Efficient Parameterized Algorithms for Computing All-Pairs Shortest Paths

Stefan Kratsch
Humboldt-Universität zu Berlin, Germany
kratsch@informatik.hu-berlin.de

Florian Nelles
Humboldt-Universität zu Berlin, Germany
nelles@informatik.hu-berlin.de

Abstract

Computing all-pairs shortest paths is a fundamental and much-studied problem with many applications. Unfortunately, despite intense study, there are still no significantly faster algorithms for it than the $\mathcal{O}(n^3)$ time algorithm due to Floyd and Warshall (1962). Somewhat faster algorithms exist for the vertex-weighted version if fast matrix multiplication may be used. Yuster (SODA 2009) gave an algorithm running in time $\mathcal{O}(n^{2.842})$, but no combinatorial, truly subcubic algorithm is known.

Motivated by the recent framework of efficient parameterized algorithms (or “FPT in P”), we investigate the influence of the graph parameters clique-width (cw) and modular-width (mw) on the running times of algorithms for solving ALL-PAIRS SHORTEST PATHS. We obtain efficient (and combinatorial) parameterized algorithms on non-negative vertex-weighted graphs of times $\mathcal{O}(\mathsf{cw}^2\, n^2)$, resp. $\mathcal{O}(\mathsf{mw}^2\, n + n^2)$. If fast matrix multiplication is allowed then the latter can be improved to $\mathcal{O}(\mathsf{mw}^{1.842}\, n + n^2)$ using the algorithm of Yuster as a black box. The algorithm relative to modular-width is adaptive, meaning that the running time matches the best unparameterized algorithm for parameter value mw equal to n, and they outperform them already for $\mathsf{mw} \in \mathcal{O}(n^{1-\varepsilon})$ for any $\varepsilon > 0$.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Shortest paths

Keywords and phrases All-pairs shortest Paths, efficient parameterized Algorithms, parameterized Complexity, Clique-width, Modular-width

Digital Object Identifier 10.4230/LIPIcs.STACS.2020.38

Related Version A full version of the paper is available at [17], `http://arxiv.org/abs/2001.04908`.

1 Introduction

ALL-PAIRS SHORTEST PATHS (APSP) is a fundamental and much-studied problem in the field of algorithmic graph theory. Next to the theoretical interest in the problem, ALL-PAIRS SHORTEST PATHS is important for many practical applications, e.g., it is closely related to several vertex centrality measures in networks (for example, the betweenness centrality of a vertex v is defined as the sum of the fraction of all-pairs shortest paths that pass through v). The ALL-PAIRS SHORTEST PATHS problem is also considered as the core of many routing problems and has applications for example in areas such as routing protocols, driving direction on web mappings, transportation, and traffic assignment problems, and many more. See also the survey of Susmita [23] for more applications.

Despite the large interest in ALL-PAIRS SHORTEST PATHS, there are only small improvements known since the well-known $\mathcal{O}(n^3)$-time algorithm by Floyd and Warshall [8, 26] from 1962: Chan [3] as well as Han and Takaoka [12] gave an algorithm running in $\mathcal{O}(n^3/\log^2 n)$ (omitting $poly(\log\log n)$ factors) and Williams [27] gave an randomized algorithm running in time $\mathcal{O}(n^3/2^{\Omega(\log n)^{1/2}})$. While there are no unconditional lower bounds known, it has been conjectured that there is no truly subcubic algorithm for ALL-PAIRS SHORTEST PATHS, i.e., that no algorithm achieves time $\mathcal{O}(n^{3-\varepsilon})$ for any $\varepsilon > 0$. Using suitable subcubic reductions, this is tightly connected to the existence of subcubic algorithms for several network centrality measures, finding a directed triangle of negative total edge length, finding the second shortest simple path between two nodes in an edge-weighted graph, or checking if a given matrix defines a metric. This means that if one of those problems can be solved in truly subcubic time (i.e., can be solved in time $\mathcal{O}(n^{3-\varepsilon}) \cdot poly(\log M)$ for an $\varepsilon > 0$ and weights in $[-M, M]$ for weighted problems), then all of the problems admit algorithms with truly subcubic running time [28]. The situation is different for VERTEX-WEIGHTED ALL-PAIRS SHORTEST PATHS: While it is conjectured that there is no truly subcubic *combinatorial* algorithm, faster algorithms are known if fast matrix multiplication may be used. The currently fastest algorithm is due to Yuster [29] and runs in time $\mathcal{O}(n^{2.842})$. For sparse graphs there is an algorithm running in time $\mathcal{O}(nm + n^2 \log\log n)$ for directed graphs [20] and an algorithm for undirected graphs [21] with a running time of $\mathcal{O}(mn \log \alpha(m, n))$, where α is the inverse of the Ackermann's function.

Independently of whether one believes in conditional lower bounds and hypotheses, the fact remains that we do not know any truly subcubic algorithms for ALL-PAIRS SHORTEST PATHS nor truly subcubic, combinatorial algorithms for the vertex-weighted case. Besides heuristics or approximation algorithms, one possible solution for faster algorithms for at least *some* input graphs is to exploit *structure* in the input graph. In addition to measuring the complexity of a problem relative to the input size of a graph (number n of vertices and number m of edges), one may additionally consider some *parameter*, say k, that quantifies structure that may be exploited by an algorithm; i.e., we may study the *parameterized complexity* of the problem. This framework typically aims at NP-hard problems and a key goal is to obtain *fixed-parameter tractable* (FPT) algorithms that run in time $f(k)n^c$ for some constant c and some (usually exponential) function $f(k)$ of the parameter. Initiated by the work of Giannopoulou et al. [11], also efficient parameterized algorithms for tractable problems are considered (apart from many older results that predate even parameterized complexity). In this framework, also called "FPT in P", one is interested in running times $\mathcal{O}(k^\alpha n^\beta)$ when the best dependence on the input size alone is $\mathcal{O}(n^\gamma)$ with $\gamma > \beta$, which then results in a better running time for sufficiently small parameter k. Typically, the parameter k is at most n, thus, in the case of $\alpha + \beta = \gamma$, one already achieves truly better running times for $k \in o(n)$. We call such algorithms, which even for $k = n$ are not worse than the best unparameterized algorithm, *adaptive* algorithms.

Several recent publications dealt with efficient parameterized algorithms for different problems and parameters [9, 14, 4, 1, 13, 19], however, ALL-PAIRS SHORTEST PATHS got very little attention. Coudert et al. [4] considered the *clique-width* of a graph as a parameter for tractable problems related to cycle problems. Intuitively, clique-width captures the closeness of a graph to a cograph, with cographs being exactly the graphs of clique-width at most two. Alongside some positive results for TRIANGLE COUNTING or GIRTH, they proved a conditional lower bound for DIAMETER namely that there is no $\mathcal{O}(2^{o(\mathsf{cw})} \cdot n^{2-\varepsilon})$ time algorithm for any $\varepsilon > 0$. That is, even computing just the greatest length of any shortest path in an unweighted graph admits no such algorithm. A weaker parameter and an upper bound for clique-width is the modular-width of a graph, which is another parameter that has been previously studied regarding its use for efficient parameterized algorithms [4, 16].

Note that small clique-width or small modular-width does not imply the sparsity of the graph, e.g. cliques have clique-width and modular-width two. For parameters that do imply the sparsity of the graph (meaning that for parameter value k, the number of edges is bounded by $\mathcal{O}(kn)$, where n denotes the number of vertices in the graph), the algorithm of Pettie and Ramachandran directly yield a running time of $\mathcal{O}(kn^2 + n^2 \log\log n)$, which is nearly optimal.

Our work. We study efficient parameterized algorithms for ALL-PAIRS SHORTEST PATHS for its vertex-weighted variant. We consider the structural parameters *clique-width* (cw) and *modular-width* (mw). As our main result, we present an $\mathcal{O}(\mathsf{cw}^2\, n^2)$-time algorithm for VERTEX-WEIGHTED ALL-PAIRS SHORTEST PATHS, yielding a truly subcubic algorithm for $\mathsf{cw} \in \mathcal{O}(n^{0.5-\varepsilon})$. This immediately allows to solve the DIAMETER problem in the same asymptotic time $\mathcal{O}(\mathsf{cw}^2\, n^2)$, even with vertex weights, and thereby nicely complements the lower bound ruling out $\mathcal{O}(2^{o(\mathsf{cw})} \cdot n^{2-\varepsilon})$ for any $\varepsilon > 0$ [4].

Further, we present a general framework to determine the running time for many algorithms that use modular-width and the related modular decomposition tree. We use this framework to prove an algorithm of time $\mathcal{O}(\mathsf{mw}^2\, n + n^2)$ for VERTEX-WEIGHTED ALL-PAIRS SHORTEST PATHS on graphs of modular-width at most mw. This algorithm is combinatorial, however, it can benefit from subcubic algorithms for VERTEX-WEIGHTED ALL-PAIRS SHORTEST PATHS that use fast matrix multiplication. For example, we achieve a running time of $\mathcal{O}(\mathsf{mw}^{1.842}\, n + n^2)$ by using an $\mathcal{O}(n^{2.842})$-time algorithm for the vertex-weighted case by Yuster [29] in each prime node; this algorithm uses fast matrix multiplication whereas all other algorithms (previous and new) are combinatorial.

Related Work. Following the work of Floyd and Warshall [8, 26], Fredman [10] achieved the first subcubic algorithm, running in time $\mathcal{O}(n^3 \log^{1/3} \log n / \log^{1/3} n)$. Chan [3] and Han and Takaoka [12] both achieved a running time of $\mathcal{O}(n^3/\log^2 n)$ (omitting $poly(\log\log n)$ factors). Recently, Williams [27] solved APSP in randomized time $\mathcal{O}(n^3/2^{\Omega(\log n)^{1/2}})$. For sparse graphs, Pettie and Ramachandran [21] get a running time of $\mathcal{O}(n^2\alpha(n,m) + mn)$. All these algorithms solve the standard edge-weighted case.

In the vertex-weighted case, the currently fastest algorithm by Yuster [29] runs in $\mathcal{O}(n^{2.842})$ and relies on fast matrix multiplication. Shapira et al. [22] considered some variants of APSP, namely the ALL-PAIRS BOTTLENECK PATHS, where one seeks the maximum bottleneck weight on a graph, and provided an algorithm of time $\mathcal{O}(n^{2.575})$ for vertex-weighted graphs. Czumaj and Lingas [6] analyzed the related problem of finding the minimum-weight triangle in vertex-weighted graphs and achieved a running time of $\mathcal{O}(n^\omega + n^{2+o(1)})$. All of these algorithms for vertex-weighted graphs exploit fast matrix multiplication. There is no truly subcubic *combinatorial* algorithm known for VERTEX-WEIGHTED ALL-PAIRS SHORTEST PATHS.

There are some subcubic algorithms known for APSP on special graph classes, such as uniform disk graphs with non-negative vertex weights, induced by point sets of bounded density within a unit square. Lingas and Sledneu [18] showed how to solve APSP on such graphs in time $\mathcal{O}(\sqrt{r}n^{2.75})$, where r is the radius of the disk around the vertices in a unit square. Bentert and Nichterlein [2] considered the related problem of computing the diameter of a graph, parameterized by several parameters.

Organization. Section 2 contains the preliminaries, in particular, the definition of clique-width. In Section 3, we present the algorithm for VERTEX-WEIGHTED ALL-PAIRS SHORTEST PATHS parameterized by the clique-width. Due to space restrictions, most of the proofs, the algorithm for modular-width as well as the running time framework can only be found in the full version of this paper [17]. We conclude in Section 4.

2 Preliminaries

We follow basic graph notations [7]. For a natural number $k \in \mathbb{N}$, define $[k] = \{1, \dots, k\}$. All graphs are simple, i.e., without loops or multiple edges. In a graph $G = (V, E)$, a path $P = (v_1, v_2, \dots, v_n)$ is a sequence of vertices $v_i \in V$ with $\{v_i, v_{i+1}\} \in E$ for $i \in [n-1]$. We define by $P_{[v_i,v_j]}$ the subpath of P starting in v_i and ending in v_j for $i, j \in [n]$ with $i < j$. The *length* of a path is the number of edges in it. In a vertex-weighted graph $G = (V, E)$ with weights $\omega \colon V \to \mathbb{R}_{\geq 0}$, the *weight* (also called *cost*) of a path P is defined as $\omega(P) = \sum_{i=1}^{n} \omega(v_i)$. Thus, every paths between two distinct vertices u and v has minimum weight $\omega(v) + \omega(u)$ and a path of length 0 from a vertex v to itself has always weight $\omega(v)$. For a graph $G = (V, E)$ and $u, v \in V$, we denote the minimum weight of all paths between u and v as $dist_G(u, v)$. For a set of vertices $X \subseteq V$ and a vertex $u \in V$ we define $dist_G(u, X) = \min_{v \in X} dist_G(u, v)$ and for two sets of vertices $X, Y \subseteq V$, we define $dist_G(X, Y) = \min_{u \in X, v \in Y}(u, v)$.

For two sets A and B we denote the disjoint union by $A \dot{\cup} B$ and we say that two sets A and B *overlap* if $A \cap B \neq \emptyset$, $A \setminus B \neq \emptyset$, and $B \setminus A \neq \emptyset$.

2.1 Clique-width and NLC-width

A *k-labeled* graph is a graph in which each vertex is assigned one out of k labels. Formally, a vertex-labeled graph G is a triple (V, E, lab) with V being the vertex set, E denotes the set of edges, and $lab : V \to [k]$ is a function that defines the label for each vertex. For a k-labeled graph $G = (V, E, lab)$ we denote by $unlab(G) = (V, E)$ the underlying unlabeled graph. Intuitively, a graph G has clique-width at most k, if it is the underlying graph of some k-labeled graph that can be constructed by using four operations: (1) Introducing a single labeled vertex, (2) redefining one label to another label, (3) taking the disjoint union of two already created k-labeled graphs, and (4) adding all edges between vertices of label i to vertices of label j for a pair (i, j) of labels.

▶ **Definition 1** (Clique-width, [5]). *Let $k \geq 2$. The class CW_k consists of all k-labeled graphs that can be constructed by the following operations:*

- *The nullary operation $\bullet_a$, that corresponds to a graph consisting of a single vertex with a label $a \in [k]$.*
- *Let $G = (V, E, lab) \in CW_k$ be a k-labeled graph, and let $a, b \in [k]$. Then*

$$\rho_{a,b}(G) = (V, E, lab') \qquad \text{with } lab'(v) = \begin{cases} lab(v) & , \text{ if } lab(v) \neq a \\ b & , \text{ if } lab(v) = a \end{cases}$$

  is in CW_k.
- *Let $G = (V_G, E_G, lab_G) \in CW_k$ and $H = (V_H, E_H, lab_H) \in CW_k$ be two k-labeled graphs in CW_k with $V_G \cap V_H = \emptyset$. Then the disjoint union, defined by*

$$G \oplus H = (V_G \dot{\cup} V_H, E_G \dot{\cup} E_H, lab') \qquad \text{with } lab'(v) = \begin{cases} lab_G(v) & , \text{ if } v \in V_G \\ lab_H(v) & , \text{ if } v \in V_H \end{cases}$$

  is in CW_k.
- *Let $G = (V, E, lab) \in CW_k$ be a k-labeled graph, and let $a, b \in [k]$ with $a \neq b$. Then*

$$\eta_{a,b}(G) = (V, E', lab) \qquad \text{with } E' = E \cup \{\{u, v\} \mid lab(u) = a, lab(v) = b\}$$

  is in CW_k.

The clique-width of a graph G, denoted by $cw(G)$, is the smallest $k \geq 2$ such that there is a labeled graph $G' \in CW_k$ with $unlab(G') = G$. The expression consisting of the operations defined in Definition 1 is called a (clique-width) k-expression. For a k-expression t, we denote with $val(t)$ the resulting labeled graph and by $tree(t)$ the so called k-expression tree of t, which is the canonical tree representation of t. Clique-width is a strict generalization of modular-width, which is defined in the appendix. In fact, the clique-width of a graph G is equal to the maximum clique-width of any quotient graph of a prime node in the modular decomposition tree of G. On the other hand, modular-width cannot be bounded by a function of clique-width.

Very similar to clique-width, one can define NLC-width, which was introduced by Wanke [25]. The main differences are that the join operation η and the disjoint union operation $\oplus$ are somewhat combined and consecutive relabel operations are compressed into one operation.

▶ **Definition 2** (NLC-width). *Let $k \geq 1$. The class NLC_k consists of all k-labeled graphs that can be constructed by the following operations:*

- *The nullary operation $\bullet_a$, that corresponds to a graph consisting of a single vertex with a label $a \in [k]$.*
- *Let $G = (V, E, lab) \in NLC_k$ and let $R\colon [k] \to [k]$. Then*

  $$\circ_R(G) = (V, E, lab') \qquad \text{with } lab'(v) = R(lab(v))$$

  is in NLC_k.
- *Let $G = (V_G, E_G, lab_G) \in NLC_k$ and $H = (V_H, E_H, lab_H) \in NLC_k$ be two k-labeled graphs in NLC_k. Let $S \subseteq [k]^2$. Then*

  $$G \times_S H = (V_G \cup V_H, E', lab') \qquad \text{with } lab'(v) = \begin{cases} lab_G(v) & , \text{ if } v \in V_G \\ lab_H(v) & , \text{ if } v \in V_H \end{cases}$$
  $$\text{and } E' = E_G \cup E_H \cup \{\{u, v\} \mid u \in V_G, v \in V_H, \text{ and } (lab_G(u), lab_H(v)) \in S\}$$

  is in NLC_k.

The NLC-width of a graph G, denoted by $nlc(G)$, is the smallest $k \geq 2$ such that there is a labeled graph $G' \in NLC_k$ with $unlab(G') = G$. As for clique-width, the expression consisting of the operations defined in Definition 2 is called a (NLC-width) k-expression. For a k-expression t, we again denote with $val(t)$ the resulting labeled graph and by $tree(t) = T$ canonical tree representation of t, the so called k-expression tree of t. This means each leaf node of T is marked with $\bullet_a$ for some $a \in [k]$ and each internal node is either marked with $\circ_R$ for some $R\colon [k] \mapsto [k]$ or with $\times_S$ for some $S \subseteq [k]^2$, according to the operations defined in Definition 1 resp. Definition 2. For a node $x \in V(T)$ we denote by G^x the labeled graph defined by the k-expression represented by the subtree of T rooted in x and we define by $L_i^x = \{v \in V(G^x) \mid lab(v) = i\}$ the set of vertex in G^x with label $i \in [k]$. For a node $x \in V(T)$, we will use the shortcut $dist^x(u, v) := dist_{G^x}(u, v)$ to denote the distance between two vertices u and v in G^x.

The following lemma shows that we can safely focus on NLC k-expression trees, since the NLC-width and clique-width only differs by a factor of two at most.

▶ **Lemma 3** ([15]). *For any graph G it holds that $nlc(G) \leq cw(G) \leq 2 \cdot nlc(G)$.*

3 APSP parameterized by clique-width

Assuming SETH, one cannot solve DIAMETER (and thus, unweighted ALL-PAIRS SHORTEST PATHS) in time $2^{o(\mathsf{cw})} \cdot n^{2-\varepsilon}$ [4]. In this section, we show how to solve VERTEX-WEIGHTED ALL-PAIRS SHORTEST PATHS in time $\mathcal{O}(\mathsf{cw}^2\, n^2)$.

▶ **Theorem 4.** *For every graph $G = (V, E)$, given together with a clique-width k-expression and vertex weights $\omega\colon V \to \mathbb{R}_{\geq 0}$, VERTEX-WEIGHTED ALL-PAIRS SHORTEST PATHS can be solved in time $\mathcal{O}(k^2 n^2)$.*

For an input graph $G = (V, E)$, given together with a clique-width k-expression for some $k \geq 2$, we transform in a first step the clique-width k-expression to an NLC-width k-expression in linear time as described for example in [15]. For the rest of this section, by writing k-expression we always refer to an NLC-width k-expression instead of a clique-width k-expression. We interpret the (NLC-width) k-expression as a k-expression tree T, in which each node $v \in V(T)$ is marked with an operation of the k-expression that is applied to the children of v. Accordingly, T has exactly n leafs, each marked with an operation $\bullet_i$ for $i \in [k]$, and exactly $n - 1$ nodes marked with an operation $\times_S$ for some $S \subseteq [k]^2$. For ease of presentation, we assume that there is exactly one node marked with an operation $\circ_R$ for some $R\colon [k] \to [k]$ in between any two nodes marked with $\times_S$ (using $R(i) = i$ when no actual relabeling is necessary), hence, the length of the k-expression is $\mathcal{O}(n)$. Note, that the length of a clique-width k-expression is $\mathcal{O}(n + m)$ in general. For a node $x \in V(T)$ we denote by G^x the labeled graph that is defined by the subexpression tree of T rooted in x.

The algorithm consists of three phases. In the first phase, we traverse T in a bottom-up manner: For each node $x \in V(T)$ we partition the vertex set into sets of same-labeled vertices and compute the shortest distance for each single vertex to (the closest vertex in) each label set. Additionally, we compute the distance between each pair of label sets, i.e., the shortest distance of two vertices of the respective sets. Note, however, that in the first phase we only consider for each each node $x \in V(T)$ the distances in the graph G^x. In the second phase, we perform a top-down traversal of the k-expression tree T and consider the whole graph G. Once we have computed the necessary values in phase one and two, we traverse T one last time and finally compute the shortest path distances between all pairs of vertices.

First Phase. For a node $x \in V(T)$, which corresponds to the k-labeled graph G^x, we define $L_i^x = \{v \in V(G^x) \mid \mathit{lab}(v) = i\}$ as the set of all vertices in G^x with label i. Note, that $\mathit{unlab}(G^x)$ is an induced subgraph of G for any $x \in V(T)$. We traverse T in a bottom-up manner and compute for each node $x \in V(T)$ and for all pairs $(i, j) \in [k]^2$ of labels the shortest distance between some vertex in L_i^x and some vertex in L_j^x. Additionally, we compute for any vertex $v \in V(G^x)$ and any label $i \in [k]$ the shortest distance from v to some vertex in L_i^x. To be precise, for a node $x \in V(T)$ we compute the following values:

$$c_{i,j}^x = \mathit{dist}^x(L_i^x, L_j^x) \qquad \text{for } i, j \in [k]$$
$$a_{v,i}^x = \mathit{dist}^x(v, L_i^x) \qquad \text{for } v \in V(G^x), i \in [k]$$

For nodes $x \in V(T)$ that are marked with $\times_S$ for some $S \subseteq [k]^2$ we need to compute some auxiliary values. Let y and z be the two children of x in T. This means that G^x consists of the disjoint union of G^y and G^z together with a full join between the vertex sets L_i^y and L_j^z for each $(i, j) \in S$. Thus, one can partition the vertex set of G^x into the $2k$ sets $\{L_1^y, \ldots, L_k^y, L_1^z, \ldots, L_k^z\} = \mathcal{L}^x$. For each pair $(A, B) \in \mathcal{L}^x \times \mathcal{L}^x$ of vertex sets, we compute the shortest distance between some vertex in A to some vertex of B. In addition, we compute the shortest distance between A and B with the constraint that either the first edge, the last edge, or the first and the last edge of the shortest path is an edge of a newly inserted full join defined by S. This achieves the effect that we additionally compute the shortest distance from (1) *all* vertices of A to *some* vertex of B, (2) from *some* vertex of A to *all* vertices of B, and (3) from *all* vertices of A to *all* vertices of B. Doing this, one can e.g. combine a path that ends at *some* vertex of A with a path that can start at *any* vertex of A. In the following, we will describe how to compute the required values for each of the three different types of nodes in the k-expression tree T.

For the base case, let x be a leaf of the k-expression tree T. Thus, the node $x \in V(T)$ is marked with $\bullet_\ell$ for some $\ell \in [k]$. This means that G^x consists of a single vertex v with label ℓ. In this case the following holds:

$$c^x_{i,j} = \begin{cases} \omega(v) & \text{if } i = j = \ell \\ \infty & \text{otherwise} \end{cases} \qquad \text{for } i, j \in [k]$$

$$a^x_{v,i} = \begin{cases} \omega(v) & \text{if } i = \ell \\ \infty & \text{otherwise} \end{cases} \qquad \text{for } v \in V(G^x), i \in [k]$$

Now, let $x \in V(T)$ be an internal node of the k-expression tree T marked with $\circ_R$ for some $R\colon [k] \to [k]$. Let $y \in V(T)$ be the unique child of x in T. Since we traverse T in a bottom-up manner, we have already computed the values $a^y_{v,i}$ for all $v \in V(G^y)$ and $i \in [k]$ and the values $c^x_{i,j}$ for all $i, j \in [k]$. Note, that $\mathit{unlab}(G^x) = \mathit{unlab}(G^y)$, which, in particular, implies that distances between vertices are identical in both graphs (though distances between label sets may be not, as these sets may be different).

▶ **Lemma 5.** *Let $x \in V(T)$ be an internal node of a k-expression tree T marked with $\circ_R$ for some $R\colon [k] \to [k]$ and let y be the child of x in T. Then $c^x_{i,j} = \min_{i' \in R^{-1}(i), j' \in R^{-1}(j)} c^y_{i',j'}$ for all $i, j \in [k]$.*

Note, that the computation of all $c^x_{i,j}$ can be realized in time $\mathcal{O}(k^2)$ by updating for every $c^y_{i,j}$ the corresponding value $c^x_{R(i),R(j)}$. The values $a^x_{v,i}$ can be similarly computed from the values at the child node:

▶ **Lemma 6.** *Let $x \in V(T)$ be an internal node of a k-expression tree T marked with $\circ_R$ for some $R\colon [k] \to [k]$ and let y be the unique child of x in T. Then $a^x_{v,i} = \min_{j \in R^{-1}(i)} a^y_{v,j}$ for all $v \in V(G^x)$ and $i \in [k]$.*

The running time for computing the values $a^x_{v,i}$ is $\mathcal{O}(nk)$ since we need to consider each value $a^y_{v,i}$ for any $v \in V(G^y)$ and $i \in [k]$ exactly once.

Finally, let $x \in V(T)$ be an internal node of the k-expression tree T marked with $\times_S$ for some $S \subseteq [k]^2$. Denote by $y \in V(T)$ and $z \in V(T)$ the two children of x in T, meaning that G^x combines the two labeled graphs G^y and G^z by introducing for each $(i, j) \in S$ a full join between the vertices in L^y_i and those in L^z_j. Thus, $V(G^x) = V(G^y) \dot\cup V(G^z)$ and one can partition the vertices of G^x into the $2k$ vertex sets $\{L^y_1, \ldots, L^y_k, L^z_1, \ldots, L^z_k\}$ and $L^x_i = L^y_i \dot\cup L^z_i$. To compute the desired distances between the label sets $\{L^y_1, \ldots, L^y_k, L^z_1, \ldots, L^z_k\}$, we construct an edge-weighted directed graph H^x that represents all the distances between the label sets in a graph with only $4k$ vertices.

For each label set L^a_i of G^x with $i \in [k]$ and $a \in \{y, z\}$ we create two vertices v^a_i and u^a_i. Let $V^a = \{v^a_i \mid i \in [k]\}$ resp. $U^a = \{u^a_i \mid i \in [k]\}$ for $a \in \{y, z\}$. We add a directed full join from V^y to U^y resp. from V^z to U^z with weight equal to the length of a shortest path between the two corresponding label sets. Finally, we connect vertices in U^y with vertices in

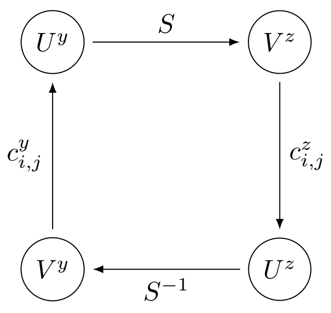

Figure 1 Construction of the auxiliary graph H^x. Each large node consists of k disjoint vertices corresponding to one of the k label sets in G^y resp. G^z. Between V^y and U^y there is a full join, each edge between the corresponding vertex of L^y_i and L^y_j is weighted by $c^y_{i,j}$, analogously for V^z and U^z. Vertices in U^y are only connected to those vertices in V^z for which the corresponding label sets are connected via S, analogously for U^z and V^y.

V^z, resp. vertices in U^z with vertices in V^y, if and only if the corresponding pair is contained in S, i.e., if there is a full join in G^x between the two corresponding label sets. See also Figure 1 for an illustration. Formally, we define the directed, edge-weighted graph H^x as follows.

▶ **Definition 7.** *Let $x \in V(T)$ be an internal node of a k-expression tree T marked with $\times_S$ for some $S \subseteq [k]^2$ and let $y \in V(T)$ and $z \in V(T)$ be the children of x. We define H^x as a directed, edge-weighted graph on $4k$ vertices created as follows:*

- *For each label set $L^a_i \in \{L^y_1, \ldots, L^y_k, L^z_1, \ldots, L^z_k\}$ we create two vertices v^a_i and u^a_i for $i \in [k]$ and $a \in \{y, z\}$.*
- *Add edges (v^y_i, u^y_j) with cost $c^y_{i,j}$ for all $i, j \in [k]$.*
- *Add edges (v^z_i, u^z_j) with cost $c^z_{i,j}$ for all $i, j \in [k]$.*
- *Add edges (u^y_i, v^z_j) with cost zero for all $(i, j) \in S$.*
- *Add edges (u^z_i, v^y_j) with cost zero for all $(j, i) \in S$.*

Note, that some edges may have cost ∞ as there is no path of the requested type exists. Next, we will see that H^x exhibits all the desired distances from G^x in a compact way.

▶ **Theorem 8.** *Let $x \in V(T)$ be an internal node of a k-expression tree T marked with $\times_S$ for some $S \subseteq [k]^2$ with children y and z. Let H^x be the graph as defined in Definition 7. Then the following holds:*

(1) *$dist_{H^x}(v^a_i, u^b_j) = dist^x(L^a_i, L^b_j)$ for all $a, b \in \{y, z\}$ and $i, j \in [k]$.*

(2) *$dist_{H^x}(u^a_i, u^b_j) = \min_{P \in \mathcal{P}} \omega(P) - \min_{v \in L^a_i} \omega(v)$ where $\mathcal{P}$ is the set of all paths in G^x starting in L^a_i, ending in L^b_j, and having the second vertex in $V(G^x) \setminus V(G^a)$.*

(3) *$dist_{H^x}(v^a_i, v^b_j) = \min_{P \in \mathcal{P}} \omega(P) - \min_{v \in L^b_j} \omega(v)$ where $\mathcal{P}$ is the set of all paths in G^x starting in L^a_i, ending in L^b_j, and having the penultimate vertex in $V(G^x) \setminus V(G^b)$.*

(4) *$dist_{H^x}(u^a_i, v^b_j) = \min_{P \in \mathcal{P}} \omega(P) - \min_{v \in L^a_i} \omega(v) - \min_{v \in L^b_j} \omega(v)$ where $\mathcal{P}$ is the set of all paths in G^x starting in L^a_i, ending in L^b_j, and having the second vertex in $V(G^x) \setminus V(G^a)$ and the penultimate vertex in $V(G^x) \setminus V(G^b)$.*

We prove Theorem 8 in two steps. We first prove that every path in H^x corresponds to some path in G^x. Later, we prove that also each optimal path between two label sets in G^x corresponds to some shortest path in H^x. We start with statement (1) of Theorem 8.

▶ **Lemma 9.** *Let $x \in V(T)$ be an internal node of the k-expression tree T marked with $\times_S$ for some $S \subseteq [k]^2$ with children y and z. Let H^x be the auxiliary graph as defined in Definition 7 and let P^* be an arbitrary v_i^y-u_j^z-path in H^x for some $i, j \in [k]$. Then there exists an L_i^y-L_j^z-path P in G^x with $\omega(P) = \omega_{H^x}(P^*)$.*

Proof. Due to the circular structure of H^x, each v_i^y-u_j^z-path in H^x will repeat the sequence $(v_p^y, u_q^y, v_r^z, u_s^z)$ for some $p, q, r, s \in [k]$ until reaching u_j^z at the end of a sequence. Thus, each v_i^y-u_j^z-path in H^x consists of 4ℓ vertices for $\ell \in \mathbb{N}$ and can be written as

$$P^* = (v_i^y = v_{p_1}^y, u_{q_1}^y, v_{r_1}^z, u_{s_1}^z, v_{p_2}^y, u_{q_2}^y, v_{r_2}^z, u_{s_2}^z, \dots, v_{p_\ell}^y, u_{q_\ell}^y, v_{r_\ell}^z, u_{s_\ell}^z = u_j^z).$$

One can construct a path P in G^x from P^* as follows: For each edge $(v_{p_i}^y, u_{q_i}^y)$ in P^* of cost c_{p_i,q_i}^y pick a shortest path in G^y of total cost c_{p_i,q_i}^y and for each edge $(v_{r_i}^z, u_{s_i}^z)$ in P^* of cost c_{r_i,s_i}^y pick a shortest path in G^z of total cost c_{r_i,s_i}^z for each $i \in [\ell]$. Those paths always exist since c_{p_i,q_i}^y resp. c_{r_i,s_i}^y are defined as the cost of a shortest $L_{p_i}^y$-$L_{q_i}^y$-path in G^y, resp. as the cost of a shortest $L_{r_i}^z$-$L_{s_i}^z$ in G^z. Since each edge (u_{q_i}, v_{r_i}) only exists if and only if there is a full join between the sets $L_{q_i}^y$ and $L_{r_i}^z$, one can connect the last vertex of the path corresponding to the previous edge in P^* (that ends in some vertex in $L_{q_i}^y$) to the first vertex of the path corresponding to the following edge in P^* (that starts at some vertex in $L_{r_i}^z$). In the same manner one can argue that due to each edge $(u_{s_i}^z, v_{p_{i+1}}^y)$ one can connect the last vertex of the path corresponding to the edge $(v_{r_i}^z, u_{s_i}^z)$ with the first vertex of the path corresponding to the edge $(v_{p_{i+1}}^y, u_{q_{i+1}}^y)$. In both cases, the cost of the vertices is already accounted for in the resp. $c_{\cdot,\cdot}^{\cdot}$ value. Thus, each v_i^y-u_j^z-path in H^x corresponds to an L_i^y-L_j^z-path in G^x of same cost. ◀

Next, we generalize this argumentation to the following corollary:

▶ **Corollary 10.** *Let $x \in V(T)$ be an internal node of the k-expression tree T marked with $\times_S$ for some $S \subseteq [k]^2$ with children y and z. Let H^x be the auxiliary graph as defined in Definition 7. Then for every $i, j \in [k]$ and $a, b \in \{y, z\}$ the following holds:*

(1) *For any v_i^a-u_j^b-path P^* in H^x there exists an L_i^a-L_j^b-path P in G^x with $\omega_{H^x}(P^*) = \omega(P)$.*

(2) *For any u_i^a-u_j^b-path P^* in H^x there exists an L_i^a-L_j^b-path $P = (p_1, p_2, \dots, p_\ell)$ in G^x with the property that $p_2 \in V(G^x) \setminus V(G^a)$ and $\omega_{H^x}(P^*) = \omega(P) - \omega(p_1)$.*

(3) *For any v_i^a-v_j^b-path P^* in H^x there exists an L_i^a-L_j^b-path $P = (p_1, \dots, p_{\ell-1}, p_\ell)$ in G^x with the property that $p_{\ell-1} \in V(G^x) \setminus V(G^b)$ and $\omega_{H^x}(P^*) = \omega(P) - \omega(p_\ell)$.*

(4) *For any u_i^a-v_j^b-path P^* in H^x there exists an L_i^a-L_j^b-path $P = (p_1, p_2, \dots, p_{\ell-1}, p_\ell)$ in G^x with the property that $p_2 \in V(G^x) \setminus V(G^a)$, $p_{\ell-1} \in V(G^x) \setminus V(G^b)$, and $\omega_{H^x}(P^*) = \omega(P) - \omega(p_1) - \omega(p_\ell)$.*

For any path in H^x that starts at some vertex u_i^a (resp. ends at some vertex v_j^b) one can find a corresponding path P in G^x with the property that the second vertex (resp. the penultimate vertex) is connected to *all* vertices of L_i^a (resp. L_j^b). Thus, one can extend any path that ends at *some* vertex in L_i^a by such a path (resp. one can prepend any path that starts in L_j^b by such a path). Hence, the cost of the first vertex (resp. last vertex) is neglected if the path starts in some vertex u_i^a or ends at some vertex v_j^b for $a, b \in \{x, y\}$. In general, every path that one can find in H^x corresponds to a path in G^x of essentially the same cost, possibly without the first or last vertex (which can be chosen as the minimum of the label set). This proves "$\leq$" in the equations of Theorem 8.

For the other direction, we will show that every optimal shortest path between two label sets in G^x is represented by a path in H^x.

▶ **Lemma 11.** *Let $x \in V(T)$ be an internal node of a k-expression tree T marked with $\times_S$ for some $S \subseteq [k]^2$ with children y and z. Let H^x be the auxiliary graph as defined in Definition 7. Let P be a shortest L_i^y-L_j^z-path in G^x for some $i, j \in [k]$. Then there exist a v_i^y-u_j^z-path P^* in H^x with $\omega_{H^x}(P^*) = \omega(P)$.*

Again, one can generalize the argumentation of Lemma 11 to the following corollary:

▶ **Corollary 12.** *Let $x \in V(T)$ be an internal node of the k-expression tree T marked with $\times_S$ for some $S \subseteq [k]^2$ with children y and z. Let H^x be the auxiliary graph as defined in Definition 7. Then for every $i, j \in [k]$ and $a, b \in \{y, z\}$ the following holds:*

(1) *For every shortest L_i^a-L_j^b-path P in G^x there exists a v_i^a-u_j^b-path P^* in H^x of cost $\omega_{H^x}(P^*) = \omega(P)$.*

(2) *For every shortest L_i^a-L_j^b-path P in G^x with the property that the second vertex is in $V(G^x) \setminus V(G^a)$ there exists a u_i^a-u_j^b-path P^* in H^x of cost $\omega_{H^x}(P^*) = \omega(P) - \min_{v \in L_i^a} \omega(v)$.*

(3) *For every shortest L_i^a-L_j^b-path P in G^x with the property that the penultimate vertex is in $V(G^x) \setminus V(G^b)$ there exists a v_i^a-v_j^b-path P^* in H^x of cost $\omega_{H^x}(P^*) = \omega(P) - \min_{v \in L_j^b} \omega(v)$.*

(4) *For every shortest L_i^a-L_j^b-path P in G^x with the property that the second vertex is in $V(G^x) \setminus V(G^a)$ and the penultimate vertex is in $V(G^x) \setminus V(G^b)$ there exists a u_i^a-v_j^b-path in H^x of cost $\omega_{H^x}(P^*) = \omega(P) - \min_{v \in L_i^a} \omega(v) - \min_{v \in L_j^b} \omega(v)$.*

Corollary 12 shows that every *shortest* L_i^a-L_j^b-path in G^x is represented in H^x for $i, j \in [k]$ and $a, b \in \{y, z\}$. Together with Corollary 10, this proves Theorem 8.

After the construction of the auxiliary graph H^x as defined in Definition 7, we compute and store the shortest distances for all pairs of vertices in H^x. With those values one can now compute the values $c_{i,j}^x$ and $a_{v,i}^x$ for $i, j \in [k]$ and $v \in V(G^x)$. Note that some of the values are only required in the second phase.

▶ **Corollary 13.** *Let $x \in V(T)$ be a node in the k-expression tree T marked with $\times_S$ for some $S \subseteq [k]^2$. For all $i, j \in [k]$ it holds that*

$$c_{i,j}^x = \min \left\{ dist_{H^x}(v_i^y, u_j^y), dist_{H^x}(v_i^y, u_j^z), dist_{H^x}(v_i^z, u_j^y), dist_{H^x}(v_i^z, u_j^z) \right\}.$$

▶ **Corollary 14.** *Let $x \in V(T)$ be a node in the k-expression tree T marked with $\times_S$ for some $S \subseteq [k]^2$. Then for any $v \in V(G^y)$ and $i \in [k]$ it holds that $a_{v,i}^x = \min_{j \in [k], a \in \{y,z\}} \left\{ a_{v,j}^y + dist_{H^x}(u_j^y, u_i^a) \right\}$.*

Second Phase. In this phase, we process the k-expression tree T in a top-down manner and use the local values that we have computed in the first phase to determine distances in the whole graph G.

Consider an internal node $x \in V(T)$ of the k-expression tree T marked with $\times_S$ for some $S \subseteq [k]^2$ and let y and z be the children of x in T. Let L_i^y resp. L_i^z denote the set of vertices with label i in G^y resp. G^z for $i \in [k]$. For an internal node x with children y and z we will compute for any vertex set L_i^y resp. L_i^z and every vertex $v \in V(G^x)$ the minimum cost of all paths in G that start in v and end in L_i^y resp. L_i^z with the property that the penultimate vertex is in $V(G) \setminus V(G^y)$, resp. in $V(G) \setminus V(G^z)$. Thus, the penultimate vertex is connected to *all* vertices of the vertex set L_i^y resp. L_i^z. It will therefore be convenient not to include the cost of the final vertex in these costs (cf. definition below). Note, that we consider the whole graph G in this step instead of just G^x.

Formally, for a node $x \in V(T)$ marked with $\times_S$ for some $S \subseteq [k]^2$ with children y and z we compute for every $v \in V(G^x)$ and $i \in [k]$ the following values:

- $d^x_{v,i,y} = \min_{P\in\mathcal{P}} \omega(P) - \min_{u\in L^y_i} \omega(u)$ where $\mathcal{P}$ is the set of all paths in G starting in v, ending in L^y_i, and having the penultimate vertex in $V(G) \setminus V(G^y)$.
- $d^x_{v,i,z} = \min_{P\in\mathcal{P}} \omega(P) - \min_{u\in L^z_i} \omega(u)$ where $\mathcal{P}$ is the set of all paths in G starting in v, ending in L^z_i, and having the penultimate vertex in $V(G) \setminus V(G^z)$.

For a node $x \in V(T)$ marked with $\circ_R$ for some $R\colon [k] \mapsto [k]$ and the child y, we only compute $d^x_{v,i,y}$. We start by computing those values for the root node. We can assume, w.l.o.g., that the root node has label $\times_S$ for some $S \subseteq [k]^2$.

▶ **Lemma 15.** *Let $r \in V(T)$ be the root node of the k-expression tree T marked with $\times_S$ for some $S \subseteq [k]^2$ and let y and z be the children of r. Let further H^r be the graph defined in Definition 7. Then, for any $v \in V(G^y)$ and for every $i \in [k]$ it holds that*

$$d^r_{v,i,y} = \min_{j\in[k]} \left\{a^y_{v,j} + dist_{H^r}(u^y_j, v^y_i)\right\} \quad \text{and} \quad d^r_{v,i,z} = \min_{j\in[k]} \left\{a^y_{v,j} + dist_{H^r}(u^y_j, v^z_i)\right\}.$$

Analogously, for any $v \in V(G^z)$ and for every $i \in [k]$ it holds that

$$d^r_{v,i,y} = \min_{j\in[k]} \left\{a^z_{v,j} + dist_{H^r}(u^z_j, v^y_i)\right\} \quad \text{and} \quad d^r_{v,i,z} = \min_{j\in[k]} \left\{a^z_{v,j} + dist_{H^r}(u^z_j, v^z_i)\right\}.$$

Next, we show how to propagate those values downwards in the k-expression tree, starting with a node marked with $\circ_R$ for some $R : [k] \mapsto [k]$.

▶ **Lemma 16.** *Let $x \in V(T)$ be an internal node of the k-expression tree T marked with $\circ_R$ for some $R\colon [k] \to [k]$. Let y be the unique child of x and w be the unique ancestor of x in T. Then $d^x_{v,i,y} = d^w_{v,R(i),x}$*

We now show the propagation for nodes x of T that are marked with $\times_S$. We start with one specific case and then conclude the general case as a corollary.

▶ **Lemma 17.** *Let $x \in V(T)$ be an internal node of the k-expression tree T that is marked with $\times_S$ for some $S \subseteq [k]^2$. Let y and z be the two children of x in T, let w be the unique ancestor of x in T, and let $v \in V(G^y)$ be arbitrary. Then $d^x_{v,i,y}$ is the minimum of the following three values:*

- $d^w_{v,i,x}$
- $\min_{j\in[k]} \left\{a^y_{v,j} + dist_{H^x}(u^y_j, v^y_i)\right\}$
- $\min_{j\in[k], c\in\{y,z\}} \left\{d^w_{v,j,x} + dist_{H^x}(v^c_j, v^y_i)\right\}$

Proof. After possibly adding nodes marked with $\circ_{id}$ to the k-expression tree, with id being the identity function, one can assume, that w is marked with $\circ_R$ for some $R\colon [k] \to [k]$ and that x is the only child of w.

Let $P = (p_1, \dots, p_{n-1}, p_n)$ be a shortest v-L^x_i-path in G with penultimate vertex in $V(G) \setminus V(G^y)$, i.e., with $p_1 = v$, $p_n \in L^y_i$, and $p_{n-1} \in V(G) \setminus V(G^y)$; thus, $\omega(P) - \omega(p_n) = d^x_{v,i,y}$. We distinguish three cases:

Case 1: $p_{n-1} \in V(G) \setminus V(G^x)$. In this case, P is also a v-L^x_i-path with the property that the penultimate vertex is in $V(G) \setminus V(G^x)$; thus, $d^x_{v,i,y} \geq d^w_{v,i,x}$.

Case 2: $p_{n-1} \in V(G^z)$ and all vertices of P are in G^x. In this case, we can compute the value in the same way as done in Lemma 15 for the root node and get $d^x_{v,i,y} = \min_{j\in[k]}\{a^y_{v,j} + dist_{H^x}(u^y_j, v^y_i)\}$.

Case 3: $p_{n-1} \in V(G^z)$ and at least one vertex in P is in $V(G) \setminus V(G^x)$. Let p_ℓ be the last vertex of P that is in $V(G) \setminus V(G^x)$; clearly, $p_{\ell+1} \in V(G^x)$. We split the path P into the two subpaths $P_1 = (p_1, \dots, p_\ell)$ and $P_2 = (p_{\ell+1}, \dots, p_n)$. Let $j \in [k]$ such that

$p_{\ell+1} \in L_j^x$. Since p_ℓ is connected to $p_{\ell+1}$, the vertex p_ℓ is connected to every vertex in L_j^x. We extend P_1 by $p' = \arg\min_{u \in L_j^x} \omega(u)$ and denote the resulting path by P_1'. Now it holds by definition that $\omega(P_1') - \omega(p') \geq d_{v,j,x}^w$, as the penultimate vertex p_ℓ of P_1' is in $V \setminus V(G^x)$. Let further $c \in \{y, z\}$ such that $p_{\ell+1} \in L_j^c$, noting that it does not change its label at x. Then $\omega(P_2) - \omega(p_n) \geq dist_{H^x}(v_j^c, v_i^y)$ by Theorem 8, as P_2 is a path in G^x. Note, that $\omega(P) = \omega(P_1') + \omega(P_2) - \omega(p')$. Thus, in this case it holds that

$$\begin{aligned}
d_{v,i,y}^x &= \omega(P) - \min_{u \in L_i^y} \omega(u) \\
&= \omega(P_1') - \omega(p') + \omega(P_2) - \min_{u \in L_i^y} \omega(u) \\
&\geq d_{v,j,x}^w + dist_{H^x}(v_j^c, v_i^y) + \omega(p_n) - \min_{u \in L_i^y} \omega(u) \\
&\geq d_{v,j,x}^w + dist_{H^x}(v_j^c, v_i^y) \\
&\geq \min_{j \in [k], c \in \{y,z\}} \left\{ d_{v,j,x}^w + dist_{H^x}(v_j^c, v_i^y) \right\}
\end{aligned}$$

We have seen in the case analysis above that in each case $d_{v,i,y}^x$ is at least the value considered in the case; in particular, it is at least equal to their collective minimum value. On the other hand, for each case there is a path P fulfilling the definition of $d_{v,i,y}^x$ such that $\omega(P) - \min_{u \in L_i^y} \omega(u)$ equals the value of the considered case. Thus, $d_{v,i,y}^x$ is also at most equal to the minimum taken over all three cases. This completes the proof. ◀

▶ **Corollary 18.** *Let $x \in V(T)$ be an internal node of the k-expression tree T marked with $\times_S$ for some $S \subseteq [k]^2$. Let y and z be the unique children of x in T, w be the unique ancestor of x in T, and let $\alpha, \beta \in \{y, z\}$ be arbitrary. Then, for $v \in V(G^\alpha)$ the value $d_{v,i,\beta}^x$ is the minimum of the following three values:*

- $d_{v,i,x}^w$
- $\min_{j \in [k]} \left\{ a_{v,j}^\alpha + dist_{H^x}(u_j^\alpha, v_i^\beta) \right\}$
- $\min_{j \in [k], c \in \{y,z\}} \left\{ d_{v,j,x}^w + dist_{H^x}(v_j^c, v_i^\beta) \right\}$

Third Phase. In the third phase, we traverse the k-expression tree T one final time; the ordering is immaterial. We go over all nodes x with label $\times_S$ for some $S \subseteq [k]^2$ and compute for each pair of vertices (u, v) with $u \in V(G^y)$ and $v \in V(G^z)$ the shortest u-v-path in G, where y and z are the two children of x in T. Since the leaves of T correspond one-to-one to single-vertex graphs, one for each vertex of G, this procedure will consider every pair of vertices in G at some node $x \in V(T)$.

▶ **Lemma 19.** *Let $x \in V(T)$ be an internal node of the k-expression tree T marked with $\times_S$ for some $S \subseteq [k]^2$. Let y and z be the two children of x and let $u \in V(G^y)$ and $v \in V(G^z)$. Then $dist_G(u, v) = \min_{i \in [k]} \left\{ d_{u,i,z}^x + a_{v,i}^z \right\}$.*

Running time. First, we need to transform the clique-width k-expression into a NLC-width k-expression tree T, which can be done in linear time $\mathcal{O}(n + m)$ [15].

In the first traversal, we compute for every node $x \in V(T)$ the values $a_{v,i}^x$ for $v \in V(G^x)$ and $i \in [k]$. Thus, we compute at most $n \cdot k$ values, each in time $\mathcal{O}(k)$, which results in a running time of $\mathcal{O}(nk^2)$ per node of T. In the case of a node x with label $\times_S$ for some $S \subseteq [k]^2$ we first compute the auxiliary graph H^x in time $\mathcal{O}(|V(H)| + |E(H)|) = \mathcal{O}(k^2)$ and solve (edge-weighted) ALL-PAIRS SHORTEST PATHS on H^x in time $\mathcal{O}(k^3)$. After this, by using Corollary 13 resp. Corollary 14, we compute each $c_{i,j}^x$ in constant time resp. each $a_{v,i}^x$ in time $\mathcal{O}(k)$ resulting in a running time per node $x \in V(T)$ of $\mathcal{O}(k^3 + k^2 \cdot n)$.

In the second phase we perform a top-down traversal of T to compute the for each node x the values $d^x_{v,i,y}$ and $d^x_{v,i,z}$ for all $v \in G^x$ and $i \in [k]$. Again, we compute at most $n \cdot k$ values, each in time $\mathcal{O}(k)$, which results in a running time of $\mathcal{O}(nk^2)$ per node of T. Since there are $\mathcal{O}(n)$ nodes in the k-expression tree T, the total running time for Phase One and Phase Two is $\mathcal{O}(nk^3 + n^2k^2) = \mathcal{O}(n^2k^2)$.

In the last phase, we consider each pair (u, v) of vertices exactly once and compute each pairwise distance in time $\mathcal{O}(k)$. Thus, running time for the last phase is $\mathcal{O}(n^2k)$. In total, we obtain the claimed bound of $\mathcal{O}(k^2n^2)$.

4 Conclusion

We started the study of VERTEX-WEIGHTED ALL-PAIRS SHORTEST PATHS in the FPT in P framework and obtained efficient parameterized algorithms with respect to clique-width and modular-width. The algorithm parameterized by modular-width is adaptive, i.e., even if the parameter reaches its upper bound of n, the algorithm is not worse than the best unparameterized algorithm, and even for $k \in \mathcal{O}(n^{1-\varepsilon})$ for any $\varepsilon > 0$, it outperforms the best unparameterized algorithm. The algorithm parameterized by the stronger parameter clique-width is truly subcubic if $\mathsf{cw} \in \mathcal{O}(n^{0.5-\varepsilon})$ for any $\varepsilon > 0$. It also permits us to solve DIAMETER in the same time $\mathcal{O}(\mathsf{cw}^2\, n^2)$, complementing the lower bound ruling out $\mathcal{O}(2^{o(\mathsf{cw})} \cdot n^{2-\varepsilon})$ for any $\varepsilon > 0$, due to Coudert et al. [4]. The algorithms only apply to the vertex-weighted case. Note also that the algorithm relative to clique-width assume to be given a suitable expression or decomposition, whereas the modular decomposition of a graph, and hence its modular-width, can be computed in linear time [24].

As mentioned in [16], considering edge-weighted graphs with (low) clique-width resp. low modular-width is hopeless, as one could modify an arbitrary input graph by adding all the missing edges with sufficiently large weights. Clearly, the shortest path lengths do not change, but the resulting graph is a clique and has constant clique-width and modular-width.

Apart from considering other parameters, one interesting open question is whether there is an *adaptive* algorithm for ALL-PAIRS SHORTEST PATHS parameterized by clique-width, e.g., can the running time be reduced to $\mathcal{O}(\mathsf{cw}\, n^2)$? This seems quite challenging, since even computing some variant of ALL-PAIRS SHORTEST PATHS for each node in the expression tree (on a graph with cw many nodes) results in a non-adaptive running time.

References

1 Matthias Bentert, Till Fluschnik, André Nichterlein, and Rolf Niedermeier. Parameterized aspects of triangle enumeration. *J. Comput. Syst. Sci.*, 103:61–77, 2019. doi:10.1016/j.jcss.2019.02.004.

2 Matthias Bentert and André Nichterlein. Parameterized complexity of diameter. In Pinar Heggernes, editor, *Algorithms and Complexity - 11th International Conference, CIAC 2019, Rome, Italy, May 27-29, 2019, Proceedings*, volume 11485 of *Lecture Notes in Computer Science*, pages 50–61. Springer, 2019. doi:10.1007/978-3-030-17402-6_5.

3 Timothy M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. *SIAM J. Comput.*, 39(5):2075–2089, 2010. doi:10.1137/08071990X.

4 David Coudert, Guillaume Ducoffe, and Alexandru Popa. Fully polynomial FPT algorithms for some classes of bounded clique-width graphs. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2765–2784. SIAM, 2018. doi:10.1137/1.9781611975031.176.

5 Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3):77–114, 2000. `doi:10.1016/S0166-218X(99)00184-5`.

6 Artur Czumaj and Andrzej Lingas. Finding a heaviest vertex-weighted triangle is not harder than matrix multiplication. *SIAM J. Comput.*, 39(2):431–444, 2009. `doi:10.1137/070695149`.

7 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

8 Robert W Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

9 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, Michal Pilipczuk, and Marcin Wrochna. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. *ACM Trans. Algorithms*, 14(3):34:1–34:45, 2018. `doi:10.1145/3186898`.

10 Michael L. Fredman. New bounds on the complexity of the shortest path problem. *SIAM J. Comput.*, 5(1):83–89, 1976. `doi:10.1137/0205006`.

11 Archontia C. Giannopoulou, George B. Mertzios, and Rolf Niedermeier. Polynomial fixed-parameter algorithms: A case study for longest path on interval graphs. *CoRR*, abs/1506.01652, 2015. `arXiv:1506.01652`.

12 Yijie Han and Tadao Takaoka. An $\mathcal{O}(n^3 \log\log n/\log^2 n)$ time algorithm for all pairs shortest paths. *J. Discrete Algorithms*, 38-41:9–19, 2016. `doi:10.1016/j.jda.2016.09.001`.

13 Thore Husfeldt. Computing graph distances parameterized by treewidth and diameter. In Jiong Guo and Danny Hermelin, editors, *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*, volume 63 of *LIPIcs*, pages 16:1–16:11. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. `doi:10.4230/LIPIcs.IPEC.2016.16`.

14 Yoichi Iwata, Tomoaki Ogasawara, and Naoto Ohsaka. On the power of tree-depth for fully polynomial FPT algorithms. In Rolf Niedermeier and Brigitte Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, volume 96 of *LIPIcs*, pages 41:1–41:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. `doi:10.4230/LIPIcs.STACS.2018.41`.

15 Öjvind Johansson. Clique-decomposition, NLC-decomposition, and modular decomposition-relationships and results for random graphs. In *Congr. Numer.* Citeseer, 1998.

16 Stefan Kratsch and Florian Nelles. Efficient and adaptive parameterized algorithms on modular decompositions. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*, volume 112 of *LIPIcs*, pages 55:1–55:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. `doi:10.4230/LIPIcs.ESA.2018.55`.

17 Stefan Kratsch and Florian Nelles. Efficient parameterized algorithms for computing all-pairs shortest paths. *arXiv e-prints*, page arXiv:2001.04908, January 2020. `arXiv:2001.04908`.

18 Andrzej Lingas and Dzmitry Sledneu. A combinatorial algorithm for all-pairs shortest paths in directed vertex-weighted graphs with applications to disc graphs. In Mária Bieliková, Gerhard Friedrich, Georg Gottlob, Stefan Katzenbeisser, and György Turán, editors, *SOFSEM 2012: Theory and Practice of Computer Science - 38th Conference on Current Trends in Theory and Practice of Computer Science, Špindlerův Mlýn, Czech Republic, January 21-27, 2012. Proceedings*, volume 7147 of *Lecture Notes in Computer Science*, pages 373–384. Springer, 2012. `doi:10.1007/978-3-642-27660-6_31`.

19 George B. Mertzios, André Nichterlein, and Rolf Niedermeier. Fine-grained algorithm design for matching. *CoRR*, abs/1609.08879, 2016. `arXiv:1609.08879`.

20 Seth Pettie. A new approach to all-pairs shortest paths on real-weighted graphs. *Theoretical Computer Science*, 312(1):47–74, 2004.

21 Seth Pettie and Vijaya Ramachandran. A shortest path algorithm for real-weighted undirected graphs. *SIAM Journal on Computing*, 34(6):1398–1431, 2005.

22 Asaf Shapira, Raphael Yuster, and Uri Zwick. All-pairs bottleneck paths in vertex weighted graphs. *Algorithmica*, 59(4):621–633, 2011. `doi:10.1007/s00453-009-9328-x`.

23 Susmita Susmita and Manish Pandey. Algorithms of all pair shortest path problem. *International Journal of Computer Applications*, 120(15):1–6, 2015.

24 Marc Tedder, Derek G. Corneil, Michel Habib, and Christophe Paul. Simpler linear-time modular decomposition via recursive factorizing permutations. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, volume 5125 of *Lecture Notes in Computer Science*, pages 634–645. Springer, 2008. `doi:10.1007/978-3-540-70575-8_52`.

25 Egon Wanke. k-NLC graphs and polynomial algorithms. *Discrete Applied Mathematics*, 54(2-3):251–266, 1994. `doi:10.1016/0166-218X(94)90026-4`.

26 Stephen Warshall. A theorem on boolean matrices. In *Journal of the ACM*. Citeseer, 1962.

27 R. Ryan Williams. Faster all-pairs shortest paths via circuit complexity. *SIAM J. Comput.*, 47(5):1965–1985, 2018. `doi:10.1137/15M1024524`.

28 Virginia Vassilevska Williams and R. Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM*, 65(5):27:1–27:38, 2018. `doi:10.1145/3186893`.

29 Raphael Yuster. Efficient algorithms on sets of permutations, dominance, and real-weighted APSP. In Claire Mathieu, editor, *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 950–957. SIAM, 2009. `doi:10.1137/1.9781611973068`.