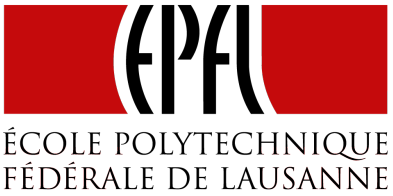

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Swiss Federal Institute of Technology
Lausanne, EPFL

Master Optional Project

Dynamic matching in crowdsourcing platform

Author:
Tasorn Sornnarong

Supervisor:
Prof.Boi Faltings
Dr.Sujit Gujar

School of Computer and Communication Sciences

July 2014

Swiss Federal Institute of Technology Lausanne, EPFL
School of Computer and Communication Sciences
Master Optional Project

Abstract

Dynamic matching in crowdsourcing platform

by Tasorn Sornnarong

We experimentally study a dynamic matching without money when one side of the market is dynamic with arrivals and departures and the other is static. Each agents have strict preferences over agents on the other side of the market. We introduce two algorithms which are not truthful for agents on the static side of the market. The two algorithms consist of two different approaches of a final matching. Firstly, we match the agents on each side of the market by applying the minimum weight bipartite matching approach. Then, run a heuristic to select a set of men that is to consider the most frequently matched with her with most preference. To obtain a rank-efficiency, the final matching is based on two different approach: man-proposing DA algorithm (ALG1) and the minimum weight bipartite matching approach based on Hungarian algorithm (ALG2). Comparing the performance of the two algorithms and the Consensus approach by evaluating the rank-efficiency, we have found that the ALG2 dominates in rank-efficiency in any cases. The best rank-efficiency is ordered by ALG2>ALG1>Consensus respectively. ALG2 improves the rank-efficiency up to 25% while ALG1 improves up to 5-7% approximately compared to Consensus. It is clearly seen that the two algorithms proposed improve the previous approaches, e.g. Consensus and GSODAS in term of rank-efficiency, which accomplish the objective of the project.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The motivation for studying dynamic matching theory is that its applications in the real world are interesting, especially the two main aspects: financial engineering and crowdsourcing platforms.

1.1.1 Financial transactions

In financial market, there are many possibilities to make a transaction or even hedging, speculating the arbitrages by using the different models for making a decision. Let us look at the game theoric perspective. Suppose that each agents in the market are financial products, e.g. coupon bonds. At a certain time, we have a preference to decide whether to buy or sell those products. The availability to take action for each products depends on its different maturity and other factors. We would like to match the financial products in each sides of the financial markets dynamically and see for each time which patterns of matching give the most efficiency and truthfulness. The idea is similar to the problem in financial engineering. Probably, we can be apply the stochastic optimization approaches in financial engineering to dynamic matching such as generating different stochastic processes with respect to time to see the movement of the products.

1.1.2 Crowdsourcing

Crowdsourcing is new paradigm introduced in last decade for getting work done in the form of an open call over internet. It is the practice of obtaining services, ideas, or

content by the contributions from a large group of people, and especially from an online community.

Crowdsourcing is distinguished from outsourcing in that the work comes from an undefined public rather than a specific group. Crowdsourcing platforms such as AMT, crowdflower enable micro tasks get done quickly. It has become very popular and it has become source of living for many workers in such crowdsourcing platforms.

However, in this marketplace, requestors, the ones who want tasks to be completed and pay workers, have preferences over workers based on its performance on particular task or over all performance, or demographics etc. Workers also prefer certain tasks more over others, are good at certain tasks even though they are not keen on those type of tasks.

Currently workers are displayed tasks based on few selected criteria such as reward, no of tasks by the requestors, most recents first etc. Here, one of the interesting question is whether the crowdsourcing platform is able to display tasks to workers more intelligently and efficiently, to cater to preferences of the requestors as well workers.

There are many challenges in real implementations. The participating workers, requestors will be typically smart and manipulate the system to report their preferences. One can leverage techniques from matching theory in this context for matching tasks and workers. However, in this settings, both sides of market are dynamic. This leads more interesting challenges.

1.2 Objectives

In this project, we intend to develop dynamic matching theory and implement the proposed algorithms practically. We need to study various matching algorithms and how they perform. An primary objective of the project that is to study and analysis of dynamic matching algorithm when one side of the market is dynamic, or to improve on existing algorithms and implement. Plus, the second goal is to design of matching algorithms for matching workers and tasks when both sides are dynamic. The algorithm should satisfy nice game theoretic properties such as stability or incentive compatibility.

Chapter 2

The state of arts of dynamic matching

There are many studies about the matching problem in static settings. In this system in static settings, we observe the agents and their preferences. The objective of the problem is to find a good matching algorithm with desirable properties; e.g. stability, rank-efficiency or strategyproofness [1]. Also, there has been some studies of matching algorithm in dynamic settings. That is to find good algorithms for matching markets that evolve over time with an uncertain future. In this chapter, we introduce the fundamental studies for both static and dynamic matching.

2.1 Review of Deferred Acceptance (DA) algorithm or Gale-Shapley algorithm

The example of matching in static setting is the stable marriage problem introduced by Gale and Shapley[2][3]. From now on we mainly focus on the marriage problem, which is a very practical and simple problem in the real world. The matching problem has been solved by choosing which agents match each other. In contrast, the change of time provides more possibilities for transactions in a dynamic setting.

Gale and Shapley proved that every instance of the stable marriage problem admits at least one stable matching. They describe an algorithm that is guaranteed to find such a matching. Furthermore, they have found that their algorithm gives a stable matching with an interesting property providing all men or all women the best partner that is possible in any stable matching. The Gale and Shapley algorithm is expressed as a sequence of proposals from men to women in male proposing case. During the

algorithm's implementation, each person is either engaged or free. Once a woman is committed, she is not free anymore. A man can be engaged more than once. It obtains fiancees who are successively less desirable to him, while each successive engagement leads a woman more preferable. When a free woman has a proposal for a proposer, she will immediately accept it and become engaged. When an engaged woman has a proposal again, she makes a comparison between the proposer and her current fiancf one of them is less preferable, she rejects. Each man proposes to the women on his preference list until he becomes engaged. If the engagement is broken, then he becomes free again, and he resumes his sequence of proposals, starting with the next woman on his list. The algorithm terminates when everyone is engaged and no rejections. The Gale and Shapley algorithm is stated in figure 2.1 [3].

```
assign each person to be free
while some man m is free do
begin
        w=first woman on m's list to whom m has not yet proposed
        if w is free then
                assign m and w to be engaged (to each other)
        else
                if w prefers m to her fiance m then
                        assign m and w to be engaged and m' to be free
                else
                        w rejects m {and m remains free}
        end
output the stable matching consisting of the n engaged pairs
```

FIGURE 2.1: Basic Gale-Shapley algorithm

2.2 Review of minimum weight bipartite matching

Finding the minimum weight bipartite matching is one of the fundamental problems in combinatorial optimization. In weighted bipartite graphs, each edge has an associated value. A minimum weighted bipartite matching is defined as a matching where the sum of the values of the edges in the matching have a minimum value[4]. If the graph is not complete bipartite, missing edges are inserted with value zero. Finding such a matching is known as the assignment problem. We think that it can be applied for designing the matching algorithms to assign tasks to workers when both sides are dynamic, which is one of the objective of the project. In the project, we run the Hungarian algorithm developed by Harold Kuhn [5]. The original Hungarian algorithm solves the assignment

problem in polynomial time using a modified shortest path search in the augmenting path algorithm.

Here, we skip explaining the mathematical definition of Hungarian algorithm. We would rather explain the Hungarian algorithm by giving matrix interpretation [4]. To be more intuitive, we consider a marriage problem instead of explaining the original Hungarian algorithm which is to assign each worker to a task. We use the same setting as the marriage problem in the previous part. We denote M the number of women and N the number of men, and an M matrix containing the weight of woman to men, then find the minimum weight.

The expression of the matrix of the weight of each edges is given by

$$\begin{bmatrix} w(w_1,m_1) & w(w_1,m_2) & \cdots & w(w_1,m_N) \\ w(w_2,m_1) & \ddots & \cdots & w(w_2,m_N) \\ \vdots & \vdots & \ddots & w(w_1,m_N) \\ w(w_M,m_1) & w(w_M,m_2) & \cdots & w(w_M,m_N) \end{bmatrix}$$

where $w_1, w_1, ...w_M$ are the women who are getting proposed by the men $m_1, m_2, ..., m_N$. with the weight $w(w_i, m_j)$ between i^{th} woman and j^{th} man. For simplicity, let the size of matrix be $4 \times 4 (M = 4, N = 4)$.

Step 1: Perform row operations on the matrix. To do this, the lowest of all element is taken and subtracted from each element in that row. This procedure is done repeatedly for all rows. We now have a matrix with at least one zero per row. Now, we try to propose men to women such that each woman is proposed by only one man. The corresponding result is given by,

$$\begin{bmatrix} 0' & w(w_1,m_2)' & 0' & w(w_1,m_4)' \\ w(w_2,m_1)' & w(w_2,m_2)' & w(w_2,m_3)' & 0' \\ 0' & w(w_3,m_2)' & w(w_3,m_3)' & w(w_3,m_4) \\ w(w_4,m_1)' & 0' & w(w_4,m_3)' & w(w_4,m_4)' \end{bmatrix}$$

The zeros that are indicated as 0' are the proposed women.

Step 2: Sometimes it is possible that the matrix at this stage cannot be used for proposing, as is the case in for the matrix below.

$$\begin{bmatrix} 0 & w(w_1,m_2)' & w(w_1,m_3)' & w(w_1,m_4)' \\ w(w_2,m_1)' & w(w_2,m_2)' & w(w_2,m_3)' & 0' \\ 0 & w(w_3,m_2)' & w(w_3,m_3)' & w(w_3,m_4) \\ w(w_4,m_1)' & 0' & w(w_4,m_3)' & w(w_4,m_4)' \end{bmatrix}$$

In the above case, no proposing can be done. Note that man m_1 proposes efficiently by both woman w_1 and w_3. Both cannot be proposed by the same man. Also note that no woman is proposed by man m_3 efficiently. To solve this problem, we run the above procedure for all columns repeatedly and then verify whether the proposing is possible.

Step 3: It will give the result in most situations, but if it is still impossible to be proposed then all zeros in the matrix must be covered by marking as few rows and/or columns as possible. The following procedure is one way to accomplish this:

Initially being proposed as many men as possible then do the following (men propose in rows 2, 3 and 4)

$$\begin{bmatrix} 0 & w(w_1,m_2)' & w(w_1,m_3)' & w(w_1,m_4)' \\ w(w_2,m_1)' & w(w_2,m_2)' & w(w_2,m_3)' & 0' \\ 0' & w(w_3,m_2)' & w(w_3,m_3)' & w(w_3,m_4) \\ w(w_4,m_1)' & 0' & 0 & w(w_4,m_4)' \end{bmatrix}$$

Mark all rows having no proposing (row 1). Then mark all columns having zeros in marked rows (column 1). Then mark all rows having proposing in marked columns (row 3). Repeat this until we obtain a closed loop.

$$\begin{bmatrix} \times & - & - & - & - \\ 0 & w(w_1,m_2)' & w(w_1,m_3)' & w(w_1,m_4)' & \times \\ w(w_2,m_1)' & w(w_2,m_2)' & w(w_2,m_3)' & 0' & - \\ 0' & w(w_3,m_2)' & w(w_3,m_3)' & w(w_3,m_4) & \times \\ w(w_4,m_1)' & 0' & 0 & w(w_4,m_4)' & - \end{bmatrix}$$

Step 4: From the elements that are left, find the lowest value. Subtract this from every unmarked element and add it to every element covered by two lines.

Repeat steps 3 and 4 until it is possible to be proposed; this is when the minimum number of lines used to cover all the 0's is equal to the min(number of women, number of men), assuming the minimum weights are used to fill in when the number of women is greater than the number of men.

Basically we can find the second minimum weight among the two rows. The procedure is repeated until we can distinguish among the women in terms of minimum weight.

2.3 Studies of dynamic matching

2.3.1 GSODAS

The GSODAS (Generalized Online Deferred Acceptance with Substitutes) is introduced by Sujit Gujar and David C. Parkes [6]. GSODAS is both truthful and stable for static agents as there are no man or woman pair would prefer to deviate from their successive matches and rematch among themselves. Such a man-woman pair is a blocking pair. The blocking pairs consist of a pair between man and woman where the woman has matched with a fall-back option and insists that for any such pair, that the woman would prefer the fall-back option or the man prefers his match. The number of fall-back options required by GSODAS is worst-case optimal across online mechanisms that provide stability. They compare the match quality from GSODAS with two randomized, truthful matching mechanisms that operate without using a fall-back option. For seeing the quality of matching, they take into account both the stability by computing the average number of men which are in at least one blocking pair, and the rank-efficiency. The rank efficiency interprets the rank-order for a woman matched with a substitute as equivalent to that for the man with which she was first matched, but ignore the substitute himself in determining rank efficiency.

2.3.2 CONCENSUS

For a rank-efficiency baseline, we also consider the performance of a non-truthful algorithm called Consensus. The algorithm is proposed by Van Hentenryck and Bent [7]. It adopts an online stochastic optimization in dynamic setting and provides a strong baseline target for rank-efficiency[6]. In simulation, Consensus performs better than GSODAS, but it has poor many blocking pairs leading to the instability.

The Consensus approach adopts a generative model of the future by generating large number of random future instances with the arrivals of agents on the dynamic side of the market. Then use these samples to evaluate the decision of matching for each agents. Let us consider for any period when at least one woman departs, Consensus generate an instance of arrival and departure of the remaining women. The final decision is done by checking which agent is most frequently matched as a result of running man-proposing DA algorithm.

Chapter 3

Experimental studies

3.1 Preliminaries

We consider a marriage problem using the same setting and definition as the one proposed by Sujit Gujar and David C. Parkes [6]. Let us consider the universe of the market consists of n men (set M) and n women (set W). The men are static and the women are dynamic.

$$\{m_1, m_2, m_3, ..., m_n\} \in M \tag{3.1}$$

$$\{w_1, w_2, w_3, ..., w_n\} \in W \tag{3.2}$$

and denote a_i is the arrival and d_i is the departure of woman $i \in W$. $a_i, d_i \in \{1, ..., T\}$ where T is the time periods. We suppose that each agents prefers to be matched than unmatched. Each agents in each side of the market has a preference $\succ_i$ on agents on the other side of the market. Denote $w_1 \succ_m w_2$ a strict preference by man m for woman w_1 over woman w_2. Similarly, we write $m_1 \succ_w m_2$ to denote a preference by woman w for man m_1 over m_2. A match to a man can be made in any of the T periods, and preferences may be determined dynamically as women arrive as long as the preference rank on earlier arrivals is unchanged. For a woman, a match must be made between a_i and d_i and preferences must be well defined upon arrival. Let $M(t)$ and $W(t)$ respectively denote the set of men and women available for matching in period t. Let $AW(t)$ denote the set of women to arrive in t, $DW(t)$ the set of women to depart in t, and W'(t) the set of women yet to arrive. Let μ denote a match, with $\mu(m) \in W \cup \{\phi\}$ the match to man m and $\mu(w) \in M \cup \{\phi\}$ the match to woman w, with $\mu(i) = \phi$ to indicate that agent i is unmatched. A woman is available for matching while present, and a match $\mu(w) \neq \phi$ to a woman must be finalized by period d_i. Upon the departure of woman

w with $\mu(w) \neq \phi$, then the matched man $\mu(w) \in M$ ordinarily becomes unavailable for matching and $M(t)$ is updated.

To assess the performance of the algorithm, we use the evaluation given by Budish and Vantillon [2]. The parameter for the evaluation is called "rank-efficiency". For a matching μ, the rank of an agent i can be expressed as $rank_i(\mu)$. It is the rank order of the agent with whom he or she is matched. A match by i with the most-preferred agent in $\succ_i$ receives rank order 1 and with the least-preferred receives rank order n. Let the rank-order be $n+1$ if it is unmatched $\mu(i) = \phi$. Based on this setting, the rank of a matching is $rank(\mu) = \frac{1}{2n} \sum_{i \in M \cup W}^{2n} rank_i(\mu)$ We assume that a distribution function Φ on $(\succ, \rho)$ and compute the expected rank over the distribution to give a mathematical definition of the rank-efficiency of an online mechanism f. The expression is the following:

$$rank^f = E_{(\succ,\rho)\ \Phi}[\text{rank}(f(\succ, \rho))] \tag{3.3}$$

For simplicity of explaining the algorithms in the latter parts, we denote $DA(M, W)$ as male proposing DA or Gale-Shapley algorithm with set of men M and set of women W.

In order to run the Hungarian algorithm to find bipartite matching with the minimum weight approach, we need to know the assignment matrix, namely the matrix expressing the weight of each edges of the bipartite graph. Recall the chapter 2, this matrix can be expressed by

$$\begin{bmatrix} w(w_1, m_1) & w(w_1, m_2) & \cdots & w(w_1, m_n) \\ w(w_2, m_1) & \ddots & \cdots & w(w_2, m_n) \\ \vdots & \vdots & \ddots & w(w_1, m_n) \\ w(w_n, m_1) & w(w_n, m_2) & \cdots & w(w_n, m_n) \end{bmatrix}$$

where $w_1, w_1, ...w_n$ are the women who are getting proposed by the men $m_1, m_2, ..., m_n$.

Generally, the best matching is such that both man and woman have least rank-efficiency on each other. The more weight is, the less rank-efficiency should be. Apart from the rank-efficiency, we should take the number of the agents on each side of the market into account of the weight.

To get a satisfactory result by adopting the minimum weight bipartite matching based on Hungarian algorithm, we define the weight $w(w_i, m_j)$ between i^{th} woman and j^{th} man is given by the following:

$$w(w_i, m_j) = rank_{w_i}(\mu) + rank_{m_j}(\mu) \tag{3.4}$$

3.2 Comparison between DA algorithm and Hungarian algorithm in term of rank-efficiency

To see the comparison more obviously, let a set of men $M = \{m_1, m_2, m_3\}$ and a set of women $W = \{w_1, w_2, w_3\}$. Suppose the preferences of men and women on the another side are the followings:

$$m_1 : w_1 \succ_{m_1} w_2 \succ_{m_1} w_3 \tag{3.5}$$

$$m_2 : w_1 \succ_{m_2} w_3 \succ_{m_2} w_2 \tag{3.6}$$

$$m_3 : w_1 \succ_{m_3} w_3 \succ_{m_3} w_2 \tag{3.7}$$

$$w_1 : m_1 \succ_{w_1} m_2 \succ_{w_1} m_3 \tag{3.8}$$

$$w_2 : m_1 \succ_{w_2:} m_2 \succ_{w_2:} m_3 \tag{3.9}$$

$$w_3 : m_1 \succ_{w_3} m_2 \succ_{w_3} m_3 \tag{3.10}$$

Denote $rank_{m_i,w_j}(\mu)$ is the sum of the rank-efficiency of m_i man and woman w_j. For the DA algorithm, If the agents are truthful, the mechanism will match m_1 with w_1 with $rank_{m_1,w_1}(\mu) = 2$, m_2 with w_3 with $rank_{m_2,w_3}(\mu) = 4$ and m_3 with w_2 with $rank_{m_3,w_2}(\mu) = 6$. The total rank-efficiency is 12.

For Hungarian algorithm, if the agents are truthful, the mechanism will match m_1 with w_2 with $rank_{m_1,w_2}(\mu) = 3$, m_2 with w_1 with $rank_{m_2,w_1}(\mu) = 3$, and m_3 with w_3 with $rank_{m_3,w_3}(\mu) = 5$. The total rank-efficiency is 11.

It is obvious to see that the Hungarian algorithm gives a better rank-efficiency than DA algorithm.

3.3 Stochastic Optimazation

We use MATLAB for coding the algorithm consisting of two main parts: testing the performance and run each algorithms separately. In this project, we initiate two algorithms which are not truthful for agents on the static side of the market. The two algorithms consist of two different approaches.

To obtain a baseline performance for rank-efficiency, we apply the online stochastic optimization algorithm, based on the Consensus approach of Van Hentenryck and Bent [7] as it provides a good rank-efficiency.

For all algorithms, we adopt a generative model of the future by generating large number of random future samples with the arrivals of agents on the dynamic side of the market.

Then we use these samples to evaluate the decision of matching for each agents. Now, we know the arrival, the departure and the preference of agents. Whenever, there is a call to sample a future, generates an instance of arrival-departure schedule along with preferences for the remaining women. Best possible decision pertaining to current period is taken for each sample generated. Then based on the frequencies of various current possible decision, the final current period decision is made. In regards to our matching problem, in every period in which at least one woman departs, The samples multiple possible future arrivals and matches each departing woman with which she is frequently matched with first preference when running two different approaches on each sample. We divide the code into two main parts: testing of the performance and algorithm part.

The test of the performance:
(i) generate the arrival, the departure on the dynamic side of the market.
(ii) assign the preference of agents on both sides of the market by generating random permutation.
(i) generate large K_1 samples to find matching and the relevant parameters, e.g. rank-efficiency for each algorithms and each K_1.
(ii) compute the average case of the rank-efficiency for time t and each side of the market has n number of agents.

The algorithm part:
Check whether there is at least one woman departs for any period t, if yes,
(i) generate K_2 samples of the preferences for $n-l$ women, where l women have already arrived. In other word, the preference list for the women who have already arrived is unchanged.
(ii) for each sample W_k, for $k = 1, ..., K_2$, run bipartite matching $bipartite(M(t), W(t) \cup W_k)$. Note that we use the corresponding rank order of both woman and man based on its preference list. Compute the weight of each edge and construct the assignment matrix A. Then, plug it into the Hungarian algorithm function in MATLAB publicly provided by Alexander Melin [8]. Note that in coding Hungarian algorithm function gives a minimum weight of bipartite matching, so the matrix we have to find is $Hungarian(A)$.
(iii) select the set of men to be available for matching with the following heuristic.

Heuristic 1: For each woman $w \in W(t)$, find $L_1(w)$ denoting the man most frequently matched with her in the result of running minimum weight bipartite algorithm on each of the K_2 samples.

(iv) Run two different approaches on the set of women, W(t), and men in the set $\{L_1(w)|w \in W(t)\}$. Commit the matches in this two approaches that involve departing women, updating M(t) accordingly.

Approach 1: man-proposing DA
Approach 2: minimum weight bipartite algorithm (Hungarian algorithm)

(v) Reset the set of men and women available to be zero. Consider next period of time and repeat the procedures from (i) again until reaching time T.

We can rewrite the algorithms used in the project as followings:

Algorithm Consensus: Run Consensus approach
Algorithm1(ALG1): Run man-proposing DA with heuristic 1
Algorithm2(ALG2): Run Hungarian algorithm with heuristic 1

3.4 Experimental results

For simplicity, we suppose that the number of men and women are equal. We compare the rank-efficiency and stability of two algorithms. In all simulations, we generate preference profiles uniformly at random for all men and women, and also the arrival time a_i of a woman uniformly between [1, T], and the departure time d_i uniformly between [a_i, T] by using function randi(). In both cases, we investigate the average case performance over $K_1 = 1000$ random instances. To compare the results, we consider a problem with two different horizons. We find the rank-efficiency for each algorithm fixing the number of samples $K_2 = 10$ when increasing the number of agents on each side of the market N from 4 to 20 increased by 2 (4,6,...,20), and the number of periods T from 4 to 20 increased by 2 (4,6,...,20).

The results are illustrated in figures 3.1, 3.2, 3.3 and 3.4. Figures 3.1 and 3.2 show the average rank-efficiency when fixing time T=10 and 20 respectively and increasing the number of men or women N from 4 to 20. Figures 3.3 and 3.4 show the average rank-efficiency when fixing the number of men or women $N = 10$ and 20 and increasing time T from 4 to 20. The results are again averaged over 1000 instances. Each algorithm is not strategyproof, and that rank-efficiency assigns a rank order of n+1 to unmatched agents. Note that the least number of rank-efficiency (e.g. 1) gives the best rank-efficiency. It is obvious to see that the ALG2 dominates in rank-efficiency in any cases. The best rank-efficiency is ordered by ALG2>ALG1>CONSENSUS respectively. According to figures 3.1 and 3.2, there is not that change in term of the sharp and the rank-efficiency between fixed T=10 and T=20. One of the characteristics is that if increasing N, the rank-efficiency gets worse (the number of rank-efficiency increases) and the ALG2 improves better. Concretely, for less N, ALG1 and ALG2 have almost no improvement but if considering until $N = 20$, ALG2 improves up to 25% while ALG1 improves about 5%

compared to Concensus. In contrast, there is quite a difference between fixing $N = 10$ and $N = 20$ against the time period T changing from 4 to 20. According to figure 3.3, it is obvious that ALG2 dominates all and ALG1 dominates Consensus. ALG2 improves the rank-efficiency up to 15% and ALG1 improves up to 5% approximately compared to Concensus. However, in figure 3.4 for $N = 20$, the rank-efficiency gets worse for all algorithm, but the improvement of rank-efficiency is better. ALG2 still gives the best rank-efficiency that improves Concensus up to 25% approximately, while the improvement of ALG1 to Concensus is lower, about 7%. The shape of the rank-efficiency tends to be constant over time based on the assumption that if we run more instances, e.g. $K_1 = 10000$ instances, we will get the constant rank-efficiency.

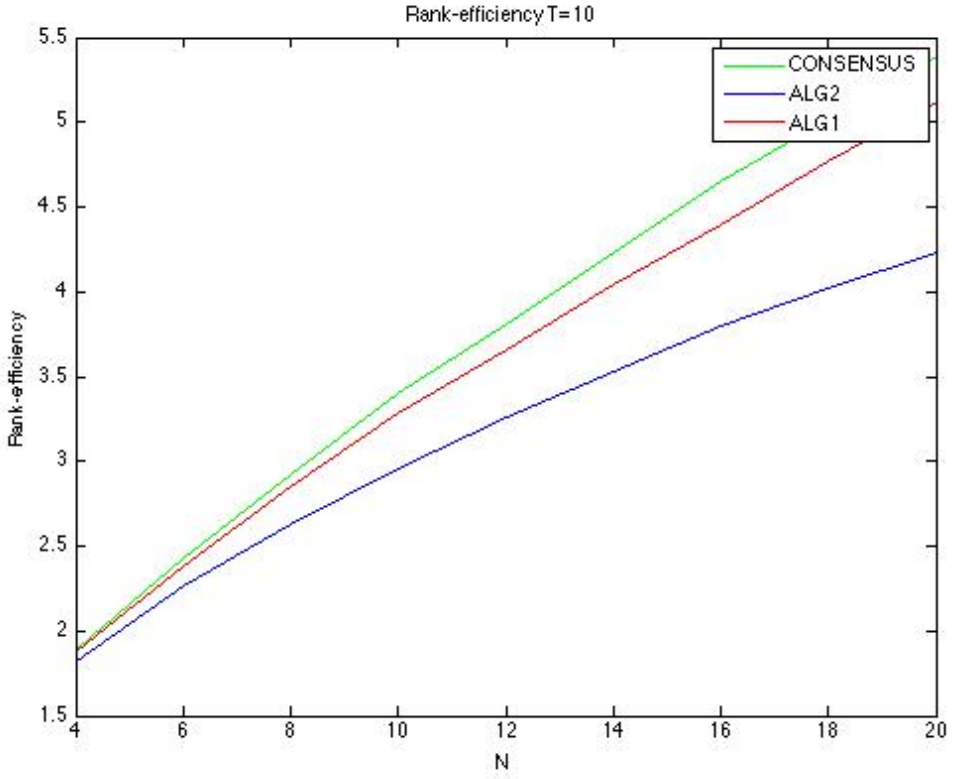

FIGURE 3.1: The rank-efficiency given T=10

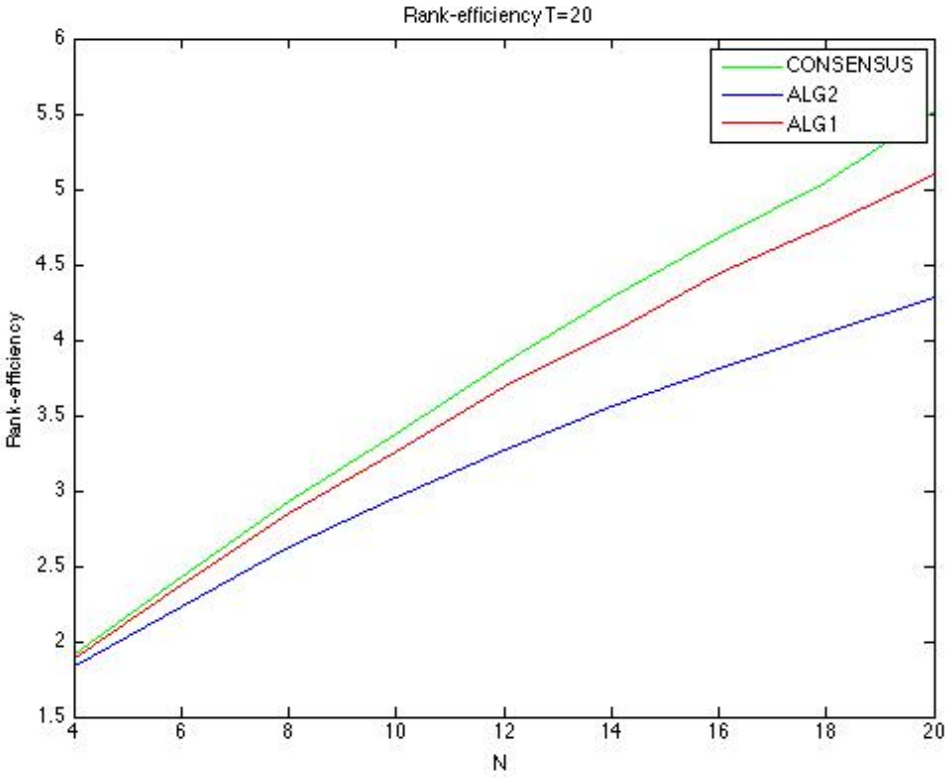

FIGURE 3.2: The rank-efficiency given T=20

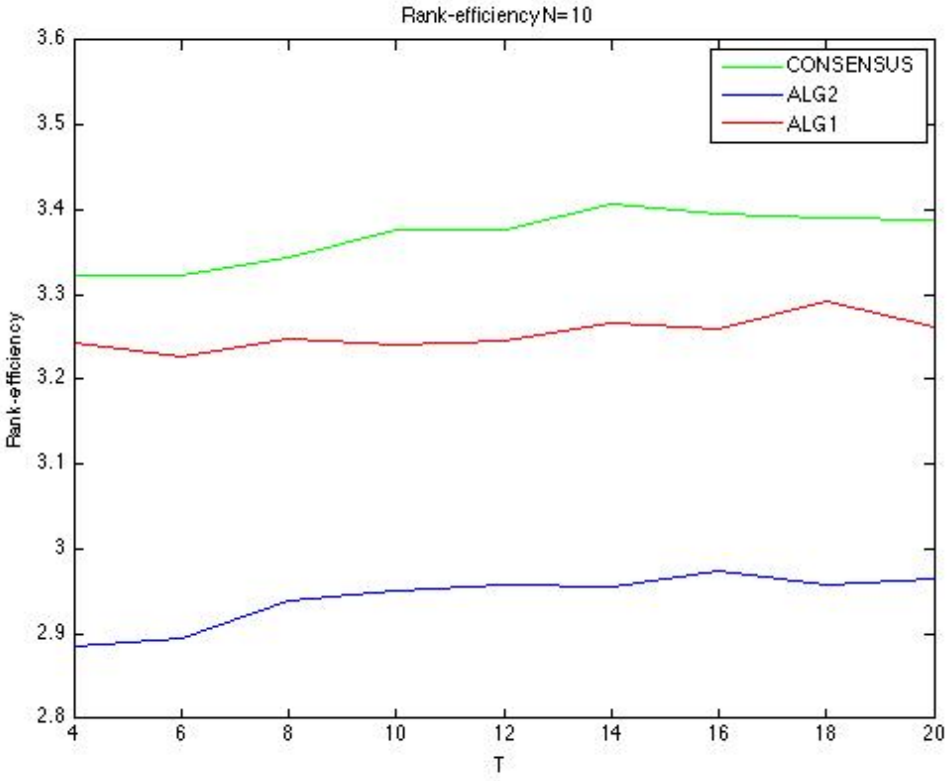

FIGURE 3.3: The rank-efficiency given N=10

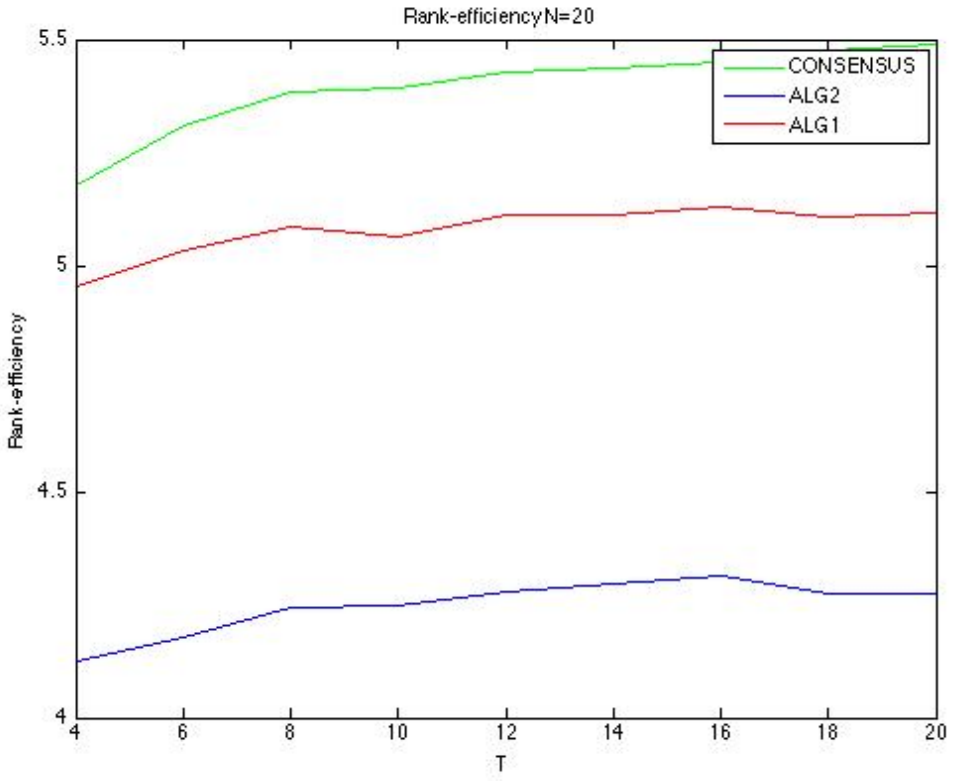

FIGURE 3.4: The rank-efficiency given N=20

Chapter 4

Conclusion

In this project, we have experimentally studied into dynamic matching in two-sided markets without money. One side of the market is dynamic with arrivals and departures and the other is static. For simplicity, we consider and set the problem as marriage problem. We have discovered that the minimum weight bipartite matching gives better rank-efficiency than the man-proposing DA algorithm based on Gale-Shapley. Then, we prove this by doing a simulation and obtain the same result. We accomplish improving the two algorithms that give a better rank-efficiency than Consensus, GSODAS mechanism or simpler methods. The more challenging future work is also to evaluate the stability and strategyproofness of the proposed two algorithms. To do this, we may possibly run a different heuristic of each mechanism to choose a set of men such that the woman w is matched with the man m minimum number of times in the result of running minimum weight bipartite algorithm on each instances. The cardinality of unmatched women would imply the instability of matching.

Chapter 5

References

[1] Mohammad Akbarpour et al. (2014), "Dynamic Matching Market Design", Cornell University.

[2] D. Gale and L. S. Shapley (1962), "College admissions and the stability of marriage", The American Mathematical Monthly, 69(1).

[3] Dan Gusfield and Robert W. Irving (1989), "The stable Marriage Problem: Structure and Algorithms", Massachusetts Institute of Technology.

[4] R.E. Burkard, M. Dell'Amico, S. Martello (2012), "Assignment Problems (Revised reprint)". SIAM, Philadelphia.

[5] Harold W. Kuhn (1955), "The Hungarian Method for the assignment problem", Naval Research Logistics Quarterly, Kuhn's original publication.

[6] Sujit Gujar and David C. Parkes (2010). "Dynamic matching with a fall-back option". In Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence.

[7] P. Van Hentenryck and R. Bent (2006), "Online Stochastic Combinatorial Optimization", MIT Press.

[8] Alexander Melin (2006), "MATLAB code for Hungarian algorithm", URL: "http://www.mathworks.com/matlabcentral/fileexchange/11609-hungarian-algorithm" available on 30 May 2014.