Assignment Problems in Rental Markets*

David Abraham[†] Ning Chen[‡] Vijay Kumar[§] Vahab S. Mirrokni[¶]

Abstract

Motivated by the dynamics of the ever-popular online movie rental business, we study a range of assignment problems in rental markets. The assignment problems associated with rental markets possess a rich mathematical structure and are closely related to many well-studied one-sided matching problems. We formalize and characterize the assignment problems in rental markets in terms of one-sided matching problems, and consider several solution concepts for these problems. In order to evaluate and compare these solution concepts (and the corresponding algorithms), we define some "value" functions to capture our objectives, which include fairness, efficiency and social welfare. Then, we bound the value of the output of these algorithms in terms of the chosen value functions.

We also consider models of rental markets corresponding to static, online, and dynamic customer valuations. We provide several constant-factor approximation algorithms for the assignment problem, as well as hardness of approximation results for the different models. Finally, we describe some experiments with a discrete event simulator compare the various algorithms in a practical setting, and present some interesting experimental results.

1 Introduction

Online movie rental services such as Blockbuster.com, Netflix.com and Amazon.co.uk are perhaps the most familiar instances of *rental markets* in the Internet. The primary function of centralized rental markets such as these is to repeatedly allocate a rental inventory in accordance with customer demand at successive time instances. Customers return assigned items after some time steps and a central authority reassigns the items to other customers. The basic model behind these markets involves (partial) customer preferences over items, and the rental service aims to satisfy these preferences within the constraints of available inventory. Other objectives include trying to maximize overall resource (inventory) utilization and limiting (perceived) unfairness in the allocation process.

Given the collection of competing objectives, resource constraints and challenging business characteristics (popularity of movies tends to be highly non-uniform and extremely short-lived; there is a deep catalog with very sparse demand in the tail), it is natural that the allocation process involves complex decisions. As we shall see, specific considerations involved in these tradeoffs are to some extent captured by familiar matching problems from mathematical literature. Thus several natural questions of the following form arise: how well does maximization of one objective (such as inventory utilization) serve another (such as fairness)? How can one objective be generalized to include another? And when one objective (such as fairness or popularity) does not have a unique maximum, how do the different maxima compare under another objective? In this paper, we consider a range of such issues, identify several interesting questions, and (partly) answer many of them.

Formally speaking, a rental service repeatedly computes a matching between the two sides of the market (i.e., customers and items), given the preference lists of one (and only one) side of the market. This type of matching markets are called *one-sided matching markets* as only one side of the market has preference over the

*This work was done when the authors were visiting Amazon.com.

[†]Carnegie Mellon University. Email: dabraham@cs.cmu.edu.

[‡]University of Washington. Email: ning@cs.washington.edu.

[§]Amazon.com. Email: vijayk@amazon.com.

[¶]Microsoft Research. Email: mirrokni@theory.csail.mit.edu.

other side. This is contrast to *two-sided matching markets* in which both sides of the market have preferences over the other side.

The preferences of customers are often *ordinal* in that they only explain the relative ranking of the items for individual customers. As noted above, optimality in allocation is not clearly defined as two matchings only based on the ordinal preferences of customers may not be comparable. This nuance underlies several notions of one-sided matching objectives studied in recent literature, such as *pareto-optimal matchings* [1], *fair matchings* [15], *rank-maximal matchings* [11], and so forth. In this context, we examine different measures that may be used to choose between non-comparable matchings and analyze different one-sided matching algorithms in terms of these measures.

Two leading criteria to measure the allocation performance of the rental service are that of *social welfare* and the *fairness* of allocation. In this following, we consider different algorithms for a single one-sided matching seeking a reasonable social welfare and fairness and compare the *value* of the output of these algorithms. For this purpose we require the measures of the *value* of the allocations, which must be defined in respect to the preference lists of customers to capture the social welfare and the fairness of the output. Under different measurese, we analyze the value of the output of different one-sided matchings for a single assignment; and then extend the results to repeated matchings for the rental market problem. Our metric to measure these matching algorithms is similar to that of the competitive analysis. That is, we study the ratio of the value of the matchings resulting from the one-sided matching algorithms over the value of the optimal matching.

1.1 One-Sided Matching Markets

Consider the following classical *one-sided matching problem*: we are given a set A of m customers and a set B of n items with one copy[1] for each $j \in B$. Each customer $i \in A$ has a preference list $\mathcal{L}_i = (b_i^1, \ldots, b_i^{\ell_i})$ over different items, where $\ell_i = |\mathcal{L}_i|$ and $b_i^j \in B$ for all $1 \leq j \leq \ell_i$. In a *matching*, items are assigned to customers so that each customer i gets at most one item and each item j is assigned to at most one customer. Since the vertices of one (and only one) side of the corresponding bipartite graph has a preference list, we call the matching in this setting *one-sided matching.*

Roughly speaking, our goal is to assign the customers to items which are among the top of their lists. More formally, let us consider associating a value $v(i, j)$ for assigning item j to customer i, and let our general goal be to find one-sided matchings to maximize the total value of the assignments in terms of the given valuations. We denote such valuations on the items to customers by $\vec{v}$.

This valuation function, however, should have some desired properties. The first natural property is that the function should be non-decreasing, i.e., $v(i, b_i^1) \geq v(i, b_i^2) \geq \cdots \geq v(i, b_i^{\ell_i}) > 0$ and $v(i, j) = 0$ for all other items j that are not on the list $\mathcal{L}_i$, where equality only reflects ties among items. Secondly, the customers tend to have stronger preference over the top choices in their preference list. We can model this fact by considering the concave valuation functions, i.e., $v(i, b_i^j) - v(i, b_i^{j+1}) \geq v(i, b_i^{j+1}) - v(i, b_i^{j+2})$, for $1 \leq j < \ell_i - 1$. Moreover, in the valuation function, we would not want to "favor" any customer too much. For simplicity, let us say we would like to give the same value to the first choices of all customers and the same value to the second choices of all customers and so on. We call such functions satisfying the above conditions by the *universal ranking valuation functions.*

In particular, we are interested in the following special universal ranking valuation functions: for each customer i, the value of her jth-ranked item is $(n - j + 1)^k$ (i.e., $v(i, b_i^j) = (n - j + 1)^k$), for all $1 \leq i \leq n$, $1 \leq j \leq \ell_i$, and some fixed constant $k \geq 0$. We denote this valuation vector by $\vec{x^k}$. Note that when $k = 0$, this valuation function models the cardinality of the matching (i.e., $v(i, b_i^j) = 1$).

In this paper, we consider and analyze several one-sided matching frameworks that are listed below:

Maximum Weighted Matching. As described above, in the maximum weighted matching, we associate a valuation vector $\vec{v}$ to items and customers and maximize the total value of the one-sided matching. The maximum weighted matching associated with valuation vector $\vec{v}$ is denoted by $\mathsf{MaxWeightMatch}(\vec{v})$.

Rank-Maximal Matching. The *profile* of a one-sided matching M is a vector, where the jth element of the profile is the number of customers allocated to their jth-ranked item by M. M is *rank-maximal* if it has

[1]Note that all our results in the paper apply to the multiple copies case.

the lexicographically maximum profile. The rank-maximal matching M is denoted by $\mathsf{RankMaxMatch}$. This solution concept for one-sided matching has been suggested by Irving [12] and later explored by Irving et al. [11].

Weighted Rank-Maximal Matching. Given a valuation vector $\vec{v}$, the *weighted profile* of a one-sided matching M for $\vec{v}$ is a vector, where the jth element of the profile is the value of the jth largest value among the values of the pairs of M. M is *weighted rank-maximal* for vector $\vec{v}$ if it has the lexicographically maximum weighted profile for the value vector $\vec{v}$. A weighted rank-maximal matching is denoted by $\mathsf{WeightRankMaxMatch}$.

Fair Matching. A fair matching has the fewest number of unmatched customers (i.e., it has maximum-cardinality), and subject to this, matches the fewest number of customers to their nth-ranked item, and subject to this, matches the fewest number of customers to their $(n-1)$th-ranked item, and so on. (This definition can be formalized in terms of lexicographically-minimum reverse profiles, where we pad each customer's preference list with dummy items). The fair matching is denoted by $\mathsf{FairMatch}$. Mehlhorn and Michail [15] considered this solution concept for the one-sided matching problems.

Order-Based Matching. Consider an arbitrary ordering $\pi : A \rightarrow \{1, \dots, m\}$ of customers, the order-based matching algorithm for the ordering π goes over the list of customers according to π and for each customer i, it assigns the first available item on i's preference list to i. This algorithm is very simple and scalable to implement. Moreover, in order to achieve different goals in the assignment, we could change the ordering of customers. For example, in order to favor the new customers or the more profitable customers, we can put them at the beginning of the ordering. A matching resulted from the order-based matching algorithm for the ordering π is denoted by $\mathsf{OrderMatch}(\pi)$.

Note that the ordering of customers may differ at different time steps and may depend on the allocations of the previous time steps. For example, in order to achieve some fairness properties, we can favor the customers who did not get their first choices recently in the ordering and put them at the beginning of the ordering.

Stable Matching. Stable matchings are the well-known solution concepts for two-sided matching problems. In a two-sided matching problem, both sides have preference lists over the elements of the other side. In order to extend our setting from a one-sided matching to a two-sided matching problem, we need to define a preference list over customers for each item. We define the preference list for an item j by first listing the customers who have item j as their first choice in an arbitrary order, then listing all customers who have item j as their second choice, and so on. By defining these preference lists for items, we can apply the stable matching algorithm on the two-sided matching instance and output the resulting assignment. This matching is denoted by $\mathsf{StableMatch}$.

The above solutions are the main algorithms that we study in this paper.

1.2 The Rental Market Problem

Rental markets seek to compute one-sided matching, of course, but they also have a time dimension. Roughly speaking, rental markets are frameworks for repeated one-sided matching. More formally, let us say that we are given a set B of n items and a set A of m customers with preference lists $\mathcal{L}_i$ over items. In the *rental market problem*, we need to assign a matching of items to customers at each discrete time step $t = 1, 2, \dots, T$, where T is the common deadline. We assume that customers use the items for one time step and items can be reused after that. Besides the requirements that at each time, one customer can be assigned at most one item and one item can be assigned to at most one customer, the other requirement in the rental market problem is that one item can be assigned to one customer at most once.

We associate a value $v_t(i, j)$ for assigning item j to customer i at time step t. Our goal in the rental market problem is to find a set of matchings for all time steps to maximize the total value. We consider three different types of valuations: the static, online and the dynamic valuations. Roughly speaking, in the *static valuation model*, the value $v_t(i, j)$ is determined at the beginning $t = 1$, whereas in the *dynamic valuation model*, $v_t(i, j)$

directly depends on the position of j on i's preference list at time step t (i.e., it may change according to the assignments of previous steps). On the other hand, in the *online valuation model*, customers can update their preference lists (add new items or remove available items). We will elaborate the details of these models in Section 3.

1.3 Our Contribution

In this paper, we formalize the rental market problem as a repeated one-sided matching problem. We propose some value functions to measure the performance of the assignments in the rental market problem to capture the fairness and the social value of the output of different algorithms. We analyze several one-sided matching algorithms and give (almost) tight bounds on the performance in terms of those value functions for a single one-sided matching problem. These bounds are summarized in Table 1.

Then, we formalize the rental market in three models: The static valuation setting, online valuation setting and the dynamic valuation setting. In the static valuation setting, we show that there exists a 2-approximation algorithm by a reduction from the problem to the weighted 3-dimensional matching problem. As a hardness result, we prove the APX-hardness of the rental market problem. For the online valuation model, we derive a 2-competitive online algorithm to maximize the total value of the assignments. For the dynamic valuation model, we observe that the problems are similar to general variants of the job shop scheduling problems. As a result, we get a constant-factor approximation for the problem of minimizing the number of time steps to satisfy all the demand where at each step, we are allowed to assign one of the few top choices of each customer to her. We also give a hardness result of approximation for this model.

Finally, we give a description of our discrete event simulator for measuring the performance of most of these algorithms on a sample data (and will report some practical evaluations of our algorithms). We conclude the paper with some directions and open problems in the last section.

1.4 Related Work

As mentioned earlier, stable matchings are extensively studied as a solution concept for two-sided markets in which both sides of the market have preferences over the other side [5, 7]. For one-sided matchings, Irving [12] introduced the concept of the rank maximal matchings and observed that they can be found by computing the maximum weighted matching in an edge-weighted bipartite graph where the edge weights are exponentially decreasing with respect to the preferences. Irving et al. [11] derived an algorithm with the running time $O(nm2\sqrt{m+n})$ for this problem. Mehlhorn and Michail [15] studied fair matchings and gave some efficient algorithms to find them. To the best of our knowledge, none of the above work analyzed the value of these one-sided matching algorithms for their worst-case performance. The only related paper in this regard is by Abraham et al. [1] in which the authors studied the structure of pareto-optimal solutions and pareto-optimality of some of the one-sided matching algorithms.

1.5 Notations

Recall that for a given valuation vector $\vec{v}$, $\mathsf{MaxWeightMatch}(\vec{v})$ denotes the one-sided matching with the maximum value. If the value of all items in the preference list is one, i.e., $v(i, b_i^j) = 1$ for all $1 \le i \le n$ and $1 \le j \le \ell_i$, the maximum-value one-sided matching is indeed the maximum cardinality matching and denoted by $\mathsf{MaxCardMatch} = \mathsf{MaxWeightMatch}(\vec{1})$. In addition, the value of a one-sided matching M for the valuation vector $\vec{v}$ is denoted by $\mathsf{Val}(M, \vec{v})$, and the cardinality of a one-sided matching M is denoted by $\mathsf{Card}(M)$.

2 Single Matching Algorithms

To understand the performance of different one-sided matching algorithms in the rental market problem, we need to define some universal objective functions to evaluate these matching algorithms. In particular, we evaluate the performance of a one-sided matching in terms of the value function over the pairs of customers and items. As discussed in the Introduction, we are interested in the special universal ranking valuation functions

$\vec{x^k}$, for some fixed constant $k \geq 0$, where for each customer i, the value of its jth-ranked item is $(n-j+1)^k$ (i.e., $v(i, b_i^j) = (n-j+1)^k$), for all $1 \leq i \leq n$ and $1 \leq j \leq \ell_i$.

In this section, we prove several bounds on the ratio of the value of our proposed algorithms by the worst-case analysis. We summarize the results of this section in Table 1.

	Approximation factor				Pareto-optimal	Running time
	Card	$\mathsf{Val}(\vec{x})$	$\mathsf{Val}(\vec{x^4})$	$\mathsf{Val}(\vec{x^k})$		
MaxCardMatch	1^*	ϵ^*	ϵ^*	ϵ^*	Exist	$O(e\sqrt{v})$ [9]
MaxWeightMatch($\vec{x}$)	0.5, $\frac{2}{3}$	1^*	$0.2, 1$	-	Yes	$O(e\sqrt{v}\log v)$ [4]
MaxWeightMatch($\vec{x^4}$)	0.5, $\frac{4}{7}$	$\frac{n+1}{4n}, 1$	1^*	-	Yes	$O(e\sqrt{v}\log v)$ [4]
MaxWeightMatch($\vec{x^{k'}}$)	0.5^*	$\frac{n+1}{4n}, 1$	$0.1, 1$	-	Yes	$O(ek'\sqrt{v}\log v)$ [15]
RankMaxMatch	0.5^*	0.5^*	0.5^*	0.5^*	Yes	$O(ev)$ [11]
WeightRankMaxMatch	0.5^*	0.5^*	0.5^*	0.5^*	Yes	$O(ev)$ [11]
FairMatch	1^*	$\frac{n+1}{2n}^*$	-	ϵ^*	Yes	$O(e\sqrt{v}\log v)$ [15]
OrderMatch	0.5^*	0.5^*	0.5^*	0.5^*	Exist	$O(e+v)$
StableMatch	0.5^*	0.5^*	0.5^*	0.5^*	Yes	$O(e+v)$ [5]

Notes: (i) The star symbol $(*)$ implies that the ratio is (almost) tight.
(ii) Two numbers a, b implies the best known lower and upper bound.
(iii) "ϵ" means that the ratio can be arbitrarily close to zero.
(iv) In the running time, $v = \max\{m, n\}$ and $e = \sum_{i=1}^{m} \ell_i$.

Table 1: The performance of one-sided matching algorithms.

2.1 Approximation Factor: Lower and Upper Bounds

Due to space limit, we move the discussions of the tight bounds for cardinality of the maximum weighted matching, order-based matching, stable matching, and (weighted) rank-maximal matching to the Appendix. In the following discussions, we consider the bounds for the fair and maximum weighted matching. To prove our bounds, We first establish the following two lemmas.

Lemma 1 *Let M be either a* FairMatch *or a* MaxWeightMatch *w.r.t valuation function $\vec{x}$, and $|M| = \ell$. For any w, where $n - \ell + 1 \leq w \leq n$, we have*

$$|\{(a_i, b_i) \in M \mid v(a_i, b_i) \leq w\}| \leq w - n + \ell.$$

Basically, the lemma says that in the FairMatch or MaxWeightMatch M w.r.t valuation function $\vec{x}$, there is at most one edge with value smaller than or equal to $n - \ell + 1$, at most two edges with value smaller than or equal to $n - \ell + 2$, and so on.

For any constant $k \geq 1$, we can show similarly the following result.

Lemma 2 *For any constant $k \geq 1$, let M be a* MaxWeightMatch($\vec{x^k}$) *and $|M| = \ell$. For any w, where $n-\ell+1 \leq w \leq n$, we have*

$$\left|\{(a_i, b_i) \in M \mid v(a_i, b_i) \leq w^k\}\right| \leq w - n + \ell.$$

Note that any FairMatch first try to minimize the number of unmatched customers, thus it's essentially a MaxCardMatch. To compare the FairMatch with MaxWeightMatch($\vec{x^k}$), we will first give an example to show the upper bound for any $k \geq 1$, and then study the lower bound for the case of $k = 1$.

Assume there are n customers $(a_1, \ldots, a_n)$ and n items $(b_1, \ldots, b_n)$. Each customer a_i prefers items $b_i, b_{i+1}, \ldots, b_n$ on her list. For $i = 1, \ldots, n-1$, customer a_i puts b_i the last choice and b_{i+1} the first choice on her list, respectively. All other items on the list can be ranked arbitrarily. Thus, $\{(a_1, b_1), \ldots, (a_n, b_n)\}$ is a $\mathsf{FairMatch}$ with total value $1^k + 2^k + \cdots + n^k$. The $\mathsf{MaxWeightMatch}$ is $\{(a_1, b_2), \ldots, (a_{n-1}, b_n)\}$ with total value $(n-1)n^k$. Note that when $k = 1$, the ratio is $\frac{n+1}{2n-2}$; when $k = 4$, the ratio approaches to $1/5$ when n goes to infinity; and when n and k are sufficiently large, the ratio can be arbitrarily close to zero.

Now let's consider the relation between the $\mathsf{FairMatch}$ and $\mathsf{MaxWeightMatch}(\vec{x})$. Assume M is a $\mathsf{MaxWeightMatch}(\vec{x})$ and $|M| = \ell$. Note that $\mathsf{Val}(M) \le \ell n$. Let M^* be a $\mathsf{FairMatch}$. Since the $\mathsf{FairMatch}$ is also a $\mathsf{MaxCardMatch}$, we know $|M^*| \ge \ell$. Due to Lemma 1, we know $\mathsf{Val}(M^*) \ge \sum_{w=n-\ell+1}^{n} w$. Thus,

$$\begin{aligned}\frac{\mathsf{Val}(M^*)}{\mathsf{Val}(M)} &\ge \frac{(n-\ell+1) + \cdots + n}{\ell \cdot n} \\ &= \frac{2n - \ell + 1}{2n} \\ &\ge \frac{n+1}{2n}\end{aligned}$$

We conclude the above analysis as the following proposition.

Proposition 1 *For the universal ranking valuation function $\vec{x}$, we have*

$$\mathsf{Val}(\mathsf{FairMatch}, \vec{x}) \ge \frac{n+1}{2n} \cdot \mathsf{Val}(\mathsf{MaxWeightMatch}(\vec{x}))$$

Finally, we give some bounds for the value of the maximum weighted matching algorithms with valuation functions $\vec{x}$ and $\vec{x^4}$. We first consider the valuation function $\vec{x}$. Let M be a $\mathsf{MaxWeightMatch}(\vec{x})$ where $|M| = \ell$. Due to Lemma 1, we know that $\mathsf{Val}\left(M, \vec{x^4}\right) \ge \sum_{w=n-\ell+1}^{n} w^4$. On the other hand, consider the $\mathsf{MaxWeightMatch}(\vec{x^4})$ M^*, note that $\mathsf{Val}(M^*, \vec{x}) \le \mathsf{Val}(M, \vec{x}) \le \ell n$, which implies that $\mathsf{Val}\left(M^*, \vec{x^4}\right) \le \ell n^4$. Thus,

$$\begin{aligned}\frac{\mathsf{Val}\left(M, \vec{x^4}\right)}{\mathsf{Val}\left(M^*, \vec{x^4}\right)} &\ge \frac{\sum_{w=n-\ell+1}^{n} w^4}{\ell \cdot n^4} \\ &\ge \frac{\sum_{w=1}^{n} w^4}{n \cdot n^4} \\ &= \frac{1/30 \cdot n(n+1)(2n+1)(3n^2+3n-1)}{n \cdot n^4} \\ &\ge 1/5\end{aligned}$$

Proposition 2 $\mathsf{Val}\left(\mathsf{MaxWeightMatch}(\vec{x}), \vec{x^4}\right) \ge 1/5 \cdot \mathsf{Val}\left(\mathsf{MaxWeightMatch}(\vec{x^4}), \vec{x^4}\right)$.

On the other hand, Let M be a $\mathsf{MaxWeightMatch}(\vec{x^k})$ where $|M| = \ell$, for any constant $k > 1$. Due to Lemma 2, we know that $\mathsf{Val}\left(M, \vec{x^4}\right) \ge \sum_{w=n-\ell+1}^{n} w^4$. Consider the $\mathsf{MaxWeightMatch}(\vec{x^4})$ M^*, it is easy to see that $|M^*| \le 2|M| = 2\ell$, which implies that $\mathsf{Val}\left(M^*, \vec{x^4}\right) \le 2\ell n^4$. Thus,

$$\begin{aligned}\frac{\mathsf{Val}\left(M, \vec{x^4}\right)}{\mathsf{Val}\left(M^*, \vec{x^4}\right)} &\ge \frac{\sum_{w=n-\ell+1}^{n} w^4}{2\ell \cdot n^4} \\ &\ge 1/10\end{aligned}$$

Proposition 3 $\mathsf{Val}\left(\mathsf{MaxWeightMatch}(\vec{x^k}), \vec{x^4}\right) \ge 0.1 \cdot \mathsf{Val}\left(\mathsf{MaxWeightMatch}(\vec{x^4}), \vec{x^4}\right)$, *for any $k > 1$.*

Consider another case: For any constant $k \geq 1$, let M^* be a $\mathsf{MaxWeightMatch}(\vec{x^k})$ where $|M^*| = \ell$. Due to Lemma 2, we know $\mathsf{Val}(M^*, \vec{x}) \geq \sum_{w=n-\ell+1}^{n} w$. Let M be a $\mathsf{MaxWeightMatch}(\vec{x})$. Again, note that $|M| \leq 2|M^*| = 2\ell$, thus, $\mathsf{Val}(M, \vec{x}) \leq 2\ell n$. Therefore,

$$\begin{aligned}\frac{\mathsf{Val}\left(M^*, \vec{x}\right)}{\mathsf{Val}\left(M, \vec{x}\right)} &\geq \frac{\sum_{w=n-\ell+1}^{n} w}{2\ell \cdot n} \\ &= \frac{\ell(2n-\ell+1)}{4\ell \cdot n} \\ &\geq \frac{n+1}{4n}\end{aligned}$$

Proposition 4 *For any $k \geq 1$,*

$$\mathsf{Val}\left(\mathsf{MaxWeightMatch}(\vec{x^k}), \vec{x}\right) \geq \frac{n+1}{4n} \cdot \mathsf{Val}\left(\mathsf{MaxWeightMatch}(\vec{x}), \vec{x}\right)$$

2.2 Pareto-Optimality

We say an allocation of items to customers is *Pareto-optimal* if there is no other allocation with some customers better and no one worse.

Proposition 5 *There is a* MaxCardMatch *that is Pareto-optimal.*

Proposition 6 *For any universal ranking valuation function $\vec{v}$,* MaxWeightMatch($\vec{v}$)*,* RankMaxMatch*,* WeightRankMaxMatch*,* FairMatch*,* OrderMatch*, and* StableMatch *are Pareto-optimal.*

Note that for the OrderMatch with ties, we can show similar to Proposition 5 that there exists a Pareto-optimal solution, but in general, the OrderMatch is not Pareto-optimal. However, if it is not allowed to have ties, the OrderMatch guarantees Pareto-optimal.

3 The Rental Market Problem

In this section, we study the rental market problem with a focus on static, online, and dynamic valuations, respectively.

3.1 Static Valuation Model

In the following discussion, we study a static valuation model to evaluate the performance of the rental market problem. In the *static valuations model*, at each time t, $t = 1, \ldots, T$, where T is the deadline that customers can get the items, there is a valuation $v_t(i, j)$, which is determined at the beginning, associated with the pair (i, j), for any $(i, j) \in A \times B$, representing the valuation of the customer $i \in A$ for item $j \in B$ at time step t. Our goal is to select one-sided matchings M_t for each time t that maximizes $\sum_{t=1}^{T} \sum_{e \in M_t} v_t(e)$, given the condition that every pair is selected at most once, that is, each item can be assigned to each customer at most once. We denote the rental market problem in the static valuation model by StaticRentMark.

We reduce StaticRentMark problem with arbitrary valuation functions to the weighted 3-dimensional matching problem (W3DM). This implies a local search 2-approximation algorithm for this problem. In an instance of W3DM, given a subset D of triples in set $X \times Y \times Z$ where X, Y, and Z are disjoint sets, and a weight w_e for each triple of D, we need to find a set of triples $C \subseteq D$ with the maximum weight such that no two elements of C agree in any coordinate. W3DM is APX-complete [13] and a local search two-approximation algorithm is known for it [2].

Theorem 1 *For any static valuation function $\vec{v}$, there exists a 2-approximation algorithm for the* StaticRentMark *problem.*

Proof. Given an instance $\mathcal{S}(A, B; T, \vec{v})$ of the StaticRentMark problem, where T is the deadline time and $\vec{v}$ is the valuation function, we construct an instance $\mathcal{G}(\mathcal{S})$ of W3DM as follows: Let $[T] = \{1, \ldots, T\}$. Define $X = (A \times B)$, $Y = (A \times [T])$ and $Z = (B \times [T])$. For any triple $e = ((i, j), (i', t), (j', t')) \in X \times Y \times Z$, define the weight

$$w_e = \begin{cases} v_t(i, j) & \text{if } i = i', j = j', t = t' \\ 0 & \text{otherwise} \end{cases}$$

Now, it is easy to check that there exists a set of T one-sided matchings as the solution to the instance $\mathcal{S}$ with the total value w, if and only if, there exists a weighted 3-dimensional matching of weight w in the instance $\mathcal{G}(\mathcal{S})$. As a result, we can use the local search two-approximation algorithm of Arkin and Hassin [2] to achieve a two-approximation algorithm for StaticRentMark with any valuation function. □

Next, we complement this result by showing that StaticRentMark with arbitrary valuation functions is APX-Hard.

Theorem 2 *The* StaticRentMark *problem is APX-hard.*

3.2 Online Valuation Model

The 2-approximation algorithm for StaticRentMark above is based on a local search algorithm and does not provide an online algorithm. In this section, we consider a *online valuation model* in which customers can arbitrarily update their preference lists (add new items or remove available items[2]) for every time step, and at time t, we only know the preference lists and the corresponding valuations till this time. We denote this problem by OnlineRentMark.

Similar to the above models, our goal in OnlineRentMark is to assign items to customers at each time step to maximize the total value of assignments. In order to evaluate the performance of the online the algorithm, we follow the approach of the competitive analysis that compares the efficiency of the solution with a global optimum (assuming the knowledge of the future in advance).

Before studying the online algorithm for OnlineRentMark, we need to specify the valuation functions. In general, the valuations of customers over items decrease as time passes by. Therefore, it is reasonable to assume *non-increasing valuation functions*, i.e., $v_{t'}(i, j) \geq v_t(i, j)$, for any $1 \leq t' < t \leq T$, when item j is on customer i's list at both time t' and t.

A natural greedy strategy is that at each time t, we compute the current maximum weighted matching in terms of the available pairs and valuations at time t, and allocate the items to customers according to that matching. As the following theorem shows, this greedy algorithm has a good competitive ratio.

Theorem 3 *For any non-increasing valuation function $\vec{v}$, the above greedy algorithm gives a 2-competitive algorithm to the* OnlineRentMark *problem.*

Proof. Let OPT_t be the set of pairs selected by the optimal solution at time step t, and ALG_t be the set of pairs selected by the greedy online algorithm at time t. Let

$$OPT^* = \sum_{t=1}^{T} \sum_{e \in OPT_t} v_t(e)$$

be the total value of the optimal offline solution, and

$$ALG^* = \sum_{t=1}^{T} \sum_{e \in ALG_t} v_t(e)$$

[2]For a more general setting in which users can change their preferences arbitrarily, we cannot hope to get any bounded competitive ratio.

be the total value of the greedy online algorithm. Let

$$X_t = OPT_t \cap \left(\bigcup_{i=1}^{t} ALG_i \right).$$

That is, X_t is the set of selected pairs in the optimal solution at time t that appear in the greedy online algorithm no later than time step t. Let $Y_t = OPT_t - X_t$.

For any $e = (i, j) \in X_t$, assume $e \in ALG_{t'}$, where $t' \in \{1, \ldots, t\}$. Note that item j appears on customer i's list at both time t' and t. Due to the non-increasing property, we have $v_t(e) \leq v_{t'}(e)$. Therefore,

$$\sum_{t=1}^{T} \sum_{e \in X_t} v_t(e) \leq \sum_{t=1}^{T} \sum_{e \in ALG_t} v_t(e) = ALG^*.$$

For the set Y_t, note that all pairs in Y_t are available in the greedy online algorithm at time t. Thus, Y_t is a feasible candidate set of the greedy algorithm. Due to the maximum weighted matching strategy, we have

$$\sum_{e \in Y_t} v_t(e) \leq \sum_{e \in ALG_t} v_t(e),$$

for any t. Thus, we have

$$\sum_{t=1}^{T} \sum_{e \in Y_t} v_t(e) \leq \sum_{t=1}^{T} \sum_{e \in ALG_t} v_t(e) = ALG^*.$$

Therefore,

$$OPT^* = \sum_{t=1}^{T} \sum_{e \in X_t} v_t(e) + \sum_{t=1}^{T} \sum_{e \in Y_t} v_t(e) \leq 2ALG^*,$$

which completes the proof of the theorem. □

Another advantage of the above greedy online algorithm is that it can be easily modified by changing the one-sided matching algorithm at each time step to get another competitive algorithm which may satisfy some extra desirable properties. We can combine the proofs of Section 2 and the above proof to bound the efficiency of the assignments resulting from using one of the aforementioned one-sided matching algorithms at each time step. For example, we could run a stable matching algorithm to find an assignment at each time step. The resulting algorithm is thus a $\frac{1}{4}$-competitive online algorithm for any non-increasing universal ranking valuation function.

3.3 Dynamic Valuation Model

A drawback of StaticRentMark and OnlineRentMark is that it ignores the effect of allocations in the previous time steps on the valuations of later time steps (OnlineRentMark essentially reflects the perspectives and changes of customers, but not allocations). We illustrate this by the following example. Let the preference list of customer i be (b_1, b_2, b_3). If we assign items b_1, b_2, and b_3 to customer i at the first three time steps, respectively, we have assigned the first choice of i to her every time step. In other words, the value of assigning b_2 to i at time step 2 for customer i is larger if item b_1 is assigned to i at time step 1. To capture this aspect, we formalize the rental market problem with dynamic valuations, denoted by DynamicRentMark, as follows: Let $r_t(i, j)$ be the jth-ranked item on customer i's preference list at time t. For every time step t, $t = 1, \ldots, T$, the value of assigning $r_t(i, j)$ to customer i at time t is $g(i, j)$.

The main difference between DynamicRentMark and the other two models is that the value of assigning an item in DynamicRentMark only depends on the position of the item on the preference list of the customer at the time of the assignment (that is, $r_t(i, j)$ is a dynamic function in terms of the previous allocation), but in

StaticRentMark and OnlineRentMark, the value depends on the time step of the assignment and not directly on the position of the item at the time of the assignment[3].

First, we observe that a special case of the DynamicRentMark problem is the job-shop scheduling problem with unit-length jobs on parallel machines (JobShopSch) [14, 17, 6, 18]. In the JobShopSch problem, we have a set of m jobs and n machines. Each machine can run at most one job at a time. Each job i consists of n_i operations o_j^i. Each operation o_i^j has a type t_i^j and can only be scheduled on machine with t_i^j. We need to schedule the operations of each job in the order $(o_i^1, o_i^2, \dots, o_i^{n_i})$. There are two variations of this problem: In the minimization variant (MinJobShopSch [14]), we need to schedule all the operations of all jobs in the minimum number of time steps, that is we need to minimize the makespan of the schedule. In the maximization variant (MaxJobShopSch), we want to maximize the number (or the total value) of the operations that are scheduled before a deadline T. Constant-factor approximation algorithms are known for MinJobShopSch [14], but no constant-factor approximation algorithm is known for MaxJobShopSch.

JobShopSch is a special case of DynamicRentMark in which $g(i, 1) = 1$ and $g(i, j) = -M$ for any $j > 1$ and sufficiently large value M. Each machine corresponds to an item in DynamicRentMark. Jobs in JobShopSch correspond to customers in DynamicRentMark and their operations correspond to the preference list of customers. As a result, for this value function, the known results for MinJobShopSch give a good approximation algorithm for the minimization version of DynamicRentMark. Moreover, designing a constant-factor approximation for the maximization version of DynamicRentMark will solve the open problem of approximating MaxJobShopSch.

Here, we formalize a more general value function and prove similar results for the DynamicRentMark problem. Consider the following dynamic value function: Given any constant $k \geq 1$, let $g(i, j) = 1$ for any $j \leq k$ and $g(i, j) = -M$ for any $j > k$ and a sufficiently large value M. In other words, at each step, we can assign only one of the first k choices of any customer to her. Our goal is to minimize the number of time steps for assigning all the items to customers (with the restriction of assigning only the first k choices). We call this problem MinDynamicRentMark(k). The MinJobShopSch problem corresponds to MinDynamicRentMark(1). We observe that the constant-factor approximation for MinJobShopSch can be used to give a constant-factor approximation for MinDynamicRentMark(k) for any $k \geq 1$.

Corollary 1 *For any constant $k \geq 1$, there exists a polynomial-time constant-factor approximation algorithm for* MinDynamicRentMark(k) .

In the following, we give a hardness result for MinDynamicRentMark(2).

Theorem 4 *It is NP-hard to approximate the* MinDynamicRentMark(2) *problem within a factor better than* 1.2.

4 Practical Evaluation

In this section, we describe our discrete event simulator and report the performance of different algorithms.

4.1 Discrete Event Simulator

DVD rental businesses are more complicated than the theoretical models we have analyzed here. For example, customers typically have a choice of *subscription plans*, which determine, say, how many DVDs they can borrow at once and in a given month. These plans have a big influence on the DVD *return times* - i.e. the time between when a DVD is borrowed and returned. This complicates matching decisions: should we allocate a DVD to a customer today, or wait until tomorrow when we may be able to give them a better DVD?

In our theoretical models, there is a fixed collection of DVDs available for rental. However, rental businesses have control over their inventory levels: if there aren't enough DVDs of a particular title, more can be purchased. Of course, with this control comes the problem of determining optimum inventory levels, which to some degree involves trading-off between customer satisfaction and financial sustainability. Inventory levels must also take

[3]A more general model is that of the combination of OnlineRentMark and DynamicRentMark, where the value of assigning at item only depends on its current position on the list and customers can update their preference lists. We do not study this model in this paper.

into account return times. Customers with slower return times may keep a high-demand DVD for several days beyond watching it. Knowing this, additional copies of the DVD must be bought, even though only a fraction of the DVDs are actively being watched on a given day.

These are just some of the complications dealt with by a real-world DVD Rental business. We cannot hope to fully capture the underlying model and analyze it theoretically. Instead, we have built a discrete event simulator to see the effects of various subscription plans, matching algorithms, inventory planning strategies and so on. The simulator can be seeded by real-world data, including actual customer preference lists, distributions of return times and forecasts of demand. This simulator is used by the DVD Rental business unit at Amazon.com.

4.2 Performance of Different Algorithms

The following table contains some sample results from our simulator. We constructed a small instance from real-world data containing 2000 customers (with existing rental histories and preference lists), 150 DVD titles (5000 DVDs in total) and two types of subscription plans (one with a maximum of 4 DVDs per month, and at most 2 borrowed at any one time; the other with unlimited DVDs per month, and at most 3 borrowed at any one time). Using forecast demand data, we then ran the simulator for a (virtual) three-month period to test the different matching algorithms.

The three objective functions are $\mathsf{Val}(\vec{x})$, $\mathsf{Val}(\vec{x^4})$ and the total number of *skips*. A skip occurs for each higher-ranked DVD a customer misses out on when we perform a matching. If an eligible customer receives no DVD, a skip is recorded for each DVD on his/her preference list. We report this value for different matching algorithms as an alternative measure to compare the results. Note that in Table 1, we report the worst case analysis for a single matching, but in Table 2, we report the total value of matchings for several time steps. As expected from the theoretical results in Table 1, the total value for different algorithms are close to each other.

	Total Skips	$\mathsf{Val}(\vec{x})$	$\mathsf{Val}(\vec{x^4})$
$\mathsf{MaxWeightMatch}(\vec{x})$	12,522	1,137,456	3.7921e+012
$\mathsf{MaxWeightMatch}(\vec{x^4})$	12,962	1,137,555	3.7963e+012
RankMaxMatch	15,802	1,139,654	3.8078 e+012
FairMatch	12,316	1,135,668	3.7861 e+012
OrderMatch	17,059	1,141,082	3.8148 e+012
StableMatch	25,788	1,139,025	3.8126 e+012

Table 2: Simulator Results

Although these objective functions capture the social welfare, they do not reveal the utility variability amongst the customers. Figure 1 shows the number of skips experienced by the 50 customers with the most number of skips. It is of interest to note that fair matching is substantially better for these customers. This is achieved with very little loss in utility w.r.t. $\mathsf{Val}(\vec{x})$ and $\mathsf{Val}(\vec{x^4})$.

5 Conclusions

In this paper, we studied different algorithms for the rental market problem, defined universal measures to compare these algorithms, and analyzed them theoretically and practically. An open problem of this paper is to design a constant-factor approximation algorithm for the maximization version of DynamicRentMark. Such a constant-factor approximation algorithm also gives a constant-factor approximation for MaxJobShopSch.

Designing algorithms with extra fairness properties is an interesting subject of study. For example, we would like to minimize the maximum number of skips that any customer observes. Dealing with strategic agents is another interesting topic. This can be done by proving that for random preference lists, the probability that a customer has incentive to lie tends to zero as the number of customers approaches to ∞.

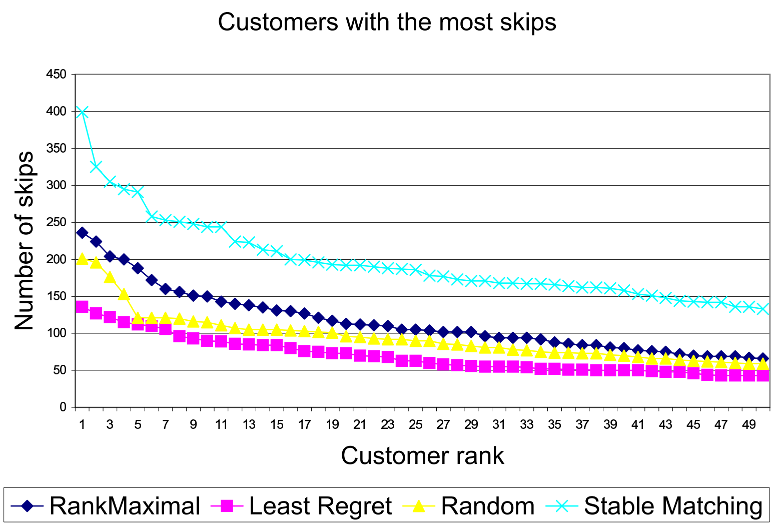

Figure 1: The worst customer experience in each scenario

References

[1] D. Abraham, K. Cechlarov, D. F. Manlove and K. Mehlhorn. *Pareto Optimality in House-Allocation Problems*. ISAAC 2004, 3-15.

[2] E. M. Arkin and R. Hassin, *On Local Search for Weighted Packing Problems*. Math. Oper. Res., V.10(3), 640-648, 1998.

[3] D. Avis. *A Survey of Heuristics for the Weighted Matching Problem*. Networks, V.13, 475C493, 1983.

[4] H. N. Gabow and R. E. Tarjan. *Faster Scaling Algorithms for Network Problems*. SIAM Journal on Computing, 18(5), 1013-1036, 1989

[5] D. Gale and L. S. Shapley. *College Admissions and the Stability of Marriage*. American Mathematical Monthly, V.69, 9-15, 1962.

[6] L. Goldberg, M. Paterson, A. Srinivasan and E. Sweedyk. *Better Approximation Guarantees for Job-Shop Scheduling*. SODA 1997, 599-608.

[7] D. Gusfield and R. W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, 1989.

[8] Holyer. *The NP-Completeness of Some Edge Partitioning Problems*. SIAM journal of Computing, V.10(3), 713-717, 1981.

[9] J. E. Hopcroft and R. M. Karp. *An $n^{5/2}$ Algorithm for Maximum Matching in Bipartite Graphs*. SIAM J. Computing, V.4, 225-231, 1973.

[10] N. Immorlica, M. Mahdian, and V. S. Mirrokni. *Cycle Cover with Short Cycles*. STACS 2005, 641-653.

[11] R. W. Irving, T. Kavitha, K. Mehlhorn, D. Michail and K. Paluch. *Rank-Maximal Matchings*. SODA 2004, 68-75.

[12] R. W. Irving. *Greedy Matchings*. Technical Report TR-2003-136, University of Glasgow, 2003.

[13] V. Kann. *Maximum Bounded 3-Dimensional Matching is MAX SNP-Complete*. Inform. Process. Lett., V.37, 27-35, 1991.

[14] F. T. Leighton, B. M. Maggs and S. B. Rao. *Packet Routing and Job-Shop Scheduling in O(Congestion +Dilation) Steps.* Combinatorica, V.14(2), 167-180, 1994.

[15] K. Mehlhorn and D. Michail. *Network Problems with Non-Polynomial Weights and Applications.* Manuscript, 2005.

[16] R. Preis. *Linear Time 1/2-Approximation Algorithm for Maximum Weighted Matching in General Graphs.* STACS 1999, 259-269.

[17] D. B. Shmoys, C. Stein and J. Wein. *Improved Approximation Algorithms for Shop Scheduling Problems.* SIAM Journal of Computing, V.23, 617-632, 1994.

[18] D. Williamson, L. Hall, J. Hoogeveen, C. Hurkens, J. Lenstra, S. Sevastianov and D. Shmoys. *Short Shop Schedules.* Operations Research, V.45(2), 288-294, 1997.

A Tight Bounds in Section 2.1

First, we give some bounds for the cardinality of the maximum weighted matching.

Proposition 7 *For any $k \geq 1$,*

$$\mathsf{Card}\left(\mathsf{MaxWeightMatch}(\vec{x^k})\right) \geq 0.5 \cdot \mathsf{Card}(\mathsf{MaxCardMatch})$$

Proof. Let M denote the $\mathsf{MaxWeightMatch}(\vec{x^k})$, and M^* the $\mathsf{MaxCardMatch}$. Our goal is to compare the number of edges in M to that of M^*, say $\mathsf{Card}(M)/\mathsf{Card}(M^*)$. Consider the subgraph induced by $M \cup M^*$, to compare $\mathsf{Card}(M)$ and $\mathsf{Card}(M^*)$, we need to compare the number of edges in M and M^* for each connected component. It is easy to see that in each connected component C, the number of edges in $C \cap M^*$ is at most twice of the edges in $C \cap M$, thus we are done. $\square$

The 0.5 ratio is tight when n and k are large. Consider the following example: There are $n+1$ customers $\{a_0, a_1, \ldots, a_n\}$, where n is an even integer, and $n+1$ items $\{b_0, b_1, \ldots, b_n\}$. For $i = 1, \ldots, n$, customer a_i ranks item b_i as her 2nd choice. For $i = 1, 3, \ldots, n-1$, customer a_i ranks item b_{i+1} as her 1st choice, and for $i = 0, 2, \ldots, n$, customer a_i ranks item b_0 as her 1st choice. For this example, we know $\mathsf{Card}(\mathsf{MaxCardMatch}) = n+1$ and $\mathsf{Card}\left(\mathsf{MaxWeightMatch}(\vec{n^k})\right) = 1 + n/2$, when $n^k \geq 2(n-1)^k$. Note that when $k = 4$, we should have $n \leq 6$, thus, the ratio is at most $(1 + 6/2)/(6+1) = 4/7$.

For $k = 1$, the upper bound is $2/3$, as the following example shows: There are three customers $\{a_1, a_2, a_3\}$ and three items $\{b_1, b_2, b_3\}$. The preference lists of customers are:

$$\begin{array}{lllll} a_1 & : & b_3 & b_1 & \\ a_2 & : & b_1 & b_3 & b_2 \\ a_3 & : & b_3 & & \end{array}$$

According to the definition of the valuation function, we have $v(a_1, b_1) = 2$, $v(a_1, b_2) = 0$, $v(a_1, b_3) = 3$, $v(a_2, b_1) = 3$, $v(a_2, b_2) = 1$, $v(a_2, b_3) = 2$, $v(a_3, b_1) = 0$, $v(a_3, b_2) = 0$, $v(a_3, b_3) = 3$. Thus, the $\mathsf{MaxWeightMatch}$ is (a_2, b_1) and (a_3, b_3) whose cardinality is two, whereas the $\mathsf{MaxCardMatch}$ has size three.

Now, we give tight bounds for the order-based and stable matching algorithms.

Proposition 8 *For any $k \geq 0$,*

$$\mathsf{Val}\left(\mathsf{StableMatch}, x^k\right) \geq 0.5 \cdot \mathsf{Val}\left(\mathsf{MaxWeightMatch}(\vec{x^k}, x^k\right)$$

Proof. Let M^* be the $\mathsf{StableMatch}$ and M be the $\mathsf{MaxWeightMatch}(\vec{n^k})$. Our goal is to compare the value of M and M^*, say $\mathsf{Val}(M^*, x^k)/\mathsf{Val}(M, x^k)$.

Consider the subgraph induced by $M \cup M^*$. For each connected component C, assume C is composed by edges $e_1, \ldots, e_l$, where each e_i is connected to e_{i+1} and edges are alternatively contained in M and M^*. For any $e_i \in M$, $1 < i < l$, due to the definition of stable matching, we know $v(e_i) \leq \max\{v(e_{i-1}), v(e_{i+1})\} \leq v(e_{i-1}) + v(e_{i+1})$, since otherwise, the two vertices of the endpoints of e_i would have incentive to be paired together. If e_l is connected to e_1, we know that for any $e_i \in M$, the value of e_i is at most the sum of values of the two adjacent edges, which belong to M^* and we are done. Otherwise, if $e_1 \in M$ (or $e_l \in M$), we must have $v(e_1) \leq v(e_2)$ (or $v(e_l) \leq v(e_{l-1})$), because otherwise, similarly, the two vertices of the endpoints of e_1 (or e_l) would have incentive to be paired together. No matter which cases, $\mathsf{Val}(M^*, x^k)$ is at least half of $\mathsf{Val}(M, x^k)$ in the connected component C. Thus $\mathsf{Val}(M^*, x^K) \geq 0.5 \cdot \mathsf{Val}(M, x^k)$, and the conclusion follows. $\square$

Again the 0.5-approximation ratio is tight. Consider the following example: There are two customers $\{a_1, a_2\}$ and n items $\{b_1, \ldots, b_n\}$. Customer a_1 prefers both b_1 and b_2 as her 1st choice (with a tie) and a_2 only prefers b_1. According to the definition of the valuation function, we have $v(a_1, b_1) = v(a_1, b_2) = v(a_2, b_1) = n^k$, and the value of all other edges are zero. Then the $\mathsf{MaxWeightMatch}(\vec{x^k})$, (a_1, b_2) and (a_2, b_1), has value $2 \cdot n^k$. But (a_1, b_1) and (a_2, b_2) give a solution of $\mathsf{StableMatch}$ with value n^k only.

It is well known that OrderMatch with arbitrary order of vertices gives a 0.5-approximation to the value of the MaxWeightMatch [3, 16]. In the universal ranking valuation function, this ratio is still tight. The tight example for StableMatch algorithm also gives a tight bound for OrderMatch. In that example, the OrderMatch only selects the edge (a_1, b_1) with value n^k, which is half of the MaxWeightMatch.

Now, we analyze the rank-maximal matching.

Proposition 9 *For any universal ranking valuation function $\vec{v}$, we have*

$$\mathsf{Val}(\mathsf{RankMaxMatch}, \vec{v}) \geq 0.5 \cdot \mathsf{Val}\left(\mathsf{MaxWeightMatch}(\vec{v})\right)$$

Proof. Observe that essentially, RankMaxMatch greedily chooses edges in the non-increasing order of values. As discussed above, OrderMatch with any ordering of customers gives at least half approximation of the MaxWeightMatch, we may fix the ordering of customers by the solution of the RankMaxMatch. At this point, RankMaxMatch is equivalent to the OrderMatch, which gives a 0.5-approximation to the MaxWeightMatch. □

The RankMaxMatch can not approximate the MaxWeightMatch by a factor better than 0.5 even for the universal ranking valuation functions. We consider an example of Irving [11] as follows (for simplicity, assume there are five customers and items):

$$\begin{array}{lcll} a_1 & : & b_1 & \\ a_2 & : & b_1 & b_2 \\ a_3 & : & b_1 & b_3 \\ a_4 & : & b_2 & b_4 \\ a_5 & : & b_3 & b_5 \end{array}$$

Note that $M^* = \{(a_1, b_1), (a_4, b_2), (a_5, b_3)\}$ is a RankMaxMatch with profile $\langle 3, 0, 0, 0, 0\rangle$, and $M = \{(a_i, b_i) \mid i = 1, \ldots, 5\}$ is a MaxWeightMatch with profile $\langle 1, 4, 0, 0, 0\rangle$. With the universal ranking valuation function $\vec{x^k}$, we have $\mathsf{Val}(M^*) = \frac{n+1}{2}n^k = \frac{n^{k+1}+n^k}{2}$, and $\mathsf{Val}(M) = n^k + (n-1)(n-1)^k$ (in our example above, we have $n = 5$). It is easy to see that $\frac{w(M^*)}{w(M)} \to 0.5$ when n approaches to infinity.

B Proofs in Section 2

B.1 Proof of Lemma 1

Proof. Assume without loss of generality that $M = \{(a_1, b_1), \ldots, (a_\ell, b_\ell)\}$. Thus, items in $\{b_{\ell+1}, \ldots, b_n\}$ is not allocated to any customer. Our first observation is that for any $1 \leq i \leq \ell$, $v(a_i, b_i) \geq n - \ell + 1$, because otherwise, there exists an edge from a_i to some item in $\{b_{\ell+1}, \ldots, b_n\}$ with value higher than $v(a_i, b_i)$, which is a contradiction.

Note that the result follows trivially when $w = n$. In the following we only consider the case when $w < n$. Assume there are at least $w-n+\ell+1$ edges in M^*, without loss of generality say $M' = \{(a_1, b_1), \ldots, (a_{w-n+\ell+1}, b_{w-n+\ell+1})\}$, with value at most w. Let $A' = \{a_1, \ldots, a_{w-n+\ell+1}\}$ and $B' = \{b_1, \ldots, b_{w-n+\ell+1}\}$. Note that besides these edges, there are $n-w-1$ edges in M, which implies that for each $a_i \in A'$, there is $b_j \in B'$ such that $v(a_i, b_j) > w$. Define

$$M'' = \{(a_i, b_j) \mid a_i \in A', b_j \in B', v(a_i, b_j) > w\}.$$

Now consider the subgraph $G' = M' \cup M''$. Since the degree of each vertex in A' is at least two in G', there is a loop L in G', which is composed of edges in M' and M'' alternatively. Now we can replace those edges in $L \cap M'$ by edges in $L \cap M''$, which gives a better matching solution, a contradiction. □

B.2 Proof of Proposition 5

Proof. Let M be a MaxCardMatch. Assume M is not Pareto-optimal, which implies that there is another matching M' such that some customers get better assignments and no customer gets worse, compared with the assignments of M. Let $A_1 \subseteq A$ be the set of customers that get better assignments in M' and $A_2 = A - A_1$.

Note that if a customer gets an assignment in M, the customer must also get an assignment in M', which implies that $\mathsf{Card}(M) \leq \mathsf{Card}(M')$. Thus, M' is a MaxCardMatch.

Let $\vec{v}$ be any universal ranking valuation function. Due to the Pareto-optimality of M', we have

$$\begin{aligned}\mathsf{Val}(M', \vec{v}) &= \mathsf{Val}_{A_1}(M', \vec{v}) + \mathsf{Val}_{A_2}(M', \vec{v}) \\ &> \mathsf{Val}_{A_1}(M, \vec{v}) + \mathsf{Val}_{A_2}(M', \vec{v}) \\ &= \mathsf{Val}_{A_1}(M, \vec{v}) + \mathsf{Val}_{A_2}(M, \vec{v}) \\ &= \mathsf{Val}(M, \vec{v})\end{aligned}$$

Therefore, for any MaxCardMatch M, when M is not Pareto-optimal, we can always strictly improve the weight of M to another MaxCardMatch M'. Since the values of edges are finite, in the end we can reach to a MaxCardMatch which is Pareto-optimal. □

B.3 Proof of Proposition 6

Proof. To get the flavor of the proof, we only prove here for the RankMaxMatch and all other matchings are similar.

Let M be a RankMaxMatch. Assume M is not Pareto-optimal, which implies that there is another matching M' such that some customers get better assignments and no customer gets worse, compared with the assignments of M. Let $A_1 \subseteq A$ be the set of customers that get better assignments in M' and $A_2 = A - A_1$. Thus, we know that customers in A_1 get higher rank assignments in M' than M, and customers in A_2 get the same assignments in M' and M. Thus, the profile of M' is striclty larger than M, which contradicts to the rank-maximization of M. □

C Proof of Theorem 2

Note that the reduction constructed in the theorem above implies that W3DM is at least as hard as StaticRentMark, but not vice-versa. In order to prove that StaticRentMark is APX-hard, we prove that the partitioning of tripartite graphs into edge-disjoint triangles (EdgeDisjTrianglePar) is APX-complete and we give a reduction from the StaticRentMark to EdgeDisjTrianglePar. Formally, in the EdgeDisjTrianglePar problem, given a tripartite graph $G(V_1, V_2, V_3; E)$ where V_1, V_2 and V_3 are disjoint sets of vertices, and $E \subseteq \{V_1 \times V_2\} \cup \{V_1 \times V_3\} \cup \{V_2 \times V_3\}$, the goal is to find the maximum number of edge-disjoint triangles in G.

Given an instance $\mathcal{G}(V_1, V_2, V_3; E)$ of the EdgeDisjTrianglePar problem, we construct an instance $\mathcal{S}(A, B; T, \vec{v})$ of StaticRentMark by setting $A = V_1$, $B = V_2$, and $T = |V_3|$. Let π be any one to one and onto mapping from $[T] = \{1, \ldots, T\}$ to V_3. Define $v_t(i, j) = 1$ if and only if $\{(i, j), (j, \pi(t)), (i, \pi(t))\} \subseteq E$. It follows that there exist w disjoint triangles in $\mathcal{G}$, if and only if there exists a set of matchings for StaticRentMark in T time steps with total value w. To show the APX-hardness of StaticRentMark, it remains to prove that EdgeDisjTrianglePar is APX-hard.

Holyer [8] proved that edge-partitioning of general graphs into edge-disjoint triangles is NP-complete. A more careful analysis of this proof shows that the edge-partitioning of general graphs is APX-hard [10]. We observe that the set of graphs that Holyer used in his NP-hardness proof for edge-partitioning triangles is in fact tripartite. To see this, we need to define some notations from [8]. Let graph $H_{3,n}$ be a graph with n^3 vertices $V = \{(x_1, x_2, x_3) \in \{0, 1, 2\}^n \mid \sum_{i=1}^{3} x_i = 0(\text{mod } n)\}$. Let $((x_1, x_2, x_3), (y_1, y_2, y_3))$ be an edge in $H_{3,n}$ if there exist $i, j \in \{1, 2, 3\}$, $i \neq j$, such that $x_k = y_k(\text{mod } n)$ for $k \neq i, j$ and $y_i = (x_i + 1)(\text{mod } n)$ and $y_j = (x_j + 1)(\text{mod } n)$. The resulting graph reduced from any 3SAT instance in Holyer's proof is a result of combining and joining $H_{3,p}$'s. It is not hard to verify that $H_{3,n}$ is 3-vertex-colorable and any combination and joint of these graphs is also 3-vertex-colorable. As a result, Holyer's proof of NP-hardness [8] and its extension for APX-hardness [10] of edge-partitioning of general graphs implies the APX-hardness of EdgeDisjTrianglePar. This, in turn, implies that StaticRentMark is APX-hard. □

D Proof of Theorem 4

Proof. We give a reduction from 3SAT problem where each clause has exactly three variables to the problem of deciding if the solution to an instance MinDynamicRentMark(2) is 5 or 6. This shows that approximating MinDynamicRentMark(2) within a factor better than $6/5 = 1.2$ is NP-hard. The idea of the proof is similar to that of [18] for shop scheduling problem. Consider an instance of the 3SAT problem with set of literals $U = \{x_1, \ldots, x_n\}$ and clauses $C = \{c_1, \ldots, c_m\}$, for each literal x_i, we consider its unnegated and negated occurrence: Let the k-th occurrence of x_i be $x_{i,k}$ and of $\bar{x}_i$ be $\bar{x}_{i,k}$, for $k = 1, \ldots, 3m$. For each variable x_i, we associate $6m$ customers $x_{i,1}, \bar{x}_{i,1}, \ldots, x_{i,3m}, \bar{x}_{i,3m}$, with their preference lists as follows:

$$\begin{aligned}
\mathcal{L}_{x_{i,1}} &= \big(I_{a(x_{i,1})}, I_{b(x_{i,1})}, I_{c(x_{i,1})}, I_{d(x_{i,1})}\big)\\
\mathcal{L}_{\bar{x}_{i,1}} &= \big(I_{a(x_{i,1})}, I_{b(x_{i,1})}, I_{c(x_{i,3m})}, I_{d(\bar{x}_{i,1})}\big)\\
\mathcal{L}_{x_{i,2}} &= \big(I_{a(x_{i,2})}, I_{b(x_{i,2})}, I_{c(x_{i,2})}, I_{d(x_{i,2})}\big)\\
\mathcal{L}_{\bar{x}_{i,2}} &= \big(I_{a(x_{i,2})}, I_{b(x_{i,2})}, I_{c(x_{i,1})}, I_{d(\bar{x}_{i,2})}\big)\\
&\cdots\\
\mathcal{L}_{x_{i,3m}} &= \big(I_{a(x_{i,3m})}, I_{b(x_{i,3m})}, I_{c(x_{i,3m})}, I_{d(x_{i,3m})}\big)\\
\mathcal{L}_{\bar{x}_{i,3m}} &= \big(I_{a(x_{i,3m})}, I_{b(x_{i,3m})}, I_{c(x_{i,3m-1})}, I_{d(\bar{x}_{i,3m})}\big)
\end{aligned}$$

We have the following two observations:

- for any $x_{i,k}$ and $\bar{x}_{i,k}$, their 1st two items (say $I_{a(x_{i,k})}$ and $I_{b(x_{i,k})}$) are the same.
- the 3rd item of $x_{i,k}$ is equal to that of $\bar{x}_{i,k+1}$ (say $I_{c(x_{i,k})}$), for $k = 1, \ldots, 3m-1$, and the 3rd item of $x_{i,3m}$ is equal to that of $\bar{x}_{i,1}$ (say $I_{c(x_{i,3m})}$).

For each x_i, we construct $5 \times 3m$ dummy customers with their preference lists as follows:

$$\begin{aligned}
\mathcal{L}_{x'(1)} &= \big(I_{\alpha(x'(1))}, I_{\beta(x'(1))}, I_{\gamma(x'(1))}, I_{b(x)}\big)\\
\mathcal{L}_{x'(2)} &= \big(I_{\alpha(x'(2))}, I_{\beta(x'(2))}, I_{\gamma(x'(2))}, I_{b(x)}\big)\\
\mathcal{L}_{x'(3)} &= \big(I_{\alpha(x'(3))}, I_{\beta(x'(3))}, I_{\gamma(x'(3))}, I_{b(x)}\big)
\end{aligned}$$

and

$$\begin{aligned}
\mathcal{L}_{x''(1)} &= \big(I_{\alpha(x''(1))}, I_{\beta(x''(1))}, I_{\gamma(x''(1))}, I_{\delta(x''(1))}, I_{c(x)}\big)\\
\mathcal{L}_{x''(2)} &= \big(I_{\alpha(x''(2))}, I_{\beta(x''(2))}, I_{\gamma(x''(2))}, I_{\delta(x''(2))}, I_{c(x)}\big)
\end{aligned}$$

for $x \in \{x_{i,1}, \ldots, x_{i,3m}\}$.

For each clause $c_j = x \vee y \vee z$, let $I_{d(x)} = I_{d(y)} = I_{d(z)}$. Note that the equation means that the items are the same. All unspecified items in the above construction are different.

Assume we can allocate items to all customers according to their top two choices in 5 time steps. For any x_i and $x \in \{x_{i,1}, \ldots, x_{i,3m}\}$, consider customer $x'(1)$, $x'(2)$ and $x'(3)$, we know $I_{b(x)}$ must be allocated to them at the last three time steps. Thus, $I_{b(x)}$ should be allocated to customer x at either the 1st or 2nd time step. Similarly we know that $I_{c(x)}$ should be allocated to customer x at either 2nd or 3rd time step.

If for some customer $x_{i,k}$, we allocate $I_{a(x_{i,k})}$ to her at the 1st time step, we have to allocate $I_{b(x_{i,k})}$ and $I_{c(x_{i,k})}$ to her at the 2nd and 3rd time step, respectively. Thus, we must allocate $I_{c(x_{i,k})}$ to customer $\bar{x}_{i,k+1}$ at the 2nd time step, which implies that we allocate $I_{b(x_{i,k+1})}$ to $\bar{x}_{i,k+1}$ at the 1st time step. Therefore, we must allocate $I_{a(x_{i,k+1})}$ to $x_{i,k+1}$ at the 1st time step. By this way, we know that at the 1st time step, we allocate $I_{a(x_{i,k})}$ to $x_{i,k}$ for any $k = 1, \ldots, 3m$. Similarly, If there exists k so that we allocate $I_{b(x_{i,k})}$ to $x_{i,k}$ at the 1st time step, then this holds for all $k = 1, \ldots, 3m$. For the former case, we define x_i to be FALSE, and for the latter case, we define it to be TRUE. Thus, this gives a feasible assignments for all variables.

For each clause $c_j = x \vee y \vee z$, since all allocations can be done in 5 time steps, it implies that one of $I_{d(x)}$, $I_{d(y)}$, and $I_{d(z)}$ is allocated at the 3rd time step. Assume without loss of generality that this time step is x. For this variable, we must allocate its $I_{c(x)}$ at the 2nd time step and $I_{b(x)}$ at the 1st time step, which implies that x is assigned to be TRUE. Thus, clause c_j is satisfied.

On the other hand, assume the instance of 3SAT is satisfiable. For each x_i, we allocate items to customers as follows: If x_i is TRUE,

- at the 1st time step, allocate $I_{b(x_{i,k})}$ to customer $x_{i,k}$ and $I_{a(x_{i,k})}$ to $\bar{x}_{i,k}$, respectively, for $k = 1, \ldots, 3m$.
- at the 2nd time step, allocate $I_{c(x_{i,k})}$ to customer $x_{i,k}$ and $I_{b(x_{i,k})}$ to $\bar{x}_{i,k}$, respectively, for $k = 1, \ldots, 3m$.
- at the 3rd time step, allocate $I_{c(x_{i,k})}$ to customer $\bar{x}_{i,k}$, for $k = 1, \ldots, 3m$. For each customer $x_{i,k}$, $k = 1, \ldots, 3m$, if $I_{d(x_{i,k})}$ is available, allocate it to her, otherwise, allocate $I_{a(x_{i,k})}$ to her.
- at both 4th and 5th time step, for each customer x, if $I_{d(x)}$ is available and not assigned to her yet, allocate it to her, otherwise, allocate any other items that are on her list.

It is easy to see the above allocation is feasible. If x_i is FALSE, we can define the allocation similarly. Thus, we can allocate items to customers in 5 time step.

Note that the instance constructed above can be done easily in 6 time steps, thus the MinDynamicRentMark(2) problem is NP-hard to approximate within ratio $6/5 = 1.2$. $\square$