Received 9 December 2022, accepted 3 February 2023, date of publication 8 February 2023, date of current version 14 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3243245

METHODS

Maintaining Stability for a Matching Problem Under Dynamic Preference

AKHMAD ALIMUDIN[1,2], **YOSHITERU ISHIDA**[1], **AND KOUTAROU SUZUKI**[1]
[1]Department of Computer Science and Engineering, Toyohashi University of Technology, Toyohashi 441-8580, Japan
[2]Department of Multimedia Creative Technology, Politeknik Elektronika Negeri Surabaya, Surabaya 60111, Indonesia

Corresponding author: Akhmad Alimudin (akhmad.alimudin.qq@tut.ac.jp)

This work was supported in part by the Toyohashi University of Technology, and in part by the Ministry of Education, Culture, Sports, Science and Technology - Japan.

ABSTRACT This study investigates two-sided matching and considers dynamic preference. In a stable matching problem, dynamic preference is a situation that often happens in real-world situations where the agent cannot express their preference with certainty. This study proposes a new concept to find stable matching using the blocking pair perspective. A stable matching is determined by identifying a matching by finding a matching with the minimum blocking pairs in multiple instances. In addition, the definition of stability is extended for the stable matching problem under dynamic preference to propose three new notions of stability. The proposed concept demonstrates more detailed information to assist in determining a stable matching with a dynamic preference. Moreover, the experiment results show that matching with the lowest expected value of the blocking pair gains the highest satisfaction score of agents in the market.

INDEX TERMS Stable matching, dynamic preference, blocking pair, expected value.

I. INTRODUCTION

Gale and Shapley [1] introduced the Stable Marriage Problem (SMP), the most popular two-sided matching problem. The SMP instance is $I = (M, W, L)$, where $M = \{m_1, m_2, \ldots, m_n\}$ and $W = \{w_1, w_2, \ldots, w_n\}$ denote a set of men agents and women agents, respectively, and L is the preference list of each agent towards the opposing side. If a man m and a woman w are paired in matching μ, they are called partners, and we write $m = \mu(w)$ and $w = \mu(m)$. SMP is the one-to-one stable matching problem with an equal number of men and women. For n size of SMP, the number of men equals the number of women, $|M| = |W| = n$. In a classic SMP, each agent defines their preference in strict order, requiring that they include all available members of the opposite sex. Matching μ is considered stable if no blocking pair (BP) is discovered. A BP comprises a man m and a woman w who are not paired in matching μ but like each other to their current partner. We write $a \succ_b c$ to say that agent b prefers a to c. Therefore, m and w is a blocking pair in matching μ if $w \succ_m \mu(m)$ and $m \succ_w \mu(w)$.

The associate editor coordinating the review of this manuscript and approving it for publication was Gustavo Olague.

Since its introduction in 1962, numerous versions of SMP have been introduced, such as the stable roommate problem and hospitals/residents problem [2], [3], [4], [5]. The SMP has also been implemented to solve real-world problems. For example, the hospitals/residents problem variant is applied to assign the residents (medical interns) to hospitals. Recently, the SMP has been extensively used for large-scale computer applications such as in content delivery networks technology [6] and scheduling strategy for assigning virtual machines (VMs) to servers [7], [8], [9].

In a real-world situation, the agent could change their preference. The preference changes of agents can be caused by several things, such as the lack of information about their potential partners or certain conditions that force an agent to change their preferences. We study the stable matching problem under dynamic preference. In classical SMP, it is assumed that each agent expresses their preferences with certainty, i.e., the agents will never change their preferences. Nowadays, preference uncertainty is the future trend and a challenge to the stable matching problem [10], and preference uncertainty leads to the formation of a dynamic preference.

A classical SMP instance is $I = (M, W, L)$. If agents change their preferences, this leads to SMP with dynamic preference. Moreover, an instance of SMP with dynamic preference leads to the formation of a dynamic instance, $DI = (M, W, L_1, L_2, \ldots, L_k)$, where k is the number of unique preference lists that occur because of 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 following 3×3 SMP illustrates the problem:

Example 1: There is a set of men $M = \{m_1, m_2, m_3\}$ and women $W = \{w_1, w_2, w_3\}$. Suppose that agent m_1 expresses two different preferences. Consequently, these settings produce two SMP instances, as seen in Figure 1:

Instance 1 (I_1)

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

Instance 2 (I_2)

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

FIGURE 1. **The 3 × 3 SMP with dynamic preference.**

A related study conducted prior to this proposed a mechanism to update stable matching when the preference changes [11]. It is termed a short-term stability strategy since the preference infrequently changes. However, this study assumes that preference frequently changes, and employing a short-term strategy might be costly when preference changes frequently occur.

In this study, we propose a new concept to find a stable matching under dynamic preference. We use the blocking pair perspective to find a stable matching under dynamic preference. The most stable matching concept [12], [13], [14], [15] is an existing solution to solve the stable matching problem under dynamic preference. The most stable matching concept determines stable matching by selecting a matching with the highest α value, where α indicates the number of stable matchings against instances. The α value is determined by counting the stability number of matching against available instances. When there is no blocking pair, α is increased. However, if at least one blocking pair is discovered, the matching is considered unstable, and α is not increased. The most stable concept only considers matching stability and ignores the number of blocking pairs within an unstable matching. However, when the matching is unstable, the number of blocking pairs may be greater than one. This motivates the use of blocking pairs as a baseline for determining stable matching by selecting a matching with the minimum number of blocking pairs.

Our contributions. This study proposes a strategy to find stable matching under dynamic preference. We introduce a new concept to find stable matching under dynamic preference using the blocking pair perspective. Moreover, the study broadens the definition of stability for stable matching with dynamic preference. Three notions of stability are introduced for the matching problem under dynamic preference.

Definition 1: 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 2: α-most stable is a criterion for a matching that admits stability in at least 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 $1 \leq \alpha \leq k$. When $\alpha = k$, the matching μ admits stability in all instances which equals to fully stable.

Definition 3: β-Least BP is matching with the minimum blocking pairs in a dynamic preference. β indicates the number of blocking pairs for a matching μ in a dynamic instance. When $\beta = 0$, the matching μ admits stability in all instances which equals to fully stable.

The structure of this study is as follows: Section II contains the relevant research for this study. Section III discusses the proposed concept of finding stable matching under dynamic preference. Section IV shows the relation between our new concept and the existing concept, along with some special preference cases. Section V discusses the implementation of the findings in computer applications. Section VI discusses the time complexity of our findings. Finally, VII provides the conclusions of our work.

II. PRELEMINARIES

A. THE STABLE MARRIAGE PROBLEM

Gale and Shapley introduced the Stable Marriage Problem (SMP) [1]. An SMP is a two-sided matching problem in which the number of participants on each side is equal. Each agent's preference is expressed in strict order. The primary objective of the Gale-Shapley algorithm is to establish stable pairings for all agents involved. The matching procedure aims to find the agent couples (sets of men and women) who meet the specified criteria.

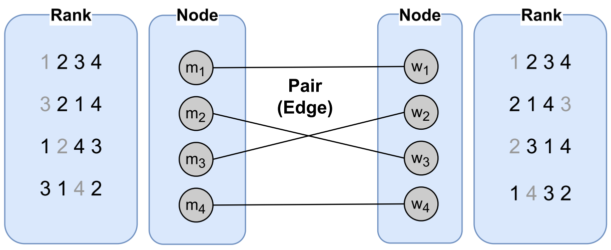

FIGURE 2. **The illustration of a stable marriage problem.**

An SMP instance of size n is $I = (M, W, L)$, where M and W are a set of men and women agents, respectively, such that $|M| = |W| = n$, and L is a set of preference lists for each agent. The preference list of an agent p is denoted by $L(p)$. In SMP, each agent's preferences are strictly ordered, indicating that each agent must include every agent available to the opposite sex in his preferences. If an agent p is in $L(q)$,

p is acceptable to q. Therefore, if p can be accepted by q and vice versa, then (p, q) is an acceptable pair.

A blocking pair determines matching stability in the SMP. A couple (m, w) is a blocking pair for matching μ if m and w are not a pair in μ, but m like w over $\mu(m)$ and w like m over $\mu(w)$. Such that, $(w \succ_m \mu(m)) \wedge (m \succ_w \mu(w))$. A matching is unstable if it contains at least one blocking pair; otherwise, it is stable. The illustration of SMP is depicted in Figure 2.

B. STABLE MATCHING WITH DYNAMIC PREFERENCE

In the stable matching problem, generally, each agent expresses a preference with certainty. However, as discussed in the introduction, sometimes in real-world situations, 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.

Definition 4: Dynamic preference is a matching problem preference in which the agent can express two or more different preferences.

Definition 5: A dynamic instance is a group of stable matching problem instances generated by dynamic preference.

A classical SMP instance is $I = (M, W, L)$. In the SMP under dynamic preference, agents can change their preferences by changing a set of preference lists L on an instance I. An instance of SMP under dynamic preference, called a 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 our previous study [11], a simple approach was used to maintain the matching stability under dynamic preference. Assuming that preference changes do not frequently occur, a matching is updated when a change in preference affects the stability of the obtained stable matching. The previous research focused on minimizing revision costs when updating a matching. In our study, the findings show that matching can be updated without a cyclic process. However, the previous approach to maintain stability is the short-term strategy that will be costly when preference changes frequently occur.

Therefore, the current study 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. Some studies [12], [13], [15], [16], [18] 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. [13] 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. To quantify the strengths of the stability, they use α 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 this current study. In this study, the *fully stable* notion is equivalent to *$\ell-$layer globally stable*. Furthermore, Aziz et al. [18] defined the *certain stable* and *possibly stable* concept when a probability of preference is given. The *certain stable* concept is also equivalent to the *fully stable* notion of stability in this study.

The proposed concept in this study finds a stable matching under dynamic preference. In contrast with the existing concept, which counts the stability of matching against a dynamic instance, the new concept uses the blocking pair perspective to find a stable matching under dynamic preference. The number of blocking pairs of each matching is quantified to determine the stable matching. In classical SMP, matching is considered unstable if at least one blocking pair is discovered. The blocking pair is a pair that does not satisfy the offered matching combination. However, in stable matching with dynamic preferences, obtaining a matching with zero blocking pairs against a dynamic instance is difficult [13]. Therefore, the blocking pair is allowed in the stable matching problem under dynamic preference. This motivates the present study to use blocking pairs as a reference for determining stable matching. Example 3 demonstrates the motivation behind the usage of the blocking pair for determining stable matching under dynamic preference. This study assumes that the probability of preference is given. Using probability distribution of preference and the number of blocking pairs in each matching, this study aims to find the expected value of the blocking pair in each matching to determine stable matching. The stable matching solution has the minimum expected value of the number of blocking pairs.

Table 1 describes several references in line with stable matching under dynamic preference. 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. [19] 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. [12] mention this idea in their open questions part.

C. MOST STABLE MATCHING CONCEPT

Agent preference changes are likely to occur in real-world situations, which 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 the proposed concept, the concept

TABLE 1. Related work to stable matching under dynamic preference.

Authors	Objective	Model Problem	Algorithm/ Technique
Miyazaki et al. [15]	Finding long-term Stable Matching	One-to-one	Most stable matching
Aziz et al. [12]	Finding Complexity and characterizing matching for long-term stable matching	One-to-one	Most stable matching
Chen et al. [13], [14]	Finding Complexity and characterizing matching for long-term stable matching	One-to-one	Most stable matching
Menon Vijay [16]	Stable matching with incomplete and inaccurate preference	One-to-one	Most stable matching
Alimudin and Ishida [11]	Finding short-term Stable Matching	One-to-one	Updating Matching
Bredereck et al. [17]	Finding short-term Stable Matching	One-to-one	Swap matching
This paper	Finding long-term Stable Matching	One-to-one	Least blocking pair

of finding stable matching using the most stable concept is summarized. To illustrate the issue, the following 3 × 3 SMP instance is used.

Example 2: Given an SMP instance under dynamic preference, a set of men $M = m_1, m_2, m_3$ and women $W = w_1, w_2, w_3$, and the two sets of preference, L_1 and L_2 are depicted in Figure 3.

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

FIGURE 3. The 3 × 3 SMP instances with dynamic preference.

The preferences in Figure 3 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 $\mu_3 = \{w_1 : m_3, w_2 : m_2, w_3 : m_1\}$. Moreover, 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, the α value is needed for each matching. α shows their respective stability strengths of each matching against a dynamic instance; α is a function that contains the probability of stability for each matching against a dynamic instance.

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

where:

- $\alpha(\mu)$ = Number of matching μ that are 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 (1), 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 2, the α for matching μ_1 and μ_3 is 1, whereas μ_2 is $2 = k$. It can be simply decided that μ_2 is the stable matching for the current matching problem.

The steps to find the stable matching using the most stable concept are summarized as follows:

1) Find all stable matching for each instance
2) Check the stability of each stable matching against the dynamic instance
3) Calculate the α of each matching using eq (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 [20]. Wirth's method [21] of trial-and-error and backtracking is a straightforward but inefficient approach to discovering all solutions of stable matching. Algorithm 1 shows how to find a stable matching under dynamic preference using the most stable concept.

Algorithm 1 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:      $\alpha \leftarrow 0$
8:      for $i = 1$ to $k$ do
9:          checkStability($sm$,$i$) //check stability of sm in i
10:         if ($sm$ is stable) then
11:             $\alpha \leftarrow \alpha + 1$
12:         end if
13:     end for
14:     $mostStable[sm] \leftarrow \alpha$
15: end for
16: $SORT_{DESC}(mostStable[sm])$
```

III. PROPOSED SOLUTION FOR STABLE MATCHING UNDER DYNAMIC PREFERENCE

The previous section discussed the concept that has been widely used to find stable matching under dynamic preference. This section introduces a novel approach to find the stable matching under dynamic preference by utilizing the blocking pair. In classical SMP, the matching is unstable when at least one blocking pair is found. Therefore, this study finds the minimum expected value of the blocking pair by quantifying the blocking pair for each matching.

A. FINDING THE LEAST BLOCKING PAIR MATCHING

The proposed concept finds long-term stable matching. In Section II-C, the stable matching under dynamic preference based on the "stable matching" perspective is found 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 by using the blocking pair that identifies the stable matching from 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 the implementation of the blocking pair approach to find stable matching under dynamic preference.

Example 3: Given the 3×3 *SMP instance* $I = (M, W, L)$*:*

$$\begin{array}{ll} L(m_1) = w_3, w_2, w_1 & L(w_1) = m_3, m_1, m_2 \\ L(m_2) = w_1, w_3, w_2 & L(w_2) = m_3, m_2, m_1 \\ L(m_3) = w_3, w_1, w_2 & L(w_3) = m_2, m_3, m_1 \end{array}$$

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, the aim is to identify the best matching among the two worst choices. Recalling the definition of stability in the stable marriage problem, a matching is stable if no blocking pair is found. Furthermore, the proposed concept attempts to find the best among the worst. Therefore, the number of blocking pairs is quantified for each matching to determine the best matching. A matching with the minimum number of blocking pairs is the best matching. In Example 3, 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. Therefore, this example motivates the usage of the blocking pair as a perspective to find stable matching in the stable matching problem under dynamic preference.

Example 3 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. This concept is also applied to find the matching with the minimum number of blocking pairs in the dynamic instance. To determine the best matching in a dynamic instance, we calculate the β value of each matching μ. β is the cardinality of distinct BPs in a dynamic instance and indicates the number of BPs for matching in a dynamic instance. The BP of matching μ in dynamic instance *DI* is defined as follows:

$$BP_{DI}(\mu) = \cup_{i=1}^{k} BP_i(\mu) = \{BP_1(\mu) \cup \cdots \cup BP_k(\mu)\} \quad (2)$$

$$\beta(\mu) = |\cup_{i=1}^{k} BP_i(\mu)| \quad (3)$$

where: BP_{DI} = Comprises 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 other matchings. Algorithm 2 shows how to find a stable matching under dynamic preference using the least blocking pair concept.

Algorithm 2 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:   $cBP_{smDI} \leftarrow count.unique(BP_{smDI})$
15: end for
16: $SORT_{ASC}(cBP_{smDI})$
```

B. 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. This study also considers the probability of instances occurring. Therefore, the stable matching with the minimum expected value of the BP needs to be identified. To find the BP expected value for each matching, we calculate with the following (4).

$$EV(\mu) = \sum_{i=1}^{k} \#(BP(\mu)|I_i)P(I_i) \quad (4)$$

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

The steps to find stable matching by finding the expected value of the BP is summarized as follows:

1) Find all stable matching for each instance
2) For each instance, find the blocking pairs of a matching in the dynamic instance
3) Calculate the expected value of the blocking pairs (EV) for each matching using eq. (4)
4) Find the matching with the minimum expected value of the blocking pair

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

FIGURE 4. **The 3 × 3 SMP instances with dynamic preferences with a probability of instance.**

Example 4: Consider the stable matching problem under dynamic preference example depicted in Figure 4.

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 $\mu_4 = \{w_1 : m_2, w_2 : m_3, w_3 : m_1\}$. Moreover, μ_1 and μ_2 are 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. Therefore, the next step is to quantify the number of blocking pairs for each matching, as shown in Table 2.

TABLE 2. **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

Now, the BP expected value for each matching can be found using eq. 4. 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 matching problem. If the previous mechanism is used in finding the α for each matching, all the corresponding α values will be 1.

IV. STABILITY NOTIONS FOR MATCHING PROBLEM UNDER DYNAMIC PREFERENCE

In a stable matching problem with dynamic preference, it is difficult to satisfy the classical SMP stability definition, where BPs are not allowed in a matching. Moreover, as the dimensions of the instance become more expansive, multiple instances stability need to be considered wherein a matching might be stable in one instance but unstable in other. A matching with *fully stable* character means it can satisfy the classical stability definition since it does not admit any BP in a dynamic instance. A *fully stable* matching means $\alpha = k$, where k is the number of instances in a dynamic instance. A *fully stable* matching also means matching with $\beta = 0$, a *fully stable* matching admits no BP in all instances.

In classical SMP, the definition of stability is determined by the existence of a BP in a matching. If one BP is found, the 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, a stable matching with dynamic preference is discussed. Therefore, an SMP with dynamic preference is $DI = (M, W, L_1, L_2, \ldots, L_k)$. Thus, it can be written as $DI = \{I_1, I_2, \ldots, I_k\}$. 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, two approaches to find a stable matching under dynamic preference are presented in this study. While the existing approach finds the most stable matching to the dynamic instance, the proposed approach finds a matching with the minimum blocking pairs.

In the most stable concept, as explained in Chapter II-C, the value of α indicates the strength of matching in a dynamic instance. To find α, the number of stability is counted for each matching against all instances. In quantifying the stability number, the classical SMP definition is referenced where matching is unstable if at least one blocking pair is found. To determine the α value of each matching, the number of stable matching for all instances is counted. As discussed in Chapter III-A, the proposed approach calculates the number of BPs for each matching in a dynamic instance.

The most stable concept and the proposed least BP concept depend on the existence of a BP to determine the stable matching. In the most stable concept, the BP is used to determine the stability for each instance to find α. Whereas for the least BP concept, the number of BPs is quantified to determine β.

Given an SMP instance, $I = (M, W, L)$ and 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 based on the most stable matching and the expected value of the number of BPs. BP is a key factor for the two approaches used in this study. The matrix notation illustrates the relationship between the two approaches for determining stable matching under dynamic preference.

$$\mu = \begin{array}{c} \\ \\ \end{array}\begin{array}{c} \begin{array}{cccc} I_1 & I_2 & \cdots & I_k \end{array} \\ \begin{bmatrix} BP_1 & BP_2 & \cdots & BP_k \\ P_1 & P_2 & \cdots & P_k \end{bmatrix} \end{array} \tag{5}$$

where:

μ = matching of dynamic instance
I = an instance of dynamic instance
P = Probability 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, the best matching is determined by finding the value of α for each obtained matching where the one with the highest α value is the stable matching solution. To find the α of matching μ, the matrix notation is utilized (5).

$$\alpha(\mu) = \sum_{i=1}^{k} SM_i(\mu) \tag{6}$$

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

In the least BP approach, the intent is to find the matching with minimum BP. Therefore, the number of BP in the dynamic instance is quantified by finding the β of matching μ, where β is the cardinality of blocking pairs for matching μ in the dynamic instance DI.

$$\beta(\mu) = | \cup_{i=1}^{k} BP_i(\mu)| \tag{7}$$

This section discusses the relationship between the proposed concept and the existing concept to find stable matching. The results show that α (6) and β (7) can be found using the matrix notation (5). In the most stable matching concept, α indicates the strength of matching in the dynamic instance, where $1 \leq \alpha \leq k$. The value of α is obtained by counting the instances where a matching can be stable. When $\alpha = k$, a matching is stable in all instances. Whereas $\alpha = 1$ means the matching is only stable in a single instance among all instances.

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.

A. SPECIAL CASE OF PREFERENCE: DYNAMIC PREFERENCE ON ONE SIDE

This section considers some special preference cases for the stable matching problem under dynamic preference which can arise in real-world scenarios. Consider the special case where the dynamic preference only occurs on one side. For example, matching between the servers and containers in a data center. It is reasonable to assume that the server evaluates its potential client by resource usage behavior, which dynamically changes, thereby having a dynamic preference. In contrast, containers evaluate the servers based on the server specification, which is static specification.

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

Theorem 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 have their first option by applying man as a proposer (man-optimal). Even if the women change their preference, all men agents can still have their first choice of the woman. Thus, the man-optimal matching is *fully stable* and always exists. □

Corollary 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: In Theorem 1, if the men's preference has the distinct first option, then fully stable always exists. The cyclic preference also has a similar pattern to Theorem 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 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 k times. Then $L_i = \{L_1, L_2, \dots, L_k\}$ and k instances, where $i = 1, 2, \dots, k$. Based on Theorem 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 Theorem 2 state holds, there is no need to find all the stable matching for each instance. By referring to Theorem 2, only the Gale-Shapley algorithm is used, where a man is 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 1 holds, rather than checking all instances, the only one that needs to be found is a man-optimal stable matching in a single instance.

Theorem 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, it is assumed that w_1 expresses m different preferences. Then the dynamic instance is $DI = \{I_1, \dots, I_m\}$. If matching μ_1 is stable in instance I_1, assume agent w_1 is paired with agent m_1 in matching μ_1, then $(w_1, m_1) = (w_1, \mu_1(w_1))$, $\mu_1 = \{(w_1, \mu_1(w_1)), (w_2, \mu_1(w_2)), \dots, (w_n, \mu_1(w_n))\}$. However, if preference of w_1 is changed in instance I_2, such that $L_1(w_1) \neq L_2(w_1)$, then $W' \equiv W \setminus w_1$ wherein $W' = \{w_2, \dots, w_n\}$, and $M' \equiv M \setminus \mu_1(w_1)$, such that $M' = \{\mu_1(w_2), \dots, \mu_1(w_n)\}$. 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 another man rather than her current partner, then w_1 could possibly form a blocking pair with the man in M'. The other women in W' will not form a blocking pair since they and their respective partners do not change their preferences. □

Corollary 2: 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 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. Because the current pair $(w_1, \mu(w_1))$ cannot be a blocking pair of itself. The maximum number of blocking pairs is β_{max} = (n × number of dynamic agents) - number of dynamic agents. □

B. RELATION OF DYNAMIC PREFERENCE AND STABLE MATCHING WITH TIES

In stable matching research, many extensions have been discussed [5]. One widely known variant of the stable matching problem is Stable Matching with Ties (SMT) [22]. Under certain conditions, the stable matching problem under dynamic preference can form a preference with ties. In SMT, an agent can express two or more agents with equal positions in the agent's preference. That is, 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, then $w_x =_{m_z} w_y$. In SMT, there are three notations of stability: weakly stable, strongly stable, and super stable matching. For weakly stable matching, a blocking pair of matching μ is defined as a pair (m, w) such that $\mu(m) \neq w$, $w \succ_m \mu(m)$ and $m \succ_w \mu(w)$. For 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 5: The 3 × 3 SMP instance with ties

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

In Example 5, m_1 expresses the preference with tie, where w_3 and w_2 m_1 are in the same position. These preference settings can be broken down into the stable matching problem under dynamic preference to obtain two SMP instances as shown in Figure 5. Each instance has a probability distribution in a stable matching problem with dynamic preference. However, the SMT stated that an agent might express two or more agents with an equal position in his/her preference. This means each instance must have an equal probability. In Example 5, I_1 and I_2 must have the same probability of occurrence, where the probability $I_1 = I_2 = 0.5$.

Instance 1 (I_1)

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

Instance 2 (I_2)

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

FIGURE 5. Transformation of SMP with a tie to SMP under dynamic preference.

Theorem 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 a couple (x, y) is found such that $\mu(x) \neq y$, $y \succ_x \mu(x)$ and $x \succ_y \mu(y)$ 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 5: Given an SMP Instance $I = (M, W, L)$ with a stable matching μ, suppose agent m expresses dynamic preference and forms a tie. If $\mu(m)$ is not tied, then strongly stable exists.

Proof: Without loss of generality, suppose agent m_1 changes his preference. There is a dynamic preference due to m_1 changing 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''. □

V. APPLICATION OF STABLE MATCHING WITH DYNAMIC PREFERENCE

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 (intern medical students) to hospitals. The stable matching problem is widely implemented in computer applications. For example, stable matching is implemented on a wireless network technology to gain a more efficient allocation of resources [23], [24]. Research in this area [25], [26] uses stable matching to migrate virtual machines (VMs) between servers in the data center. The objective is to improve energy efficiency in the data center while maintaining the virtual machines' quality of service. Furthermore, other studies [7], [8], [27] uses 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 [28], wherein the stable matching algorithm helps to balance the loads within server clusters. However, all mentioned references still use the static data of containers or VMs resource utilization to define the agent's preferences, i.e., they do not consider the dynamic resource usage that affects the preferences. 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 for the job scheduling of containers and servers.

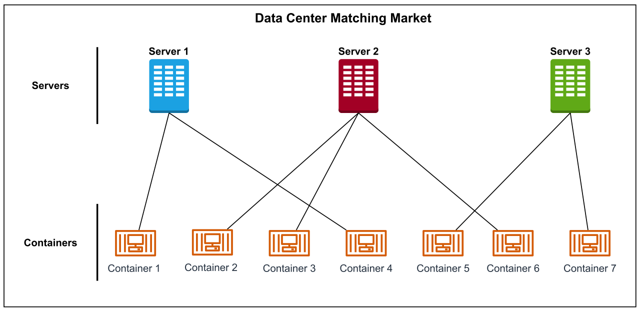

FIGURE 6. Illustration of job scheduling between servers and containers.

Figure 6 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 scheduling has several objectives, such as increasing the availability of containers or reducing the power consumption in a data center. Therefore, a 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 high-speed CPU and 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.

1) STABLE MATCHING PROBLEM MODEL

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 [4], [29] 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 $I = (S, C, L, Q)$, 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 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 2 is employed to solve the problem.

2) 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 3).

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.

3) 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.

A. 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 3) 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 3 shows the servers' specifications for this simulation. Moreover, we define the servers' quota for container placement.

TABLE 3. 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 $\frac{1}{7}$.

B. EVALUATION OF AGENT SATISFACTION

In this section, we evaluate the results of experiments by calculating the agent's satisfaction score. As demonstrated in Table 4, seven unique matchings occurred during the experiment.

TABLE 4. The score of α, β, and the blocking pair expected value of obtained matchings.

Matching	α	β	Blocking Pair EV
μ_1	1	31	2.44
μ_2	1	41	**1.99**
μ_3	1	43	2.89
μ_4	1	43	2.32
μ_5	1	37	2.30
μ_6	1	43	3.22
μ_7	1	37	2.34

As seen in Table 4, 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 4, μ_2 has the lowest expected value of the number of blocking pairs.

Xu et al. [25] 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.

$$sat(s) = \sum_{c \in \mu(s)} n - R(c) \tag{8}$$

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.

$$sat(c) = m - R(\mu(c)) \tag{9}$$

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.

$$sat(\mu) = \sum_{s \in S} sat(s) + \sum_{c \in C} sat(c) \tag{10}$$

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.

$$BP_{score}(\mu) = \sum_{bp \in BP} (m - R_{bp}(s)) + (n - R_{bp}(c)) \tag{11}$$

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.

$$sat(\mu) = \sum_{s \in S} sat(s) + \sum_{c \in C} sat(c) - BP_{score}(\mu) \tag{12}$$

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.

$$sat_{DI}(\mu) = \frac{\sum_{i=1}^{k}(sat_i(\mu))}{k} \tag{13}$$

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

Table 5 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 4 and 5, 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. Agent satisfaction score against obtained matching.

Matching	Satisfaction Score
μ_1	770
μ_2	**827.42**
μ_3	733.25
μ_4	788.28
μ_5	779.57
μ_6	633.14
μ_7	768.28
AVG	757.13

C. TRADE-OFF ANALYSIS

Stable matching is utilized in this study to enhance energy efficiency while maintaining container and server satisfaction. This section evaluates the servers' power consumption for each matching. Several studies show a linear relationship between power consumption and CPU usage of computers [26], [30], [31]. 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:

$$P(S) = P_{\max} \times (0.7 + (0.3 \times U(CPU))) \tag{14}$$

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

TABLE 6. Total servers power consumption.

Matching	Power consumption (Wh)
μ_1	933.59
μ_2	931.68
μ_3	931.69
μ_4	931.68
μ_5	931.66
μ_6	**931.33**
μ_7	931.66
AVG	931.89

Table 6 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, μ_6 is the 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 7 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.

VI. DISCUSSION

A. TIME COMPLEXITY

This section discusses the computational cost of finding stable matching under dynamic preference. Algorithm 1 discovers stable matching under dynamic preference using the most stable matching approach. There are two main loops in Algorithm 1. The first is a single loop used to discover all

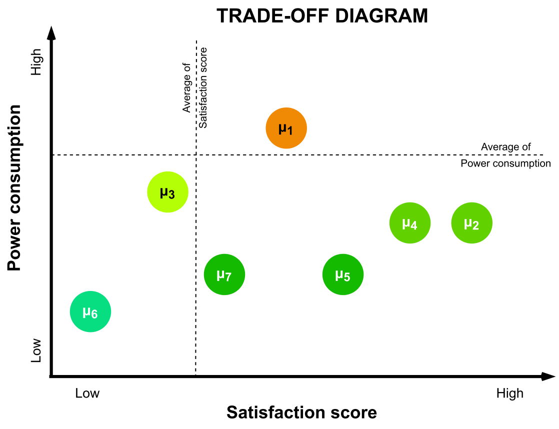

FIGURE 7. Trade-off diagram between the total server's power consumption and the agents' satisfaction score. The color of the circle represents green energy.

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 [3]. 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 $O(k \cdot n! \cdot n^2)$.

Following the most stable matching concept, the least blocking pair concept (Algorithm 2) 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 IV-A, a stable matching can be found in less time. In Theorem 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 2 is $O(k \cdot n^2)$. The constraints in Theorem 2 employ the Gale-Shapley algorithm in k times, where k represents the number of instances contained in the dynamic instance.

B. STRATEGIES TO MAINTAIN THE STABILITY OF MATCHING WITH DYNAMIC PREFERENCE

We discussed stable matching with dynamic preferences in the previous sections. 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 instances. However, this study proposes a different concept where the number of blocking pairs is counted.

The most stable and proposed concepts are similar in determining stable matching. Both concepts observe the presence of blocking pairs in a matching. We select the matching with the highest α in the most stable concept to solve the problem. The α is obtained by counting the number of stable instances to a matching. If the matching is stable against an instance, the α value increases. If at least one blocking pair is found, the matching is counted as unstable, and the α is not increasing. However, the number of the blocking pair in the unstable matching could be more than one pair. In contrast to the most stable concept, the blocking pair perspective is proposed in this study to determine stable matching. The number of BPs from the unstable matching is counted, and matching with the minimum number of BPs is selected.

In the most stable matching concept, obtaining $\alpha = k$ is difficult even for $k = 2$ [13]. Our proposed concept helps determine the stable matching under dynamic preference in more detail. For example, when we are provided with the worst option of matchings, where $\alpha < k$ and all matchings have the same score of α, our proposed concept selects the matching with the minimum blocking pair as the solution to the problem.

During the implementation of our findings in Section V, we show how to find a matching solution for the container and server scheduling problem. Our findings help to determine the matching solution in more detail. The matching with the minimum expected value of the blocking pairs gains the highest satisfaction score of agents in the market. However, the trade-off between power efficiency and the satisfaction score is worth considering to define the matching solution.

VII. CONCLUSION

We propose a new concept to find stable matching under dynamic preference. The strategy employs the blocking pair to determine stable matching when dynamic preference occurs. In stable matching with dynamic preference, the worst possible matching options may be provided, such as that all the obtained matchings are only stable to a single instance, where $\alpha = 1$. The proposed strategy assists in determining the stable matching under dynamic preference by choosing the best among the worst options. In addition, the expected value (EV) of blocking pairs is calculated to obtain more detailed results by considering the preference probability.

Moreover, we implement stable matching under dynamic preference for the job scheduling problem between servers and containers in a data center. Our findings help to determine the matching solution in more detail. The matching with the minimum expected value of the blocking pairs gains the highest satisfaction score of agents in the market. However, the trade-off between energy efficiency and the agents' satisfaction is considered in selecting the matching solution.

Our study considers stable matching with dynamic preference. In real-world applications, many constraints may lead to a new variant of stable matching with the dynamic conditions, such as considering dynamic quotas in many-to-one stable matching (hospitals/residents problem). The hospitals/residents problem instance is $I = (H, R, L, q)$, where H and R denote a set of hospitals and residents, respectively. L denotes the preference of each agent, and q represents the quota of each hospital. In the hospitals/residents problem, the quota on the hospital also determines the stability of a matching. When the quota on an instance changes, getting a stable matching in the dynamic instance becomes difficult. For example, there were two hospitals/residents problem instances, I_1 and I_2, with the same preference. The μ_1 is a matching that is stable in I_1. If the quotas (q) of I_1 and I_2 were different, it would be difficult to maintain the stability of μ_1 against I_2, even if the preference of I_1 and I_2 were the same.

REFERENCES

[1] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *Amer. Math. Monthly*, vol. 69, no. 1, pp. 9–15, Jan. 1962.

[2] R. W. Irving, "An efficient algorithm for the 'stable roommates' problem," *J. Algorithms*, vol. 6, no. 4, pp. 577–595, 1985.

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

[4] D. F. Manlove, "Hospitals/residents problem," in *Encyclopedia of Algorithms*, 2008, pp. 390–394, doi: 10.1007/978-0-387-30162-4_180.

[5] K. Iwama and S. Miyazaki, "A survey of the stable marriage problem and its variants," in *Proc. Int. Conf. Informat. Educ. Res. Knowl.-Circulating Soc. (ICKS)*, Jan. 2008, pp. 131–136.

[6] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, "Globally distributed content delivery," *IEEE Internet Comput.*, vol. 6, no. 5, pp. 50–58, Sep. 2002.

[7] A. Alimudin and Y. Ishida, "Dynamic assignment based on a probabilistic matching: Application to server-container assignment," *Proc. Comput. Sci.*, vol. 176, pp. 3863–3872, Jan. 2020.

[8] F. Chen, X. Zhou, and C. Shi, "The container deployment strategy based on stable matching," in *Proc. IEEE 4th Int. Conf. Cloud Comput. Big Data Anal. (ICCCBDA)*, Apr. 2019, pp. 215–221.

[9] J. V. Wang, K.-Y. Fok, C.-T. Cheng, and C. K. Tse, "A stable matching-based virtual machine allocation mechanism for cloud data centers," in *Proc. IEEE World Congr. Services (SERVICES)*, Jul. 2016, pp. 103–106.

[10] J. Ren, F. Xia, X. Chen, J. Liu, M. Hou, A. Shehzad, N. Sultanova, and X. Kong, "Matching algorithms: Fundamentals, applications and challenges," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 5, no. 3, pp. 332–350, Apr. 2021.

[11] A. Alimudin and Y. Ishida, "Matching-updating mechanism: A solution for the stable marriage problem with dynamic preferences," *Entropy*, vol. 24, no. 2, p. 263, Feb. 2022.

[12] H. Aziz, P. Biró, T. Fleiner, S. Gaspers, R. de Haan, N. Mattei, and B. Rastegari, "Stable matching with uncertain pairwise preferences," *Theor. Comput. Sci.*, vol. 909, pp. 1–11, Mar. 2022.

[13] J. Chen, R. Niedermeier, and P. Skowron, "Stable marriage with multi-modal preferences," in *Proc. ACM Conf. Econ. Comput.*, Jun. 2018, pp. 269–286.

[14] J. Chen, P. Skowron, and M. Sorge, "Matchings under preferences: Strength of stability and tradeoffs," *ACM Trans. Econ. Comput.*, vol. 9, no. 4, pp. 1–55, Dec. 2021.

[15] S. Miyazaki and K. Okamoto, "Jointly stable matchings," in *Proc. Leibniz Int. Informat. (LIPIcs)*, vol. 92, 2017, pp. 1–56.

[16] V. Menon, *Making Decisions With Incomplete and Inaccurate Information*. Waterloo, ON, Canada: University of Waterloo, 2021.

[17] R. Bredereck, J. Chen, D. Knop, J. Luo, and R. Niedermeier, "Adapting stable matchings to evolving preferences," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 1830–1837.

[18] H. Aziz, P. Biró, S. Gaspers, R. D. Haan, N. Mattei, and B. Rastegari, "Stable matching with uncertain linear preferences," in *Proc. Int. Symp. Algorithmic Game Theory*. Cham, Switzerland: Springer, pp. 195–206, 2016.

[19] P. Biró, D. F. Manlove, and S. Mittal, "Size versus stability in the marriage problem," *Theor. Comput. Sci.*, vol. 411, nos. 16–18, pp. 1828–1841, Mar. 2010.

[20] E. M. Fenoaltea, I. B. Baybusinov, J. Zhao, L. Zhou, and Y.-C. Zhang, "The stable marriage problem: An interdisciplinary review from the physicist's perspective," *Phys. Rep.*, vol. 917, pp. 1–79, Jun. 2021.

[21] N. Wirth, *Algorithms & Data Structures*. Upper Saddle River, NJ, USA: Prentice-Hall, 1985.

[22] R. W. Irving, "Stable marriage and indifference," *Discrete Appl. Math.*, vol. 48, no. 3, pp. 261–272, Feb. 1994.

[23] T. Hößler, P. Schulz, E. A. Jorswieck, M. Simsek, and G. P. Fettweis, "Stable matching for wireless URLLC in multi-cellular, multi-user systems," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 5228–5241, Aug. 2020.

[24] A. Pratap, "Cyclic stable matching inspired resource provisioning for IoT-enabled 5G networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 11, pp. 12195–12205, Nov. 2022.

[25] H. Xu and B. Li, "Egalitarian stable matching for VM migration in cloud computing," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2011, pp. 631–636.

[26] A. Kella and G. Belalem, "VM live migration algorithm based on stable matching model to improve energy consumption and quality of service," in *Proc. CLOSER*, 2014, pp. 118–128.

[27] A. Alimudin and Y. Ishida, "Service-based container deployment on kubernetes using stable marriage problem," in *Proc. 6th Int. Conf. Frontiers Educ. Technol.*, Jun. 2020, pp. 164–167.

[28] B. M. Maggs and R. K. Sitaraman, "Algorithmic nuggets in content delivery," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 3, pp. 52–66, Jul. 2015.

[29] R. W. Irving, D. F. Manlove, and S. Scott, "The hospitals/residents problem with ties," in *Proc. Scandin. Workshop Algorithm Theory*. Berlin, Germany: Springer, 2000, pp. 259–271.

[30] U. Arshad, M. Aleem, G. Srivastava, and J. C.-W. Lin, "Utilizing power consumption and SLA violations using dynamic VM consolidation in cloud data centers," *Renew. Sustain. Energy Rev.*, vol. 167, Oct. 2022, Art. no. 112782.

[31] R. Sinha, N. Purohit, and H. Diwanji, "Energy efficient dynamic integration of thresholds for migration at cloud data centers," *Int. J. Control Automat. Special Issue Commun. Netw.*, vol. 1, pp. 44–49, Dec. 2011.

AKHMAD ALIMUDIN received the B.S. degree in computer science and the M.S. degree from the Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia, in 2010 and 2013, respectively. He is currently pursuing the Ph.D. degree in computer science with the Toyohashi University of Technology, Japan. His research interests include intelligent systems, machine learning, and computer network applications.

YOSHITERU ISHIDA is currently an Emeritus Professor in computer science. He received the Dr.-Eng. degree in applied mathematics and physics from Kyoto University, Japan, in 1986. He was a Professor with the Toyohashi University of Technology, from 1998 to 2022. His research interests include biological complexity typified by the immune systems, self-organization by a game theoretic approach, and qualitative theory on large-scale dynamic networks.

KOUTAROU SUZUKI received the B.S., M.S., and Ph.D. degrees from The University of Tokyo, in 1994, 1996, and 1999, respectively. He has been a Professor with the Toyohashi University of Technology, since 2018. He was with NTT Laboratories, from 1999 to 2018.