Vol. 37, No. 1 (2027) | DOI: 10.37190/ord/236718

OPEN ACCESS

Operations Research and Decisions

www.ord.pwr.edu.pl

OPERATIONS RESEARCH AND DECISIONS QUARTERLY

Latin Rectangles and Stable Matchings in Many-Sided Problems

Marcin Anholcer*[1]

[1] *Institute of Informatics and Electronic Economics, Poznań University of Economics and Business, Poznań, Poland*

* *Corresponding author, email address: marcin.anholcer@ue.poznan.pl*

Abstract

In 1962, Gale and Shapley proved that there exists a stable matching in every instance of the 2-Sided Stable Matching Problem. It was shown very quickly that it does not have to be true in the case of 3-Sided Stable Matching Problems (or more generally, Many-Sided Stable Matching Problems). A natural question was posed: What system of preferences guarantees stable matching in many-sided problems? We give a partial answer to this question, using Latin rectangles.

Keywords: *Many-Sided Matching, Stable Matching, Latin rectangle, Deferred Acceptance Algorithm*

MSC 2020: 05C70, 91B08, 91B10
JEL: C78

1. Introduction

In 1962, Gale and Shapley [12] considered the following (Two-Sided) Stable Matching Problem. Assume two n-element sets of men M and women W are given. Each woman $w \in W$ has a preference relation (i.e. linear order) $\succ_w$ on the set M and each man $m \in M$ has a preference relation $\succ_m$ on the set W, where $m_1 \succ_w m_2$ means that the woman w prefers m_1 over m_2 (similarly $w_1 \succ_m w_2$ means that m prefers w_1 over w_2). A matching $\mu : M \cup W \to M \cup W$ is a function such that for every $m \in M$, $\mu(m) \in W$, for each $w \in W$, $\mu(w) \in M$ and for each $x \in M \cup W$, $\mu(\mu(x)) = x$. A *blocking pair* is a pair (m, w) for which we have

$$w \succ_m \mu(m) \wedge m \succ_w \mu(w).$$

A matching μ is stable if no blocking pair exists. Gale and Shapley proved that every instance of the Two-Sided Stable Matching Problem has at least one stable solution (matching).

The problem may be generalized to any number of sets as follows. Given a family of $s+1$ mutually disjoint sets $S_0, S_1, S_2, \ldots, S_s$, where $s \geq 1$, for every $j = 0, 1, \ldots, s$ agent $x \in S_j$ has a preference

Received 3 August 2025, accepted 12 September 2026, published online 12 September 2026
ISSN 2391-6060 (Online)/© 2026 Authors

relation $\succ_x$ on the set of all $(s+1)$-tuples $(x_0, x_1, \ldots, x_s)$ such that $x_j = x$, where $x_i \in S_i$, $i = 0, 1, \ldots s$. A matching

$$\mu : \bigcup_{j=0}^{s} S_j \to \bigotimes_{i=0}^{s} S_i$$

is a function such that for every $j = 0, 1, \ldots, s$, every $x \in S_j$, $x \in \mu(x)$ and for each

$$x \in S_i, y \in S_j, i \neq j,$$

we have $x \in \mu(y) \Leftrightarrow y \in \mu(x)$. A *blocking tuple* is an $(s+1)$-tuple $x = (x_0, x_1, \ldots, x_s)$ for which we have

$$x \succ_{x_j} \mu(x_j) \text{ for } j = 0, 1, \ldots, s.$$

A stable matching is a matching for which no blocking tuple exists.

The 2-Sided Matching Problem can be applied to model a situation where two groups of agents must be matched, and their preferences must be considered. Already in the first paper by Gale and Shapley [12], the problem of college admissions has been considered. Another example is the housing market considered by Shapley and Scarf in [24].

The above-mentioned generalization to many dimensions may be applied, for example, for matching companies with employees and technologies [18]. There are also possible technical applications: Cui and Jia in [9] applied such a model to solve the problem of connecting the data sources, servers, and final users in a computer network, while Ren et al. [23] applied it in mobile edge computing. Also, other allocation problems involving three or more groups of agents have been considered, such as student-project assignments [1]. Yet another field where the many-sided problems may be applied is organizing the project teams. This possible application will be discussed in more detail at the end of Section 2.

Although it was proved in [12] that a stable matching always exists in the case of two-sided settings, Knuth [17] showed that the problem of finding a stable matching changes dramatically when one switches from two to more dimensions, and Alkan [2] proved that stable matchings in three-sided systems may not exist in general. Knuth conjectured that a stable matching exists in every three-sided problem with lexicographically cyclic preferences. Boros et al. in [7] disproved this conjecture by presenting a few counterexamples. They showed, however, that the conjecture is true for purely cyclic preferences if the number n of agents in every group is not greater than the number of groups $s + 1$. Eriksson et al. in [11] extended this result to the case when $n = 4$ and $s = 2$, which Hofbauer improved [13] to the case of $n \leq s+2$. Later, Pashkovich and Poirrie [22] proved the case $n \leq 5$ for $s = 2$. Finally, Lam and Plaxton [19] showed that in general the stable matching in a cyclic preference system does not need to exist when $s \geq 2$, and the minimum size of a counterexample for $s = 2$ was recently reduced to $n = 20$ by Lerner [20].

The lexicographic preferences were analyzed, e.g., by Danilov [10], where the author proved that in such circumstances a stable matching must always exist, also for some special generalizations of the problem to more dimensions. Some variants of lexicographic preferences with additional restrictions were in turn studied by Lahiri [18]. The stable matching problem can also be extended by allowing indifference in the agents' preferences [15, 21].

The problem has been widely studied in the context of its computational complexity [8, 16, 21]. In

particular, Biró and McDermid in [6], and independently Huang in [14], showed that the problem of determining whether there exists a stable matching in a three-sided system with lexicographically cyclic preferences is NP-complete when the list of preferences is incomplete or a stronger stability criterion is considered.

Zhang and Zhong [25] wrote a paper about two modifications of the cyclic preferences. Soon after that, Arenas and Torres-Martínez [5], by indicating some gaps in the proofs, showed that one of the proposed settings does not need to allow a stable solution. Actually, the considered structure cannot guarantee a stable solution since it is less restrictive than cyclic preferences, for which a stable matching does not always exist, as Lam and Plaxton [19] showed. However, a stable matching always exists for the other proposed configuration and even for its generalization to more dimensions, as shown recently by Anholcer and Bartkowiak [4].

In [10], Danilov proved the existence of stable matching in every three-sided system with a special kind of lexicographic preferences and generalized it to many-sided problems with preferences that can be formed sequentially. A more detailed description of this system of conditions can be found in Section 4. Lahiri [18] considered a special case of such systems. In [3], a Three-Sided Stable Matching Problem with another type of lexicographic preferences was considered. Namely, the author proved the special case of Theorem 1 for the case when $s = 2$. In this paper, we generalize this result to $(s + 1)$-Sided Stable Matching Problems for arbitrary $s \geq 1$.

In Section 2, we define the concept of a preference system compatible with a Latin rectangle and present the paper's main result. In section 3, we constructively prove the main theorem and discuss the properties of the associated matching algorithm. The last Section 4 consists of some final remarks, particularly a few open problems for further research.

2. Preference Systems and Latin Rectangles

Before formulating this paper's main result, we must recall the concept of a Latin rectangle.

Definition 1. Latin rectangle $L = L(m, s)$, where $m \leq s$, is a matrix $L = [l_{ij}]_{m\times s}$ with the entries from the set $\{1, 2, \ldots, s\}$, each appearing exactly m times, such that

$$l_{ij_1} \neq l_{ij_2}$$

for every $i = 1, 2, \ldots, m$ and every $1 \leq j_1 < j_2 \leq s$ and

$$l_{i_1 j} \neq l_{i_2 j}$$

for every $j = 1, 2, \ldots, s$ and every $1 \leq i_1 < i_2 \leq m$, i.e., every row and every column consists of distinct elements.

Note that every row of a Latin rectangle is actually a permutation of the set $\{1, \ldots, s\}$. Let us define a special kind of preferences, based on Latin rectangles.

Definition 2. Given a Latin rectangle $L = L(m, s)$, $m \leq s$, and $s + 1$ n-element sets $S_0, S_1, \ldots, S_s$, the system of preferences is said to be compatible with L if it has the following properties.

P1 The set S_0 can be partitioned into m disjoint nonempty subsets $S_{0,1}, S_{0,2}, \ldots, S_{0,m}$ such that for every $i = 1, \ldots, m$ and for every $x_0 \in S_{0,i}$ there are preference relations $\succ^j_{x_0}$, $j = 1, \ldots, s$ defined on the sets $S_{l_{ij}}$ such that for any $x^1_{l_{ij}}, x^2_{l_{ij}} \in S_{l_{ij}}$ we have

- for $j = 1$:
$$\left\{x^1_{l_{i1}} \succ^1_{x_0} x^2_{l_{i1}}\right\} \Rightarrow (x^1_0, x^1_1, \ldots, x^1_s) \succ_{x_0} (x^2_0, x^2_1, \ldots, x^2_s)$$
- for $j > 1$:
$$\left\{x^1_{l_{ij'}} = x^2_{l_{ij'}}, j' < j\right\} \wedge \left\{x^1_{l_{ij}} \succ^j_{x_0} x^2_{l_{ij}}\right\} \Rightarrow (x^1_0, x^1_1, \ldots, x^1_s) \succ_{x_0} (x^2_0, x^2_1, \ldots, x^2_s)$$

for every
$$(x^1_0, x^1_1, \ldots, x^1_s), (x^2_0, x^2_1, \ldots, x^2_s) \in \bigotimes_{i=0}^{s} S_i$$
such that $x^1_0 = x^2_0 = x_0$, i.e. every agent x_0 from set $S_{0,i}$ cares mainly about the agents from the set $S_{l_{i1}}$, next about the agents from $S_{l_{i2}}$ (which means that if in two given tuples there is the same agent from the set $S_{l_{i1}}$, then x_0 orders these tuples based on their preferences on the agents from $S_{l_{i2}}$) and so on. In other words, the order of preferences of all the agents from the set $S_{0,i}$ is the same as the order of the indices j of sets S_j in the row i of L.

P2 For every $j = 1, \ldots, s$ and for every $x_j \in S_j$ there is a preference relation $\succ^\star_{x_j}$ defined on the set S_0 such that for any $x^1_0, x^2_0 \in S_0$ we have
$$x^1_0 \succ^\star_{x_j} x^2_0 \Rightarrow (x^1_0, x^1_1, \ldots, x^1_s) \succ_{x_j} (x^2_0, x^2_1, \ldots, x^2_s)$$
for every
$$(x^1_0, x^1_1, \ldots, x^1_s), (x^2_0, x^2_1, \ldots, x^2_s) \in \bigotimes_{i=0}^{s} S_i$$
such that $x^1_j = x^2_j = x_j$, i.e., the agent x_j cares mainly about the agents from the set S_0 and then about the remaining ones.

P3 For every $j = 1, \ldots, s$ and for every $x_j \in S_j$, the preference relation $\succ^\star_{x_j}$ satisfies the following condition. Assume that we have $1 \leq j_1 < j_2 \leq s$ and let k_1 and k_2, $1 \leq k_1 \leq m$, $1 \leq k_2 \leq m$, be such indices that $l_{k_1 j_1} = l_{k_2 j_2} = j$. Then
$$\forall_{x^{k_1}_0 \in S_{0,k_1}} \forall_{x^{k_2}_0 \in S_{0,k_2}} x^{k_1}_0 \succ^\star_{x_j} x^{k_2}_0,$$
i.e. x_j prefers every agent from S_{0,k_1} to every agent from S_{0,k_2}, whenever entry j of L has lower second index in row k_1 than in row k_2 or in other words every $x_j \in S_j$ orders the subsets $S_{0,i}$ of S_0 in the same way as the first indices i of the number j are ordered in the columns of L.

Property P3 in the above definition can also be explained in the following way. Every agent $x_0 \in S_0$ has a preference order over the sets $S_1, S_2, \ldots, S_s$. Property P3 says that every $x_j \in S_j$ prefers the agents from the set $S_{0,k_1} \subset S_0$ to the agents from the set $S_{0,k_2} \subset S_0$ if and only if the set S_j occurs earlier on the preference list of every agent $x^{k_1}_0 \in S_{0,k_1}$ than on the preference list of any agent $x^{k_2}_0 \in S_{0,k_2}$. Note that

property P3 can be satisfied only if P2 is satisfied: one cannot prefer one agent in S_0 to another if they do not have a preference order on S_0 that is consistent with the preference order on the set of all triples.

The following example illustrates the above properties. Let us consider five systems of preferences. In all the cases we assume that $s = 2$, $n = 2$, $S_0 = \{x_0^1, x_0^2\}$, $S_1 = \{x_1^1, x_1^2\}$ and $S_2 = \{x_2^1, x_2^2\}$.

System 1.

$$
\begin{array}{l}
(x_0^1, x_1^2, x_2^1) \succ_{x_0^1} (x_0^1, x_1^2, x_2^2) \succ_{x_0^1} (x_0^1, x_1^1, x_2^2) \succ_{x_0^1} (x_0^1, x_1^1, x_2^1),\\
(x_0^2, x_1^2, x_2^2) \succ_{x_0^2} (x_0^2, x_1^2, x_2^1) \succ_{x_0^2} (x_0^2, x_1^1, x_2^1) \succ_{x_0^2} (x_0^2, x_1^1, x_2^2),\\
(x_0^2, x_1^1, x_2^1) \succ_{x_1^1} (x_0^2, x_1^1, x_2^2) \succ_{x_1^1} (x_0^1, x_1^1, x_2^2) \succ_{x_1^1} (x_0^1, x_1^1, x_2^1),\\
(x_0^1, x_1^2, x_2^2) \succ_{x_1^2} (x_0^1, x_1^2, x_2^1) \succ_{x_1^2} (x_0^2, x_1^2, x_2^1) \succ_{x_1^2} (x_0^2, x_1^2, x_2^2),\\
(x_0^2, x_1^2, x_2^1) \succ_{x_2^1} (x_0^1, x_1^1, x_2^1) \succ_{x_2^1} (x_0^2, x_1^1, x_2^1) \succ_{x_2^1} (x_0^1, x_1^2, x_2^1),\\
(x_0^2, x_1^1, x_2^2) \succ_{x_2^2} (x_0^1, x_1^2, x_2^2) \succ_{x_2^2} (x_0^1, x_1^1, x_2^2) \succ_{x_2^2} (x_0^2, x_1^2, x_2^2).
\end{array}
$$

System 2.

$$
\begin{array}{l}
(x_0^1, x_1^1, x_2^1) \succ_{x_0^1} (x_0^1, x_1^1, x_2^2) \succ_{x_0^1} (x_0^1, x_1^2, x_2^1) \succ_{x_0^1} (x_0^1, x_1^2, x_2^2),\\
(x_0^2, x_1^1, x_2^2) \succ_{x_0^2} (x_0^2, x_1^2, x_2^2) \succ_{x_0^2} (x_0^2, x_1^1, x_2^1) \succ_{x_0^2} (x_0^2, x_1^2, x_2^1),\\
(x_0^1, x_1^1, x_2^1) \succ_{x_1^1} (x_0^1, x_1^1, x_2^2) \succ_{x_1^1} (x_0^2, x_1^1, x_2^2) \succ_{x_1^1} (x_0^2, x_1^1, x_2^1),\\
(x_0^1, x_1^2, x_2^2) \succ_{x_1^2} (x_0^1, x_1^2, x_2^1) \succ_{x_1^2} (x_0^2, x_1^2, x_2^1) \succ_{x_1^2} (x_0^2, x_1^2, x_2^2),\\
(x_0^2, x_1^2, x_2^1) \succ_{x_2^1} (x_0^2, x_1^1, x_2^1) \succ_{x_2^1} (x_0^1, x_1^1, x_2^1) \succ_{x_2^1} (x_0^1, x_1^2, x_2^1),\\
(x_0^2, x_1^2, x_2^2) \succ_{x_2^2} (x_0^2, x_1^1, x_2^2) \succ_{x_2^2} (x_0^1, x_1^2, x_2^2) \succ_{x_2^2} (x_0^1, x_1^1, x_2^2).
\end{array}
$$

System 3.

$$
\begin{array}{l}
(x_0^1, x_1^1, x_2^1) \succ_{x_0^1} (x_0^1, x_1^1, x_2^2) \succ_{x_0^1} (x_0^1, x_1^2, x_2^2) \succ_{x_0^1} (x_0^1, x_1^2, x_2^1),\\
(x_0^2, x_1^2, x_2^1) \succ_{x_0^2} (x_0^2, x_1^1, x_2^1) \succ_{x_0^2} (x_0^2, x_1^1, x_2^2) \succ_{x_0^2} (x_0^2, x_1^2, x_2^2),\\
(x_0^2, x_1^1, x_2^1) \succ_{x_1^1} (x_0^2, x_1^1, x_2^2) \succ_{x_1^1} (x_0^1, x_1^1, x_2^1) \succ_{x_1^1} (x_0^1, x_1^1, x_2^2),\\
(x_0^2, x_1^2, x_2^2) \succ_{x_1^2} (x_0^2, x_1^2, x_2^1) \succ_{x_1^2} (x_0^1, x_1^2, x_2^2) \succ_{x_1^2} (x_0^1, x_1^2, x_2^1),\\
(x_0^1, x_1^1, x_2^1) \succ_{x_2^1} (x_0^1, x_1^2, x_2^1) \succ_{x_2^1} (x_0^2, x_1^2, x_2^1) \succ_{x_2^1} (x_0^2, x_1^1, x_2^1),\\
(x_0^1, x_1^2, x_2^2) \succ_{x_2^2} (x_0^1, x_1^1, x_2^2) \succ_{x_2^2} (x_0^2, x_1^1, x_2^2) \succ_{x_2^2} (x_0^2, x_1^2, x_2^2).
\end{array}
$$

System 4.

$$
\begin{array}{l}
(x_0^1, x_1^2, x_2^1) \succ_{x_0^1} (x_0^1, x_1^1, x_2^2) \succ_{x_0^1} (x_0^1, x_1^1, x_2^1) \succ_{x_0^1} (x_0^1, x_1^2, x_2^2),\\
(x_0^2, x_1^2, x_2^2) \succ_{x_0^2} (x_0^2, x_1^1, x_2^1) \succ_{x_0^2} (x_0^2, x_1^2, x_2^1) \succ_{x_0^2} (x_0^2, x_1^1, x_2^2),\\
(x_0^1, x_1^1, x_2^1) \succ_{x_1^1} (x_0^1, x_1^1, x_2^2) \succ_{x_1^1} (x_0^2, x_1^1, x_2^1) \succ_{x_1^1} (x_0^2, x_1^1, x_2^2),\\
(x_0^1, x_1^2, x_2^2) \succ_{x_1^2} (x_0^1, x_1^2, x_2^1) \succ_{x_1^2} (x_0^2, x_1^2, x_2^2) \succ_{x_1^2} (x_0^2, x_1^2, x_2^1),\\
(x_0^2, x_1^1, x_2^1) \succ_{x_2^1} (x_0^2, x_1^2, x_2^1) \succ_{x_2^1} (x_0^1, x_1^2, x_2^1) \succ_{x_2^1} (x_0^1, x_1^1, x_2^1),\\
(x_0^2, x_1^1, x_2^2) \succ_{x_2^2} (x_0^2, x_1^2, x_2^2) \succ_{x_2^2} (x_0^1, x_1^1, x_2^2) \succ_{x_2^2} (x_0^1, x_1^2, x_2^2).
\end{array}
$$

System 5.

$$
\begin{aligned}
&(x_0^1, x_1^1, x_2^1) \succ_{x_0^1} (x_0^1, x_1^1, x_2^2) \succ_{x_0^1} (x_0^1, x_1^2, x_2^1) \succ_{x_0^1} (x_0^1, x_1^2, x_2^2),\\
&(x_0^2, x_1^2, x_2^1) \succ_{x_0^2} (x_0^2, x_1^1, x_2^1) \succ_{x_0^2} (x_0^2, x_1^2, x_2^2) \succ_{x_0^2} (x_0^2, x_1^1, x_2^2),\\
&(x_0^2, x_1^1, x_2^1) \succ_{x_1^1} (x_0^1, x_1^1, x_2^1) \succ_{x_1^1} (x_0^1, x_1^1, x_2^2) \succ_{x_1^1} (x_0^2, x_1^1, x_2^2),\\
&(x_0^2, x_1^2, x_2^2) \succ_{x_1^2} (x_0^2, x_1^2, x_2^1) \succ_{x_1^2} (x_0^1, x_1^2, x_2^1) \succ_{x_1^2} (x_0^1, x_1^2, x_2^2),\\
&(x_0^1, x_1^2, x_2^1) \succ_{x_2^1} (x_0^2, x_1^1, x_2^1) \succ_{x_2^1} (x_0^1, x_1^1, x_2^1) \succ_{x_2^1} (x_0^2, x_1^2, x_2^1),\\
&(x_0^2, x_1^2, x_2^2) \succ_{x_2^2} (x_0^1, x_1^1, x_2^2) \succ_{x_2^2} (x_0^2, x_1^1, x_2^2) \succ_{x_2^2} (x_0^1, x_1^2, x_2^2).
\end{aligned}
$$

Observe that System 1 does not have the desired properties. Indeed, none of the agents from S_2 cares mainly about the agents from S_0 or S_1, thus property P2 is not satisfied, which also means that property P3 cannot be satisfied. One may also quickly verify that property P1 does not hold, even if S_1 is considered instead of S_0. This does not mean, however, that no matching exists (see the discussion in the last section).

Let us consider System 2. Let $S_{0,1} = \{x_0^1\}$ and $S_{0,2} = \{x_0^2\}$. As one can easily check, the only agent from $S_{0,1}$ cares mainly about the agents from S_1, and then about the agents from S_2 (so, we can say that their preference order over the sets is (S_1, S_2)). Similarly, the only agent from $S_{0,2}$ cares mainly about the agents from S_2, and then about the agents from S_1 (so, we can say that their preference order over the sets is (S_2, S_1)). Thus, Property P1 is satisfied. One can also verify that all the agents from S_1 and S_2 care mainly about the agents from S_0, so Property P2 is satisfied. Finally, observe that the agents from S_1 prefer the agent x_0^1 to the agent x_0^2, which is consistent with the fact that the set S_1 occurs earlier on the preference order list of the agent x_0^1 (first position) than on the list of agent x_0^2 (second position). Analogously, the agents from S_2 prefer the agent x_0^2 to the agent x_0^1, which is consistent with the fact that the set S_2 occurs earlier on the preference order list of the agent x_0^2 (first position) than on the list of agent x_0^1 (second position). It means that Property P3 is also satisfied, and

$$L = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}.$$

Now, let us analyze System 3. Obviously, Property P2 holds. Note that one can switch the indices of sets S_1 and S_2, since the agents in these sets care primarily about the members of S_0, but the agents from S_0 do not care mainly about the same set: x_0^1 cares primarily about S_1, while x_0^2 cares mainly about S_2. Thus, S_0 cannot be substituted by any other set. Now it is easy to see that Property P3 does not hold. If it did, based on the preferences of the members of S_0 we would deduce that L would be the same as in the case of System 2. That would mean in particular, that the agents from S_1 should prefer x_0^1 to x_0^2, since S_1 appears at the first position on the order list of x_0^1, while S_2 appears at the first position on the order list of x_0^2. However, they prefer x_0^2 to x_0^1.

A similar analysis shows that System 4 satisfies properties P2 and P3, but not P1, while System 5 satisfies P1, but violates P2 and P3. In both cases, the only possible candidate for the set satisfying P1 is

S_0, and in both cases, the only possible hypothetical Latin rectangle that the preference system could be compatible with is L defined above.

Let us present another possible application of the many-sided matching problem. Assume that several projects are running in a company. Each project team must consist of a project manager and some other members. Assume that the company creates Web applications. Then the possible members may be frontend developers, backend developers, graphic designers, database administrators, etc. Of course, the members of each group have preferences over the possible project teams. Assume that each employee must be assigned to exactly one project team. Now, let us consider the system of preferences compatible with a Latin rectangle. In the discussed example of an IT company, the interpretation of the properties of the preference system would be as follows. Property P1 means that each project manager has a lexicographic preference order over the set of all the other types of employees, i.e., thinks that the most important for the project's success is the type of employee, then the employees themselves. For example, some project managers may think that for the kind of project that they were assigned, the most important thing is to pick a good database administrator, then the developers, then the graphic designer, and so on. Property P2 means that every employee cares mostly about the project manager, and the rest of the team is not that important. Finally, Property P3 means that each employee prefers the project managers who care more about their role. For example, assume that manager x is primarily concerned about the developers (position 1 on their preference order list), then the database administrators (position 2), and finally the graphic designers (position 3). At the same time, manager y thinks that the graphic designers are the most important (position 1), then the developers (position 2), and finally the database administrators (position 3). Then, all the developers would prefer x to y (position 1 vs. 2), all the graphic designers y to x (position 1 vs. 3), and all the database administrators would prefer x to y (position 2 vs 3).

The main result of this paper reads as follows. We are going to prove it in the following section.

Theorem 1. Assume that we are given $s+1$ sets $S_0, \ldots, S_s$ with their respective systems of preferences. If there exists a Latin rectangle $L(m, s)$, $m \leq s$, compatible with this system of preferences, then a stable matching exists.

3. Proof of Theorem 1

We will use the fact that Gale and Shapley proved in [12] that a stable matching exists in every two-sided system. The proof will consist of two parts. First, we are going to define the matching algorithm. Then we will show that no blocking $(s+1)$-tuple exists, thus, the resulting matching is stable.

Let us consider the two-sided problem with the sets S_1 and S_2 such that $n_1 = |S_1| < |S_2| = n_2$. Obviously, no complete (or perfect) matching exists in such a setting, but one can consider partial matchings (or simply matchings), where $n_2 - n_1$ agents from S_2 remain unmatched. Such a matching will be called stable if there is no blocking pair among all the agents, where we assume that being assigned to anyone is always preferred to remaining unmatched. Suppose we add to the set S_1 $n_2 - n_1$ fictional agents, being least preferred for every agent from S_2. In that case, it is easy to see that every stable complete matching in such an extended problem uniquely defines a stable matching in the initial problem (its restriction to the pairs including the members of S_1). More than one such matching may exist. If this is the case, then we choose the Pareto-optimal one from the point of view of the agents from S_1. It can be obtained by

the Gale-Shapley deferred acceptance algorithm by taking S_1 to be the proposing party and S_2 to be the accepting (rejecting) one. In particular, in such a matching, no agent $x \in S_1$ is assigned to some $y_2 \in S_2$ if there is an unmatched $y_1 \in S_2$ such that $y_1 \succ_x y_2$.

Now we are going to present the matching method in the many-sided problem. The procedure consists of s stages.

At the first stage, for every i, $1 \leq i \leq m$, we match the elements of $S_{0,i}$ with $s_i = |S_{0,i}|$ members of $S_{l_{i1}}$, i.e. with s_i elements of the first set on the list of preferences of the agents in $S_{0,i}$. As the agents from $S_{0,i}$ care mainly about the agents from $S_{l_{i1}}$ and the agents from $S_{l_{i1}}$ prefer the agents from $S_{0,i}$ to the agents from all the remaining subsets of S_0, we obtain a two-sided matching problem with non-equal numbers of agents with the preferences defined in the natural way. The algorithm returns the stable matching that is (Pareto) optimal from the point of view of the agents $x_i \in S_{0,i}$. It follows directly from the fact that the set of proposers in Algorithm 1 (which is called at every iteration of Algorithm 2) is its first argument X with preferences defined as the preferences of $x_i \in S_{0,i}$. On the other hand, we know that the original Deferred Acceptance Algorithm of Gale and Shapley [12] returns the men-optimal solution.

Before every stage j, where $2 \leq j \leq s$, the j-tuples are matched. As for every i, each agent $x_i \in S_{0,i}$ is matched with the agents from $j-1$ sets they care about the most, one can treat each such j-tuple as one agent with preference on the remaining sets defined exactly as the preferences of x_i. On the other hand, we have j^{th} set on the list of the preferences order of x_i, namely $S_{l_{ij}}$, where all the agents care mostly about the agents from S_0 (in particular, about the agents from $S_{0,i}$). This defines a two-sided problem. We choose the (partial) stable matching, that is (Pareto) optimal from the point of view of the agents $x_i \in S_{0,i}$. The matchings obtained for all values of i define the $(j+1)$-tuples. After stage s we obtain an $(s+1)$-sided complete matching. The described method is summarized as Algorithm 2 (an illustrative example showing the performance of the algorithm has been presented at the end of this section). For the sake of completeness, firstly, we recall the version of the Gale-Shapley deferred acceptance algorithm that allows the matching of tuples (cf. [4, Algorithm 2]).

Algorithm 1. Rephrased Gale-Shapley Deferred Acceptance Algorithm (GS)

Input: System X, Y of a set of disjoint tuples $X \subseteq X_1 \times X_2 \times \cdots \times X_j$ and a set of agents Y, $|X| \leq |Y|$, with appropriately defined mutual preferences.

Output: Stable matching (partial if $|X| < |Y|$) $\mu_{X \cup Y'}$ of the members of X and $Y' \subseteq Y$, $|Y'| = |X|$, being also a stable matching of the members of $X_1, X_2, \ldots, X_j, Y$, optimal from the point of view of the members of X.

Initialize: Mark all elements of $X \cup Y$ as available for matching, i.e., set $\mu(x) \leftarrow null$ for every $x \in X \cup Y$; set $Y' \leftarrow \emptyset$;

while *there exists available element* $x \in X$ **do**

  Choose the most preferred tuple $(x, y) : y \in Y$ on x's preference list and cross it out from this list;

  if y *is available* **then**

    set $Y' \leftarrow Y' \cup \{y\}$;

    set $\mu_{X \cup Y'}(x) \leftarrow (x, y)$, $\mu_{X \cup Y'}(y) \leftarrow (x, y)$, mark x and y as unavailable;

  else if y *prefers* (x, y) *over its current assignment* (x', y) **then**

    set $\mu(x) \leftarrow (x, y)$, $\mu(y) \leftarrow (x, y)$, $\mu(x') \leftarrow null$, mark x as unavailable, mark x' as available;

return $\mu = \mu_{X \cup Y'}$, $U = Y \setminus Y'$

Algorithm 2. Matching Algorithm for Latin Rectangle-Compatible Preferences

Input: s+1 sets S_j where $j \in \{0, 1, \ldots, s\}$, $|S_j| = n$. Preference lists satisfying properties P1, P2, and P3 from Definition 2, compatible with a Latin rectangle $L = [l_{ij}]_{m \times s}$.

Output: Stable matching μ of the members of $\bigcup_{j=0}^{s} S_j$.

Initialize: Mark all elements of $\bigcup_{j=0}^{s} S_j$ as available for matching, i.e., set $\mu(x) \leftarrow null$ for every $x \in \bigcup_{j=0}^{s} S_j$;

for $i = 1$ *to* m **do**

  $\mu_i \leftarrow S_{0,i}$; `// Initialize all the partial matchings`

end

for $j = 1$ *to* s **do**

  for $i = 1$ *to* m **do**

    $(\mu_i, S_{l_{ij}}) \leftarrow$ **Algorithm 1**$(\mu_i, S_{l_{ij}})$; `// Run GS algorithm on` μ_i `and` $S_{l_{ij}}$

  end

end

return $\mu = \bigcup_{i=1}^{m} \mu_i$; `// Return the complete matching` μ

We prove the stability of the solution μ produced by Algorithm 2 by contradiction. The point made repeatedly below is that every call to Algorithm 1 within Algorithm 2 returns a stable partial two-sided matching between the current tuples and the agents of the next set. Thus, after any such call, no current tuple and no agent available in the corresponding two-sided problem can both improve by being matched to each other.

Assume that $y = (x_0, x_1, \ldots, x_s)$ is a blocking tuple in the final matching, where $x_0 \in S_{0,i}$ for some i. Without loss of generality, reindex the rows of L and the sets $S_1, \ldots, S_s$ so that $i = 1$ and $l_{1j} = j$ for $j = 1, \ldots, s$. Write

$$\mu(x_0) = (x_0, x_1^0, x_2^0, \ldots, x_s^0)$$

and, for $r = 1, \ldots, s$,

$$\mu(x_r) = (x_0^r, x_1^r, \ldots, x_{r-1}^r, x_r, x_{r+1}^r, \ldots, x_s^r).$$

Since $y \succ_{x_0} \mu(x_0)$, Property P1 implies that there is a first index

$$k = \min\{r \in \{1, \ldots, s\} : x_r \neq x_r^0\}$$

such that $x_r = x_r^0$ for all $r < k$ and $x_k \succ_{x_0}^{k} x_k^0$.

If $k = 1$, then x_0 and x_1 are not matched to each other under μ. Since y is blocking, x_1 also prefers y to $\mu(x_1)$; by Property P2 this gives $x_0 \succ_{x_1}^{\star} x_0^1$. At the first stage, Algorithm 1 matches the elements of $S_{0,1}$ with agents from S_1. The element x_0 prefers x_1 to its assigned partner x_1^0 by Property P1, and x_1 prefers x_0 to the S_0-component of its final assignment by Property P2. If x_1 was assigned later to an agent from another subset of S_0, then Property P3 only strengthens this conclusion, because agents from S_1 rank $S_{0,1}$ before subsets in whose row S_1 appears later. Thus, (x_0, x_1) would block the stable partial matching obtained at the first stage, a contradiction.

It remains to consider $k = 2, \ldots, s$. At the beginning of stage k, the current tuple containing x_0 already contains the same agents $x_1, \ldots, x_{k-1}$ as the alleged blocking tuple y. By Property P1, this tuple therefore prefers x_k to the agent x_k^0 assigned to it by Algorithm 2. Because y is blocking, Property P2 gives $x_0 \succ_{x_k}^{\star} x_0^k$ unless x_k was unmatched in the relevant two-sided problem, in which case being matched is preferred to remaining unmatched.

If x_k had already been taken in an earlier stage, then it would have been matched to a tuple whose S_0-component belongs to a subset $S_{0,t}$ such that the set S_k appears earlier in row t of L than in row 1. Property P3 would then imply that x_k prefers every agent from $S_{0,t}$ to every agent from $S_{0,1}$, in particular to x_0, contradicting $x_0 \succ^{\star}_{x_k} x_0^k$. Hence, x_k was available for the two-sided problem solved at stage k for the tuple containing x_0. The tuple containing x_0 and the agent x_k would both prefer to be matched to each other, contradicting the stability of the partial matching returned by Algorithm 1 at that stage.

All possible values of k lead to a contradiction. Therefore, no blocking $(s+1)$-tuple exists, and the matching μ produced by Algorithm 2 is stable.

□

Since the proof presented above is constructive, let us give some remarks on Algorithm 2. It is straightforward to see that its running time is polynomial in n and s. Indeed, if the number of agents in every set is k, then the original Gale-Shapley method runs in the worst case in time $T(k) = (k-1)^2+1 = O(k^2)$ [12], and so does Algorithm 1. It is called $s \times m$ times from Algorithm 2, where the cardinalities of sets to be matched are precisely $s_i = |S_{0,i}|$ for $i = 1, \dots m$. This means that the time complexity of the inner loop of Algorithm 2 (for any fixed $j = 1, \dots, s$) is in the worst case

$$T_0 = \sum_{i=1}^{m} T(s_i) = m + \sum_{i=1}^{m} (s_i - 1)^2,$$

which means that the worst-case running time of Algorithm 2 is

$$T = sT_0 = s(m + \sum_{i=1}^{m} (s_i - 1)^2) = O(sn^2).$$

We conclude this section with a presentation of the performance of Algorithm 2. We will run it on System 2. For clarity, we present all partial matchings as complete systems of triples, possibly with some (not yet assigned) agents represented by the symbol "$\cdot$".

- Input: $n = 2$, $m = 2$, $s = 2$, $S_{0,1} = \{x_0^1\}$, $S_{0,2} = \{x_0^2\}$, $S_1 = \{x_1^1, x_1^2\}$ and $S_2 = \{x_2^1, x_2^2\}$, preferences defined as in System 2.
- Initialization: $\mu_1 = \{(x_0^1, \cdot, \cdot)\}$, $\mu_2 = \{(x_0^2, \cdot, \cdot)\}$
- Main loop, $j = 1$, $i = 1$: call Algorithm 1 with argument (μ_1, S_1). The output is the partial matching of $(x_0^1, \cdot, \cdot)$ with x_1^1 (preferred to x_1^2 by x_0^1) and x_1^1 is removed from S_1. Thus, the updated matchings and sets are $\mu_1 = \{(x_0^1, x_1^1, \cdot)\}$, $\mu_2 = \{(x_0^2, \cdot, \cdot)\}$, $S_1 = \{x_1^2\}$ and $S_2 = \{x_2^1, x_2^2\}$.
- Main loop, $j = 1$, $i = 2$: call Algorithm 1 with argument (μ_2, S_2). The output is the partial matching of $(x_0^2, \cdot, \cdot)$ with x_2^2 (preferred to x_2^1 by x_0^2) and x_2^2 is removed from S_2. Thus, the updated matchings and sets are $\mu_1 = \{(x_0^1, x_1^1, \cdot)\}$, $\mu_2 = \{(x_0^2, \cdot, x_2^2)\}$, $S_1 = \{x_1^2\}$ and $S_2 = \{x_2^1\}$.
- Main loop, $j = 2$, $i = 1$: call Algorithm 1 with argument (μ_1, S_2). The output is the partial matching of $(x_0^1, x_1^1, \cdot)$ with x_2^1 (the only remaining member of S_2) and x_2^1 is removed from S_2. Thus, the updated matchings and sets are $\mu_1 = \{(x_0^1, x_1^1, x_2^1)\}$, $\mu_2 = \{(x_0^2, \cdot, x_2^2)\}$, $S_1 = \{x_1^2\}$ and $S_2 = \emptyset$.

- Main loop, $j = 2$, $i = 2$: call Algorithm 1 with argument (μ_2, S_1). The output is the partial matching of $(x_0^2, \cdot, x_2^2)$ with x_1^2 (the only remaining member of S_1) and x_1^2 is removed from S_1. Thus, the updated matchings and sets are $\mu_1 = \{(x_0^1, x_1^1, x_2^1)\}$, $\mu_2 = \{(x_0^2, x_1^2, x_2^2)\}$, $S_1 = \emptyset$ and $S_2 = \emptyset$.
- Return the union of all the matchings μ_i, i.e., the complete matching $\mu = \{(x_0^1, x_1^1, x_2^1), (x_0^2, x_1^2, x_2^2)\}$.

4. Final Remarks

As one can check, the properties described in Theorem 1, although lexicographic, are independent from the ones described by Danilov in [10]. Namely, let us consider the preference systems 1 and 2 again. Danilov considers three-sided systems of lexicographic preferences, where the agents from set S_0 care mainly about agents from S_1 and vice versa (with some further generalizations). It is straightforward to see that System 1 satisfies the conditions imposed by Danilov and, as we already know, does not satisfy the assumptions of Theorem 1. On the other hand, System 2 satisfies the assumptions of Theorem 1, but it does not meet the conditions of Danilov: the agents from the sets S_1 and S_2 care mainly about the agents from S_0, but the agents from S_0 do not care primarily about the agents of any of the sets S_1 and S_2. Indeed, for one of them, the agents from S_1 are more critical, while for the other, the agents from S_2 are.

Observe that in a special case of two-sided matching, we have

$$L = \begin{bmatrix} 1 \end{bmatrix},$$

while a preference system similar to the one considered by Danilov is, in turn, compatible with

$$L = \begin{bmatrix} 1 & 2 \end{bmatrix}.$$

The only difference is that here, the agents from S_2 also have lexicographic preferences, caring mostly about the agents from S_0.

The presented result defines new sufficient conditions for the Many-Sided Stable Matching Problem to have a stable solution. As mentioned in the introduction, other sufficient conditions were also discussed. However, what sufficient and necessary conditions would look like is still unknown. The only exception is the case $s = 1$, i.e., the case of the Two-Sided Stable Matching Problem, where every preference system trivially has a stable solution. If $s > 1$, one can easily find examples of the preference systems that do not fulfill any of the presented conditions and possess a stable matching despite it. Thus, the following seems to be a significant open problem.

Problem 1. What are the necessary conditions of the existence of a stable matching in a $(s+1)$-Sided Stable Matching Problem for arbitrary $s > 1$?

One could ask if using all the assumptions in this article, in particular the ones including Latin rectangles, is necessary, i.e., if it could be possible to skip some of the properties in Definition 2. As we already observed, System 1 does not fulfil any of P1, P2, P3, but since it satisfies Danilov's assumptions, it has a stable matching. It comes out that it would not be enough if only Property P2 holds, as the example

of System 3 shows. We already know that this system does not satisfy Property P3. Moreover, there are four possible matchings:

- $\{(x_0^1, x_1^1, x_2^1), (x_0^2, x_1^2, x_2^2)\}$ with blocking triple (x_0^2, x_1^1, x_2^2),
- $\{(x_0^1, x_1^1, x_2^2), (x_0^2, x_1^2, x_2^1)\}$ with blocking triple (x_0^1, x_1^1, x_2^1),
- $\{(x_0^1, x_1^2, x_2^1), (x_0^2, x_1^1, x_2^2)\}$ with blocking triple (x_0^1, x_1^2, x_2^2),
- $\{(x_0^1, x_1^2, x_2^2), (x_0^2, x_1^1, x_2^1)\}$ with blocking triple (x_0^2, x_1^2, x_2^1).

We conclude that no stable matching exists in System 3. The fulfillment of properties P2 and P3 with the violation of P1 is also not enough, as System 4 shows. This time the blocking triples are:

- (x_0^1, x_1^2, x_2^1) for the matching $\{(x_0^1, x_1^1, x_2^1), (x_0^2, x_1^2, x_2^2)\}$,
- (x_0^2, x_1^2, x_2^2) for the matching $\{(x_0^1, x_1^1, x_2^2), (x_0^2, x_1^2, x_2^1)\}$,
- (x_0^2, x_1^1, x_2^1) for the matching $\{(x_0^1, x_1^2, x_2^1), (x_0^2, x_1^1, x_2^2)\}$,
- (x_0^1, x_1^1, x_2^2) for the matching $\{(x_0^1, x_1^2, x_2^2), (x_0^2, x_1^1, x_2^1)\}$.

Finally, note that a stable matching need not exist when only P1 is satisfied, but P2 and P3 are not, as shown by System 5. Also, in this case, there is a blocking triple for every possible matching:

- (x_0^2, x_1^1, x_2^1) for the matching $\{(x_0^1, x_1^1, x_2^1), (x_0^2, x_1^2, x_2^2)\}$,
- (x_0^1, x_1^1, x_2^1) for the matching $\{(x_0^1, x_1^1, x_2^2), (x_0^2, x_1^2, x_2^1)\}$,
- (x_0^1, x_1^1, x_2^2) for the matching $\{(x_0^1, x_1^2, x_2^1), (x_0^2, x_1^1, x_2^2)\}$,
- (x_0^1, x_1^2, x_2^1) for the matching $\{(x_0^1, x_1^2, x_2^2), (x_0^2, x_1^1, x_2^1)\}$.

Let us consider the following generalization of the Many-Sided Stable Matching Problem. Some agents may be unacceptable for other agents (in other words, some matchings are forbidden). The question is, how would it influence the existence of the stable matching? Of course, the stability should be redefined: one must assume the feasibility of a matching in addition to the nonexistence of a blocking $(s+1)$-tuple. The problem is that if too many pairs of agents cannot be members of the same $(s+1)$-tuple, no matching is possible. On the other hand, if all the $(s+1)$-tuples with forbidden agents are placed at the end of the preference lists, then the system of preferences will possibly not satisfy some of the properties imposed in Definition 2. Thus, the following open problem arises.

Problem 2. What conditions imposed on a preference system guarantee that the properties from Definition 2 are still satisfied despite forbidden pairs of agents?

One can also consider a generalization of the last question.

Problem 3. What conditions imposed on a preference system guarantee that, despite the existence of forbidden pairs of agents, a stable matching exists?

Another interesting open problem is how likely it is to get a preference system with a stable matching.

Problem 4. Given the integers $s > 1$ and $n > 1$, what is the share of the $(s+1)$-Sided Stable Matching Problems with n agents in each set, for which a stable matching exists?

Acknowledgements

I am very thankful to the two anonymous Reviewers for their comments, which helped me improve the initial version of the manuscript. The research was funded in whole or in part by the National Science Centre, Poland (grant number: 2023/51/B/HS4/00829). For the purpose of Open Access, the author has applied a CC-BY public copyright licence to any Author Accepted Manuscript (AAM) version arising from this submission.

References

[1] Abraham, D. J., Irving, R. W., and Manlove, D. F. Two algorithms for the student-project allocation problem. *Journal of Discrete Algorithms 5*, 1 (2007), 73–90.

[2] Alkan, A. Nonexistence of stable threesome matchings. *Mathematical Social Sciences 16*, 2 (1988), 207–209.

[3] Anholcer, M. Rozwiązania stabilne w trójstronnym zagadnieniu przydziału. In *Modelowanie Preferencji a Ryzyko'12* (Katowice, 2012), T. Trzaskalik, Ed., Zeszyty Naukowe Uniwersytetu Ekonomicznego w Katowicach, Studia Ekonomiczne 97, Uniwersytet Ekonomiczny w Katowicach, pp. 17–22.

[4] Anholcer, M., and Bartkowiak, M. On a many-sided matching problem with mixed preferences. *Operations Research and Decisions 34*, 3 (2024), 1–13.

[5] Arenas, J., and Torres-Martínez, J. P. Reconsidering the existence of stable solutions in three-sided matching problems with mixed preferences. *Journal of Combinatorial Optimization 45*, 2 (2023), 62.

[6] Biró, P., and McDermid, E. Three-sided stable matchings with cyclic preferences. *Algorithmica 58*, 1 (2010), 5–18.

[7] Boros, E., Gurvich, V., Jaslar, S., and Krasner, D. Stable matchings in three-sided systems with cyclic preferences. *Discrete Mathematics 289*, 1 (2004), 1–10.

[8] Cseh, Á., Escamocher, G., Genç, B., and Quesada, L. A collection of constraint programming models for the three-dimensional stable matching problem with cyclic preferences. *Constraints 27*, 3 (Jul 2022), 249–283.

[9] Cui, L., and Jia, W. Cyclic stable matching for three-sided networking services. *Computer Networks 57*, 1 (2013), 351–363.

[10] Danilov, V. Existence of stable matchings in some three-sided systems. *Mathematical Social Sciences 46*, 2 (2003), 145–148. Mathematical Theories of Allocation of Discrete Resources:Equilibria,Matchings,Mechanisms.

[11] Eriksson, K., Sjöstrand, J., and Strimling, P. Three-dimensional stable matching with cyclic preferences. *Mathematical Social Sciences 52*, 1 (2006), 77–87.

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

[13] Hofbauer, J. d-dimensional stable matching with cyclic preferences. *Mathematical Social Sciences 82* (2016), 72–76.

[14] Huang, C.-C. Circular stable matching and 3-way kidney transplant. *Algorithmica 58*, 1 (2010), 137–150.

[15] Kamiyama, N. A new approach to the pareto stable matching problem. *Mathematics of Operations Research 39*, 3 (2014), 851–862.

[16] Kato, A. Complexity of the sex-equal stable marriage problem. *Japan Journal of Industrial and Applied Mathematics 10*, 1 (1993), 1–19.

[17] Knuth, D. *Mariages stables et leurs relations avec d'autres problèmes combinatoires: introduction à l'analyse mathématique des algorithmes*. Collection de la Chaire Aisenstadt. Presses de l'Université de Montréal, 1976.

[18] Lahiri, S. Three-sided matchings and separable preferences. *Contributions to Game Theory and Management 2* (2009), 251–259.

[19] Lam, C.-K., and Plaxton, C. G. On the existence of three-dimensional stable matchings with cyclic preferences. *Theory of Computing Systems 66*, 3 (June 2022), 679–695. TY - JOUR.

[20] Lerner, E. A counterexample of size 20 for the problem of finding a 3-dimensional stable matching with cyclic preferences. *Discrete Applied Mathematics 333* (2023), 1–12.

[21] Manlove, D. F. The structure of stable marriage with indifference. *Discrete Applied Mathematics 122*, 1 (2002), 167–181.

[22] Pashkovich, K., and Poirrier, L. Three-dimensional stable matching with cyclic preferences. *CoRR abs/1807.05638* (2018).

[23] Ren, M., Yang, L., Jiang, H., Chen, J., and Zhou, Y. *Energy-Delay Tradeoff in Helper-Assisted NOMA-MEC Systems: A Four-Sided Matching Algorithm*. arXiv, 2023. arXiv:2301.10624.

[24] Shapley, L., and Scarf, H. On cores and indivisibility. *Journal of Mathematical Economics 1*, 1 (1974), 23–37.

[25] Zhang, F., and Zhong, L. Three-sided matching problem with mixed preferences. *Journal of Combinatorial Optimization 42* (2021), 928–936.