Kernelization and Approximation Algorithms for Finding a Perfect Phylogeny from Mixed Tumor Samples

Wen-Horng Sheu[†] and Biing-Feng Wang[‡]
[†]Department of Computer Science, University of California, Davis
Davis, CA 95616, USA
[‡]Department of Computer Science, National Tsing Hua University
Hsinchu, Taiwan 30013, Republic of China
wsheu@ucdavis.edu, bfwang@cs.nthu.edu.tw

Abstract.

The *split-row problem* (SR), introduced by Hajirasouliha and Raphael [WABI 2014], models an effective method for reconstructing a perfect phylogeny from mixed tumor samples. In this problem, an $m \times n$ binary matrix M is given. A *split-row operation* on M is defined as replacing a row r by $k > 1$ rows whose bitwise OR is equal to r. The cost of the operation is the number of additional rows induced, that is, $k - 1$. The objective is to find a sequence of operations that transforms M into a matrix corresponding to a perfect phylogeny and the total cost is minimized. Let $\varepsilon(M)$ denote the minimum total cost. In this paper, we show that SR admits a polynomial size kernel, which has at most $3\varepsilon(M)$ rows and $4\varepsilon(M) - 1$ columns. We also study the approximability of SR. Let $\alpha(M) = m + \varepsilon(M)$. Hujdurović *et al.* [TALG 2018] proposed an equivalent formulation of SR, in which M is represented by a digraph and the goal is to find an optimal spanning forest of the digraph. Based on this formulation, they gave a w-approximation algorithm and an h-approximation algorithm for $\alpha(M)$, where w and h are, respectively, the width and height of the derived digraph; and they left the approximability of $\varepsilon(M)$ as a direction for future work. In this paper, we present three new approximation algorithms for $\alpha(M)$, which have approximation factors $0.5w$, $0.5h + 0.5$, and $0.368h + 0.633$, respectively. In addition, we present first approximation algorithms for $\varepsilon(M)$. More specifically, we show that our approximation algorithms for $\alpha(M)$ are also approximation algorithms for $\varepsilon(M)$, which have approximation factors $w - 1$, h, and $0.736h + 0.266$, respectively.

Key words. algorithms, fixed-parameter tractability, kernelization, approximation algorithms, split-row problem, perfect phylogenies

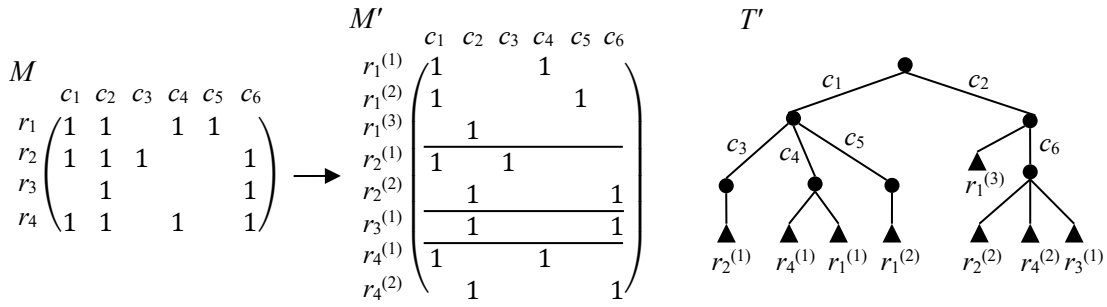
AMS subject classifications. 05C05, 68Q27, 68W25, 68W40, 92D15

Abbreviated title. Kernelization and Approximation Algorithms for SR

1. Introduction

A *perfect phylogeny* is a rooted unordered tree representing the evolutionary relationships of m objects in terms of n *characters*. The objects correspond bijectively to the leaves of the tree, and each character labels exactly one edge of the tree. An object r has a character c if and only if the edge with label c is on the path from r to the root. Figure 1(b) shows a perfect phylogeny T', where the objects are $r_1^{(1)}$, $r_1^{(2)}$, $r_1^{(3)}$, $r_2^{(1)}$, $r_2^{(2)}$, $r_3^{(1)}$, $r_4^{(1)}$, $r_4^{(2)}$, the characters are c_1, c_2, ..., c_6, and the leaf $r_1^{(2)}$ has characters c_1 and c_5. A *matrix representation* of a perfect phylogeny is an $m \times n$ binary matrix in which each row is associated with an object, each column is associated with a character, and the entry at row r and column c is 1 if and only if r has character c. Figure 1(a) gives a matrix representation M' of the perfect phylogeny in Figure 1(b). Although each perfect phylogeny naturally corresponds to a binary matrix, the opposite does not necessarily hold. The *perfect phylogeny problem* asks if a given binary matrix M represents a perfect phylogeny. A binary matrix M is *conflict-free* if for every pair of columns c, c', there do not exist three rows r, r', r'' such that (1) $M_{r,c} = 1$ and $M_{r,c'} = 1$, (2) $M_{r',c} = 0$ and $M_{r',c'} = 1$, and (3) $M_{r'',c} = 1$ *and* $M_{r'',c'} = 0$, where $M_{r,c}$ denotes the entry of M at row r and column c. By the Perfect Phylogeny Theorem [12, 13], a binary matrix exhibits a perfect phylogeny if and only if it is conflict-free. The perfect phylogeny problem and its generalizations have been extensively studied in the past decades [7, 8, 12, 21, 25]. In this paper, we consider a generalization introduced by Hajirasouliha and

Raphael [15], called the *split-row problem* (SR). The input is an $m \times n$ binary matrix M. A *split-row operation* on M is defined as replacing a row r of M by $k > 1$ rows $r^{(1)}, r^{(2)}, ..., r^{(k)}$ whose bitwise OR is equal to r. The cost of the operation is the number of additional rows induced, that is, $k - 1$. The goal is to find a sequence of split-row operations that transforms M into a conflict-free matrix and the total cost is minimized. Consider the example in Figure 1(a). The matrix M' can be obtained by performing three split-row operations on the matrix M: one replaces r_1 by $r_1^{(1)}$, $r_1^{(2)}$ and $r_1^{(3)}$, another replaces r_2 by $r_2^{(1)}$ and $r_2^{(2)}$, and the other replaces r_4 by $r_4^{(1)}$ and $r_4^{(2)}$. The total cost is $2 + 1 + 1 = 4$.



(a) Transforming a binary matrix M into a conflict-free matrix M' by split-row operations

(b) A perfect phylogeny T' corresponding to M'

Figure 1. An illustrative example, in which zeros in a binary matrix are omitted.

The study of SR was motivated by an application in inferring phylogenetic trees from tumor samples [15]. Tumor evolution is assumed to exhibit a perfect phylogeny, in which each object is a tumor cell subpopulation and each character represents a mutation [17]. This phylogenetic tree can offer a better understanding of the processes associated with tumor growth [2, 24] and will contribute to the development of novel treatments [23]. Recent DNA sequencing technologies have enabled the identification of the mutations involved in a tumor cell subpopulation. Assume that each measured mutation in a tumor sample has two possible states: 0 = normal and 1 = mutated. Then, the sequencing data can be represented as a binary matrix, in which each row corresponds to a tumor sample, each column is associated with a mutation, and the entry at row r and column c indicates whether r carries the mutation c. In an ideal condition, the matrix should exhibit a perfect phylogeny. However, in the data used by most cancer sequencing studies, the tumor samples often consist of several genetically distinct cell subpopulations. Given such data, solving SR is to minimally decompose the samples and thereby reconstruct a perfect phylogeny. Several other models have been introduced for inferring phylogenetic trees from tumor samples [6, 14, 20, 22, 26, 28]. An excellent survey can be found in [19]. In [18], SR was tested against four popular models. Experiments showed that the perfect phylogenies reconstructed by solving SR were generally more faithful on real data and were mostly more accurate on simulated data. We refer the reader to [15, 18] for more information about the biological aspect of SR.

Hujdurović *et al.* [17] showed that SR is NP-hard and gave an efficient heuristic algorithm. Hujdurović *et al*. [16] further proved the APX-hardness of SR. A naive algorithm for SR is to examine all possible ways to split each row into a set of distinct rows, which requires $2^{\Omega(mn)}$ time in the worst-case [16]. Hujdurović *et al*. [16] proved that SR can be equivalently reformulated as a graph problem, named the *uncovering branching problem* (UB). In this problem, a set family Φ is given, and we consider its *containment digraph*, in which each vertex represents a set of Φ and there is an arc from a vertex v to a vertex v' if $v \subset v'$. A *branching* is defined to be a subset of arcs which induces a directed spanning forest of the digraph. Each branching is associated with a cost. The goal is to find a branching of the minimum cost. A set family Φ over a ground set Σ can be represented by a $|\Sigma| \times |\Phi|$ binary matrix, called a *matrix representation* of Φ, as follows: each row is associated with an element of Σ, each column is associated with a set of Φ, and the entry at row e and column v is 1 if and only if the set v contains the element e. Similarly, given a binary matrix M, a set family represented by M can be constructed as follows: the ground set is $\Sigma = \{r_1, r_2, ..., r_m\}$, where r_i denote row i of M; and the set family is $\Phi = \{v_1, v_2, ..., v_n\}$, where $v_j = \{r_i \mid 1 \le i \le m, M_{i,j} = 1\}$. For example, for the matrix M in Figure 1, we have $v_1 = \{r_1, r_2, r_4\}$, $v_2 = \{r_1, r_2, r_3, r_4\}$, $v_3 = \{r_2\}$, $v_4 = \{r_1, r_4\}$, $v_5 = \{r_1\}$, and $v_6 = \{r_2, r_3, r_4\}$. For convenience, the containment digraph of a set family represented by a binary matrix M is simply called the

containment digraph of M. Hujdurović *et. al*. proved that SR is equivalent to UB. More specifically, they proved that solving SR on a binary matrix M is equivalent to solving UB on a set family represented by M. A *laminar* set family is a set family in which each pair of sets are either disjoint or related by containment. We remark that, as indicated in [27], UB is equivalent to the following interesting problem: Transform a given set family Φ to a laminar one by copying the elements of the ground set; it is required that each set in Φ receives at least one copy of each of its original elements; and we want to minimize the number of additional copies.

Hujdurović *et al*. [16] called UB the *branching formulation* of SR. Using this formulation, they showed that SR can be solved more efficiently in $O^*(n^n)$ time by examining all branchings of the containment digraph of M, where the O^* notation suppresses factors that are bounded by a polynomial in the input size. Husić *et al*. [18] formulated SR as an Integer Linear Program (ILP) and implemented it into a software package called MIPUP. Let $\varepsilon(M)$ denote the minimum cost of transforming a binary matrix M into a conflict-free matrix. Recently, Sheu and Wang [27] showed that SR is fixed-parameter tractable (FPT) by presenting an algorithm that runs in time $O(mn^2 + mn\ 2^{\min(n,\ 2\varepsilon(M))} + n^2\ 3^{\min(n,\ 2\varepsilon(M))})$ or $O(mn^5 \lg(mn) \lg\lg(mn)\ 2^{\min(n,\ 2\varepsilon(M))})$. In the worst case, their algorithm requires $O^*(2^n)$ time. In the application of perfect phylogeny reconstruction, $\varepsilon(M)$ measures the amount of mixing within the tumor samples. Therefore, Sheu and Wang's result indicates that SR can be solved efficiently when the amount of mixing is small. Both the results in [18, 27] were based on the branching formulation of SR. Husić *et al*.'s ILP formulation has $O(mn + n^2)$ variables and $O(mn)$ constraints. Sheu and Wang [27] remarked that the size of this ILP can be reduced by the first two phases of their algorithm. A careful reading of [27] shows that the reduced ILP has $O(mn + n{\cdot}\min(n, 2\varepsilon(M)))$ variables and $O(mn)$ constraints. In [16], Hujdurović *et al*. also studied the approximability of SR. Let $\alpha(M)$ be the minimum number of rows of a conflict-free matrix that can be obtained by performing split-row operations on the matrix M. Any sequence of split-row operations that transforms M into a matrix of m' rows costs $m' - m$. Hence, $\alpha(M) = \varepsilon(M) + m$ and an exact algorithm of SR finds a sequence of split-row operations that transforms M into a conflict-free matrix of $\alpha(M)$ rows. An algorithm that finds a sequence of split-row operations that transforms M into a conflict-free matrix M' is a ρ-*approximation algorithm for* $\varepsilon(M)$ if M' has at most $\rho\varepsilon(M) + m$ rows, and is a ρ-*approximation algorithm for* $\alpha(M)$ if M' has at most $\rho\alpha(M)$ rows. We call ρ the *approximation factor* of a ρ-approximation algorithm. Let w and h be, respectively, the width and height of the containment digraph of M. Using the branching formulation of SR, Hujdurović *et al.* [16] gave a w-approximation algorithm and an h-approximation algorithm for $\alpha(M)$. Their result indicates that $\alpha(M)$ can be approximated to within a constant factor if the derived containment digraph has bounded width or height. The approximability of $\varepsilon(M)$ was left as a direction for future work in [16].

This paper studies the kernelization of SR. A *kernelization algorithm* for a parameterized problem is a polynomial-time preprocessing algorithm transforming an input instance I to an equivalent instance I', called the *kernel*, such that the size of I' is bounded by a function of the parameter. It is known that a parameterized problem is FPT if and only if it admits a kernelization algorithm [4]. By applying this result, the FPT algorithm in [27] can be used to obtain a kernel of SR. However, the matrix size of the kernel is $O(2^{2\varepsilon(M)})$, which is exponential in $\varepsilon(M)$. In this paper, we present the first polynomial size kernel, which has at most $3\varepsilon(M)$ rows and $4\varepsilon(M) - 1$ columns. Our kernelization algorithm requires $O(\max(m^{0.373}n^2, mn^{1.373}))$ time. When $\varepsilon(M)$ is small, it can be used as a preprocessing procedure to speed up all previous exact algorithms for SR. Table 1 summarizes the results.

Our polynomial kernel is designed based on the branching formulation in [16]. More specifically, we give a kernelization algorithm for UB. An overview is as follows. Recall that each vertex of the containment digraph of Φ represents a set of Φ and the goal is to find an optimal branching of the containment digraph. Let $\delta(\Phi)$ denote the cost of an optimal branching. We find that two kinds of vertices, called *friendly vertices* and *trailing vertices*, can be removed without changing $\delta(\Phi)$. In addition, we find that some elements, called *doubly-chain-like elements*, can be deleted from each vertex without changing $\delta(\Phi)$. Based on the findings, our kernelization algorithm reduces the size of Φ by exhaustively removing friendly vertices, trailing vertices, and doubly-chain-like elements. To estimate the kernel size, we prove that if a set family Φ has no friendly vertices, trailing vertices, and doubly-chain-like elements, then it contains at most $4\delta(\Phi) - 1$ sets and its ground set contains at most $3\delta(\Phi)$ elements. By applying this result, a kernel of SR is obtained.

This paper also studies the approximability of SR. Our approximation algorithms are also designed based on the branching formulation. An overview is as follows. We find that the cost of a branching can be divided

into two parts, named the *chain-like part* and the *non-chain-like part*; and we find that for any given branching, by a linear-time conversion to another branching, we can always minimize the cost of the chain-like part, without increasing the cost of non-chain-like part. According to the findings, we propose a two-phase approach for designing approximation algorithms for UB. Based on the approach, we present for $\alpha(M)$ a $0.5w$-approximation algorithm, a $(0.5h + 0.5)$-approximation algorithm, and a randomized $(0.368h + 0.633)$-approximation algorithm. Then, we further show that a ρ-approximate algorithm for $\alpha(M)$ is a $(2\rho - 1)$-approximation algorithm for $\varepsilon(M)$ if it is designed based on the two-phase approach. Accordingly, our three approximation algorithms for $\alpha(M)$ are also approximation algorithms for $\varepsilon(M)$, which have approximation factors $w - 1$, h, and $0.736h + 0.266$, respectively. Table 2 summarizes approximation algorithms for $\alpha(M)$ and $\varepsilon(M)$.

Previous results	New results
$O^*(n^n)$ [16]	$O^*((4\varepsilon - 1)^{4\varepsilon})$
ILP; $O(mn + n^2)$ variables; $O(mn)$ constraints [18]	ILP; $O(\varepsilon^2)$ variables; $O(\varepsilon^2)$ constraints
$O(mn^2 + mn\ 2^{\min(n,\ 2\varepsilon)} + n^2\ 3^{\min(n,\ 2\varepsilon)})$ [27]	$O(mn^2 + \varepsilon^2\ 3^{2\varepsilon})$
$O(mn^5 \lg (mn) \operatorname{lglg}(mn)\ 2^{\min(n,\ 2\varepsilon)})$ [27]	$O(mn^2 + \varepsilon^6 \lg \varepsilon \operatorname{lglg} \varepsilon\ 2^{2\varepsilon})$

Table 1. Exact algorithms for SR, in which $\varepsilon(M)$ is simply denoted as ε.

Objective	Factor	Time	Remark
$\alpha(M)$	w	$O(mn^2) + \tilde{O}(n^{2.373})$	[16]
	$0.5w$	$O(mn^2) + \tilde{O}(n^{2.373})$	this
	h	$O(mn^2)$	[16]
	$0.5h + 0.5$	$O(mn^2)$	this
	$0.368h + 0.633$	$\tilde{O}(mn^{13})$	this; randomized
$\varepsilon(M)$	$w - 1$	$O(mn^2) + \tilde{O}(n^{2.373})$	this
	h	$O(mn^2)$	this
	$0.736h + 0.266$	$\tilde{O}(mn^{13})$	this; randomized

Table 2. Approximation algorithms for $\alpha(M)$ and $\varepsilon(M)$.

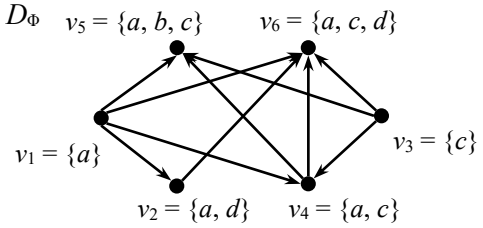
This paper is organized as follows. Section 2 reviews the branching formulation. Sections 3 and 4 present a kernelization algorithm. Section 3 gives the reduction rules and kernel size; and Section 4 shows the time complexity. Section 5 gives approximation algorithms. Section 6 concludes this paper with some possible directions for future research.

2. Review of the branching formulation

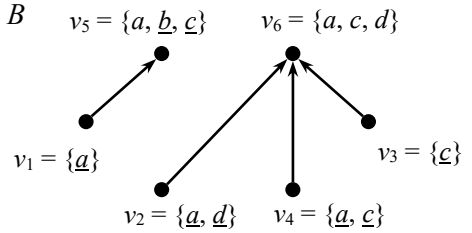
This section describes Hujdurović *et al.*'s branching formulation of SR. Let $M \in \{0, 1\}^{m \times n}$ be a binary matrix. The input matrix is assumed to contain no rows or columns whose entries are all zeros. Clearly, the removal of such rows or columns does not change $\varepsilon(M)$ [15]. We also assume that all columns in the input matrix are distinct. In case this is not true, the removal of duplicates also does not change $\varepsilon(M)$ [16].

Let Φ be a set family over a set Σ. Without loss of generality, we assume that Φ does not contain the empty set, all sets in Φ are distinct, and each element of Σ is contained in at least one set of Φ. The *containment digraph* D_Φ of Φ is the directed acyclic graph with vertex set Φ and arc set $\{(v, v') \mid v, v' \in \Phi, v \subset v'\}$. Figure 2(a) gives an example, in which $\Sigma = \{a, b, c, d\}$ and $\Phi = \{v_1, v_2, ..., v_6\}$, where $v_1 = \{a\}$, $v_2 = \{a, d\}$, $v_3 = \{c\}$, $v_4 = \{a, c\}$, $v_5 = \{a, b, c\}$, and $v_6 = \{a, c, d\}$. A *directed rooted tree* is a rooted tree with its edge directed toward the root. A *directed rooted forest* is a disjoint union of directed rooted trees. A *branching* of D_Φ is a subset B of arcs such that (Φ, B) is a digraph in which for each vertex v there is at most one arc leaving v. In other words, a branching B is a subset of arcs such that (Φ, B) is a directed rooted forest. See Figure 2(b)

for an example, in which $B = \{(v_1, v_5), (v_2, v_6), (v_3, v_6), (v_4, v_6)\}$.



(a) The containment digraph D_Φ



(b) A branching B, in which uncovered elements of each vertex are underlined

Figure 2. An illustrative example.

Consider a branching B of D_Φ. Let F_B denote the forest (Φ, B). We say that a vertex p is the *B-parent* of a vertex v if p is the parent of v in F_B. The term B-child is defined similarly. The B-parent of a vertex v is denoted by $p_B(v)$, which may be null. For an element $e \in \Sigma$ and a vertex $v \in \Phi$, we say that (e, v) is a *target pair* if $e \in v$. Consider a target pair (e, v). We say that e is *B-covered* in v if v has a B-child u such that $e \in u$; otherwise, we say that e is *B-uncovered* in v. A B-uncovered pair is a target pair (e, v) such that e is B-uncovered in v. We denote by $U(B)$ the set of all B-uncovered pairs. For each element $e \in \Sigma$, we denote by $U_B(e)$ the set of all B-uncovered pairs (e, v), where $v \in \Phi$. For the example in Figure 2(b), $U_B(a) = \{(a, v_1), (a, v_2), (a, v_4)\}$, $U_B(b) = \{(b, v_5)\}$, $U_B(c) = \{(c, v_3), (c, v_4), (c, v_5)\}$, $U_B(d) = \{(d, v_2)\}$, and $U(B)$ is the union of $U_B(a)$, $U_B(b)$, $U_B(c)$, and $U_B(d)$. Observe that if a vertex v contains an element e, there is at least one descendant of v in which e is B-uncovered. Thus, $|U_B(e)| \geq 1$ for each element $e \in \Sigma$. The *cost* of a branching B, denoted by $cost(B)$, is defined as $\sum_{e \in \Sigma} (|U_B(e)| - 1) = |U(B)| - |\Sigma|$. We denote by $\beta(\Phi)$ the minimum number of B-uncovered pairs over all branchings B of D_Φ. Let $\delta(\Phi) = \beta(\Phi) - |\Sigma|$ be the minimum cost of a branching of D_Φ. Define an optimization problem, which is referred as the *branching formulation* of SR, as follows.

Definition 1. The Uncovering Branching Problem (UB):
Input: A set family Φ.
Task: Find a branching B of D_Φ with $cost(B) = \delta(\Phi)$.

Hujdurović *et. al*. proved that SR is equivalent to UB. An interested reader may refer to [16] for the proof. For convenience, a sequence of split-row operations is said to be *feasible* if it transforms M into a conflict-free matrix. The equivalence between SR and UB is formally stated as the following theorem.

Theorem 2.1. [16] Let M be an $m \times n$ binary matrix and Φ be a set family represented by M. Then, $\varepsilon(M) = \delta(\Phi)$. In addition, any branching B of D_Φ can be transformed in time $O(mn^2)$ to a feasible sequence of split-row operations on M with a total cost of $cost(B)$; and any feasible sequence of split-row operations on M with a total cost of $m' - m$ can be transformed in time $O(m'n^2 + n^{2.373})$ to a branching B of D_Φ with $cost(B) = m' - m$ (i.e., $|U(B)| = m'$).

Without loss of generality, in this paper, we assume that the input Φ has $\delta(\Phi) > 0$. When $\delta(\Phi) = 0$, UB can be solved in $O(mn)$ time as follows. Let Φ be a set family of n sets over a ground set of m elements and let M be its matrix representation. By Theorem 2.1 and the Perfect Phylogeny Theorem, $\delta(\Phi) = 0$ if and only if M is conflict-free. Gusfield [12] had an $O(mn)$-time algorithm that determines whether M is conflict-free and if so, builds a perfect phylogeny corresponding to M. Assume that M is conflict-free. Recall that each column of M is associated with a set of Φ. Since M is conflict-free, the set family Φ is laminar [12]. The first step of Gusfield's algorithm for building a perfect phylogeny is to construct a forest F as follows: create a node for each set $v \in \Phi$; and if no set in Φ contains v, set its parent as null, otherwise set its parent as the smallest set containing v. Note that the smallest set containing a set is unique, for otherwise Φ is not laminar. Let B be the edge set of F, which is a branching of D_Φ. Consider an element e. Since Φ is laminar, all the sets containing e are pairwise nested and hence form a path in F. As a result, $|U_B(e)| = 1$ can be concluded. Thus, B is a branching of D_Φ with zero cost (or with $|U(B)| = m$). Therefore, using Gusfield's algorithm, in $O(mn)$ time, we can determine whether $\delta(\Phi) = 0$ and if so, construct an optimal branching of D_Φ.

Remark 2.2. We remark that the $O(m'n^2 + n^{2.373})$ time in Theorem 2.1 can be reduced to $O(m'n)$. Given a feasible sequence of split-row operations on M, Hujdurović *et. al*. [16] constructed a corresponding branching B in three steps as follows. Step 1 obtains a conflict-free matrix M' by performing the given sequence of operations. Let Φ' be a set family represented by M'. Step 2 finds all *elementary* arcs of $D_{\Phi'}$, where an arc (u, v) is elementary if there exists no vertex w such that (u, w) and (w, v) are both arcs of $D_{\Phi'}$. For each set v in Φ', there is a unique set, denoted by v', in Φ such that the position of the column associated with v' in M is the same as the position of the column associated with v in M'. Step 3 constructs the branching B as $\{(u', v') \mid (u, v)$ is an elementary arc of $D_{\Phi'}\}$. Steps 1 and 3 require $O(m'n)$ time. Step 2 is the bottleneck. Hujdurović *et. al*. implemented this step by constructing $D_{\Phi'}$ in $O(m'n^2)$ time and then finding the elementary arcs in $O(n^{2.373})$ time using an algorithm of Aho *et. al* [1]. Since M' is conflict-free, Φ' is laminar. As a result, it is easy to check that an arc (u, v) of $D_{\Phi'}$ is elementary if and only if v is the smallest set containing u. Consequently, Step 2 can be done in $O(m'n)$ time by simply running Gusfield's algorithm in [12] on M'.

3. Reduction rules and kernel size

This section presents the reduction rules and kernel size of our kernelization algorithm for UB. More specifically, we present three rules that can be used to reduce a set family while preserving the minimum cost; and we show that if a set family Φ is irreducible under the rules, it contains at most $4\delta(\Phi) - 1$ sets and its ground set contains at most $3\delta(\Phi)$ elements. Let Φ be the input family and Σ be the ground set of Φ. For ease of discussion, in this section, we assume that $\Sigma \in \Phi$. In case this is not true, we simply add Σ into Φ. It is easy to observe that this adding does not change $\delta(\Phi)$ [27]. The set Σ contains all the other sets in Φ. We call Σ the *root vertex* of D_Φ. Clearly, in the finding of an optimal branching, we may consider only inclusion-wise maximal branchings. In other words, we may consider only branchings B such that F_B is a tree rooted at Σ.

Section 3.1 presents the reduction rules. Section 3.2 gives the kernel size. Section 3.3 proves that the reduction rules preserve the minimum cost.

3.1 Reduction rules

Consider two vertices $u, v \in \Phi$. The relation between u and v is classified into three types: u and v are *disjoint* if $u \cap v = \emptyset$; u and v are *nested* if $u \subset v$ or $v \subset u$; and otherwise, u and v are *in conflict*.

Definition 2. A vertex $v \in \Phi$ is *friendly* if it is not in conflict with any other vertex.

For an element $e \in \Sigma$, let V_e denote the set of vertices containing e in Φ. An element $x \in \Sigma$ is *chain-like* if any two vertices of V_x are nested. The *size* of a vertex v, denoted by $|v|$, is the number of elements in v. Consider a chain-like element x. Let $(u_1, u_2, ..., u_{s-1}, u_s = \Sigma)$ be the sequence of vertices obtained by arranging the vertices of V_x in increasing order of their sizes. Since any two vertices of V_x are nested, we have $u_1 \subset u_2 \subset ... \subset u_s$ and thus U represents a chain (or a path), from u_1 to Σ, in D_Φ. We call u_1, the smallest vertex in V_x, the *tail vertex* of x, denoted by $t(x)$. Let $t^-(x) = t(x) - \{x\}$.

Definition 3. For a chain-like element x, if $t^-(x)$ is a vertex in Φ, we call $t^-(x)$ the *trailing vertex* of x.

Definition 4. An element $x \in \Sigma$ is *doubly chain-like* if there exists a chain-like element $y \neq x$ with $V_y \supseteq V_x$.

Note that that a doubly chain-like element is also chain-like. A *reduction rule* is a polynomial-time procedure that replaces a set family Φ by a new set family Φ'; and the rule is *safe* if $\delta(\Phi) = \delta(\Phi')$. Our three reduction rules are as follows.

Rule 1. If Φ contains a non-root friendly vertex v, remove v from Φ.

Rule 2. If Φ contains the trailing vertex, $t^-(x)$, of a chain-like element x, remove $t^-(x)$ from Φ.

Rule 3. If Σ contains a doubly chain-like element x with no trailing vertex (i.e., Φ does not contain $t^-(x)$), remove x by replacing Φ as $\{v - \{x\} \mid v \in \Phi\}$ and updating Σ as $\Sigma - \{x\}$.

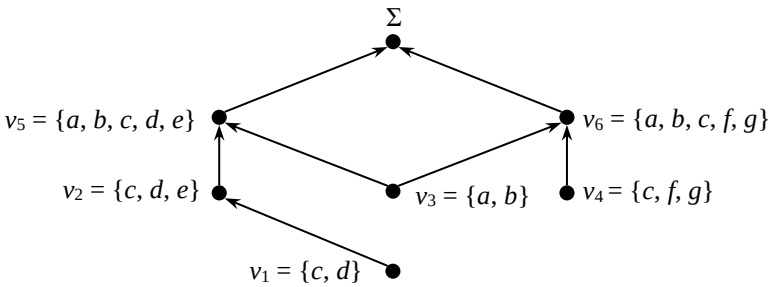


Figure 3. An illustrative example, in which non-elementary arcs are omitted.

For example, consider the containment digraph D_Φ of a set family Φ in Figure 3, in which $\Sigma = \{a, b, c, d, e, f, g\}$. In this figure, we display only arcs that are elementary. The set of elementary arcs possesses the following property [1]: for any two vertices u, v, there is an arc (u, v) if and only if there is a path consisting of only elementary arcs from u to v. For example, according to Figure 3, D_Φ has an arc (v_1, v_5), since there is a path consisting of only elementary arcs from v_1 to v_5. We remark that omitting non-elementary arcs is simply for clarity of illustration and a branching may contain non-elementary arcs. In Figure 3, the vertex v_3 and Σ are friendly. All the other vertices are non-friendly: v_5 is in conflict with v_6; and both v_1 and v_2 are in conflict with v_4. The vertex v_3 can be removed by applying Rule 1. The elements d, e, f, g are chain-like elements with $V_d = \{v_1, v_2, v_5, \Sigma\}$, $V_e = \{v_2, v_5, \Sigma\}$, and $V_f = V_g = \{v_4, v_6, \Sigma\}$. Since $V_d \supset V_e$ and $V_f = V_g$, the elements e, f, and g are doubly chain-like. In this example, since $t^-(e) = v_2 - \{e\} = \{c, d\}$ $(= v_1)$ is a vertex in Φ, Rule 2 can be applied to remove v_1; and since none of $t^-(f) = v_4 - \{f\} = \{c, g\}$ and $t^-(g) = v_4 - \{g\} = \{c, f\}$ is a vertex in Φ, Rule 3 can be applied to remove f or g. Note that, in this example, after applying Rule 2 to remove v_1, e is still doubly chain-like and d becomes doubly chain-like; and thus we can further apply Rule 3 to remove d or e.

Lemma 3.1. Rules 1, 2, 3 preserve the following properties: (1) Σ is a vertex in Φ; (2) Φ does not contain an empty set; (3) no two vertices in Φ contain the same set of elements.

Proof. Recall that we have assumed that the input Φ satisfies (1), (2), and (3). Since Rules 1, 2, and 3 do not remove the root vertex, (1) is preserved. Since Rules 1 and 2 do not modify the elements contained in each vertex, they preserve (2) and (3). In the following, we complete the proof by showing that Rule 3 also preserves (2) and (3). Suppose that Rule 3 is applied to remove a doubly chain-like element x. Let y be a chain-like element with $V_y \supseteq V_x$. To prove the preservation of (2), it suffices to show that the singleton $\{x\}$ is not in Φ, which is true because any vertex containing x also contains y. To prove the preservation of (3), we show that Φ does not contain a pair of vertices u and v such that $u = v \cup \{x\}$. Suppose, by contradiction, that such a pair of vertices exists. Then, we have $u \in V_x$. Since $V_y \supseteq V_x$, we have $y \in u$ and thus $v \in V_y - V_x$. By the definition of doubly chain-like elements, each vertex in $V_y - V_x$ is a subset of every vertex in V_x. Since u and v differ only by one element, we know that u is the smallest vertex in V_x. That is, u is the tail vertex of x. As

a result, $t^{-}(x) = u - \{x\}$ $(= v)$ is a vertex in Φ, contradicting the condition of Rule 3. Therefore, the lemma holds. ■

Sheu and Wang [27] called a vertex of size one a *trivial vertex* and observed that the removal of trivial vertices does not change $\delta(\Phi)$. We remark that trivial vertices are friendly and their observation is a special case of Rule 1.

3.2 Kernel size

Let Φ' be a set family such that $\delta(\Phi') > 0$, the properties in Lemma 3.1 are satisfied, and none of Rules 1, 2, and 3 is applicable to Φ'. Let Σ' be the ground set of Φ'. This section shows that $|\Sigma'| \leq 3\delta(\Phi')$ and $|\Phi'| \leq 4\delta(\Phi') - 1$.

Lemma 3.2. For each vertex $v \in \Phi'$, we have $|v| \geq 2$ and v contains at least one non-chain-like element.
Proof. Clearly, Φ' contains more than one vertex, for otherwise, $\delta(\Phi') = 0$. Consider a non-root vertex v. Since Rule 1 is not applicable, v is in conflict with some vertex $u \in \Phi'$. Since u and v are in conflict, $v \cap u$ and $v - u$ are not empty. Thus, $|v| \geq 2$. Moreover, since u and v are in conflict, each element in $v \cap u$ is non-chain-like. Therefore, the lemma holds for any non-root vertex v. Since the root vertex Σ' contains all other vertices, the lemma holds for Σ' as well, which completes the proof. ■

Recall that for each element $e \in \Sigma$, $U_B(e)$ denotes the set of all B-uncovered pairs (e, v), where $v \in \Phi$. We need the following.

Lemma 3.3. [27] Let Φ be a set family over a ground set Σ. For each element $e \in \Sigma$, there exists a branching B of D_Φ with $|U_B(e)| = 1$ if and only if e is chain-like. Furthermore, there exists an optimal branching B such that $|U_B(e)| = 1$ for all chain-like elements $e \in \Sigma$.

Observe that for a chain-like element $x \in \Sigma'$, V_x contains exactly the set of vertices v in Φ' such that $t(x) \subseteq v$. From this observation, it is easy to conclude the following.

Lemma 3.4. Let $a, b \in \Sigma'$ be two chain-like elements. Then, $V_a = V_b$ if and only if $t(a) = t(b)$; and $V_a \supset V_b$ if and only if $t(a) \subset t(b)$.

Define an *antichain* as a set of pairwise non-nested vertices. Since a doubly chain-like element is also chain-like, it is easy to see that Σ' does not contain any doubly chain-like element; otherwise, either Rule 2 or Rule 3 can be applied. Let A be the set containing the tail vertices of all chain-like elements in Σ'. Since Σ' contains no doubly chain-like elements, by Lemma 3.4, no two chain-like elements have the same tail vertex; and the tail vertices of any two chain-like elements are non-nested. Therefore, A is an antichain. In addition, we know that $|A|$ equals the number of chain-like elements in Σ'.

Lemma 3.5. Let B be a branching of $D_{\Phi'}$ and let $e \in \Sigma'$ be an element. If there exists an antichain X such that e is contained in every vertex of X, then $|U_B(e)| \geq |X|$.
Proof. Let X be an antichain such that e is contained in every vertex of X. Consider the forest F_B. Since the vertices of X are pairwise non-nested, no two vertices in X have ancestor-descendant relationship in F_B. Therefore, the subtrees rooted at the vertices in X are pairwise disjoint. For each $v \in X$, since $e \in v$, the subtree rooted at v contains at least one vertex in which e is B-uncovered. Therefore, $U_B(e)$ contains at least $|X|$ uncovered pairs, which completes the proof. ■

The following two theorems give upper bounds on the sizes of Σ' and Φ'.

Theorem 3.6. Let Φ' be a set family such that $\delta(\Phi') > 0$ and the properties in Lemma 3.1 are satisfied. Let Σ' be the ground set of Φ'. If none of Rules 1, 2, and 3 is applicable to Φ', we have $|\Sigma'| \leq 3\delta(\Phi')$.
Proof. Let p and q be, respectively, the numbers of non-chain-like and chain-like elements in Σ'. We prove this theorem by showing that $p \leq \delta(\Phi')$ and $q \leq 2\delta(\Phi')$. Let B be an optimal branching of $D_{\Phi'}$. The cost of B

is $\delta(\Phi') = \sum_{e \in \Sigma'} (|U_B(e)| - 1)$. By Lemma 3.3, each non-chain-like element $e \in \Sigma'$ contributes $|U_B(e)| - 1 \geq 1$ to $\delta(\Phi')$. Therefore, p, the number of non-chain-like elements, is at most $\delta(\Phi')$.

Next, we prove that $q \leq 2\delta(\Phi')$. Recall that the set A, which contains the tail vertices of all chain-like elements, is an antichain. Let Σ_N be the set of non-chain-like elements in Σ'. Consider an element $e \in \Sigma_N$. Let A_e be the set of vertices in A that contain e. Since A_e is an antichain, by Lemma 3.5, $|U_B(e)| \geq |A_e|$. Since e is non-chain-like, by Lemma 3.3, $|U_B(e)| \geq 2$. By Lemma 3.2, each vertex in A contains at least one non-chain-like element. Thus, $\sum_{e \in \Sigma_N} |A_e| \geq |A|$. Consequently, we have

$$
\begin{aligned}
& q = |A| && \text{(since } |A| \text{ equals the number of chain-like elements in } \Sigma') \\
\leq\ & \textstyle\sum_{e \in \Sigma_N} |A_e| \\
\leq\ & \textstyle\sum_{e \in \Sigma_N} |U_B(e)| && \text{(since } |U_B(e)| \geq |A_e| \text{ for each } e \in \Sigma_N) \\
\leq\ & \textstyle\sum_{e \in \Sigma_N} 2(|U_B(e)| - 1) && \text{(since } |U_B(e)| \geq 2 \text{ for each } e \in \Sigma_N) \\
\leq\ & \textstyle\sum_{e \in \Sigma'} 2(|U_B(e)| - 1) && \text{(since } |U_B(e)| \geq 1 \text{ for each } e \in \Sigma' - \Sigma_N) \\
=\ & 2\delta(\Phi'),
\end{aligned}
$$

which completes the proof. ■

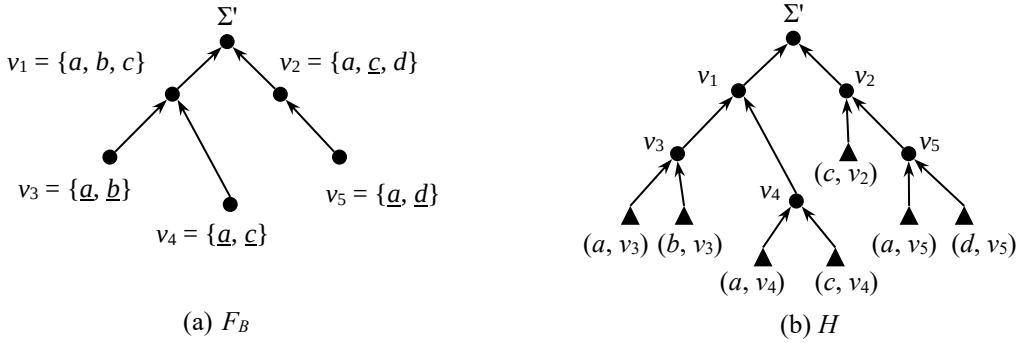


Figure 4. An illustration for the proof of Theorem 3.7, in which uncovered elements of each vertex are underlined and triangles are vertices created for the uncovered pairs.

Theorem 3.7. Let Φ' be a set family such that $\delta(\Phi') > 0$ and the properties in Lemma 3.1 are satisfied. If none of Rules 1, 2, and 3 is applicable to Φ', we have $|\Phi'| \leq 4\delta(\Phi') - 1$.

Proof. Let B be an optimal branching of $D_{\Phi'}$. By Theorem 3.6, $|U(B)| = \beta(\Phi') = \delta(\Phi') + |\Sigma'| \leq 4\delta(\Phi')$. In the following, we prove this theorem by showing that $|\Phi'| \leq |U(B)| - 1$. Let H be the rooted tree obtained from F_B by doing the following: for each uncovered pair $(e, v) \in U(B)$, create a vertex labeled with (e, v) and attach it as a leaf child of v. See Figure 4 for an example. Note that the leaf set and internal node set of H are, respectively, $U(B)$ and Φ'. We claim that every internal node of H has at least two children. We prove the claim first. Consider an internal node v of H. Three cases are considered: v has more than one B-child, v has exactly one B-child, and v has no B-child. By definition, the child set of v in H includes its B-children and the leaves created for its elements that are B-uncovered in v. If v has more than one B-child, the claim holds trivially. Assume that v has exactly one B-child, say u. Since Φ' satisfies the properties in Lemma 3.1, no two vertices contain the same set of elements. Thus, $v - u$ is nonempty. That is, at least one element of v is B-uncovered in v. Therefore, v has at least two children in H. Consider the last case, in which v has no B-child. All elements of v are B-uncovered in v. By Lemma 3.2, the size of v is at least two and thus v has at least two children in H, which completes the proof of the claim.

We proceed to prove the theorem. It is known that a rooted tree of l leaves contains at most $l - 1$ internal nodes that have more than one child, where $l \geq 1$ is an integer [3]. Since H contains $|U(B)|$ leaves, $|\Phi'|$, the number of internal nodes of H, is at most $|U(B)| - 1$. Consequently, the theorem holds. ■

We remark that the upper bounds in Theorems 3.6 and 3.7 are both tight. For any d of a power of 2, we can construct a set family Φ_d over a ground set Σ_d such that $|\Phi_d| = 4d - 1$, $|\Sigma_d| = 3d$, $\delta(\Phi_d) = d$, and none of

the three reduction rules is applicable. This construction is given in Appendix A.

3.3 Safeness

3.3.1 Rules 1 and 2

This section proves that removing vertices by Rules 1 and 2 does not change $\delta(\Phi)$. Recall that $\beta(\Phi) = \delta(\Phi) + |\Sigma|$ is the number of uncovered pairs in an optimal branching of D_Φ. Since only non-root vertices are removed by Rules 1 and 2, reductions according to Rules 1 and 2 do not change the ground set Σ. Therefore, proving the safeness of Rules 1 and 2 can be done by showing that removing vertices by Rules 1 and 2 does not change $\beta(\Phi)$.

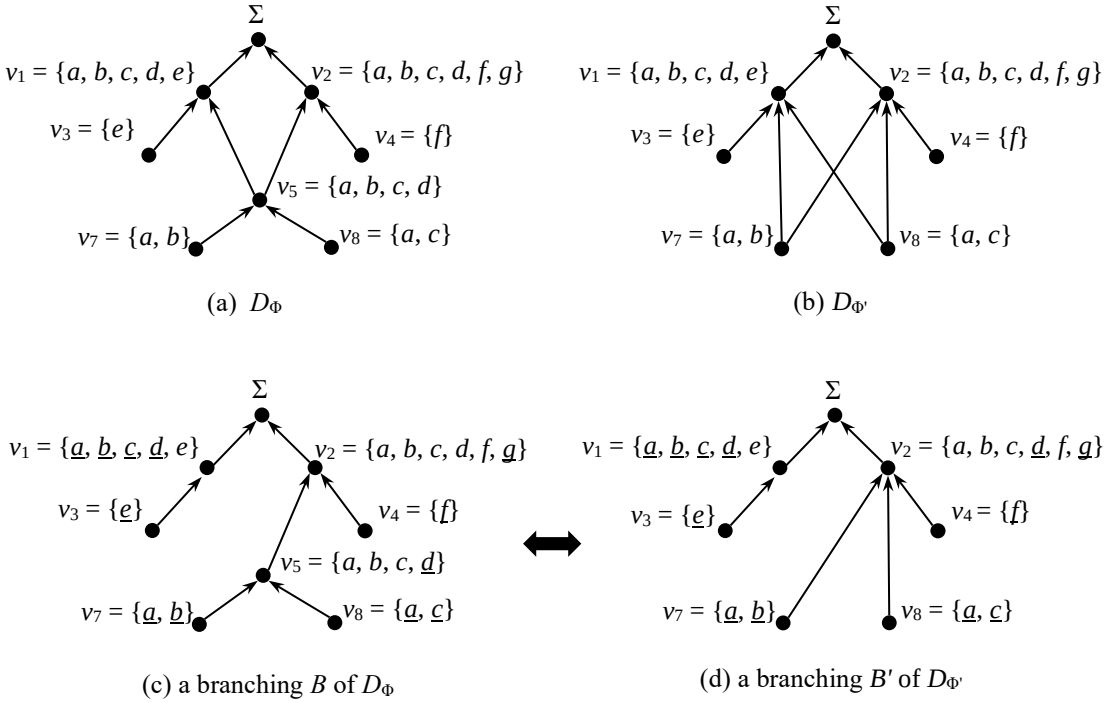


Figure 5. An illustrative example, in which v_5 is removed from Φ.

For convenience, for two vertices $u, v \in \Phi$, we call u an *in-neighbor* of v if $u \subset v$, and we call u an out-neighbor of v if $v \subset u$. An out-neighbor z of a vertex $v \in \Phi$ is called an *ideal parent* of v if the following condition holds: for each in-neighbor $u \neq v$ of z, either $u \subset v$ or $u \cap v = \emptyset$. For example, in Figure 5(a), v_3 has an ideal parent v_1; v_4 has an ideal parent v_2; v_5 has two ideal parents v_1 and v_2; and the other vertices do not have ideal parents. The following lemma says that removing from Φ a vertex that has an ideal parent does not decrease $\beta(\Phi)$.

Lemma 3.8. Let $v \in \Phi$ be a vertex that has an ideal parent and $\Phi' = \Phi - \{v\}$. Then, $\beta(\Phi) \leq \beta(\Phi')$.
Proof. We prove this lemma by showing that for any branching B' of $D_{\Phi'}$, we can find a branching B of D_Φ with $|U(B')| = |U(B)|$. Consider a fixed branching B' of $D_{\Phi'}$. Let z be an ideal parent of v in Φ. We obtain from B' a branching B of D_Φ as follows:

1. add the vertex v and assign z to be its parent; and
2. for each B'-child u of z with $u \subset v$, reassign v to be its parent.

See Figure 5(c)(d) for an example, in which $v = v_5$ and $z = v_2$ is an ideal parent of v_5. For each vertex $u \in \Phi$, let $U_B{}^u$ be the set of B-uncovered elements in u. Define $U_{B'}{}^u$ similarly. For each vertex $u \notin \{v, z\}$, since its child sets in B' and B are the same, we have $|U_B{}^u| = |U_{B'}{}^u|$. Since $|U(B)| = \sum_{u \in \Phi} |U_B{}^u|$ and $|U(B')| = \sum_{u \in \Phi,\, u \neq v}$

$|U_{B'}{}^u|$, to prove $|U(B)| = |U(B')|$, it suffices to show that $|U_B{}^z| + |U_B{}^v| = |U_{B'}{}^z|$. Let H be the set of B'-children of z. By the definition of ideal parents, H can be partitioned into two subsets: $H_1 = \{c \mid c \in H, c \cap v = \emptyset\}$ and $H_2 = \{c \mid c \in H, c \subset v\}$. Since every vertex in H_1 is disjoint with all vertices in H_2, we have $|U_{B'}{}^z| = |z| - |\bigcup_{c \in H_1 \cup H_2} c| = |z| - |\bigcup_{c \in H_1} c| - |\bigcup_{c \in H_2} c|$. We proceed to determine the sizes of $U_B{}^z$ and $U_B{}^v$. The child set of z in B is $\{v\} \cup H_1$. Since v is disjoint with all vertices in H_1, we have $|U_B{}^z| = |z| - |\bigcup_{c \in H_1} c| - |v|$. The child set of v in B is H_2 and thus we have $|U_B{}^v| = |v| - |\bigcup_{c \in H_2} c|$. As a result, we obtain $|U_B{}^z| + |U_B{}^v| = |z| - |\bigcup_{c \in H_1} c| - |\bigcup_{c \in H_2} c| = |U_{B'}{}^z|$, which completes the proof of this lemma. ■

The following lemma says that removing any vertex from Φ does not increasing $\beta(\Phi)$.

Lemma 3.9. Let $v \in \Phi$ be a vertex and $\Phi' = \Phi - \{v\}$. Then, $\beta(\Phi) \geq \beta(\Phi')$
Proof. We prove this lemma by showing that for any branching B of D_Φ, we can find a branching B' of $D_{\Phi'}$ with $|U(B')| \leq |U(B)|$. Consider a fixed branching B of D_Φ. Let z be the B-parent of v, which may be null. We obtain from B a branching B' of $D_{\Phi'}$ as follows:

1. for each B-child c of v, reassign z to be its parent; and
2. remove the arc (v, z) from B.

See Figure 5(c)(d) for an example, in which $v = v_5$ and $z = v_2$. Define $U_B{}^u$ and $U_{B'}{}^u$ as in the proof of Lemma 3.8. If z is null, it is easy to see that $|U(B')| = |U(B)| - U_B{}^v$, from which we have $|U(B')| \leq |U(B)|$ and thus the lemma holds. Assume that z is not null. Clearly, to prove $|U(B')| \leq |U(B)|$, we only need to show that $|U_{B'}{}^z| \leq |U_B{}^z| + |U_B{}^v|$. Let C be the set of B-children of v, and H be the set of B-children of z except v. Then, we have $U_B{}^v = v - \bigcup_{c \in C} c$, $U_B{}^z = z - v - \bigcup_{c \in H} c$, and $U_{B'}{}^z = z - \bigcup_{c \in C \cup H} c$. Observe that $U_{B'}{}^z$ can be written as the union of two disjoint parts $(z - v) - \bigcup_{c \in C \cup H} c$ and $v - \bigcup_{c \in C \cup H} c$. Since these two parts are, respectively, a subset of $U_B{}^z$ and a subset of $U_B{}^v$, we know that $|U_{B'}{}^z| \leq |U_B{}^z| + |U_B{}^v|$, which completes the proof. ■

We are ready to prove the safeness of Rules 1 and 2.

Theorem 3.10. Rules 1 and 2 are safe.
Proof. By Lemmas 3.8 and 3.9, removing from Φ a vertex that has an ideal parent does not change $\beta(\Phi)$. Thus, to prove this theorem, it suffices to show that each vertex removed by Rules 1 and 2 has an ideal parent. Suppose that Rule 1 is applied to remove a vertex v. Let z be the vertex in Φ such that $z \supset v$ and $|z|$ is minimum (with ties broken arbitrarily). Consider an in-neighbor $u \neq v$ of z. Since v is friendly, u is not in conflict with v. Since z is the smallest among all vertices strictly containing v, we have $v \not\subset u$. As a result, either $u \subset v$ or $u \cap v = \emptyset$. Therefore, z is an ideal parent of v.

Next, consider a vertex v that is removed by Rule 2. Let $x \in \Sigma$ be the chain-like element such that $v = t^-(x)$. By definition, the tail vertex of x is $t(x) = t^-(x) \cup \{x\} = v \cup \{x\}$. In the following, we show that $t(x)$ is an ideal parent of v. Consider an in-neighbor $u \neq v$ of $t(x)$. Since $t(x)$ is the smallest vertex containing x, we know that $x \notin u$. From $u \subset t(x)$ and $x \notin u$, we know that $u \subset t(x) - \{x\} = v$. As a result, we know that $t(x)$ is an ideal parent of v. Consequently, the theorem holds. ■

We remark that the proof of Lemma 3.8 also exhibits that given an ideal parent of v, we can obtain in $O(n)$ time an optimal branching of D_Φ from an optimal branching of $D_{\Phi'}$.

3.3.2 Rule 3

Let Φ' be the family obtained from Φ by applying Rule 3 to remove a doubly chain-like element x. The ground set of Φ' is $\Sigma' = \Sigma - \{x\}$. This section shows that $\delta(\Phi') = \delta(\Phi)$. Since x is doubly chain-like, there is a chain-like element, say y, such that $V_y \supseteq V_x$. For each vertex $v \in \Phi$, we use v' to denote $v - \{x\}$. The following lemma says that $D_{\Phi'}$ can be obtained from D_Φ by simply replacing each vertex v with v'.

Lemma 3.11. The arc set of $D_{\Phi'}$ is $\{(u', v') \mid (u, v)$ is an arc of $D_\Phi\}$.
Proof. Consider two vertex $u, v \in \Phi$. We prove this lemma by showing that the relation between u' and v' is the same as that of u and v. Four cases are considered: $u \subset v$, $u \supset v$, $u \cap v = \emptyset$, and u and v are in conflict. Consider the case $u \subset v$. Clearly, $u' \subseteq v'$. By Lemma 3.1, $u' \neq v'$ and hence we have $u' \subset v'$. Similarly, if $u \supset$

v, we have $u' \supset v'$. If $u \cap v = \emptyset$, it is easy to see that $u' \cap v'$ is also empty. Now, consider the last case, in which u and v are in conflict. Clearly, if none of u and v contains x, u' and v' are still in conflict. Assume that x is contained in u or v. Since u and v are in conflict, they cannot both contain x, for otherwise x is not chain-like. By symmetry, assume that u contains x (and v does not). Since $x \notin v$, we have $u' \cap v' = u \cap v$ and $v' - u' = v - u$, which are not empty. Since $V_y \supseteq V_x$, u also contains y. Since u and v are in conflict and y is chain-like, v cannot contain y. Hence, $u' - v'$ contains y and thus is not empty. Consequently, we conclude that u' and v' are in conflict, which completes the proof. ■

An immediate consequence of Lemma 3.11 is as follows.

Corollary 3.12. An element $e \in \Sigma'$ is a chain-like element of Φ' if and only if it is a chain-like element of Φ.

By Corollary 3.12, y is a chain-like element of Φ'. According to Lemma 3.11, there is a one-to-one correspondence between the branchings of D_Φ and the branchings of $D_{\Phi'}$. More specifically, each branching B of D_Φ corresponds to the branching $B' = \{(u', v') \mid (u, v) \in B\}$ of $D_{\Phi'}$. We have the following.

Lemma 3.13. Let B be a branching of D_Φ and B' be the branching corresponding to B in $D_{\Phi'}$. We have $|U(B')| = |U(B)| - |U_B(x)|$.
Proof. Since B' can be obtained from B by replacing each vertex v with v', it is easy to see that for each $e \in \Sigma'$, we have $|U_B(e)| = |U_{B'}(e)|$. Thus, $|U(B')| = \sum_{e \in \Sigma'} |U_{B'}(e)| = \sum_{e \in \Sigma} |U_B(e)| - |U_B(x)| = |U(B)| - |U_B(x)|$. Therefore, the lemma holds. ■

Lemma 3.14. [27] Let e be a chain-like element of Φ and $(u_1, u_2, ..., u_s = \Sigma)$ be the sequence obtained by arranging the vertices of V_x in increasing order of their sizes. For each branching B of D_Φ, $|U_B(e)| = 1$ if and only if $p_B(u_i) = u_{i+1}$ for $i = 1, 2, ..., s - 1$.

Lemma 3.15. Let B be a branching of D_Φ and B' be the branching corresponding to B in $D_{\Phi'}$. If $|U_{B'}(y)| = 1$, we have $|U(B')| = |U(B)| - 1$.
Proof. Let $(u_1, u_2, ..., u_{s-1}, u_s = \Sigma)$ be the sequence obtained by arranging the vertices of V_y in increasing order of their sizes. Recall that y is a chain-like element of Φ'. Since $|U_{B'}(y)| = 1$, by Lemma 3.14, we know that $p_{B'}(u'_i) = u'_{i+1}$ for $i = 1, 2, ..., s - 1$, where u'_i denotes $u_i - \{x\}$. Since $V_y \supseteq V_x$, there exists an index q such that $(u_q, u_{q+1}, ..., u_{s-1}, u_s)$ is the sequence obtained by arranging the vertices of V_x in increasing order of their sizes. Since B' can be obtained from B by replacing each vertex v with v', we know that $p_B(u_i) = u_{i+1}$ for $i = q, q + 1, ..., s - 1$. Again, by Lemma 3.14, we have $|U_B(x)| = 1$. As a result, by Lemma 3.13, $|U(B')| = |U(B)| - 1$ and thus the lemma holds. ■

We are ready to show the safeness of Rule 3.

Theorem 3.16. Rule 3 is safe.
Proof. For each branching B of D_Φ, by Lemma 3.13, we have $cost(B') = |U(B')| - |\Sigma'| = |U(B)| - |\Sigma| - |U_B(x)| + 1 = cost(B) - |U_B(x)| + 1 \leq cost(B)$, where B' is the branching corresponding to B in $D_{\Phi'}$. Therefore, $\delta(\Phi') \leq \delta(\Phi)$. In the following, we prove that $\delta(\Phi') \geq \delta(\Phi)$ by constructing a branching of D_Φ with cost $\delta(\Phi')$. Since y is a chain-like element of Φ', by Lemma 3.3, there exists an optimal branching B' of $D_{\Phi'}$ such that $|U_{B'}(y)| = 1$. Let B be the branching corresponding to B' in D_Φ. Since $|U_{B'}(y)| = 1$, by Lemma 3.15, $|U(B)| = |U(B')| + 1$. Therefore, $cost(B) = |U(B)| - |\Sigma| = |U(B')| + 1 - (|\Sigma'| + 1) = |U(B')| - |\Sigma'| = cost(B') = \delta(\Phi')$, from which $\delta(\Phi') \geq \delta(\Phi)$ is concluded. Consequently, the theorem holds. ■

By Lemma 3.3, there is an optimal branching B' of $D_{\Phi'}$ such that $|U_{B'}(y)| = 1$. In Section 5, we further prove that any optimal branching B' of $D_{\Phi'}$ satisfies $|U_{B'}(y)| = 1$. (See Corollary 5.5.) We remark that according to this result, the proof of Theorem 3.16 also shows that for any optimal branching B' of $D_{\Phi'}$, the branching corresponding to B' in D_Φ is an optimal branching of D_Φ. Therefore, an optimal branching of D_Φ can be obtained in $O(n)$ time from an optimal branching of $D_{\Phi'}$.

In summary, from Theorems 3.6, 3.7, 3.10, and 3.16, we know that by exhaustively applying Rules 1, 2, and 3, an input instance Φ of UB can be reduced to an equivalent instance of at most $4\delta(\Phi) - 1$ sets and at

most $3\delta(\Phi)$ elements. We conclude this section by showing that the number of sets, $4\delta(\Phi) - 1$, can be further reduced by one. More specifically, we show that after Φ is exhaustively reduced by Rules 1, 2, and 3, we can remove the root vertex from the output kernel. Consider a set family Φ that contains its ground set Σ. If the root vertex contains an element e that does not appear in any other vertex, since $|U_B(e)| = |\{(\Sigma, e)\}| = 1$ for any branching B of D_Φ, we can remove e from the ground set and the root vertex without changing $\delta(\Phi)$. Therefore, we may assume that each element of Σ appears in at least one non-root vertex. Let $\Phi' = \Phi - \Sigma$. Since Σ is the ground set of Φ', as mentioned, adding Σ to Φ' does not change $\delta(\Phi')$ [27]. Thus, $\delta(\Phi') = \delta(\Phi' \cup \{\Sigma\}) = \delta(\Phi)$. That is, removing the root vertex from Φ does not change $\delta(\Phi)$. As a result, we know that after Φ is exhaustively reduced by Rules 1, 2, and 3, we can further remove the root vertex.

4. Implementing the Kernelization Algorithm

For two vertices $u, v \in \Phi$, there are four possible relations: $u \supset v$, $u \subset v$, $u \cap v = \varnothing$, and u and v are in conflict. Let R be an $n \times n$ table that records the relation between every pair of vertices in Φ. Section 4.1 preprocesses Φ to compute R and the containment digraph D_Φ. Section 4.2 gives an efficient implementation of our kernelization algorithm, which requires $O(\max(m^{0.373}n^2, mn^{1.373}))$ time. Section 4.3 shows that given an optimal branching of the output kernel, an optimal branching of D_Φ can be obtained in $O(mn + n^2)$ time. Section 4.4 summarizes our kernelization results on UB and SR. Section 4.5 gives applications of our kernelization algorithm.

4.1 Preprocess

We precompute the relation table R by matrix multiplication. Denote the vertices in Φ by $v_1, v_2, ..., v_n$ and denote the elements in Σ by $e_1, e_2, ..., e_m$. Let X be an $n \times m$ matrix in which $X[i, k] = 1$ if v_i contains e_k and $X[i, k] = 0$ otherwise, where $1 \le i \le n$ and $1 \le k \le m$. Let Y be the multiplication of X and X^T, where X^T is the transpose of X. Clearly, each $Y[i, j]$, $1 \le i, j \le n$, equals the number of elements e_k, $1 \le k \le m$, that are contained in both v_i and v_j. That is, $Y[i, j] = |v_i \cap v_j|$. Given Y, the relation between any two vertices v_i and v_j can be determined as follows: $v_i \supset v_j$ if $Y[i, j] = |v_j|$; $v_i \subset v_j$ if $Y[i, j] = |v_i|$; $v_i \cap v_j = \varnothing$ if $Y[i, j] = 0$; and otherwise, u and v are in conflict.

Let ω be a fixed real number such that there exists an $O(n^\omega)$-time algorithm for multiplying two $n \times n$ square matrices. The two matrices X and X^T may not be square. We need the following folklore lemma.

Lemma 4.1. *The multiplication of an $n \times m$ matrix and an $m \times n$ matrix can be computed in $O(\max(mn^2 / m^{3-\omega}, mn^2 / n^{3-\omega}))$ time.*

A poof of Lemma 4.1 can be found in [5]. It is known that $\omega \le 2.373$ [11]. Consequently, the computation of R requires $O(\max(m^{0.373}n^2, mn^{1.373}))$ time. The containment digraph D_Φ is also precomputed, which can be easily done in $O(n^2)$ time by using the table R.

All existing exact and approximation algorithms for UB and SR [16, 18, 27] precompute the containment digraph of the input. A naive construction of the digraph requires $O(mn^2)$ time. Our construction takes $\max\{O(m^{0.373}n^2), O(mn^{1.373})\}$ time. We remark that this is the first $o(mn^2)$-time algorithm.

4.2 An efficient implementation

Our implementation consists of three phases: Phase 1 repeatedly applies Rule 1 until there are no non-root friendly vertices; Phase 2 repeatedly applies Rules 2 and 3 until there are no doubly chain-like elements; and Phase 3 repeatedly applies Rule 2 until the trailing vertex of all remaining chain-like elements are removed. The correctness of this three-phase implementation is ensured by the following three lemmas.

Lemma 4.2. Let Φ be a set family that contains no non-root friendly vertices. Let Φ' be obtained from Φ by a single application of Rule 2. Then, Φ' also contains no non-root friendly vertices.

Proof. Consider a non-root vertex $u \in \Phi'$. Since Φ does not contain non-root friendly vertices, u is in conflict with some vertex, say v, in Φ. If the application of Rule 2 does not remove v, u remains not friendly in Φ'. Assume that v is the vertex removed by Rule 2. Then, there is a chain-like element x with $t^-(x) = v$. We claim that u and $t(x)$ are in conflict. We prove the claim first. Clearly, $t(x) \not\subset u$, for otherwise, $t^-(x) \subset u$. In the proof of Theorem 3.10, it has been shown that $t(x)$ is an ideal parent of $t^-(x)$. Thus, $u \not\subset t(x)$, for otherwise, u and

$t^-(x)$ are not in conflict. Since $t^-(x) \cap u \neq \emptyset$, we have $t(x) \cap u \neq \emptyset$, which completes the proof the claim. Since $t(x) \in \Phi'$, u remains not friendly in Φ'. Therefore, the lemma holds. ■

Lemma 4.3. Let Φ be a set family that contains no non-root friendly vertices. Let Φ' be obtained from Φ by a single application of Rule 3. Then, Φ' also contains no non-root friendly vertices.
Proof. From the proof of Lemma 3.11, we know that Rule 3 preserves the relation between any pair of sets in Φ. Hence, a vertex v is friendly in Φ if and only if $v' = v - \{x\}$ is friendly in Φ', where x is the doubly chain-like element removed by Rule 3. Therefore, the lemma holds. ■

Lemma 4.4. Let Φ' be obtained by applying Rule 2 on a set family Φ. Then, Φ and Φ' has the same set of chain-like elements. In addition, if Φ contains no doubly chain-like elements, Φ' also contains no doubly chain-like elements.
Proof. Consider an element e. Clearly, if e is chain-like in Φ, it remains chain-like in Φ'. Assume that e is non-chain-like in Φ. Since e is not chain-like in Φ, there exist two vertices u, v in Φ that both contain e and are in conflict. If neither of them is removed, e remains non-chain-like. By symmetry, assume that v is the vertex removed by Rule 2. Then, there is a chain-like element x such that $t^-(x) = v$. Since u and $t^-(x)$ are in conflict, from the proof of Lemma 4.2, we know that u and $t(x)$ are also in conflict. Since Φ' contains both u and $t(x)$, e is not chain-like in Φ'. As a result, we know that e is chain-like in Φ if and only if it is chain-like in Φ'. Therefore, Φ and Φ' has the same set of chain-like elements.

Let C be the set of chain-like elements of Φ. Assume that Φ contains no doubly chain-like element. By Lemma 3.4, no two elements $a, b \in C$ have $t(a) \subseteq t(b)$. Let $v = t^-(x)$ be the vertex removed by Rule 2, where $x \in C$. The vertex $t^-(x)$ is not the tail vertex of any element in C, for otherwise, x is doubly chain-like. Therefore, removing v does not change the tail vertex of any element in C. That is, after the removal of v, no two elements $a, b \in C$ have $t(a) \subseteq t(b)$. Since C is the set of chain-like elements of Φ', by Lemma 3.4, Φ' contains no doubly chain-like elements. This completes the proof. ■

When a vertex v is removed by Rules 1 or 2, we use a variable $\tau(v)$ to store an ideal parent of v, so that an optimal branching of Φ can be recovered from an optimal branching of the output kernel. We proceed to describe the details.

4.2.1 Phase 1

For each vertex $v \in \Phi$, let $conflict(v)$ be the set of vertices in Φ that are in conflict with v. By definition, a vertex v is friendly if and only if $conflict(v) = \emptyset$. Clearly, after removing a friendly vertex v from Φ, $conflict(\cdot)$ of all the other vertices do not change. Phase 1 is done as follows. First, $|conflict(v)|$ is computed for each vertex $v \in \Phi$, which takes $O(n^2)$ time by using the table R. Next, for each non-root vertex v with $|conflict(\cdot)| = 0$, we set $\tau(v)$ as the smallest out-neighbor of v and then remove v from Φ. Note that from the proof of Theorem 3.10, $\tau(v)$ is an ideal parent of v. For each vertex v with $|conflict(\cdot)| = 0$, its removal and the computation of $\tau(v)$ take $O(n)$ time. Therefore, Phase 1 requires $O(n^2)$ time.

4.2.2 Phase 2

Let C be the set of chain-like elements of Φ. Using the table R, the elements of C and their tail vertices are computed as follows. First, we create a sequence L containing all vertices of Φ, in non-decreasing order of their sizes. Next, for each element e, we determine whether it is chain-like by extracting from L the sequence $(u_1, u_2, ..., u_{s-1}, u_s)$ of vertices containing e and then check whether $u_i \subset u_{j+1}$ for $1 \leq j < s$; and if e is chain-like, we set u_1 as the tail of e. The computation of L takes $O(n \lg n)$ time. The determination for each element takes $O(n)$ time. Thus, the above computation requires $O(mn)$ time.

Let Z be the set of vertices $v \in \Phi$ such that v is the tail vertex of at least one chain-like element. For each $v \in Z$, let $X(v)$ be the set of chain-like elements x with $t(x) = v$, $Z_{in}(v)$ be the set of vertices in Z that are in-neighbors of v, and $Z_{out}(v)$ be the set of vertices in Z that are out-neighbors of v. By Lemma 3.4, a chain-like element x is doubly chain-like if and only if there is a chain-like element x' such that $t(x') = t(x)$ or $t(x')$ is an in-neighbor of $t(x)$. As a result, we know that for each vertex $v \in Z$, an element in $X(v)$ is doubly chain-like if and only if $|X(v)| \geq 2$ or $|Z_{in}(v)| > 0$. Using the table R, the set C, and the tail vertex $t(x)$ of each $x \in C$, we find Z and compute $X(v)$, $Z_{in}(v)$, and $Z_{out}(v)$ for each $v \in Z$. Clearly, this computation requires $O(n^2)$ time.

We proceed to describe Phase 2. If there is a vertex v in Z with $|X(v)| \geq 2$ or $|Z_{in}(v)| > 0$, we perform the

following. First, take an element $x \in X(v)$. Note that x is doubly chain-like and $t(x) = v$. Next, determine whether $t^-(x) = v - \{x\}$ is a vertex in Φ. Then, there are three cases.

Case 1: $t^-(x) \notin \Phi$. Rule 3 is used to remove the element x. By Corollary 3.12, any other element in C is still chain-like. We update $X(v) = X(v) - \{x\}$; and then, if $X(v)$ is empty, we delete v from the following sets: Z, $Z_{\text{in}}(u)$ of each $u \in Z_{\text{out}}(v)$, and $Z_{\text{out}}(u)$ of each $u \in Z_{\text{in}}(v)$.

Case 2: $t^-(x) \in \Phi$ and $t^-(x) \notin Z$. Rule 2 is used to remove $t^-(x)$. According to the proof of Theorem 3.10, we set $\tau(t^-(x))$ as $t(x)$. Note that no updates on $X(\cdot)$, $Z_{\text{in}}(\cdot)$, and $Z_{\text{out}}(\cdot)$ are required, since $t^-(x) \notin Z$ and by Lemma 4.4, the set of chain-like elements is the same.

Case 3: $t^-(x) \in Z$. As in Case 2, Rule 2 is used to remove $t^-(x)$ and we set $\tau(t^-(x))$ as $t(x)$. By Lemma 4.4, the set of chain-like elements is the same. However, the removal of $t^-(x)$ makes $t(x)$ the new tail vertices of the elements in $X(t^-(x))$. Thus, we update $X(t(x)) = X(t(x)) \cup X(t^-(x))$; and we delete $t^-(x)$ from the following sets: Z, $Z_{\text{in}}(u)$ of each $u \in Z_{\text{out}}(t^-(x))$, and $Z_{\text{out}}(u)$ of each $u \in Z_{\text{in}}(t^-(x))$.

Phase 2 repeatedly perform the above steps until no vertices v in Z with $|X(v)| \geq 2$ or $|Z_{\text{in}}(v)| > 0$. Consider a chain-like element x. Since $t(x)$ is the smallest vertex containing x, determining whether $t^-(x)$ is a vertex in Φ can be done in $O(n)$ time by simply checking whether $t(x)$ has an in-neighbor of size $|t(x)| - 1$. As a result, it is easy to see that each iteration of Phase 2 takes $O(n)$ time. Since there are at most $m + n$ iterations, Phase 2 requires $O(mn + n^2)$ time.

4.2.3 Phase 3

Let $C' \subseteq C$ be the set of chain-like elements that are not removed in Phase 2. Note that by Lemma 4.4, applying Rule 2 does not change the set of chain-like elements. Phase 3 performs the following for each $x \in C'$: if $t^-(x) \in \Phi$, set $\tau(t^-(x))$ as $t(x)$ and then remove $t^-(x)$ from Φ. Since whether $t^-(x) \in \Phi$ can be determined in $O(n)$ time and C' contains at most m elements, Phase 3 requires $O(mn)$ time.

4.3 Solution lifting

Let Φ' be the set family obtained from Φ by performing Rules 1, 2, or 3 once. If Rules 1 or 2 is used to remove a vertex v, as remarked, the proof of Theorem 3.10 shows that given an ideal parent of v and an optimal branching of $D_{\Phi'}$, we can obtain in $O(n)$ time an optimal branching of D_{Φ}. If Rule 3 is used, as remarked, the proof of Theorem 3.16 shows that for any optimal branching of $D_{\Phi'}$, its corresponding branching in D_{Φ} is an optimal branching of D_{Φ}. The implementation in Section 4.2 records an ideal parent for each removed vertex. At most n vertices and m elements are removed. Consequently, we conclude that a solution to the input Φ can be obtained in $O(mn + n^2)$ time from an optimal branching of the output kernel.

4.4 Kernelization results on UB and SR

The preprocessing in Section 4.1 requires $O(\max(m^{0.373}n^2, mn^{1.373}))$ time. The implementation in Section 4.2 and the solution lifting algorithm in Section 4.3 both require $O(mn + n^2)$ time. By combining these and Theorems 3.6, 3.7, 3.10, and 3.16, we obtain the following.

Theorem 4.6. In $O(\max(m^{0.373}n^2, mn^{1.373}))$ time, we can transform a set family Φ with $\delta(\Phi) > 0$ into another set family Φ' such that $\delta(\Phi') = \delta(\Phi)$, Φ' contains at most $4\delta(\Phi) - 1$ sets, and the ground set of Φ' contains at most $3\delta(\Phi)$ elements. In addition, an optimal branching of Φ' can be lifted to an optimal branching of Φ in $O(mn + n^2)$ time.

Our kernel result is formally presented, using the language of parameterized complexity, as follows. For a parameterized problem, two instances (I, k) and (I', k') are *equivalent* if (I, k) is a yes-instance if and only if (I', k') is a yes-instance. A *kernelization algorithm* for a parameterized problem is a polynomial-time algorithm that transforms an instance (I, k) to an equivalent instance (I', k') (the *kernel*) such that k' and the size of I' are bounded by a function of k [4]. As customary in the field, we allow kernelization algorithms to return the answer yes or the answer no when it can correctly decide whether the input is a yes-instance or a no-instance. Define the *parameterized uncovering branching problem* (PUB) as deciding whether $\delta(\Phi) \leq k$ for

a set family Φ and a parameter k. Then, we have the following.

Theorem 4.7. PUB admits an $O(\max(m^{0.373}n^2, mn^{1.373}))$-time kernelization algorithm that computes a kernel that contains at most $4k - 1$ sets over a ground set of at most $3k$ elements.
Proof. Let (Φ, k) be an instance of PUB. Recall that whether $\delta(\Phi) = 0$ can be determined in $O(mn)$ time. If $\delta(\Phi) = 0$, we know that (Φ, k) is a yes-instance. Assume that $\delta(\Phi) > 0$. By Theorem 4.6, in $O(\max(m^{0.373}n^2, mn^{1.373}))$ time, we obtain a reduced instance Φ' with $\delta(\Phi') = \delta(\Phi)$. According to Theorem 4.6, if $|\Phi'| > 4k - 1$ or the ground set of Φ' contains more than $3k$ elements, (Φ, k) is a no-instance. Otherwise, (Φ', k) is a kernel of the desired size. ■

Define the *parameterized split-row problem* (PSR) as determining whether $\varepsilon(M) \leq k$ for a given binary matrix M and a parameter k. Similarly, we have the following.

Theorem 4.8. PSR admits an $O(\max(m^{0.373}n^2, mn^{1.373}))$-time kernelization algorithm that computes a kernel that contains at most $4k - 1$ columns and at most $3k$ rows.
Proof. Let (M, k) be an instance of PSR. As mentioned in Section 1, it is easy to construct a set family represented by M. We construct such a set family Φ, which contains n sets over a ground set of m elements. The algorithm in Theorem 4.7 is performed on (Φ, k). If (Φ, k) is a yes (resp. no) instance, then (M, k) is a yes (resp. no) instance; otherwise, we obtain a reduced instance Φ'. Let M' be the matrix representation of Φ'. Since Φ' contains at most $4k - 1$ sets over a ground set of at most $3k$ elements, M' is a binary matrix with at most $4k - 1$ columns and at most $3k$ rows. By Theorems 2.1 and 4.6, $\varepsilon(M) = \delta(\Phi) = \delta(\Phi') = \varepsilon(M')$. As a result, (M', k) is a kernel of the desired size. The above algorithm requires $O(\max(m^{0.373}n^2, mn^{1.373}))$ time. Therefore, the theorem holds. ■

4.5 Leveraging the kernel

When $\delta(\Phi)$ is small, our kernelization algorithm can be used to reduce the running time of an algorithm for UB as follows. First, compute a kernel Φ'. Next, run the algorithm to find an optimal branching B' of $D_{\Phi'}$. Finally, lift the branching B' to an optimal branching of D_Φ. Hujdurović *et al.* [16] had an $O^*(n^n)$-time algorithm for UB. Husić *et al*. [18] formulated UB as an ILP of $O(mn + n^2)$ variables and $O(mn)$ constraints. Sheu and Wang [27] had an FPT algorithm that runs in time $O(mn^2 + mn\ 2^{\min(n,\ 2\delta(\Phi))} + n^2\ 3^{\min(n,\ 2\delta(\Phi))})$ or $O(mn^5 \lg(mn) \operatorname{lglg}(mn)\ 2^{\min(n,\ 2\delta(\Phi))})$. Accordingly, we obtain the following.

Theorem 4.9. UB can be solved in time $O^*((4\delta(\Phi) - 1)^{4\delta(\Phi)})$, $O(\max(m^{0.373}n^2, mn^{1.373}) + \delta(\Phi)^2\ 3^{2\delta(\Phi)})$, or $O(\max(m^{0.373}n^2, mn^{1.373}) + \delta(\Phi)^6 \lg \delta(\Phi) \operatorname{lglg} \delta(\Phi)\ 2^{2\delta(\Phi)})$, where $m = |\Sigma|$ and $n = |\Phi|$. In addition, UB can be reduced to solving an ILP of $O(\delta(\Phi)^2)$ variables and constraints.

By Theorem 2.1, an optimal feasible sequence of split-row operations on a matrix M can be obtained in $O(mn^2)$ time from an optimal branching of D_Φ, where Φ is a set family represented by M. Therefore, Theorem 4.9 immediately leads to the following.

Theorem 4.10. SR can be solved in time $O^*((4\varepsilon(M) - 1)^{4\varepsilon(M)})$, $O(mn^2 + \varepsilon(M)^2\ 3^{2\varepsilon(M)})$, or $O(mn^2 + \varepsilon(M)^6 \lg \varepsilon(M) \operatorname{lglg} \varepsilon(M)\ 2^{2\varepsilon(M)})$ time. In addition, SR can be reduced to solving an ILP of $O(\varepsilon(M)^2)$ variables and constraints.

5. Approximability results

We say that an algorithm is a ρ-*approximation algorithm* (or that its *approximation factor* is ρ) *for* $\beta(\Phi)$ if it finds a branching B with $|U(B)| \leq \rho \cdot \beta(\Phi)$. A ρ-*approximation algorithm for* $\delta(\Phi)$ is defined similarly. This section presents approximation algorithms for $\beta(\Phi)$ and $\delta(\Phi)$. As in [16], the approximation factors of our algorithms are in terms of the height and width of Φ. Recall that an antichain of Φ is a set of pairwise non-nested vertices. The *width* of Φ, denoted by $w(\Phi)$, is the largest size of an antichain in Φ. A *chain* of Φ is a set of pairwise nested vertices in Φ. The *height* of Φ, denoted by $h(\Phi)$, is the largest size of a chain in Φ. In the case when $\Sigma \notin \Phi$, adding Σ to Φ will increase $h(\Phi)$ by 1. Therefore, in this section, we do not assume that $\Sigma \in \Phi$.

Assume that D_Φ is precomputed by using the preprocessing in Section 4.1. Section 5.1 presents a two-phase approach to design approximation algorithms for UB. Section 5.2 gives three new approximation algorithms for $\beta(\Phi)$, which have approximation factors $0.5w(\Phi)$, $0.5h(\Phi) + 0.5$, and $0.368h(\Phi) + 0.633$, respectively. Section 5.3 shows that the approximation algorithms in Section 5.2 are also approximation algorithms for $\delta(\Phi)$, which have approximation factors $w(\Phi) - 1$, $h(\Phi)$, and $0.736h(\Phi) + 0.266$, respectively.

5.1 A two-phase approach

A vertex $v \in \Phi$ is not *regular* (or *non-regular*) if it is contained in both of a pair of conflicting vertices. For example, in Figure 5(a), v_5, v_7, and v_8 are non-regular, since they are contained in both of the pair of conflicting vertices v_1 and v_2. Let v be a vertex and $(u_1, u_2, ..., u_{s-1}, u_s)$ be the sequence of vertices strictly containing v, in non-decreasing order of their sizes. It is easy to observe that v is regular if and only if $u_1 \subset u_2 \subset ... \subset u_s$. Therefore, no two vertices containing a regular vertex have the same size. For each regular vertex v, the *predetermined parent* of v, denote by $\pi(v)$, is the smallest vertex strictly containing v. Note that $\pi(v)$ may be null. Let B_R be the branching $\{(v, \pi(v)) \mid v \text{ is a regular vertex}\}$. Consider a branching B which does not contain B_R as a subset. Let r be the largest regular vertex such that $p_B(r) \neq \pi(r)$ (with ties broken arbitrarily). A *regularization operation* on B is defined as re-assigning $p_B(r)$ to be $\pi(r)$. We need the following two lemmas.

Lemma 5.1. [27] Let $B \supseteq B_R$ be a branching. For each chain-like element e, we have $|U_B(e)| = 1$.

Lemma 5.2. [27] Let B' be the branching obtained from a branching $B \not\supseteq B_R$ by performing a regularization operation. Then, $U(B') \subseteq U(B)$.

For ease of presentation, the proofs of Lemmas 5.1 and 5.2 in [27] assumed that $\Sigma \in \Phi$. We remark that it is straightforward to adapt the proofs to handle the case when $\Sigma \notin \Phi$. Recall that Σ_N is the set of non-chain-like elements. We denote by Σ_C the set of chain-like elements. The set $U(B)$ of a branching B is partitioned into two subsets: $U_1(B) = \{(e, v) \mid e \in \Sigma_N, (e, v) \in U(B)\}$ and $U_2(B) = \{(e, v) \mid e \in \Sigma_C, (e, v) \in U(B)\}$. Elements in $U_1(B)$ (resp. $U_2(B)$) are called B-uncovered pairs of non-chain-like (resp. chain-like) elements. Define a subroutine as follows.

REGULARIZE(B): transform an input branching B into a branching B' by repeatedly performing a regularization operation on B until $B \supseteq B_R$; and then return B'.

Lemma 5.3. Assume that D_Φ is precomputed. REGULARIZE(B) returns in $O(n^2)$ time a branching B' with $|U_1(B')| \leq |U_1(B)|$ and $|U_2(B')| = |\Sigma_C|$.

Proof. By Lemma 5.2, $U(B') \subseteq U(B)$. Since $U(B') \subseteq U(B)$, we have $U_1(B') \subseteq U_1(B)$ and thus $|U_1(B')| \leq |U_1(B)|$. Since $B' \supseteq B_R$, by Lemma 5.1, $|U_{B'}(e)| = 1$ for each $e \in \Sigma_C$ and thus $|U_2(B')| = |\Sigma_C|$. Each regularization operation re-assigns the parent of a regular vertex to be its predetermined parent, which requires $O(1)$ time if the predetermined parent is given. Predetermined parents of all regular vertices can be computed in $O(n^2)$ time [27]. Therefore, REGULARIZE(B) requires $O(n^2)$ time, which completes the proof. ■

Note that $|U_2(B)| \geq |\Sigma_C|$ holds for any branching B, since $|U_B(e)| \geq 1$ for each $e \in \Sigma_C$. Lemma 5.3 plays an important role in our approximation algorithms. It indicates that in the finding of an (near) optimal branching, we can first focus on reducing uncovered pairs of non-chain-like elements; and then perform REGULARIZE to minimize uncovered pairs of chain-like elements. Let $\mathcal{B}$ be the set of all branchings of D_Φ and let $\beta_1(\Phi) = \min_{B \in \mathcal{B}} |U_1(B)|$. Since every branching B has $|U_1(B)| \geq \beta_1(\Phi)$ and $|U_2(B)| \geq |\Sigma_C|$, we have $\beta(\Phi) \geq \beta_1(\Phi) + |\Sigma_C|$. Given a branching B with $|U_1(B)| = \beta_1(\Phi)$, by Lemma 5.3, it can be transformed into a branching B' with $|U_1(B')| \leq \beta_1(\Phi)$ and $|U_2(B')| = |\Sigma_C|$. Therefore, we have the following.

Lemma 5.4. $\beta(\Phi) = \beta_1(\Phi) + |\Sigma_C|$ and a branching B is optimal if and only if $|U_1(B)| = \beta_1(\Phi)$ and $|U_2(B)| = |\Sigma_C|$.

Since $|U_2(B)| = |\Sigma_C|$ if and only if $|U_B(e)| = 1$ for each $e \in \Sigma_C$, Lemma 5.4 immediately leads to the following, which ensures the correctness of the solution lifting algorithm in Section 4.3.

Corollary 5.5. For any optimal branching B, we have $|U_B(e)| = 1$ for each chain-like element e.

Theorem 5.6. Let B be the output of a ρ-approximation algorithm for $\beta_1(\Phi)$, where $\rho \geq 1$. Then, REGULARIZE(B) returns a branching B' that is a ρ-approximate solution for $\beta(\Phi)$.
Proof. By the definition of a ρ-approximation algorithm, we have $|U_1(B)| \leq \rho\beta_1(\Phi)$. By Lemma 5.3, $|U(B')| = |U_1(B')| + |U_2(B')| \leq \rho\beta_1(\Phi) + |\Sigma_C|$. By Lemma 5.4, $\beta(\Phi) = \beta_1(\Phi) + |\Sigma_C|$. Since $\rho \geq 1$, we have $|U(B')| \leq \rho\beta_1(\Phi) + |\Sigma_C| \leq \rho(\beta_1(\Phi) + |\Sigma_C|) = \rho\beta(\Phi)$. Therefore, the theorem holds. ■

Theorem 5.6 suggests the following two-phase approach, referred as TWOPHASEUB, for designing ρ-approximation algorithms for $\beta(\Phi)$.

Phase 1. Perform a ρ-approximation algorithm for $\beta_1(\Phi)$ to obtain a branching B.
Phase 2. Perform REGULARIZE(B) to obtain a ρ-approximate solution for $\beta(\Phi)$.

5.2 Approximating β(Φ)

Based on TWOPHASEUB, this section gives three approximation algorithms for $\beta(\Phi)$. We proceed to present the first.

Theorem 5.7. There is a $0.5w(\Phi)$-approximation algorithm for $\beta(\Phi)$ running in time $O(\max(m^{0.373}n^2, mn^{1.373})) + \tilde{O}(n^{2.373})$.
Proof. Hujdurović *et al.*'s [16] had a $w(\Phi)$-approximation algorithm for $\beta(\Phi)$. Assume that this algorithm is performed in Phase 1 of TWOPHASEUB. In the following, we show that TWOPHASEUB is a $0.5w(\Phi)$-approximation algorithm for $\beta(\Phi)$ by showing that Hujdurović *et al.*'s algorithm is a $0.5w(\Phi)$-approximation algorithm for $\beta_1(\Phi)$. Let B be the output of Hujdurović *et al.*'s algorithm. Hujdurović *et al*. showed that $|U_B(e)| \leq w(\Phi)$ for each $e \in \Sigma$. Therefore, we have $|U_1(B)| \leq w(\Phi)|\Sigma_N|$. By Lemma 3.3, for any branching, each non-chain-like element contributes at least two uncovered pairs. Thus, $\beta_1(\Phi) \geq 2|\Sigma_N|$. As a result, $|U_1(B)| \leq w(\Phi)|\Sigma_N| \leq 0.5w(\Phi)\beta_1(\Phi)$ is concluded. Therefore, Hujdurović *et al.*'s algorithm is a $0.5w(\Phi)$-approximation algorithm for $\beta_1(\Phi)$. Given D_Φ, Hujdurović *et al.*'s algorithm requires $\tilde{O}(n^{2.373})$ time. The construction of D_Φ needs $O(\max(m^{0.373}n^2, mn^{1.373}))$ time. Therefore, TWOPHASEUB requires $O(\max(m^{0.373}n^2, mn^{1.373})) + \tilde{O}(n^{2.373})$ time if Hujdurović *et al.*'s algorithm is performed in Phase 1. This completes the proof. ■

We proceed to present the other two algorithms. For a branching B, define $g(B)$ as the number of B-covered pairs (e, v) with $e \in \Sigma_N$; and define $g_{\max}(\Phi) = \max_{B \in \mathcal{B}} g(B)$. The relation between $\beta_1(\Phi)$ and $g_{\max}(\Phi)$ is described as follows. Consider a branching B. Recall that V_e is the set of vertices containing an element e. The number of target pairs of non-chain-like elements is $\sum_{e\in\Sigma_N} |V_e|$. Each target pair of non-chain-like elements is either B-covered or B-uncovered. Thus, $|U_1(B)| + g(B) = \sum_{e\in\Sigma_N} |V_e|$ holds for all branchings B. As a result, $\beta_1(\Phi) + g_{\max}(\Phi) = \sum_{e\in\Sigma_N} |V_e|$; and thus finding a branching B that minimizes $|U_1(B)|$ is equivalent to finding a branching B that maximizes $g(B)$. An *r-approximation algorithm for* $g_{\max}(\Phi)$ is an algorithm that finds a branching B with $g(B) \geq r g_{\max}(\Phi)$, where $r \leq 1$. The following lemma shows that an approximation algorithm for $g_{\max}(\Phi)$ is also an approximation algorithm for $\beta_1(\Phi)$.

Lemma 5.8. Let B be a branching with $g(B) \geq r g_{\max}(\Phi)$, where $r \leq 1$. Then, $|U_1(B)| \leq \rho\beta_1(\Phi)$, where $\rho = ((1 - r)h(\Phi) + r)$.
Proof. To prove the lemma, we first give an upper bound on $\sum_{e\in\Sigma_N} |V_e|$. For each element e, let h_e denote the size of a largest chain in V_e. Hujdurović *et al.* had the following interesting observation: for any optimal branching B^*, we have $h_e \cdot |U_{B^*}(e)| \geq |V_e|$ for each $e \in \Sigma$ [16, Theorem 4.11]. From this observation, we know that

$$\sum_{e\in\Sigma_N} |V_e| \leq \sum_{e\in\Sigma_N} h_e|U_{B^*}(e)| \leq h(\Phi)\sum_{e\in\Sigma_N} |U_{B^*}(e)| = h(\Phi)\beta_1(\Phi).$$

From $g(B) \geq r g_{\max}(\Phi)$ and $\sum_{e\in\Sigma_N} |V_e| \leq h(\Phi)\beta_1(\Phi)$, we obtain

$$\begin{aligned} |U_1(B)| &= \sum_{e\in\Sigma_N} |V_e| - g(B) \\ &\leq \sum_{e\in\Sigma_N} |V_e| - r g_{\max}(\Phi) \end{aligned}$$

$= \sum_{e \in \Sigma_N} |V_e| - r(\sum_{e \in \Sigma_N} |V_e| - \beta_1(\Phi))$
$= (1 - r)\sum_{e \in \Sigma_N} |V_e| + r\beta_1(\Phi).$
$\leq (1 - r)h(\Phi)\, \beta_1(\Phi) + r\beta_1(\Phi)$
$= ((1 - r)h(\Phi) + r)\beta_1(\Phi).$

Therefore, the lemma holds. ■

According to Theorem 5.6 and Lemma 5.8, if an r-approximation algorithm for $g_{\max}(\Phi)$ is performed in Phase 1, the approximation factor of TWOPHASEUB is $\rho = ((1 - r)h(\Phi) + r)$. In the following, two approximation algorithms for $g_{\max}(\Phi)$ are presented. Both the presented algorithms are based on a reduction to *the submodular maximization subject to a matroid problem* (SUB-M) [9, 10], which is defined as follow. A *matroid* is a pair $(W, \mathcal{I})$, where W is a finite set and $\mathcal{I} \subseteq 2^W$, that satisfies the following:

(C1) for a set $Y \in \mathcal{I}$, $X \subset Y$ implies $X \in \mathcal{I}$,
(C2) for all $X, Y \in \mathcal{I}$, $|X| < |Y|$ implies that there exists $w \in Y - X$ such that $X \cup \{w\} \in \mathcal{I}$.

A function f: $2^W \rightarrow \mathbb{R}^+$ is *monotone* if $f(X) \leq f(Y)$ for all $X \subset Y \subseteq W$. The function f is *submodular* if for any two sets $X \subset Y$ and element $w \in W - Y$, we have $f(X \cup \{w\}) - f(X) \geq f(Y \cup \{w\}) - f(Y)$. Given a matroid $(W, \mathcal{I})$ and a monotone submodular function f, SUB-M is to find a set $X \in \mathcal{I}$ that maximizes $f(X)$. Note that in SUB-M, the domain of f is 2^W. That is, $f(X)$ is defined for each $X \subseteq W$. We denote a SUB-M instance by the tuple $(W, \mathcal{I}, f)$, where $(W, \mathcal{I})$ is the input matroid and f is the objective function. SUB-M is NP-hard and approximation algorithms for SUB-M can be found in [9, 10].

Let A be the arc set of D_Φ. In the following, we show that the computation of $g_{\max}(\Phi)$ can be solved by using algorithms for SUB-M. More specifically, we show in the following two lemmas that $(A, \mathcal{B}, g)$ is an instance of SUB-M.

Lemma 5.9. $(A, \mathcal{B})$ is a matroid.
Proof. By the definition of branchings, $\mathcal{B} \subseteq 2^A$. Since any subset of a branching is also a branching, (C1) is satisfied. Consider two branchings $B, B' \in \mathcal{B}$ such that $|B| < |B'|$. Since $|B| < |B'|$, there is a vertex v such that $p_B(v) = \text{null}$ and $p_{B'}(v) \neq \text{null}$. Clearly, $B \cup \{(v, p_{B'}(v))\} \in \mathcal{B}$ and $(v, p_{B'}(v)) \in B' - B$. Therefore, (C2) also holds, which completes the proof. ■

According to the definition of SUB-M, the objective function g should be defined for any subset $X \subseteq A$, even if X is not a branching. For this purpose, we generalize the notion of B-covered pairs as follows. Let $X \subseteq A$ be a subset, which may not be a branching. For a target pair (e, v), we say e is *X-covered* in v if X contains at least one arc (u, v) such that $e \in u$. An *X-covered pair* is a target pair (e, v) such that e is X-covered in v. The terms X-uncovered and X-uncovered pairs are defined similarly. We define $g(X)$ as the number of X-covered pairs (e, v) with $e \in \Sigma_N$.

Lemma 5.10. g is monotone and submodular.
Proof. Adding an arc to a subset $B \subseteq A$ cannot decrease the number of B-covered pairs. Thus, g is monotone. Consider two subsets X and Y of A such that $X \subset Y$. Let (u, v) be an arc of D_Φ that is not in Y. In the following, we prove the submodularity of g by showing that $g(X \cup \{(u, v)\}) - g(X) \geq g(Y \cup \{(u, v)\}) - g(Y)$. Define C_X to be the set of vertices z with $(z, v) \in X$. Define C_Y similarly. For each vertex $a \in \Phi$, let a_N denote $a \cap \Sigma_N$. The value $g(X \cup \{(u, v)\}) - g(X)$ is the number of non-chain-like elements in v that is covered in $(X \cup \{(u, v)\})$, but uncovered in X. Hence, $g(X \cup \{(u, v)\}) - g(X) = |u \cap (v_N - \cup_{c \in C_X} c_N)|$. Similarly, $g(Y \cup \{(u, v)\}) - g(Y) = |u \cap (v_N - \cup_{c \in C_Y} c_N)|$. Since $C_X \subseteq C_Y$, it is easy to conclude that $|u \cap (v_N - \cup_{c \in C_X} c_N)| \geq |u \cap (v_N - \cup_{c \in C_Y} c_N)|$. Hence, the lemma holds. ■

Fisher *et al*. [10] had a 0.5-approximation algorithm for SUB-M. Since $(A, \mathcal{B}, g)$ is an instance of SUB-M, their algorithm is a 0.5-approximation algorithm for $g_{\max}(\Phi)$. Furthermore, by Lemma 5.8, their algorithm is a ρ-approximation algorithm for $\beta_1(\Phi)$, where $\rho = (1 - 0.5)h(\Phi) + 0.5 = 0.5h(\Phi) + 0.5$. By applying this result to Phase 1, TWOPHASEUB finds a $(0.5h(\Phi) + 0.5)$-approximate solution for $\beta(\Phi)$. Fisher *et al*. did not provide a time complexity analysis for their algorithm. In Appendix B, we show that their algorithm requires

$O(mn^2)$ time when the input is $(A, \mathcal{B}, g)$. We obtain the following.

Theorem 5.11. There is a $(0.5h(\Phi) + 0.5)$-approximation algorithm for $\beta(\Phi)$ running in time $O(mn^2)$.

Filmus and Ward [9] had a randomized algorithm for SUB-M. They presented the algorithm on the oracle model, in which f is specified as a value oracle and $\mathcal{I}$ is specified as a membership oracle. More specifically, given a subset $X \subseteq W$, the value oracle returns $f(X)$, and the membership oracle returns whether $X \in \mathcal{I}$. Filmus and Ward's randomized algorithm requires $\tilde{O}(\sigma l^7 |W|^2)$ time and $\tilde{O}(\sigma l^7 |W|^2)$ oracle queries, where l is the size of the largest set in $\mathcal{I}$ and $\sigma > 0$ is a given parameter. With probability $1 - O(|W|^{-\sigma})$, their algorithm achieves an approximation factor of $1 - 1/e$, where $e \approx 2.718$ is the Euler's number. By Lemmas 5.8, Filmus and Ward's randomized algorithm is a ρ-approximation algorithm for $\beta_1(\Phi)$, where $\rho = (1 - (1 - 1/e))h(\Phi) + (1 - 1/e) \approx 0.368h(\Phi) + 0.633$. By applying this result to Phase 1, TwoPhaseUB finds a $(0.368h(\Phi) + 0.633)$-approximate solution for $\beta(\Phi)$ with probability $1 - O(|A|^{-\sigma})$. Given a subset $X \subseteq A$, it is easy to compute $g(X)$ in $O(m|X|) = O(m|A|)$ time; and it is easy to determine whether $X \in \mathcal{B}$ in $O(|X|) = O(|A|)$ time. In our application, $l \leq n - 1$ and $|W| = |A|$; and therefore, the time complexity is $\tilde{O}(\sigma n^7 |A|^2 \times m|A|) = \tilde{O}(\sigma mn^7 |A|^3)$.

Theorem 5.12. There is a randomized approximation algorithm for $\beta(\Phi)$ running in time $\tilde{O}(\sigma mn^7 |A|^3)$, where $\sigma > 0$ is a given parameter. With probability $1 - O(|A|^{-\sigma})$, the algorithm has an approximation factor of $0.368h(\Phi) + 0.633$.

The success probability, $1 - O(|A|^{-\sigma})$, in Theorem 5.12 is a constant when $|A| = 1$. The following theorem provides an alternative bound on the success probability.

Theorem 5.13. There is a randomized approximation algorithm for $\beta(\Phi)$ running in time $\tilde{O}(\sigma mn^{13})$, where $\sigma > 0$ is a given parameter. With probability $1 - O(n^{-\sigma})$, the algorithm has an approximation factor of $0.368h(\Phi) + 0.633$.

Proof. Three cases are considered.

Case 1: $|A| = 1$. That is, D_Φ contains only one arc. In this case, UB is solved by simply outputting A.

Case 2: $2 \leq |A| \leq n$. The algorithm in Theorem 5.12 runs in $\tilde{O}(\sigma mn^7 |A|^3) = \tilde{O}(\sigma mn^{10})$ time and succeeds with probability $1 - O(|A|^{-\sigma})$. By repeating the algorithm $\lg n$ times and selecting the best output, we obtain an algorithm with success probability at least $1 - O(n^{-\sigma})$ running in time $\tilde{O}(\sigma mn^{10} \lg n) = \tilde{O}(\sigma mn^{13})$.

Case 3: $|A| > n$. Since $|A| \leq n^2$, the algorithm in Theorem 5.12 runs in $\tilde{O}(\sigma mn^7 |A|^3) = \tilde{O}(\sigma mn^{13})$ time and succeeds with probability $1 - O(|A|^{-\sigma}) = 1 - O(n^{-\sigma})$.

Therefore, the theorem holds. ■

According to Theorem 2.1, we also obtain the following approximability results for $\alpha(M)$.

Theorem 5.14. The following approximation algorithms for $\alpha(M)$ exist, where Φ is a set family with M as its matrix representation:
(1) a $0.5w(\Phi)$-approximation algorithm running in time $O(mn^2) + \tilde{O}(n^{2.373})$,
(2) a $(0.5h(\Phi) + 0.5)$-approximation algorithm running in time $O(mn^2)$, and
(3) a $\tilde{O}(\sigma mn^{13})$-time randomized algorithm having an approximation factor of $0.368h(\Phi) + 0.633$ with probability $1 - O(n^{-\sigma})$, where $\sigma > 0$ is a given parameter.

5.3 Approximating δ(Φ)

This section presents approximation algorithms for $\delta(\Phi)$. The presented algorithm is mainly based upon the following lemma, which says that TwoPhaseUB also has a performance guarantee for $\delta(\Phi)$.

Lemma 5.15. Let B be the output of a ρ-approximate algorithm for $\beta_1(\Phi)$. Then, Regularize(B) returns a

branching B' that is a $(2\rho - 1)$-approximate solution for $\delta(\Phi)$.

Proof. We need to prove that $cost(B') \le (2\rho - 1)\delta(\Phi)$. Since $|U_1(B)| \le \rho\beta_1(\Phi)$, by Lemma 5.3, we have $|U_1(B')| \le \rho\beta_1(\Phi)$ and $|U_2(B')| = |\Sigma_C|$. The cost of B' is $cost(B') = |U(B')| - |\Sigma| = |U_1(B')| - |\Sigma_N| \le \rho\beta_1(\Phi) - |\Sigma_N|$. By Lemma 5.4, $\beta(\Phi) = \beta_1(\Phi) + |\Sigma_C|$ and thus $\delta(\Phi) = \beta(\Phi) - |\Sigma| = \beta_1(\Phi) - |\Sigma_N|$. As showed in the proof of Theorem 5.7, $\beta_1(\Phi) \ge 2|\Sigma_N|$. Thus, $\delta(\Phi) = \beta_1(\Phi) - |\Sigma_N| \ge 0.5\beta_1(\Phi)$. Consequently,

$$\begin{aligned} cost(B') &\le \rho\beta_1(\Phi) - |\Sigma_N| \\ &= (\beta_1(\Phi) - |\Sigma_N|) + (\rho - 1)\beta_1(\Phi) \\ &\le \delta(\Phi) + 2(\rho - 1)\delta(\Phi) \qquad \text{(since } \delta(\Phi) = \beta_1(\Phi) - |\Sigma_N| \text{ and } \delta(\Phi) \ge 0.5\beta_1(\Phi)\text{)} \\ &= (2\rho - 1)\delta(\Phi). \end{aligned}$$

Therefore, the lemma holds. ■

By Lemma 5.15, a ρ-approximate algorithm for $\beta(\Phi)$ is a $(2\rho - 1)$-approximation algorithm for $\delta(\Phi)$ if it is designed based on TWOPHASEUB. Therefore, the algorithm for Theorem 5.7 is a $(w(\Phi) - 1)$-approximation algorithm for $\delta(\Phi)$; and the algorithm for Theorem 5.13 is a $(0.736h(\Phi) + 0.266)$-approximation algorithm for $\delta(\Phi)$. We have the following.

Theorem 5.16. There is a $(w(\Phi) - 1)$-approximation algorithm for $\delta(\Phi)$ running in time $O(\max(m^{0.373}n^2, mn^{1.373})) + \tilde{O}(n^{2.373})$.

Theorem 5.17. There is a randomized approximation algorithm for $\delta(\Phi)$ requiring $\tilde{O}(\sigma mn^{13})$ time, where $\sigma > 0$ is a given parameter. With probability $1 - O(n^{-\sigma})$, the algorithm has an approximation factor of $0.736h(\Phi) + 0.266$.

By Lemma 5.15, the algorithm for Theorem 5.11 is an $h(\Phi)$-approximation algorithm for $\delta(\Phi)$ running in time $O(mn^2)$. In the following we give another $h(\Phi)$-approximation algorithm, which is more efficient in time.

Theorem 5.18. There is an $h(\Phi)$-approximation algorithm for $\delta(\Phi)$ running in time $O(\max(m^{0.373}n^2, mn^{1.373}))$.

Proof. Let B be the branching such that the parent of each vertex v is the smallest vertex strictly containing v (with ties broken arbitrarily). We claim that $cost(B) \le h(\Phi)\delta(\Phi)$. Consider an element $e \in \Sigma$. Recall that h_e is the size of a largest chain in V_e. Let C_e be a chain of size h_e such that $C_e \subseteq V_e$. Let $(u_1, u_2, ..., u_s)$ be the sequence of vertices obtained by arranging the vertices of C_e in increasing order of their sizes, where $s = h_e$. By the definitions of B and C_e, it is easy to see that $p_B(u_s) = \text{null}$ and $p_B(u_i) \ne \text{null}$ for $1 \le i < s$. Consider two vertices u_i, u_j, where $1 \le i < j < s$. Since $p_B(u_i)$ is the smallest vertex strictly containing u_i, we have $|p_B(u_i)| \le |u_j| < |p_B(u_j)|$. Thus, the B-parents of u_i and u_j are different. As a result, we know that no two vertices in C_e have the same B-parent. Therefore, B induces at least $s - 1$ distinct B-covered pairs $(e, p_B(u_1))$, $(e, p_B(u_2))$, ..., $(e, p_B(u_{s-1}))$. Hence, $|U_B(e)| \le |V_e| - (s - 1) = |V_e| - h_e + 1$ and thus,

$$cost(B) = \sum_{e\in\Sigma} (|U_B(e)| - 1) \le \sum_{e\in\Sigma} (|V_e| - h_e + 1 - 1) = \sum_{e\in\Sigma} (|V_e| - h_e).$$

Let B^* be an optimal branching of D_Φ. Recall that Hujdurović *et al.* [16] had the following observation: $h_e \cdot |U_{B^*}(e)| \ge |V_e|$ for each $e \in \Sigma$. From this observation, we know that

$$\sum_{e\in\Sigma} (|V_e| - h_e) \le \sum_{e\in\Sigma} (h_e \cdot |U_{B^*}(e)| - h_e) \le \sum_{e\in\Sigma} h(\Phi)(|U_{B^*}(e)| - 1) = h(\Phi) \cdot cost(B^*) = h(\Phi)\delta(\Phi).$$

Consequently, $cost(B) \le h(\Phi)\delta(\Phi)$ and hence B is an $h(\Phi)$-approximate solution for $\delta(\Phi)$. Given D_Φ, B can be easily constructed in $O(n^2)$ time. Therefore, the computation of B requires $O(\max(m^{0.373}n^2, mn^{1.373}))$ time, which completes the proof. ■

According to Theorem 2.1, we also obtain the following approximability results for $\varepsilon(M)$.

Theorem 5.19. The following approximation algorithms for $\varepsilon(M)$ exist, where Φ is a set family with M as its matrix representation:

(1) a $(w(\Phi) - 1)$-approximation algorithm running in time $O(mn^2) + \tilde{O}(n^{2.373})$,
(2) an $h(\Phi)$-approximation algorithm running in time $O(mn^2)$, and
(3) a $\tilde{O}(\sigma mn^{13})$-time randomized algorithm having an approximation factor of $0.736h(\Phi) + 0.266$ with probability $1 - O(n^{-\sigma})$, where $\sigma > 0$ is a given parameter.

6. Conclusion and future work

In this paper, based on new insights into the structure of UB, we gave the first polynomial size kernel for SR. The proposed kernelization algorithm can be used as a preprocessing procedure to speed up all existing exact algorithms for SR. We also proposed a two-phase approach for approximating $\alpha(M)$ and $\varepsilon(M)$. Using the approach, new approximation algorithms for $\alpha(M)$ and first approximation algorithms for $\varepsilon(M)$ were presented.

The proof of Theorem 3.10 shows the safeness of the following reduction rule, which is a generalization of Rules 1 and 2: Remove a vertex that has an ideal parent. Denote this rule by Rule 4. Let $t(\Phi)$ be the number of vertices remained in Φ if Rules 3 and 4 are exhaustively applied. Theorem 3.7 shows that $t(\Phi) \leq 4\delta(\Phi) - 1$. An improvement on this upper bound immediately leads to a kernel of smaller size. An open problem is to determine whether this upper bound is tight. A variant of SR, called the *distinct split-row problem* (DSR), is as follows: perform split-row operations to transform a given matrix so that the result is conflict-free and has the minimum number of distinct rows. DSR is APX-complete [16], 2-approximable [16], and fixed-parameter tractable with respect to $\varepsilon(M)$ [27]. It would be interesting to study kernelization algorithms for DSR. Other possible directions for future research include: (1) determine whether $\varepsilon(M)$ admits a constant-factor approximation algorithm; (2) improve Sheu and Wang's FPT-time algorithm for SR; and (3) improve the 2-approximation algorithm for DSR.

ACKNOWLEDGMENTS

This research was supported by the National Science and Technology Council of the Republic of China under grant NSTC-112-2221-E-007-047-MY3.

REFERENCES

[1] A. V. Aho, M. R. Garey, and J. D. Ullman, "The transitive reduction of a directed graph," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 131–137, 1972.
[2] P. J. Campbell, E. D. Pleasance, P. J. Stephens, E. Dicks, R. Rance, I. Goodhead, G. A. Follows, A. R. Green, P. A. Futreal, and M. R. Stratton, "Subclonal phylogenetic structures in cancer revealed by ultra-deep sequencing," in *Proceedings of the National Academy of Sciences*, vol. 105, no. 35, pp. 13081–13086, 2008.
[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms* (3rd ed.), The MIT Press, 2009.
[4] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms* (1st ed.), Springer Publishing Company, Incorporated, 2015.
[5] S. Deep, X. Hu, and P. Koutris, "Fast join project query evaluation using matrix multiplication," In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (*SIGMOD '20*), pp. 1213–1223, 2020.
[6] M. El-Kebir, L. Oesper, H. Acheson-Field, and B. J. Raphael, "Reconstruction of clonal trees and tumor composition from multi-sample sequencing data," *Bioinformatics*, vol. 31, no. 12, pp. i62–i70, 2015.
[7] D. Fernández-Baca, "The perfect phylogeny problem," in *Steiner Trees in Industry*, Springer US, pp. 203–234, 2001.
[8] D. Fernández-Baca and J. Lagergren, "A polynomial-time algorithm for near-perfect phylogeny," *SIAM Journal on Computing*, vol. 32, pp. 1115–1127, 2003.
[9] Y. Filmus and J. Ward, "Monotone submodular maximization over a matroid via non-oblivious local search," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 514–542, 2014.
[10] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for maximizing submodular set functions. II.," in Polyhedral Combinatorics: Dedicated to the Memory of D.R. Fulkerson, Springer, pp. 73–87, 1978.

[11] F. L. Gall, "Powers of tensors and fast matrix multiplication," In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation* (*ISSAC '14*), pp. 296–303, 2014.
[12] D. Gusfield, "Efficient algorithms for inferring evolutionary trees," *Networks*, vol. 21, no. 1, pp. 19–28, 1991.
[13] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press, 1997.
[14] I. Hajirasouliha, A. Mahmoody, and B. J. Raphael, "A combinatorial approach for analyzing intra-tumor heterogeneity from high-throughput sequencing data," *Bioinformatics*, vol. 30, no. 12, pp. i78–i86, 2014.
[15] I. Hajirasouliha and B. J. Raphael, "Reconstructing mutational history in multiply sampled tumors using perfect phylogeny mixtures," in *Proceedings of the 14th International Workshop on Algorithms in Bioinformatics*, pp. 354–367, 2014.
[16] A. Hujdurović, E. Husić, M. Milanič, R. Rizzi, and A. I. Tomescu. "Perfect phylogenies via branchings in acyclic digraphs and a generalization of Dilworth's theorem," *ACM Transactions on Algorithms*, vol. 14, no. 2, Article 20, 26 pages, 2018.
[17] A. Hujdurović, U. Kacar, M. Milanič, B. Ries, and A. I. Tomescu, "Complexity and algorithms for finding a perfect phylogeny from mixed tumor samples," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, no. 1, pp. 96–108, 2018.
[18] E. Husić, X. Li, A. Hujdurović, M. Mehine, R. Rizzi, V. Mäkinen, M. Milanič, and A. I. Tomescu, "MIPUP: minimum perfect unmixed phylogenies for multi-sampled tumors via branchings and ILP," *Bioinformatics*, vol. 35, no. 5, pp. 769–777, 2019.
[19] W. M. Ismail, E. Nzabarushimana, and H. Tang, "Algorithmic approaches to clonal reconstruction in heterogeneous cell populations," *Quantitative Biology*, vol. 7, no. 4, pp. 255–265, 2019.
[20] W. Jiao, S. Vembu, A. G. Deshwar, L. Stein, and Q. Morris, "Inferring clonal evolution of tumors from single nucleotide somatic mutations," *BMC Bioinformatics*, vol. 15, suppl. 1, pp. 35, 2014.
[21] S. Kannan and T. Warnow, "A fast algorithm for the computation and enumeration of perfect phylogenies," *SIAM Journal on Computing*, vol. 26, no. 6, pp. 1749–1763, 1997.
[22] S. Malikic, A. W. McPherson, N. Donmez, and C. S. Sahinalp, "Clonality inference in multiple tumor samples using phylogeny," *Bioinformatics*, vol. 31, no. 9, pp. 1349–1356, 2015.
[23] D. E. Newburger, D. Kashef-Haghighi, Z. Weng, R. Salari, R. T. Sweeney, A. L. Brunner, S. X. Zhu, X. Guo, S. Varma, M. L. Troxell, R. B. West, S. Batzoglou, and A. Sidow, "Genome evolution during progression to breast cancer," *Genome Research*, vol. 23, no. 7, pp. 1097–1108, 2013.
[24] S. Nik-Zainal, P. Van Loo, D. C. Wedge, L. B. Alexandrov, C. D. Greenman, K. W. Lau, K. Raine, D. Jones, J. Marshall, M. Ramakrishna, A. Shlien, S. L. Cooke, J. Hinton, A. Menzies, L. A. Stebbings, C. Leroy, M. Jia, R. Rance, L. J. Mudie, S. J. Gamble, P. J. Stephens, S. McLaren, P. S. Tarpey, E. Papaemmanuil, H. R. Davies, I. Varela, D. J. McBride, G. R. Bignell, K. Leung, A. P. Butler, J. W. Teague, S. Martin, G. Jönsson, O. Mariani, S. Boyault, P. Miron, A. Fatima, A. Langerod, S. A. J. R. Aparicio, A. Tutt, A. M. Sieuwerts, Å. Borg, G. Thomas, A. V. Salomon, A. L. Richardson, A. L. Borresen-Dale, P. A. Futreal, M. R. Stratton, and P. J. Campbell, "The life history of 21 breast cancers," *Cell*, vol. 149, no. 5, pp. 994–1007, 2012.
[25] I. Pe'er, R. Shamir, and R. Sharan, "Incomplete directed perfect phylogeny," *SIAM Journal on Computing*, vol. 33, no. 3, pp. 590–607, 2000.
[26] V. Popic, R. Salari, I. Hajirasouliha, D. Kashef-Haghighi, R. B. West, and S. Batzoglou, "Fast and scalable inference of multi-sample cancer lineages," *Genome Biology*, vol. 16, no. 1, pp. 91, 2015.
[27] W.-H. Sheu and B.-F. Wang, " Parameterized complexity for finding a perfect phylogeny from mixed tumor samples," *SIAM Journal on Discrete Mathematics*, vol.37, iss. 3, pp. 2049-2071, 2023.
[28] F. Strino, F. Parisi, M. Micsinai, and Y. Kluger, "TrAp: a tree approach for fingerprinting subclonal tumor composition," *Nucleic Acids Research*, vol. 41, no. 17, pp. e165, 2013.

APPENDIX

A. Tightness of the upper bound on the kernel size

Let d be a power of 2. To show that the upper bounds in Theorems 3.6 and 3.7 are tight, in the following, we construct a set family Φ_d over a ground set Σ_d such that $|\Phi_d| = 4d - 1$, $|\Sigma_d| = 3d$, $\delta(\Phi_d) = d$, and none of the three reduction rules is applicable. The ground set Σ_d is $\{a_1, a_2, ..., a_d, b_1, b_2, ..., b_{2d}\}$. Let T_d be a complete

binary tree of $2d$ leaves, in which the edges are directed from a child to its parent and each node represents a set as follows:

(R1) The first half of the leaves represent, respectively, $\{a_1, b_1\}$, $\{a_2, b_2\}$, ..., $\{a_d, b_d\}$; and the second half of the leaves represent, respectively, $\{a_1, b_{d+1}\}$, $\{a_2, b_{d+2}\}$, ..., $\{a_d, b_{2d}\}$.
(R2) Each internal node represents the union of its two children.

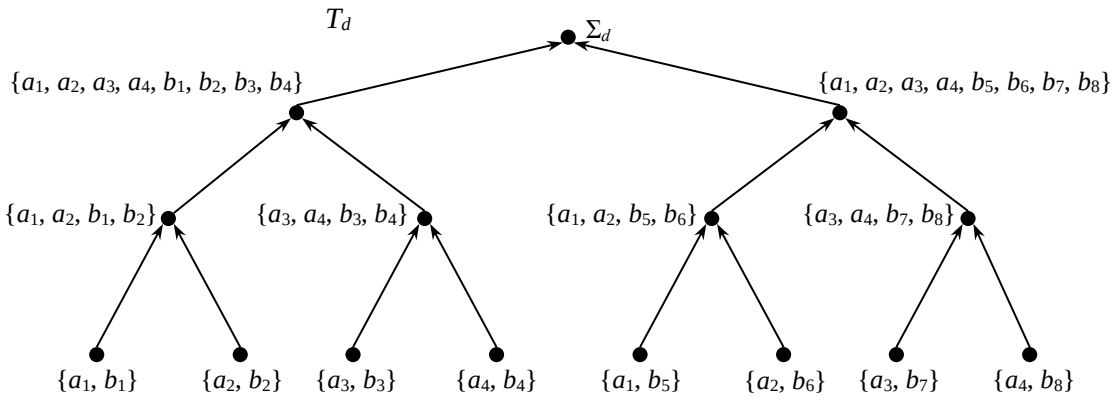


Figure 6. An illustrative example, in which $d = 4$.

See Figure 6 for an example of $d = 4$. The set family Φ_d is defined as $\{v \mid v \text{ is a node of } T_d\}$. Note that if only elementary arcs are displayed, the figure of D_{Φ_d} is the same as T_d. It is easy to check the following: each non-root vertex is in conflict with some vertex in Φ_d; each element a_i is non-chain-like; each element b_i is chain-like, but not doubly chain-like; and for each chain-like element b_i, $t^-(b_i)$ is not a vertex in Φ_d. As a result, none of Rules 1, 2, and 3 is applicable. We proceed to show that $\delta(\Phi_d) = d$. By Lemma 3.3, for any branching B, $|U_B(e)| \geq 2$ if e is non-chain-like and $|U_B(e)| \geq 1$ if e is chain-like. Since there are d non-chain-like elements and $2d$ chain-like elements, we have $\beta(\Phi_d) \geq 2 \times d + 1 \times 2d \geq 4d$. Let B_d be the edge set of T_d. Since $(\Phi_d, B_d) = T_d$ is a tree, B_d is a branching of D_{Φ_d}. Observe that in the branching B_d, the elements in a node v are all covered if v is an internal node of T_d and are all uncovered otherwise. Thus, $|U(B_d)| = 2 \times 2d = 4d$. Since $\beta(\Phi_d) \geq 4d$, we know that B_d is an optimal branching and $\beta(\Phi_d) = 4d$. Therefore, $\delta(\Phi_d) = \beta(\Phi_d) - |\Sigma_d| = 4d - 3d = d$.

B. Time complexity analysis for Theorem 5.14

Suppose that the input matroid is $(A, \mathcal{B})$ and the objective function is g, Fisher *et al.*'s algorithm for SUBM finds a solution as follows. First, initialize two sets $B = \emptyset$ and $N = A$. Then, repeat the following until N is empty: select an arc $e \in N$ that maximizes $g(B \cup \{e\}) - g(B)$; update N as $N - \{e\}$; and if $B \cup \{e\} \in \mathcal{B}$, add e into B. Finally, return B as a solution.

We implement the above algorithm as follows. During the algorithm, for each vertex $v \in \Phi$, we maintain a set $X(v)$, which stores the set of B-uncovered elements in v; for each arc $(u, v) \in N$, we maintain a value $gain(u, v) = g(B \cup \{(u, v)\}) - g(B)$, which is equal to $|X(v) \cap u \cap \Sigma_N|$. Initially, $X(v) = v$ for each $v \in \Phi$; $gain(u, v) = |u \cap \Sigma_N|$ for each $(u, v) \in N$; and the set N is represented by $m + 1$ linked list $L(0)$, $L(1)$, ..., $L(m)$, where $L(i)$ stores the set of $(u, v) \in N$ with $gain(u, v) = i$. Each iteration of Fisher *et al.*'s algorithm is done as follows. First, find an arc $(u, v) \in N$ that maximizes $gain(u, v)$. If u has a B-parent, we know that $B \cup \{(u, v)\}$ is not a branching and hence, (u, v) is simply discarded. Otherwise, we do the following: add (u, v) into B; update $X(v)$ as $X(v) - u$; and, for each $(c, v) \in N$, recalculate $gain(c, v) = |X(v) \cap c \cap \Sigma_N|$ and then move (c, v) to $L(gain(c, v))$.

The initialization of B, $X(\cdot)$, $gain(\cdot, \cdot)$, and $L(\cdot)$ can be easily done in $O(mn + n^2)$ time. Consider a fixed iteration. An arc $(u, v) \in N$ maximizing $gain(u, v)$ is found in $O(m)$ time by checking the size of each $L(i)$. Whether u has a B-parent can be determined $O(1)$ time. Suppose that (u, v) is added to B. The update of B and $X(v)$ requires $O(m)$ time. Since v has at most n incoming arcs in D_Φ, at most n $gain(\cdot, v)$ need to be

recalculated, which requires $O(mn)$ time in total. In summary, if (u, v) is not added into B, $O(m)$ time is required; otherwise, $O(mn)$ time is required. There are $|A| = O(n^2)$ iterations, in which at most $n - 1$ iterations add an arc into B. Therefore, the time for all iterations is $O(mn^2)$. As a result, the overall time complexity is $O(mn + n^2 + mn^2) = O(mn^2)$.