Stable Matching under Dynamic Preference

(動的選好順序の下での安定マッチング)

September, 2022

Doctor of Philosophy (Engineering)

Akhmad Alimudin
アフマド アリムディン

Toyohashi University of Technology

[This page is intentionally left blank]

Acknowledgements

I want to express my gratitude to my supervisors and examination committee members, Professor Yoshiteru Ishida, Professor Koutarou Suzuki, Professor Jun Miura, and Associate Professor Ren Ohmura, who guided me to this step.

Special thanks to Toyohashi University of Technology and MEXT Japan for their financial support throughout my study.

I would also like to thank my friends and family, who supported me and offered deep insight into the study.

Akhmad Alimudin, September 2022

[This page is intentionally left blank]

Abstract

We studied two-sided matching with dynamic preference. One of the most well-known problems in two-sided matching is the Stable Marriage Problem (SMP). An instance of SMP is $I = (M, W, L)$, where M and W denote a set of men and women agents, respectively, and L is the preference list of each agent in M X W. In classical SMP, a matching is said to be stable if no blocking pair is found. A blocking pair is formed of a man m and a woman w who are not partners in a matching M but prefer each other over their current partners. The stable matching problem with dynamic preference is near to the real-world situation problem, where an agent is allowed to change his/her preference at any time, affecting the stability of a matching. In this study, we propose two strategies to maintain the stability of a matching problem with dynamic preference, namely short-term stability and long-term stability.

Short-term stability is a strategy to maintain the stability of matching by updating the matching every time the preferences change. A traditional way that can be done on the short-term stability strategy is to start the matching process using the Gale-Shapley algorithm from scratch. However, the preference changes in the matching problem do not always affect the existing stable matching. Sometimes the preference changes do not trigger a forming of a blocking pair in obtained matching, which means the prior matching is stable to the new preference. Roth and Vande [25] introduce the mechanism to update the matching by satisfying the blocking pair. However, this mechanism always starts the process with one pair (initial matching) and then satisfies each agent until stable matching is found in the market. To address this issue,

we intend to maximize the initial matching, by identifying the prior stable matching against the new preference. By identifying the prior matching, we can maximize the number of initial matching of the updating process. Compared to other existing methods, our proposed concept is outperform in minimizing re-matching costs.

The second strategy to find a stable matching with dynamic preference is long-term stability, which is to maintain the stability of matching over a long period. This strategy is used when the frequency of preference changes in agents occurs frequently. Employing a short-term strategy to maintain stability would be costly when preference changes occur frequently. A classical SMP instance is $I = (M, W, L)$, in the SMP with dynamic preference, agents can change their preferences, leading to dynamic preference. An instance of SMP with dynamic preference leading the formation of a dynamic instance. The dynamic instance is $DI = (M, W, L_1, L_2, \ldots, L_k)$, where k is the number of unique preference lists that occur due to changes in agent preferences. Thus, a set of SMP instances for SMP under dynamic preference is $DI = \{I_1, I_2, \ldots, I_k\}$. Several studies used the *most stable matching* concept to obtain long-term stable matching. This concept counts the number of stability of matching against all occurred instances. They use the α as the index of strengths of provided solutions. The matching with the highest index is selected as a solution for stable matching. However, in some cases, some matching has an equal value of the index, this situation is difficult to determine the solution. In this study, we introduce a new concept to find stable matching under dynamic preference using the blocking pair perspective. Using the blocking pair, we can get the new information to select the stable matching more precisely. Considering the number of blocking pairs in the stable matching under dynamic preference is a novel concept. By understanding the amount of blocking pairs in detail, it is possible to obtain additional information regarding stable matching. Consequently, the information obtained from the number of blocking pairs can assist in determining stable matching with greater precision than the

existing method, In other words, the usage of the index (α, β, and EV) is better than α. We also define the new notions of stability for a matching problem under dynamic preference.

To demonstrate the relevance of our findings, we apply stable matching under dynamic preference to the scheduling problem of the data center. The objective of adopting stable matching in the scheduling problem is to achieve agent satisfaction while maintaining data center energy efficiency. A stable matching is needed for the scheduling job to prevent the agents from complaining about their pair in the matching. Hence, we can minimize the expenses associated with re-matching, such as migration, reconfiguration, and downtime, when the market decides to change the matching.

[This page is intentionally left blank]

Table of contents

[This page is intentionally left blank]

List of figures

List of tables

List of Algorithms

[This page is intentionally left blank]

Chapter 1

Introduction

1.1 Background

Numerous types of research have been conducted on the stable matching problem in fields including computer science, mathematics, and economics. The term "matching" refers to a collection of agents wishing to form a pair that meets each agent's criteria. The stable marriage problem (SMP) was introduced by Gale and Shapley [14]. It is one of the most well-known stable matching problems. Since its original conception in 1962, the SMP has attracted significant attention from researchers. Numerous extended variants of SMP have also emerged, such as the stable roommate problem, the college admissions problem, the hospital/resident problem, and several other stable matching problems [16, 15, 23, 19]. The SMP algorithm has been widely used to solve several real-world problems. One of the most widely used variants is the hospital/resident problem variant [23, 18], which is used to place medical students in hospitals or to select new students. Nowadays, the SMP algorithm is also widely applied to large-scale computer applications, such as content delivery networks [12] and job scheduling of virtual machines to servers [1, 2].

The SMP is a bipartite matching problem with an equal number of agents on each side. Each agent expresses a strict order preference that includes all

members of the opposite side. A matching μ is unstable when at least one blocking pair exists. A blocking pair is formed of a man m and a woman w who are not partners in a matching μ, but prefer each other over their current partners. In the classical SMP, each agent expresses a strict order preference for the opposite side. However, in real-world situations, some agents occasionally cannot express their actual preference list due to a lack of information or observations about the opposing side, leading to the agents' preferences changing dynamically. This study aims to find a stable matching for the SMP with the dynamic preference model, where dynamic preferences refer to a scenario in which agents' preferences might change dynamically over time.

Definition 1.1. *A dynamic preference is a form of preference in a matching problem in which the agent can express two or more different preferences.*

Definition 1.2. *A dynamic instance is a group of stable matching instances generated by dynamic preferences.*

To illustrate the problem, we use a simple 3 × 3 SMP.

Instance 1 **(I_1)**

$L_1(m_1) = w_1, w_2, w_3$ $\quad$ $L_1(w_1) = m_2, m_3, m_1$

$L_1(m_2) = w_2, w_3, w_1$ $\quad$ $L_1(w_2) = m_3, m_1, m_2$

$L_1(m_3) = w_3, w_1, w_2$ $\quad$ $L_1(w_3) = m_1, m_2, m_3$

Instance 2 **(I_2)**

$L_2(m_1) = w_3, w_2, w_1$ $\quad$ $L_2(w_1) = m_2, m_3, m_1$

$L_2(m_2) = w_2, w_3, w_1$ $\quad$ $L_2(w_2) = m_3, m_1, m_2$

$L_2(m_3) = w_3, w_1, w_2$ $\quad$ $L_2(w_3) = m_1, m_2, m_3$

Fig. 1.1 The 3 × 3 dynamic preference of SMP.

Example 1.1. *There is a set of men $M = \{m_1, m_2, m_3\}$ and a set of women $W = \{w_1, w_2, w_3\}$. Assume that agent m_1 has dynamic preferences by changing his preference order. Specifically, these dynamic preferences generate two SMP instances, as depicted in Figure 1.1:*

Assume instance I_1 admits matching μ_1 as a stable matching. However, μ_1 does not necessarily stable against I_2. We need further checks to ensure that μ_1 is stable against I_2. If μ_1 does not admit stable in I_2, we need to perform new matching and generate new stable matching in instance I_2.

This study considers two scenarios when preference changes might occur. In the first scenario, we assume that preference changes infrequently. Therefore, we propose a matching updating mechanism to maintain stability whenever the preference changes. We propose a mechanism to find a new stable matching for the latest preference by identifying the previous instance. In the second scenario, we assume that preference changes frequently occur. In this scenario, employing the updating mechanism would be costly. When the preference frequently changes, there will be a lot of costs that need to spend repeatedly. We propose a new concept for the second scenario to find a stable matching by considering multiple instances. We categorize our strategies into two, namely the short-term and the long-term stability strategy.

Definition 1.3. *Short-term stability is a strategy to maintain the stability of a matching every time the preference changes. This strategy requires a matching stable to the latest preference and generates new stable matching if needed.*

Definition 1.4. *Long-term stability is a strategy to maintain the stability of a matching based on multiple preferences. This strategy does not require generating new stable matching even if the preference frequently changes.*

Figures 1.2 and 1.3 illustrate the changes in preferences over the same time period. In figure 1.2, there were three changes in preference, whereas, in figure 1.3, there were six changes in preference. The illustrations enable us to determine the method for finding stable matching.

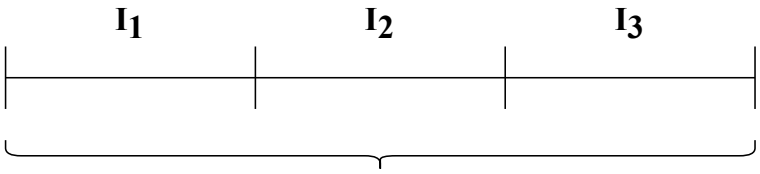

Fig. 1.2 Preference rarely changes

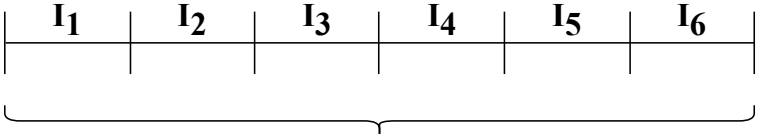

Fig. 1.3 Preference frequently changes

As shown in Table 1.1, we compare the advantage and disadvantage of each strategy. Short-term strategies produce stable matching, whereas long-term strategies produce near-stable matching. The short-term strategy always performs stable matching computations whenever the preference change, there will be significant costs associated with the computational cost, and also the cost of re-matching or reconfiguring the market. On the other hand, a long-term strategy requires only single stable matching computation and avoids the re-matching costs.

Table 1.1 Comparison of short-term and long-term stability strategy

Strategy	Advantage	Disadvantage
Short-term	Produce stable matching	Costly (re-matching cost)
Long-term	Produce near stable matching	Avoid re-matching cost

Re-matching or reconfiguration costs refer to the expenses associated with changing or modifying an existing system or process to meet new requirements or goals. These costs can include expenses for planning, design, implementation, testing, training, and any other activities required to make the necessary changes. Re-matching costs can vary depending on the complexity of the system or process being changed, the extent of the modifications required, and the resources needed to complete the re-matching. In some cases, re-matching costs can be quite high, particularly if significant changes are required or if

the system or process is critical to the market. However, re-matching costs are important in order to adapt to changing circumstances or to improve the stability in the market.

A classical SMP instance is $I = (M, W, L)$. In the SMP with dynamic preference, agents can change their preferences, leading to dynamic preference. An instance of SMP with dynamic preference leading the formation of a dynamic instance. The dynamic instance is $DI = (M, W, L_1, L_2, \ldots, L_k)$, where k is the number of unique preference lists that occur due to changes in agent preferences. Thus, a set of SMP instances for SMP under dynamic preference is $DI = \{I_1, I_2, \ldots, I_k\}$.

In a classical SMP, the stability of matching is determined by the existence of a blocking pair. In a matching μ, a pair (m, w) is said to be a blocking pair if m and w are not a pair in μ, but m prefers w over $\mu(m)$ and w prefers m over $\mu(w)$. That is, $(w \succ_m \mu(m)) \wedge (m \succ_w \mu(w))$. A matching that contains at least one blocking pair is called unstable; otherwise, it is stable. However, the stability concept of classical SMP cannot be used in our problem.

We also define the new concept of stability for the long-term strategy because the original concept of matching stability is difficult to apply in the stable matching problem under dynamic preference.

Definition 1.5. *Fully stable is a criterion for a matching that admits stability to the dynamic instance. A matching μ is stable to $\forall I \in DI$.*

Definition 1.6. *α-most stable is a criterion for a matching that admits stability at least in one instance, such that a matching μ is stable to $\exists I \in DI$. α indicates the strength of matching to all possible instances, where $1 \leq \alpha \leq k$. If $\alpha = k$, it means that the matching μ admits stable in all instances, equals to a fully stable.*

Definition 1.7. *β-Least BP is a criterion for a matching that admits minimum blocking pairs in a dynamic instance. β indicates the number of blocking pairs*

for a matching μ in a dynamic instance. If $\beta = 0$, it means that the matching μ admits stable in all instances, equals to a fully stable and $k - moststable$.

Definition 1.8. *EV-Least BP is a criterion for a matching that admits minimum blocking pairs in a dynamic instance. This notion considers the probability of an instance occurring. EV indicates the expected value of blocking pairs for a matching μ in a dynamic instance. Matching with the minimum EV is the solution of stable matching under dynamic preference*

Several studies related to our study have been carried out, and we summarize several related studies regarding stable matching with dynamic preference. Table 1.2 and 1.3 describe the related study of short-term and long-term stability strategies, respectively.

Table 1.2 Related work to short-term stable matching. n denotes the number of men (or women), r denotes the size of the reduced instance from the previous matching

Authors	Algorithm/ Technique	Computational Cost	Re-matching cost
Gale and Shapley [14]	Gale-Shapley algorithm	$O(n^2)$	n
Roth and Vande [14]	Paths to Stability	$O(n^3)$	$n-1$
Alimudin and Ishida [4] (This study)	Update Matching	$O((n-r+1)\cdot n^2)$	$n-r$

Table 1.2 describes the related work to short-term stability strategy. Gale-Shapley is the most efficient approach for finding stable matching in terms of computing cost. In stable matching with dynamic preference, however, the preference may change at any time. We must also consider the re-matching costs that must be spent. Because Gale-Shapley always begins the matching process from scratch, it charges an excess amount of costs. Roth and Vande [25] introduce the mechanism to update the matching by satisfying the blocking pair. However, this mechanism starts by satisfying one random blocking pair

and then satisfying each agent until stable matching is found in the market. In this research, we improve the Roth and Vande algorithm to update a matching whenever the agent's preferences change. When the agent's preference changes, our proposed approach can minimize the costs associated with re-matching.

Table 1.3 Related work to Long-term stable matching. n denotes the number of men (or women), k denotes the number of instances in the dynamic instance

Authors	Algorithm/ Technique	Computational Cost	Objective/ Advantage
Miyazaki et al. [24]	Most stable matching	$O(k \cdot n! \cdot n^2)$	Counting the number of stable matching in dynamic instance
Aziz et al. [7]	Most stable matching	$O(k \cdot n! \cdot n^2)$	Consider the probability of instance
Chen et al. [11]	Most stable matching	$O(k \cdot n! \cdot n^2)$	Define the concept of stability for dynamic preference
Alimudin, Ishida, and Suzuki [5] (This study)	Least blocking pair	$O(k \cdot n! \cdot n^2)$	Blocking pair as the new information to define stable matching under dynamic preference

Table 1.3 describes several references in line with the long-term stability strategy. Some studies use the most stable matching concept to find long-term stability. Several related works used the most stable matching approach to find long-term stability in stable matching under dynamic preference. The stability of a matching in each instance is used as a reference for determining stable matching in dynamic instances. In this study, the perspective of blocking pairs is used as a reference to find stable matching under dynamic preference. Then, the number of blocking pairs is quantified in all available instances to select a matching with the smallest number of blocking pairs. Biro et al. [9] try to find the maximum stable matching with the minimum blocking pair in the stable matching problem with ties (SMT). However, SMT is the restriction of the stable matching problem under dynamic preference. Based on the knowledge

of this research, there seems to be no existing study on the stable matching problems under dynamic preference that consider the blocking pair approach to find a stable matching. Only Aziz et al. [7] mention this idea in their open questions part.

1.2 Research Focus

This study focuses on the two-sided matching problem under dynamic preference. We used the Stable Marriage Problem (SMP), a one-to-one model of the two-sided matching problem, as our primary model in this study. We solve SMP under dynamic preference, where the agent is allowed to change their preference, thus affecting the stability of a matching. In this study, we offer two strategies to obtain matching stability under dynamic preferences.

Our first strategy is short-term stability, which is a step to maintain the stability of a matching whenever preferences change. In this model, an agent's preferences can change over a certain period, affecting the stability of the matching that has been obtained. The preference changes with certain specifications trigger our system to perform re-matching to get stable matching for the latest preferences. This strategy is suitable for matching problems where the intensity of preference changes is not too high.

Our second strategy is long-term stability, a step to finding stable matching under dynamic preference for a long period. Thus, the preference changes do not always trigger re-matching. In this strategy, we offer a new concept for the stability of matching under dynamic preference. The definition of stability of classical SMP needs to be extended for this model. We propose a new concept to find the stable matching under dynamic preference using the blocking pair perspective. Furthermore, we also introduce three notions of stability for the long-term stable matching.

We implemented our system into virtualization technology to show that our strategy can be applied to real-world problems. We use our algorithm for a

job scheduling application on Kubernetes container orchestration technology. In this implementation, the goal of job scheduling focuses on server benefits (stable matching with server-optimal), aiming for resource utilization efficiency in the data center.

1.3 Contributions

This thesis focuses on stable matching problem under dynamic preference. We propose several strategies to find the stable matching under dynamic preferences. This research's key contributions will be summarized as follows:

- Our first strategy, the short-term stability. We maintain matching stability by updating a matching when the preference changes (rematching). Our contribution to the first strategy is to minimize revision costs when performing rematches. Our theorem demonstrate how to update a matching starting with a smaller stable matching.
- Our second strategy, the long-term stability. We introduce a new concept to find a stable matching under dynamic preference based on the blocking pair perspective. Our findings show that our approach has more detailed results. The more detailed result can help determine the stable matching precisely.
- Since the original concept of matching stability is difficult to implement in the matching problem with dynamic preference, we introduce three notions of stability for the stable matching problem under dynamic preference.

1.4 Thesis Organization

The remainder of this thesis is organized as follows:

- In **Chapter 2**, we provide some relevant background theories, fundamental concepts, and related work that were underlying this thesis. The related background concept includes stable matching problems and applications.
- In **Chapter 3**, we provide our first proposed solution to handle stable matching under dynamic preference. We introduce a mechanism and Theorems to find a short-term stable matching under dynamic preference.
- In **Chapter 4**, we provide our second proposed solution to handle stable matching under dynamic preference. We introduce a mechanism to find a long-term stable matching under dynamic preference. We also introduce a new concept of stability for stable matching under dynamic preference.
- In **Chapter 5**, to show the applicability of our findings, we demonstrates the implementation of stable matching under dynamic preference for data center scheduling problem.
- In **Chapter 6**, we are summing up the overall results of this thesis and identifying the remaining challenges for future works.

Chapter 2

Background Concepts

2.1 Stable Matching Problem

The term "matching" refers to a collection of agents wishing to form a pair that meets each agent's criterion. The criterion in question is stability, which depends on each agent's fixed preferences. The essential characteristic of a stable matching is that it cannot be broken by unmatched agents or by any coalition of agents.

2.1.1 Stable Marriage Problem

The stable marriage problem (SMP) was introduced by Gale and Shapley [14]. An SMP is a two-sided matching problem with an equal number of agents on each side. Each agent expresses a strict order preference that includes all members of the opposite side. The goal of the Gale-Shapley algorithm is to find a stable matching for all involved agents. A matching process is a step to determine the pairs of participants (sets of men and women) to meet the specified criteria.

The size n of the SMP instance is $I = (M, W, L)$, where M denotes the set of male agents $M = \{m_1, m_2, \ldots, m_n\}$, W denotes the set of female agents $W = \{w_1, w_2, \ldots, w_n\}$, and L represents a list of the preference order of an

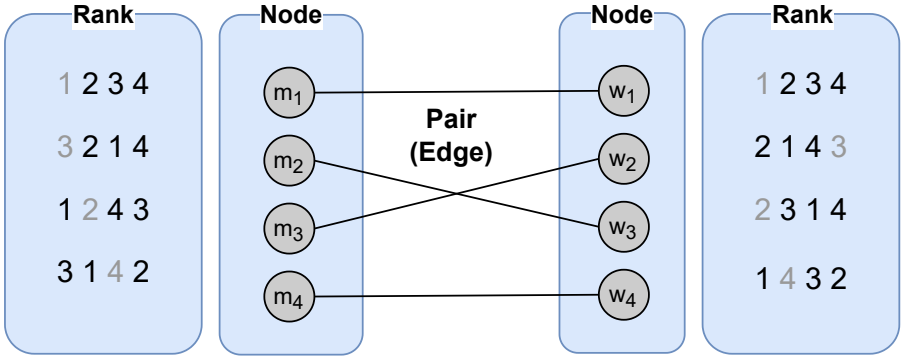

Fig. 2.1 The illustration of stable marriage problem.

agent for the opposite sex. To express the preference order list of agent m_1, we can denote it by $L(m_1)$. For such an instance, a matching μ is a one–one correspondence between the men and the women. If a man m and a woman w are matched in μ, they are referred to as partners, and we write $m = \mu(w)$ and $w = \mu(m)$. $\mu(m)$ is the partner of w in μ, and $\mu(w)$ is the partner of m in μ.

In the SMP, the stability of matching is determined by the existence of a blocking pair. In a matching μ, a pair (m, w) is said to be a blocking pair if m and w are not a pair in μ, but m prefers w over $\mu(m)$ and w prefers m over $\mu(w)$. That is, $(w \succ_m \mu(m)) \wedge (m \succ_w \mu(w))$. A matching that contains at least one blocking pair is called unstable; otherwise, it is stable. The illustration of SMP is depicted in Figure 2.1.

2.1.2 Find Stable Matching

The Gale-Shapley algorithm is a solution to find stable matching in a stable marriage problem. The Gale-Shapley algorithm can always find stable matching in a stable marriage problem. The basic Gale-Shapley algorithm in which the men as the proposer (*man-oriented* version), is summarized as follows in Algorithm 1

All possible executions of the Gale-Shapley algorithm (the men as proposers) yield the exact stable matching, and in this stable matching, each man has the

Algorithm 1: Basic Gale-Shapley algorithm

```
while some man m is free do
  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 if
  end if
end while
output the stable matching consisting of the n engaged pairs
```

best partner he can have in any stable matching. The Gale-Shapley algorithm drives the result based on the proposing side. If the men are proposers, each man will get the best partner he can, and vice versa; if the women are proposers, each woman will get the best partner she can.

In order to check the stability of a matching, the following Algorithm 2 can simply check the stability of a matching

Algorithm 2: Checking a matching stability

```
for m ← 1 to n do
  for each w such that m prefers w to M(m) do
    if w prefers m to M(w) then
      report matching unstable
      halt
    end if
  end for
end for
report matching stable
```

2.1.3 The Set of Stable Matchings

In an SMP instance, the Gale-Shapley algorithm will always produce optimal stable matching. Namely man-optimal and woman-optimal matching. In a

1		1	2	3	4	1		4	3	2	1
2		2	1	4	3	2		3	4	1	2
3		3	4	1	2	3		2	1	4	3
4		4	3	2	1	4		1	2	3	4
	Men's Preferences						Women's Preferences				

Fig. 2.2 Size 4 Stable marriage instance preference

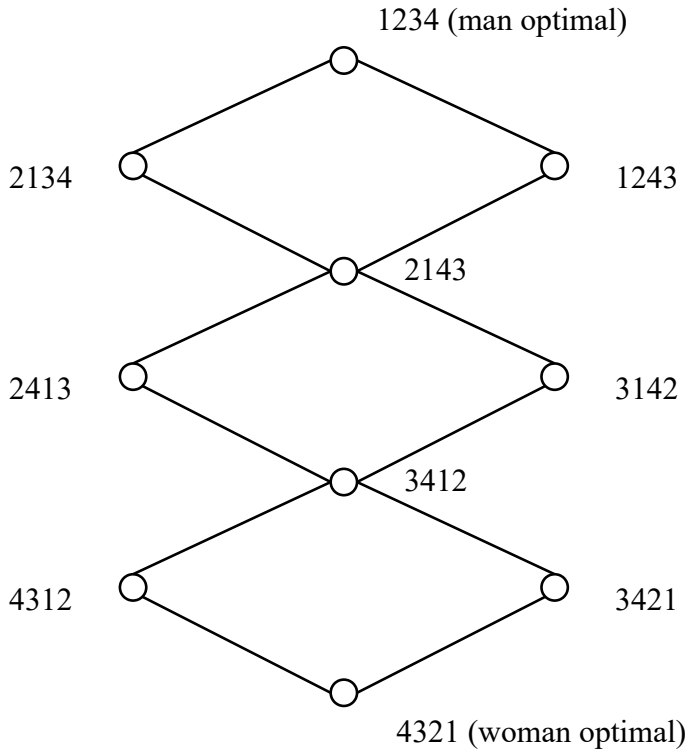

Fig. 2.3 Lattice structure for the current instance

man-optimal matching algorithm, the goal is to find a stable matching in which each man agent is matched with the best possible partner according to his own preference. Similarly, in a woman-optimal matching algorithm, the goal is to find a stable matching in which each woman agent is matched with the best possible partner according to her own preference. In SMP, sometimes man-optimal and woman-optimal stable matching can be equal. However, the

number of stable matching for each SMP instance can be more than man-optimal and woman-optimal. There are several matching intermediate values. A set of stable matching will form a lattice structure. Consider the example of size 4 described by the preference list in Figure 2.2. The instance in Figure 2.2 admits a total of ten stable matchings. The lattice structure is illustrated in Figure 2.3 is a directed graph with a node for each element of the lattice, and a directed edge from node x to node y if $x \prec y$ and there is no z such that $x \prec z \prec y$.

2.2 The Hospitals/ Residents Problem

Hospitals/Residents Problem (HRP) is a generalization of SMP. SMP is a two-sided matching with one-to-one agent correspondence, where the number of disjoint sets equals. In SMP, the agent is illustrated with men and women. HRP is a two-sided matching with a one-to-many model, where the number of disjoint agent sets is allowed to be different. In HRP, agents are represented by hospitals and residents (interns). In HRP, a hospital can accommodate one or more residents by defining their quota in advance. Figure 2.4 illustrate the hospitals/residents problem.

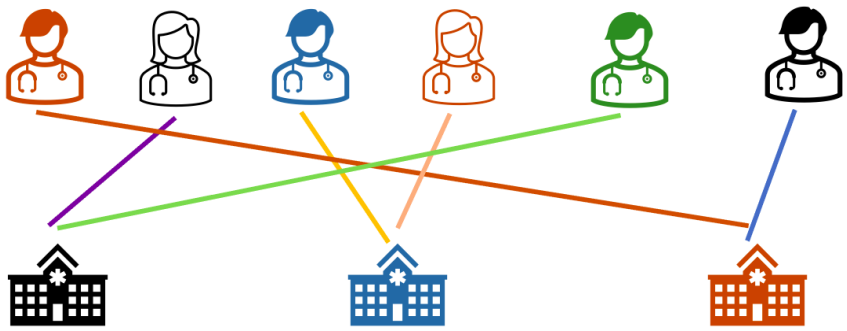

Fig. 2.4 Illustration of hospitals/residents problem

We formally define the hospitals/residents problem (HRP). An instance of HRP is a set of residents $R = r_1, r_2, \ldots, r_n$ and hospital $H = h_1, h_2, \ldots, h_m$.

Each hospital $h_j \in H$ has a positive integer of capacity value denoted by $C(h_j)$, where $C(h_j) \geq 1$. Each resident $r_i \in R$ has a set of preference list L where residents will rank each member of H in strict order. To denote the preference of agent r_i, we can write $L(r_i)$. Likewise, hospitals $h_j \in H$ also have a set of preference list where the hospital will rank each member of R in strict order. An agent $p \in R \cup H$ is acceptable in $q \in R \cup H$ if agent p appears on q's preference list, $p \in L(q)$; otherwise, p cannot be accepted at q. A matching M is a subset of RxH, where $(r, h) \in M$ with conditions (i) r and h acceptable to each other, (ii) r is assigned at most one hospital, (iii) h accepts residents at most $C(h)$. a matching M is said to be stable if no blocking pair is found. (r, h) is a blocking pair in matching M, if (i) r and h accept each other, (ii) r is unassigned, or r prefers h over $M(r)$, and (iii) h is under-subscribed or prefers r over at least one member of $M(h)$.

2.3 Paths to Stability Mechanism

Since its introduction in 1962, the Gale-Shapley algorithm has attracted much research interest. Several stable matching variants also emerged, followed by the extension of the Gale-Shapley algorithm. The Gale-Shapley algorithm always terminates when it finds a stable matching, requiring $O(n^2)$.

One of the well-known extensions of the Gale-Shapley algorithm is the paths to stability mechanism [25], known as the RV mechanism. The mechanism is based on the Gale-Shapley algorithm but can perform stable matching searches starting with an arbitrary pair. The Gale-Shapley algorithm always finds optimal stable matching (man or woman optimal), however, the Gale-Shapley algorithm cannot find stable matching other than man or woman optimal. One of the advantages of this mechanism is that it can discover stable matching other than man and woman optimal.

Jinpeng Ma [21] shows that the RV mechanism can find stable matching other than optimal man and woman, although not all stable matching in an

instance is entirely discovered. They also summarized the RV mechanism and admitted that stable matching can always be achieved with a probability of one starting from satisfying blocking pairs in arbitrary matching.

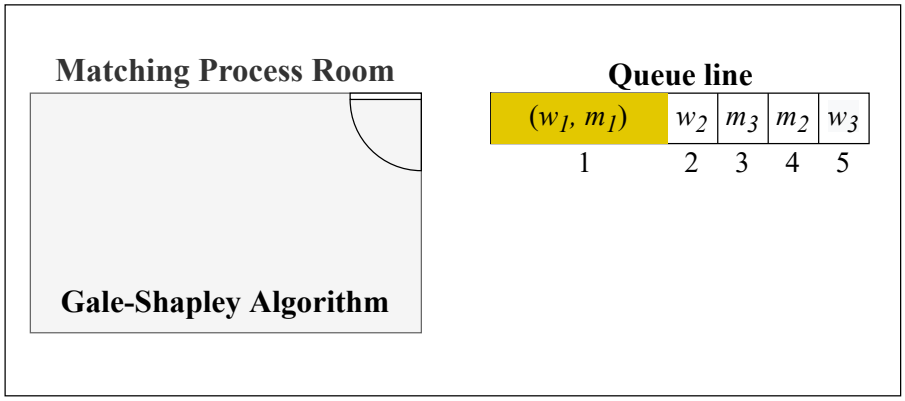

Fig. 2.5 Paths To Stability Mechanism Illustration

Roth's and Vande's work can be analogized as follows:

1. Imagine that there is one room with one entrance; randomly select a pair from each matching process. Let the selected pair enter the room. The selected pair can be confirmed as a stable matching in this room because no other choice can break the pair. Meanwhile, the rest of the agents form a queue outside the room to enter the room one by one.

2. Ask an agent who is in front of the room to enter the room. There will be a matching process inside the room. The door of the room will remain closed before a stable matching is formed in the room.

3. Repeat the second step until there are no remaining queues and a stable matching is obtained. As a result, a stable matching will be obtained without any blocking pairs.

Figure 2.5 illustrated how the RV mechanism finds stable matching.

[This page is intentionally left blank]

Chapter 3

Short-term Stability for Matching Problem under Dynamic Preference

3.1 Introduction

In the stable matching problem, generally, each agent expresses a preference with certainty. However, as we discussed in the introduction, in real-world situations, sometimes an agent cannot express their preference in certainty, causing their preference to be dynamic. Consequently, a preference change in an agent may affect the stability of the obtained matching.

This study aims to find a stable matching for the SMP with the dynamic preference model, where dynamic preferences refer to a scenario in which agents' preferences might change dynamically over time. To illustrate the problem, we use a simple 3×3 SMP.

Example 3.1. *There is a set of men* $M = \{m_1, m_2, m_3\}$ *and women* $W = \{w_1, w_2, w_3\}$*. Assume that agent* m_1 *has dynamic preferences by changing his preference order. Specifically, these dynamic preferences generate two SMP instances, as depicted in Figure 3.1:*

***Instance 1* (I_1)**

$L_1(m_1) = w_1, w_2, w_3$ $\quad$ $L_1(w_1) = m_2, m_3, m_1$

$L_1(m_2) = w_2, w_3, w_1$ $\quad$ $L_1(w_2) = m_3, m_1, m_2$

$L_1(m_3) = w_3, w_1, w_2$ $\quad$ $L_1(w_3) = m_1, m_2, m_3$

***Instance 2* (I_2)**

$L_2(m_1) = w_3, w_2, w_1$ $\quad$ $L_2(w_1) = m_2, m_3, m_1$

$L_2(m_2) = w_2, w_3, w_1$ $\quad$ $L_2(w_2) = m_3, m_1, m_2$

$L_2(m_3) = w_3, w_1, w_2$ $\quad$ $L_2(w_3) = m_1, m_2, m_3$

Fig. 3.1 The 3 × 3 dynamic preference of SMP.

In this chapter, we assume that preference changes infrequently occur. Traditionally, we can use the Gale–Shapley algorithm to find a stable matching solution for each instance. In Example 3.1, we can find the stable matching for Instances 1 and Instance 2 by processing them separately. However, the agent's preference changes do not necessarily affect all pairs' stability in a matching. Some small preference changes may not affect all agents, and only a few pairs want to change their partners, while others prefer to stay with their current partners. This motivates us to maintain stability in the SMP with dynamic preferences by updating the matching. The SMP with dynamic preferences is an extended version of the original SMP; the goal is to obtain a stable matching on the SMP instance. The Gale-Shapley algorithm is one way to find a stable matching for the SMP. However, the Gale-Shapley algorithm cannot update the matching, the algorithm always starts the matching with an empty matching. Roth and Vande [25] introduce the mechanism to update the matching by satisfying the blocking pair. This mechanism starts by satisfying one random blocking pair and then satisfying each agent until stable matching is found in the market.

We try to minimize the re-matching costs when preferences change. Using the Gale-Shapley algorithm, we can always find a stable matching, but the

matching process starts from the beginning (empty matching). We try to minimize the re-matching cost when performing the new matching by using the update mechanism. Our theorems and mechanism demonstrate the process of finding stable matching by updating the previous matching.

The main contribution of this chapter is we introduce a short-term strategy for maintaining a stable matching by updating a matching when the preference changes. We demonstrate the process of finding stable matching by updating a matching without a cyclic process.

3.1.1 Problems

In this section, we will examine the issues that will be addressed in this chapter. This chapter discusses an SMP with dynamic preferences, in which an agent's preferences can change. Many existing algorithms can be utilized to find stable matching, however, they are less efficient for SMP with dynamic preferences. Here is a summary of the problems we intend to address in this chapter:

1. What is an efficient method for obtaining stable matching when preferences change

2. How to efficiently update a matching so that it remains stable against the new preferences.

The two issues listed above are the issues that will be addressed in this chapter. In the first issue, the Gale-Shapley algorithm can be used to discover a stable matching whenever the preferences change. However, the Gale-Shapley algorithm's limitation is that it must always begin the matching process from scratch (empty matching). In this chapter, we propose a solution for finding a matching each time the preferences change by modifying the prior matching. By utilizing the update method, we may reduce the re-matching costs.

The second issue we encounter is how to efficiently update a matching to obtain a stable matching against the new preference. Roth and Vande [25]

introduce the mechanism to update the matching by satisfying the blocking pair. However, this mechanism always starts the process with one pair (initial matching) and then satisfies each agent until stable matching is found in the market. To address this issue, we intend to maximize the initial matching, by identifying the blocking pair of the prior stable matching against the new preference. Our findings help identify the blocking pair of prior matching and produce the maximum initial matching. Therefore, the re-matching costs can be minimized.

3.2 Find Stable Matching using Gale-Shapley

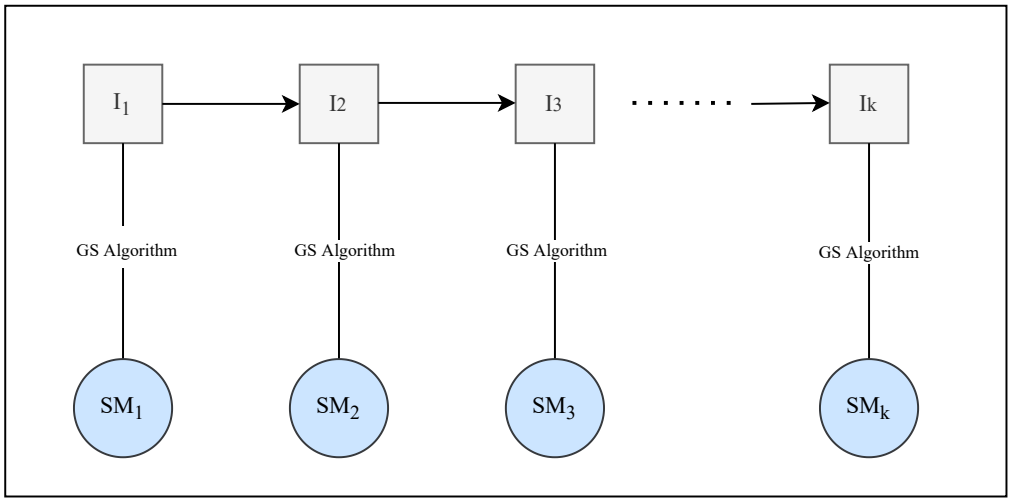

Fig. 3.2 Maintaining Stability of SMP using Gale-Shapley Algorithm

Re-matching is one of the possible strategies to maintain short-term stability in dynamic preference, this strategy requires starting the process of finding stable matching when the preference changes. A stable matching for each instance can be found by simply using the Gale-Shapley algorithm.

Figure 3.2 is an illustration of maintaining the stability for each instance using the Gale-Shapley algorithm. However, preference changes for an agent do not necessarily affect the stability of all pairs in a matching. Some small

preference changes may not affect all agents, and only a few pairs want to change their partners, while others prefer to stay with their current partners.

To illustrate the problem, consider Example 3.2.

Example 3.2. *There is a set of men* $M = \{m_1, m_2, m_3\}$ *and women* $W = \{w_1, w_2, w_3\}$*. Assume that agent* w_1 *change her preference.*

Instance 1 (I_1)

$L_2(m_1) = w_2, w_1, w_3$	$L_2(w_1) = \mathbf{m_2, m_3, m_1}$
$L_2(m_2) = w_2, w_3, w_1$	$L_2(w_2) = m_3, m_1, m_2$
$L_2(m_3) = w_3, w_1, w_2$	$L_2(w_3) = m_1, m_2, m_3$

Instance 2 (I_2)

$L_2(m_1) = w_2, w_1, w_3$	$L_2(w_1) = \mathbf{m_3, m_1, m_2}$
$L_2(m_2) = w_2, w_3, w_1$	$L_2(w_2) = m_3, m_1, m_2$
$L_2(m_3) = w_3, w_1, w_2$	$L_2(w_3) = m_1, m_2, m_3$

Suppose μ_1 is the man-optimal of I_1, where $\mu_1 = \{(w_1 : m_3), (w_2 : m_1), (w_3 : m_2)\}$. In I_2. However, we do not sure whether μ_1 is stable or not. Then we need to find the stable matching for I_2 using Gale-Shapley. The stable matching for I_2 is $\mu_2 = \{(w_1 : m_3), (w_2 : m_1), (w_3 : m_2)\}$. In I_2. Turn out that $\mu_1 \equiv \mu_2$, means that μ_1 is stable in I_2. This motivates us to maintain stability in the SMP with dynamic preferences by using a matching-updating mechanism.

3.3 Updating Matching Mechanism

Our prior research investigated the RV mechanism to establish stable matching with dynamic preference [3]. In this research, we attempt to identify every stable pairing of all potential preferences that can arise when preferences are dynamic. We attempt to identify stable matching solutions for each generated preference using the modified RV mechanism. This work is the preliminary

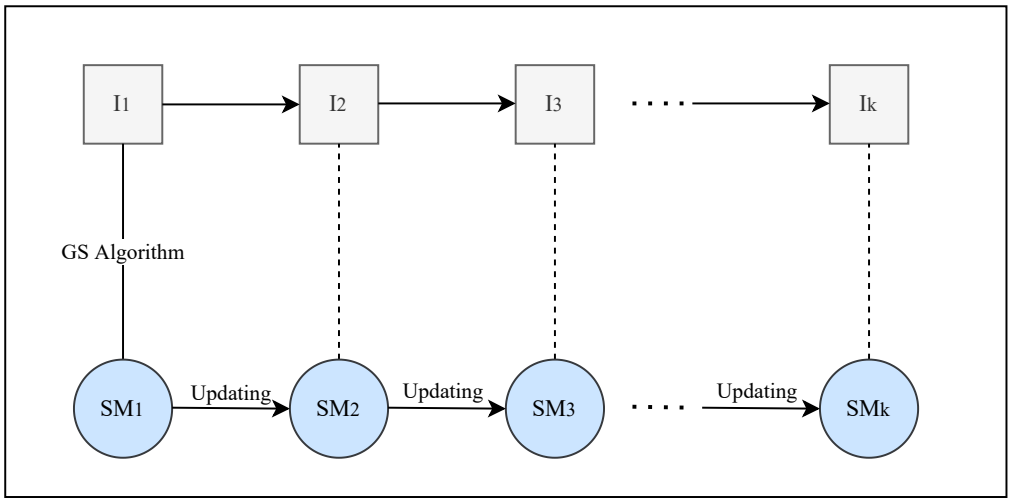

Fig. 3.3 Maintaining Stability of SMP using Updating Mechanism

step to the development of an update matching mechanism as a short-term stable matching strategy. In this study, we attempted to compare the Gale-Shapley algorithm with the RV mechanism for stable matching with dynamic preference. In terms of algorithm efficiency, the Gale-Shapley algorithm is a very efficient algorithm to discover stable matching. However, in the case of stable matching with dynamic preference, the study shows that the RV mechanism is useful when it is unclear whether the Gale-Shapley algorithm should be implemented using man-optimal or woman-optimal settings.

Re-matching can maintain the stability of a matching. Figure 3.2 shows how to find stable matching using the traditional method by performing the matching process from scratch on each existing instance using the Gale-Shapley algorithm. Instead of using the conventional method of finding stable matches for each instance, we use the previous update matching mechanism to find new stable matching, as illustrated in Figure 3.3. To update a matching, we use the following three steps:

1. Identify the preference changes for each agent, whether these changes have the potential to form a blocking pair or not

2. If a potential blocking pair is detected, initiate update matching

3. Else, the previous matching remains stable on the new instance

3.3.1 Identifying the Preference Changes

Preference changes in an instance do not necessarily affect the stability of a stable matching. Sometimes specific changes do not destroy the stability of a stable matching. Therefore, we classify preference changes into two, minor and major changes.

Definition 3.1. *A minor preference change is a change in the agent's preference that does not affect the pair involving the agent in a matching. Preference changes for this agent will not trigger the formation of a blocking pair.*

Definition 3.2. *A major preference is a change in the agent's preference that potentially breaks a pair of stable matching. This major change directly impacts the stability of an existing matching.*

Every agent can change their preferences at any time in the dynamic preference model. A change in an agent's preference does not always lead to a change in matching. However, it can occasionally allow for the appearance of blocking pairs, leading to the instability of a previous matching.

Given an SMP with a dynamic preference of size n and with the initial instance $I = (M, W, L)$ and a stable matching μ, where $M = \{m_1, m_2, \ldots, m_n\}, W = \{w_1, w_2, \ldots, w_n\}$, Suppose that a man agent (m_1) changes his preference through permutation, we will have another instance of the SMP: $I' = (M, W, L')$, where $L(m_1) \neq L'(m_1)$. By referencing the previous SMP instance, we determine the occurrence of potential blocking pairs. We identify whether the preference change for agent m_1 is a minor change or not.

Theorem 3.1. *Under the assumption that the preference change occurs on only one side. Let $l \subset L(m_1)$ and $(\mu(m_1) \succ_{m_1} l)$. IF $\mu(m_1) \succ_{m_1} l$ in L', m_1 preference changes is minor change.*

Proof. Assume that the position of agent $\mu(m_1)$ in $L(m_1)$ is the $(n-1)^{th}$ rank; this means that agents from the first rank to the $(n-2)^{th}$ rank in $L(m_1)$ prefer other male agents over m_1. The agent at the n^{th} rank is a worse agent than $\mu(m_1)$, which means that $\mu(m_1)$ is the best option for m_1 at I (man-optimal). Thus, as long as the n^{th} agent's position remains lower than $\mu(m_1)$ in $L'(m_1)$, any permutation in the agent from the 1^{st} rank to $\mu(m_1)$ in $L(m_1)$ does not affect the pair $(m_1, \mu(m_1))$. This results in $\mu(m_1) = \mu'(m_1)$. □

Based on Theorem 3.1, we describe the following corollaries.

Corollary 3.1.1. *If $\mu(m_1)$ is the last choice of m_1 in $L(m_1)$, then any permutation in $L(m_1)$ leads to minor change.*

Proof. If $\mu(m_1)$ is the last order in $L(m_1)$, $\mu(m_1)$ is considered the best choice that m_1 can get because other women agents prefer other men agents over m_1. As a result, $(m_1, \mu(m_1))$ will not change, although the rank of $\mu(m_1)$ increases in $L'(m_1)$. □

Corollary 3.1.2. *If $\mu(m_1)$ becomes the first choice of m_1 in $L'(m_1)$, then it is a minor change.*

Proof. If $\mu(m_1)$ is the first option in $L'(m_1)$, then no agent that is worse than $\mu(m_1)$ in $L(m_1)$ becomes better than $\mu(m_1)$ in $L'(m_1)$. Moreover, if the rank of $\mu(m_1)$ increases in $L'(m_1)$, it strengthens the pair between $\mu(m_1)$ and m_1. □

Corollary 3.1.3. *If there is any permutation in $L(m_1)$ from the first rank to $\mu(m_1)$ or any permutation from the rank of $\mu(m_1)$ + 1 to the n^{th} rank, then it is a minor change.*

Proof. The permutation of the first rank to $\mu(m_1)$ does not lead to a worse agent being superior to $\mu(m_1)$. Meanwhile, permutations from the rank of $\mu(m_1)$ +1 to the lowest-ranked agent will also be inferior to $\mu(m_1)$. According to Theorem 1, it is a minor change if no agent worse than $\mu(m_1)$ at $L(m_1)$ increase in $L'(m_1)$. □

Theorem 3.1 and its corollary define the constraints under which the effect of changing agents on male agents (proposing side) can be identified. However, Theorem 3.1 and its corollary hold true when female agents (the proposed side) possess a change in their preferences. We identify the impact of changing an agent's preferences using Theorem 3.1 and its corollary and whether the change has the potential to lead to the establishment of a blocking pair in the new instance or not. This identification accelerates the decision-making regarding updating a matching.

Algorithm 3: Finding potential blocking pair

Input :

- m = man agent, w = woman agent
- current SMP Instance: $P(m)$ = men preference, $P(w)$ = women preference
- previous SMP Instance: $P_0(m)$ = men preference,$P_0(w)$ = women preference
- SM_0 = Previous Stable Matching

```
Def checkPreferenceChange:
    for m,w in SM_0 do
      if P(m) != P_0(m) then
        for w_0 in P_0(m) do
          if rank(w) in P_0(m) > rank(SM_0(w_0)) in P_0(m) then
            assign w_0 to PotentialBP
          end if
        end for
        for w in P(m) do
          if w in PotentialBP then
            if rank(m) in P(w) > rank(SM_0(w_0)) in P(w) then
              remove pair (m, w) from SM_0
            end if
          end if
        end for
      end if
    end for
```

We define Algorithm 3 to detect potential blocking pairs based on Theorem 3.1 and its corollaries. If Theorem 1 and its corollaries are satisfied in an agent, the agent might not have the incentive to change his/her partner. However, if Theorem 3.1 and its corollary are not satisfied, a deeper checking process using the stability-checking algorithm [15] will be performed. If a blocking pair is detected, the algorithm will mark the pair for removal and initiate the update of the matching process.

Theorem 3.1 assumes that the preference changes only occur on one side. Theorem 3.1 is used to determine the type of preference change that occurs in an agent, i.e., whether it is a major or minor change. The aim is to determine whether the preference changes incentivize an agent to change their partner. By checking the type of preference change for all agents with Theorem 1, we are still able to identify the potential blocking pair, even though both sides of the agents change their preferences simultaneously. Given an SMP with a dynamic preference of size n and with initial instance $I = (M, W, L)$ and a stable matching μ, where $M = \{m_1, m_2, \ldots, m_n\}, W = \{w_1, w_2, \ldots, w_n\}$, if we suppose that a male agent (m_1) and his partner $(\mu(m_1))$ change their preference simultaneously, we will have another instance of the SMP: $I' = (M, W, L')$, where $L(m_1) \neq L'(m_1)$ and $L(\mu(m_1)) \neq L'(\mu(m_1))$.

Corollary 3.1.4. *If m_1 and $\mu(m_1)$ confirm a minor change in I', then the pair (m_1, $\mu(m_1)$) does not trigger a blocking pair*

Proof. If agent m_1 confirms that his preference change is minor, $\mu(m_1)$ is the best possible woman that agent m_1 can get. In addition, if agent $\mu(m_1)$ also confirms that her preference change is minor, then agent m_1 is the best possible option for agent $\mu(m_1)$ because the proposer is m_1. Thus, if both agents agree that their current partners are the best partners they can get, then the preference changes of agents μ_1 and $\mu(m_1)$ do not have the potential to form a blocking pair. □

It is worth noting that Theorem 3.1 is employed to determine whether a change in an agent's preference can motivate an agent to change their current partner. Theorem 1 is used to verify agents' wishes about their partners following their preference changes. In other words, although both agents in a pair confirm a minor change, the pair is still likely to change if provoked by another agent that does not confirm a minor change.

3.3.2 Initiating the Matching Update

Once a blocking pair is confirmed to exist in a matching, the next step is to initiate a matching update. We modified Roth's and Vande's mechanism to update the stable matching. Random Paths to Stability [25] is a mechanism for discovering a stable matching solution by satisfying the matching's blocking pair. Ref. [21] summarized the RV mechanism and admitted that stable matching can always be achieved with a probability of one starting from satisfying blocking pairs in arbitrary matching. Roth's and Vande's work can be analogized as follows:

1. Imagine that there is one room with one entrance; randomly select a pair from each matching process. Let the selected pair enter the room. The selected pair can be confirmed as a stable matching in this room because no other choice can break the pair. Meanwhile, the rest of the agents form a queue outside the room to enter the room one by one.

2. Ask an agent who is in front of the room to enter the room. There will be a matching process inside the room. The door of the room will remain closed before a stable matching is formed in the room.

3. Repeat the second step until there are no remaining queues and a stable matching is obtained. As a result, a stable matching will be obtained without any blocking pairs.

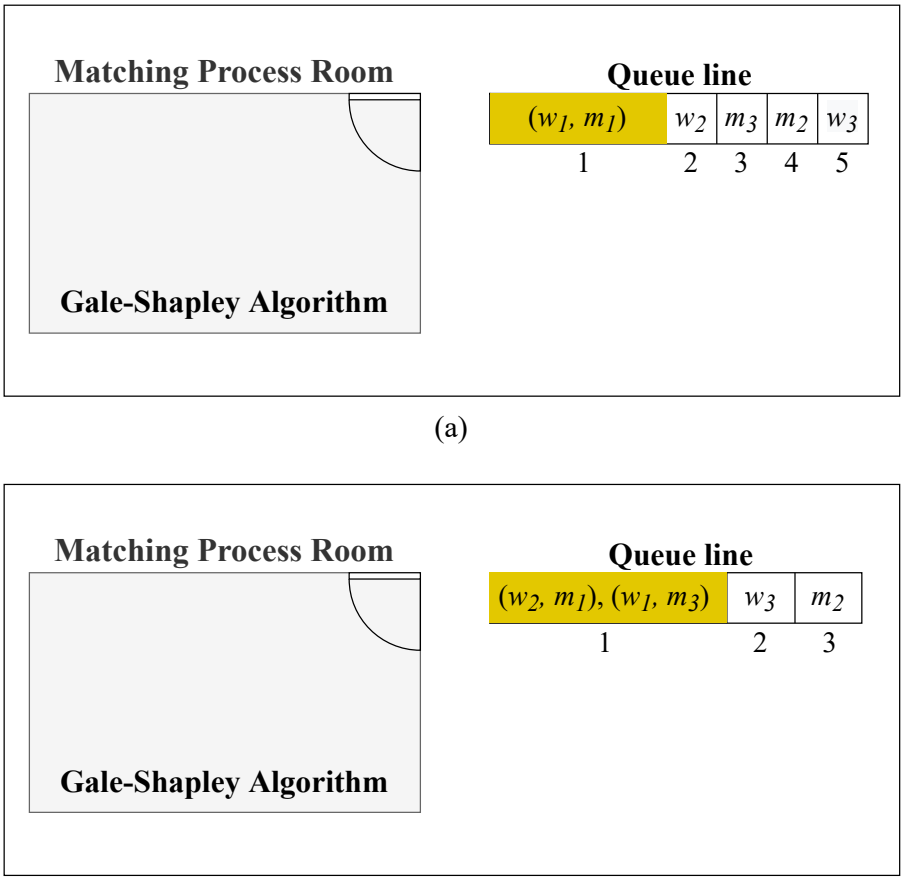

Fig. 3.4 Comparison of the original Roth and Vande mechanism (**a**) and our update mechanism

The RV mechanism begins the matching process by asking a single random pair to enter the room. Typically, a blocking pair will be selected in the initialization process. Following that, the remaining agents in the queue will be asked to enter the room separately. In the RV mechanism, new agents are not permitted to enter the room until all agents in the room reach stability. Therefore, this motivated us to shorten the finding of stable matching in the RV mechanism. The RV mechanism begins the matching process with a single pair for the initial part, while our proposed algorithm enables the initial process with multiple pairs. This significantly speeds up the process of finding stable matches by

introducing multiple pairs into the room simultaneously. Here is our mechanism for updating a stable matching:

1. Imagine that there is one room with one entrance; select some pairs that have been confirmed as a stable matching. Let of all the selected pairs enter the room together. Meanwhile, the rest of the agents form a queue outside the room to enter the room one by one.

2. Ask an agent who is in front of the room to enter the room. There will be a matching process inside the room. The door of the room will remain closed before a stable matching is formed in the room.

3. Repeat the second step until there is no remaining queue and a stable matching is obtained. As a result, a stable matching will be obtained without any blocking pairs.

Figure 3.4 shows an illustration of the original RV mechanism and our updating mechanism. Our mechanism allows more than one pair to enter the room at the initial stage.

3.3.3 Reducing the Previous Matching

Given an SMP of size n with an instance $I = (M, W, L)$, where $M = \{ m_1, m_2, \ldots, m_n \}$, $W = \{w_1, w_2, \cdots, w_n \}$, the matching μ is unstable if there exists a blocking pair (m_i, w_j) such that $(w_j \succ_{m_i} \mu(m_i)) \wedge (m_i \succ_{w_j} \mu(w_j))$ for $m_i \in M, w_j \in W$, where $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n$. Equivalently, the stability for matching μ can be expressed as: $(\mu(m_i) \succcurlyeq_{m_i} w_j) \vee (\mu(w_j) \succcurlyeq_{w_j} m_i)$ for $\forall m_i \in M, \forall w_j \in W$, where $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n$.

Theorem 3.2. *If we remove any pair from the stable matching μ and SMP instance I, the reduced matching μ_r remain stable for the reduced instance I_r.*

Proof. Without loss of generality, we assume the removal of $(m_1, w_1) = (m_1, \mu(m_1))$. Then, $M' \equiv M \backslash m_1$ and $W' \equiv W \backslash w_1$, $\mu_r = \{(m_2, \mu(m_2)), \ldots, (m_n, \mu(m_n))\}$.

The stability condition for the matching μ_r is expressed as: $(\mu(m_i) \succcurlyeq_{m_i} w_j) \vee (\mu(w_j) \succcurlyeq_{w_j} m_i)$ for $\forall m_i \in M\ ', \forall w_j \in W\ '$, where $i = 2, \ldots, n$ and $j = 2, \ldots, n$. When we remove m_1 from the preference list of women and w_1 from the preference list of men, there are only two possibilities for men: m_i prefers w_1 to the partner $\mu(m_i)$ in the stable matching μ, or m_i prefers the partner $\mu(m_i)$ to w_1 in the stable matching μ. Likewise, there are only two possibilities for women, which makes $2 \times 2 = 4$ combined cases altogether. We will check the stability condition above for the reduced matching μ_r for the reduced instance I_r.

Case 1: If $(w_1 \succ_{m_i} \mu(m_i))$ and $(\mu(w_j) \succ_{w_j} m_1)$, then the statement $(\mu_r(w_j) \succcurlyeq_{w_j} m_i)$ for the stability condition of matching μ_r is true.

Case 2: If $(\mu(m_i) \succ_{m_i} w_1)$ and $(m_1 \succ_{w_j} \mu(w_j))$, then the statement $(\mu_r(m_i) \succcurlyeq_{m_i} w_j)$ for the stability condition of matching μ_r is true.

Case 3: If $(\mu(m_i) \succ_{m_i} w_1)$ and $(\mu(w_j) \succ_{w_j} \mu(m_1))$, then both statements $(\mu_r(m_i) \succcurlyeq_{m_i} w_j)$ and $(\mu_r(w_j) \succcurlyeq_{w_j} m_i)$ for the stability conditions of matching μ_r are true.

Case 4: If $(w_1 \succ_{m_i} \mu(m_i)$ and $(m_1 \succ_{w_j} \mu(w_j))$, since both conditions of agents cannot directly provide the true statement, then we need further investigation for this condition: m_i and w_j are a pair in μ and also in μ_r. Then, we can write $m_i = \mu(w_j)$ and $w_j = \mu(m_i)$. If we remove w_1 from the men's preferences, then $\mu(m_i)$ increases; still, we cannot confirm whether $\mu(m_i)$ becomes m_i's first choice or not. If we assume that $\mu(m_i)$ is not m_i's first choice, then m_i prefers another woman (w_x) over $\mu(m_i)$, that is, $(w_x \succ_{m_i} \mu(m_i))$. Currently, the partner of w_x is $\mu(w_x)$. Since μ is stable, then $(m_i \succ_{w_x} \mu(w_x))$ is not possible in μ, although we also remove m_1 from the women's preferences. Then, w_x and m_i never become partners in μ. Thus, the statement $(\mu_r(m_i) \succcurlyeq_{m_i} w_j)$ holds true for the stability condition of matching μ_r. The true condition holds for all possible combinations of pair removals. That said, the stability condition for matching μ_r still holds true. □

As a prior section's discussion, we meant to obtain as many pairs as possible for the initial stage. The pairs wishing to enter at the initial step must be stable with each other. Therefore, we reduce the size of the previous matching to obtain a stable matching with a smaller size. The reduction procedure is performed by removing the unwanted pairs (m, w) from a matching μ that causes the formation of a blocking pair. It should be noted that removing m and w from the previous matching is not permanent. The removal is intended to keep the remaining pairs stable. Thus, the stable reduced matching can be used as the initial stage for finding a stable matching. We must introduce a procedure for determining which pairs to exclude from the matching μ. By using Theorem 3.2, we can safely remove any pair from the previous matching to get the reduced stable matching. Consider the following examples in Figure 3.5. We

Men preference

	1st	2nd	3rd	4th
m_1	w_1	w_2	w_3	w_4
m_2	w_1	w_2	w_3	w_4
m_3	w_1	w_2	w_3	w_4
m_4	w_1	w_2	w_3	w_4

Women preference

	1st	2nd	3rd	4th
w_1	m_4	m_3	m_2	m_1
w_2	m_4	m_3	m_2	m_1
w_3	m_4	m_3	m_2	m_1
w_4	m_4	m_3	m_2	m_1

Fig. 3.5 The 4 × 4 SMP instance before reduction.

have stable matching (man-optimal) $\{(w_1, m_4), (w_2, m_3), (w_3, m_2), (w_4, m_1)\}$. Now, we will try to reduce the instance by randomly removing one pair in the stable matching. Suppose that we remove (w_2, m_3). Thus, the size of the SMP instance will change to 3 × 3, as depicted in Figure 3.6.

Figure 3.6 shows the result of reducing the SMP instance from 4 × 4 to 3 × 3 by randomly deleting one pair, which keeps the reduced matching stable with respect to the reduced instance.

Example 3.3. *Given the SMP instance* $I = (M, W, L)$*, a set of men* $M = \{m_1, m_2, m_3, m_4, m_5\}$*, and a set of women* $W = \{w_1, w_2, w_3, w_4, w_5\}$*, we assume that agent* m_1 *changes his preference and generates two instances as follows:*

Men preference

	1st	2nd	3rd	4th
m_1	w_1	w_3	w_4	
m_2	w_1	w_3	w_4	
m_4	w_1	w_3	w_4	

Women preference

	1st	2nd	3rd	4th
w_1	m_4	m_2	m_1	
w_3	m_4	m_2	m_1	
w_4	m_4	m_2	m_1	

Fig. 3.6 The 3 × 3 SMP instance after reduction.

Instance 0 (I_0)

$L_0(m_1) = \mathbf{w_5, w_4, w_1, w_3, w_2}$ $L_0(w_1) = m_5, m_4, m_2, m_1, m_3$

$L_0(m_2) = w_1, w_3, w_4, w_2, w_5$ $L_0(w_2) = m_2, m_5, m_3, m_1, m_4$

$L_0(m_3) = w_1, w_3, w_2, w_4, w_5$ $L_0(w_3) = m_1, m_3, m_2, m_4, m_5$

$L_0(m_4) = w_3, w_2, w_4, w_1, w_5$ $L_0(w_4) = m_1, m_2, m_4, m_3, m_5$

$L_0(m_5) = w_5, w_2, w_1, w_3, w_4$ $L_0(w_5) = m_3, m_4, m_2, m_5, m_1$

Instance 1 (I_1)

$L_1(m_1) = \mathbf{w_2, w_5, w_1, w_3, w_4}$ $L_1(w_1) = m_5, m_4, m_2, m_1, m_3$

$L_1(m_2) = w_1, w_3, w_4, w_2, w_5$ $L_1(w_2) = m_2, m_5, m_3, m_1, m_4$

$L_1(m_3) = w_1, w_3, w_2, w_4, w_5$ $L_1(w_3) = m_1, m_3, m_2, m_4, m_5$

$L_1(m_4) = w_3, w_2, w_4, w_1, w_5$ $L_1(w_4) = m_1, m_2, m_4, m_3, m_5$

$L_1(m_5) = w_5, w_2, w_1, w_3, w_4$ $L_1(w_5) = m_3, m_4, m_2, m_5, m_1$

Fig. 3.7 The 5 × 5 SMP instances with dynamic preference.

The stable matching of instance 0 is $\mu_0 = \{(w_1, m_2), (w_2, m_4), (w_3, m_3), (w_4, m_1), (w_5, m_5)\}$. Now, we want to find the stable matching of instance 1 by updating μ_0. Comparing L_0 and L_1, it is known that the pair (w_3, m_1) will become a blocking pair in matching μ_0 if we use L_1 as the preference. (w_3, m_1) blocks (w_3, m_3) and (w_4, m_1) in matching μ_0. By using Theorem 2, we allow the removal of the pair (w_4, m_1). Removing (w_4, m_1) from instance 1 will produce the reduced matching $\mu_r = \{(w_1, m_2), (w_2, m_4), (w_3, m_3), (w_5, m_5)\}$, which is stable to the reduced SMP instance (I_r). As illustrated in Figure 3.4b, members of μ_r may immediately enter the room together. Meanwhile, the removed

pair (w_4, m_1) forms a queue outside the room. Algorithm 4 summarizes how to find stable matching by updating the previous matching. By combining Algorithm 3 with the reduced matching, we can update the previous matching and obtain the new stable matching for a new instance. Since we can start the initial process by processing several pairs simultaneously, the revision cost of rematching can be minimized.

3.3.4 Controlling the Matching Orientation

The interesting aspect of the RV mechanism is the variety of stable matchings obtained. Unlike the Gale–Shapley algorithm, which always provides the same stable matching, the RV mechanism can produce a different stable matching for each execution. While it is impossible to guarantee that all stable matchings in the lattice structure are obtained, man- and woman-optimal matching can be guaranteed. According to Jinpeng Ma's research [21], certain circumstances can result in a stable matching that leads to a particular orientation or, at the very least, in close proximity to the optimal orientation within the lattice structure. According to his study, if the last agent in the matching process is a man agent, a man-optimal or fairly stable matching is obtained that is close to the man-optimal orientation in the lattice structure. In the opposite direction, if the last agent in the matching process is a woman, it will obtain a woman-optimal or fair stable matching close to the woman-optimal orientation in the lattice structure. This allows us to control the orientation of the stable matching that we want to produce by sorting the *queueMember* variable in Algorithm 4. By simply putting all men agents at the end of the queue, we will be able to obtain the man-optimal matching, or we can put all the women agents at the end of the queue if we want woman-optimal matching.

3.4 Time Complexity

This section discusses the computational cost of finding stable matching. In this study, we modify the Roth and Vande (RV) mechanism [25] to find stable matching by updating the previous matching. The RV mechanism employs the Gale-Shapley algorithm, which requires $O(n^2)$ to find a stable matching. Thus, The RV mechanism requires $O(n \cdot n^2)$ or $O(n^3)$ to find a stable matching. Our updating mechanism is the modified version of the RV mechanism. The RV system uses a single pair to start the matching process, whereas our updating mechanism can use several pairs for the first matching procedure. The number of pairs that we use for the initial matching process is based on the size of the reduced matching (r) from the previous matching. Therefore, our updating mechanism requires $O((n - r + 1) \cdot n^2)$. In the worst-case scenario, for instance, if the reduced matching $r = 1$, our updating mechanism requires $O(n \cdot n^2)$, equivalent to the original Roth and Vande mechanism.

Table 3.1 Comparison of short-term stable matching algorithm. n denotes the number of men (or women), r denotes the size of the reduced instance from the previous matching

Algorithm	Time complexity	Re-matching cost
Gale-Shapley	$O(n^2)$	n
Roth and Vande	$O(n^3)$	$n - 1$
Proposed method	$O((n - r + 1) \cdot n^2)$	$n - r$

3.5 Summary

This chapter proposes an algorithm for finding stable matching in the SMP with dynamic preference. In the SMP with dynamic preference, the agents' preference might change dynamically, affecting the stability of the previous stable matching. One traditional method for maintaining matching stability is to perform new matching every time the preference changes. However, in some situations, preference changes do not always affect the matching stability, such

as with minor changes in preference. In other words, the prior matching remains stable in the presence of the new preference. This motivates us to update the matching to find a stable matching with dynamic preferences. Compared to other existing methods, our proposed concept is outperform in minimizing re-matching costs.

Our main contributions in this chapter are Theorems 3.1 and 3.2, which demonstrate how to update the matching when the previous preference changes. Thus, we can minimize the re-matching cost when performing the new matching. In this chapter, we assume that preference changes rarely occur, employing the short-term strategy may be costly when the preference changes frequently occur.

In the next chapter, we will discuss a long-term strategy to find a stable matching under dynamic preference, when preference changes frequently occur.

Algorithm 4: Updating Stable Matching

Input :

- m = man agent, w = woman agent
- current instance: $P(m)$ = men pref.,$P(w)$ = women pref.
- previous instance: $P_0(m)$ = men pref., $P_0(w)$ = women pref.
- SM_0 = Previous Stable Matching

Def `updateMatching`**:**

```
for m,w in SM_0 do
  if P(m) != P_0(m) then
    for w_0 in P_0(m) do
      if rank(w) in P_0(m) > rank(SM_0(w_0)) in P_0(m) then
        assign w_0 to PotentialBP
      end if
    end for
    for w in P(m) do
      if w in PotentialBP then
        if rank(m) in P(w) > rank(SM_0(w_0)) in P(w) then
          remove pair (m, w) from SM_0
        end if
      end if
    end for
  end if
  if P(w) != P_0(w) then
    for m_0 in P_0(w) do
      if rank(m) in P_0(w) > rank(SM_0(m_0)) in P_0(w) then
        assign m_0 to PotentialBP
      end if
    end for
    for m in P(w) do
      if m in PotentialBP then
        if rank(w) in P(m) > rank(SM_0(m_0)) in P(m) then
          remove pair (m, w) from SM_0
        end if
      end if
    end for
  end if
end for
SM ← SM_0\removedpair
roomMember ← SM_0
queueMember ← removedpair
for newMember in queueMember do
  assign newMember to roomMember
  pathToStability(newMember)
end for
```

Chapter 4

Long-term Stability for Matching Problem under Dynamic Preference

4.1 Introduction

In the short-term stability strategy, we assume that preference changes rarely occur. We propose an update matching mechanism as a short-term strategy to address the problem. In this chapter, we assume that preference changes frequently occur, and employing a short-term strategy will be costly. This chapter introduces a new concept of finding stable matching under dynamic preference and assuming the preference changes frequently occur.

This chapter proposes a long-term stability strategy for a matching problem under dynamic preference. Several concepts of stability for the matching problem with dynamic preference have been published. References [24, 11, 8, 7] use the most stable approach to find stable matching under dynamic preference. The most stable approach means finding a matching that is most stable to all instances of a dynamic instance. Chen et al. [11] introduce the α-*layer global stability* to define the stability for a matching problem under dynamic

preference by extending the original stability concept of SMP. They use α to quantify the strengths of the stability with $1 \leq \alpha \leq \ell$, where ℓ is the number of layers, which is similar to the number of instances in a dynamic instance in our current study. Our *fully stable* notions is equivalent with $\ell-$*layer globally stable*. Aziz et al. [8] defined the *certain stable* and *possibly stable* concept when a probability of preference is given. The *certain stable* concept is also equivalent to our *fully stable* notions of stability.

A classical SMP instance is $I = (M, W, L)$. In the SMP with dynamic preference, agents can change their preferences, leading to dynamic preference. An instance of SMP with dynamic preference leading the formation of a dynamic instance. The dynamic instance is $DI = (M, W, L_1, L_2, \ldots, L_k)$, where k is the number of unique preference lists that occur due to changes in agent preferences. Thus, a set of SMP instances for SMP under dynamic preference is $DI = \{I_1, I_2, \ldots, I_k\}$.

The main contribution of this chapter is summarized as follows: 1. We introduce a new concept to find a stable matching under dynamic preference. 2. We introduce the stability notions for stable matching under dynamic preference.

4.1.1 Problems

In this section, we will examine the issues that will be addressed in this chapter. This chapter discusses the strategy to maintain the long-term stability of a matching. Some existing method is proposed to find stable matching for long period. However, in some cases, it is still hard to select the solution for stable matching under dynamic preference. Here is a summary of the problems we intend to address in this chapter:

1. How to find the long-term stable matching

2. How to select the matching solution more accurately

The two issues listed above are the issues that will be addressed in this chapter. In the first issue, the *most stable* matching concept is the existing concept used by other researchers [24, 7, 11] to define stable matching under dynamic preference. This concept counts the number of stability of matching against all occurred instances. They use the α as the index of strengths of provided solutions. The matching with the highest index is selected as a solution for stable matching. However, in some cases, some matching has an equal value of the index, this situation is difficult to determine the solution.

The second issue is how to determine the stable matching solution more accurately. In this study, we propose a new concept to define stable matching using the blocking pair perspective. Using the blocking pair, we can get the new information to select the stable matching more precisely. In this study, we also introduce two indexes to select stable matching solutions using the blocking pair perspective. The first index is β which represents the number of blocking pairs of a matching in all instances. The second index is EV, which represents the expected value of blocking pairs of a matching in all instances. The matching with the lowest β and EV is the solution for the stable matching problems.

4.1.2 Most stable matching concept

Agent preference changes are very likely to occur in real-world situations, and this could be due to a lack of information from agents about the opposite sex. The preference changes that continue to occur will form a probability distribution of preferences. Before discussing our proposed concept, we summarize the concept of finding stable matching using the most stable concept. Several studies [7, 24, 11, 8] employ the most stable concept to find the stable matching under dynamic preference. To illustrate the problem, we use a 3x3 SMP instance as shown in Example 4.1.

	$m_1 = w_1, w_2, w_3$	$w_1 = m_2, m_1, m_3$
L_1	$m_2 = w_2, w_3, w_1$	$w_2 = m_3, m_2, m_1$
	$m_3 = w_3, w_1, w_2$	$w_3 = m_1, m_3, m_2$
	$m_1 = w_2, w_3, w_1$	$w_1 = m_2, m_1, m_3$
L_2	$m_2 = w_2, w_3, w_1$	$w_2 = m_3, m_2, m_1$
	$m_3 = w_3, w_1, w_2$	$w_3 = m_1, m_3, m_2$

Fig. 4.1 The 3x3 SMP instances with dynamic preference.

Example 4.1. *Given an SMP instance under dynamic preference. A set of men* $M = m_1, m_2, m_3$ *and women* $W = w_1, w_2, w_3$*. The two sets of preference,* L_1 *and* L_2*, are depicted in Figure 4.1.*

The preferences in Figure 4.1 imply the occurrence of two SMP instances, such that $DI = I_1, I_2$ where $I_1 = (M, W, L_1)$ and $I_2 = (M, W, L_2)$. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be the stable matching sets for I_1 and I_2, respectively. This setting admits three unique matching with positive probability: $\mu_1 = \{w_1 : m_1, w_2 : m_2, w_3 : m_3\}$, $\mu_2 = \{w_1 : m_2, w_2 : m_3, w_3 : m_1\}$, and $\mu_3 = \{w_1 : m_3, w_2 : m_2, w_3 : m_1\}$. μ_1 and μ_2 stable against I_1, such that $\mathcal{M}_1 = \{\mu_1, \mu_2\}$. Whereas I_2 admits μ_2 and μ_3 as the stable matchings, such that $\mathcal{M}_2 = \{\mu_2, \mu_3\}$. To find the most stable matching, we need to find the α value for each matching. α shows the stability strength of each matching against a dynamic instance; α is a function that contains the probability of stability for each matching against a dynamic instance.

$$\alpha(\mu) = \sum_{i=1}^{k} SM_i(\mu) \tag{4.1}$$

Where:

- $\alpha(\mu)$ = Number of matching μ being stable in a dynamic instance

- DI = a set of dynamic instance, such that $I_1, I_2, \ldots, I_k$
- $SM_i(\mu) = x \mid x = 1$ if μ stable to I_i, otherwise $x = 0$
- k = cardinality of dynamic instance DI

Using (4.1), we get that α for μ_1, μ_2, and μ_3 are 1, 2, and 1, respectively. The next step is finding the matching with the highest α value. In Example 4.1, the α for matching μ_1 and μ_3 is 1, whereas μ_2 is $2 = k$. We can simply decide that μ_2 is the stable matching for the current matching problem.

We summarize the algorithm to find the stable matching using the most stable concept, as follows:

1. Find all stable matching for each instance
2. Check the stability of each stable matching found in all instances
3. Calculate the α of each matching, eq (4.1)
4. Find the matching with the highest α

The Gale-Shapley algorithm cannot generate all stable matching of an SMP instance because it only generates the optimal matching (man- or woman-optimal). Currently, the most efficient algorithm to find all stable matching solutions for a single instance is brute force [13]. Wirth's method [27] of trial-and-error and backtracking is a straightforward but inefficient approach to discovering all solutions of stable matching. Algorithm 5 shows how to find a stable matching under dynamic preference using the most stable concept.

4.1.3 Probability Distribution of Instances

This section considers a preference that changes frequently and repeatedly. Assuming the probability of an instance is given, we can determine the probability of stability of matching against a dynamic instance. In the most stable

Algorithm 5: Finding the most stable matching

Input: k = number of instances in dynamic instance
n = size of SMP

```
1: SM_DI = array() //matchings in dynamic instance
2: for i = 1 to k do
3:    allSM(I_i) //find all stable matching of instance i
4:    SM_DI.append(allSM(I_i))
5: end for
6: for each sm in unique(SM_DI) do
7:    α ← 0
8:    for i = 1 to k do
9:       checkStability(sm,i) //check stability of sm in i
10:      if (sm is stable) then
11:         α ← α + 1
12:      end if
13:   end for
14:   mostStable[sm] ← α
15: end for
16: SORT_DESC(mostStable[sm])
```

matching concept, [8] considers the probability of stability of matching to determine the stable matching with uncertain preference.

We can perform a deeper search based on the preference probability distribution to get the most stable matching. Using the Bayesian formula, we can get the probability of the occurrence of a stable matching against each instance of the dynamic instance.

$$P(\mu) = \sum_{i=1}^{k} P(I_i \cap \mu) = \sum_{i=1}^{k} P(I_i)P(\mu|I_i) \tag{4.2}$$

Where:

- $P(I_i)$ = Probability of instance i
- $P(\mu|I_i)$ = Probability of matching μ in instance i
- k is the number of instances in the dynamic instance

Recalling the Example 4.1, we set the probability of each instance as depicted in Figure 4.2

	$m_1 = w_1, w_2, w_3$	$w_1 = m_2, m_1, m_3$	
L_1	$m_2 = w_2, w_3, w_1$	$w_2 = m_3, m_2, m_1$	$Pr=0.4$
	$m_3 = w_3, w_1, w_2$	$w_3 = m_1, m_3, m_2$	
	$m_1 = w_2, w_3, w_1$	$w_1 = m_2, m_1, m_3$	
L_2	$m_2 = w_2, w_3, w_1$	$w_2 = m_3, m_2, m_1$	$Pr=0.6$
	$m_3 = w_3, w_1, w_2$	$w_3 = m_1, m_3, m_2$	

Fig. 4.2 Probability distribution of 3x3 SMP dynamic instance

Using the Bayesian formula (4.2), we can find the probability of each stable matching relative to a dynamic instance. $P(\mu_1)$, $P(\mu_2)$, and $P(\mu_3)$ are 0.2, 1, and 0.3, respectively. Since μ_2 is stable in all instances, μ_2 still has the highest probability of being stable.

4.2 The Least Blocking Pair Concept

We propose a new concept to find long-term stable matching. In Section 4.1.2, we find the stable matching under dynamic preference based on the "stable matching" perspective by counting the number of instances that the matching can be stable. This section intends to find stable matching under dynamic preference from another perspective. We want to find the stable matching using the blocking pair, which means we intend to find the stable matching using the "unstable matching" perspective. A matching is unstable if at least one blocking pair is found. In unstable matching, the number of blocking pairs is ≥ 1. Whereas in stable matching, the number of blocking pairs is *zero*. The following example motivates us to use the blocking pair approach to find stable matching under dynamic preference.

Example 4.2. *Given the 3x3 SMP instance* $I = (M, W, L)$ *as follows:*

$$L(m_1) = w_3, w_2, w_1 \qquad L(w_1) = m_3, m_1, m_2$$
$$L(m_2) = w_1, w_3, w_2 \qquad L(w_2) = m_3, m_2, m_1$$
$$L(m_3) = w_3, w_1, w_2 \qquad L(w_3) = m_2, m_3, m_1$$

Given $\mathcal{M}_1 = \{(w_1 : m_1), (w_2 : m_3), (w_3 : m_2)\}$ and $\mathcal{M}_2 = \{(w_1 : m_2), (w_2 : m_3), (w_3 : m_1)\}$. Both $\mathcal{M}_1$ and $\mathcal{M}_2$ are unstable to instance I. However, we want to know the best matching among the two worst choices. Recall the definition of stability of the stable marriage problem; a matching is stable if no blocking pair is found. Our proposed concept is to try to find the best among the worsts.

Table 4.1 Quantifying the number of blocking pairs for each matching

Matching	#BP	Description
$\mathcal{M}_1$	1	$BP = \{(m_3, w_1)\}$
$\mathcal{M}_2$	2	$BP = \{(m_3, w_1), (m_3, w_3)\}$

We quantify the number of blocking pairs for each matching to determine the best matching as shown in Table 4.1. Matching with the minimum number of blocking pairs is the best matching. In Example 4.2, we found that a pair (m_3, w_1) is the blocking pair in $\mathcal{M}_1$. Whereas (m_3, w_1) and (m_3, w_3) are the blocking pairs in $\mathcal{M}_2$. Because the number of blocking pairs in $\mathcal{M}_1$ is less than the number of blocking pairs in $\mathcal{M}_2$, $\mathcal{M}_1$ is better than $\mathcal{M}_2$ in the instance I. This motivates us to use the blocking pair as a perspective to find stable matching in the stable matching problem under dynamic preference.

Example 4.2 shows how to find the best matching among the worst choices in a single instance by looking for matching with the lowest number of blocking pairs. We also apply this concept to find the matching with the minimum blocking pairs in the dynamic instance. To determine the best matching on a dynamic instance, we calculate the β value of each matching μ. β is the cardinality of distinct blocking pairs in a dynamic instance and indicates the number of the blocking pairs for matching in a dynamic instance. The blocking

pair (BP) of matching μ in dynamic instance DI is defined as follows:

$$BP_{DI}(\mu) = \cup_{i=1}^{k} BP_i(\mu) = \{BP_1(\mu) \cup \cdots \cup BP_k(\mu)\} \tag{4.3}$$

$$\beta(\mu) = |\cup_{i=1}^{k} BP_i(\mu)| \tag{4.4}$$

where:

- BP_{DI} = consists of the union of blocking pairs in instance i.
- $\beta(\mu)$ = The number of blocking pairs in a dynamic instance.

Matching with the smallest β is the most stable among the other matchings. Algorithm 6 shows how to find a stable matching under dynamic preference using the least blocking pair concept.

Consider the following Example 4.3 to understand how to find the stable matching problem under dynamic preference using the blocking pair perspective.

Example 4.3. *Given the 3x3 SMP instance $I = (M, W, L)$, suppose agent m_3 changes his preference, and the preferences are as follows:*

***Instance 1* (I_1)**

$L_1(m_1) = w_3, w_2, w_1$	$L_1(w_1) = m_3, m_1, m_2$
$L_1(m_2) = w_1, w_3, w_2$	$L_1(w_2) = m_3, m_2, m_1$
$L_1(m_3) = w_3, w_1, w_2$	$L_1(w_3) = m_2, m_3, m_1$

***Instance 2* (I_2)**

$L_2(m_1) = w_3, w_2, w_1$	$L_2(w_1) = m_3, m_1, m_2$
$L_2(m_2) = w_1, w_3, w_2$	$L_2(w_2) = m_3, m_2, m_1$
$L_2(m_3) = w_2, w_3, w_1$	$L_2(w_3) = m_2, m_3, m_1$

This setting admits four unique matching with positive probability: $\mu_1 = \{w_1 : m_2, w_2 : m_1, w_3 : m_3\}$, $\mu_2 = \{w_1 : m_3, w_2 : m_1, w_3 : m_2\}$, $\mu_3 = \{w_1 :$

$m_1, w_2 : m_3, w_3 : m_2\}$, and $\mu_4 = \{w_1 : m_2, w_2 : m_3, w_3 : m_1\}$. μ_1 and μ_2 stable against I_1, such that $\mathcal{M}_1 = \{\mu_1, \mu_2\}$. Whereas I_2 admits μ_3 and μ_4 as the stable matchings, such that $\mathcal{M}_2 = \{\mu_3, \mu_4\}$.

Table 4.2 Find β of each matching

Matching	β	Description
μ_1	1	$BP = \{(w_2, m_3)\}$
μ_2	1	$BP = \{(w_2, m_3)\}$
μ_3	1	$BP = \{(w_1, m_3)\}$
μ_4	2	$BP = \{(w_1, m_3), (w_3, m_3)\}$

There are no obtained matching in Example 4.3 admits stable to the dynamic instance. Each matching only stable to a single instance. Our proposed concept can help determine the stable matching among the worst options. Table 4.2 shows β of each matching. Using obtained β, we can eliminate μ_4 because it has the most blocking pairs. We intend to find the stable matching with least blocking pair. Thus, the solution for Example 4.3 is μ_1, μ_2, and μ_3. For temporary solution, we can select randomly from three provided options. Our next discussion discuss how to find the specific stable matching when the probability of instance is given.

4.2.1 The Expected Value of The Blocking Pairs

In the stable matching problem with dynamic preference, the number of instances that appear is more than one, where each instance has a probability of occurrence. In this study, we also consider the probability of instances occurring. Therefore, we want to find the stable matching by finding the stable matching that has the minimum expected value of the blocking pair. To find the expected value of the blocking pair for each matching, we need to quantify the number of blocking pairs for each matching, and we calculate the expected

Algorithm 6: Finding the least blocking pairs

Input: k = number of instances in dynamic instance
n = size of SMP

```
1: SM_DI = array() //matchings in dynamic instance
2: for i = 1 to k do
3:    allSM(I_i) //find all stable matching of instance i
4:    SM_DI.append(allSM(I_i))
5: end for
6: for each sm in unique(SM_DI) do
7:    for i = 1 to k do
8:       checkStability(sm,i) //check stability of sm in i
9:       if (sm is unstable) then
10:         BP.append(BP_smi)
11:      end if
12:   end for
13:   BP_smDI.append(BP)
14:   cBP_smDI ← count.unique(BP_smDI)
15: end for
16: SORT_ASC(cBP_smDI)
```

value of blocking for each matching using the following (4.5).

$$EV(\mu) = \sum_{i=1}^{k} \#(BP(\mu)|I_i)P(I_i) \tag{4.5}$$

where:

- $EV(\mu)$ = Expected value of the number of blocking pairs of matching μ in dynamic instance.
- $\#(BP(\mu)|I_i)$ = number of blocking pairs in μ in instance i
- $P(I_i)$ = Probability of Instance i
- k = number of instances in dynamic instance

Now we summarize the algorithm to find the stable matching by finding the expected value of the blocking pair, as follows:

1. Find all stable matching for each instance

2. For each instance, find the blocking pairs in the dynamic instance
3. Calculate the expected value of the blocking pairs (EV) of each matching, eq. (4.5)
4. Find the matching with the minimum expected value of the blocking pair

Example 4.4. *Consider the following example of the stable matching problem under dynamic preference, depicted in Figure 4.3.*

L_1	$m_1 = w_3, w_2, w_1$ $m_2 = w_1, w_3, w_2$ $m_3 = w_3, w_1, w_2$	$w_1 = m_3, m_1, m_2$ $w_2 = m_3, m_2, m_1$ $w_3 = m_2, m_3, m_1$	$Pr=0.4$
L_2	$m_1 = w_3, w_2, w_1$ $m_2 = w_1, w_3, w_2$ $m_3 = w_2, w_3, w_1$	$w_1 = m_3, m_1, m_2$ $w_2 = m_3, m_2, m_1$ $w_3 = m_2, m_3, m_1$	$Pr=0.6$

Fig. 4.3 The 3x3 SMP instances with dynamic preferences with a probability of instance.

This setting admits four unique matching with positive probability: $\mu_1 = \{w_1 : m_2, w_2 : m_1, w_3 : m_3\}$, $\mu_2 = \{w_1 : m_3, w_2 : m_1, w_3 : m_2\}$, $\mu_3 = \{w_1 : m_1, w_2 : m_3, w_3 : m_2\}$, and $\mu_4 = \{w_1 : m_2, w_2 : m_3, w_3 : m_1\}$. μ_1 and μ_2 stable against I_1, such that $\mathcal{M}_1 = \{\mu_1, \mu_2\}$. Whereas I_2 admits μ_3 and μ_4 as the stable matchings, such that $\mathcal{M}_2 = \{\mu_3, \mu_4\}$. There is no stable matching that can be stable to all instances. The next step is to quantify the number of blocking pairs for each matching, as shown in Table 4.3

Now, the BP expected value for each matching can be found using eq. 4.5. The expected value of the blocking pair for μ_1, μ_2, μ_3, and μ_4 are 0.6, 0.6, 0.4, and 0.8, respectively. Because the matching with the minimum expected value of the blocking pair is μ_3, it is the stable matching for the current stable

Table 4.3 Quantifying the number of blocking pairs for each matching

Matching	#BP|I_1	#BP|I_2	Description
μ_1	0	1	Stable in I_1
μ_2	0	1	Stable in I_1
μ_3	1	0	Stable in I_2
μ_4	2	0	Stable in I_2

matching problem. If the previous mechanism is used in finding the α for each matching, all the corresponding α values will be 1.

4.3 Stability Notions for Matching Problem Under Dynamic Preference

In a stable matching problem with dynamic preference, it is difficult to satisfy the concept of stability of a classical SMP, where blocking pairs are not allowed in a matching. In the stable matching problem with dynamic preference, the dimensions of the instance become more expansive, we need to consider multiple instances' stability. A matching might be stable in one instance but unstable in other instances. A matching with *fully stable* character is a matching that can satisfy the classical stability concept since *fully stable* admits no blocking pair in a dynamic instance. A *fully stable* means matching with $\alpha = k$, where k is the number of instances in a dynamic instance; Or also means matching with $\beta = 0$.

In classical SMP, the definition of stability is determined by the existence of a blocking pair in a matching. If one blocking pair is found in a matching μ, matching μ is unstable. Otherwise, it is stable. an instance of classic SMP is $I = (M, W, L)$, and the matching stability is tied only to a single instance. In this study, we discuss a stable matching with dynamic preference. An instance of an SMP with dynamic preference is $DI = (M, W, L_1, L_2, \ldots, L_k)$. Thus, we can write $DI = \{I_1, I_2, \ldots, I_k\}$. Stable matching under dynamic preference considers not only the stability of matching on a single instance but also

the stability of matching against multiple instances. As discussed earlier, we discussed two approaches to find a stable matching under dynamic preference in this study. The existing approach is to find the most stable matching to the dynamic instance, and our proposed approach is to find a matching with the minimum blocking pairs in a dynamic instance.

In the *most stable* concept, as explained in Chapter 4.1.2, we calculate the value of α as an indication of the strength of matching in a dynamic instance. To find the α, we count the number of stability for each matching against all instances in the dynamic instance. In quantifying the number of stability, we refer to the concept of classic SMP, where matching is unstable if at least one blocking pair is found. To determine the α value of each matching, we count the number of stable matching to all instances. In our proposed approach, as discussed in Chapter 4.2, we calculate the number of blocking pairs for each matching in a dynamic instance.

The most stable concept and least blocking pair concept (our proposed concept) depends on the existence of a blocking pair to determine the stable matching. In the *most stable* concept, the blocking pair is used to determine the stability for each instance to find the α. Whereas for the least blocking pair concept, we quantify the number of blocking pairs to determine the β.

Given an SMP instance, $I = (M, W, L)$ and matching μ, a matching μ is stable if the number of blocking pairs is zero. For the SMP under dynamic preference, the instance is $DI = (M, W, L_1, L_2, \ldots, L_k)$ or $DI = \{I_1, I_2, \ldots, I_k\}$. We have the probability distribution of preference in the SMP with dynamic preference. In this study, we define the stability of matching for the stable matching problem under dynamic preference based on the most stable matching and the expected value of the number of blocking pairs. The blocking pair is the key factor for the two approaches we employ. Using matrix notation, we illustrate the relationship between the two approaches for determining stable

matching under dynamic preference.

$$\mu = \begin{matrix} I_1 & I_2 & & I_k \end{matrix} \begin{bmatrix} BP_1 & BP_2 & \cdots & BP_k \\ P_1 & P_2 & \cdots & P_k \end{bmatrix} \tag{4.6}$$

Where:

μ = matching of dynamic instance

I = an instance of dynamic instance

P = Probability of instance occurs

BP = a set of blocking pairs of matching μ in an instance

k = number of instances in dynamic instance

In the most stable matching approach, we determine the best matching by finding the value of α for each obtained matching. For this approach, the matching with the highest α value is a solution for stable matching under dynamic preference. Based on the matrix notation (4.6), we can find the α of matching μ as follow:

$$\alpha(\mu) = \sum_{i=1}^{k} SM_i(\mu) \tag{4.7}$$

where: $SM_i = x|x = 0$ if $|BP_i| > 0$, otherwise $x = 1$.

We intend to find the matching with minimum blocking pairs in the least blocking pair approach. Therefore, we quantify the number of blocking pairs in the dynamic instance by finding the β of matching μ, where β is the cardinality of blocking pairs for matching μ in the dynamic instance DI.

$$\beta(\mu) = |\cup_{i=1}^{k} BP_i(\mu)| \tag{4.8}$$

In this study, we also consider finding the expected value of the blocking pair based on the probability distribution of instances in the dynamic instance.

Based on the matrix notation 4.6, we can find the expected value (EV) of blocking pair of matching μ as follow:

$$EV(\mu) = \sum_{i=1}^{k} \#(BP(\mu)|I_i)P(I_i) \tag{4.9}$$

where: $\#(BP(\mu)|I_i)$= the number of blocking pairs in instance i.

This section discusses the relationship between our concept and the existing concept to find stable matching. We shows that we can find α (4.7), β (4.8) and 4.9 using the matrix notation (4.6). To find the α, it is only considered the matching without any blocking pair in an instance i. For the most stable approach, $1 \leq \alpha \leq k$. When $\alpha = k$, it means a matching is stable to all instances. Whereas $\alpha = 1$ means the matching is only stable in a single instance among a set of dynamic instances. Our proposed concept has the advantage of quantifying the number of blocking pairs. Moreover, considering the probability of instance gives a more detailed result to define the stable matching using the expected value of the blocking pairs.

In contrast to the most stable concept, which counts the number of stable matchings and selects the one with the largest α, our proposed concept counts the number of blocking pairs that cause instability of matching on the dynamic instance. Further, we select a matching with the minimum number of the blocking pair as the solution of the stable matching under dynamic preference. For example, when all provided matchings have the same stability score (α), and none of the matchings has an $\alpha = k$. Our concept offers a solution by choosing the matching with the minimum number of the blocking pair. Example 4.5 shows the merit of our proposed concept compared to the most stable matching concept.

Example 4.5. *Given an SMP instance with dynamic preference. A set of men* $M = m_1, m_2, m_3$ *and women* $W = w_1, w_2, w_3$. *Suppose we have three different preferences* L_1, L_2 *and* L_3.

$$
\begin{array}{lll}
 & m_1 = w_1, w_2, w_3 & w_1 = m_2, m_1, m_3 \\
L_1 & m_2 = w_2, w_3, w_1 & w_2 = m_3, m_2, m_1 \\
 & m_3 = w_3, w_1, w_2 & w_3 = m_1, m_3, m_2 \\
 & & \\
 & m_1 = w_1, w_2, w_3 & w_1 = \boldsymbol{m_1, m_3, m_2} \\
L_2 & m_2 = w_2, w_3, w_1 & w_2 = m_3, m_2, m_1 \\
 & m_3 = w_3, w_1, w_2 & w_3 = m_1, m_3, m_2 \\
 & & \\
 & m_1 = \boldsymbol{w_2, w_3, w_1} & w_1 = \boldsymbol{m_2, m_1, m_3} \\
L_3 & m_2 = w_2, w_3, w_1 & w_2 = m_3, m_2, m_1 \\
 & m_3 = w_3, w_1, w_2 & w_3 = m_1, m_3, m_2
\end{array}
$$

Since we have three different preferences in Example 4.5, the dynamic instance is $DI = \{I_1, I_2, I_3\}$, where $I_1 = (M, W, L_1)$, $I_2 = (M, W, L_2)$ and $I_3 = (M, W, L_3)$. I_1 obtained two stable matchings, $\mu_1 = \{(w_1 : m_1), (w_2 : m_2), (w_3 : m_3)\}$ and $\mu_2 = \{(w_1 : m_2), (w_2 : m_3), (w_3 : m_1)\}$, I_2 obtained two stable matchings, μ_1 and $\mu_3 = \{(w_1 : m_3), (w_2 : m_2), (w_3 : m_1)\}$, and I_3 obtained μ_2 and μ_3. From the dynamic instance DI, we have a set of matching $\mu_{DI} = \{\mu_1, \mu_2, \mu_3\}$. Table 4.4 shows the α and β of each matching respective to a dynamic instance.

Table 4.4 α and β of each matching respective to a dynamic instance

Matching	β	α	Description
μ_1	1	2	$BP = \{(m_1, w_3)\}$
μ_2	2	2	$BP = \{(m_1, w_3), (m_1, w_1)\}$
μ_3	1	2	$BP = \{(m_1, w_1)\}$

The results in Table 4.4 shows $\beta(\mu_1) < \beta(\mu_2)$, then $\alpha(\mu_1) \geq \alpha(\mu_2)$.

4.3.1 Special case of preference: Dynamic preference on one side

This section considers some special preference cases for the stable matching problem under dynamic preference. We will discuss how the special case of our problem can be solved. Consider the special case of the problem where the dynamic preference only occurs on one side. Dynamic preference on one side can arise in real-world scenarios. For example, matching between the cloud service providers and the cloud service users. It is reasonable to assume that the cloud service provider evaluates its potential client by resource usage behavior, which dynamically changes (Thus, having a dynamic preference). In contrast, the cloud service users evaluate the cloud service providers based on the price and SLA, which is stated by providers (static preference).

Given an SMP instance of size n with dynamic preference $I = (M, W, L)$, where $M = \{m_1, m_2, \ldots, m_n\}$ and $W = \{w_1, w_2, \ldots, w_n\}$. Assume a woman agent expresses two different preferences $I = (M, W, L_1, L_2)$, such that $DI = \{I_1, I_2\}$.

Theorem 4.1. *Under the assumption that the men's preference is static. If the first option of each man is distinct, man-optimal matching is fully-stable and always exists*

Proof. If the men's preference is static and the first option of M is distinct, all men agents get their first option by applying man as a proposer (man-optimal). Even if the women change their preference, all men agents can still get their first choice of the woman. Thus, the man-optimal matching is fully-stable and always exists. □

Corollary 4.1.1. *Given n-size SMP with a dynamic preference on one side. If the men's preference is cyclic, man-optimal is fully-stable and always exists*

Proof. On Theorem 4.1, if men's preference has the distinct first option, the fully stable always exists. The cyclic preference also has a similar pattern with

Theorem 4.1, which also has a distinct first preference option. Therefore, the cyclic preference of men will generate man-optimal matching as a fully-stable characteristic of matching. □

Theorem 4.2. *Under the assumption that the men's preference is static. If women's preference is dynamic,* $\alpha(\mu_{m-opt}) \geq \alpha(\mu_{w-opt})$.

Proof. Assume both sides of agents express a distinct first choice of preference, thus, there will be at least two different stable matching, man-optimal stable matching(μ_{m-opt}) and woman-optimal stable matching(μ_{w-opt}). Without loss of generality, assume a woman agent, say w_1 changes her preference in k times. Then we have $L_i = \{L_1, L_2, \ldots, L_k\}$ and k instances, where $i = 1, 2, \ldots, k$. Based on Theorem 4.1, (μ_{m-opt}) remain stable to L_i, however, (μ_{w-opt}) is not necessarily stable to L_i since w_1 changes her preference. Since μ_{m-opt} is obviously stable to L_i and μ_{m-opt} is not necessarily stable to L_i, then $\alpha(\mu_{m-opt}) \geq \alpha(\mu_{w-opt})$. □

When the state in Theorem 4.2 holds, we do not need to find all the stable matching on each instance. By referring to the Theorem 4.2, we only need to use the Gale-Shapley algorithm using a man as a proposer when the men's preferences are static; likewise, we can use a woman as a proposer when women's preferences are static. In addition, if Theorem 4.1 holds, we only need to find a man-optimal stable matching in a single instance, rather than checking all instances.

Theorem 4.3. *Given matching* μ_1 *is stable to instance* I_1. *Under the assumption that the men's preference is static. If agent* w_1 *changes her preference, the possible set of blocking pairs that will appear is* $BP = \{(w_1, M \setminus \mu_1(w_1))\}$.

Proof. Without loss of generality, we assume w_1 expresses m different preferences. Then the dynamic instance is $DI = \{I_1, \ldots, I_m\}$. Let's say matching μ_1 is stable in instance I_1. Assume agent w_1 is paired with agent m_1 in matching μ_1, then $(w_1, m_1) = (w_1, \mu_1(w_1))$, $\mu_1 = \{(w_1, \mu_1(w_1)), (w_2, \mu_1(w_2)), \ldots, (w_n, \mu_1(w_n))\}$.

If preference of w_1 is changed in instance I_2, such that $L_1(w_1) \neq L_2(w_1)$. Then $W' \equiv W \setminus w_1$, such that $W' = \{w_2, \ldots, w_n\}$, and $M' \equiv M \setminus \mu_1(w_1)$, such that $M' = \{\mu_1(w_2), \ldots, \mu_1(w_n)\}$. Then μ_1 may not be stable against I_2. When w_1 changes her preference, the possibilities are as follows:

Case 1: $(\mu_1(w_1) \succ_{w_i} M')$. If $\mu_1(w_1)$ is the first choice of w_1, then w_1 cannot form a blocking pair.

Case 2: $(M' \succ_{w_i} \mu_1(w_1))$ If w_1 prefers other man rather than her current partner, then w_1 could possibly form a blocking pair with the man in M'.

The other woman in W' will not form a blocking pair since they and their partner do not change their preference. □

Corollary 4.3.1. *Given the n-size of SMP with dynamic preference. If the preference of each man (m) in men M is static and women's preference is dynamic, then* β_{max}*= (n × number of dynamic agents) - number of dynamic agents.*

Proof. Based on Theorem 4.3, if agent w_1 has the dynamic preference, the possible blocking pairs in matching μ is the combination of w_1 and the member of M, (w_1, M). The maximum combination is the number of dynamic agents multiplied by the members of the opposite side. The current pair (w_1, $\mu(w_1)$) cannot be a blocking pair of itself. Thus, the maximum number of blocking pairs is

β_{max}= (n × number of dynamic agents) - number of dynamic agents. □

4.3.2 Relation of Dynamic Preference and Stable Matching With Ties

In stable matching research, many extensions have been discussed [19]. One variant of the stable matching problem that is widely known is Stable Matching with Ties (SMT) [17]. Under certain conditions, the stable matching problem under dynamic preference can form a preference with ties. In SMT, an agent

is allowed to express two or more agents with equal positions in the agent's preference. In other words, SMT is a restriction of the stable matching problem under dynamic preference. If agent w_x and w_y are in the same position of m_z's preference, we write $w_x =_{m_z} w_y$. In SMT, there are three notations of stability: weakly stable, strongly stable, and super stable matching. In the weakly stable matching, blocking pair of matching μ is defined as a pair (m, w) such that $\mu(m) \neq w$, $w \succ_m \mu(m)$ and $m \succ_w \mu(w)$. In the strongly stable matching, (x, y) is a blocking pair of matching μ if $\mu(x) \neq y$, $y \succ_x \mu(x)$ and $x \succcurlyeq_y \mu(y)$. Finally, for super stable matching, (m, w) is said to be a blocking pair of matching μ if $\mu(m) \neq w$, $w \succcurlyeq_m \mu(m)$ and $m \succcurlyeq_w \mu(w)$. Consider the following example.

Example 4.6. *Given 3x3 SMP instance with tie*

$$L(m_1) = (w_3, w_2), w_1 \qquad L(w_1) = m_3, m_1, m_2$$
$$L(m_2) = w_1, w_3, w_2 \qquad L(w_2) = m_3, m_2, m_1$$
$$L(m_3) = w_3, w_1, w_2 \qquad L(w_3) = m_2, m_3, m_1$$

In Example 4.6, m_1 expresses the preference with tie, where w_3 and w_2 m_1 are in the same position. We can break down these preference settings into the stable matching problem under dynamic preference, and we get two SMP instances, as shown in Figure 4.4. Each instance has a probability distribution in a stable matching problem with dynamic preference. Worth to noting the SMT stated that an agent might express two or more agents with an equal position in his/her preference. This means for each instance must have an equal probability. In Example 4.6, I_1 and I_2 must have the same probability of occurrence, where the probability $I_1 = I_2 = 0.5$.

Theorem 4.4. *Fully stable of stable matching with dynamic preference is strongly stable of stable matching with ties*

Proof. A matching μ in stable matching under dynamic preference is fully stable unless found a couple (x, y) such that $\mu(x) \neq y$, $y \succ_x \mu(x)$ and $x \succ_y \mu(y)$

Instance 1 (I_1)

$L_1(m_1) = \mathbf{w_3, w_2, w_1}$ $\quad$ $L_1(w_1) = m_3, m_1, m_2$

$L_1(m_2) = w_1, w_3, w_2$ $\quad$ $L_1(w_2) = m_3, m_2, m_1$

$L_1(m_3) = w_3, w_1, w_2$ $\quad$ $L_1(w_3) = m_2, m_3, m_1$

Instance 2 (I_2)

$L_2(m_1) = \mathbf{w_2, w_3, w_1}$ $\quad$ $L_2(w_1) = m_3, m_1, m_2$

$L_2(m_2) = w_1, w_3, w_2$ $\quad$ $L_2(w_2) = m_3, m_2, m_1$

$L_2(m_3) = w_3, w_1, w_2$ $\quad$ $L_2(w_3) = m_2, m_3, m_1$

Fig. 4.4 Transformation of SMP with a tie to SMP under dynamic preference

to $\exists I \in DI$.

Consider the definition of strongly stable in stable matching with ties and fully stable in stable matching under dynamic preference. The blocking pair of strongly stable matching is (x, y) such that $\mu(x) \neq y$, $y \succ_x \mu(x)$ and $x \succcurlyeq_y \mu(y)$. Or (x, y) such that $\mu(x) \neq y$, $y \succ_x \mu(x)$ and $x \succ_y \mu(y)$ or $x =_y \mu(y)$.

Based on the definitions of both stability notions, the Theorem statement is true. □

Theorem 4.5. *Given an SMP Instance $I = (M, W, L)$ with stable matching μ. Suppose agent m expresses dynamic preference and form a tie. If $\mu(m)$ is not tied, then strongly stable exists.*

Proof. Without loss of generality, suppose agent m_1 changes his preference, we have a dynamic preference due to m_1 changes his preference such that $DI = (M, W, L_1, L_2)$, where $L_1 \neq L_2$. If $\mu(m_1)$ is not tied, there are two possibilities of m_1's new preference.

Case 1: The women who are better than $\mu(m_1)$ in m_1's preference, say W', form a tie. This condition will not form a blocking pair because W' prefers other men over m_1.

Case 2: The women who are worst than $\mu(m_1)$ in m_1's preference, say W",

form a tie. This condition will not form a blocking pair because m_1 prefers $\mu(m_1)$ over W". □

4.4 The Generalization to Many-to-one Model

In this study, we also try to generalize the one-to-one model to the many-to-one model. Hospitals/ Residents Problem (HRP) is a generalization of SMP. An instance of classical HRP $I = (H, R, C, L)$ is a set of residents $R = r_1, r_2, \ldots, r_n$ and hospital $H = h_1, h_2, \ldots, h_m$. Each hospital $h_j \in H$ has a positive integer of capacity value denoted by $C(h_j)$, where $C(h_j) \geq 1$. Each resident $r_i \in R$ has a set of preference list L where residents will rank each member of H in strict order. In the HRP with dynamic preference, each agent also can change their preference and lead to the dynamic preference. An instance of HRP with dynamic preference also leads to the formation of a dynamic instance. The instance of HRP with dynamic preference is $DI = (H, R, C, L_1, L_2, \ldots, L_k)$, where k is the number of unique preference lists that occur due to changes in agent preferences. Thus, a set of SMP instances for SMP under dynamic preference is $DI = \{I_1, I_2, \ldots, I_k\}$.

In line with SMP, to solve HRP with dynamic preferences, we can use the most stable matching concept by finding index α, and also by using the least blocking pair concept by finding β and EV. To find α, we can refer to section 4.1.2. Whereas to find the β and EV, we can refers to section 4.2 and 4.2.1.

4.5 Time complexity

This section discusses the computational cost of finding stable matching under dynamic preference. Algorithm 5 discovers stable matching under dynamic preference using the most stable matching approach. There are two main loops in Algorithm 5. The first is a single loop used to discover all stable matchings of each instance. Within the loop is a function that identifies all stable matchings

for every instance i. The brute-force technique was utilized to discover all stable matching on a given instance. Function $allSM(I_i)$ is used to discover all stable matching of each instance i, which requires $O(n!)$ to find all combinations and $O(n^2)$ time to verify whether each matching is stable against an instance i. Therefore, the time required to complete the first loop is $O(n! \cdot n^2)$. The second loop verifies the stability of a found matching over k instances in which the *checkStability* function is used to check the stability of a matching in an instance, requiring $O(n^2)$ time [14]. Assuming the maximum number of unique stable matchings is $n!$, the time required to check all matchings and determine the α value is $O(k \cdot n! \cdot n^2)$. The matching with the highest α value is selected in the final step. Hence, the time required to find a stable matching with the most stable approach is $O(k \cdot n! \cdot n^2)$.

Table 4.5 Comparison of the long-term stable matching algorithm. n denotes the number of men (or women), k denotes the number of instances arise in the dynamic instance.

Algorithm	Time complexity
Most stable matching	$O(k \cdot n! \cdot n^2)$
Least blocking pair	$O(k \cdot n! \cdot n^2)$

Following the most stable matching concept, the least blocking pair concept (Algorithm 6) also requires $O(k \cdot n! \cdot n^2)$ time to find a stable matching. The cost of achieving stable matching is extremely high due to the necessity of finding all stable matchings for each instance, regardless of whether the most stable matching or least blocking pair concept is employed. However, in the special case preference outlined in Section 4.3.1, a stable matching can be found in less time. In Theorem 4.1 conditions, the Gale-Shapley algorithm is only employed once; hence the time required to discover a stable matching solution is $O(n^2)$. Meanwhile, the cost to find a stable matching that satisfies the constraints of Theorem 4.2 is $O(k \cdot n^2)$. The constraints in Theorem 4.2 employ the Gale-Shapley algorithm in k times, where k represents the number of instances contained in the dynamic instance.

4.6 Summary

This chapter introduces a new concept for finding stable matching with dynamic preference. We propose β and EV as the index to define the number of blocking pairs in the stable matching under dynamic preference. In several studies, the most stable concept is widely used to find stable matching under dynamic preference, which finds a matching that is the most stable for all occurred instances. Existing studies use α to indicate the strengths of stable matching under dynamic preference.

The most stable concept and our proposed concept are similar in determining stable matching. Both concepts observe the presence of blocking pairs in a matching. In the most stable concept, a matching is unstable if at least one blocking pair is found. This concept ignores the diversity of blocking pairs, whether a matching has the same or a different number of blocking pairs. A matching remains unstable if a blocking pair is found. For the computational cost, both concepts require the same cost $O(k \cdot n! \cdot n^2)$.

Considering the number of blocking pairs in the stable matching under dynamic preference is a novel concept. By understanding the amount of blocking pairs in detail, it is possible to obtain additional information regarding stable matching. Consequently, the information obtained from the number of blocking pairs can assist in determining stable matching with greater precision than the existing method, In other words, the usage of the index (α, β, and EV) is better than α.

[This page is intentionally left blank]

Chapter 5

Applications of Stable Matching under Dynamic Preference

5.1 Introduction

The stable matching problem has been implemented extensively to solve real-world problems. For example, the hospitals/residents problem variant is employed to assign residents (the intern medical students) to hospitals NRMP, JRMP, etc. Several studies have been conducted regarding implementing a stable matching problem for computer applications. References [28, 20] use stable matching to migrate virtual machines (VMs) between servers in the data center. The objective of migrating virtual machines is to improve energy efficiency in the data center while maintaining the virtual machines' quality of service. References [10, 1, 2] use stable matching to deploy containers on the server by implementing the hospitals/ residents problem to improve the power efficiency of servers. The Akamai engineers use a stable marriage problem to assign users to servers in content delivery networks [22], the stable matching algorithm helps to balance the loads within server clusters. However, all mentioned references do not consider when agents' preference changes. For example, the resource usage of a VM at different times, such as during the day

and night, might be different. In this study, a stable matching problem with dynamic preferences is applied to the job scheduling of containers and servers.

5.1.1 Stable Matching and Optimization

Stable matching and optimization are essential concepts in numerous fields, but they do not compete with one another. A stable matching is a concept that focuses on finding a matching between two sets of agents that is stable, meaning no agent prefers to be paired with each other over their current partner. On the other hand, optimization refers to the process of locating the optimal solution to a problem within the confines of a set of constraints. It is utilized in many different industries, including engineering, economics, and finance, and can entail either the maximization or minimization of an objective function that is subject to restrictions.

Stable matching can be viewed as a sort of optimization in certain circumstances, where the objective is to discover the optimal pairing that satisfies specific requirements (such as stability). However, stable matching is frequently more concerned with fairness and stability than with optimization, and there may be instances in which an ideal solution is neither stable nor optimal. In conclusion, both stable matching and optimization are significant notions that can be utilized in a variety of contexts and are not mutually exclusive. In certain instances, stable matching can be viewed as a sort of optimization. Nevertheless, the two concepts have different objectives and priorities, and may necessitate distinct solutions.

5.2 Scheduling Problem in the Data Center

Figure 5.1 illustrates the job scheduling problem between servers and containers in a data center. It is a job assignment that involves two groups of agents consisting of a set of containers and a set of servers. Virtualization technology

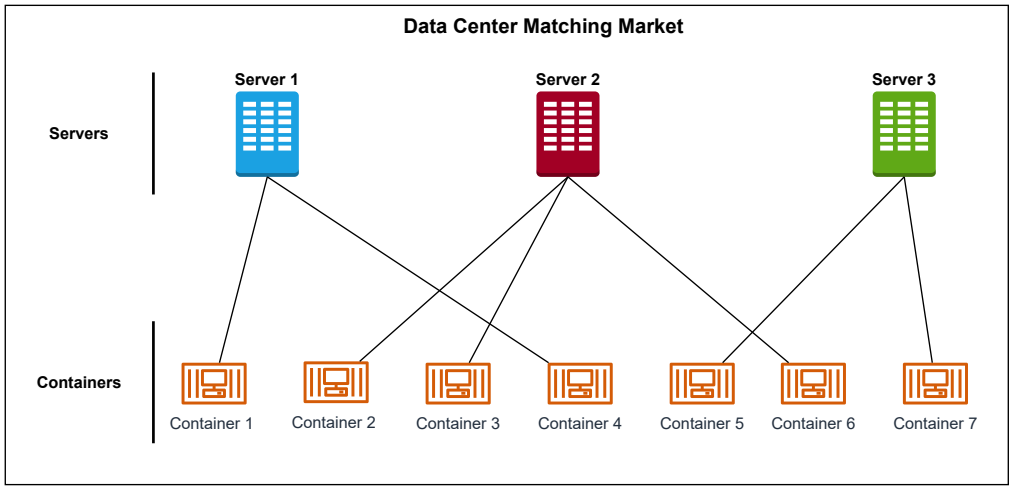

Fig. 5.1 Illustration of data center matching market, between servers and containers.

scheduling has several objectives, such as increasing the availability of containers or reducing the power consumption in a data center. The data center manager may use the optimization technique to solve this scheduling problem to maximize the company profit or optimize the application availability. However, job scheduling involves a collection of containers and servers, each of which has a different profile and preferences for the other side. For example, a container requires a server with a high-speed CPU and an internet connection, while another requires a large memory or storage capacity. On the servers, each wants to maximize their resources to optimize the company's benefit. Using the optimization technique, conflicts of interest between agents are resolved arbitrarily so that not all agents are satisfied with the results obtained. For instance, some containers may be dissatisfied with the outcome if server resource utilization is optimized. This is because optimization only works to achieve group goals but ignores each individual's wishes.

In the job scheduling problem, it is essential to maintain stability between the containers and servers. Stability is important to minimize the cost of re-matching or re-pairing between containers and servers. When an agent decides to change the partner, this entails costs that need to spend, such as

migration, reconfiguration, and downtime of the application while performing the migration.

5.2.1 Problem Definition

A traditional SMP is a two-sided matching problem based on the one-to-one model, meaning that one male agent can pair with one female agent and vice versa. In the containers and server scheduling problems, the hospitals/residents problem [18] [23] is employed, which is a two-sided matching problem for a many-to-one model in which one hospital may couple with one or more residents (medical interns). In the current problem, a server acts as a resource provider, whereas a container acts as a resource consumer. We provide a formal definition of the problem. An instance of the problem is $I = (S, C, Q, L)$, where S and C denote a set of servers and containers, respectively. A set of servers $S = \{s_1, s_2, \ldots, s_m\}$ and containers $C = \{c_1, c_2, \ldots, c_n\}$. Each server $s_j \in S$ has a positive integer of quota value denoted by $Q(s_j)$, where $Q(s_j) \geq 1$. Each container $c_i \in C$ has a preference list L where containers rank each member of S. The preference of agent c_i is denoted as $L(c_i)$. Likewise, server $s_j \in S$ also has a preference list where the server ranks each member of C. In this simulation scenario, we assume that the preference for containers may change frequently. Thus, the instance is $I = (S, C, Q, L_1, L_2, \ldots, L_k)$, and the dynamic instance is $DI = \{I_1, I_2, \ldots, I_k\}$, k denotes the number of instances that arises in the simulations. Since preference changes occur frequently, we implement a long-term strategy to address the problem.

In this implementation, a server's resource specifications, such as CPU and memory, are persistently defined; this indicates that server resources remain static. Whereas for containers, resource usage fluctuates dynamically in response to the amount of computation and requests made by applications within the container. In the stable matching problem, each agent defines the order of preference for the opposite side. A container defines the preference

order based on the server's specifications. At the same time, the server also defines the order of preference based on the resource utilization of containers. Based on the agent's characteristics, the container's preference for the server is static because the resources provided by the server are fixed. At the same time, the server's preference for containers is dynamic because resource utilization changes dynamically. Since the resource utilization of containers dynamically changes, the problem is defined as a stable matching problem under dynamic preference. Since the containers' preferences are static, Theorem 4.2 is employed to solve the problem.

5.2.2 The Preference Rule

The preference rule of servers to containers

It is typical for a company to maximize its profit. The data center company can increase profits by improving the power efficiency of each server in the data center. Thus, servers tend to select containers that increase their resource utilization rate. To determine a server's preference, a server prefers a container that requires as many resources as possible; the greater a server's utility, the greater its potential profit.

In this study, CPU and memory usage of the container are used to determine the preference ranking. Using the Euclidean distance formula, the similarity between the resource capacity provided by the server and the resources utilized by the containers is determined (see Table 5.1). Since the container resource utilization is dynamically changed, the dynamic preference of the server is defined for a periodic time. Two daily preferences, for day and night utilization, over the course of seven days are generated for the simulation.

The preference rule of containers to servers

In this simulation, we define the container's preference based on the similarity between the container's initial resource requests and the server's resource

capacity. It is assumed that containers have defined their initial resource requests. For this simulation, the average resource usage data (CPU and memory) is used for one week. Therefore, the containers' preferences remain static. The distance between the initial CPU request and the servers' resource capacity is calculated to determine the containers' preference.

5.3 Evaluation Scenario

For the simulation scenario, it is necessary to make clear assumptions first. In this evaluation scenario, a company that manages its private data center is assumed. We have five servers with various resource specifications (see Table 5.1) and 50 containers containing web applications with different resource needs. This simulation assumes that the data center model is a shared resource, where each container must determine its minimal resource requests. If the container's resource utilization exceeds the minimum request, a burstable scheme will be applied, i.e., the containers are allowed to use the remaining resources of the server if available. Table 5.1 shows the servers' specifications for this simulation. Moreover, we define the servers' quota for container placement.

Table 5.1 Servers' resource specification and quota for containers

Hostname	**CPU**	**Memory**	**Max Power**	**Quota**
server01	2×VCPUs	2 GB	220 Wh	12
server02	2×VCPUs	4 GB	225 Wh	12
server03	4×VCPUs	2 GB	240 Wh	14
server04	4×VCPUs	4 GB	250 Wh	14
server05	4×VCPUs	8 GB	260 Wh	14

For the simulation scenario, 50 web page applications that perform CPU and memory-intensive computations to simulate load in the cluster were deployed. The *Locust* load testing framework was used to generate load traffic on each container with varying behavior as experimental data. The resource usage of the containers are generated for seven days. Each server's CPU usage was recorded

for the evaluation scenarios to evaluate each server's power consumption. In this experiment, we obtained seven different server-to-container preferences. While the preference of the container-to-server is static. Thus, we have a dynamic instance consisting of seven instances, with the probability of each being $\frac{1}{7}$.

5.3.1 Evaluation of Agent Satisfaction

In this section, we evaluate the results of experiments by calculating the agent's satisfaction score. As demonstrated in Table 5.2, seven unique matchings occurred during the experiment.

Table 5.2 The score of α, β, and the blocking pair expected value of obtained matchings.

Matching	α	β	Blocking Pair EV
μ_1	1	31	17.08
μ_2	1	41	**13.93**
μ_3	1	43	20.23
μ_4	1	43	16.24
μ_5	1	37	16.1
μ_6	1	43	22.54
μ_7	1	37	16.38

As seen in Table 5.2, the seven matchings obtained have the same α value of 1, which means that all matchings are only stable against a single instance. When we use the most stable concept, it will be challenging to determine which matching will be selected, and we can only determine the stable matching by choosing randomly. Furthermore, using the least blocking pair concept, several variants of the β value of the matching are obtained, where μ_1 is the matching with the minimum total number of blocking pairs. In this experiment, we consider the probability of the instance and calculate the expected value of the blocking pair on the dynamic instance. As shown in Table 5.2, μ_2 has the lowest expected value of the number of blocking pairs.

Xu et al. [28] analyze their work by calculating the satisfaction level of VMs and servers. In this study, we also analyze the satisfaction level of matching by using the satisfaction score of each matching in a dynamic instance. The satisfaction score reflects the level to which each agent on the market is satisfied with the acquired matching based on their defined preferences.

First, we introduce some notations to obtain the satisfaction score of matching in a dynamic instance. Given a set of containers $C = \{c_1, c_2, \ldots, c_n\}$ and a set of servers $S = \{s_1, s_2, \ldots, s_m\}$. Container-server matching is a many-to-one stable matching problem where a server can pair with more than one container. Whereas a container is only paired with a server. We define the satisfaction score of a server as follows.

$$sat(s) = \sum_{c \in \mu(s)} n - R(c) \tag{5.1}$$

where $R(c)$ denotes the rank given by s to c in s's preference, n is the cardinality of a set of container C, and c is the containers paired with s. Since a container can only pair with a server, the satisfaction score of the container is as follows.

$$sat(c) = m - R(\mu(c)) \tag{5.2}$$

where $R(\mu(c))$ denotes the rank given by c to $\mu(c)$ in c's preference, and m is the cardinality of a set of server S. The satisfaction of matching μ is then the sum of the score of all involved agents.

$$sat(\mu) = \sum_{s \in S} sat(s) + \sum_{c \in C} sat(c) \tag{5.3}$$

Since this study considers a stable matching under dynamic preference, the total satisfaction score in the dynamic instance is needed. The satisfaction score of matching in the stable matching problem under dynamic preference might not be the same for every instance. Moreover, a potential blocking pair

might occur in some instances. Therefore, we must consider the blocking pair occurrence to find the satisfaction score of stable matching under dynamic preference. In a stable matching under dynamic preference, a set of blocking pair $BP = \{bp_1, \ldots, bp_j\}$ may occur. A pair (s_x, c_y) is said to be a blocking pair in matching μ if they are not partners in μ, but s_x prefers c_y to $\mu(s_x)$ and c_y prefers s_x to $\mu(c_y)$. A matching is stable if no blocking pair is found, such as $BP = \{\}$.

If there is a set of blocking pair $BP = \{bp_1, \ldots, bp_j\}$ in matching μ, we need to calculate the score of the blocking pair before finding the satisfaction score.

$$BP_{score}(\mu) = \sum_{bp \in BP} (m - R_{bp}(s)) + (n - R_{bp}(c)) \tag{5.4}$$

where $R_{bp}(s)$ denotes the rank of blocking agent s in $bp(s)$'s preference, and $R_{bp}(c)$ denotes the rank of blocking agent c in $bp(c)$'s preference. Thus, the satisfaction score of matching μ is as follows.

$$sat(\mu) = \sum_{s \in S} sat(s) + \sum_{c \in C} sat(c) - BP_{score}(\mu) \tag{5.5}$$

To obtain the satisfaction score of matching μ in the dynamic example $DI = \{I_1, I_2, \ldots, I_k\}$, the average satisfaction score of matching μ in DI is calculated.

$$sat_{DI}(\mu) = \frac{\sum_{i=1}^{k}(sat_i(\mu))}{k} \tag{5.6}$$

where k is the number of instances in a dynamic instance DI.

Table 5.3 shows the agent's satisfaction score for each matching. The results show that μ_2 has the highest satisfaction score. In contrast, μ_6 is the matching with the lowest score among the others. Based on the results in Tables 5.2 and 5.3, we select μ_2 as the solution for the problem because μ_2 has the lowest expected value of the number of blocking pairs, and μ_2 also gains the highest satisfaction score among other matchings.

Table 5.3 Agent satisfaction score against obtained matching

Matching	α	β	EV	Satisfaction Score
μ_1	1	31	17.08	770
μ_2	1	41	13.93	**827.42**
μ_3	1	43	20.23	733.25
μ_4	1	43	16.24	788.28
μ_5	1	37	16.1	779.57
μ_6	1	43	22.54	633.14
μ_7	1	37	16.38	768.28
AVG				757.13

5.3.2 Trade-off analysis

Table 5.4 Total servers power consumption

Matching	α	β	EV	Power consumption (Wh)
μ_1	1	31	17.08	933.59
μ_2	1	41	13.93	931.68
μ_3	1	43	20.23	931.69
μ_4	1	43	16.24	931.68
μ_5	1	37	16.1	931.66
μ_6	1	43	22.54	**931.33**
μ_7	1	37	16.38	931.66
AVG				931.89

Stable matching is utilized in this simulation to maintain the containers' and the servers' satisfaction while enhancing energy efficiency. This section evaluates the servers' power consumption for each matching. Several studies show a linear relationship between power consumption and CPU usage of computers [6, 20, 26]. According to these studies, the average power consumption of an idle server is 70% of a fully utilized server. Thus, the power consumption $P(S)$ formula is described as follows:

$$P(S) = P_{\max} \times (0.7 + (0.3 \times U(CPU))) \tag{5.7}$$

where:

$P(S)$ = Power consumption of server S in Watt per hour (Wh)

$P_{\max}$ = Maximum power of server in Watt per hour (Wh)

$U(CPU)$ = % CPU usage of server

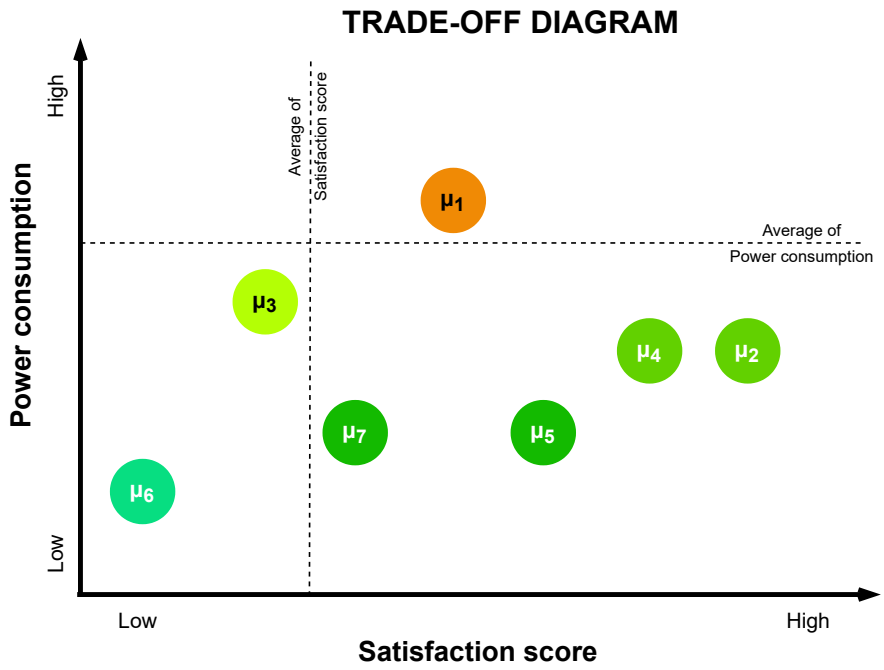

Fig. 5.2 Trade-off diagram between the total server's power consumption and the agents' satisfaction score. The color of the circle represents green energy

Table 5.4 shows the total power consumption of servers for each matching. The results show that μ_6 is the matching with the lowest power consumption compared to the others. However, considering the results in Table 5.3, μ_6 is matching with the lowest satisfaction score among the others. The purpose of implementing stable matching in this application is to obtain energy efficiency while maintaining agent satisfaction. In this experiment, we consider the trade-off between power consumption and the satisfaction score of agents in the market. Despite the fact that μ_2's power efficiency is not the best among the others, μ_2's power consumption is still lower than the average power consumption, and μ_2 gains the highest satisfaction rating among the others.

Figure 5.2 shows the trade-off between the total server's power consumption and the satisfaction score of agents in the market. μ_6 is the most energy-efficient (the least power consumption) compared to other matchings. However, μ_6 has the lowest satisfaction score compared to different matchings. Conversely, μ_2 has the highest satisfaction score of agents, despite not being superior in terms of energy efficiency. Considering the trade-off between energy efficiency and the satisfaction score of agents, μ_2 can be selected as a matching solution for this problem.

5.4 Summary

This chapter demonstrates the implementation of a stable matching problem under dynamic preference in the data center scheduling problem, between servers and containers. The main objective of using stable matching in this simulation is to maintain all agents' satisfaction and obtain stability. In stable matching under dynamic preference, the matching result is not perfectly stable in all instances, however, we try to minimize the number of blocking pairs. In the evaluation part, we show the power consumption of servers in the data center for each obtained matching, as described in the simulation result, our selected matching does not gain the most efficient result in power consumption, however, our selected matching obtain the highest score of satisfaction score.

A stable matching is needed for the scheduling job to prevent the agents from complaining about their pair in the matching outcome. Hence, we can minimize the expenses associated with re-matching, such as migration, reconfiguration, and downtime, when the market decides to change the matching.

Chapter 6

Summary and Future Work

6.1 Summary

We study stable matching under dynamic preference, which is near the problem of the real-world situation. We introduce two strategies to find stable matching, namely short-term and long-term stability strategies. Short-term stability is the most straightforward strategy to maintain matching stability but is costly if the frequency of preference changes is high. Our second strategy, long-term stability, finds stable matching based on multiple preferences, long-term stability can avoid the re-matching costs.

In the short-term strategy, we intend to find an efficient way the stable matching every time the preference changes. We propose an update matching mechanism to find a matching by updating the prior matching so that it can be stable with the new preference. Our findings show how to update a matching and maximized the initial matching. Compared to other existing methods, our proposed concept is outperform in minimizing re-matching costs.

In the long-term strategy, we intend to find the solution for long-term stable matching more accurately. We use a blocking pair perspective to find the stable matching with the minimum blocking pair in the dynamic instance. Considering the number of blocking pairs in the stable matching under dynamic

preference is a novel concept. By understanding the amount of blocking pairs in detail, it is possible to obtain additional information regarding stable matching. Consequently, the information obtained from the number of blocking pairs can assist in determining stable matching with greater precision than the existing method, In other words, the usage of the index (α, β, and EV) is better than α. We also define the concept of stability for the stable matching problem under dynamic preference.

Stable matching with dynamic preference is an often encountered stable matching problem in real-world situations. We implement our findings to the job scheduling of the data center. A stable matching is needed for the scheduling job to prevent the agents from complaining about their pair in the matching. Hence, we can minimize the expenses associated with re-matching, such as migration, reconfiguration, and downtime, when the market decides to change the matching.

6.2 Future Work

In this study, we propose solutions to find a stable matching under dynamic preference. For the short-term strategy, we propose the update matching mechanism to find stable matching for the stable marriage problem every time the preference changes. In the two-sided matching field of study, there are a lot of stable matching problem variants have been proposed. Generalizing our findings to other variants would be challenging in the future.

For the long-term strategy, generalizing our findings to other variants would also a challenging in the future. Such as considering the dynamic quota in the Hospitals/Residents Problem (HRP) is a new problem in stable matching. In a dynamic situation, the quota of hospitals in HRP may also dynamically change.

Based on what we learned during this research, the creation of a database comprising many instances and matchings will be of great assistance for stable matching research with dynamic preferences. Instead of using a matching

algorithm or brute force to find all matching of each instance, providing a database will speed up the process of finding the matching.

[This page is intentionally left blank]

References

[1] Alimudin, A. and Ishida, Y. (2020a). Dynamic assignment based on a probabilistic matching: Application to server-container assignment. *Procedia Computer Science*, 176:3863–3872.

[2] Alimudin, A. and Ishida, Y. (2020b). Service-based container deployment on kubernetes using stable marriage problem. In *Proceedings of the 2020 The 6th International Conference on Frontiers of Educational Technologies*, pages 164–167.

[3] Alimudin, A. and Ishida, Y. (2021). A study of the random order mechanism for uncertain preferences in the stable marriage problem. In *2021 8th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, pages 1–6. IEEE.

[4] Alimudin, A. and Ishida, Y. (2022). Matching-updating mechanism: A solution for the stable marriage problem with dynamic preferences. *Entropy*, 24(2):263 (14 pages).

[5] Alimudin, A., Ishida, Y., and Suzuki, K. (2023). Maintaining stability for a matching problem under dynamic preference. *IEEE Access*, 11:24203–24215.

[6] Arshad, U., Aleem, M., Srivastava, G., and Lin, J. C.-W. (2022). Utilizing power consumption and sla violations using dynamic vm consolidation in cloud data centers. *Renewable and Sustainable Energy Reviews*, 167:112782.

[7] Aziz, H., Biró, P., Fleiner, T., Gaspers, S., de Haan, R., Mattei, N., and Rastegari, B. (2022). Stable matching with uncertain pairwise preferences. *Theoretical Computer Science*, 909:1–11.

[8] Aziz, H., Biró, P., Gaspers, S., De Haan, R., Mattei, N., and Rastegari, B. (2016). Stable matching with uncertain linear preferences. In *International Symposium on Algorithmic Game Theory*, pages 195–206. Springer.

[9] Biró, P., Manlove, D. F., and Mittal, S. (2010). Size versus stability in the marriage problem. *Theoretical Computer Science*, 411(16-18):1828–1841.

[10] Chen, F., Zhou, X., and Shi, C. (2019). The container deployment strategy based on stable matching. In *2019 IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pages 215–221. IEEE.

[11] Chen, J., Niedermeier, R., and Skowron, P. (2018). Stable marriage with multi-modal preferences. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 269–286.

[12] Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., and Weihl, B. (2002). Globally distributed content delivery. *IEEE Internet Computing*, 6(5):50–58.

[13] Fenoaltea, E. M., Baybusinov, I. B., Zhao, J., Zhou, L., and Zhang, Y.-C. (2021). The stable marriage problem: An interdisciplinary review from the physicist's perspective. *Physics Reports*, 917:1–79.

[14] Gale, D. and Shapley, L. S. (1962). College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15.

[15] Gusfield, D. and Irving, R. W. (1989). *The stable marriage problem: structure and algorithms*. MIT press.

[16] Irving, R. W. (1985). An efficient algorithm for the "stable roommates" problem. *Journal of Algorithms*, 6(4):577–595.

[17] Irving, R. W. (1994). Stable marriage and indifference. *Discrete Applied Mathematics*, 48(3):261–272.

[18] Irving, R. W., Manlove, D. F., and Scott, S. (2000). The hospitals/residents problem with ties. In *Scandinavian Workshop on Algorithm Theory*, pages 259–271. Springer.

[19] Iwama, K. and Miyazaki, S. (2008). A survey of the stable marriage problem and its variants. In *International conference on informatics education and research for knowledge-circulating society (ICKS 2008)*, pages 131–136. IEEE.

[20] Kella, A. and Belalem, G. (2014). Vm live migration algorithm based on stable matching model to improve energy consumption and quality of service. In *CLOSER*, pages 118–128.

[21] Ma, J. (1996). On randomized matching mechanisms. *Economic Theory*, pages 377–381.

[22] Maggs, B. M. and Sitaraman, R. K. (2015). Algorithmic nuggets in content delivery. *ACM SIGCOMM Computer Communication Review*, 45(3):52–66.

[23] Manlove, D. F. (2008). Hospitals/residents problem. *Encyclopedia of Algorithms*, pages 390–394.

[24] Miyazaki, S. and Okamoto, K. (2017). Jointly stable matchings. *Leibniz International Proceedings in Informatics, LIPIcs*, 92:56–1.

[25] Roth, A. E. and Vate, J. H. V. (1990). Random paths to stability in two-sided matching. *Econometrica: Journal of the Econometric Society*, pages 1475–1480.

[26] Sinha, R., Purohit, N., and Diwanji, H. (2011). Energy efficient dynamic integration of thresholds for migration at cloud data centers. *IJCA Special Issue on Communication and Networks*, 1:44–49.

[27] Wirth, N. (1985). *Algorithms & data structures*. Prentice-Hall, Inc.

[28] Xu, H. and Li, B. (2011). Egalitarian stable matching for vm migration in cloud computing. In *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 631–636. IEEE.

[This page is intentionally left blank]

Publication List

List of Papers with Referee's Review

1. Alimudin A, Ishida Y. Matching-Updating Mechanism: A Solution for the Stable Marriage Problem with Dynamic Preferences. Entropy. 2022 Feb 11;24(2):263. (14 pages)

2. Alimudin A, Ishida Y, Koutarou Suzuki. Maintaining Stability for a Matching Problem under Dynamic Preference. *IEEE Access*. 11:24203-24215

List of Papers at International Conference with Referee's Review

1. Alimudin A, Ishida Y. A Study of the Random Order Mechanism for Uncertain Preferences in the Stable Marriage Problem. In2021 8th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA) 2021 Sep 29 (pp. 1-6). IEEE.