Algorithmica (2020) 82:2039–2062
https://doi.org/10.1007/s00453-020-00685-8

The Generalized Definitions of the Two-Dimensional Largest Common Substructure Problems

Huang-Ting Chan[1] · Hsuan-Tsung Chiu[1] · Chang-Biau Yang[1] · Yung-Hsing Peng[2]

Received: 26 June 2017 / Accepted: 27 January 2020 / Published online: 14 February 2020

Abstract

The similarity of two one-dimensional sequences is usually measured by the longest common subsequence (LCS) algorithms. However, these algorithms cannot be directly extended to solve the two or higher dimensional data. Thus, for the two-dimensional data, computing the similarity with an LCS-like approach remains worthy of investigation. In this paper, we utilize a systematic way to give the generalized definition of the two-dimensional largest common substructure (TLCS) problem by referring to the traditional LCS concept. With various matching rules, eight possible versions of TLCS problems may be defined. However, only four of them are shown to be valid. We prove that all of these four TLCS problems are $\mathcal{NP}$-hard and $\mathcal{APX}$-hard. To accomplish the proofs, two of the TLCS problems are reduced from the 3-satisfiability problem, and the other two are reduced from the 3-dimensional matching problem.

Keywords Longest common subsequence · 2D string · NP-hard · APX-hard · Satisfiability · 3-Dimensional matching

This research work was partially supported by the Ministry of Science and Technology of Taiwan under Contract MOST 104-2221-E-110-018-MY3. And, it was also partially supported by the "Online and Offline Integrated Smart Commerce Platform (3/4)" of the Institute for Information Industry, which is subsidized by the Ministry of Economy Affairs of Taiwan.

✉ Chang-Biau Yang
cbyang@cse.nsysu.edu.tw

[1] Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan

[2] Innovative DigiTech-Enabled Applications and Services Institute Institute for Information Industry, Kaohsiung, Taiwan

1 Introduction

The similarity or distance of one-dimensional data can usually be measured by the algorithms for the *longest common subsequence* (LCS) problem [3, 17–21, 32] or the *edit distance* problem [31]. These problems have been extensively studied for several decades since 1970.

However, with the increasing motivation for computing the similarity of higher dimensional data, the LCS algorithms may not be applicable on various practical domains. For example, to find the functionality of the secondary structure of RNA and the tertiary structure of proteins [8], and to compute the similarity of the pictures for identifying hot spots in clinical diagnosis and criminal investigation are applications with data dimension higher than one. The demand for computing the similarity of two-dimensional data is increasing. In 1977, Knuth et al. [25] presented a method for finding the pattern matching in one-dimensional data. And then the researches for computing two-dimensional data focused on pattern matching [7, 26]. Therefore, finding the similarity of data with two or higher dimensions by LCS-like definitions becomes more interesting.

In 1987, Chang et al. [11] presented a way of representing a two-dimensional picture by 2D strings. With this method, one can transform a picture into a one-dimensional string with iconic indices [35]. They also presented three types of similarity relation between two iconic objects. Chang et al. [10] proposed a similarity retrieval algorithm, called 2D-string-LCS, to retrieve the most similar picture whose type similarity is the largest in the image database. In 1992, Lee and Hsu [28] proposed another similarity retrieval definition with 2D C-strings. The most previous researches of 2D strings focused on the similarity retrieval of images, pattern matching, image database design and its variants [9, 10, 12, 16, 27–29, 34]. With theoretical interest, in 2000, Guan et al. [16] proved that the problems of finding maximum similar subpictures of relations type-0 and type-1 of 2D strings are $\mathcal{NP}$-hard.

With another problem developing way, some researchers [2, 6] directly study the relationship of two matrices. They seemed not to know the above research progress on the 2D strings. Without transforming the two-dimensional data into 2D strings and iconic indices, in 1998, Baeza-Yates [6] presented a method to compute the edit distance between two matrices. In 2008, Amir et al. [2] gave another similarity definition of two matrices, which is the *two-dimensional largest common substructure* (TLCS, which was shortened as 2D LCS by Amir et al., but it is called TLCS in this paper.) problem. The matching rules of the TLCS problem given by Amir et al. are too strict to represent the matrix similarity. In fact, the definition of Amir et al. on the TLCS problem is exactly the same as the type-1 relation in 2D strings [11, 16]. However, there is no connection between 2D strings and the work of Amir et al. Amir et al. proved the $\mathcal{NP}$-hardness of the TLCS problem with another way.

In this paper, we present more general definitions of the TLCS problem. We first give eight possible matching rules for the TLCS problem. Then, we show that four of them are valid, and the others are invalid. Among these valid generalized TLCS problems, $P(ENE)$ (introduced later) is exactly the same as type-0 relation in 2D strings [11, 16]. The $\mathcal{NP}$-hardness proof of type-0 relation in 2D strings [16] can also be applied to the proof that $P(ENL)$ (introduced later) is $\mathcal{NP}$-hard.

The organization of this paper is given as follows. In Sect. 2, we will give some notations, and review the picture retrieval problem and the TLCS problem. In Sect. 3, we will present the formal definitions of *two-dimensional largest common substructure* (TLCS) problems with our matching rules. In Sect. 4, we prove that these TLCS problems are $\mathcal{NP}$-hard and $\mathcal{APX}$-hard. Finally, in Sect. 5, we will give our conclusions.

2 Preliminaries

In this section, we first present the notations used in this paper in the following. Let S be a sequence of elements, where $S = s_1 s_2 s_3 \dots s_{|S|}$.

- S is called a *sequence* or a *string*.
- s_i is the ith element of S.
- $|S|$ represents the length of S, i.e., the number of elements in S.
- $S_{i..j} = s_i s_{i+1} s_{i+2} \dots s_j$ denotes a substring starting from the ith element and ending at the jth element of S.

Let A and B be two matrices, with sizes $r_A \times c_A$ and $r_B \times c_B$ respectively, where $A = a_{1,1}, a_{1,2}, \dots, a_{1,c_A}, a_{2,1}, a_{2,2}, \cdots, a_{2,c_A}, \dots, a_{r_A,1}, a_{r_A,2}, \dots, a_{r_A,c_A}$ and $B = b_{1,1}, b_{1,2}, \dots, b_{1,c_B}, b_{2,1}, b_{2,2}, \dots, b_{2,c_B}, \dots, b_{r_B,1}, b_{r_B,2}, \dots, b_{r_B,c_B}$. In addition, each $a_{i,j} \in \Sigma, 1 \le i \le r_A, 1 \le j \le c_A$, and each $b_{p,q} \in \Sigma, 1 \le p \le r_B, 1 \le q \le c_B$.

- $a_{i,j}$ is the entry at the ith row and the jth column of matrix A.
- $a_{i,*}$ denotes the ith row and $a_{*,j}$ denotes the jth column of matrix A.
- r_A represents the number of the rows and c_A represents the number of columns of matrix A.
- $\pi_A = r_A \times c_A$ represents the size of matrix A, i.e., the number of elements in A.

2.1 The Picture Retrieval Problem

In the pictorial information retrieval problem, the goal is to retrieve pictures which satisfy a certain picture query criterion. For example, one may want to find all pictures with a car to the right of a tree. In 1987, Chang et al. [11] proposed a way for representing the two-dimensional data with 2D strings. The problem of pictorial information retrieval can be regarded as the 2D subsequence matching problem.

A 2D string can be represented with (S_r, S_c), where S_r and S_c denote the row relation and column relation, respectively. To represent the spatial relation, three special symbols $T = \{$ '=', '<', ':' $\}$ are used. For two objects, '=' represents the same spatial relation, '<' means the below-above spatial relation in S_r, or the left-right spatial relation in S_c. ':' represents that two objects are at the same position, that is, the same coordinate. Figure 1 shows an example for the 2D string.

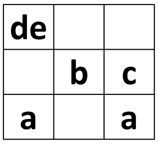

Fig. 1 An example of the 2D string defined by Chang et al. [11], where the 2D string is $(d : e < b = c < a = a, a = d : e < b < a = c)$

There are many ways for describing 2D strings, including absolute 2D strings, normal 2D strings, 2D C-strings and so on [9–11, 27–29].

Furthermore, Chang et al. defined three different types of matches. Let a_{i_1,j_1} and a_{i_2,j_2} be two distinct objects (elements) in matrix A, and b_{p_1,q_1} and b_{p_2,q_2} be two distinct objects in matrix B. Suppose that a_{i_1,j_1} matches to b_{p_1,q_1} and a_{i_2,j_2} matches to b_{p_2,q_2} with type-i relation. The definitions are given as follows [11].

- type-0: $(i_1 - i_2) \times (p_1 - p_2) \geq 0$ and $(j_1 - j_2) \times (q_1 - q_2) \geq 0$
- type-1: $[(i_1 - i_2) \times (p_1 - p_2) > 0$ or $i_1 - i_2 = p_1 - p_2 = 0]$ and $[(j_1 - j_2) \times (q_1 - q_2) > 0$ or $j_1 - j_2 = q_1 - q_2 = 0]$
- type-2: $i_1 - i_2 = p_1 - p_2$ and $j_1 - j_2 = q_1 - q_2$

In 2000, Guan et al. [16] proved that the problems of finding maximum similar subpictures of relations type-0 and type-1 are $\mathcal{NP}$-hard by reducing from the 3-satisfiability (3SAT) problem [13, 33].

2.2 The Two-Dimensional Largest Common Substructure Problem

In 2008, Amir et al. [2] defined the *two-dimensional largest common substructure* (TLCS, which originally was named as two-dimensional longest common substructure and shortened as 2D-LCS by Amir et al.) problem. With the concept of the one-dimensional LCS, they defined the TLCS as the largest identical substructure which is obtained from two input matrices by deleting zero or more of their entries. A substructure is obtained from a matrix by deleting some entries but preserving the orientation of remaining entries in the plane. In a common substructure of two input matrices, the orientations of any two common elements (matches) are the same as those in the input matrices. In the one-dimensional LCS, the LCS is ordered on a line. The concept is also applicable in the two-dimensional problem as well. Hence, the TLCS problem is to find the maximal identical symbols which preserve their order in the two matrices. The formal TLCS definition and matching rules are given as follows.

Definition 1 [2] Two-dimensional largest common substructure (TLCS) problem

Input: Matrix A of size $\pi_A = r_A \times c_A$ and matrix B of size $\pi_B = r_B \times c_B$.

Output: The maximum domain size of a one-to-one function $f : \{1, \dots, r_A\} \times \{1, \dots, c_A\} \rightarrow \{1, \dots, r_B\} \times \{1, \dots, c_B\}$ such that $a_{i,j} = b_{f(i,j)} = b_{p,q}$, $(i,j) \in$ *domain*(f). For every pair $(i_1,j_1), (i_2,j_2) \in$ *domain*(f) that $(p_1, q_1) = f(i_1,j_1)$ and $(p_2, q_2) = f(i_2,j_2)$, the following holds:

- $i_1 < i_2$ if and only if $p_1 < p_2$.
- $j_1 < j_2$ if and only if $q_1 < q_2$.
- $i_1 = i_2$ if and only if $p_1 = p_2$.
- $j_1 = j_2$ if and only if $q_1 = q_2$.

Amir et al. also proved that the TLCS problem is $\mathcal{NP}$-hard by reducing from the k-clique problem. In fact, the TLCS problem defined by Amir et al. (Definition 1) is exactly the same as the similarity relation of type-1 on 2D strings, which has been proved to be $\mathcal{NP}$-hard [16]. However, Amir et al. did not refer to the papers studying the latter problem [11, 16].

As an example, Fig. 2 shows three matrices A, B and C. It is clear that matrix B can be obtained by rotation within some columns of matrix A. Thus, A and B are more similar than A and C. However, the TLCS size of A and B is equal to that of A and C, which are both 3. Consequently, in this paper, we propose some different matching rules to disclose more similarity as how we look.

2.3 Approximability Classes

Definition 2 [1, 14, 15] $\mathcal{NP}$ optimization ($\mathcal{NPO}$) problem

An $\mathcal{NPO}$ problem F is represented by a 4-tuple $(\oint_F, Sol_F, oj_F, opt_F)$, explained as follows.

(i) $\oint_F$ is the set of the instances of F, and $\oint_F$ is recognizable in polynomial time.
(ii) $Sol_F(I)$ denotes the set of the feasible solutions with instance $I \in \oint_F$. $Sol_F(I)$ should be recognized in polynomial time and there exists a polynomial function p such that $\forall sol \in Sol_F(I)$, $|sol| \leq p(|I|)$.

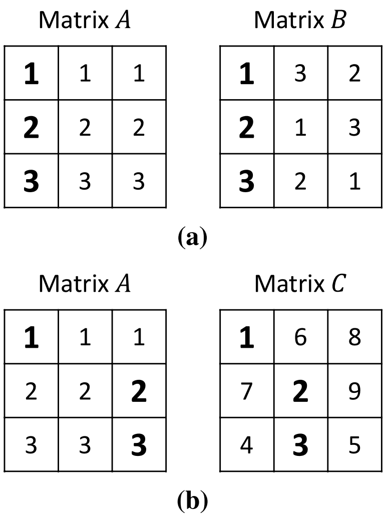

Fig. 2 The TLCS of matrix A and matrix B or matrix C with the definition of Amir et al., where the answers are shown by the boldface symbols. **a** The TLCS of matrices A and B. **b** The TLCS of matrices A and C

(iii) $oj_F(I, sol)$ denotes the *objective function*, which is an integer and nonnegative function, where $I \in \oint_F$ and $sol \in Sol_F(I)$.
(iv) $opt_F \in \{max, min\}$ means that F is a maximization or minimization problem.

Solving an $\mathcal{NPO}$ problem F with $I \in \oint_F$ corresponds to find a feasible solution $sol \in Sol_F(I)$ which maximizes $oj_F(I, sol)$ over all feasible solutions if $opt_F = max$ or minimizes $oj_F(I, sol)$ if $opt_F = min$ [15]. Taking the TLCS problem as an example, let $\oint_{TLCS}$ be the set of all possible pairs of matrices. Let I be an instance of matrices A and B, with $|A| = r_A \times c_A$ and $|B| = r_B \times c_B$. For a solution sol of instance I, there is a polynomial function $p(x) = x$, such that $|sol| \leq r_A \times c_A$ and $|sol| \leq r_B \times c_B$. We have $oj_F(I, sol) = |sol|$ for instance I and any solution $sol \in Sol(I)$. Also, we have $opt_{TLCS} = max$, because we want to find the common substructure with maximal size.

Given $I \in \oint_F$, we denote $opt_F(I)$ as the optimal solution set for I. The approximation ratio is defined as $Rat_F = \max\{\frac{|opt_F(I)|}{oj_F(I,sol)}, \frac{oj_F(I,sol)}{|opt_F(I)|}\}$. It is clear that the ratio is always greater than or equal to 1. The closer to 1 the approximation ratio is, the better performance the algorithm has.

Definition 3 [30] Approximability ($\mathcal{APX}$) problem
An $\mathcal{NPO}$ problem F is in $\mathcal{APX}$ if F can be approximated within a constant c, that is , there exists a polynomial-time algorithm D such that for all instances $I \in \oint_F$, $D(I) \in Sol_F$ and $Rat_F(D(I)) \leq c$, where $Rat_F(D(I))$ is the approximation ratio of F.

L-reduction was proposed by Papadimitriou and Yannakakis [30] in 1991. In various proofs of $\mathcal{APX}$ problems, L-reduction is the easiest and the most restrictive one.

Definition 4 [30] L-reduction
Given two $\mathcal{NPO}$ problems F and G, and a polynomial-time transformation $f : F \rightarrow G$, f is an L-reduction from F to G if there exist positive constants δ and μ such that the following conditions hold for every instance I of F.

(i) $|opt_G(f(I))| \leq \delta \cdot |opt_F(I)|$,
(ii) For every feasible solution sol of $f(I)$ with objective value $oj_G(f(I), sol) = oj_2$, we can find a solution, in polynomial time, sol' of I with $oj_F(I, sol') = oj_1$ such that $|opt_F(I) - oj_1| \leq \mu |opt_G(f(I)) - oj_2|$.

A problem is $\mathcal{APX}$-complete if it is in $\mathcal{APX}$ and it is $\mathcal{APX}$-hard [5]. Unless $\mathcal{P}$=$\mathcal{NP}$, an $\mathcal{APX}$-hard problem is not in $\mathcal{PTAS}$ (polynomial time approximation scheme) [5]. An optimization problem in $\mathcal{PTAS}$ has an algorithm that can produce a solution within a factor $1 + \epsilon$ (or $1 - \epsilon$ for a maximization problem) of the optimal solution in polynomial time, where ϵ is any fixed constant. Unless $\mathcal{P}$=$\mathcal{NP}$, $\mathcal{PTAS}$ is a proper subset of $\mathcal{APX}$ [22].

For example, the Euclidean traveling salesman problem is in $\mathcal{PTAS}$ [4]. However, the *maximum 3-satisifiability bounded by K* (MAX 3SAT-K) problem with $K \geq 3$ is $\mathcal{MAXSNP}$-complete, proved by Papadimitriou and Yannakakis [30], where each variable appears in at most K clauses. And, the *maximum 3-dimensional matching bounded by K* (MAX 3DM-K) problem with $K \geq 3$ is $\mathcal{MAXSNP}$-complete, proved by Kann [23], where each symbol appears at most K times in input. It should be noticed that $\mathcal{MAXSNP} \subseteq \mathcal{APX}$ and $\mathcal{MAXSNP}$-complete $\subseteq \mathcal{APX}$-complete [30]. Thus, MAX 3SAT-K and MAX 3DM-K are both $\mathcal{APX}$-complete. In addition, their proofs also imply that 3SAT-K and 3DM-K (without maximization) are both $\mathcal{NP}$ -complete.

Definition 5 [30] Maximum 3-satisifiability bounded by 3 (MAX 3SAT-3) problem
Input: A set of variables X, and a set of disjunctive clauses C over the variables X, where each clause consists of at most three variables, and each variable occurs in at most $K = 3$ clauses.
Output: Truth assignment of X for the maximal number of satisfied clauses.

For example, suppose that there is a set $X = \{x_1, x_2, x_3, x_4\}$ and $C = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_3} \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4)$, where the number of each variable occurrences in C is bounded by 3. C is satisfied by $\{x_1, \overline{x_2}, \overline{x_3}, x_4\}$, and the number of satisfied clauses is 3.

Definition 6 [23] Maximum 3-dimensional matching bounded by 3 (MAX 3DM-3) problem
Input: A set $M \subseteq X \times Y \times Z$, where X, Y and Z are disjoint finite sets. Each element of X, Y and Z occurs in M at most $K = 3$ times.
Output: The 3-dimensional matching M' with the maximal size, where $M' \subseteq M$ and each element of X, Y and Z occurs in M' at most once.

For example, suppose that there is a set $M = \{(x_1, y_1, z_1), (x_1, y_3, z_3), (x_2, y_3, z_2), (x_3, y_2, z_4), (x_1, y_2, z_2)\}$, where each element of X, Y and Z occurs at most 3 times. Then $M' = \{(x_1, y_1, z_1), (x_2, y_3, z_2), (x_3, y_2, z_4)\}$ is the maximum 3-dimensional matching of M.

The *3-dimensional matching* (3DM) problem (without bound K) is the general version, which determines if there exists a 3-dimensional matching with a constant size k. In 1972, Karp presented some $\mathcal{NP}$-complete problems, including the 3DM problem [24].

3 Problem Definitions

In this section, we present our formal definitions of the two-dimensional largest common substructure (TLCS) problem and give some properties.

3.1 Operator Definitions

To find the similarity of the two-dimensional data, we extend the concept for defining the one-dimensional LCS problem to the TLCS problem. The inputs of the TLCS problem are two matrices of characters, and we aim to find the *largest common substructure*. We regard matrix A of size $r_A \times c_A$ as the index set $T_A = \{(i,j) | 1 \le i \le r_A$ and $1 \le j \le c_A\}$. A *substructure* of A is defined as an index subset of T_A. A *common substructure* U of two matrices A and B is defined as a set of index pairs, one from A and the other from B, which obey the matching rules. Since each index of A or B has two tuples, U consists of four-tuples. That is, the set of *matching index pairs* is defined as $U = \{ (i,j,p,q) \mid a_{i,j} = b_{p,q}$, and the matching rules are obeyed.$\}$. The matching rules determine whether every two matches (two four-tuples) are allowed or not. We will introduce the matching rules in detail later. In the following, we first present the definition of the TLCS problem, which is the generalization of Definition 1 given by Amir et al. [2].

Definition 7 Two-dimensional largest common substructure problem (TLCS)
Input: Matrix A with size $\pi_A = r_A \times c_A$ and matrix B with size $\pi_B = r_B \times c_B$.
Output: The set U of four-tuple indices of A and B with maximum cardinality, where every two elements in U obey the matching rules (given later).

The goal of the TLCS problem is to find the maximum number of matches such that every two matches satisfy the matching rules (determining whether the two matches are valid or not). In other words, the TLCS problem aims to find the *largest common substructure* U between two character matrices. In Fig. 3, we show an example for the TLCS definition with the matching rule $P(ENL)$, one of the matching rules we define (explained later).

In the traditional one-dimensional LCS problem, two matching rules have to be obeyed for one common sequence: (1) no duplicate match; and (2) no crossing matches. Similarly, in the TLCS problem, these two rules are still applicable. However, the concept of no crossing matches becomes more complicated in TLCS. In other words, the orientation of TLCS would be more various, and the matching rules would become more diverse. In the TLCS problem, there are eight possible directions in the orientation. They are left, right, upper, lower, upper left, upper right, lower left and lower right. Therefore, we divide the two-dimensional matching rules into two parts, *logical operator* and *orientation*. The logical operator combines the relationship between rows and columns. For logical operators,

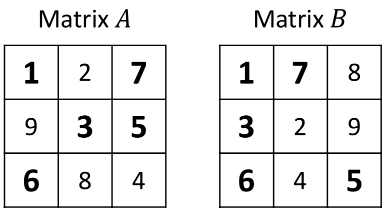

Fig. 3 An example of the TLCS with $P(ENL)$ of two matrices

there are two possible ways, one is *And* (N) for considering both the orientation of rows and columns together, and the other is *Or* (R) for treating rows and columns as independent. The formal definition of logical operators is given as follows.

Definition 8 Logical operator

And (N): both row and column relationships are satisfied.

Or (O): the row relationship or the column relationships is satisfied.

For two elements a_{i_1,j_1} and a_{i_2,j_2}, there are two different conditions of the orientation. *Corner* means that $i_1 \neq i_2$ and $j_1 \neq j_2$, and *side* means that $i_1 = i_2$ or $j_1 = j_2$.

There are two possible corner rules. Suppose two elements a_{i_1,j_1} and a_{i_2,j_2} are in matrix A, and b_{p_1,q_1} and b_{p_2,q_2} are in matrix B, where (i_1,j_1,p_1,q_1), $(i_2,j_2,p_2,q_2) \in U$. For the row relationship, if $i_1 < i_2$ in matrix A, we may stipulate either $p_1 < p_2$ or $p_1 \leq p_2$, but not $p_1 > p_2$ in matrix B, because the orientation should be preserved. Similarly, for the column relationship, there are still two feasible cases, $q_1 < q_2$ or $q_1 \leq q_2$, when $j_1 < j_2$. On the other hand, if $i_1 > i_2$ in matrix A, we may have either $p_1 > p_2$ or $p_1 \geq p_2$ in matrix B. We denote $p_1 < p_2$ or $p_1 > p_2$ as *less than* (L), and $p_1 \leq p_2$ or $p_1 \geq p_2$ as *less than or equal to* (E) when $i_1 < i_2$ or $i_1 > i_2$, respectively. The definition of the corner is given as follows.

Definition 9 Corner: let two elements a_{i_1,j_1} and a_{i_2,j_2} be in A where $i_1 \neq i_2$ and $j_1 \neq j_2$. And let two elements b_{p_1,q_1} and b_{p_2,q_2} be in matrix B. (i_1,j_1,p_1,q_1), $(i_2,j_2,p_2,q_2) \in U$.

Less than (L): $p_1 < p_2$ when $i_1 < i_2$, and $q_1 < q_2$ when $j_1 < j_2$.

Less than or equal to (E): $p_1 \leq p_2$ when $i_1 < i_2$, and $q_1 \leq q_2$ when $j_1 < j_2$.

In Fig. 4, we show examples for the definition of corners. Suppose the element a_{i_1,j_1} is of character α, and a_{i_2,j_2} is marked by dark in matrix A. In matrix B, the allowed positions of element b_{p_2,q_2} are illustrated by dark cells if (i_1,j_1,p_1,q_1), $(i_2,j_2,p_2,q_2) \in U$.

For the definition of side, if $i_1 = i_2$ and $j_1 < j_2$, then we need to keep only the column orientation because we could say that the element (i_1,j_1) is on the left side of (i_2,j_2). There are also two possible side rules, *less than* (L) and *less than or equal to* (E). In addition, we also define rule A for Amir et al. [2].

Definition 10 Side: Let two elements a_{i_1,j_1} and a_{i_2,j_2} be in matrix A where $i_1 = i_2$ or $j_1 = j_2$. And let two elements b_{p_1,q_1} and b_{p_2,q_2} be in matrix B. (i_1,j_1,p_1,q_1), $(i_2,j_2,p_2,q_2) \in U$.

Less than (L): $p_1 < p_2$ when $i_1 < i_2$ and $j_1 = j_2$. $q_1 < q_2$ when $i_1 = i_2$ and $j_1 < j_2$.

Less than or equal to (E): $p_1 \leq p_2$ when $i_1 < i_2$ and $j_1 = j_2$. $q_1 \leq q_2$ when $i_1 = i_2$ and $j_1 < j_2$.

Amir's rule (A): $p_1 < p_2$ and $q_1 = q_2$ when $i_1 < i_2$ and $j_1 = j_2$. $p_1 = p_2$ and $q_1 < q_2$ when $i_1 = i_2$ and $j_1 < j_2$.

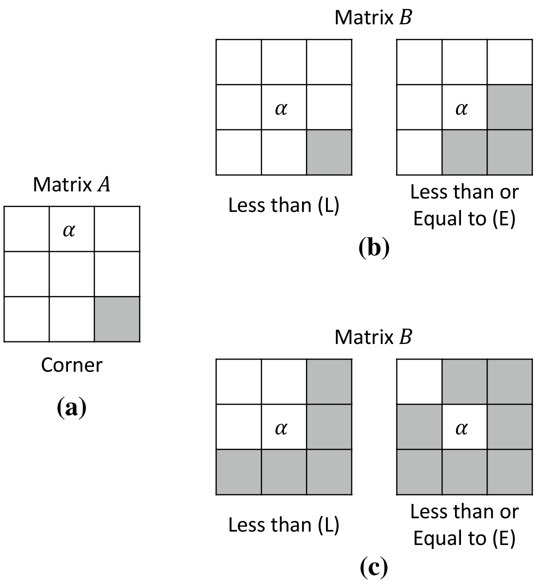

Fig. 4 Examples for illustrating corners. **a** Matrix A. **b** The definition of corners with logical operator N. **c** The definition of corners with logical operator O

Figure 5 shows two examples for the definition of sides. Suppose a_{i_1,j_1} is of character α, and a_{i_2,j_2} is marked by dark in matrix A. In matrix B, the allowed positions of element b_{p_2,q_2} are presented by dark cells if $(i_1,j_1,p_1,q_1), (i_2,j_2,p_2,q_2) \in U$.

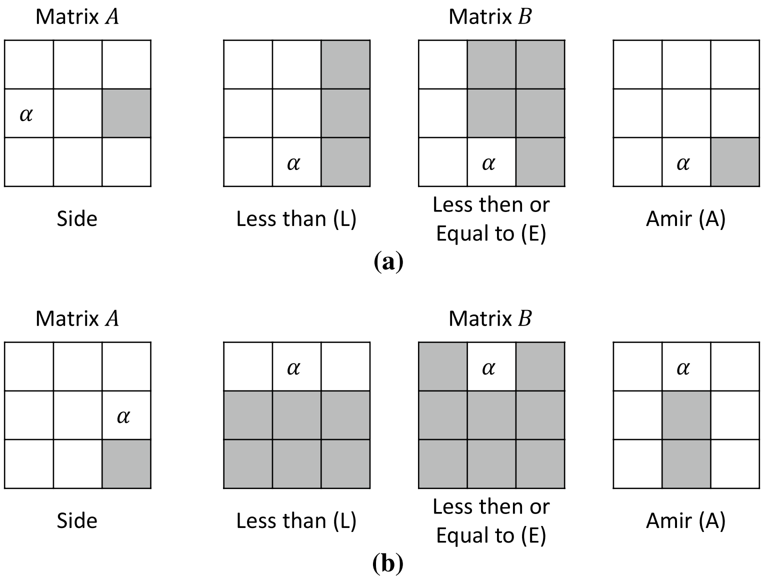

Fig. 5 Two examples for illustrating the definition of sides

3.2 Problem Definitions

When we define a TLCS problem, we partition the definitions into three parts and denote the TLCS problem as *P*(*corner*, *operator*, *side*), where *operator* denotes the logical operator for dealing with the relationship between the row index and column index, *corner* and *side* represent the row and column relationship. Accordingly, there are $2 \times 2 \times 2$ different versions of problems, two possible values (*L*, *E*) for *corner*, two possible values (*And*, *Or*) for *operator*, and two possible values (*L*, *E*) for *side*. Besides, the TLCS problem defined by Amir et al. is expressed as *P*(*LNA*). Let a_{i_1,j_1} and a_{i_2,j_2} be in matrix *A*, and b_{p_1,q_1} and b_{p_2,q_2} be in matrix *B*, where (i_1,j_1,p_1,q_1), $(i_2,j_2,p_2,q_2) \in U$. Then we present the matching rules of the eight possible versions of the TLCS problem in the following.

P(*LNL*): *P*(***L**ess than*, *a**N**d*, ***L**ess than*)

(i) $i_1 < i_2, j_1 < j_2 \rightarrow p_1 < p_2$ and $q_1 < q_2$
(ii) $i_1 < i_2, j_1 > j_2 \rightarrow p_1 < p_2$ and $q_1 > q_2$
(iii) $i_1 < i_2, j_1 = j_2 \rightarrow p_1 < p_2$
(iv) $i_1 = i_2, j_1 < j_2 \rightarrow q_1 < q_2$

P(*LNE*): *P*(***L**ess than*, *a**N**d*, *less than or **E**qual to*)

(i) $i_1 < i_2, j_1 < j_2 \rightarrow p_1 < p_2$ and $q_1 < q_2$
(ii) $i_1 < i_2, j_1 > j_2 \rightarrow p_1 < p_2$ and $q_1 > q_2$
(iii) $i_1 < i_2, j_1 = j_2 \rightarrow p_1 \leq p_2$
(iv) $i_1 = i_2, j_1 < j_2 \rightarrow q_1 \leq q_2$

P(*ENL*): *P*(*less than or **E**qual to*, *a**N**d*, ***L**ess than*)

(i) $i_1 < i_2, j_1 < j_2 \rightarrow p_1 \leq p_2$ and $q_1 \leq q_2$
(ii) $i_1 < i_2, j_1 > j_2 \rightarrow p_1 \leq p_2$ and $q_1 \geq q_2$
(iii) $i_1 < i_2, j_1 = j_2 \rightarrow p_1 < p_2$
(iv) $i_1 = i_2, j_1 < j_2 \rightarrow q_1 < q_2$

P(*ENE*): *P*(*less than or **E**qual to*, *a**N**d*, *less than or **E**qual to*)

(i) $i_1 < i_2, j_1 < j_2 \rightarrow p_1 \leq p_2$ and $q_1 \leq q_2$
(ii) $i_1 < i_2, j_1 > j_2 \rightarrow p_1 \leq p_2$ and $q_1 \geq q_2$
(iii) $i_1 < i_2, j_1 = j_2 \rightarrow p_1 \leq p_2$
(iv) $i_1 = i_2, j_1 < j_2 \rightarrow q_1 \leq q_2$

P(*LOL*): *P*(***L**ess than*, ***O**r*, ***L**ess than*)

(i) $i_1 < i_2, j_1 < j_2 \rightarrow p_1 < p_2$ or $q_1 < q_2$
(ii) $i_1 < i_2, j_1 > j_2 \rightarrow p_1 < p_2$ or $q_1 > q_2$
(iii) $i_1 < i_2, j_1 = j_2 \rightarrow p_1 < p_2$
(iv) $i_1 = i_2, j_1 < j_2 \rightarrow q_1 < q_2$

$P(LOE)$: $P($*Less than*, *Or*, *less than or Equal to*$)$

(i) $i_1 < i_2, j_1 < j_2 \rightarrow p_1 < p_2$ or $q_1 < q_2$
(ii) $i_1 < i_2, j_1 > j_2 \rightarrow p_1 < p_2$ or $q_1 > q_2$
(iii) $i_1 < i_2, j_1 = j_2 \rightarrow p_1 \le p_2$
(iv) $i_1 = i_2, j_1 < j_2 \rightarrow q_1 \le q_2$

$P(EOL)$: $P($*less than or Equal to*, *Or*, *Less than*$)$

(i) $i_1 < i_2, j_1 < j_2 \rightarrow p_1 \le p_2$ or $q_1 \le q_2$
(ii) $i_1 < i_2, j_1 > j_2 \rightarrow p_1 \le p_2$ or $q_1 \ge q_2$
(iii) $i_1 < i_2, j_1 = j_2 \rightarrow p_1 < p_2$
(iv) $i_1 = i_2, j_1 < j_2 \rightarrow q_1 < q_2$

$P(EOE)$: $P($*less than or Equal to*, *Or*, *less than or Equal to*$)$

(i) $i_1 < i_2, j_1 < j_2 \rightarrow p_1 \le p_2$ or $q_1 \le q_2$
(ii) $i_1 < i_2, j_1 > j_2 \rightarrow p_1 \le p_2$ or $q_1 \ge q_2$
(iii) $i_1 < i_2, j_1 = j_2 \rightarrow p_1 \le p_2$
(iv) $i_1 = i_2, j_1 < j_2 \rightarrow q_1 \le q_2$

In the above eight matching rules, we can find that $P(ENE)$ is exactly the same as the type-0 relation of 2D strings [11, 16] and $P(LNA)$, defined by Amir et al., is exactly the same as the type-1 relation [11, 16]

In the above definitions, not all matching rules can form a valid TLCS problem. A valid matching rule should be *symmetric*. That is, the optimal solution of two matrices A and B should also be the optimal solution of matrices B and A.

Let R_x denote the binary relation for satisfying $P(x)$ matching rule, where $x \in \{ LNL, LNE, ENL, ENE, LOL, LOE, EOL, EOE \}$. Let a_{i_1,j_1} and a_{i_2,j_2} be two distinct elements in matrix A, and b_{p_1,q_1} and b_{p_2,q_2} be two distinct elements in matrix B. We say that $(i_1,j_1,i_2,j_2)\ R_x\ (p_1,q_1,p_2,q_2)$ if they satisfy the matching rule x. Furthermore, the TLCS problem with matching rule x is a valid problem if and only if the binary relation R_x is symmetric.

Lemma 1 *The binary relation R_{ENL} is symmetric.*

Proof Suppose $(i_1,j_1,i_2,j_2)\ R_{ENL}\ (p_1,q_1,p_2,q_2)$. Here, we want to prove that $(p_1,q_1,p_2,q_2)\ R_{ENL}\ (i_1,j_1,i_2,j_2)$ is true.

To prove the symmetry of R_{ENL}, four cases are considered as follows.

Case 1 a_{i_1,j_1} and a_{i_2,j_2} are *corner*, b_{p_1,q_1} and b_{p_2,q_2} are *corner*.
Case 2 a_{i_1,j_1} and a_{i_2,j_2} are *side*, b_{p_1,q_1} and b_{p_2,q_2} are *side*.
Case 3 a_{i_1,j_1} and a_{i_2,j_2} are *corner*, b_{p_1,q_1} and b_{p_2,q_2} are *side*.
Case 4 a_{i_1,j_1} and a_{i_2,j_2} are *side*, b_{p_1,q_1} and b_{p_2,q_2} are *corner*.

In cases 1 and 2, the condition of the orientation of the a_{i_1,j_1}, a_{i_2,j_2} and b_{p_1,q_1}, b_{p_2,q_2} are the same. Hence, if (i_1,j_1,i_2,j_2) R_{ENL} (p_1,q_1,p_2,q_2), then (p_1,q_1,p_2,q_2) R_{ENL} (i_1,j_1,i_2,j_2). In rule (i) of $P(ENL)$, $p_1 \le p_2$ and $q_1 \le q_2$ when $i_1 < i_2$ and $j_1 < j_2$. Hence, in case 3, it should be: (1) $p_1 < p_2$, $q_1 = q_2$; or (2) $p_1 = p_2$, $q_1 < q_2$. In these two conditions, according to $P(ENL)$ rules (iii) and (iv) respectively, (p_1,q_1,p_2,q_2) R_{ENL} (i_1,j_1,i_2,j_2) is true. Consequently, R_{ENL} is symmetric in case 3. Similarly, case 4 is also symmetric. Thus, R_{ENL} is symmetric. □

With similar proofs (omitted here), we have the following lemma.

Lemma 2 *The binary relations R_{ENE}, R_{LOL} and R_{LOE} are symmetric.*

By the above two lemmas, the following theorem can be obtained.

Theorem 1 *Each of the TLCS problem with $P(ENL)$, $P(ENE)$, $P(LOL)$ and $P(LOE)$ is valid.*

In Fig. 6, we present an example for showing that R_{ENL} is symmetric. In Fig. 6a, b, suppose $a_{i_1,j_1} = \alpha$ and a_{i_2,j_2} is marked by dark in matrix A. In matrix B, the allowed positions of element b_{p_2,q_2} are illustrated by dark cells. In Fig. 6c, d, we can find that (2, 2, 3, 3) R_{ENL} (2, 2, 2, 3) and (2, 2, 2, 3) R_{ENL} (2, 2, 3, 3), respectively. This example can also be applied to R_{ENE}, R_{LOL} and R_{LOE}.

Lemma 3 *The binary relations R_{LNL}, R_{LNE}, R_{EOL} and R_{EOE} are not symmetric.*

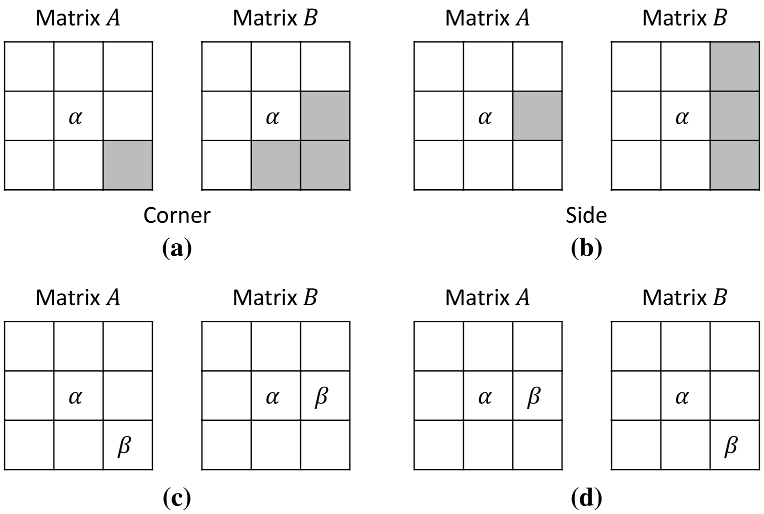

Fig. 6 Examples of the valid matching rule for $P(ENL)$. **a** The definition of corner for $P(ENL)$. **b** The definition of side for $P(ENL)$. **c** An example for $P(ENL)$. **d** Another example for $P(ENL)$

Proof For each binary relation, we give an example to show that it is not symmetric. It is clear that (2, 2, 2, 3) R_{LNL} (2, 2, 3, 3). However, (2, 2, 3, 3) R_{LNL} (2, 2, 2, 3) is not true. Hence, R_{LNL} is not symmetric.

Similarly, (2, 2, 2, 3) R_{LNE} (2, 2, 3, 3). However, (2, 2, 3, 3) R_{LNE} (2, 2, 2, 3) is not true. (2, 2, 3, 3) R_{EOL} (2, 2, 2, 1). Rule (iv) of $P(EOL)$ can be rewritten as $[i_1 = i_2, j_1 > j_2 \rightarrow q_1 > q_2]$. Thus, (2, 2, 2, 1) R_{EOL} (2, 2, 3, 3) is not true. (2, 2, 3, 3) R_{EOE} (2, 2, 2, 1). Rule (iv) of $P(EOE)$ can be rewritten as $[i_1 = i_2, j_1 > j_2 \rightarrow q_1 \geq q_2]$. However, (2, 2, 2, 1) R_{EOE} (2, 2, 3, 3) is not true.

Thus, this lemma holds. □

Figure 7 illustrates an example that R_{LNE} is not symmetric. In Fig. 7a, b, suppose $a_{i_1,j_1} = \alpha$ and a_{i_2,j_2} is marked by dark in matrix A. In matrix B, the allowed positions of element b_{p_2,q_2} are illustrated by dark cells. It is clear that in Fig. 7c, (2, 2, 2, 3) R_{LNE} (2, 2, 3, 3), but in Fig. 7d, (2, 2, 3, 3) R_{LNE} (2, 2, 2, 3) is not true.

With Lemma 3, we get the following result.

Theorem 2 *The problems $P(LNL)$, $P(LNE)$, $P(EOL)$ and $P(EOE)$ are not valid.*

We summarize the possible TLCS problems with various matching rules in Table 1. In the table, (i_1,j_1) and (i_2,j_2) are in matrix A, and the relation inside each cell indicates the relation between (p_1, q_1) and (p_2, q_2) in matrix B. Taking $P(ENL)$ as an example, two matching elements in the corner positons of matrix A with $[i_1 < i_2$ and $j_1 < j_2]$ means that $[p_1 \leq p_2$ and $q_1 \leq q_2]$ in matrix B. And two matching elements in the side positons of matrix A with $[i_1 < i_2$ and $j_1 = j_2]$ means that $p_1 < p_2$ in matrix B. RA and CA represent row alignment and column alignment,

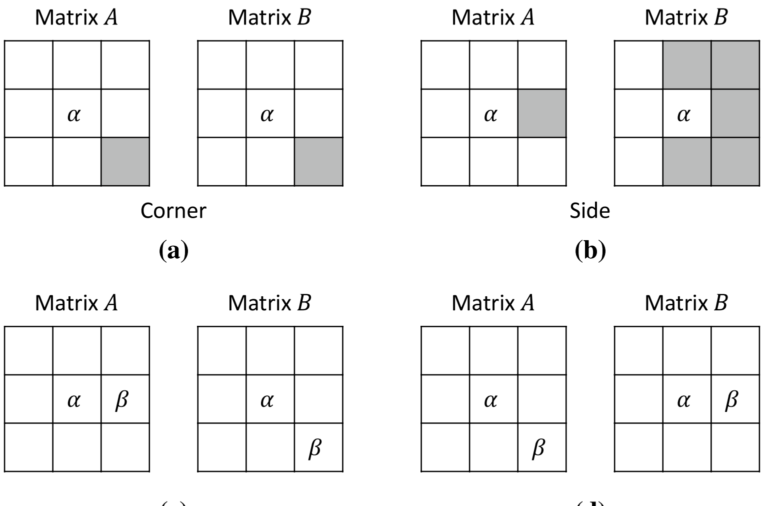

Fig. 7 An example of the invalid matching rule for $P(LNE)$. **a** The definition of corner for $P(LNE)$. **b** The definition of side for $P(LNE)$. **c** An example for the valid $P(LNE)$ matching rule. **d** An example of an invalid matching rule for $P(LNE)$, which shows that R_{LNE} is not symmetric with respect to **c**

Table 1 The TLCS problems with various matching rules. Here, (i_1,j_1) and (i_2,j_2) are in matrix A, and each cell in column (i_1, i_2) indicates the relation between (p_1, p_2) of matrix B and in column (j_1,j_2) indicates the relation between (q_1, q_2)

Problem	Operator	Corner				Side				Valid
		$i_1 < i_2$	$j_1 < j_2$	$i_1 < i_2$	$j_1 > j_2$	$i_1 < i_2$	$j_1 = j_2$	$i_1 = i_2$	$j_1 < j_2$	
RA	and	$<$		$<$		$<$		$=$	$<$	Y
CA			$<$		$>$	$<$	$=$		$<$	Y
LNA		$<$	$<$	$<$	$>$	$<$	$=$	$=$	$<$	Y
LNL		$<$	$<$	$<$	$>$	$<$			$<$	N
LNE		$<$	$<$	$<$	$>$	$\le$			$\le$	N
ENL		$\le$	$\le$	$\le$	$\ge$	$<$			$<$	Y
ENE		$\le$	$\le$	$\le$	$\ge$	$\le$			$\le$	Y
LOL	or	$<$	$<$	$<$	$>$	$<$			$<$	Y
LOE		$<$	$<$	$<$	$>$	$\le$			$\le$	Y
EOL		$\le$	$\le$	$\le$	$\ge$	$<$			$<$	N
EOE		$\le$	$\le$	$\le$	$\ge$	$\le$			$\le$	N

respectively. Row alignment means that the two-dimensional matrix is mapped into a one-dimensional sequence by the row-major scheme. Hence, the row alignment only considers the order of the rows. Similarly, the column alignment corresponds to the column-major scheme.

Definition 11 Let CU_x denote the largest common substructure (optimal solution) of $P(x)$, $x \in \{ENL, ENE, LOL, LOE, RA, CA, LNA\}$, where RA and CA denote the problem with the row alignment and column alignment, respectively. And, let $|CU_x|$ denote its size.

In the above definition, for example, CU_{ENL} denotes the largest common substructure of $P(ENL)$, and $|CU_{ENL}|$ denotes its size. CU_{ENL} is represented by the maximal set of the four-tuples of the matching index pairs $\{(i,j,p,q) \mid a_{i,j} = b_{p,q}$, and the $P(ENL)$ matching rule is obeyed.$\}$. The solution sizes of these TLCS problems can be partially ordered. Figure 8 illustrates the solutions with two input matrices A and B shown in Fig. 8a. We can see that with different matching rules, it is possible to obtain a different solution with the same input. We have the following Theorem.

Theorem 3 $|CU_{LNA}| \le |CU_{ENL}| \le (|CU_{ENE}|, \quad |CU_{LOL}|) \le |CU_{LOE}|,$ *and* $|CU_{LNA}| \le (|CU_{RA}|, |CU_{CA}|) \le |CU_{LOL}| \le |CU_{LOE}|.$

Here, we do not give the formal proof. With the definitions of these TLCS problems, one can easily show the correctness of the above theorem. We illustrate the conceptual solutions in Fig. 9, from which the above theorem can also be verified.

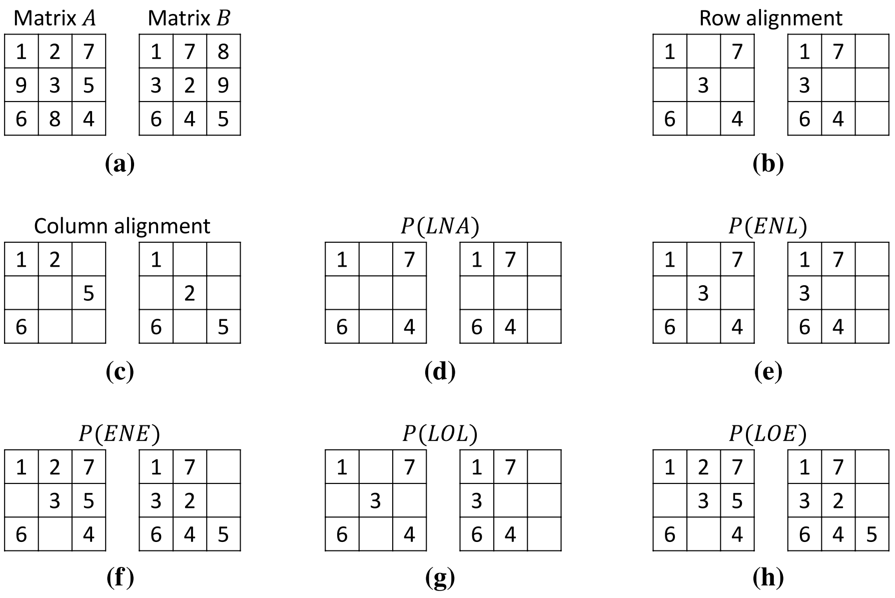

Fig. 8 Comparison of solution sizes of various TLCS problems. **a** Matrices A and B. **b**–**h** The solutions of various TLCS problems

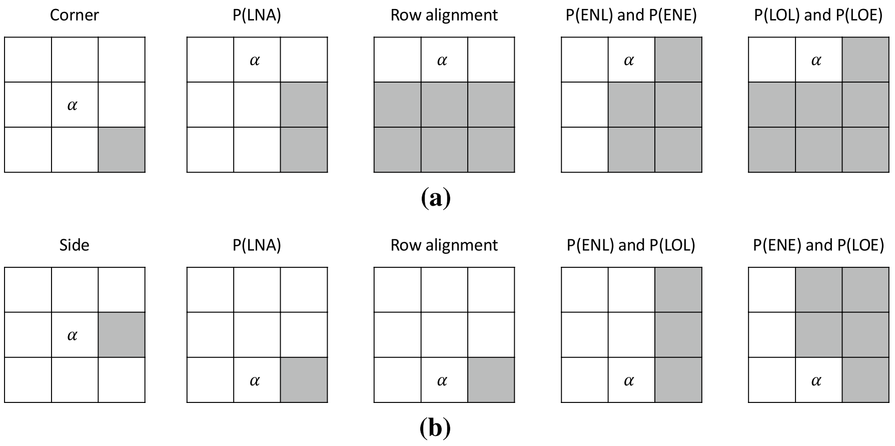

Fig. 9 The conceptual solutions of various TLCS problems. **a** The solution of corners for various TLCS problems. **b** The solution of sides for various TLCS problems

4 Proofs of $\mathcal{NP}$-Hardness and $\mathcal{APX}$-Hardness

In this section, we prove that $P(ENL)$, $P(ENE)$, $P(LOL)$ and $P(LOE)$ are $\mathcal{NP}$-hard and $\mathcal{APX}$-hard.

4.1 $\mathcal{NP}$-Hardness of *P*(*ENL*) and *P*(*ENE*)

Guan et al. [16] proved that the maximum similar subpicture problems of type-0 and type-1 are $\mathcal{NP}$-hard by reducing from 3SAT. Note that type-0 is exactly the same as $P(ENE)$. In addition, $P(ENL)$ can also be proved to be $\mathcal{NP}$-hard by reducing from 3SAT with the same transformation. The transformation Γ_{ENL} is described as follows. The instance of 3SAT is represented by a set of Boolean variables $X = \{x_1, x_2, \cdots, x_n\}$, and a Boolean formula $C = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, where each clause has the form $C_t = (v_{t,1} \vee v_{t,2} \vee v_{t,3})$, $1 \le t \le m$, $v_{t,1} = x_{\kappa_{t,1}}$ or $\overline{x_{\kappa_{t,1}}}$, $v_{t,2} = x_{\kappa_{t,2}}$ or $\overline{x_{\kappa_{t,2}}}$, $v_{t,3} = x_{\kappa_{t,3}}$ or $\overline{x_{\kappa_{t,3}}}$, $1 \le \kappa_{t,1} \ne \kappa_{t,2} \ne \kappa_{t,3} \le n$. Matrices A and B of $P(ENL)$, where $|A| = |B| = 2n \times 3m$, are constructed as follows [16].

$$a_{i,j} = \begin{cases} l^t_u \begin{cases} \text{if } i = 2\kappa - 1, j = 3(t-1) + u, \text{ where the } u\text{th literal of } C_t \text{ is } x_\kappa; \\ \text{if } i = 2\kappa, j = 3(t-1) + u, \text{ where the } u\text{th literal of } C_t \text{ is } \overline{x_\kappa}; \\ u = 1, 2, 3; \end{cases} \\ \alpha \ \ \text{otherwise.} \end{cases} \tag{1}$$

$$b_{i,j} = \begin{cases} l^t_u \begin{cases} \text{if } i = 2\kappa, j = 3t + 1 - u, \text{where the } u\text{th literal of } C_t \text{ is } x_\kappa; \\ \text{if } i = 2\kappa - 1, j = 3t + 1 - u, \text{where the } u\text{th literal of } C_t \text{ is } \overline{x_\kappa}; \\ u = 1, 2, 3; \end{cases} \\ \beta \ \ \text{otherwise.} \end{cases} \tag{2}$$

Figure 10 shows an example of transformation Γ_{ENL}. Every two rows of A or B correspond to one variable of 3SAT, but they are in reverse order in A and B. Every three columns of A or B correspond to one clasue of 3SAT, but they are in reverse order in A and B. Rows 1 and 2 correspond to x_1 and $\overline{x_1}$ in matrix A, but they are for $\overline{x_1}$ and x_1 in matrix B, respectively. Columns 1, 2 and 3 correspond to the first, second and third literals in matrix A, but they are for the third, second and first literals in matrix B, respectively. For example, l^2_1, l^2_2 and l^2_3 correspond to $C_2 = (x_1 \vee \overline{x_3} \vee \overline{x_2})$.

In transformation Γ_{ENL}, if l^t_u is selected into the solution CU_{ENL}, then it means that its corresponding element $v_{t,u}$ in C_t is assigned to be true. Any pair of elements l^t_1, l^t_2 and l^t_3 are never selected in CU_{ENL} at the same time (proved in Lemma 4). It implies that if one of the literals in C_t is true, then C_t is true for 3SAT. Accordingly, $|CU_{ENL}| \le m$.

Lemma 4 *Suppose that* $a_{i_1,j_1} = b_{p_1,q_1} = l^t_u$ *and* $a_{i_2,j_2} = b_{p_2,q_2} = l^t_{u'}$, *where* $1 \le u \ne u' \le 3$. *Then* $(a_{i_1,j_1}, b_{p_1,q_1})$ *and* $(a_{i_2,j_2}, b_{p_2,q_2})$ *cannot be both in* CU_{ENL}.

Proof If $u < u'$, then $j_1 < j_2$ and $q_1 > q_2$ do not obey the column relationship of corner for $P(ENL)$. Similarly, if $u > u'$, then $j_1 > j_2$ and $q_1 < q_2$ do not obey the column relationship of corner for $P(ENL)$. Thus, the lemma holds. □

$$X = \{x_1, x_2, x_3, x_4\}$$
$$C = \left\{ \begin{array}{l} (x_1 \vee x_2 \vee x_3)\wedge \\ (x_1 \vee \overline{x_3} \vee \overline{x_2})\wedge \\ (\overline{x_1} \vee \overline{x_2} \vee x_4) \end{array} \right\}$$

(a)

Matrix A

	1	2	3	4	5	6	7	8	9
1	l_1^1	α	α	l_1^2	α	α	α	α	α
2	α	α	α	α	α	α	l_1^3	α	α
3	α	l_2^1	α	α	α	α	α	α	α
4	α	α	α	α	α	l_3^2	α	l_2^3	α
5	α	α	l_3^1	α	α	α	α	α	α
6	α	α	α	α	l_2^2	α	α	α	α
7	α	α	α	α	α	α	α	α	l_3^3
8	α	α	α	α	α	α	α	α	α

Matrix B

	1	2	3	4	5	6	7	8	9
1	β	β	β	β	β	β	β	β	l_1^3
2	β	β	l_1^1	β	β	l_1^2	β	β	β
3	β	β	β	l_3^2	β	β	β	l_2^3	β
4	β	l_2^1	β	β	β	β	β	β	β
5	β	β	β	β	l_2^2	β	β	β	β
6	l_3^1	β	β	β	β	β	β	β	β
7	β	β	β	β	β	β	β	β	β
8	β	β	β	β	β	β	l_3^3	β	β

(b)

Fig. 10 An example of transformation Γ_{ENL} for proving the $\mathcal{NP}$-hardness of $P(ENL)$. **a** Instance sets X and C in 3SAT. **b** Matrices A and B of $P(ENL)$, constructed from X and C

Suppose that two elements l_u^t and $l_{u'}^{t'}$ correspond to the same variable x_κ, one for x_κ and the other for $\overline{x_\kappa}$, where $t \neq t'$ and $1 \leq u, u' \leq 3$. They cannot be both in CU_{ENL}. It is proved in the following lemma.

Lemma 5 *Suppose that C_t and $C_{t'}$ have a common variable, $t \neq t'$, one is x_κ and the other is $\overline{x_\kappa}$, $a_{i_1,j_1} = b_{p_1,q_1} = l_u^t$, and $a_{i_2,j_2} = b_{p_2,q_2} = l_{u'}^{t'}$. Then $(a_{i_1,j_1}, b_{p_1,q_1})$ and $(a_{i_2,j_2}, b_{p_2,q_2})$ cannot be both in CU_{ENL}.*

Proof If C_t contains x_κ and $C_{t'}$ contains $\overline{x_\kappa}$, then $i_1 < i_2$ and $p_1 > p_2$ do not obey the row relationship of corner for $P(ENL)$. Similarly, if C_t contains $\overline{x_\kappa}$ and $C_{t'}$contains x_κ, then $i_1 > i_2$ and $p_1 < p_2$ do not obey the row relationship of corner for $P(ENL)$. □

Except the conflict conditions mentioned in Lemmas 4 and 5, any other pair of matchings can be both in CU_{ENL}.

With Lemmas 4 and 5, 3SAT reduces to $P(ENL)$, and thus we have the following result.

Theorem 4 *The TLCS problem with $P(ENL)$ is $\mathcal{NP}$-hard.*

The proof for $P(ENL)$ can be applied to $P(ENE)$.

Theorem 5 [16] *The TLCS problem with $P(ENE)$ is $\mathcal{NP}$-hard.*

4.2 $\mathcal{APX}$-Hardness of *P*(*ENL*) and *P*(*ENE*)

For proving the $\mathcal{APX}$-hardness, we use the same transformation Γ_{ENL} from the MAX 3SAT-3 problem, instead of the 3SAT problem, to $P(ENL)$ and $P(ENE)$.

Theorem 6 *The TLCS problem with $P(ENL)$ is $\mathcal{APX}$-hard.*

Proof For instance I, let $k = |opt_{MAX\,3SAT-3}(I)|$. We have $opt_{ENL}(f(I)) = k = |opt_{MAX\,3SAT-3}(I)|$. For each $sol \in Sol_{ENL}(f(I))$ with $|sol| = oj_2$, we can get corresponding $sol' \in Sol_{MAX\,3SAT-3}(I)$ with $|sol'| = oj_1 = oj_2$. $|opt_{MAX\,3SAT-3}(I) - oj_1| = k - oj_1 = k - oj_2 = |opt_{ENL}(f(I)) - oj_2|$. Hence, Γ_{ENL} is an L-reduction from MAX 3SAT-3 to $P(ENL)$ with $\delta = 1$ and $\mu = 1$. □

The proof for $P(ENL)$ can be applied to $P(ENE)$, thus the following theorem is also obtained.

Theorem 7 *The TLCS problem with $P(ENE)$ is $\mathcal{APX}$-hard.*

4.3 $\mathcal{NP}$-Hardness of *P*(*LOL*) and *P*(*LOE*)

We prove that $P(LOL)$ and $P(LOE)$ are $\mathcal{NP}$-hard by reducing from the 3DM problem. The transformation Γ_{LOL} for $P(LOL)$ is described as follows. The input instance of 3DM is represented by a set $M = \{M_1, M_2, \cdots, M_m\} \subseteq X \times Y \times Z$, where X, Y and Z are disjoint finite sets, $|M| = m$, and $|X| + |Y| + |Z| = n$. Let $M_t = (x_\kappa, y_{\kappa'}, z_{\kappa''}) \in M$, where $1 \le t \le m, x_\kappa \in X, y_{\kappa'} \in Y, z_{\kappa''} \in Z, 1 \le \kappa \le |X|, 1 \le \kappa' \le |Y|$ and $1 \le \kappa'' \le |Z|$. The matrices A and B of $P(LOL)$ are constructed as follows, where $|A| = m \times (2m + n)$ and $|B| = (m + nm) \times 5m$.

$$a_{i,j} = \begin{cases} l_u^t & \text{if } i = t, \text{ and } j = 2(t-1) + u, u = 1, 2; \\ l_x^t & \text{if } i = t, \text{ and } j = 2m + \kappa; \\ l_y^t & \text{if } i = t, \text{ and } j = 2m + |X| + \kappa'; \\ l_z^t & \text{if } i = t, \text{ and } j = 2m + |X| + |Y| + \kappa''; \\ \alpha & \text{otherwise.} \end{cases} \tag{3}$$

$$b_{i,j} = \begin{cases} l_u^t & \text{if } i = t, \text{ and } j = 5(m-t) + 3 + u, u = 1, 2; \\ l_x^t & \text{if } i = m(1+\kappa) - (t-1), \text{ and } j = 5(m-t) + 1; \\ l_y^t & \text{if } i = m(1 + |X| + \kappa') - (t-1), \text{and } j = 5(m-t) + 2; \\ l_z^t & \text{if } i = m(1 + |X| + |Y| + \kappa'') - (t-1), \text{and } j = 5(m-t) + 3; \\ \beta & \text{otherwise.} \end{cases} \tag{4}$$

Figure 11 shows an example of Γ_{LOL}, where $m = 3$ and $n = 6$. Each row in matrix A and every submatrix of size $(m + nm) \times 5 = 21 \times 5$ in matrix B correspond to one

$$M = \left\{ \begin{array}{l} (x_1, y_1, z_1), \\ (x_1, y_2, z_1), \\ (x_2, y_2, z_2) \end{array} \right\}$$

(a)

Matrix A

	1	2	3	4	5	6	7	8	9	10	11	12
1	l_1^1	l_2^1	α	α	α	α	l_x^1	α	l_y^1	α	l_z^1	α
2	α	α	l_1^2	l_2^2	α	α	l_x^2	α	α	l_y^2	l_z^2	α
3	α	α	α	α	l_1^3	l_2^3	α	l_x^3	α	l_y^3	α	l_z^3

Matrix B

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	β	β	β	β	β	β	β	β	β	β	β	β	β	l_1^1	l_2^1
2	β	β	β	β	β	β	β	β	l_1^2	l_2^2	β	β	β	β	β
3	β	β	β	l_1^3	l_2^3	β	β	β	β	β	β	β	β	β	β
4	β	β	β	β	β	β	β	β	β	β	β	β	β	β	β
5	β	β	β	β	β	l_x^2	β	β	β	β	β	β	β	β	β
6	β	β	β	β	β	β	β	β	β	β	l_x^1	β	β	β	β
7	l_x^3	β	β	β	β	β	β	β	β	β	β	β	β	β	β
8	β	β	β	β	β	β	β	β	β	β	β	β	β	β	β
9	β	β	β	β	β	β	β	β	β	β	β	β	β	β	β
10	β	β	β	β	β	β	β	β	β	β	β	β	β	β	β
11	β	β	β	β	β	β	β	β	β	β	β	β	β	β	β
12	β	β	β	β	β	β	β	β	β	β	β	l_y^1	β	β	β
13	β	l_y^3	β	β	β	β	β	β	β	β	β	β	β	β	β
14	β	β	β	β	β	β	l_y^2	β	β	β	β	β	β	β	β
15	β	β	β	β	β	β	β	β	β	β	β	β	β	β	β
16	β	β	β	β	β	β	β	β	β	β	β	β	β	β	β
17	β	β	β	β	β	β	β	l_z^2	β	β	β	β	β	β	β
18	β	β	β	β	β	β	β	β	β	β	β	β	l_z^1	β	β
19	β	β	l_z^3	β	β	β	β	β	β	β	β	β	β	β	β
20	β	β	β	β	β	β	β	β	β	β	β	β	β	β	β
21	β	β	β	β	β	β	β	β	β	β	β	β	β	β	β

(b)

Fig. 11 An example of transformation Γ_{LOL} for proving the $\mathcal{NP}$-hardness of $P(LOL)$. **a** An input set M of the 3DM problem. **b** Matrices A and B of $P(LOL)$, constructed from M

element M_t of M. l_x^t, l_y^t and l_z^t correspond to elements x_κ, $y_{\kappa'}$ and $z_{\kappa''}$ of M_t. l_1^t and l_2^t are the expanded symbols generated from M_t.

In matrix A of Fig. 11, row t corresponds to M_t, columns 1 to 6 are for l_1^t and l_2^t, columns 7 and 8 are for x_1 and x_2, respectively, columns 9 and 10 are for y_1 and y_2, respectively. In matrix B, columns 11 to 15 are for M_1, columns 6 to 10 are for M_2 (reverse order). Rows 1 to 3 correspond to l_1^t and l_2^t. Rows 4 to 6 correspond to x_1, in which rows 4, 5 and 6 are for M_3, M_2 and M_1 (reverse order), respectively. Rows 7 to 9 correspond to x_2, in which rows 7, 8 and 9 are for M_3, M_2 and M_1, respectively. Rows 10 to 15 are for Y, and rows 16 to 21 are for Z.

Each symbol, except α and β, appears exactly once in matrix A and once in matrix B. Obviously, $|CU_{LOL}| \leq 5m$ (including all symbols, but excluding α and β). To obtain a tighter bound, for one element $M_t \in M$, exactly one of two possible matches (l_1^t, l_2^t) and (l_x^t, l_y^t, l_z^t) can be made between A and B. Thus, $2m \leq |CU_{LOL}| \leq 3m$.

In the following, we will prove that $|CU_{LOL}| = 2m + k$ if and only if there exists a 3-dimensional matching with size k in M. Moreover, if $|CU_{LOL}| = 2m + k$ and (l^t_x, l^t_y, l^t_z) are three of the common elements in CU_{LOL}, then M_t will be picked as one match in the optimal solution of M. The formal proof is accomplished by the following lemmas.

Lemma 6 *Suppose that* $a_{i_1,j_1} = b_{p_1,q_1} = l^t_u$, *where* $u = 1, 2$, *and* $a_{i_2,j_2} = b_{p_2,q_2} = l^t_v$, *where* $v = x, y, z$. *Then* $(a_{i_1,j_1}, b_{p_1,q_1})$ *and* $(a_{i_2,j_2}, b_{p_2,q_2})$ *cannot be both in* CU_{LOL}.

Proof By the transformation Γ_{LOL} defined in (3) and (4), it is clear that $i_1 = i_2$, $j_1 < j_2$, $p_1 < p_2$ and $q_1 > q_2$. We have that $j_1 < j_2$ and $q_1 > q_2$ do not obey the column relationship of side $(i_1 = i_2)$ in $P(LOL)$. Thus, $(a_{i_1,j_1}, b_{p_1,q_1})$ and $(a_{i_2,j_2}, b_{p_2,q_2})$ cannot be both in CU_{LOL}. □

Lemma 7 *Suppose that* M_t *and* $M_{t'}$ *have a common element,* $a_{i_1,j_1} = b_{p_1,q_1} = l^t_v$ *and* $a_{i_2,j_2} = b_{p_2,q_2} = l^{t'}_v$, *where* $v = x, y, z$ *and* $t \neq t'$. *Then* $(a_{i_1,j_1}, b_{p_1,q_1})$ *and* $(a_{i_2,j_2}, b_{p_2,q_2})$ *cannot be both in* CU_{LOL}.

Proof Assume that $t < t'$. By the transformation Γ_{LOL} defined in (3) and (4), it is clear that $i_1 < i_2$, $j_1 = j_2$, $p_1 > p_2$ and $q_1 > q_2$. We have that $i_1 < i_2$ and $p_1 > p_2$ do not obey the row relationship of side $(j_1 = j_2)$ in $P(LOL)$. A similar result can be obtained when $t > t'$. Therefore, $(a_{i_1,j_1}, b_{p_1,q_1})$ and $(a_{i_2,j_2}, b_{p_2,q_2})$ cannot be both in CU_{LOL}. □

Except the conflict conditions in Lemmas 6 and 7, any other pair of matchings can be both in CU_{LOL}.

Lemma 8 $|CU_{LOL}| = 2m + k$ *if and only if there exists a 3-dimensional matching with size* k.

Proof If there exists a 3-dimensional matching with size k, it is obvious that in Γ_{LOL}, $2(m - k) + 3k = 2m + k$ elements in matrix A can be matched with elements in B.

With Lemma 6, we pick up either 2 or 3 elements from each row in A. If 3 elements are picked up in one row of A, then l^t_x, l^t_y and l^t_z are the targets. It means that M_t is picked up in the solution of 3DM. If $|CU_{LOL}| = 2m + k$, it means that k rows of A are picked up with 3 elements. With Lemma 7, the picked elements are all distinct in X, Y or Z. Therefore, by Γ_{LOL}, the matches we pick up in matrices A and B of $P(LOL)$ correspond to a 3-dimensional matching with size k. □

With Lemma 8, 3DM reduces to $P(LOL)$, and thus we have the following result.

Theorem 8 *The TLCS problem with* $P(LOL)$ *is* $\mathcal{NP}$*-hard.*

Similarly, the reduction and Lemma 8 can also be applied to $P(LOE)$, thus we have the following result.

Theorem 9 *The TLCS problem with $P(LOE)$ is $\mathcal{NP}$-hard.*

4.4 $\mathcal{APX}$-Hardness of $P(LOL)$ and $P(LOE)$

For proving the $\mathcal{APX}$-hardness of $P(LOL)$ and $P(LOE)$, we use the same transformation Γ_{LOL} from the MAX 3DM-3 problem, instead of the 3DM problem, to $P(LOL)$ and $P(LOE)$.

If there is a matching $(x, y, z) \in opt_{MAX\,3DM\text{-}3}$, then at most 6 matches may not be in $opt_{MAX\,3DM\text{-}3}$. For example, if $(x, y, z) \in opt_{MAX\,3DM\text{-}3}$, then (x, y_1, z_1), (x, y_2, z_2), (x_1, y, z_1), (x_2, y, z_2), (x_1, y_1, z), (x_2, y_2, z) cannot be in $opt_{MAX\,3DM\text{-}3}$. It is clear that $m \le (6+1)k = 7k$, where $k = |opt_{MAX\,3DM\text{-}3}|$.

Theorem 10 *The TLCS problem with $P(LOL)$ is $\mathcal{APX}$-hard.*

Proof For instance I, let $k = |opt_{MAX\,3DM\text{-}3}(I)|$. We have $|opt_{LOL}(f(I))| = 2m + k \le 14k + k = 15k$. That is, $|opt_{LOL}(f(I))| \le 15|opt_{MAX\,3DM\text{-}3}(I)|$. Thus, $\delta = 15$. For each $sol \in Sol_{LOL}(f(I))$ with $|sol| = oj_2$, we can get corresponding $sol' \in Sol_{MAX\,3DM\text{-}3}(I)$ with $|sol'| = oj_1 \ge oj_2 - 2m$, since three elements picked up in one row of A of $P(LOL)$ corresponds to one picked M_t of MAX 3DM-3, and two or fewer elements picked up in one row of A of $P(LOL)$ corresponds to nothing in MAX 3DM-3. It implies $|opt_{MAX\,3DM\text{-}3}(I) - oj_1| = k - oj_1 \le k - (oj_2 - 2m) = (2m + k) - oj_2 = |opt_{LOL}(f(I)) - oj_2|$. Hence, Γ_{LOL} is an L-reduction from MAX 3DM-3 to $P(LOL)$ with $\delta = 15$ and $\mu = 1$. $\square$

Similarly, the same reduction and proof can also be applied to $P(LOE)$, thus we have the following result.

Theorem 11 *The TLCS problem with $P(LOE)$ is $\mathcal{APX}$-hard.*

5 Conclusion

In this paper, we present the more general definitions of the *two-dimensional largest common substructure* (TLCS) problems with various matching rules. To meet different demands, we present different types of corners, operators and sides. With various combinations of corners, operators and sides, we define four valid TLCS problems, including $P(ENL)$, $P(ENE)$, $P(LOL)$ and $P(LOE)$. We prove that all of these four TLCS problems are $\mathcal{NP}$-hard and $\mathcal{APX}$-hard. In the future, it is worthy to design approximation algorithms for the TLCS problems. It is also interesting to discover whether these problems are $\mathcal{APX}$-complete or not.

References

1. Amaldi, E., Kann, V.: The complexity and approximability of finding maximum feasible subsystems of linear relations. Theor. Comput. Sci. **147**(1–2), 181–210 (1995)
2. Amir, A., Hartman, T., Kapah, O., Shalom, B.R., Tsur, D.: Generalized LCS. Theor. Comput. Sci. **409**(3), 438–449 (2008)
3. Ann, H.Y., Yang, C.B., Tseng, C.T., Hor, C.Y.: A fast and simple algorithm for computing the longest common subsequence of run-length encoded strings. Inf. Process. Lett. **108**(6), 360–364 (2008)
4. Arora, S.: Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. JACM **45**(5), 753–782 (1998)
5. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties. Springer, Berlin (2012)
6. Baeza-Yates, R.A.: Similarity in two-dimensional strings. In: International Computing and Combinatorics Conference, pp. 319–328. Springer, Berlin (1998)
7. Bird, R.S.: Two dimensional pattern matching. Inf. Process. Lett. **6**(5), 168–170 (1977)
8. Branden, C.I., et al.: Introduction to Protein Structure. Garland Science, New York (1999)
9. Chang, S., Li, Y.: Representation of multi-resolution symbolic and binary pictures using 2D H-strings. In: IEEE Workshop on Languages for Automation: Symbiotic and Intelligent Robots, 1988, pp. 190–195. IEEE (1988)
10. Chang, S.K., Jungert, E., Li, Y.: Representation and retrieval of symbolic pictures using generalized 2D strings. In: 1989 Symposium on Visual Communications, Image Processing, and Intelligent Robotics Systems, pp. 1360–1372. International Society for Optics and Photonics (1989)
11. Chang, S.K., Shi, Q.Y., Yan, C.W.: Iconic indexing by 2-D strings. IEEE Trans. Pattern Anal. Mach. Intell. **3**, 413–428 (1987)
12. Chang, S.K., Yan, C., Dimitroff, D.C., Arndt, T.: An intelligent image database system. IEEE Trans. Softw. Eng. **14**(5), 681–688 (1988)
13. Cook, S.A.: The complexity of theorem-proving procedures. In: Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, pp. 151–158. ACM (1971)
14. Crescenzi, P., Panconesi, A.: Completeness in approximation classes. Inf. Comput. **93**(2), 241–262 (1991)
15. Galbiati, G., Maffioli, F., Morzenti, A.: A short note on the approximability of the maximum leaves spanning tree problem. Inf. Process. Lett. **52**(1), 45–49 (1994)
16. Guan, D., Chou, C.Y., Chen, C.W.: Computational complexity of similarity retrieval in a pictorial database. Inf. Process. Lett. **75**(3), 113–117 (2000)
17. Hirschberg, D.S.: A linear space algorithm for computing maximal common subsequences. Commun. ACM **18**(6), 341–343 (1975)
18. Huang, K.S., Yang, C.B., Tseng, K.T., Ann, H.Y., Peng, Y.H.: Efficient algorithms for finding interleaving relationship between sequences. Inf. Process. Lett. **105**(5), 188–193 (2008)
19. Huang, K.S., Yang, C.B., Tseng, K.T., Peng, Y.H., Ann, H.Y.: Dynamic programming algorithms for the mosaic longest common subsequence problem. Inf. Process. Lett. **102**(2–3), 99–103 (2007)
20. Hunt, J.W., Szymanski, T.G.: A fast algorithm for computing longest common subsequences. Commun. ACM **20**(5), 350–353 (1977)
21. Iliopoulos, C.S., Rahman, M.S.: Algorithms for computing variants of the longest common subsequence problem. Theor. Comput. Sci. **395**(2–3), 255–267 (2008)
22. Jansen, T.: Introduction to the theory of complexity and approximation algorithms. In: Mayr, E.W., Prömel, H.J., Steger, A. (eds.) Lectures on Proof Verification and Approximation Algorithms, pp. 5–28. Springer, Berlin (1998)
23. Kann, V.: Maximum bounded 3-dimensional matching is MAX SNP-complete. Inf. Process. Lett. **37**(1), 27–35 (1991)
24. Karp, R.M.: Reducibility among combinatorial problems. In: Proceedings of a Symposium on the Complexity of Computer Computations, pp. 85–103. IBM Thomas J. Watson Research Center, Yorktown Heights, New York (1972)
25. Knuth, D.E., Morris Jr., J.H., Pratt, V.R.: Fast pattern matching in strings. SIAM J. Comput. **6**(2), 323–350 (1977)

26. Krithivasan, K., Sitalakshmi, R.: Efficient two-dimensional pattern matching in the presence of errors. Inf. Sci. **43**(3), 169–184 (1987)
27. Lee, S.Y., Hsu, F.J.: 2D C-string: a new spatial knowledge representation for image database systems. Pattern Recognit. **23**(10), 1077–1087 (1990)
28. Lee, S.Y., Hsu, F.J.: Spatial reasoning and similarity retrieval of images using 2D C-string knowledge representation. Pattern Recognit. **25**(3), 305–318 (1992)
29. Lee, S.Y., Shan, M.K., Yang, W.P.: Similarity retrieval of iconic image database. Pattern Recognit. **22**(6), 675–682 (1989)
30. Papadimitriou, C.H., Yannakakis, M.: Optimization, approximation, and complexity classes. J. Comput. Syst. Sci. **43**(3), 425–440 (1991)
31. Pawlik, M., Augsten, N.: RTED: a robust algorithm for the tree edit distance. Proc. VLDB Endow. **5**(4), 334–345 (2011)
32. Peng, Y.H., Yang, C.B., Huang, K.S., Tseng, C.T., Hor, C.Y.: Efficient sparse dynamic programming for the merged lcs problem with block constraints. Int. J. Innov. Comput. Inf. Control **6**(4), 1935–1947 (2010)
33. Schaefer, T.J.: The complexity of satisfiability problems. In: Proceedings of the 10th Annual ACM Symposium on Theory of Computing, pp. 216–226. ACM (1978)
34. Tamura, H., Yokoya, N.: Image database systems: a survey. Pattern Recognit. **17**(1), 29–43 (1984)
35. Tanimoto, S.L.: An iconic/symbolic data structuring scheme. In: Pattern Recognition and Artificial Intelligence, pp. 452–471 (1976)