Resource Allocation under the Latin Square Constraint

Extended Abstract

Yasushi Kawase
The University of Tokyo
Tokyo, Japan
kawase@mist.i.u-tokyo.ac.jp

Bodhayan Roy
Indian institute of Technology
Kharagpur, India
bodhayan.roy@gmail.com

Mohammad Azharuddin Sanpui
Indian institute of Technology
Kharagpur, India
azharuddinsanpui123@kgpian.iitkgp.ac.in

ABSTRACT

A Latin square is an $n \times n$ matrix filled with n distinct symbols, each appearing exactly once in each row and column. We introduce a problem of allocating n indivisible items among n agents over n rounds while satisfying the Latin square constraint. This constraint ensures that each agent receives no more than one item per round and receives each item at most once. Each agent has an additive valuation on the item–round pairs. Real-world applications like scheduling, resource management, and experimental design require the Latin square constraint to satisfy fairness or balancedness in allocation. Our goal is to find a partial or complete allocation that maximizes the sum of the agents' valuations (utilitarian social welfare) or the minimum of the agents' valuations (egalitarian social welfare). For maximizing utilitarian social welfare, we prove NP-hardness even when the valuations are binary additive. We then provide $(1 - 1/e)$ and $(1 - 1/e)/4$-approximation algorithms for partial and complete settings, respectively. Additionally, we present fixed-parameter tractable (FPT) algorithms with respect to the order of Latin square and the optimum value for both partial and complete settings. For maximizing egalitarian social welfare, we establish that deciding whether the optimum value is at most 1 or at least 2 is NP-hard for both the partial and complete settings, even when the valuations are binary. Furthermore, we prove that checking the existence of a complete allocation satisfying each of envy-free, proportional, equitable, envy-free up to any good, proportional up to any good, or equitable up to any good is NP-hard, even when the valuations are identical.

CCS CONCEPTS

• **Theory of computation** → *Approximation algorithms analysis*; *Algorithmic game theory*; *Solution concepts in game theory*.

KEYWORDS

Latin square, Utilitarian social welfare, Egalitarian social welfare, Approximation algorithm, Parameterized algorithm, NP-hardness

ACM Reference Format:
Yasushi Kawase, Bodhayan Roy, and Mohammad Azharuddin Sanpui. 2025. Resource Allocation under the Latin Square Constraint: Extended Abstract. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 2 pages.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

A	B	C	D
B	C	D	A
C	D	A	B
D	A	B	C

A	B		
B	A		
		D	C
		C	D

A	B	C	
			D

Figure 1: Examples of complete and partial Latin squares

1 INTRODUCTION

Allocating indivisible resources fairly and efficiently is crucial in many practical scenarios, such as scheduling sightseeing for multiple groups visiting various locations or assigning shifts to medical professionals.

In this paper, we introduce the problem of allocating n indivisible items among n agents over n rounds, ensuring that each agent receives each item once. This problem can be regarded as an allocation problem under a *Latin square* constraint.

The notion of indivisible item allocation constrained by a Latin square provides a systematic and effective method for distributing tasks or resources in many real-world contexts. A Latin square is a mathematical configuration in which each element occurs exactly once in each row and each column of a grid, thus preventing any repetition within the same context (see Figure 1). Various domains, such as job scheduling, school timetabling, resource allocation optimization in computing systems, and event seating arrangement structuring, utilize this constraint. See the full version of this paper [4] for more details on examples of applications.

A *complete Latin square* of order n is an $n \times n$ array filled with n different symbols, each occurring exactly once in each row and exactly once in each column. A *partial Latin square* is the case where some cells may be empty. Figure 1 illustrates one example of complete Latin squares and two examples of partial Latin squares.

In our setting, rows correspond to items, columns correspond to rounds, and symbols correspond to agents. The structure of the Latin square ensures that no item is allocated to multiple agents in each round, each agent receives at most one item per round, and no agent receives the same item more than once.

Suppose that each agent i has a valuation v_{ijk} for each pair of item j and round k. The utility of agent i is defined as the sum of the valuations that they receive. We investigate the computational complexities of finding a partial or complete allocation under the Latin square constraint that maximizes social welfare. We call this problem the *Latin square allocation (LSA)* problem. As the measure of social welfare, we employ two settings: utilitarian social welfare and egalitarian social welfare. Utilitarian social welfare is defined as the sum of the utilities of the agents, while egalitarian social welfare is defined as the minimum of the utilities of the agents.

2 MODEL

For a positive integer n, we denote the set $\{1, 2, \ldots, n\}$ by $[n]$. In this paper, we address the problem of assigning n items to n agents over n rounds, which we refer to LSA problem. Let $N = [n]$ be the set of agents, $M = [n]$ be the set of items, and $R = [n]$ be the set of rounds. An allocation A is a subset of triplets $N \times M \times R$ where $(i, j, k) \in A$ means that agent $i \in N$ receives item $j \in M$ in round $k \in R$. An allocation A is *feasible* if

(i) each agent receives at most one item per round,
(ii) no item is allocated to multiple agents in each round, and
(iii) no agent receives the same item more than once.

For a feasible allocation A, we write $A(j, k)$ to denote the agent who receives item $j \in M$ in round $k \in R$ if such an agent exists and $A(j, k) = \perp$ if no such agent exists. We will call a pair of an item and a round a cell. Additionally, we will denote by A_i the set $\{(j, k) \in M \times R : A(j, k) = i\}$. A feasible allocation where each agent receives each item exactly once (i.e., $|A| = n^2$) is called a *complete* allocation. Note that a (complete) allocation corresponds to a (complete) Latin square. We will refer to a feasible allocation that is not necessarily complete as *partial*.

Each agent $i \in N$ gets a non-negative integer value of $v_{ijk} \in \mathbb{Z}_+$ when receiving item $j \in M$ in round $k \in R$. We say that the valuations are *binary* if $v_{ijk} \in \{0, 1\}$ for all $i \in N$, $j \in M$, and $k \in R$. Additionally, we say that the valuations are *identical* if $v_{ijk} = v_{i'jk}$ for all $i, i' \in N$, $j \in M$, and $k \in R$. For $S \subseteq M \times R$, let $v_i(S)$ denote the value $\sum_{(j,k)\in S} v_{ijk}$. The *utilitarian social welfare* and the *egalitarian social welfare* of a feasible allocation A is defined as $\sum_{i\in N} v_i(A_i)$ and $\min_{i\in N} v_i(A_i)$, respectively. We call an allocation A *Umax* and *Emax* if it maximizes utilitarian social welfare and egalitarian social welfare, respectively.

The complete Umax LSA problem is a problem of finding a complete allocation that maximizes utilitarian social welfare. This problem can be represented as the following integer linear programming (ILP):

$$\begin{array}{lll} \max & \sum_{i\in N}\sum_{j\in M}\sum_{k\in R} v_{ijk}x_{ijk} & \\ \text{s.t.} & \sum_{i\in N} x_{ijk} = 1 & (\forall j \in M,\ k \in R), \\ & \sum_{j\in M} x_{ijk} = 1 & (\forall k \in R,\ i \in N), \\ & \sum_{k\in R} x_{ijk} = 1 & (\forall i \in N,\ j \in M), \\ & x_{ijk} \in \{0, 1\} & (\forall i \in N,\ j \in M,\ k \in R), \end{array} \tag{1}$$

where $A = \{(i, j, k) \in N \times M \times R : x_{ijk} = 1\}$ corresponds to a complete allocation. The partial Umax LSA problem involves finding a partial allocation that maximizes utilitarian social welfare. This can be formulated as an ILP similar to (1), except that the equalities in the constraints are replaced with inequalities (less than or equal to). We define the partial and complete Emax LSA problems in a similar manner.

Example 2.1. Suppose that $n = 2$, $v_{1,1,1} = v_{2,2,2} = 1$, and $v_{1,1,2} = v_{1,2,1} = v_{1,2,2} = v_{2,1,1} = v_{2,1,2} = v_{2,2,1} = 0$. Then, the Umax value is 2, and the Emax value is 1 for the partial LSA problem, achieved by the partial allocation $\{(1, 1, 1), (2, 2, 2)\}$. On the other hand, there are only two complete allocations: $\{(1, 1, 1), (1, 2, 2), (2, 1, 2), (2, 2, 1)\}$ and $\{(1, 1, 2), (1, 2, 1), (2, 1, 1), (2, 2, 2)\}$. Both of these complete allocations have a utilitarian social welfare of 1 and an egalitarian social welfare of 0.

3 RESULTS

We provide the following algorithmic results.

THEOREM 3.1. *There exists a $(1 - 1/e)$-approximation algorithm for the partial Umax LSA problem, and a $(1 - 1/e)/4$-approximation algorithm for the complete Umax LSA.*

This result is based on techniques developed for the Latin square extension problem [2] and the separable assignment problem [1]. The algorithm constructs a configuration LP and then solves it using the *ellipsoid method* [3]. Finally, it rounds the solution by applying a contention resolution scheme.

THEOREM 3.2. *There exist FPT algorithms with respect to n whose computational complexity is $e^{O(n^2 \log n)}$ for the LSA problems of partial/complete Umax/Emax.*

THEOREM 3.3. *There exist FPT algorithms with respect to the Umax value for both the partial and complete LSA problems.*

Moreover, we present various results on the NP-hardness of finding desirable allocations.

THEOREM 3.4. *When the valuations are binary, deciding whether there exists a complete allocation A such that $v_{ijk} = 1$ for all $(i, j, k) \in A$ is strongly NP-complete. Moreover, when the valuations are binary, computing a partial or complete allocation that satisfies each of Umax, Emax, non-wastefulness, or Pareto optimality is strongly NP-hard.*

THEOREM 3.5. *Even when the valuations are binary, deciding whether the Emax value is 1 or 2 is strongly NP-hard for both the partial and complete LSA problems. Furthermore, deciding whether the Umax value is at least 2n or less is strongly NP-hard.*

THEOREM 3.6. *There exists an approximation-preserving reduction from the max-min fair allocation problem to the Emax LSA problem.*

THEOREM 3.7. *Even when the valuations are identical, checking the existence of a complete allocation in an LSA problem that satisfies each of EF, PROP, EQ, EFX, EQX, and PROPX is strongly NP-complete. Moreover, even when the valuations are identical, checking whether the Emax value of a complete LSA problem is at least a certain value is also strongly NP-complete.*

ACKNOWLEDGMENTS

This work was partially supported by JST ERATO Grant Number JPMJER2301, JST PRESTO Grant Number JPMJPR2122, JSPS KAKENHI Grant Number JP20K19739, Value Exchange Engineering, a joint research project between Mercari, Inc. and the RIISE, Sakura Science Exchange Program, and SERB MATRICS Grant Number MTR/2021/000474.

REFERENCES

[1] Lisa Fleischer, Michel X. Goemans, Vahab S. Mirrokni, and Maxim Sviridenko. 2006. Tight approximation algorithms for maximum general assignment problems. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm.* 611–620.

[2] Carla P. Gomes, Rommel G. Regis, and David B. Shmoys. 2004. An improved approximation algorithm for the partial Latin square extension problem. *Operations Research Letters* 32, 5 (2004), 479–484.

[3] Martin Grötschel, László Lovász, and Alexander Schrijver. 2012. *Geometric algorithms and combinatorial optimization.* Vol. 2. Springer Science & Business Media.

[4] Yasushi Kawase, Bodhayan Roy, and Mohammad Azharuddin Sanpui. 2025. Resource Allocation under the Latin Square Constraint. *arXiv preprint arXiv:2501.06506* (2025).