Constraint (A.17)–(A.18) ensure that the number of servers held at any zone at any decision epoch is the difference between the number of available servers in the zone and the number of assigned servers from the zone.

As at the future stages, servers get assigned to requests if and only if they are present in the same zone, we use zones of large size (2 km) for HSS formulation at future stages as opposed to 0.5 km for our formulation. We did some experiments with 0.5 km zones as well but results were worse than the results obtained with 2 km zone size.

Appendix C. Approximate dynamic programming (ADP)

In this section, we provide the Approximate Dynamic Programming formulation for our problem of matching servers with the customer demand. The modeling is similar to the stochastic dynamic resource allocation formulation presented by Godfrey et al. [32] for multi-timestep travel time. The modeling uses the language of dynamic resource management where the servers are resources and the customers are tasks. We now describe the different dimensions of the model We define resources (servers) using the following vector

A_t = the total number of servers that we know about at time t .

$A_{it'}$ = the total number of servers that we know about at time t but will become available at time t' , with $A_{it'}$ denoting the number of servers in the zone i .¹⁶

D^t = the customer demand at time t where each element j of D^t has attributes origin zone (o_j), destination zone (d_j) and the number of requests between the origin and destination zone denoted by R_j . As exact demand is known only for the current time step, $D^t = \phi, t > 1$.

Let ξ_t^D represent the new demand arriving in time step t , where $\xi_t^{D,k}$ representing the k th sample or scenario.

$D^{t,+}$ = total customer demand at time t including the new customer demand that first become available in time step t .

Therefore, while solving for sample k , $D^{t,+}$ includes $\xi_t^{D,k}$.

Therefore, the system state is given by

$$S_t = (A_t, D_{t,+})$$

x_{ij}^t are the decision variables denoting the number of servers assigned to the demand element j at time t . We use Q to denote the planning timesteps or look ahead timesteps. y_i^t are the intermediate variables denoting the number of servers in the zone i at time t .

Therefore, the dynamic programming recursion is given by

$$V_t(S_t) = E \left[\max_{x_t, y_t} g_t(x_t, y_t) + V_{t+1}(S_{t+1}) | S_t \right] \text{ where } g_t(x_t, y_t) \text{ denotes the single step reward/objective function value.}$$

We then drop the expectation and solve for a single sample at a time. $V_t(S_t, \xi_t^{D,k}) = \left[\max_{x_t(\xi_t^{D,k}), y_t(\xi_t^{D,k})} g_t(x_t(\xi_t^{D,k}), y_t(\xi_t^{D,k})) + V_{t+1}(S_{t+1}(\xi_t^{D,k})) \right]$.

C.1. Value function approximation

The value function defined above is replaced by a suitable approximation. In general, the value function approximation is defined in terms of resources and it works well when the tasks should be served as soon as possible and in case the tasks are not served they will not be available at the next time step. This holds true for our case where if the demand is unserved, it is removed from the system. Therefore, the value function approximation can be written as

$V_t(A_t, \xi_t^{D,k}) = \left[\max_{x_t(\xi_t^{D,k}), y_t(\xi_t^{D,k})} g_t(x_t(\xi_t^{D,k}), y_t(\xi_t^{D,k})) + V_{t+1}(A_{t+1}(\xi_t^{D,k})) \right]$ The problem is solved by executing a forward pass through time, determining the set of decisions for a single sample at a time. As we are solving for a single sample at a time, we can drop the samples symbol from the value function.

Godfrey et al. [32] proposed a separable approximation, for the multi-timestep travel time, of the form

$$V_t(A_t) = \sum_{i \in \mathcal{Z}} \sum_{t'=1}^Q \left[V_{t,t+t'}(A_{t,t+t'}) \right].$$

As the travel times span multiple timesteps, some servers are next available for assignment at time $t + 1$, some at time $t + 2$ and so on. The function $V_{t,t+t'}(A_{t,t+t'})$ estimates the expected value of only those servers which are next available

¹⁶ We can relate this with the variables in OLSTM definition, if t represent current time step then $A_{it'v} = N_i + \xi_i^{S,t'}$.

Table C.12

Formulation using the Linear value function approximation.

$$\max \sum_{i \in \mathcal{Z}} \sum_{j \in D^t} C_{ij}^t * x_{ij}^t + \sum_{t'=t+1}^Q v_{i,t'}^{t+1} * y_i^{t'} \tag{C.1}$$

$$\sum_i x_{ij}^t \leq R_j^t \quad \forall j \in D^{t,+} \tag{C.2}$$

$$\sum_j x_{ij}^t \leq A_{itt} \quad \forall i, t \tag{C.3}$$

$$y_i^{t+1} = A_{it't} + A_{itt} - \sum_j x_{ij}^t + \sum_m \sum_{j,d_j=i} x_{mj}^t * \delta_{mj}^{t,t'} \quad \forall i, t' > t \tag{C.4}$$

$$y_i^{t'} = A_{it't} + \sum_m \sum_{j,d_j=i} x_{mj}^t * \delta_{mj}^{t,t'} \quad \forall i, t' > t + 1 \tag{C.5}$$

$$x_{ij}^t \text{ integer} \quad \forall i, j \tag{C.6}$$

$$y_i^{t'} \text{ integer} \quad \forall i', t' > t \tag{C.7}$$

for assignment at time $t + t'$. After taking the sum over all possible travel times and all zones, we get the estimated future contribution of all the servers.

Here, we would like to highlight that in our case, as the servers can be assigned from nearby zones, the value function may not remain separable across zones as the value of having a extra resource in zone i will depend on the number of resources in nearby zones.

But it is still possible to assume the separation and apply these approximations for our problem. Linear and piecewise linear value function approximations are typically used to solve these problems as the subproblem has network structure but as we show in the following subsections, in our case the linear and piecewise value function approximation do not have a network subproblem.

C.1.1. Linear value function approximation

Linear value function approximation generally works well when value of function is linear in terms of the number of resources or the number of resources in each state can be either 0 or 1. But even when this is not the case, sometimes linear approximations can work reasonably well. $V_t(A_t) = \sum_{i \in \mathcal{Z}} \sum_{t'=1}^Q [V_{i,t,t+t'}(A_{i,t,t+t'})]$ where each of $V_{i,t,t+t'}(A_{i,t,t+t'})$ is given by $v_{i,t,t+t'}^k * A_{i,t,t+t'}$ where $v_{i,t,t+t'}^k$ is the slope of the approximation at k th iteration.

The formulation which is solved at each time step for each sample is provided in the Table C.12.

As we can see, the formulation does not have a network structure due to the presence of binary constants δ in equations (C.4) and (C.5), i.e., the Markov property is broken.

Updating the value function

The slopes are updated in each forward simulation using the standard update equation as follows

$$V_{i,t,t+t'}^k = (1 - \alpha_{k-1}) * V_{i,t,t+t'}^{k-1} + \alpha_{k-1} * \pi_{i,t,t+t'}$$

where $\pi_{i,t,t+t'}$ are the dual variable corresponding to equations C.4 and C.5. We use the dual values obtained after solving the linear relaxation. α_{k-1} is the step size. We use the stepsize as $\frac{2}{4+k}$ as mentioned in [32]. For the multi-timestep travel time we perform adjustment to dual values using dual next as proposed in [32].

The complete algorithm is presented in Algorithm 1.

Algorithm 1 ADPLinear().

- 1: Initialize: Set $V + i, t, t + t', \forall t' = 0, 1, \dots, Q, \forall i, \forall t$
- 2: **Forward Simulation:**
- 3: **for** each sample s **do**
- 4: **for** $t = 0, 1, \dots, Q - 1$ **do**
- 5: Solve the optimization in the Table C.12
- 6: Store dual vectors $\pi_{t,t+t'}, \forall t'$
- 7: **for** $t = 0, 1, \dots, Q - 1$ **do**
- 8: **for** $t' = 1, \dots, Q$ **do**
- 9: Compute adjusted marginal contribution $\pi_{i,t,t'}$ using DualNext.
- 10: Update $V_{i,t+1,t'} = (1 - \alpha_{n-1}) * V_{t,n-1,t'} + \alpha_{n-1} * \pi_{i,t,t'}$

Table C.13

Formulation using Piecewise Linear value function approximation.

$$\max \sum_{i \in \mathcal{Z}} \sum_{j \in D^t} C_{ij}^t * x_{ij}^t + \sum_{k=0}^{|N|} \sum_{t'=t+1}^Q v_{i,t+1,t'}^k * y_i^{t',k} \quad (\text{C.8})$$

$$\sum_i x_{ij}^t \leq R_j^t \quad \forall j \in D^{t,+} \quad (\text{C.9})$$

$$\sum_j x_{ij}^t \leq A_{itt} \quad \forall i, t \quad (\text{C.10})$$

$$\sum_k y_i^{t+1,k} = A_{it't} + A_{itt} - \sum_j x_{ij}^t + \sum_m \sum_{j,d_j=i} x_{mj}^t * \delta_{mj}^{t,t'} \quad \forall i, t' > t \quad (\text{C.11})$$

$$\sum_k y_i^{t',k} = A_{it't} + \sum_m \sum_{j,d_j=i} x_{mj}^t * \delta_{mj}^{t,t'} \quad \forall i, t' > t + 1 \quad (\text{C.12})$$

$$x_{ij}^t \text{ integer} \quad \forall i, j \quad (\text{C.13})$$

$$y_i^{t'} \text{ binary} \quad \forall i, t' > t \quad (\text{C.14})$$

C.1.2. Piecewise linear value function approximation

When we need to estimate the value of having a quantity of some resource and the function is piecewise linear, i.e., the slopes of the function are monotonically increasing or decreasing, piecewise linear value function approximations are used. This is more suitable approximation for the problem of interest in this paper as the value of function increases linearly on having more resources but after reaching a threshold (when the number of servers is sufficient to serve the demand), the value remains constant. Therefore, the slope of value function is monotonic decreasing, i.e., we have a piecewise linear concave function. Let $|N|$ denote the total number of servers in the system, then value function approximation is given by

$V_t(A_t) = \sum_{i \in \mathcal{Z}} \sum_{t'=1}^Q \sum_{r=1}^{|N|} \left[v_{i,t,t+t'}^r(A_{i,t,t+t'}) \right]$ where each of $v_{i,t,t+t'}^r(A_{i,t,t+t'})$ is given by $\hat{v}_{i,t,t+t'}^r * A_{i,t,t+t'}$ where $\hat{v}_{i,t,t+t'}^r$ is the slope of the piecewise component r .

The formulation which is solved at each time step for each sample is provided in the Table C.13. As the slopes (v) are monotonic, it ensures that the right values are assigned to the y variables.

As we can see, the formulation does not have a network structure due to the presence of binary constants δ in the equations (C.11) and (C.12), i.e., the Markov property is broken.

Due to the presence of large number of servers ($|N|$ varies from 1000 to 8000 in our experiments) and the need to execute the formulation multiple times (multiple timesteps and multiple samples), it is not possible to execute this approximation in real time. We also tried using the alternate formulation which defines one variable for each component of the piecewise approximation instead of having the number of variables equal to the number of servers but as we need the dual values to update the value function approximation, we need to solve the linear relaxation of the formulation. The linear relaxation of this modified formulation provides worse result than the linear approximation presented previously.

Algorithm 2 ADPPiecewise().

- 1: Initialize: Set $v_{i,t,t+t'}^0 = 0$ and $u_{j,t,t+t'}^0 = 0$ as the piecewise-linear approximation for $V_{i,t,t+t'}$, $\forall t' = 0, 1, \dots, Q$, $\forall i, \forall t$, where $v_{i,t,t+t'}^k$ denotes the slope of the component k and $(u_{i,t,t+t'}^k, u_{i,t,t+t'}^{k+1})$ provides the range in which component k is active.
- 2: **Forward Simulation:**
- 3: **for** sample s **do**
- 4: **for** $t = 0, 1, \dots, Q - 1$ **do**
- 5: Solve the optimization in the Table C.13
- 6: Store dual vectors $\pi_{i,t,t'}$, $\forall i, t'$
- 7: Update value function using CAVE
- 8: **for** $t = 0, 1, \dots, Q - 1$ **do**
- 9: **for** $t' = 1, \dots, Q$ **do**
- 10: Compute the adjusted marginal contribution $\pi_{i,t,t'}$ using DualNext.
- 11: Update $V_{i,t+1,t'} = U^{CAVE}(V_{i,t+1,t'}, \pi_{i,t,t'}, A_{i,t+1,t'})$

Updating the value function

We use the CAVE algorithm used by Godfrey et al. [32] to preserve the concavity after the update of the value function using the piece wise value function approximations.

But in our case, as the formulation is not pure network due to the presence of binary constants δ in constraints (C.11) and (C.12), we need to solve the optimization multiple times to compute the left and right hand side dual values and this can not be done within the online time constraints. Similarly re-optimizations are needed to compute the numerical derivatives. Therefore, we use the dual values for the constraints obtained after solving the linear relaxation of the optimization problem in the Table C.13.

The complete algorithm is presented in the Algorithm 2.

Appendix D. Complete proofs of propositions

D.1. Proof of Proposition 1

Proposition 1: If $\forall j, i, i', t, t' \delta_{ij}^{t,t'} = \delta_{i'j}^{t,t'}$, then the OSTM is reducible to a min cost flow problem.

Proof: Min cost flow optimization problems are polynomial time solvable, as they consider deterministic unconditional flows. For the case when each request completes at a fixed decision epoch irrespective of the server assigned to it, we show that the OSTM problem also has deterministic unconditional flows and hence the optimization model for OSTM is equivalent to the min cost flow optimization model. We first describe the network corresponding to the OSTM problem.

Nodes: Nodes of the network represent the server zones and requests. Formally, we create a node for each element in the set \mathcal{Z} at each decision epoch. A node is also created for each element in $\mathcal{D}^t, \forall t$. We also create a source node S and a sink node T . Therefore, the maximum number of nodes in the network will be $(|\mathcal{Z}| + (|\mathcal{Z}| * |\mathcal{Z}|)) \cdot M + 2$.

Edges: Intuitively, we have edges to indicate the assignment (server zone to a request) and movement (origin zone to the destination zone). Specifically, we have the following edges in the network.

- An edge is created between a node corresponding to the zone i at the decision epoch t and a node for the element j of \mathcal{D}^t if $o_j \in f(i, t)$. The capacity of the edge is equal to R_j^t and the cost of the edge is $-C_{i,o_j,d_j}^t$. The flow on these edges is given by x_{ij}^t .
- An edge is created between the element j of \mathcal{D}^t and the node corresponding to the zone i at decision epoch t' ($t' > t$), if $d_j = i$ and $\delta_j^{t,t'} = 1$ ($\forall j, i, i', t, t' \delta_{ij}^{t,t'} = \delta_{i'j}^{t,t'}$, therefore we define $\delta_j^{t,t'} = \delta_{ij}^{t,t'} \forall i, j, t, t'$). The capacity of the edge is R_j^t and the cost is 0. We denote the flow on these edges by y_j^t .
- An edge is created between the node corresponding to the zone i at decision epoch t and the node corresponding to the zone i at decision epoch $t + 1$ to have the flow of unassigned servers, i.e., to ensure that the unassigned servers from the zone i at decision epoch t remain in the same zone at next decision epoch. The capacity of this edge is equal to $\sum_i N_i$ and the cost is 0. The flow on these edges is denoted by w_i^t .
- From source node S , we create an edge to all the zone nodes at the decision epoch 1. The capacity of edge between the source node S and the zone i node is N_i (the number of initial servers in zone i) and the cost is 0.
- If the requests present in the element j of $\mathcal{D}^t, \forall t$ complete at a decision epoch $> M$, then we create an edge between the element j of \mathcal{D}^t to the sink T . The capacity of the edge is R_j^t and the cost is 0.
- We also create edges from the zone nodes at decision epoch M (last decision epoch) to the sink node T . The capacity of the edge is $\sum_i N_i$ and the cost is 0.

Given this network, we can view the similarity of the two optimization models in Table C.14. The transformation uses slack variables (w_i^t) and the intermediate variables (y_j^t) to convert the original OSTM constraints into the flow and capacity constraints present in the min cost flow model. As the min cost flow is polynomial time solvable and gives integral flow values for the integer capacities, we can solve the OSTM also in polynomial time to get integer solutions. \square

Example 3. The example network with three zones and $M = 3$ is shown in the Fig. D.18. S and T denote the source and sink node. Circular nodes represent the server zone nodes and rectangular nodes represent the request nodes between zones. Requests between the zones Z_2 and Z_3 at decision epoch 1 complete at decision epoch 3. Requests between the zones Z_1 and Z_3 complete at decision epoch 4 and as $M = 3$, they are connected to sink node T . N denotes the total number of servers available initially, i.e., $N = N_1 + N_2 + N_3$. Each arc contains 2 values with the first value representing capacity and the second value representing the cost.

In general, depending on the server assigned, a request can have a maximum of $\lceil \frac{r}{\Delta} \rceil + 1$ different completion decision epochs. Therefore, in the network constructed in the Proposition 1, we will have edges between the decision epoch t request nodes to the decision epoch $t', t' + 1, \dots, t' + \lceil \frac{r}{\Delta} \rceil$ zone nodes with the flow on these edges conditioned to be equal to the flow on the edges from the server node to request nodes.

In the example shown in the Fig. D.18, suppose if the Z_2 server is assigned to the request between zones Z_2 and Z_3 , then the request completes at decision epoch 3 and if the Z_1 server is assigned to the request between zones Z_2 and Z_3 , the request will complete at decision epoch 4. The modified network is shown in Fig. D.19. Now the node Z_2, Z_3 at decision epoch 1 will have the capacity R_{23}^1 with the condition that flow on 2 green edges should be equal and flow on 2 orange edges should be equal. Proposition 2 shows that in the above case where there are conditional flows, the OSTM is NP-hard.

D.2. Proof of Proposition 2

Proposition 2: If $\exists j, i, i', t, t' s.t. \delta_{ij}^{t,t'} \neq \delta_{i'j}^{t,t'}$, then the OSTM is NP-hard.

Proof: To show that the OSTM is NP-hard in general case, we reduce the well known 3-SAT problem to OSTM. We construct an instance of the OSTM for any arbitrary instance of 3-SAT with L clauses and V variables. We show that we obtain the

Table C.14

MinCost Flow and OSTM.

OSTM ($\mathcal{Z}, \mathcal{D}, \mathcal{N}, C, f, g, T, \delta, M$):	MinCostFlow : $G(V, E)$:
Objective	
$\min \sum_{t=1}^M \sum_{i \in \mathcal{Z}} \sum_{\substack{j \in \mathcal{D}^t, \\ o_j \in g(i,t)}} -1 * C_{i,o_j,d_j}^t * x_{ij}^t$ $+ \sum_{t=1}^M \sum_{i \in \mathcal{Z}} w_i^t * 0 + \sum_{t=1}^M \sum_{j \in \mathcal{D}^t} y_j^t * 0 \quad (D.1)$	$\min \sum_{(u,v) \in E} c_{uv} * f_{uv} \quad (D.2)$
Flow constraints	
$\sum_{\substack{j \in \mathcal{D}^1, \\ o_j \in g(i,1)}} x_{ij}^1 + w_i^1 = \mathcal{N}_i \quad \forall i \quad (D.3)$	$\sum_{k (j,k) \in E} f_{jk} - \sum_{i (i,j) \in E} f_{ij} = b_j \quad \forall j \in V \quad (D.7)$
$y_j^t - \sum_{i \in f(o_j,t)} x_{ij}^t = 0 \quad \forall t, j \in \mathcal{D}^t \quad (D.4)$	
$\sum_{\substack{j \in \mathcal{D}^t, \\ o_j \in g(i,t)}} x_{ij}^t + w_i^t - w_i^{t-1}$ $- \sum_{\substack{t'=1 \\ d_j=i}}^{t-1} \sum_{j \in \mathcal{D}^{t'}} \delta_j^{t',t} * y_j^{t'} = 0 \quad \forall i, t > 1 \quad (D.5)$	
$-1 * \sum_{i \in \mathcal{Z}} w_i^M + \sum_{t'=1}^M \sum_{\substack{j \in \mathcal{D}^{t'}, \\ d_j=i}} \delta_j^{t',M+1} * y_j^{t'}$ $= -1 * \sum_{i \in \mathcal{Z}} N_i \quad (D.6)$	
Capacity constraints	
$0 \leq y_j^t \leq R_j^t \quad \forall t, j \in \mathcal{D}^t \quad (D.8)$	$0 \leq f_{ij} \leq u_{ij} \quad \forall (i, j) \in E \quad (D.11)$
$0 \leq x_{ij}^t \leq R_{ij}^t \quad \forall i, t, j \in \mathcal{D}^t \quad (D.9)$	
$0 \leq w_i^t \leq \sum_{i \in \mathcal{Z}} N_i \quad \forall i, t \quad (D.10)$	

optimal value for the OSTM if and only if there exists an assignment to variables in 3-SAT formula such that all the clauses in 3-SAT will evaluate to true.

For any arbitrary instance of 3-SAT with L clauses and V variables, we construct an instance of OSTM as follows:

- We create an OSTM problem instance with $4L + 1$ decision epochs, i.e., $M = 4L + 1$. At first decision epoch, we create $2V$ requests. At remaining $4L$ decision epochs, we create one request each.
- We create $2V + 1$ zones, denoted by Z_0 and $Z_i, Z'_i, i \leq V$. So the set of zones $\mathcal{Z} = \{Z_0, Z_i, Z'_i, i \leq V\}$.
- Initially zones Z_i have one server each and zones Z_0 and Z'_i have zero servers.
- We create $2V$ requests at first decision epoch. Each zone Z_i is the origin zone of two requests with one request having destination in Z_i and one request having destination in Z'_i . Requests having origin in Z_i at decision epoch 1, can only be assigned server from the zone Z_i . Therefore, $D^1 = \{< Z_i, Z_i, 1 >, < Z_i, Z'_i, 1 >\}$ and $f(Z_i, 1) = \{Z_i\}$. All the requests at the first decision epoch have unit revenue.
- If zone i server is assigned to the request between zone pairs $< Z_i, Z_i >$ then the variable x_i is true else if it is assigned to the request between zone pairs $< Z_i, Z'_i >$ then x_i is false. Requests served at decision epoch 1 determines the value of variables x_i . Therefore, at second decision epoch, server will be available in zone Z_i if x_i is true, and will be available in the zone Z'_i if x_i is false.

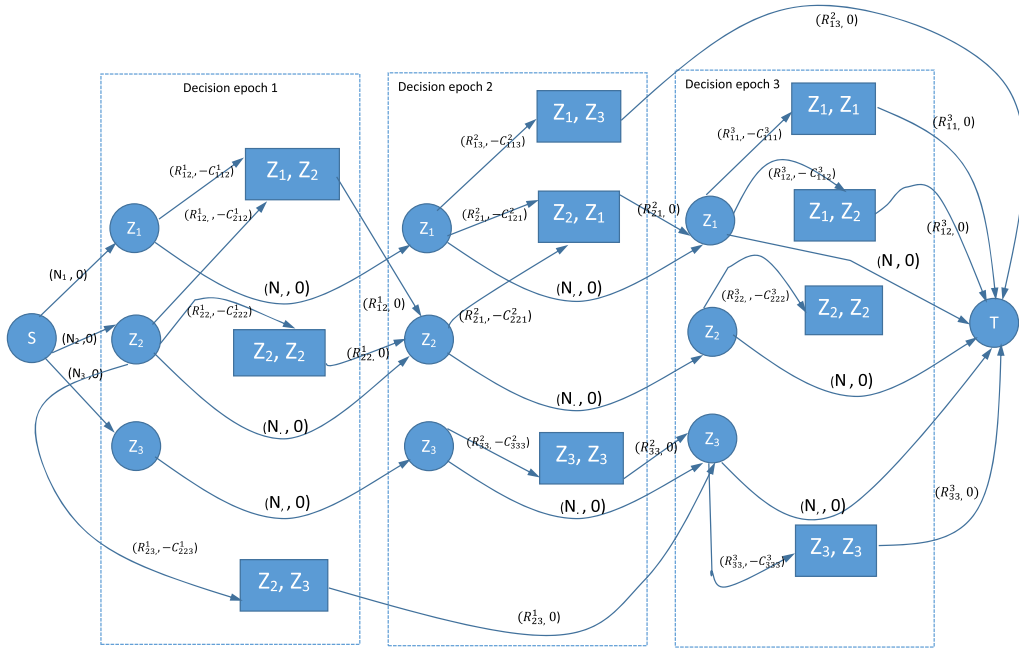


Fig. D.18. Example min cost flow network for OSTM.

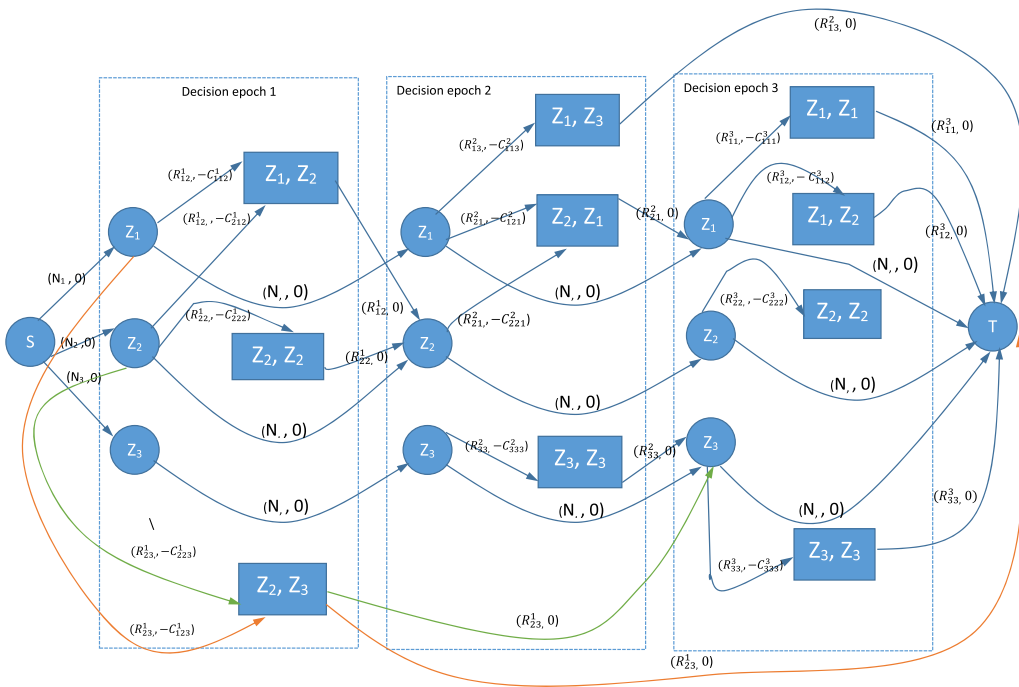


Fig. D.19. Modified example flow network for OSTM.

- If there are L clauses, we create L requests with request corresponding to k th clause at decision epoch $4k - 2$. Each of these requests have destination in the zone Z_0 . The origin zone of these requests is the zone corresponding to the first literal of the k th clause.

In addition, if the k th clause has literal x_i then the zone Z_i server can be assigned to the request at decision epoch $4k - 2$ and if the k th clause has literal $\neg x_i$, then the zone Z'_i can be assigned to the request. On the other hand, if the k th clause does not contain literal x_i then zone Z_i server can not be assigned to the request at decision epoch $4k - 2$. In short, request at the decision epoch $4k - 2$ can be assigned server only from 3 zones which correspond to the literals

present in the k th clause. Therefore, the request at decision epoch $4k - 2$ will have at least one server available if clause evaluates to true, i.e., if at least one of the literals has true value.

On assigning a server from the zone corresponding to the first literal, request completes at decision epoch $4k - 1$ earning revenue 1, on assigning a server from the zone corresponding to second literal, request completes at decision epoch $4k$ earning revenue 2 and on assigning a server from the zone corresponding to the third literal, request completes at decision epoch $4k + 1$ earning revenue 3.

- At decision epoch $4k - 1$, a request is present between zone Z_0 and the zone corresponding to first literal of k th clause with revenue 3. At decision epoch $4k$, a request is present between zone Z_0 and the zone corresponding to second literal of k th clause with revenue 2. Similarly, at the decision epoch $4k + 1$, a request is present between zones Z_0 and the zone corresponding to the third literal of k th clause with revenue 1.

The requests at decision epoch $4k - 1$, $4k$ and $4k + 1$ can be assigned a server only from zone Z_0 . The server will be available in zone Z_0 only if the request at decision epoch $4k - 2$ is served. As there is only one request available at the decision epoch $4k - 2$, maximum one of these three requests can be served. The maximum revenue which can be earned by serving the requests between decision epochs $4k - 2$ to $4k + 1$ is 4.

After serving the request at decision epoch $4k - 2$ and one of the requests at decision epochs $4k - 1$, $4k$ or $4k + 1$ such that total revenue is 4, the availability of servers in the zones at decision epoch $4k + 2$ will be same as decision epoch $4k - 2$. Therefore, next clause will be evaluated for the same assignment of variables.

We now show that there is an assignment of values to the variables in the 3-SAT instance so that the formula evaluates to true if and only if there exists a solution to the OSTM problem with objective value $V + 4L$.

The “if” direction: Suppose there exists a solution with the objective value $V + 4L$. As maximum revenue which can be earned by serving the requests between decision epochs $4k - 2$ and $4k + 1$ is 4 and the maximum revenue earned at first decision epoch is V , it means that V requests are assigned a server at first decision epoch earning a total revenue V and one request is served at each of the decision epochs $4k - 2$, $\forall k = 1..L$ and one request is served from every three decision epochs $4k - 1$, $4k$ and $4k + 1$ earning a total revenue of $4L$. The variable x_i is set to true if at first decision epoch zone Z_i server is assigned to the request having destination in zone Z_i otherwise it is set to false.

This will be a solution to 3-SAT instance as all the request at decision epoch $4k - 2$ are served (i.e., all clauses are true) and as the revenue earned between decision epochs $4k - 2$ and $4k + 1$ is 4, at decision epoch $4k + 2$ the servers availability is same as at decision epoch 2.

So if there is a solution to the OSTM problem, we can find an assignment for 3-SAT instance.

The “only if” direction: Suppose there is an assignment of values to the variables such that the 3-SAT formula evaluates to true. So at decision epoch 1, if x_i is true, we assign Z_i server to request having destination in zone Z_i otherwise it is assigned to the request having destination in zone Z'_i . Therefore, revenue earned at decision epoch 1 will be V . Now, as the 3-SAT formula evaluates to true, at decision epoch 2, we will have at least one server available to serve the request. If the first literal of the first clause is true we assign it to request at decision epoch 2 and serve the request at decision epoch 3 earning a revenue of 4. If first literal is false but second literal is true, then we assign it to request at decision epoch 2 and serve the request at decision epoch 4 earning a revenue of 4. If first and second literal are false but the third literal is true then we assign it to request at decision epoch 2 and serve the request at decision epoch 5 earning a revenue of 4. Therefore we can serve request at decision epoch 2 and one of the requests at decision epoch 3, 4 or 5 and earn a total revenue of 4. At decision epoch 6 the servers, the availability of servers will be same as at decision epoch 2. As all the clauses of 3-SAT evaluate to true, the second clause will also be true and at decision epoch 6 we will have at least one server available to serve the request. Therefore, for each clause in 3-SAT, we will serve 2 requests earning a revenue of 4 resulting in objective value of the OSTM to be $V + 4L$ □

Example 4. We show the graphical representation in the Fig. D.20 for an example 3-SAT clause $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$. Initially, there is one server available in zone Z_1 , Z_2 and Z_3 denoted by flow of 1 from the source S . At each decision epoch, the circular nodes represent the server zone nodes and rectangular nodes represent the request nodes with vertex capacity as the number of requests between the zone pairs. The edge between the server node and the request node at a decision epoch represent that server can be assigned to the request. The revenue obtained on assigning a server to the request is marked on the edge. The value of flow on the edge will represent the number of zone servers assigned to request. At first decision epoch, depending on the assignment of server to the requests, the value of x_1 , x_2 , x_3 will be 1 or 0 (true or false). At second decision epoch, the request node has edges from Z_1 , Z_2 and Z_3 . That is the server will be assigned to the request if one of these zones has a server available. Also as the capacity of node is 1, only one of the servers from these zones will be assigned a request. If server of Z_1 is assigned to the request at decision epoch 2, the black edges represent the movement of server. After serving the request at decision epoch 3, the server will become available in the zone Z_1 again. Similarly green edges show the movement of server if server from zone Z_2 is assigned and orange edges show the movement of server, if server from zone Z_3 is assigned. Unassigned servers at decision epoch 2, will remain in the same zones and their movement is represented through dotted lines in the graph. At decision epoch 6, the server distribution in the zones will be same as decision epoch 2 as the servers which moved between decision epoch 2 and 6 came back into the same zone at decision epoch 6.

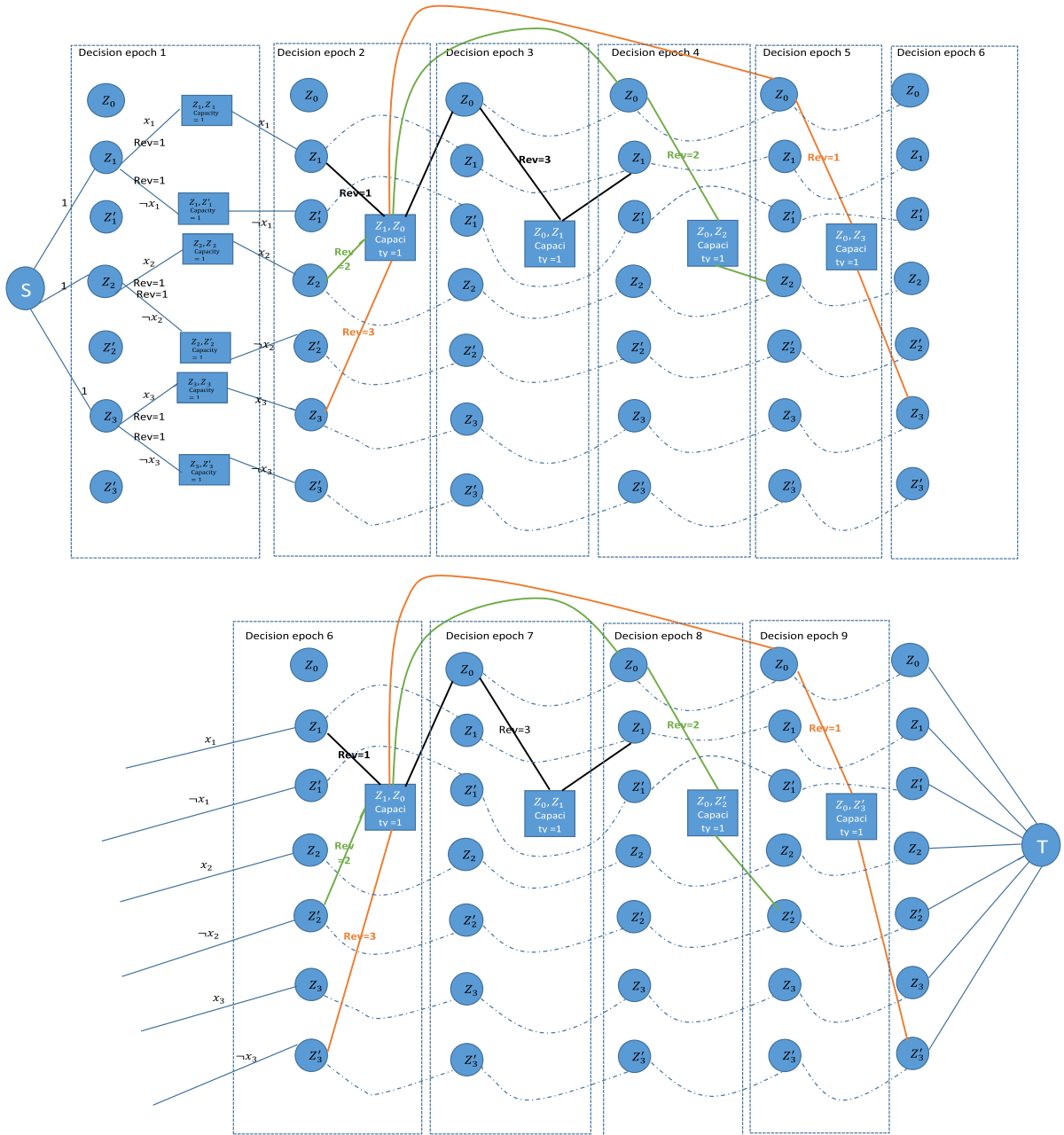


Fig. D.20. Example clause – $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$.

D.3. Proof of Proposition 3

Proposition 3: Solving TSS for more than one sample is an NP-hard problem irrespective of the δ values.

Proof: To show that solving TSS for more than one sample is NP-hard, we reduce the 3-SAT problem to TSS. We construct an instance of TSS for any arbitrary instance of 3-SAT with L clauses and V variables. We show that we obtain the optimal value of $V + 1$ for TSS if and only if there exists an assignment of variables such that the 3-SAT clause evaluates to true.

Each clause of 3-SAT corresponds to one sample in TSS. Each sample has one request, i.e., if the clause evaluates to true, request in the sample is served by a server else it will not be served. The first stage requests are created so that they consider all possible (true/false) values for the variables. Intuitively, the first stage decides the assignment of variables present in clauses and the second stage evaluates the clauses for those variable assignments.

The detailed steps are as follows:

- If there are V variables L clauses we create $2V + L$ zones, denoted by $\{Z_i, Z'_i, i \leq V\}$ and $\{Z_{sk}, k = 1, 2, \dots, L\}$.

- Initially zones Z_i have one server each and other zones have zero servers.
- We create $2V$ requests at the first stage. Each zone Z_i is the origin zone of two requests with one request having destination in Z_i and one request having destination in Z'_i . Requests having origin in zone Z_i can only be assigned servers from zones Z_i . Therefore, $D^1 = \{ \langle Z_i, Z_i, 1 \rangle, \langle Z_i, Z'_i, 1 \rangle \mid i \leq V \}$, $f(Z_i, 1) = \{Z_i\}$. All the requests have unit revenue.
- If the zone Z_i server is assigned to the request having destination in zone Z_i then the variable x_i is true else if it is assigned to the request having destination in the zone Z'_i then x_i is false. Requests served at the first decision epoch determine the value of variables x_i . Therefore, at the second stage, server will be available in the zone Z_i if x_i is true, and will be available in zone Z'_i if x_i is false.
- If there are n clauses, we create n requests (one request in each sample). $|\xi^D| = L$ and $\xi_2^{D,k}$ has one element. Request in k th sample has origin in the zone Z_{sk} and destination in Z_1 . $\xi_2^{D,k} = \{ \langle Z_{sk}, Z_1, 1 \rangle \}$. All requests have unit revenue.
- If the k th clause has literal x_i then zone Z_i server can be assigned to the request in sample k , and if the k th clause has literal $\neg x_i$, then zone Z'_i can be assigned to the request in sample k . On the other hand, if the k th clause does not contain literal x_i then zone Z_i server can not be assigned to the request in sample k . In short, request in k th sample can be assigned server only from 3 zones which correspond to literals present in k th clause.
- Now, the request in sample k will have at least one server available if clause evaluates to true, i.e., if one of the literals has true value.

We now show that there is an assignment of values to the variables in the 3-SAT instance so that the formula evaluates to true if and only if there exists a solution to **TSS** with objective value $V + 1$.

The “if” direction: Suppose there exists a **TSS** solution with the objective value $V + 1$, it means that V requests are assigned a server at the first stage and in all the L samples requests are assigned a server. The variable x_i is set to true if at the first stage zone Z_i server is assigned to the request having destination in zone Z_i otherwise it is set to false. This will be a solution to the 3-SAT instance as requests in all the samples are served so at least one of the literals in each clause is set to true. So if there is a solution to **TSS** instance, we can find an assignment for 3-SAT instance.

The “only if” direction: Suppose there is an assignment of values to the variables such that the 3-SAT formula evaluates to true. So if x_i is true, we assign Z_i server to request having destination in zone Z_i otherwise it is assigned to the request having destination in zone Z'_i . Now, in each sample for each request, we will have at least one server available so we can serve all the L requests at the second stage. Therefore objective value of **TSS** will be $V + \frac{1}{L} \cdot L = V + 1$. \square

D.4. Proof of Proposition 4

Proposition 4 In OLSTM without sample information and adversarial behavior from environment, when maximizing the number of requests satisfied for a fixed number of servers N , the competitive ratio, c for any deterministic b -stage algorithm (i.e., with information available up to the b th decision epoch) in a M -decision epoch ($M \geq b$) problem is

$$c \leq \frac{1}{M - b + 1}$$

Proof. Before we describe the key elements of the proof, we first provide the key terms that will be used in this proof:

- ALG denotes the value of the best deterministic b -stage algorithm over M decision epochs.
- OPT denotes the value obtained by an M -stage optimal algorithm over M decision epochs.
- N is the number of servers available.
- Let OPT_b denote the value of optimal solution for first b decision epochs (i.e., the maximum number of requests which can be served in first b decision epochs).

In order to show the upper bound on competitive ratio, we will consider different cases on values that can be taken by o . Since we are computing competitive ratio (least value of $\frac{ALG(I)}{OPT(I)}$), we identify the least value of the numerator and the highest value of the denominator.

(1) $OPT_b \geq N$, i.e., the number of requests served in the first b decision epochs is greater than N

As ALG denotes the number of requests served by the best deterministic b -stage algorithm over M decision epochs. Therefore, at the very least, it can obtain the optimal solution for b decision epochs and hence:

$$ALG \geq OPT_b$$

Since OPT is the value obtained by an M -stage optimal algorithm, it can potentially serve N requests for every one of the remaining $(M - b)$ time steps. Therefore,

$$OPT \leq (M - b) * N + OPT_b$$

Hence, we have:

$$c \leq \left(\frac{OPT_b}{OPT_b + N * (M - b)} \right) = \left(\frac{1}{1 + \frac{(M-b)}{\frac{OPT_b}{N}}} \right)$$

The above expression will be minimum when $\frac{OPT_b}{N}$ is minimum. As $OPT_b \geq N$, the minimum value of $\frac{OPT_b}{N}$ is 1. Therefore,

$$c \leq \frac{1}{M - b + 1}$$

(2) $OPT_b < N$, i.e., the number of requests served in first b decision epochs is lower than N

As the number of requests served in first b decision epochs is lower than N , there will be some servers which did not move from their initial position. So if optimal algorithm uses these servers to serve requests, deterministic algorithm can also serve requests using them, As the minimum value of ALG and OPT is OPT_b , we take $ALG = OPT_b + x$ and $OPT = OPT_b + y$. Assume the competitive ratio in this case is better than $\frac{1}{1+M-b}$. Therefore,

$$\frac{OPT_b + x}{OPT_b + y} < \frac{1}{1 + M - b} \tag{D.12}$$

$$\implies (OPT_b + x) * (M - b) + OPT_b + x < OPT_b + y \tag{D.13}$$

$$\implies y > OPT_b * (M - b) + x * (M - b + 1) \tag{D.14}$$

As the maximum number of requests served in remaining $M - b$ decision epochs is $N * (M - b)$,

$$y \leq N * (M - b) \tag{D.15}$$

From equation (D.14) and (D.15),

$$\begin{aligned} & OPT_b * (M - b) + x * (M - b + 1) < N * (M - b) \\ \implies & x < (N - OPT_b) * \frac{M - b}{M - b + 1} \\ \implies & x < N - OPT_b \\ \implies & x + OPT_b < N \end{aligned}$$

As the value of ALG in all M decision epochs is less than N , $N - (OPT_b + x)$ servers did not move from their initial position. Therefore, even for the optimal algorithm, $N - (OPT_b + x)$ servers will not be moving. So,

$$\begin{aligned} & y \leq (OPT_b + x) * (M - b) \\ \implies & OPT_b + y \leq (OPT_b + x) * (M - b) + OPT_b \\ \implies & \frac{OPT_b + y}{OPT_b + x} \leq (M - b) + \frac{OPT_b}{OPT_b + x} \\ \implies & \frac{OPT_b + x}{OPT_b + y} \geq \frac{1}{\frac{OPT_b}{OPT_b + x} + (M - b)} \end{aligned}$$

RHS $> \frac{1}{M-b+1}$. But we assumed $\frac{OPT_b+x}{OPT_b+y} < \frac{1}{M-b+1}$ which is a contradiction. Thus,

$$c \leq \frac{1}{M - b + 1} \quad \square$$

We can also extend the above reasoning for the case when objective is to maximize revenue. If revenue of any request $C \in \{C_{min}, C_{max}\}$, then at first b stages, requests with revenue C_{min} are served by both deterministic and optimal algorithm and for the remaining decision epochs requests with revenue C_{max} are served by optimal algorithm. Therefore, the competitive ratio will be $\frac{C_{min}}{C_{min} + (M - b) * C_{max}}$.

The competitive ratio is low mainly due to the assumption that server only moves when it is assigned to a request. Therefore, the adversary can take more advantage by creating requests in the zones which are not reachable from the server position.

It should be possible to improve the competitive ratio on removing this assumption. By taking decision to move randomly to another zone if no request is available at the current stage, we may improve the competitive ratio against an online adversary (i.e., the adversary who is not aware of the output of random decision).

D.5. Proof of Proposition 5

Proposition 5: In OLSTM with sample information and stochastic behavior from environment according to the samples, when maximizing the number of requests satisfied, the expected competitive ratio, c_μ , of the **TSS** algorithm is

$$c_\mu \leq \frac{3}{4 * (M - 1)} + \frac{3}{4 * M}$$

where M is the number of decision epochs ($M \geq 3$).

Proof: To prove the upper bound on the expected competitive ratio, we construct a worst case distribution and show the value of expected competitive ratio for that distribution. The value of expected competitive ratio for any specific distribution is upper bound on the true value.

We assume that there is a positive probability of having a request at second decision epoch.¹⁷ As the **TSS** only knows distribution for the next decision epoch, we create an instance such that **TSS** can not serve any requests after first two decision epochs.

We construct a worst case instance with one server and two requests at first two decision epochs. Initially, the server is located in zone Z_1 , the two requests are request 1, $r_1 = \langle Z_1, Z_1 \rangle$ and request 2, $r_2 = \langle Z_1, Z_2 \rangle$. Requests originating in zone Z_1 can only be served by server in zone Z_1 . Similarly request originating in zone Z_2 can only be served by server in Z_2 .

As per the distribution at decision epoch 2, the probability that request has origin and destination in Z_1 is P_1 and probability that request has origin and destination in Z_2 is P_2 and the probability is zero for all other zone pairs. P_1 and P_2 are independent of each other. Also as per the distribution, at subsequent decision epochs, there is zero probability of a request having origin in zone Z_1 and 1 probability of requests having origin and destination at Z_2 .

The **TSS** algorithm will make the first stage assignment based on the expected number of requests served, i.e., it will maximize $1 + \max(P_1, P_2)$. If $P_1 \geq P_2$, **TSS** will assign the server to request 1 at decision epoch 1 and will serve request at decision epoch 2 with probability P_1 . **TSS** will not be able to serve any more requests at subsequent decision epoch.

The M-stage optimal algorithm will assign server to request 2 at first decision epoch. At second decision epoch, the M-stage optimal algorithm will be able to serve the request if it originates in zone Z_2 otherwise it will not. The M-stage algorithm will be able to serve requests at all subsequent decision epochs.¹⁸ Therefore, the expected competitive ratio is given by

$$c_\mu \leq P_1 * (1 - P_2) * \frac{2}{M - 1} + P_1 * P_2 * \frac{2}{M} + (1 - P_1) * P_2 * \frac{1}{M} + (1 - P_1) * (1 - P_2) * \frac{1}{M - 1} \quad (D.16)$$

The four terms correspond to four possible cases of drawing 2 requests from the given distribution. Rearranging the terms, we get

$$c_\mu \leq (1 + P_1) * \frac{1}{M - 1} - P_2 * (1 + P_1) * \left(\frac{1}{M - 1} - \frac{1}{M} \right) \quad (D.17)$$

From equation (D.17), as $\frac{1}{M-1} > \frac{1}{M}$, for a fix value of P_1 , on increasing the value of P_2 the expected competitive ratio decreases. As $P_1 \geq P_2$, to minimize the expected value of competitive ratio, we take $P_2 = P_1$. Substituting $P_2 = P_1$ in equation (D.17), we get

$$c_\mu \leq (1 + P_1) * \frac{1}{M - 1} - P_1 * (1 + P_1) * \left(\frac{1}{M - 1} - \left(\frac{1}{M} \right) \right)$$

The above expression will be minimum when the derivative (with respect to P_1) is 0, i.e.,

$$\begin{aligned} (1) * \frac{1}{M - 1} - (1 + 2 * P_1) * \left(\frac{1}{M - 1} - \frac{1}{M} \right) &= 0 \\ \implies 2 * P_1 * \left(\frac{1}{M - 1} - \frac{1}{M} \right) &= \frac{1}{M} \\ \implies P_1 &= \frac{M - 1}{2} \end{aligned}$$

As $0 \leq P_1 \leq 1$, and M is a positive integer, the possible value of P_1 are 0, 0.5 and 1.0. Since, we have a positive probability of having request at second decision epoch, $P_1 = 0$ is not possible. Therefore, the competitive ratio will be minimum when

¹⁷ If the probability of having a request at second decision epoch is 0, **TSS** will not have any future information available and will be as good as deterministic one-stage algorithm. In case of zero probability, from Proposition 4, the competitive ratio will be $\frac{1}{M}$.

¹⁸ The number of decision epochs (M) is greater than or equal to 3.

$P_1 = 0.5$ or 1 . On substituting $P_1 = P_2 = 1.0$, in equation (D.17), we get $\frac{2}{M}$ and on substituting $P_1 = P_2 = 0.5$, in equation (D.17), we get $\frac{3}{4*(M-1)} + \frac{3}{4*M}$

As $\frac{2}{M} > \frac{3}{4*(M-1)} + \frac{3}{4*M}$, for $M \geq 3$, therefore

$$c\mu \leq \frac{3}{4*(M-1)} + \frac{3}{4*M} \quad (\text{D.18})$$

As there exists an instance for which the expected competitive ratio of **TSS** can not be more than $\frac{3}{4*(M-1)} + \frac{3}{4*M}$, we can say that $\frac{3}{4*(M-1)} + \frac{3}{4*M}$ is the upper bound on the expected competitive ratio of **TSS**.

References

- [1] M. Lowalekar, P. Varakantham, P. Jaillet, Online spatio-temporal matching in stochastic and dynamic domains, in: AAAI, 2016, pp. 3271–3277.
- [2] N. Agatz, A.L. Erera, M.W. Savelsbergh, X. Wang, Dynamic ride-sharing: a simulation study in metro Atlanta, Proc., Soc. Behav. Sci. 17 (2011) 532–550.
- [3] S. Ghosh, P. Varakantham, Y. Adulyasak, P. Jaillet, Dynamic repositioning to reduce lost demand in bike sharing systems, J. Artif. Intell. Res. 58 (2017) 387–430.
- [4] H. Hosni, J. Naoum-Sawaya, H. Artail, The shared-taxi problem: formulation and solution methods, Transp. Res., Part B, Methodol. 70 (2014) 303–318.
- [5] S. Saisubramanian, P. Varakantham, H.C. Lau, Risk based optimization for improving emergency medical systems, in: AAAI, 2015, pp. 702–708.
- [6] J. Yang, P. Jaillet, H.S. Mahmassani, Real-time multivehicle truckload pickup and delivery problems, Transp. Sci. 38 (2) (2004) 135–148.
- [7] U. Ritzinger, J. Puchinger, R.F. Hartl, A survey on dynamic and stochastic vehicle routing problems, Int. J. Prod. Res. 54 (1) (2016) 215–231.
- [8] R.M. Karp, U.V. Vazirani, V.V. Vazirani, An optimal algorithm for on-line bipartite matching, in: H. Ortiz (Ed.), STOC, ACM, 1990, pp. 352–358.
- [9] G. Aggarwal, G. Goel, C. Karande, A. Mehta, Online vertex-weighted bipartite matching and single-bid budgeted allocations, in: Proceedings of the Twenty-Second Annual ACM–SIAM Symposium on Discrete Algorithms, SIAM, 2011, pp. 1253–1264.
- [10] W.B. Powell, A stochastic formulation of the dynamic assignment problem, with an application to truckload motor carriers, Transp. Sci. 30 (3) (1996) 195–219.
- [11] H. Topaloglu, W.B. Powell, Dynamic-programming approximations for stochastic time-staged integer multicommodity-flow problems, INFORMS J. Comput. 18 (1) (2006) 31–42.
- [12] H.P. Simao, J. Day, A.P. George, T. Gifford, J. Nienow, W.B. Powell, An approximate dynamic programming algorithm for large-scale fleet management: a case application, Transp. Sci. 43 (2) (2009) 178–197.
- [13] W.B. Powell, Approximate Dynamic Programming: Solving the Curses of Dimensionality, vol. 703, John Wiley & Sons, 2007.
- [14] I. Katriel, C. Kenyon-Mathieu, E. Upfal, Commitment under uncertainty: two-stage stochastic matching problems, Theor. Comput. Sci. 408 (2) (2008) 213–223.
- [15] G. Zhang, K. Smilowitz, A. Erera, Dynamic planning for urban drayage operations, Transp. Res., Part E, Logist. Transp. Rev. 47 (5) (2011) 764–777.
- [16] A. Borodin, R. El-Yaniv, Online Computation and Competitive Analysis, Cambridge University Press, New York, NY, USA, 1998.
- [17] P. Jaillet, X. Lu, Online stochastic matching: new algorithms with better bounds, Math. Oper. Res. 39 (3) (2013) 624–646.
- [18] A. Mehta, et al., Online matching and ad allocation, Found. Trends Theor. Comput. Sci. 8 (4) (2013) 265–368.
- [19] V.H. Manshadi, S.O. Gharan, A. Saberi, Online stochastic matching: online actions based on offline statistics, Math. Oper. Res. 37 (4) (2012) 559–573.
- [20] A. Blum, T. Sandholm, M. Zinkevich, Online algorithms for market clearing, J. ACM 53 (5) (2006) 845–879.
- [21] Y. Wang, S.C. Wong, Two-sided online bipartite matching and vertex cover: beating the greedy algorithm, in: Automata, Languages, and Programming – 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6–10, 2015, Proceedings, Part I, 2015, pp. 1070–1081.
- [22] J.P. Dickerson, K.A. Sankararaman, A. Srinivasan, P. Xu, Allocation problems in ride-sharing platforms: online matching with offline reusable resources, in: AAAI, 2018.
- [23] E. Feuerstein, L. Stougie, On-line single-server dial-a-ride problems, Theor. Comput. Sci. 268 (1) (2001) 91–105.
- [24] M. Lipmann, X. Lu, W.E. de Paepe, R.A. Sitters, L. Stougie, On-line dial-a-ride problems under a restricted information model, in: European Symposium on Algorithms, Springer, 2002, pp. 674–685.
- [25] V. Bonifaci, M. Lipmann, L. Stougie, Online multi-server dial-a-ride problems, TU/e, Eindhoven University of Technology, Department of Mathematics and Computing Science, 2006.
- [26] S. Ropke, J.-F. Cordeau, G. Laporte, Models and branch-and-cut algorithms for pickup and delivery problems with time windows, Networks 49 (4) (2007) 258–272.
- [27] S. Ma, Y. Zheng, O. Wolfson, T-share: a large-scale dynamic taxi ridesharing service, in: 2013 IEEE 29th International Conference on Data Engineering (ICDE), IEEE, 2013, pp. 410–421.
- [28] S.N. Parragh, V. Schmid, Hybrid column generation and large neighborhood search for the dial-a-ride problem, Comput. Oper. Res. 40 (1) (2013) 490–497.
- [29] U. Ritzinger, J. Puchinger, R.F. Hartl, Dynamic programming based metaheuristics for the dial-a-ride problem, Ann. Oper. Res. 236 (2) (2016) 341–358.
- [30] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, D. Rus, On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment, Proc. Natl. Acad. Sci. 114 (3) (2017) 462–467.
- [31] M.Z. Spivey, W.B. Powell, The dynamic assignment problem, Transp. Sci. 38 (4) (2004) 399–419.
- [32] G.A. Godfrey, W.B. Powell, An adaptive dynamic programming algorithm for dynamic fleet management, II: multiperiod travel times, Transp. Sci. 36 (1) (2002) 40–54.
- [33] C. Szepesvári, Algorithms for reinforcement learning, Synth. Lect. Artif. Intell. Mach. Learn. 4 (1) (2010) 1–103.
- [34] E. Even-Dar, S.M. Kakade, Y. Mansour, Online Markov decision processes, Math. Oper. Res. 34 (3) (2009) 726–736.
- [35] H.B. McMahan, G.J. Gordon, A. Blum, Planning in the presence of cost functions controlled by an adversary, in: ICML, 2003, pp. 536–543.
- [36] Y. Abbasi, P.L. Bartlett, V. Kanade, Y. Seldin, C. Szepesvári, Online learning in Markov decision processes with adversarially chosen transition probability distributions, in: Advances in Neural Information Processing Systems, 2013, pp. 2508–2516.
- [37] L. Li, T.J. Walsh, M.L. Littman, Towards a unified theory of state abstraction for MDPs, in: ISAAC, 2006.
- [38] A. Shapiro, D. Dentcheva, A. Ruszczyński, Lectures on Stochastic Programming: Modeling and Theory, SIAM, 2009.
- [39] A. Shapiro, On complexity of multistage stochastic programs, Oper. Res. Lett. 34 (1) (2006) 1–8.
- [40] L. Mercier, P. Van Hentenryck, Performance analysis of online anticipatory algorithms for large multistage stochastic integer programs, in: IJCAI, 2007, pp. 1979–1984.
- [41] R. Bent, P. Van Hentenryck, Regrets only! Online stochastic optimization under time constraints, in: AAAI, vol. 4, 2004, pp. 501–506.
- [42] L. Mercier, P. Van Hentenryck, An anytime multistep anticipatory algorithm for online stochastic combinatorial optimization, Ann. Oper. Res. 184 (1) (2011) 233–271.

- [43] G. Ghiani, E. Manni, B.W. Thomas, A comparison of anticipatory algorithms for the dynamic and stochastic traveling salesman problem, *Transp. Sci.* 46 (3) (2012) 374–387.
- [44] R.W. Bent, P. Van Hentenryck, Scenario-based planning for partially dynamic vehicle routing with stochastic customers, *Oper. Res.* 52 (6) (2004) 977–987.
- [45] S. Rebennack, Combining sampling-based and scenario-based nested benders decomposition methods: application to stochastic dual dynamic programming, *Math. Program.* 156 (1–2) (2016) 343–389.
- [46] A. Legrain, P. Jaillet, A stochastic algorithm for online bipartite resource allocation problems, *Comput. Oper. Res.* 75 (2016) 28–37.
- [47] J. Murphy, Benders, nested benders and stochastic programming: an intuitive introduction, *CoRR*, arXiv:1312.3158.
- [48] J.F. Benders, Partitioning procedures for solving mixed-variables programming problems, *Numer. Math.* 4 (1) (1962) 238–252.
- [49] D. Bertsimas, J.N. Tsitsiklis, *Introduction to Linear Optimization*, vol. 6, Athena Scientific, Belmont, MA, 1997.
- [50] New York yellow taxi dataset, http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml.
- [51] Singapore taxi fare, <https://www.lta.gov.sg/content/ltaweb/en/public-transport/taxis/fares-and-payment-methods.html>.