CS 269I Report: Time-Sensitive Market Matching

Derek Phillips Emil Martayan Georgia Murray Theo Diamandis

December 11, 2018

1 Introduction

We examine the matching problem in the ride sharing market, where buyers (e.g., Lyft users) want to be matched with sellers (e.g, drivers or autonomous vehicles), and the goods being sold are rides from place to place. As in the various settings considered in class, we wish to find a stable matching. However, this market presents a challenge. Participants arrive at different times and then depart after some time period, and participants can only be matched while they are in the marketplace. The platform's matching algorithm, therefore, must perform matches online without full knowledge of future states of the marketplace.

We assume this platform has access to the pairwise utilities of each buyer-seller pair. The valuations and prices to compute these utilities might be shared with the platform, or the platform might compute them as expected profits from each ride. This computed pairwise utility might take into account the probability of the buyer accepting the ride and the travel time from seller to buyer pickup, among other factors. The platform is incentivized to maximize utility in the latter case, as a utility maximizing matching will result in the largest revenue.

Our investigations of this market largely extend those of Ashlagi *et al.* [Ash18] through simulation. Section 2 details the setting described in [Ash18] along with implementation specifications relevant to our simulation. Having created this platform, we proceed to develop a number of different algorithms which could be considered to perform the matching in a real-world environment. We discuss the details of each of these algorithms along with their expected advantages and disadvantages in Section 3. Section 4 compares the realized results of the various algorithms and concludes the simulation portion of our project. Section 5 introduces an additional and largely independent portion of our project where we consider a procession of simplified versions of the ride sharing market in which we can make claims regarding optimality and strategyproofness. Finally, we address the limitations of our model, the validity or lack thereof of our assumptions, and promising future directions in Section 6. All code can be found at `https://github.com/tjdiamandis/cs269i_final_project`.

2 Model and Simulation

2.1 Mathematical Model

Our work focused on simulations of the market model developed by Ashlagi *et al.* [Ash18], so we defined a *similar* market model, slightly modified to be more realistic. Consider a weighted bipartite graph G on n nodes (indexed by $i = 1, 2, \ldots, n$), where the nodes arrive over T periods

such that node $i+1$ arrives after or at the same time as node i. Each node is either a buyer or seller, but not both. Zero, one, or multiple nodes can arrive during a single period. We denote the arrival time of node i by σ_i and the weight on edge (i, j) as $w_{i,j}$ (where the edge, and therefore the weight, are only observed once the later of the two nodes arrives). This weight indicates the value of matching node i with node j. Additionally, node i leaves the market after d_i periods. Thus, edges only exist between nodes i and j that are both in the market; $w_{i,j}$ exists iff current time step $t \in \{\sigma(i), \sigma(i)+1, \ldots, \sigma(i)+d_i-1\} \cap \{\sigma(j), \sigma(j)+1, \ldots, \sigma(j)+d_j-1\}$. When simulating this model, we must consider several parameters governing the market's evolution Figure 1 shows a potential market at a period where node 1 has already departed and nodes 6 and 7 have not yet arrived.

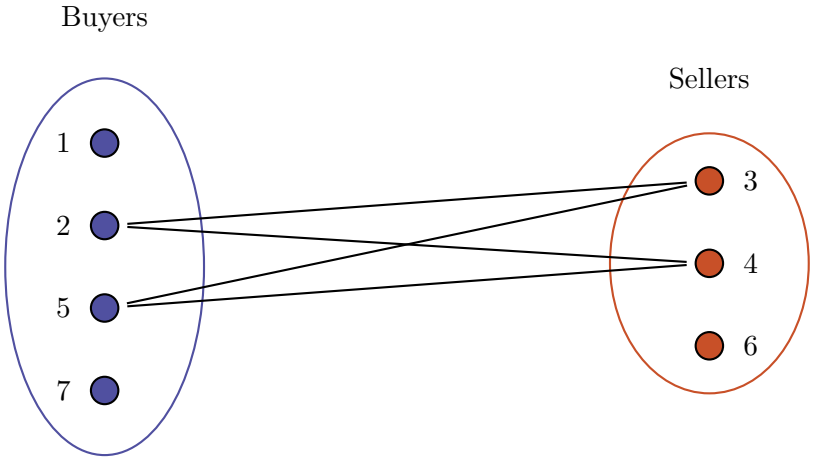

Figure 1: Example of our market as a graph at some time step where node 1 has already departed and nodes 6 and 7 have not yet arrived. Each of the edges from node i to node j has some associated weight $w_{i,j}$.

2.2 Simulation

We aimed to simulate a simplified version of the ride sharing market. Due to the difficulty of obtaining real data sets for this market, we built a simulator to generate data and tested our algorithms on these data. Note that this market is tackled from the perspective of the platform (e.g., Lyft), which is making matches based on the data it knows about the participants. We simulate by stepping sequentially through time. At each time step, with probability 90%, n_t new nodes will arrive into the market. Each node will be assigned to be a buyer with probability p or to be a seller with probability $1-p$. Each node i will be placed at a fixed two dimensional location (x_i, y_i). The node i will stay in the market for d_i periods (e.g., if $d=1$, it will leave at the next time step). This discretization of time can be thought of as intervals in which participants enter or leave the market. The matching algorithm runs at the end of each interval. In summary, at each time step:

- Some number of nodes leave the market after having stayed for d_i periods
- Some number of nodes n_t enter the market. These nodes will have
  - Status of buyer or seller
  - Location (x_i, y_i)
  - Number of time steps to stay in the market d_i

- A matching algorithm runs on the current state. Matched nodes are removed.

We used a bipartite graph structure to represent the state of the market at a given time step t, as depicted in Figure 1, implemented with the NetworkX Python package [HSSC08]. Each node stores its own attributes (e.g., departure time), and the edges between nodes represent the value of matching a particular buyer-seller pair. Because we are looking at this market from the perspective of a platform, we compute this value based on the information available to us: the locations of the nodes and the departure times.

The simulation we use clearly has quite a few tunable parameters. In summary, the following parameters can be changed as desired for simulation:

A. Per node attributes: Departure time d_i, location (x_i, y_i)
B. Fraction of buyers in the market p
C. Weight function that computes value $w_{i,j}$ of matching buyer i with seller j
D. Frequency of buyer/seller arrival

Exact choices for these parameters (or the distributions from which they are drawn) are discussed in more detail in Section 4.

3 Algorithms

3.1 Greedy and Dynamic Deferred Acceptance

Our first two algorithms are the greedy algorithm and the Dynamic Deferred Acceptance algorithm (DDA) from Ashlagi *et al.* [Ash18]. The greedy algorithm serves as a baseline, and is described in Algorithm 1 of Ashlagi [Ash18]. It proceeds by working on a constrained bipartite graph and keeps track of the best matches for each seller, which are updated with the arrival of each buyer. When a seller becomes critical (that is, is about to leave the market), it returns the currently tracked best match. If the seller has no tracked best match when going critical, the node is simply dropped.

The Dynamic Deferred Acceptance algorithm, described in Algorithm 4 of Ashlagi [Ash18], is similar to the greedy algorithm. However, DDA uses an ascending auction procedure, and tracks the previous best matches. Tracking the previous best matches will result in fewer dropped nodes.

3.2 k-lookahead

In both algorithms described above, we assume that the platform has no knowledge of incoming nodes until arrival. However, in many applications, some foreknowledge is reasonable, whether definitive as in scheduling rides in advance or probabilistic as in expected time of increased ride-traffic. To incorporate this foreknowledge without unduly complicating our model, we consider introducing a "wait time" k in addition to our departure deadline d. Each incoming node, be it buyer or seller, becomes visible to the platform k time steps before actually arriving and can at that point be including in the matching at each time step.

This raises a potential concern of foregoing actual matches in favor of potential matches which may in reality be impossible. Consider the following three node example in which $d = 2$ and $k = 2$. Three nodes will enter the market: s_1 at $t = 1$, b_1 at $t = 2$, and s_3 at $t = 4$. Assume that the relative matchings are $(s_1 - b_1) < (s_2 - b_1)$. This example is summarized in Table 1.

Time	In market	Visible to market	Tentative matchings
$t = 1$	s_1	s_1, b_1	$(s_1 - b_1)$
$t = 2$	s_1, b_1	s_1, b_1, s_2	$(s_2 - b_1)$
$t = 3$	b_1	b_1, s_2	$(s_2 - b_1)$
$t = 4$	s_2	s_2	None

Table 1: Three node explore for k-lookahead.

In this case, even though the matching $(s_1 - b_1)$ was possible and was preferable over no match, it did not occur because b_1 was tentatively matched with s_2 but left the market before the match was finalized. In order to correct for this, we invoke the idea of a "valid match" and only allow tentative matches between nodes whose time in the market will overlap.

Finally, let us consider how we expect k-lookahead to affect our performance. For out study, we added k-lookahead to the Dynamic Deferred Acceptance algorithm, but not the the greedy algorithm, so we will limit our discussion here to DDA with k-lookahead. The only change to our algorithm is that we have given our platform additional information by allowing it to partially interpret future states. (Note that the above caveat of disallowing unrealizable tentative matches ensures that it only considers future states which will actually occur.) Thus we would expect k-lookahead to only ever improve performance and to have the most effect when d is small and the rate of node arrival is high, as this corresponds to high turnover where foreknowledge will be most advantageous.

3.3 Stochastic Departures

All the algorithms described above assume that the platform has perfect knowledge of the departure deadline for each node, which is a strong and generally unrealistic assumption. In [Ash18], Ashlagi *et al.* propose stochastic departures to relax this assumption. In this setting, the departure time d_i of node i is sampled independently from a distribution, and d_i is only known once i becomes critical. We extend this further by considering the case where the algorithm knows the distribution of departure deadlines *a priori* but never knows the specific departure deadline for a given node. Since we assume that all riders are identically distributed, our solution to this problem involves making a single hypothesis as to the departure time of each node, i.e. the algorithm will determine some value $\hat{d}$ (based on the specified departure distribution) and will assume all nodes have deadline $\hat{d}$. We then implement the following procedure: once a seller s becomes critical according to $\hat{d}$ (i.e. it has been in the market for time $\hat{d} - 1$), we attempt to match it. If the seller is actually present in the market (i.e. if $\hat{d} \leq d$), then the match is executed successfully; otherwise, the match is not executed because the seller has already left. Notice that this is the point at which the true deadline d comes into play: the algorithm will commit to finalizing a match, and only then is it revealed whether or not the seller is still in the market (which is uniquely determined by d). Determining the guess $\hat{d}$ is a problem in and of itself. Because of the complex interplay among our various parameters, we consider setting $\hat{d}$ to various quantiles of the specified departure distribution. We then run our simulations to determine which value yields the best results across our various metrics.

4 Results

We ran all algorithms discussed in the previous section on several separate simulation scenarios. All scenarios used the following parameters:

- At each time step t, with probability 90% nodes arrive to the market. If nodes arrive, the number of nodes that arrive is uniformly chosen from 1 to 6, $n_t \sim \mathcal{U}(\{1, 2, \ldots, 6\})$.
- The position coordinates for each entering node $(x, y) \sim \mathcal{U}(\{[0, 10) \times [0, 10)\})$
- The market is simulated for 500 time steps

The parameters we changed for the scenarios are the proportion of buyers/sellers, the weight function used to calculate pairwise edge weights, and the statistics of departure time d. We wanted to explore the effects of changing these parameters. We started with a balanced market, a fixed departure time d of six time steps, and a weight function inversely proportional to distance. This departure time can be thought of as a deadline for the platform to match people who have been waiting d time periods (e.g., one minute if each period is 10 seconds). The inverse distance metric was chosen since matching people far away from each other should be avoided in the ride sharing market, as the seller must travel to the buyer. We varied each of these parameters while holding the others constant. We explored algorithm performance in markets with significantly more sellers ($p = 0.3$) or significantly more buyers ($p = 0.7$). We also considered a multi-objective weight function that takes into account s_i = the number of time periods until node i leaves the market. This weight function assigns a higher weight to matchings where the participants are going to leave the market most immediately. Finally, we explored the performance of the Dynamic Deferred Acceptance algorithm in markets where participants leave at random times so that d_i is a random variable, which is more realistic. These differences are outlined in Table 2.

Simulation Number	Fraction of Buyers p	Departure time d_i	Weight Function w
1	$p = 0.5$	$d_i = 6$	$w_{i,j} = \frac{1}{1+\mathbf{dist}(i,j)}$
2	$p = 0.3$	$d_i = 6$	$w_{i,j} = \frac{1}{1+\mathbf{dist}(i,j)}$
3	$p = 0.7$	$d_i = 6$	$w_{i,j} = \frac{1}{1+\mathbf{dist}(i,j)}$
4	$p = 0.5$	$d_i = 6$	$w_{i,j} = \frac{1}{r_i+r_j} + \frac{1}{1+\mathbf{dist}(i,j)}$
5	$p = 0.5$	$d_i = 6$	$w_{i,j} = \frac{1}{r_i+r_j} + \frac{10}{1+\mathbf{dist}(i,j)}$
6	$p = 0.5$	$d_i = 6$	$w_{i,j} = \frac{10}{r_i+r_j} + \frac{1}{1+\mathbf{dist}(i,j)}$
7	$p = 0.5$	$d_i \sim \mathcal{N}(6, 1)$	$w_{i,j} = \frac{1}{1+\mathbf{dist}(i,j)}$
8	$p = 0.3$	$d_i \sim \mathcal{N}(6, 1)$	$w_{i,j} = \frac{1}{1+\mathbf{dist}(i,j)}$
9	$p = 0.7$	$d_i \sim \mathcal{N}(6, 1)$	$w_{i,j} = \frac{1}{1+\mathbf{dist}(i,j)}$

Table 2: Different parameterizations of our simulation procedure.

Figure 2 depicts performance in a balanced market with a fixed departure time and weight function inversely proportional to distance. We see that among *matches*, the greedy algorithm has the highest average weight. However, the greedy algorithm has the lowest average weight per node (or sum of weights), so this high average is at the expense of the number of matches. Indeed, we see that the greedy algorithm discards significantly more buyers and sellers (i.e., leaves them unmatched). Thus, it has slightly higher weight standard deviation as well. The Dynamic Deferred Acceptance algorithm has lower average match weight but higher sum of all weights as it matches a larger portion of the buyers, as expected. Finally, we see that the k-lookahead algorithm offers some improvement; however, note that the fluctuations might be due to random chance, as we were not able to test each algorithm on the *exact* same simulation due to design decisions made at the onset of the project. We simply ran the simulation with the same parameters.

Figures 3-4 depict a market with more sellers and more buyers, respectively. We note that the greedy algorithm consistently performs worse overall, despite having higher per-matching weight.

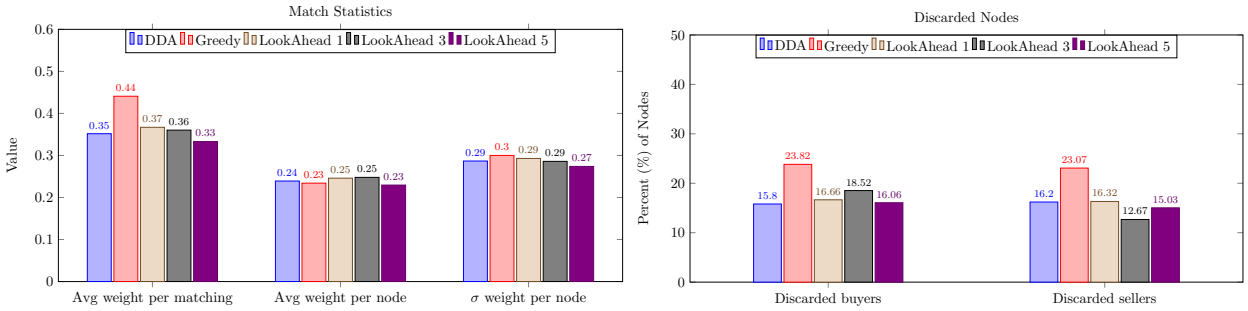

Figure 2: Simulation 1, $p = 0.5$

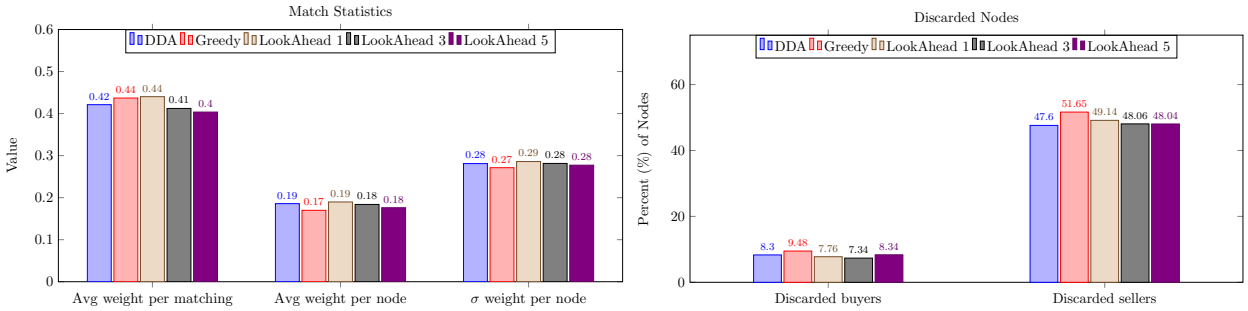

Figure 3: Simulation 2, $p = 0.3$

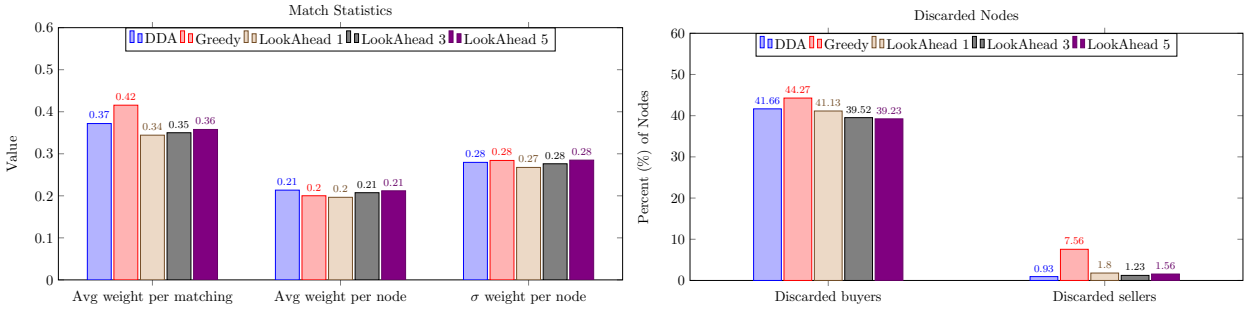

Figure 4: Simulation 3, $p = 0.7$

Additionally, the algorithm will discard more nodes, leading to unhappy ride sharing platform users.

Figures 5-7 show simulations with three different weight functions, each with an additional term inversely proportional to the time remaining in the market. Note that the weight of this term greatly affects the percent of discarded nodes, as one might intuitively expect. When we put more weight on the deadline-driven term, the percent of discarded nodes goes down significantly. Since the functions are different, weights cannot be compared one-to-one. We again see a theme of the greedy algorithm performing worse overall, except in the case where we put a lot of emphasis on the deadline-dependent term.

Finally, we explored the Dynamic Deferred Acceptance algorithm for stochastic departures. We drew $d_i \sim \mathcal{N}(6, 1)$. We decided to use a departure time guess in the algorithm, which we swept from 1 to 7. For a lower guess, we are matching soon after nodes come into the market if we can, so we match nearly all nodes but miss a lot of potential better matches. For a larger guess, we wait longer to match, which will increase our average weight up to a point, as we will have more time to

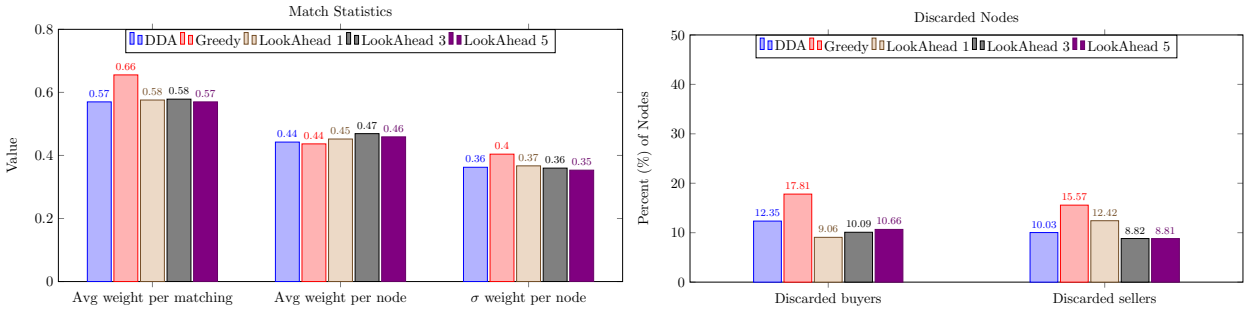

Figure 5: Simulation 4, $w_{i,j} = \frac{1}{r_i + r_j} + \frac{1}{1 + \mathbf{dist}(i,j)}$

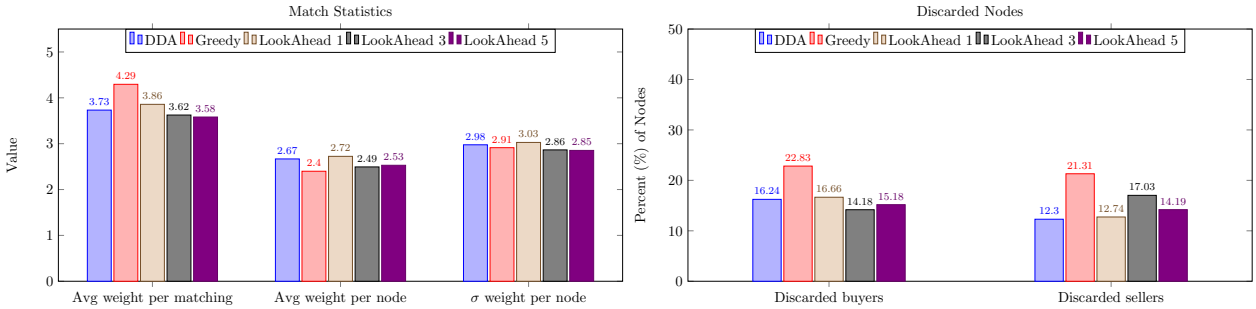

Figure 6: Simulation 5, $w_{i,j} = \frac{1}{r_i + r_j} + \frac{10}{1 + \mathbf{dist}(i,j)}$

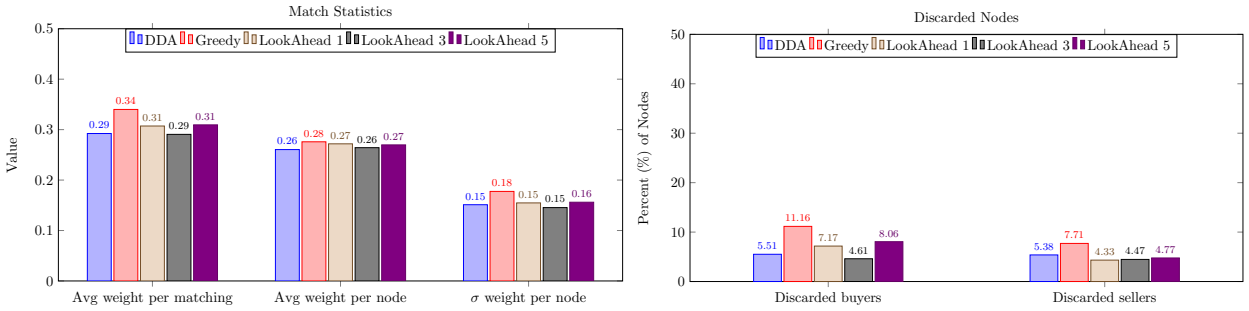

Figure 7: Simulation 6, $w_{i,j} = \frac{10}{r_i + r_j} + \frac{1}{1 + \mathbf{dist}(i,j)}$

match nodes. However, some of these nodes may leave without a match, since their true departure time was lower than our guess.

We see that the average weight per matching is largely increasing or constant to a guess of about 5 (one standard deviation below the mean of d_i). However, the optimal sum of weights (social welfare) is achieved around a guess of 3 for all cases. This result suggests that the conservative approach of early matching is better when there is uncertainty in departure time. Conveniently, this approach not only leads to high social welfare but also results in a small percentage of unmatched nodes, so everyone is happy. Note that in the balanced market case depicted in Figure 8, we see that the number of discarded nodes roughly corresponds to the CDF of a Gaussian random variable, as expected.

In conclusion, our results suggest that the greedy algorithm typically underperforms compared to the Dynamic Deferred Acceptance algorithm. Additionally, the k-lookahead algorithm may offer some benefit, but this benefit is small and more extensive testing would be required to sufficiently support this claim. Finally, we show that under stochastic departures, it is better to be conservative

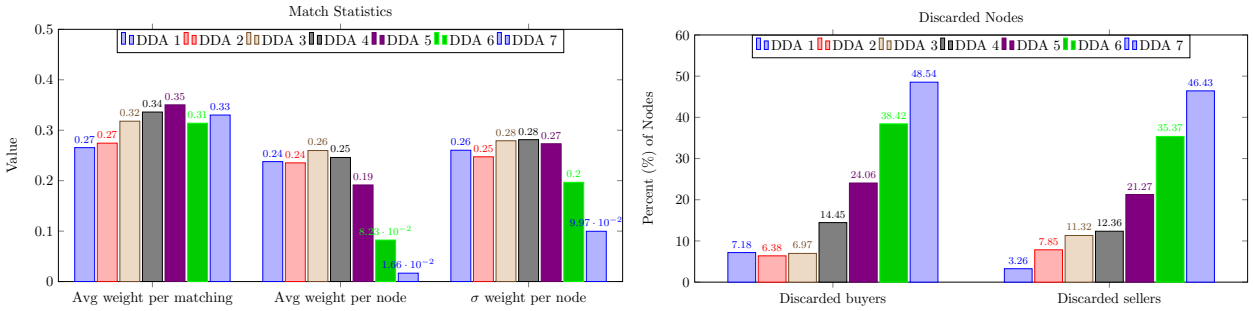

Figure 8: Simulation 7, $p = 0.5$, $d \sim \mathcal{N}(6, 1)$

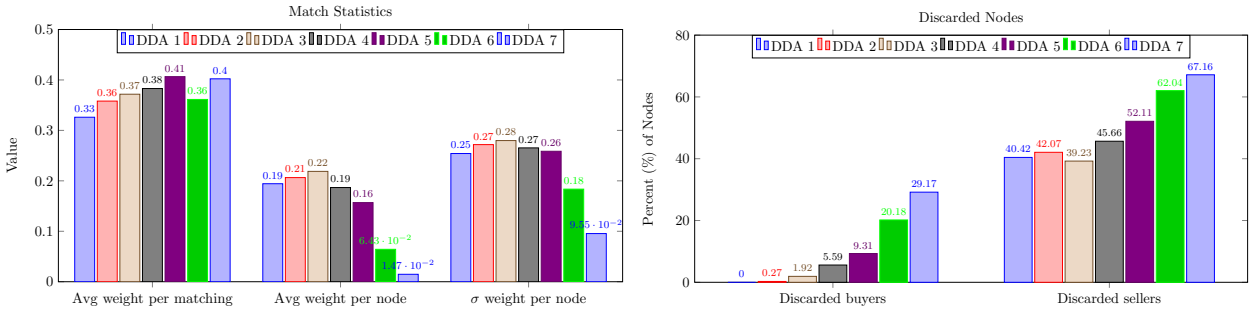

Figure 9: Simulation 8, $p = 0.3$, $d \sim \mathcal{N}(6, 1)$

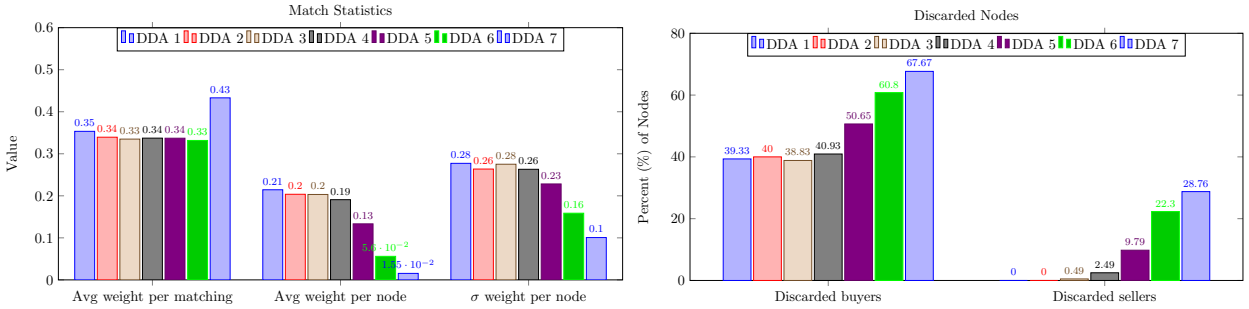

Figure 10: Simulation 9, $p = 0.7$, $d \sim \mathcal{N}(6, 1)$

and match nodes well before you might expect them to leave.

5 Incentive-Related Issues

In order to consider the incentives issues in our system, we will progress by a series of simplified models in order of increasing complexity and will address the Pareto optimality and strategypoofness of each.

While some assumptions will be successively relaxed as the model complexity increases, some assumptions will hold for all models. For clarity and ease of referrence, we detail those assumptions here.

- All discussion here refers to the original model (Section 3.1) using DDA without the inclusion of k-lookahead (Section 3.2) or stochastic departure (Section 3.3). Note that this produces a maximum-weight matching at each time step; this is shown in [Ash18], although we will not

reproduce the proof here. Recall that matches become final when and only when the seller in the match goes critical.

- Deadlines are self-reported and binding. That is, when a buyer or seller arrives at the market, they specify their deadline d after which they will exit the market, and they are indeed ejected if still unmatched when that deadline is reached.
- Re-entry is disallowed. That is, once a buyer or seller exits the market, whether due to matching or ejection upon reaching their deadline, they cannot re-enter the market.
- The position of each node is determined by a third-party (*e.g.* GPS data) and is assumed to be correct. Notably, when considering strategyproofness, we will assume that agents can lie about their deadlines but not about their positions.
- When considering Pareto optimality, we will assume that the self-reported deadline d is accurate, although we will not make this assumption when considering strategyproofness. We will also assume that any match is preferable to no match.
- As we will consider Sybil attacks specifically in Section 5.4, we will assume for all other sections that each user has a single identity. Further, excepting when we consider the Sybil attacks, we will assume that nodes cannot reject the matches they are assigned.

An additional preliminary discussion of strategypoofness will make the conclusions of following case-studies clearer. Note that, in our Dynamic Deferred Acceptance matching strategy, two mechanisms are at play: the Deferred Acceptance algorithm run at each time step, and the dynamics introduced by only finalizing each matching when the seller has gone critical. We will consider each of these mechanisms separately when discussing strategypoofness; that is, we will have two concepts of strategyproofness: DA-strategyproof, dynamics-strategyproof.

5.1 Stationary Model

Consider the following, highly simplified setting: all nodes, both buyers and sellers arrive at the market simultaneously at time t_0 and with identical deadlines d. In this case, the entire market will clear at a single time step $t = t_0 + d$. Since all nodes have the same deadline, no nodes will have exited prior to this clearing and thus all nodes will be considered in the final matching computed before clearing the market.

Claim 1. *The case of stationary buyers and sellers yields a Pareto optimal outcome and is strategyproof for buyers, but not for sellers.*

Proof. This setting is in fact identical to a single run of Deferred Acceptance over all nodes. Since Deferred Acceptance is known to produce a stable (and thus Pareto optimal) matching, Pareto optimality is achieved in this case. Also, as there is no dynamics step here, we need only consider whether the mechanism is strategyproof with respect to the DA algorithm. As discussed in class, DA is necessarily optimal for one side and non-optimal for the other, and it is strategyproof only for the side for which it is optimal. In this case, the DA algorithm executed is buyer-optimal, and thus this matching mechanism is strategyproof for the buyer, but not for the seller. ■

Note that DA-strategyproofness will not change in any of the following models. Thus for those, we will only consider dynamics-strategyproofness.

5.2 Semi-Stationary Model

Next, we extend our model and consider the case where the sellers are stationary and the buyers are dynamic. For a realistic example of such a market, consider the market for rides at the end of a concert. The sellers anticipate the buyers (and are therefore stationary) and the buyers arrive at slightly different times (some may leave just before the event ends in order to avoid the rush). Furthermore, suppose that each entity has the same departure deadline d. This, in turn, implies that the sellers become critical before any of the buyers do (as the buyers arrive after the sellers). Finally, we will assume that no new buyers arrive after d.

Claim 2. *The case of stationary sellers and dynamic buyers yields a Pareto optimal outcome.*

Proof. Since all of the sellers have the same deadline d, they all become critical at the same time and thus we have a single market-clearing time. As such, at the market-clearing time d, the buyers that are in the market are matched with the sellers that are in the market such that a maximum-weight matching M is produced. Since M is a maximum-weight matching, we have that $\sum_i v_{iM(i)} \geq \sum_i v_{iM'(i)}$, for all other matchings M'. Making an entity better off requires a modification to M. Since the total weight of the modified matching is bounded above by the total weight of M, we see that some other entity must be strictly worse off under the modification. As such, the outcome is Pareto optimal. ■

Additionally, it is important to notice that our mechanism is dynamics-strategyproof for the sellers and buyers.

Claim 3. *The case of stationary sellers and dynamic buyers is a dynamics-strategyproof mechanism for both sellers and buyers.*

Proof. Consider some seller s. Suppose s considers untruthfully reporting deadline $d' < d$. Since the sellers are stationary and have the same deadline d, no buyers will exit the market before d (that is, their own departure times will be equal to or greater than d, and no matches will be finalized until d). However, new buyers may enter the market between d' and d; thus, reporting d' is equivalent to constraining s's set of potential buyers to those buyers who arrive before d' as opposed to before d. This means that s can only be worse off (or the same) under d'.

Now, suppose s considers untruthfully reporting deadline $d' > d$. In this case, since no new buyers will arrive after d, s cannot improve its match after d. However, if its buyer has a departure time d_b such that $d < d_b < d'$, the seller runs the risk of losing his buyer. Thus he can only be disadvantaged by reporting $d' > d$.

For a buyer b, since all sellers will leave the market at time d, reporting a longer departure time will have no effect on their match, and reporting a shorter departure time will only affect them if it causes them to leave the market before d and thus remain unmatched. ■

5.3 Full Model

We now consider the full model in which the buyers and sellers are dynamic and each entity i has arbitrary departure deadline d_i. Again, as in [Ash18], we assume that sellers never get matched before they become critical and that a buyer is discarded only if she is unmatched and becomes critical. We do not discuss Pareto optimality in this case, but we do establish claims about strategyproofness.

Claim 4. *The case of dynamic sellers and buyers is not a dynamics-strategyproof mechanism for the sellers.*

Proof. From the perspective of a seller, it may be propitious to report a shorter deadline. For example, suppose that a seller is tentatively matched to a nearby buyer. If the seller suspects that no better option will arrive before she becomes critical (i.e. there won't be a closer buyer before she departs), and if she additionally suspects that another seller may arrive and snatch her tentative match, then she may as well have reported a shorter deadline to minimize the chance of such an outcome occurring. ■

Claim 5. *The case of dynamic sellers and buyers is a dynamics-strategyproof mechanism for the buyers.*

Proof. This follows directly from the assumption that sellers never get matched before they become critical and buyers are discarded only if they are unmatched and become critical. To see this, suppose a buyer b changes her reported deadline from d to d', such that $d' < d$. Since she spends less time in the market, her available options under deadline d' are a subset of the options under d (i.e. she cannot cause a closer seller to appear in the market by shortening her deadline). As such, all b could hope for is to have her match realized before another buyer comes along and snatches her match (by being closer to the seller). Nevertheless, buyers have no control over a match being realized, so there is nothing b can do to hasten the match. As such, b might as well report d (which gives the possibility of being rematched to a closer seller in the extra $d - d'$ time). Note that we do not consider the option of untruthfully reporting a longer deadline as such a deviation would be unrealistic in practice. More concretely, we suppose that a buyer or seller would incur some non-trivial waiting cost that would offset the benefit of the misreported deadline. ■

5.4 Sybil Attacks

We now consider the consequences of a single seller creating multiple identities. To motivate this discussion, consider the following example: the marketplace is the unit grid and there exists a seller s with fixed position (x_s, y_s). Furthermore, suppose that s is hellbent on being assigned to a buyer such that their Euclidean distance is no more than 0.2. A quick simulation reveals that s would have to wait, in expectation, for approximately 10 buyers (who each happen to have random location in $[0, 1]^2$) before having her criterion fulfilled.

With this example in mind, it is not hard to envisage a malicious attack in which a seller creates multiple identities and benefits from a better match. For instance, suppose the seller arrives at time σ and has true departure deadline d. By creating multiple identities and reporting each possible deadline $d' \in \{1, 2, ..., d\}$, she would obtain a match at each time step (since one of her identities would go critical) and could reject until obtaining a satisfactory assignment. This reasoning follows directly from our discussion of non-strategyproofness from the perspective of the sellers. Recall that a seller can potentially make herself better off by falsely reporting a shorter deadline, so it follows that reporting all $d' \in \{1, 2, ..., d\}$ could benefit the seller.

6 Model Limitations and Future Work

In order make this problem tractable, we have made a number of assumptions which are unrealistic to the application. We discuss the most pressing of those here.

Non-continuity of sellers In our model, after a node completes a transaction, they exit the system. In reality, while riders do tend to be single-use customers, drivers generally persist in the system, re-entering the market upon completing each ride. Importantly, the position of the seller upon re-entry is dependent upon the buyer with which it initially matches. This is related to a second limitation of our current model: imperviousness to routes.

Imperviousness to routes In the current implementation, the utility of matches depends only on the initial positions of the parties and neglects the importance of routes altogether. This is highly unrealistic, as anyone who has ever tried to use ride sharing to complete long trips knows: it is substantially harder to find a driver willing to complete a long trip as opposed to a local one.

Universal deadline d distribution For the majority of our algorithms, we assume that the deadline d of each node is known to the platform, but it is unlikely that it could be known definitively. We attempt to address this with the stochastic departures algorithm in Section 3.3; however, this assumes that all buyers and sellers are identically distributed, whereas in reality, they each have different proclivities.

No learning over time Since each of our simulations is over a relatively short time-span, we do not consider the platforms ability to improve itself over time. This is specifically relevant for determining the deadlines. As discussed above, the distributions for these deadlines are likely to be personalized rather than universal, and it would be better to make a personalized guess for a node's departure time using its past behavior (and perhaps the behavior of other similar nodes). Since we treat all arriving nodes as new individuals, we cannot extend our analysis in this way. However, if learning over time is incorporated into our model, we can consider tracking how long it historically takes a user to give up and exit the market. Note, this does introduce a new avenue of attack on the mechanism, as the user could "game the system" by repeatedly entering and immediately exiting the market in order to trick the platform into thinking that she requires a match quickly.

Amending the first two limitations discussed above would require a substantially more complex model that accounts for routes and ride duration. The second two limitations are equally challenging to address, but are more directly algorithmic nature.

References

[Ash18] Burq M. Dutta C. Jaillet P. Saberi A. Sholley C. Ashlagi, I. Maximum Weight Online Matching with Deadlines. *ArXiv e-prints*, August 2018.

[HSSC08] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.