Approximability of Latin Square Completion-Type Puzzles

Haraguchi, Kazuya
Faculty of Commerce, Otaru University of Commerce : Associate Professor

Ono, Hirotaka
Faculty of Economics, Kyushu University : Associate Professor

Approximability of Latin Square Completion-Type Puzzles[⋆]

Kazuya Haraguchi[1] and Hirotaka Ono[2]

[1] Faculty of Commerce, Otaru University of Commerce, Japan.
haraguchi@res.otaru-uc.ac.jp
[2] Faculty of Economics, Kyushu University, Japan.
hirotaka@econ.kyushu-u.ac.jp

Abstract. Among many variations of pencil puzzles, Latin square Completion-Type puzzles (LSCP), such as Sudoku, Futoshiki and BlockSum, are quite popular for puzzle fans. Concerning these puzzles, the *solvability* has been investigated from the viewpoint of time complexity in the last decade; it has been shown that, in most of these puzzles, it is NP-complete to determine whether a given puzzle instance has a proper solution. In this paper, we investigate the *approximability* of LSCP. We formulate LSCP as the maximization problem that asks to fill as many cells as possible, under the Latin square condition and the inherent condition. We then propose simple generic approximation algorithms for LSCP and analyze their approximation ratios.

Keywords: Latin square Completion-Type puzzles, approximation algorithms, Sudoku, Futoshiki, BlockSum

1 Introduction

Pencil puzzles are now very popular all over the world, and even specialized magazines are published.[3] Among many variations of pencil puzzles, *Latin square Completion-Type puzzle* (*LSCP*), such as Sudoku, is quite popular for puzzle fans. In a typical LSCP, we are given an $n \times n$ *partial Latin square*. An $n \times n$ partial Latin square is an assignment of n integers (i.e., $1, 2, \dots, n$) to n^2 cells on the $n \times n$ grid such that the *Latin square condition* is satisfied. The Latin square condition requires that, in each row and in each column, every integer in $\{1, 2, \dots, n\}$ should appear at most once. Then we are asked to fill all the empty cells with n integers so that the Latin square condition and the constraints peculiar to the puzzle are satisfied.

In this paper, we investigate the *approximability* of LSCP. We formulate LSCP as the maximization problem that asks to fill as many empty cells as possible, under the Latin square condition and the inherent condition. Picking

⋆ This work is partially supported by JSPS KAKENHI Grant Number 24106004, 25104521 and 25870661.

[3] http://www.nikoli.co.jp/en/

up Sudoku, Futoshiki and BlockSum, we present three generic algorithms for approximately solving these puzzles. The generic approximation algorithms are standard ones: a greedy approach, a matching-based approach and a local search approach. We then analyze their approximation ratios.

Let us describe the background of the research. Concerning the pencil puzzles, the main attention in the last decade is *solvability* from the viewpoint of time complexity. It has been shown that, in most of the pencil puzzles, it is NP-complete to determine whether a given puzzle instance has a proper solution; e.g., Hashiwokakero [1], Kurodoko [13], Shakashaka [5]. For LSCP, BlockSum [9] and Sudoku [19] are NP-complete. Hearn and Demaine [10] investigated computational complexity of not only pencil puzzles but also other types of puzzles.

Unlike these previous studies, we are interested in the approximability of LSCP rather than solvability because it might be more useful information for *puzzle solvers*. From the viewpoint of puzzle solvers, the NP-completeness of solvability is not necessarily useful information because the puzzle solvers are usually given solvable puzzle instances. Alternatively, a useful theoretical result for puzzle solvers could be approximability. It might be more fun to know that a certain strategy (or algorithm) always fills 50% of the empty cells, or that it is NP-hard to fill 99% of the empty cells. The complexity of solvability could be meaningful rather for *puzzle creators*. They should create solvable puzzle instances, often those having unique solutions. The intractability might imply that the task is difficult even if they can use computers.

The paper is organized as follows. We prepare terminologies and formulate the three LSCP as maximization problems in Sect. 2. In Sect. 3, we review the previous results on computational hardness of the LSCP and present our new results on Futoshiki. Then in Sect. 4, we present generic approximation algorithms for the LSCP, along with their approximation ratios for the respective puzzles. Finally we give concluding remarks in Sect. 5.

2 Preliminaries

2.1 Latin Square

Let $n \geq 2$ be a natural number. First we introduce notations on the $n \times n$ grid of cells. Let us denote $[n] = \{1, 2, \ldots, n\}$. For any $i, j \in [n]$, we denote the cell in the row i and in the column j by (i, j). We say that two cells (i, j) and (i', j') are *adjacent* if $|i - i'| + |j - j'| = 1$. The adjacency defines the connectivity of cells. A *block* is a set of connected cells. We denote a block by $B \subseteq [n]^2$. We call B a τ*-block* if it consists of τ cells. In particular, we call 1-block a *unit block*. When the cells in the block form a $p \times q$ rectangle, we call it a $(p \times q)$*-block*.

Next we introduce notations on assignment of values to the grid. The values to be assigned are the n integers $1, 2, \ldots, n$. We represent a partial assignment of values by an $n \times n$ array, say A. For each cell (i, j), we denote the assigned value by $A_{ij} \in [n] \cup \{0\}$, where $A_{ij} = 0$ indicates that (i, j) is *empty*. When all the cells are empty, we call A *empty*. We define the *size of* A as the number of non-empty cells of A. We denote the size of A by $|A|$, that is, $|A| = |\{(i,j) \in [n]^2 \mid A_{ij} \neq 0\}|$. We call A a *partial Latin square* (*PLS*) if it satisfies the Latin square condition that we introduced in Sect. 1. In particular, if all the cells are assigned values, then we simply call A a *Latin square* (*LS*). Two PLSs A and L are *compatible* if the following two conditions hold:

(i) For every cell $(i,j) \in [n]^2$, at least one of $A_{ij} = 0$ and $L_{ij} = 0$ holds.
(ii) The assignment $A \oplus L$ defined as follows is a PLS:

$$(A \oplus L)_{ij} = \begin{cases} A_{ij} & \text{if } A_{ij} \neq 0 \text{ and } L_{ij} = 0, \\ L_{ij} & \text{if } A_{ij} = 0 \text{ and } L_{ij} \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

A PLS L' is an *extension of a PLS L* (or equivalently, L is a *restriction of L'*) if $L'_{ij} = L_{ij}$ whenever $L_{ij} \neq 0$. When L' is an extension of L, we write $L' \geqslant L$. One readily sees that $L' \geqslant L$ holds iff there is a PLS A such that A and L are compatible and $L' = A \oplus L$. Given a PLS L, the *partial Latin square extension* (*PLSE*) problem asks to construct a PLS A of the maximum size such that A and L are compatible.

2.2 Sudoku, Futoshiki and BlockSum as Maximization Problems

We formulate the three puzzles as maximization problems. We illustrate instances and solutions of these puzzles in Fig. 1. The rules of the respective puzzles are described as follows: Sudoku asks to complete the Latin square so that, in each block indicated by bold lines, every integer appears exactly once. Futoshiki asks to complete the Latin square so that, when there is an inequality sign between two adjacent cells, two integers assigned to them should satisfy the inequality. BlockSum asks to complete the Latin square so that, in each block indicated by bold lines, the sum of the assigned integers over the block should be equal to the small value that is depicted in the block.

The puzzle maximization problems ask not to complete the Latin square but to fill as many cells as possible. An optimal solution is not necessarily an LS, whereas puzzle instances that are given to human solvers usually have unique LS solutions. Each problem is a special type of the PLSE problem in the sense that, given a PLS L and possibly additional parameters, we are asked to construct a PLS A of the maximum size so that A and L are compatible, and at the same time, $A \oplus L$ satisfies the condition $\mathcal{C}$ peculiar to the puzzle. The extra condition $\mathcal{C}$ is peculiar to each puzzle, coming from the rule of the puzzle.

To deal with the ordinary PLSE and the three maximization problems in a unified way, we denote the PLSE with extra condition $\mathcal{C}$ by *$\mathcal{C}$-PLSE*. When we write $\mathcal{C}$-PLSE, $\mathcal{C}$ can be any of $\mathcal{C}_{\text{SUD}}$, $\mathcal{C}_{\text{FUT}}$, $\mathcal{C}_{\text{BS}}$ and $\mathcal{C}_{\text{NULL}}$, where $\mathcal{C}_{\text{SUD}}$ (resp., $\mathcal{C}_{\text{FUT}}$ and $\mathcal{C}_{\text{BS}}$) denotes the constraint peculiar to Sudoku (resp., Futoshiki and BlockSum) and $\mathcal{C}_{\text{NULL}}$ denotes the null condition; $\mathcal{C}_{\text{NULL}}$-PLSE represents the ordinary PLSE problem. For a $\mathcal{C}$-PLSE instance, we call a PLS A a *$\mathcal{C}$-solution* if A and L are compatible and $A \oplus L$ satisfies $\mathcal{C}$ with respect to the given instance. We may abbreviate it into simply a *solution* when $\mathcal{C}$ is clear from the context.

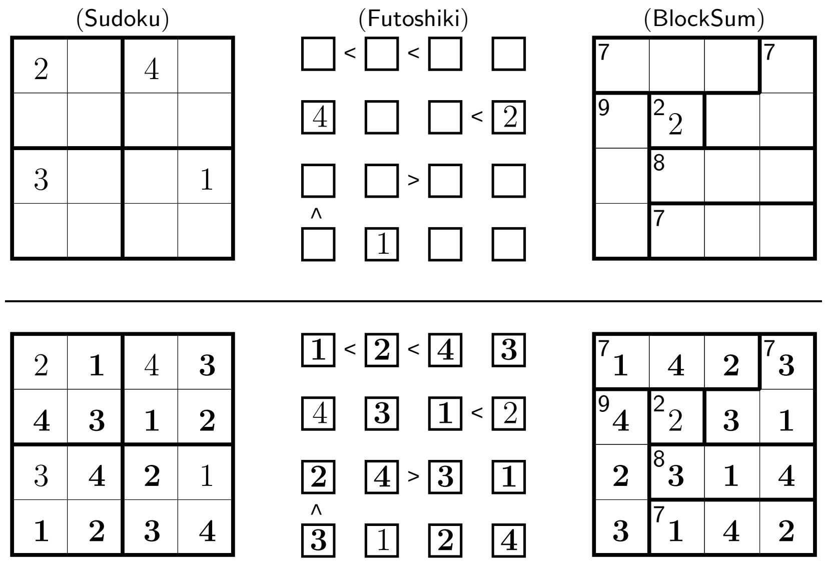

Fig. 1. Instances (upper) and solutions (lower) of Sudoku, Futoshiki and BlockSum ($n = 4$; $n_0 = n_1 = 2$ for Sudoku)

Below we explain the condition $\mathcal{C}$ and what is given as an instance besides a PLS L in the respective puzzles.

In Sudoku, the grid length n is assumed to be a composite number. We are given two positive integers n_0 and n_1 such that $n = n_0 n_1$. Note that the $n \times n$ grid is partitioned into n $(n_0 \times n_1)$-blocks.

Condition $\mathcal{C}_{\mathrm{SUD}}$: In every $(n_0 \times n_1)$-block, each integer in $[n]$ appears at most once.

We call a PLS a *Sudoku PLS* if it satisfies $\mathcal{C}_{\mathrm{SUD}}$. Given a Sudoku PLS L, the $\mathcal{C}_{\mathrm{SUD}}$-PLSE problem asks to construct a Sudoku PLS A of the maximum size such that A and L are compatible, and that $A \oplus L$ is a Sudoku PLS as well.

Problem $\mathcal{C}_{\mathrm{SUD}}$-PLSE (Sudoku)
Input: Two positive integers n_0 and n_1 such that $n = n_0 n_1$ and an $n \times n$ Sudoku PLS L.
Output: An $n \times n$ Sudoku PLS A of the maximum size such that A and L are compatible, and at the same time, $A \oplus L$ is a Sudoku PLS.

In Futoshiki, we are given a set of inequality signs such that each inequality sign is located between two adjacent cells. Let Q_L be the set of all the ordered pairs of two adjacent cells such that at least one of them is empty in L, that is,

$$Q_L = \big\{((i,j),(i',j')) \in [n]^2 \times [n]^2 \mid (i,j) \text{ and } (i',j') \text{ are adjacent, and at least one of } (i,j) \text{ and } (i',j') \text{ is empty in } L\big\}.$$

We call a subset Q of Q_L a *sign set* (*with respect to* L) when $((i,j),(i',j')) \in Q$ implies $((i',j'),(i,j)) \notin Q$. Each $((i,j),(i',j')) \in Q$ represents a constraint such that (i,j) should be assigned a smaller integer than (i',j'). Note that Q contains at most one inequality sign between any two adjacent cells, and in particular, it contains no inequality sign between two adjacent cells such that both cells are non-empty; such an inequality sign would be redundant in the puzzle. The $\mathcal{C}_{\mathrm{FUT}}$-PLSE problem asks to construct a PLS A of the maximum size such that A and L are compatible and $A \oplus L$ satisfies the following condition.

Condition $\mathcal{C}_{\mathrm{FUT}}$: For every pair $((i,j),(i',j'))$ of adjacent cells in Q, either (i) or (ii) holds: **(i)** $(A \oplus L)_{ij} = 0$ or $(A \oplus L)_{i'j'} = 0$, or **(ii)** $(A \oplus L)_{ij} < (A \oplus L)_{i'j'}$.

Problem $\mathcal{C}_{\mathrm{FUT}}$-PLSE (Futoshiki)
Input: An $n \times n$ PLS L and a sign set $Q \subseteq Q_L$.
Output: An $n \times n$ PLS A of the maximum size such that A and L are compatible, and at the same time, that $A \oplus L$ satisfies $\mathcal{C}_{\mathrm{FUT}}$.

In BlockSum, we are given a partition $\mathcal{B}$ of n^2 cells into blocks, and a function $\sigma : \mathcal{B} \to [n^2(n+1)/2]$. The $\mathcal{B}$ is a partition such that every non-empty cell (i,j) in L constitutes a unit block, i.e., $\{(i,j)\} \in \mathcal{B}$, and that every empty cell is contained in a non-unit block. The function σ is called a *capacity function*. The integer $\sigma(B)$ assigned to each block $B \in \mathcal{B}$ is the *capacity of* B. For any unit block $\{(i,j)\}$, its capacity $\sigma(\{i,j\})$ is set to L_{ij}. Also σ satisfies $\sum_{B \in \mathcal{B}} \sigma(B) = n^2(n+1)/2$, where the right hand side is the sum of n^2 integers in any $n \times n$ LS. The $\mathcal{C}_{\mathrm{BS}}$-PLSE problem asks to construct an $n \times n$ PLS A of the maximum size such that A and L are compatible and that and $A \oplus L$ satisfies the following condition.

Condition $\mathcal{C}_{\mathrm{BS}}$: For every block B in the partition $\mathcal{B}$,

$$\sum_{(i,j) \in B} (A \oplus L)_{ij} \leq \sigma(B). \tag{1}$$

In (1), we relax the condition of the orignal BlockSum by replacing the equality with the inequality in order to treat the puzzle as the maximization problem.

Problem $\mathcal{C}_{\mathrm{BS}}$-PLSE (BlockSum)
Input: An $n \times n$ PLS L, a partition $\mathcal{B}$ of n^2 cells into blocks, and a capacity function $\sigma : \mathcal{B} \to [n^2(n+1)/2]$.
Output: An $n \times n$ PLS A of the maximum size such that A and L are compatible, and at the same time, that $A \oplus L$ satisfies $\mathcal{C}_{\mathrm{BS}}$.

Note that we have only to consider how we assign integers to the empty cells, all of which are contained in non-unit blocks; in any unit block $\{(i,j)\} \in \mathcal{B}$, the cell is already assigned the integer L_{ij}. Since it is equal to the capacity $\sigma(\{i,j\})$ of the block, (1) is automatically satisfied.

We have finished explaining the three maximization problems. In each problem, one can easily confirm the solution monotonicity such that, when A is a solution, any restriction $A' \leqslant A$ is a solution as well. A solution A is *blocked* if any extension A' of A ($A' \neq A$) is not a solution.

Let us denote a maximization problem instance by I and its global optimal solution by $A^*(I)$. For a real number $\rho \in [0,1]$, a solution A to the instance I is a *ρ-approximate solution* if $|A|/|A^*(I)| \geq \rho$ holds. A polynomial time algorithm is called a *ρ-approximation algorithm* if it delivers a ρ-approximate solution for any instance. The bound ρ is called the *approximation ratio* of the algorithm.

3 Hardness

We review previous studies on computational complexity of $\mathcal{C}$-PLSE and present our new results on $\mathcal{C}_{\mathrm{FUT}}$-PLSE. First we mention that $\mathcal{C}_{\mathrm{NULL}}$-PLSE (i.e., the ordinary PLSE) is computationally expensive.

Theorem 1 (Colbourn [3]). *$\mathcal{C}_{\mathrm{NULL}}$-PLSE is NP-hard.*

Theorem 2 (Easton and Parker [6]). *$\mathcal{C}_{\mathrm{NULL}}$-PLSE is NP-hard even if at most three empty cells exist in any row or in any column, and only three values are available.*

Theorem 3 (Hajirasouliha et al. [7]). *$\mathcal{C}_{\mathrm{NULL}}$-PLSE is APX-hard.*

$\mathcal{C}_{\mathrm{SUD}}$-PLSE is NP-hard in general [19]. Interestingly, it is still NP-hard even if each row (or column) is either empty or full, whereas $\mathcal{C}_{\mathrm{NULL}}$-PLSE in this case can be solved in polynomial time [2]. $\mathcal{C}_{\mathrm{BS}}$-PLSE is NP-hard even if every block consists of at most two cells [9].

$\mathcal{C}_{\mathrm{FUT}}$-PLSE has been hardly studied in the literature except [8], which discusses how many inequality signs should be given in automatic instance generation. We summarize the computational hardness of $\mathcal{C}_{\mathrm{FUT}}$-PLSE in Table 1. A $\mathcal{C}_{\mathrm{FUT}}$-PLSE instance is given in terms of (L, Q) such that L is a PLS and $Q \subseteq Q_L$ is a sign set. When L is empty, we know nothing about the hardness except the trivial case of $Q = \emptyset$, where any LS is as an optimal solution. We leave the case of empty L open. Let us turn our attention to the case of non-empty L. When $Q = \emptyset$, the problem is equivalent to $\mathcal{C}_{\mathrm{NULL}}$-PLSE, and thus is NP-hard by Theorem 1. When Q is a non-empty subset of Q_L, it is NP-hard by the following Theorem 4.

Theorem 4. *$\mathcal{C}_{\mathrm{FUT}}$-PLSE is NP-hard if L is a non-empty PLS and Q is a non-empty subset of Q_L.*

Proof. We prove the theorem by reduction from the special case of $\mathcal{C}_{\mathrm{NULL}}$-PLSE in Theorem 2; at most three cells are empty in each row and in each column, and only three values are available. Permuting the n values appropriately, we can set the three available values to 1, 2 and 3. Let L be the PLS that is given in this way. We transform L into a $\mathcal{C}_{\mathrm{FUT}}$-PLSE instance on the $2n \times 2n$ grid.

Table 1. Computational hardness of $\mathcal{C}_{\text{FUT}}$-PLSE

		Sign set $Q \subseteq Q_L$		
		empty	**non-empty**	
			(Q is any subset of Q_L)	($Q = Q_L$)
PLS L	**empty**	trivial	?	?
	non-empty	NP-hard (Theorem 1)	NP-hard (Theorem 4)	NP-hard (Corollary 1)

Let L' be an arbitrary $n \times n$ LS. We define a $2n \times 2n$ PLS L'' as follows; for $k, \ell = 1, 2, \ldots, n$,

$$L''_{(2k-1)(2\ell-1)} = L_{k\ell}, \quad L''_{(2k-1)(2\ell)} = L''_{(2k)(2\ell-1)} = L'_{k\ell} + n, \quad L''_{(2k)(2\ell)} = L'_{k\ell}.$$

Let us emphasize that the PLS L should be copied to the n^2 $(2k-1, 2\ell-1)$'-s, i.e., the cells such that both row order and column order are odd. All the remaining cells are assigned values in $[2n]$ so that, in each row and column, any value in $[2n]$ appears at most once; they play the role of garbage collection. The empty cells appear in only $(2k-1, 2\ell-1)$'-s and any two of them are not adjacent. Then for any empty cell $(2k-1, 2\ell-1)$ and any non-empty cell (i, j) adjacent to it, we let $((2k-1, 2\ell-1), (i, j)) \in Q$, i.e., $(2k-1, 2\ell-1)$ should be assigned a smaller value than (i, j), where (i, j) is already assigned an integer larger than n in L''. We have finished constructing the $\mathcal{C}_{\text{FUT}}$-PLSE instance. The construction time is obviously polynomial. In the decision problem versions, the answers to $\mathcal{C}_{\text{NULL}}$-PLSE instance and the constructed $\mathcal{C}_{\text{FUT}}$-PLSE instance agree. $\square$

In the above proof, the sign set Q is set to the full sign set $Q_{L''}$.

Corollary 1 *When L is non-empty, $\mathcal{C}_{\text{FUT}}$-PLSE is still NP-hard even if Q is restricted to $Q = Q_L$.*

4 Approximation Algorithms

In this section, we present approximation algorithms for $\mathcal{C}$-PLSE. The algorithms generalize existing ones for $\mathcal{C}_{\text{NULL}}$-PLSE. We borrow three types of algorithms from the literature: *greedy algorithm*, *matching based approach*, and *local search*. All the algorithms introduced below run in polynomial time. See the referred papers for time complexity analysis.

4.1 Greedy Algorithm

The greedy algorithm in this case refers to an algorithm as follows; starting from an empty solution, we repeat choosing an arbitrary empty cell and assigning a value in $[n]$ to the cell so that the resulting assignment remains a solution. This is repeated until the solution is blocked. For $\mathcal{C}_{\text{NULL}}$-PLSE, Kumar et al. [14] showed that it is a 1/3-approximation algorithm.

Theorem 5 (Kumar et al. [14]). *For any instance of $\mathcal{C}_{\mathrm{NULL}}$-PLSE, a blocked solution is a $1/3$-approximate solution.*

To extend this theorem, we give the detailed proof of the theorem.

Proof. Let A be a blocked solution and A^* be an optimal solution. We cannot assign the value A^*_{pq} to any cell (p, q) in A since at least one element in A "blocks" (p, q) from taking A^*_{pq}. We claim that each element A_{ij} in A should block at most three cells (p, q)'-s from taking A^*_{pq}; denoted by $S^{\mathrm{NULL}}_{ij}(A, A^*)$, the set of such blocked cells is defined as follows:

$$\begin{aligned} S^{\mathrm{NULL}}_{ij}(A, A^*) = &\{(i, j)\} \cup \{(i', j) \mid A_{i'j} = 0 \text{ and } A^*_{i'j} = A_{ij}\} \\ &\cup \{(i, j') \mid A_{ij'} = 0 \text{ and } A^*_{ij'} = A_{ij}\}, \end{aligned} \tag{2}$$

that is, (i, j) itself, the cell (i', j) in the same column with $A^*_{i'j} = A_{ij}$, and the cell (i, j') in the same row with $A^*_{ij'} = A_{ij}$. Clearly $|S^{\mathrm{NULL}}_{ij}(A, A^*)| \le 3$ holds for any (i, j).

We see that $|A^*| \le \sum_{ij} |S^{\mathrm{NULL}}_{ij}(A, A^*)|$ holds; if not so, there exists $(p, q) \notin \bigcup_{ij} S^{\mathrm{NULL}}_{ij}(A, A^*)$ such that (p, q) is non-empty in A^*. Then in A, (p, q) is empty and is not blocked by any A_{ij} from taking A^*_{pq}. This means that we can extend A by assigning the value A^*_{pq} to (p, q), contradicting that A is blocked.

Finally we have the inequalities $|A^*| \le \sum_{ij} |S^{\mathrm{NULL}}_{ij}(A, A^*)| \le 3|A|$, which proves that A is a $1/3$-approximate solution. $\square$

The point is the size of $S^{\mathrm{NULL}}_{ij}(A, A^*)$ in (2). Since it is at most three, any blocked solution is a $1/3$-approximate solution. Then for $\mathcal{C}$-PLSE, designing the similar set $S^{\mathcal{C}}_{ij}(A, A^*)$ appropriately, we can prove any blocked solution to be a $1/\beta^{\mathcal{C}}$-approximate solution in the analogous way, where $\beta^{\mathcal{C}}$ denotes an upper bound on the size of $S^{\mathcal{C}}_{ij}(A, A^*)$. For $\mathcal{C}_{\mathrm{SUD}}$-PLSE, we can set the upper bound to $\beta^{\mathrm{SUD}} = 4$ by taking the set $S^{\mathrm{SUD}}_{ij}(A, A^*)$ as follows:

$$\begin{aligned} S^{\mathrm{SUD}}_{ij}(A, A^*) = &S^{\mathrm{NULL}}_{ij}(A, A^*) \cup \{(p, q) \mid A_{pq} = 0,\ A^*_{pq} = A_{ij} \text{ and} \\ &(i, j) \text{ and} (p, q) \text{ belong to the same } (n_0 \times n_1)\text{-block}\}. \end{aligned}$$

Theorem 6. *For any $\mathcal{C}_{\mathrm{SUD}}$-PLSE instance, a blocked solution is a $1/4$-approximate solution.*

For $\mathcal{C}_{\mathrm{FUT}}$-PLSE, the approximation ratio depends on how many inequality signs are around a cell. Let δ denote the maximum number of inequality signs that surround an empty cell over the given instance. Clearly we have $\delta \in \{0, 1, \ldots, 4\}$. Then we can set the bound to $\beta^{\mathrm{FUT}} = 3 + \delta$ by taking the set $S^{\mathrm{FUT}}_{ij}(A, A^*)$ as follows since, in the right hand, the size of the second set is at most δ.

$$\begin{aligned} S^{\mathrm{FUT}}_{ij}(A, A^*) = &S^{\mathrm{NULL}}_{ij}(A, A^*) \cup \{(p, q) \mid A_{pq} = 0, \text{ and either} \\ &(A^*_{pq} > A_{ij} \text{ and } ((p, q), (i, j)) \in Q) \text{ or} \\ &(A^*_{pq} < A_{ij} \text{ and } ((i, j), (p, q)) \in Q)\}. \end{aligned}$$

Theorem 7. *Suppose that we are given a $\mathcal{C}_{\text{FUT}}$-PLSE instance such that the number of inequality signs surrounding a cell is at most δ. Then any blocked solution is a $1/(3+\delta)$-approximate solution.*

For $\mathcal{C}_{\text{BS}}$-PLSE, the approximation ratio depends on the maximum size of the block over the instance, which we denote by Δ. We set the bound to $\beta^{\text{BS}} = 2+\Delta$, taking the set $S_{ij}^{\text{BS}}(A, A^*)$ as follows since, in the right hand, the size of the second set is at most $\Delta - 1$.

$$S_{ij}^{\text{BS}}(A, A^*) = S_{ij}^{\text{NULL}}(A, A^*) \cup \{(p,q) \mid A_{pq} = 0,\ (i,j) \text{ and } (p,q) \text{ belong to the same block } B \in \mathcal{B},\ \text{and } A^*_{pq} + \sum_{(i',j') \in B} A_{i'j'} > \sigma(B)\}.$$

Theorem 8. *Suppose that we are given a $\mathcal{C}_{\text{BS}}$-PLSE instance such that the block size is at most Δ. Then any blocked solution is a $1/(2+\Delta)$-approximate solution.*

We observe that these approximation ratios are tight, but we omit the tight examples due to space limitation.

4.2 Matching Based Approach

Another approximation algorithm for $\mathcal{C}_{\text{NULL}}$-PLSE is based on matching. We call this algorithm MATCHING. The algorithm behaves as follows. Let I^{ijk} be a PLS such that $I^{ijk}_{pq} = k$ if $(p,q) = (i,j)$ and $I^{ijk}_{pq} = 0$ otherwise. For a given instance, it assigns the value k to empty cells in the order $k = 1, 2, \ldots, n$. Let A^{k-1} be the solution that has been constructed so far such that the values from 1 to $k-1$ are already assigned. Initially, A^0 is set to an empty solution. Which empty cells are assigned k is determined by a maximum matching in the graph $G^k = (R \cup C, E^k)$ such that $R = \{r_1, r_2, \ldots, r_n\}$ and $C = \{c_1, c_2, \ldots, c_n\}$ are the node sets that represent rows and columns of the grid respectively, and

$$E^k = \{(r_i, c_j) \in R \times C \mid A^{k-1}_{ij} = 0 \text{ and } A^{k-1} \oplus I^{ijk} \text{ is a solution}\}$$

is the edge set. Computing a maximum matching $M \subseteq E^k$, the algorithm extends A^{k-1} by assigning k to (i,j) for each edge $(r_i, c_j) \in M$, which is used as the next solution A^k. The algorithm repeats this process from $k = 1$ to n and outputs A^n.

Theorem 9 (Kumar et al. [14]). *The algorithm* MATCHING *is a 1/2-approximation algorithm for $\mathcal{C}_{\text{NULL}}$-PLSE.*

See the proof for [14]. The point is that any matching in G^k provides a set of cells to which k can be assigned simultaneously. This property holds because, in $\mathcal{C}_{\text{NULL}}$-PLSE, $A_{ij} = k$ never blocks any other cells out of row i and column j from taking k, i.e., the set $S_{ij}^{\text{NULL}}(A, A^*)$ in (2) contains no $(p,q) \in [n]^2$ such that $p \neq i$ and $q \neq j$. To $\mathcal{C}$-PLSE that has the property, we can apply the algorithm MATCHING directly so that the approximation ratio remains 1/2. Then it is applicable to $\mathcal{C}_{\text{FUT}}$-PLSE in general.

Theorem 10. *The algorithm* MATCHING *is a* $1/2$*-approximation algorithm for* $\mathcal{C}_{\mathrm{FUT}}$*-PLSE.*

On the other hand, the algorithm is not applicable to $\mathcal{C}_{\mathrm{SUD}}$-PLSE directly since the problem does not have the above property; once value k is assigned to (i, j), we cannot assign k to any other cell (p, q) in the same $(n_0 \times n_1)$-block even though (i, j) and (p, q) belong to different rows and columns, i.e., $i \neq p$ and $j \neq q$. In this case, a matching in G^k does not necessarily provide a set of empty cells that can be assigned k simultaneously. The algorithm is not applicable to $\mathcal{C}_{\mathrm{BS}}$-PLSE either, except the special case in the following theorem. The point is that each block is closed in one row or in one column.

Theorem 11. *Suppose that we are given a* $\mathcal{C}_{\mathrm{BS}}$*-PLSE instance such that each block is either a* $(1 \times \ell)$*-block or an* $(\ell \times 1)$*-block. To such an instance, the algorithm* MATCHING *delivers a* $1/2$*-approximate solution.*

4.3 Local Search

Let t denote a positive integer. We introduce the *t-set packing problem*; Let S be a finite set of elements and suppose that we are given a family $\mathcal{F} = \{F_1, \dots, F_q\}$ of q subsets of S such that each $F_i \in \mathcal{F}$ contains at most t elements. A collection $\mathcal{F}' \subseteq \mathcal{F}$ is called a *packing* if any two subsets in $\mathcal{F}'$ are disjoint. The problem asks to find a largest packing in $\mathcal{F}$, belonging to Karp's list of 21 NP-hard problems [12].

For this problem, we consider a local search algorithm that behaves as follows; given a positive integer r as a parameter, let $\mathcal{F}' \subseteq \mathcal{F}$ be an arbitrary packing. Then repeat replacing $r' \leq r$ sets in $\mathcal{F}'$ with $r' + 1$ sets in $\mathcal{F}$ such that $\mathcal{F}'$ continues to be a packing, as long as the replacement is possible. The following result is well-known.

Theorem 12 (Hurkens and Schrijver [11]). *Suppose that an instance of the* t*-set packing problem is given in terms of a family* $\mathcal{F}$ *of subsets of an element set* S*. For any parameter* $r \geq 1$*, there exists a constant* $\varepsilon > 0$ *such that the local search algorithm delivers a* $(2/t - \varepsilon)$*-approximate solution.*

Hajirasouliha et al. [7] applies the local search to $\mathcal{C}_{\mathrm{NULL}}$-PLSE by reducing it to the 3-set packing problem. Given a $\mathcal{C}_{\mathrm{NULL}}$-PLSE instance in terms of a PLS L, the packing problem instance $\mathcal{F}$ is constructed as follows. Let the element set be $S^{\mathrm{NULL}} = (R \times C) \cup (R \times [n]) \cup (C \times [n])$. Then let $\mathcal{F}$ contain a subset $\{(r_i, c_j), (r_i, k), (c_j, k)\} \subseteq S^{\mathrm{NULL}}$ iff the value k can be assigned to (i, j), i.e., L does not assign k to any cell in row i or column j. Obviously there is one-to-one, size-preserving correspondence between the solution sets of the two problem instances.

Theorem 13 (Hajirasouliha et al. [7]). *For any parameter* $r \geq 1$*, there exists a constant* $\varepsilon > 0$ *such that the local search is a* $(2/3 - \varepsilon)$*-approximation algorithm for* $\mathcal{C}_{\mathrm{NULL}}$*-PLSE.*

We can apply the local search to $\mathcal{C}_{\mathrm{SUD}}$-PLSE, regarding it as the 4-set packing problem. Suppose that a $\mathcal{C}_{\mathrm{SUD}}$-PLSE instance is given. Let $\mathcal{B} = \{B_1, \dots, B_n\}$ denote the set of $(n_0 \times n_1)$-blocks in the grid, and the element set be $S^{\mathrm{SUD}} = S^{\mathrm{NULL}} \cup (\mathcal{B} \times [n])$. We then construct the family $\mathcal{F}$ so that it contains a subset $\{(r_i, c_j), (r_i, k), (c_j, k), (B_p, k)\} \subseteq S^{\mathrm{SUD}}$ iff k can be assigned to an empty cell (i, j) that belongs to the block B_p. The solution correspondence is immediate.

Theorem 14. *For any $\varepsilon > 0$, there exists a $(1/2 - \varepsilon)$-approximation algorithm for $\mathcal{C}_{\mathrm{SUD}}$-PLSE.*

Recently, Cygan [4] improved the approximation ratio for the t-set packing problem from $2/t - \varepsilon$ to $3/(t+1) - \varepsilon$ by means of *bounded pathwidth local search*. This improves the approximation ratios for $\mathcal{C}_{\mathrm{NULL}}$-PLSE and $\mathcal{C}_{\mathrm{SUD}}$-PLSE.

Theorem 15. *For any $\varepsilon > 0$, there exists a $(3/4 - \varepsilon)$-approximation algorithm for $\mathcal{C}_{\mathrm{NULL}}$-PLSE.*

Theorem 16. *For any $\varepsilon > 0$, there exists a $(3/5 - \varepsilon)$-approximation algorithm for $\mathcal{C}_{\mathrm{SUD}}$-PLSE.*

5 Concluding Remarks

In summary, the current best approximation ratios for $\mathcal{C}$-PLSEs are as follows:

- $\mathcal{C}_{\mathrm{NULL}}$-PLSE: $3/4 - \varepsilon$ (Theorem 15).
- $\mathcal{C}_{\mathrm{SUD}}$-PLSE (Sudoku): $3/5 - \varepsilon$ (Theorem 16).
- $\mathcal{C}_{\mathrm{FUT}}$-PLSE (Futoshiki): 1/2 (Theorem 10).
- $\mathcal{C}_{\mathrm{BS}}$-PLSE (BlockSum): $1/(2 + \Delta)$ (Theorem 8); when each block is closed in one row or in one column, there is a 1/2-approximation algorithm (Theorem 11).

It is interesting future work to pursuit the limit by improving these ratios. For $\mathcal{C}_{\mathrm{NULL}}$-PLSE, since it is APX-hard (Theorem 3), there exists a constant $\rho^* \in (0, 1)$ such that no ρ^*-approximation algorithm exists unless P=NP. The above result indicates $\rho^* \geq 3/4$. For the other $\mathcal{C}$-PLSEs, whether they are APX-hard or not is open.

We described previous results on NP-hardness of PLSE, Sudoku and BlockSum and presented our results on Futoshiki in Sect. 3. Still, it is open whether the spacial case of Futoshiki such that an empty PLS is given is NP-hard (see Table 1).

An LSCP called KenKen [15–18] is a generalization of BlockSum. BlockSum deals with the summation of the assigned integers in its inherent condition $\mathcal{C}_{\mathrm{BS}}$, while subtraction, multiplication and division are also treated in KenKen. Since its special case is NP-hard, KenKen is also NP-hard. Furthermore, we can apply the greedy algorithm and the matching based algorithm to KenKen similarly to BlockSum, which achieves the same approximation ratios. We omit the details due to space limitation.

We have studied approximability and inapproximability of LSCP in general settings in the sense that we do not make any assumption on whether a puzzle instance has an LS solution or not. As pointed out in the introductory section, however, a puzzle instance given to a human solver usually has a unique solution. Hence it may be more meaningful to restrict our attention to such puzzle instances. This suggests an interesting direction of our future research.

References

1. Andersson, D.: Hashiwokakero is NP-complete. Information Processing Letters 109(19), 1145–1146 (2009)
2. Béjar, R., Fernández, C., Mateu, C., Valls, M.: The Sudoku completion problem with rectangular hole pattern is NP-complete. Discrete Mathematics 312(22), 3306–3315 (2012)
3. Colbourn, C.J.: The complexity of completing partial latin squares. Discrete Applied Mathematics 8(1), 25–30 (1984)
4. Cygan, M.: Improved approximation for 3-dimensional matching via bounded pathwidth local search. arXiv preprint arXiv:1304.1424 (2013)
5. Demaine, E.D., Okamoto, Y., Uehara, R., Uno, Y.: Computational complexity and an integer programming model of Shakashaka. In: CCCG. pp. 31–36 (2013)
6. Easton, T., Gary Parker, R.: On completing latin squares. Discrete Applied Mathematics 113(2), 167–181 (2001)
7. Hajirasouliha, I., Jowhari, H., Kumar, R., Sundaram, R.: On completing latin squares. In: STACS 2007, pp. 524–535. Springer (2007)
8. Haraguchi, K.: The number of inequality signs in the design of Futoshiki puzzle. Journal of Information Processing 21(1), 26–32 (2013)
9. Haraguchi, K., Ono, H.: Blocksum is NP-complete. IEICE Transactions on Information and Systems 96(3), 481–488 (2013)
10. Hearn, R.A., Demaine, E.D.: Games, puzzles, and computation. AK Peters, Limited (2009)
11. Hurkens, C.A.J., Schrijver, A.: On the size of systems of sets every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems. SIAM Journal on Discrete Mathematics 2(1), 68–72 (1989)
12. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of Computer Computations, pp. 85–103 (1972)
13. Kölker, J.: Kurodoko is NP-complete. Journal of Information Processing 20(3), 694–706 (2012)
14. Kumar, S.R., Russell, A., Sundaram, R.: Approximating latin square extensions. Algorithmica 24(2), 128–138 (1999)
15. Miyamoto, T.: Black Belt KenKen: 300 Puzzles. Puzzlewright (2013)
16. Miyamoto, T.: Brown Belt KenKen: 300 Puzzles. Puzzlewright (2013)
17. Miyamoto, T.: Green Belt KenKen: 300 Puzzles. Puzzlewright (2013)
18. Miyamoto, T.: White Belt KenKen: 300 Puzzles. Puzzlewright (2013)
19. Yato, T., Seta, T.: Complexity and completeness of finding another solution and its application to puzzles. IEICE transactions on fundamentals of electronics, communications and computer sciences 86(5), 1052–1060 (2003)