Journal of Information Processing Vol.23 No.3 276–283 (May 2015)

[DOI: 10.2197/ipsjjip.23.276]

Regular Paper

How Simple Algorithms Can Solve Latin Square Completion-Type Puzzles Approximately

Kazuya Haraguchi[1,a)] Hirotaka Ono[2,b)]

Received: July 31, 2014, Accepted: October 8, 2014

Abstract: Among many variations of pencil puzzles, Latin square Completion-Type puzzles (LSCPs) 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 three LSCPs: Sudoku, Futoshiki and KenKen. We formulate each LSCP as a 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 them and analyze their approximation ratios.

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

1. Introduction

Pencil puzzles are now very popular all over the world, and even specialized magazines are published (e.g., Ref. [20]). Among many variations of pencil puzzles, *Latin square Completion-Type puzzles* (*LSCPs*) are quite popular for puzzle fans. Even the most influential newspapers such as The Times and The New York Times deal with Sudoku, Futoshiki, KenKen, and their variants.

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., $1, 2, \ldots, n$) 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 $\{1, 2, \ldots, n\}$ 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 an LSCP as a maximization problem that asks to fill as many empty cells as possible, under the Latin square condition and the inherent condition. Focusing on Sudoku, Futoshiki and KenKen, 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.

Let us describe the reason why we study the approximability. First we are motivated by our observation on children's behavior in solving LSCPs. The first author once participated as an organizer in an event where ten- or eleven-year-old children are encouraged to solve many LSCPs [13], [18]. We found that some of them solve the puzzles by a greedy method, that is, focusing on an arbitrary empty cell, they fill the cell with an arbitrary integer that does not match the value in the true solution, but that does not violate the rule of the puzzle to the extent of the interim solution. On the other hand, reasonable puzzle solvers solve LSCPs by utilizing a kind of inference rule; they assign an integer k to an empty cell when they find out that k is the unique integer assignable to the cell, as a result of the inference. Then we are interested in how the greedy method or other simple strategies, which adults do not usually use, solve LSCPs approximately. This invokes our first motivation.

Concerning pencil puzzles, puzzle approximability has rarely been studied before. It is *solvability* (in terms of time complexity) that has been a key issue in the literature of theoretical computer science or discrete mathematics. It has been shown that, in most pencil puzzles, it is NP-complete to determine whether a given puzzle instance has a proper solution; e.g., Hashiwokakero [1], Kurodoko [17], Shakashaka [5]. For LSCP, Sudoku [21] is NP-complete. KenKen is also NP-complete since it includes a certain NP-complete puzzle called BlockSum as a special case [10]. Hearn and Demaine [12] investigated the computational complexity of not only pencil puzzles but also other types of puzzles.

Approximability might provide useful information for *puzzle solvers*. It might be more fun for them 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 NP-completeness of solvability is not necessarily useful information because the puzzle solvers are usually given solvable puzzle instances. 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.

[1] Faculty of Commerce, Otaru University of Commerce, Otaru, Hokkaido 047–8501, Japan.
[2] Faculty of Economics, Kyushu University, Fukuoka 812–8581, Japan.
[a)] haraguchi@res.otaru-uc.ac.jp
[b)] hirotaka@econ.kyushu-u.ac.jp

This work is partially supported by JSPS KAKENHI Grant Number 24106004, 25104521 and 25870661. The preliminary version of this paper appeared in the proceedings of FUN 2014 [11].

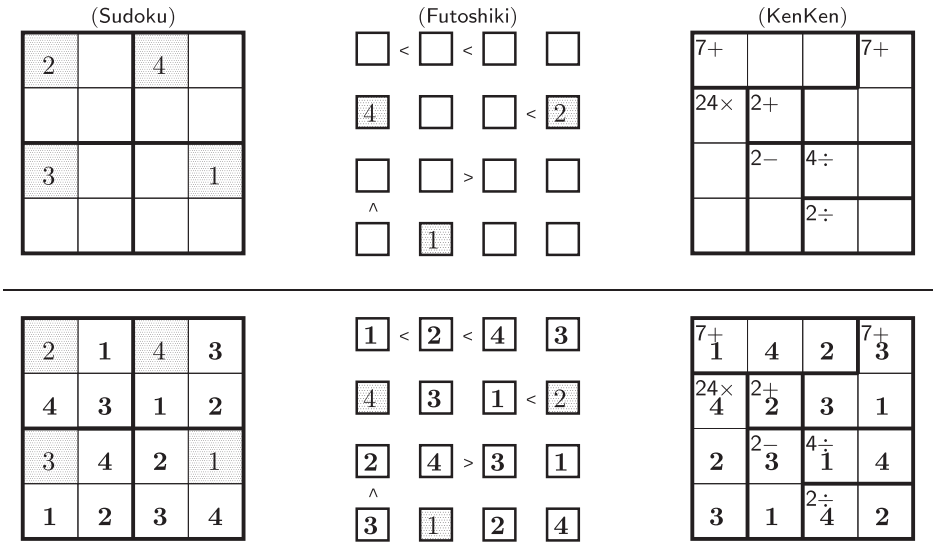

Fig. 1 Instances (upper) and solutions (lower) of Sudoku, Futoshiki and KenKen: $n = 4$, $n_0 = n_1 = 2$ for Sudoku, and shaded cells indicate integers given in the instances.

This paper is organized as follows. We prepare terminologies and formulate the three LSCPs as maximization problems in Section 2. In Section 3, we review the previous results on computational hardness of the LSCPs and present our new results on Futoshiki. Then in Section 4, we present generic approximation algorithms for the LSCPs, along with their approximation ratios for the respective puzzles. Finally we give concluding remarks in Section 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. 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 Section 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. (i.e., (i, j) is empty in at least one of A and L.)

(ii) The assignment $A \oplus L$ defined as follows is a PLS:

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

When A and L are compatible, we can construct a PLS by "merging" A and L. 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 $L' \succcurlyeq L$. One readily sees that $L' \succcurlyeq L$ 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. We analyze the approximability of the LSCPs by utilizing some previous results on the PLSE problem.

2.2 Sudoku, Futoshiki and KenKen as Maximization Problems

First we describe the rules of Sudoku, Futoshiki and KenKen. We illustrate instances and solutions of the three puzzles in **Fig. 1**. Sudoku asks to complete a given PLS so that, in each block indicated by bold lines, every integer appears exactly once. Futoshiki asks to complete a given PLS so that the assigned integers satisfy all the inequalities in the grid. KenKen asks to complete a given PLS [*1] so that, in each block indicated by bold lines, the summation (+), subtraction (−), multiplication (×) or division (÷) of the assigned integers over the block should be equal to a certain integer given to the block, which we call the *goal value* of the block. A block is also given the calculation type, where (−) and (÷) are given only to 2-blocks. The given goal value and the calculation type are depicted in each block.

We formulate the three puzzles as maximization problems. We call the problems MaxSudoku, MaxFutoshiki, and MaxKenKen respectively. The problems ask not to complete the given PLS but to fill as many empty cells with integers 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

[*1] In the KenKen instance in Fig. 1, the empty PLS is given.

C peculiar to the puzzle. The extra condition C is peculiar to each puzzle, arising from the rule of the puzzle.

Below we explain the condition C and what is given as an instance along with a PLS L in the respective problems. Given C and L, we call a PLS A a *solution* to the given instance if A and L are compatible and $A \oplus L$ satisfies C with respect to the instance.

In MaxSudoku, 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 can be partitioned into n $(n_0 \times n_1)$-blocks.

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

We call an $n \times n$ PLS a *Sudoku PLS* if it satisfies C_{SUD} with respect to given n_0 and n_1. Given n_0, n_1, and a Sudoku PLS L, the MaxSudoku 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 MaxSudoku
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 MaxFutoshiki, we are given not only a PLS L but also 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,

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

We call a subset Q of Q_L a *sign set* when $((i,j),(i',j')) \in Q$ implies $((i',j'),(i,j)) \notin Q$. Each $((i,j),(i',j')) \in Q$ represents a "smaller than" inequality sign 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 given integers by L; such an inequality sign would be redundant in the puzzle. The MaxFutoshiki 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.

Futoshiki Condition C_{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 MaxFutoshiki
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 C_{FUT}.

In MaxKenKen, we are given a PLS L, a partition $\mathcal{B}$ of n^2 cells into blocks, a function $\sigma : \mathcal{B} \to \mathbb{Z}^*$, and a function $\pi : \mathcal{B} \to \{+, -, \times, \div\}$. For any block $B \in \mathcal{B}$, the non-negative integer $\sigma(B)$ represents the goal value of B, and the sign $\pi(B)$ represents the calculation type of B. The π is a function such that $\pi(B) \in \{-, \div\}$ only when $|B| = 2$. The MaxKenKen 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.

KenKen Condition C_{KEN}: *Let us denote $X = A \oplus L$. For every block $B = \{(i_1,j_1),(i_2,j_2),\ldots,(i_p,j_p)\}$ in the partition $\mathcal{B}$, the following should hold:*

$$\begin{cases} \sum_{(i,j)\in B} X_{ij} \le \sigma(B) & \text{if } \pi(B) = +, \\ X_{i_1 j_1} = 0,\ X_{i_2 j_2} = 0, \text{ or} & \\ \max\{X_{i_1 j_1} - X_{i_2 j_2}, X_{i_2 j_2} - X_{i_1 j_1}\} \le \sigma(B) & \text{if } \pi(B) = -, \\ \prod_{(i,j)\in B:\ X_{ij}>0} X_{ij} \le \sigma(B) & \text{if } \pi(B) = \times, \\ X_{i_1 j_1} = 0,\ X_{i_2 j_2} = 0, \text{ or} & \\ \max\{\frac{X_{i_1 j_1}}{X_{i_2 j_2}}, \frac{X_{i_2 j_2}}{X_{i_1 j_1}}\} \le \sigma(B) & \text{if } \pi(B) = \div. \end{cases} \quad (1)$$

In Eq. (1), we relax the original condition of KenKen by replacing equalities with inequalities in order to admit not only an LS but also a PLS as a solution.

Problem MaxKenKen
Input: An $n \times n$ PLS L, a partition $\mathcal{B}$ of n^2 cells into blocks, a goal value function $\sigma : \mathcal{B} \to \mathbb{Z}^*$, and a calculation type function $\pi : \mathcal{B} \to \{+, -, \times, \div\}$.
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 C_{KEN}.

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. Given A, we say that an integer k is *assignable* to an empty cell (i,j) if we can extend A by assigning k to (i,j) without violating any constraints. If no integer is assignable to any empty cell, or equivalently, if no extension A' of A $(A' \neq A)$ is a solution. we say that A is *blocked*.

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 $|A|/|A^*(I)| \ge \rho$ 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 the problems described so far. We then present our new results on MaxFutoshiki.

First we mention that the ordinary PLSE problem is computationally expensive.

Theorem 1 (Colbourn [3]) *The PLSE problem is NP-hard.*

Theorem 2 (Easton et al. [6]) *The PLSE problem 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. [8]) *The PLSE problem is APX-hard.*

MaxSudoku is NP-hard in general [21]. Interestingly, it is still NP-hard even if each row (or column) is either empty or full, whereas the PLSE problem in this case can be solved in polynomial time [2].

MaxKenKen is NP-hard because it includes a certain NP-hard problem as a special case. It is the maximization problem version of BlockSum, in which the calculation type $\pi(B)$ of any block B is restricted to summation, i.e., $\pi(B) = +$. The decision problem to ask whether a given BlockSum instance has a solution or not is NP-complete even if every block is either 1-block or 2-block [10].

We investigate the computational hardness of MaxFutoshiki in the rest of the section. We summarize the result in **Table 1**. A MaxFutoshiki 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 $Q = \emptyset$; in this case, any LS is as an optimal solution. We leave the case of $Q \neq \emptyset$ open.

Let L be a non-empty PLS. When $Q = \emptyset$, the problem is equivalent to the ordinary PLSE problem, 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 MaxFutoshiki *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 PLSE problem. We transform a PLS L into a MaxFutoshiki instance on the $2n \times 2n$ grid. The $2n \times 2n$ grid consists of four gadgets: **(i)** the cells $(2k-1, 2\ell-1)$'-s whose row order and column order are odd, **(ii)** the cells $(2k, 2\ell)$'-s whose row order and column order are even, **(iii)** the cells $(2k-1, 2\ell)$'-s whose row order is odd and column order is even, and **(iv)** the cells $(2k, 2\ell-1)$'-s whose row order is even and column order is odd. Note that each gadget forms an $n \times n$ subgrid.

We copy the PLS L to the gadget (i), fill all the cells in (ii) with an arbitrary Latin square ranging from 1 to n, and fill all the cells in (iii) and those in (iv) with an arbitrary Latin square ranging from $n+1$ to $2n$ respectively. The assignment defined in this way is a $2n \times 2n$ PLS. The point is that empty cells appear only in (i) and that any two of them are not adjacent; any empty cell is adjacent to cells in (iii) and/or (iv), all of which are given integers from $n+1$ to $2n$. Then we allocate "smaller than" inequality signs around the empty cells arbitrarily.

Clearly, the PLSE instance L has an LS solution iff the constructed MaxFutoshiki instance has an LS solution. The construction time is polynomial. □

In the above proof, we can set the sign set Q to the full sign set.

Corollary 1 *When L is non-empty,* MaxFutoshiki *is still NP-hard even if Q is restricted to $Q = Q_L$.*

Table 1 Computational hardness of MaxFutoshiki.

		Sign set $Q \subseteq Q_L$ (empty)	Sign set $Q \subseteq Q_L$ (non-empty)
PLS L	(empty)	trivial	?
	(non-empty)	NP-hard (Theorem 1)	NP-hard (Theorem 4)

4. Approximation Algorithms

In this section, we present approximation algorithms for the puzzle maximization problems. The algorithms generalize existing ones for the PLSE problem. We borrow three types of algorithms from the literature: greedy algorithm, matching based approach, and local search. We show the result of our analyses in **Table 2**.

4.1 Greedy Algorithm

The greedy algorithm in this case refers to an algorithm as follows; starting from an empty solution, we repeat assigning an integer k to an empty cell (i, j) such that k is assignable to (i, j) until the solution is blocked. The algorithm runs in $O(n^3)$ time; we need $O(n^3)$ time to construct the list of triple (i, j, k)'-s such that k is assignable to (i, j). We then pick up at most n^2 triples from the list. Once we pick up one triple, $O(n)$ time is required to update the list.

For the PLSE problem, Kumar et al. [19] showed that it is a $1/3$-approximation algorithm.

Theorem 5 (Kumar et al. [19]) *For any instance of the PLSE problem, any blocked solution is a $1/3$-approximate solution.*

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

Proof: Let A be a blocked solution and A^* be an optimal solution. Suppose substituting the value $k = A^*_{ij}$ of any cell (i, j) in the optimal solution to A_{ij} in the blocked solution. Of course we cannot do so since A is blocked. More concretely, in A, (i, j) is already filled with an integer or the integer k is already assigned to a certain cell (i, j') in the row i or (i', j) in the column j. Then one sees that each value A_{ij} in the blocked solution A "prevents" us from copying at most three values in the optimal solution A^* to the same cells in A. We denote by $S_{ij}(A, A^*)$ the set of cells such that A_{ij} prevents the copy in that way;

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

Since $|S_{ij}(A, A^*)| \leq 3$, we have an upper bound of the summation of $|S_{ij}(A, A^*)|$ over A, that is,

Table 2 Approximation ratios: ε denotes any positive real number, δ denotes the largest number of inequality signs around a cell, and Δ denotes the largest block size. The bound $*$ can be achieved only for special instances which are explained in Section 4.2.

Problem	Greedy	Matching	Local search
PLSE	$1/3$ (Theorem 5)	$1/2$ (Theorem 9)	$2/3 - \varepsilon$ (Theorem 14) $3/4 - \varepsilon$ (Theorem 16)
MaxSudoku	$1/4$ (Theorem 6)	$1/2$ (Theorem 12)	$3/5 - \varepsilon$ (Theorem 17)
MaxFutoshiki	$1/(3+\delta)$ (Theorem 7)	$1/2$ (Theorem 10)	-
MaxKenKen	$1/(2+\Delta)$ (Theorem 8)	$1/2*$ (Theorem 11)	-

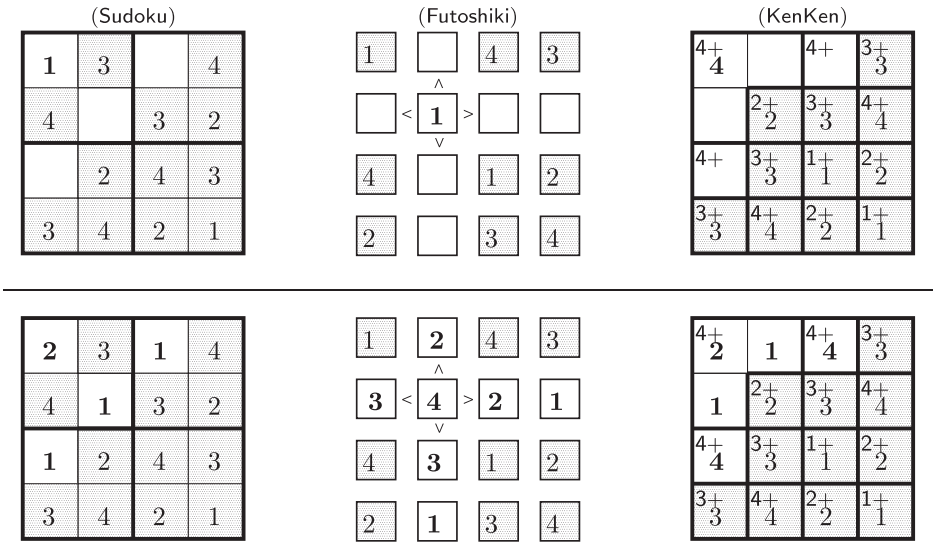

Fig. 2 Tight examples: blocked solution (upper) and optimal solution (lower).

$$\sum_{(i,j)\in[n]^2:\ A_{ij}>0} |S_{ij}(A,A^*)| \le 3|A|.$$

We claim that the size of the optimal solution should give a lower bound of the summation, that is,

$$|A^*| \le \sum_{(i,j)\in[n]^2:\ A_{ij}>0} |S_{ij}(A,A^*)|.$$

If not, A^* would have a non-empty cell (p,q) such that $(p,q) \notin \bigcup_{ij} S_{ij}(A,A^*)$. Then in A, (p,q) is empty and is not prevented by any A_{ij} from copying the integer A^*_{pq} to (p,q), which contradicts the fact that A is blocked.

Finally we have $|A^*| \le 3|A|$, which proves that A is a 1/3-approximate solution. □

The point is the size of $S_{ij}(A,A^*)$ in Eq. (2). Since it is at most three, any blocked solution is a 1/3-approximate solution. Then for a puzzle maximization problem with a peculiar constraint C, designing the similar set $S^C_{ij}(A,A^*)$ appropriately, we can prove any blocked solution to be a $1/\beta^C$-approximate solution in the analogous way, where β^C denotes an upper bound on the size of $S^C_{ij}(A,A^*)$.

For MaxSudoku, we can set the upper bound to $\beta^{\mathrm{SUD}} = 4$ by taking the set $S^{\mathrm{SUD}}_{ij}(A,A^*)$ as follows:

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

Theorem 6 *For any* MaxSudoku *instance, any blocked solution is a* 1/4*-approximate solution.*

For MaxFutoshiki, 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.

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

Theorem 7 *Suppose that we are given a* MaxFutoshiki *instance such that the number of inequality signs around a cell is at most δ. Then any blocked solution is a $1/(3+\delta)$-approximate solution.*

For MaxKenKen, the approximation ratio depends on the maximum size of the block over the instance, which we denote by Δ. We set the bound to $\beta^{\mathrm{KEN}} = 2 + \Delta$, taking the set $S^{\mathrm{KEN}}_{ij}(A,A^*)$ as follows. In the right hand, the size of the second set is at most $\Delta - 1$.

$$\begin{aligned} S^{\mathrm{KEN}}_{ij}(A,A^*) =& S_{ij}(A,A^*) \cup \{(p,q) \mid A_{pq} = 0, \\ & (i,j) \text{ and } (p,q) \text{ belong to the same block } B, \\ & \text{and } A^*_{pq} \text{ is not assignable to } (p,q) \text{ in } A\}. \end{aligned}$$

Theorem 8 *Suppose that we are given a* MaxKenKen *instance such that the block size is at most Δ. Then any blocked solution is a $1/(2+\Delta)$-approximate solution.*

The above bounds on approximation ratios are tight. We show tight examples in **Fig. 2**. In the figure, we see $\delta = 4$ for MaxFutoshiki and $\Delta = 3$ for MaxKenKen.

4.2 Matching Based Approach

Another approximation algorithm is based on matching. Let us call this algorithm Matching. First we describe how Matching behaves for the PLSE problem. Given an instance, it assigns the value k to empty cells in the order $k = 1, 2, \ldots, n$. Let $A^{(k-1)}$ be an interim solution such that the values from 1 to $k-1$ are assigned by Matching. Initially, $A^{(0)}$ is set to the empty PLS. Which empty cells are assigned k is determined by a maximum matching in a bipartite 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

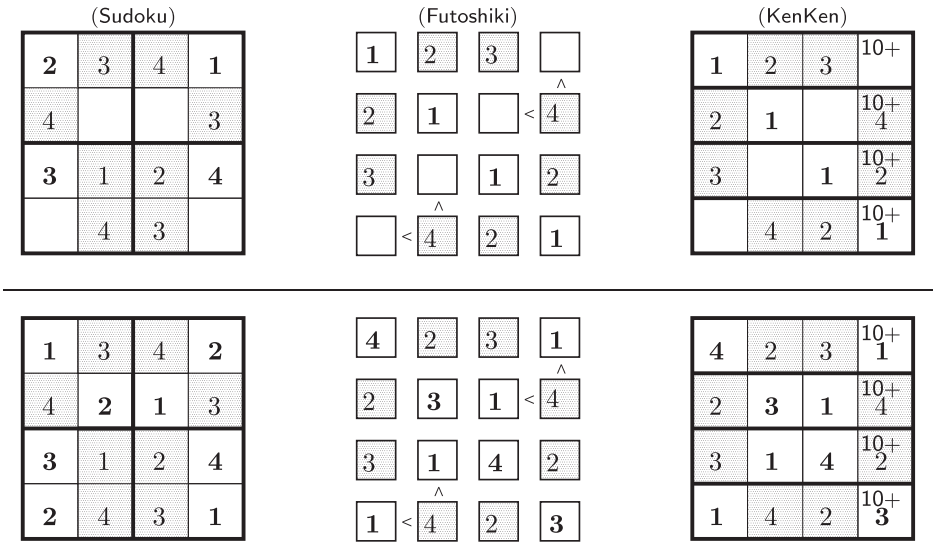

Fig. 4 Tight examples: MATCHING solution (upper) and optimal solution (lower).

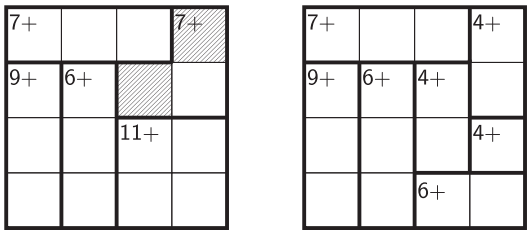

Fig. 3 MaxKenKen instances to which the algorithm MATCHING is not applicable (left) and is applicable (right).

$$E^{(k)} = \{(r_i, c_j) \in R \times C \mid k \text{ is assignable to } (i, j)\}$$

is the edge set. Computing a maximum matching $M \subseteq E^{(k)}$, MATCHING 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 $A^{(k)}$. MATCHING repeats this process from $k = 1$ to n and then halts, outputting $A^{(n)}$. The algorithm runs in $O(n^{3.5})$ time; for each value $k = 1, 2, \ldots, n$, we need $O(n^2)$ time to construct the bipartite graph $G^{(k)}$ and $O(n^{2.5})$ time to compute a maximum matching [14].

Theorem 9 (Kumar et al. [19]) *The algorithm* MATCHING *is a* $1/2$*-approximation algorithm for the PLSE problem.*

See the proof for Ref. [19]. 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 the PLSE problem, $A_{ij} = k$ never prevents any other cells out of row i and column j from taking k, i.e., the set $S_{ij}(A, A^*)$ in Eq. (2) contains no $(p, q) \in [n]^2$ such that both $p \neq i$ and $q \neq j$. To a puzzle maximization problem that has the property, we can apply the algorithm MATCHING directly so that the approximation ratio remains $1/2$. Then it is applicable to MaxFutoshiki in general.

Theorem 10 *The algorithm* MATCHING *is a* $1/2$*-approximation algorithm for* MaxFutoshiki.

The algorithm is not applicable to MaxKenKen in general because it does not have the above property. See the instance in the left of **Fig. 3**. The integer 4 is assignable to shaded cells, that is $(1, 4)$ and $(2, 3)$, but we cannot assign 4 to both cells simultaneously since it would violate Eq. (1) in the KenKen Condition. However, when every block is closed in one row or in one column, as in the right of Fig. 3, the algorithm is applicable so that the same approximation ratio is achieved.

Theorem 11 *Suppose that we are given a* MaxKenKen *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.*

We cannot apply MATCHING to MaxSudoku directly since, in MaxSudoku, a matching may correspond to a set of cells to which the same values cannot be assigned simultaneously; we have the constraint such that the values in an $(n_0 \times n_1)$-block should be all-different. However, we can apply the algorithm by modifying it in the following way. Recall that the roles of row, column and value in a Latin square are interchangeable. Even if we interchange their roles, the approximation ratio remains the same. Specifically we run the algorithm not by determining the location of each value $k = 1, 2, \ldots, n$ based on maximum matchings in row-column bipartite graphs, but by determining how the values are assigned in each row $i = 1, 2, \ldots, n$ based on maximum matchings in value-column bipartite graphs. In each row, any matching in a value-column bipartite graph provides a feasible assignment because it satisfies the all-different constraint in the row, as do the all-different constraints in the blocks that intersect the row.

Theorem 12 MaxSudoku *is* $1/2$*-approximable by the matching based approach.*

For all the three problems, the approximation ratio of MATCHING is $1/2$. This bound is tight. We show tight examples in **Fig. 4**.

4.3 Local Search

Let $t \geq 1$ denote a positive integer. We introduce the *t-set packing problem*; Let S be a finite set of elements. Suppose that we are given a family $\mathcal{F} = \{F_1, \ldots, F_q\}$ 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 [16].

For this problem, we consider a local search algorithm that behaves as follows. Let $r \geq 1$ be a positive integer. We use r as a parameter that represents the "radius" of local search. 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 13 (Hurkens et al. [15]) *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 within a polynomial time.*

Hajirasouliha et al. [8] applies the local search to the PLSE problem by reducing it to the 3-set packing problem. Given a PLSE instance in terms of a PLS L, the packing problem instance $\mathcal{F}$ is constructed as follows. Let the element set be $S = (R \times C) \cup (R \times [n]) \cup (C \times [n])$. Then let $\mathcal{F}$ contain a subset $\{(r_i, c_j), (r_i, k), (c_j, k)\} \subseteq S$ iff the value k is assignable to (i, j). Obviously there is one-to-one, size-preserving correspondence between the solution sets of the two problem instances.

Theorem 14 (Hajirasouliha et al. [8]) *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 the PLSE problem.*

Naïvely implemented, this local search runs in $O(n^{3r+5})$ time; We have $|\mathcal{F}| = O(n^3)$. Given a solution, we need $O(\binom{|\mathcal{F}|}{r+1})$ time to decide whether replacement is possible or not. The solution size is at most $O(n^2)$. Recently, Haraguchi [9] developed how to implement the local search efficiently for the PLSE problem, where we can identify whether replacement is possible or not in $O(n^2)$ time (resp., $O(n^3)$ time) when $r = 1$ (resp., $r = 2$).

We can apply the local search to MaxSudoku, regarding it as the 4-set packing problem. Suppose that a MaxSudoku instance is given. Let $\mathcal{B} = \{B_1, \ldots, B_n\}$ denote the set of $(n_0 \times n_1)$-blocks in the grid, and the element set be $S^{\mathrm{SUD}} = S \cup (\mathcal{B} \times [n])$. 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 is assignable an empty cell (i, j) that belongs to the block B_p. The solution correspondence is immediate.

Theorem 15 *For any $\varepsilon > 0$, there exists a $(1/2 - \varepsilon)$-approximation algorithm for* MaxSudoku.

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*. Further, Martin and Huiwen [7] improved its running time. This improves the approximation ratios for both PLSE and MaxSudoku.

Theorem 16 *For any $\varepsilon > 0$, there exists a $(3/4 - \varepsilon)$-approximation algorithm for the PLSE problem.*

Theorem 17 *For any $\varepsilon > 0$, there exists a $(3/5 - \varepsilon)$-approximation algorithm for* MaxSudoku.

5. Concluding Remarks

We have studied the computational hardness of MaxFutoshiki and the approximabilities of puzzle maximization problems. The results are summarized in Table 1 and Table 2 respectively.

We describe future work. We showed that, when the given sign set is non-empty, MaxFutoshiki is NP-hard if the given PLS is non-empty (see Theorem 4). Still, the complexity is open in the other case such that the given PLS is empty. It is interesting to improve the approximation ratios in Table 2. For the PLSE problem, 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; Table 2 implies $\rho^* \geq 3/4$. For puzzle maximization problems, whether they are APX-hard or not is open.

We have not made 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 for our future research.

References

[1] Andersson, D.: Hashiwokakero is NP-Complete, *Information Processing Letters*, Vol.109, No.19, pp.1145–1146 (2009).

[2] Béjar, R., Fernández, C., Mateu, C. and Valls, M.: The Sudoku Completion Problem with Rectangular Hole Pattern is NP-Complete, *Discrete Mathematics*, Vol.312, No.22, pp.3306–3315 (2012).

[3] Colbourn, C.J.: The Complexity of Completing Partial Latin Squares, *Discrete Applied Mathematics*, Vol.8, No.1, pp.25–30 (1984).

[4] Cygan, M.: Improved Approximation for 3-Dimensional Matching via Bounded Pathwidth Local Search, *FOCS 2013*, pp.509–518, IEEE Computer Society (2013).

[5] Demaine, E.D., Okamoto, Y., Uehara, R. and Uno, Y.: Computational Complexity and an Integer Programming Model of Shakashaka, *CCCG 2013*, pp.31–36, Carleton University, Ottawa, Canada (2013).

[6] Easton, T. and Parker, R.G.: On Completing Latin Squares, *Discrete Applied Mathematics*, Vol.113, No.2, pp.167–181 (2001).

[7] Fürer, M. and Yu, H.: Approximating the k-Set Packing Problem by Local Improvements, *ISCO 2014*, pp.408–420, Springer (2014).

[8] Hajirasouliha, I., Jowhari, H., Kumar, R. and Sundaram, R.: On Completing Latin Squares, *STACS 2007*, pp.524–535, Springer (2007).

[9] Haraguchi, K.: An Efficient Local Search for Partial Latin Square Extension Problem, *CoRR*, Vol.abs/1405.2571 (2014).

[10] Haraguchi, K. and Ono, H.: BlockSum is NP-Complete, *IEICE Trans. Inf. Syst.*, Vol.E96.D, No.3, pp.481–488 (2013).

[11] Haraguchi, K. and Ono, H.: Approximability of Latin Square Completion-Type Puzzles, *FUN 2014*, pp.218–229, Springer (2014).

[12] Hearn, R.A. and Demaine, E.D.: *Games, Puzzles, and Computation*, AK Peters, Limited (2009).

[13] Hirose, Y.: A Report on Essay Writing Instruction for Elementary School Students, *Bulletin of Ishinomaki Senshu University*, Vol.22 (supplement), pp.61–68 (2011). (online), available from ⟨http://www.senshu-u.ac.jp/ishinomaki/ilibif/lib/kenkyu-kiyou/22_02.html⟩ (in Japanese).

[14] Hopcroft, J.E. and Karp, R.M.: An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs, *SIAM J. Computing*, Vol.2, No.4, pp.225–231 (1973).

[15] Hurkens, C.A.J. and 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 J. Discrete Mathematics*, Vol.2, No.1, pp.68–72 (1989).

[16] Karp, R.M.: Reducibility among Combinatorial Problems, *Complexity of Computer Computations*, Plenum Press, pp.85–103 (1972).

[17] Kölker, J.: Kurodoko is NP-Complete., *J. Information Processing*, Vol.20, No.3, pp.694–706 (2012).

[18] Kudou, S., Sasaki, Y., Kawamura, S. and Haraguchi, K.: Application of ICT for Education Contribution of the University to Local Community, *ICT Strategy Meeting for Education Reform*, No.A-13, pp.130–131, Japan Universities Association for Computer Education (2011). (in Japanese).

[19] Kumar, S.R., Russell, A. and Sundaram, R.: Approximating Latin Square Extensions, *Algorithmica*, Vol.24, No.2, pp.128–138 (1999).

[20] Nikoli: available from ⟨http://www.nikoli.co.jp/en/⟩.

[21] Yato, T. and Seta, T.: Complexity and Completeness of Finding Another Solution and Its Application to Puzzles, *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, Vol.E86.A, No.5, pp.1052–1060 (2003).

Kazuya Haraguchi received his B.E., Master of Informatics, and Doctor of Informatics from Kyoto University, in 2001, 2003 and 2007, respectively. He is currently with the Department of Information and Management Science, Faculty of Commerce, Otaru University of Commerce. His interests include discrete algorithms, discrete optimization, and their application to artificial intelligence, operations research and recreational mathematics.

Hirotaka Ono received his B.E., M.E. and Doctor of Informatics degrees from Kyoto University in 1997, 1999 and 2002 respectively. He is currently an associate professor of the Department of Economic Engineering of Kyushu University. His research interests include combinatorial optimization, logical analysis of data and distributed algorithms. He is a member of IPSJ and the Operation Research Society of Japan.