Speeding up Networks Mining via Neighborhood Diversity

Gennaro Cordasco
Dipartimento di Psicologia, Università della Campania "Luigi Vanvitelli", Caserta, Italy
gennaro.cordasco@unicampania.it

Luisa Gargano
Dipartimento di Informatica, Università di Salerno, Fisciano, Italy
lgargano@unisa.it

Adele A. Rescigno
Dipartimento di Informatica, Università di Salerno, Fisciano, Italy
arescigno@unisa.it

Abstract

Parameterized complexity was classically used to efficiently solve NP-hard problems for small values of a fixed parameter. Then it has also been used as a tool to speed up algorithms for tractable problems. Following this line of research, we design algorithms parameterized by neighborhood diversity (nd) for several graph theoretic problems in P (e.g., Maximum Matching, Triangle counting and listing, Girth and Global minimum vertex cut). Such problems are known to admit algorithms parameterized by modular-width (mw) and consequently – being the nd a "special case" of mw – by nd. However, the proposed novel algorithms allow to improve the computational complexity from a time $O(f(\mathsf{mw}) \cdot n + m)$ – where n and m denote, respectively, the number of vertices and edges in the input graph – which is multiplicative in n to a time $O(g(\mathsf{nd}) + n + m)$ which is additive only in the size of the input.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Parameterized Complexity, Neighborhood Diversity, Maximum Matching, Triangle Counting, Girth, Global minimum vertex cut

Digital Object Identifier 10.4230/LIPIcs.FUN.2021.21

1 Introduction

A large online marketplace *Grove* is about to design a novel marketing strategy exploiting the data available thanks to their largely adopted premium program *Grove-FUN*. The CEO of Grove realized that customers' activities, reactions, and interactions on *Grove-FUN* can be used to perform predictive analysis on marketing, which increases both customer satisfaction and company returns. Indeed, the predictive analysis can be used to devise:

- Personalized recommendation system. For example, when a user adds a comic to his/her online shopping cart, similar comics purchased by other customers or other products purchased by customers having some similarity with the user can be recommended.
- Anticipatory Shipping Model for predicting the products customers are going to purchase, when and where they might need the products. According to the analysis, the items are pushed to a local distribution center or warehouse so they will be ready for shipping once a customer orders them. This approach increases product sales and profit margins because it reduces delivery time and overall expenses.
- Price optimization. Prices are set according to customer activities, item preferences, order history, and other factors.

- Viral marketing strategies. Sales of a new product can be improved by taking advantage of the human tendency to conform [5] by means of a viral marketing campaign based on targeted discounts [10].

The Grove Marketing Analysts realized that the data available on the premium program Grove-FUN can be easily modeled as a networked structure in terms of nodes (customers) and edges, or links (friendship or interactions) that connect them. This approach enables them to study the data through the use of networks and graph theory [16]. Several classical graph theory algorithms gained popularity in social network analysis. For instance, Triangle counting is used to detect communities and measure the cohesiveness of those communities, Maximum matching can be used to devise matching market strategies, the betweenness centrality is used to individuate influential vertices and their importance. Unfortunately, the Grove-FUN network is very large, with millions of customers and billions of edges. It is often prohibitively expensive to perform network analysis using standard algorithms on very large networks.

Fortunately, the clever Grove analysts have observed that the structure of Grove-FUN relationships follows the rules of social relationships. For instance, it is possible to observe how the connections are balanced between staying within a well identified community (where everybody knows each other) and cutting across communities. Such a specific structure of the network suggests a different way of thinking about standard algorithms in terms of their dense communities, and the ways in which they interact with each other.

On the basis of this observation, the following mathematical model was put forward. The network of customers is represented by a graph $G = (V, E)$, where V is the set of customers, and there is an edge between two customers if they reviewed the same product. Then the analysts were able to compress the network (keeping most of the information described by the original network) – exploiting the neighborhood diversity approach, introduced by Lampis in [32] – grouping similar nodes. A new graph H, called the type graph of G, characterized by a smaller number of nodes (each representing a group of customers) is generated in linear time and, starting from it, analysts have started having FUN redesigning and speeding up some classical algorithms such as maximum matching, triangle counting, girth, and global minimum vertex cut to use the graph H solving problems defined on G.

2 The hardness in P context

Algorithmic research aims to determine the best possible running time algorithms for computational problems. A first goal toward the classification of the problems according to their complexity was achieved by the NP-completeness theory, whose goal is to identify problems that are unlikely to be solved in polynomial time. However, there are still many intensively studied problems in P for which the worst-case running time of the best current algorithm is not known to be optimal. Namely, many important problems have classical algorithms running in $\tilde{O}(n^k)$ time[1] for some constant k, and this running time has not been significantly improved upon, in spite on many years of intensive research.

Recently, the *Hardness in P* tool for determine a hierarchy of the complexity of polynomial-time solvable problems has been introduced [37]. The key starting point here is the conjecture that there are problems that do not admit algorithms that perform significantly better than the already known ones. In particular, it has been conjectured that there is no $O(n^{2-\epsilon})$ time algorithm for 3-SUM and no $O(n^{3-\epsilon})$ time algorithm for All-Pairs Shortest Paths. Moreover,

[1] The $\tilde{O}(f)$ notation ignores factors of $\log(f)$

the *Hardness in P* theory is based on the Strong Exponential Time Hypothesis (SETH): There is no $c < 2$ such that k-SAT can be solved in time $O(c^n)$ for each fixed k. This conjecture is used to obtain lower bounds to the complexity of some problems in P, in the sense that the existence of a faster algorithm for one of these problems implies the existence of a faster algorithm for one of the fundamental problems mentioned above. Recent work in the area can be found in [1, 2, 9, 22, 38].

On the positive side, efforts have been made to improve algorithms for problems in P on some restricted classes of graphs. In particular, parameterized algorithms have been recently proposed as a tool to speed up algorithms for problems in P [12]. Parameterized complexity was classically used to efficiently solve NP-hard problems for small values of a fixed parameter [14, 36]. Formally, a parameterized problem with input size n and parameter t is called *fixed parameter tractable (FPT)* if it can be solved in time $f(t) \cdot n^c$, where f is a function only depending on t and c is a constant.

Unfortunately there are several parameters whose computation is an NP-hard problem itself. As an example computing treewidth, rankwidth, and vertex cover are all NP-hard problems– even though they are computable in FPT time when their respective parameters are bounded. Moreover, the parameterized complexity of computing the clique-width of a graph is still an open problem [13]. On the contrary, modular-width [21] and neighborhood diversity [32] are two recently introduced parameters that are computable in linear time $O(m)$ on a graph with m edges [3, 7, 11, 12, 15, 18, 19, 21, 24, 25, 32].

2.1 Our results

In this paper we design algorithms parameterized by neighborhood diversity for well studied tractable problems. Namely, we consider the maximum matching, triangle counting, girth, and global minimum vertex cut problems. Such problems are known to admit algorithms parameterized, among other parameters, by modular-width (mw) of the input graph. This implies, being the neighborhood diversity (nd) a "special case" of modular width (i.e., $\mathsf{mw} \leq \mathsf{nd}$)[21], that the same algorithm can be used with respect to nd. However, for a graph with n nodes and m edges the proposed novel algorithms allow to improve the computational complexity from a time $O(f(\mathsf{mw}) \cdot n + m)$, which is multiplicative in n [12] to a time $O(g(\mathsf{nd}) + n + m)$ which is only additive in the size of the input.

3 Neighborhood diversity

Given a graph $G = (V, E)$, two nodes $u, v \in V$ have the same *type* iff $N(v) \setminus \{u\} = N(u) \setminus \{v\}$. The *neighborhood diversity* of a graph G, introduced by Lampis in [32] and denoted by $\mathsf{nd}(G)$, is the minimum number t of sets in a partition $V_1, V_2, \ldots, V_t$, of the node set V, such that all the nodes in V_i have the same type, for $i = 1, \ldots, t$. In the following we use nd, instead of $\mathsf{nd}(G)$, when the graph G is clear from the context. The family $\mathcal{V} = \{V_1, V_2, \ldots, V_t\}$ is called the *type partition* of G.

Let $G = (V, E)$ be a graph with type partition $\mathcal{V} = \{V_1, V_2, \ldots, V_t\}$. By definition, each V_i induces either a *clique* or an *independent set* in G. Whenever $V_i \in \mathcal{V}$ is a singleton, we consider it as inducing an independent set, so any V_i inducing a clique has $|V_i| \geq 2$. For each $V_i, V_j \in \mathcal{V}$, we get that either each node in V_i is a neighbor of each node in V_j or no node in V_i has a neighbor in V_j.

Starting from a graph G and its type partition $\mathcal{V} = \{V_1, \ldots, V_t\}$, we can see each element of $\mathcal{V}$ as a vertex of a new graph H, called the *type graph* of G, with $V(H) = \{1, 2, \ldots, t\}$ and $E(H) = \{(x, y) \mid x \neq y,$ there is a complete bipartite graph between V_x and V_y in $G, \}$
$\cup \{(x, x) \mid |V_x| \geq 2$ and V_x and induces a clique in $G\}$.
Determining $\mathsf{nd}(G)$ and the corresponding type partition, can be done in time $O(n + m)$ [32].

4 Maximum matching

A matching in a graph is a set of edges with pairwise disjoint end vertices. The MAXIMUM MATCHING (MM) problem consists in computing a matching of maximum size.

MAXIMUM MATCHING can be solved in polynomial time by Edmond's algorithm [17]. A simple implementation of such an algorithm requires time $O(n^4)$, however Micali and Vazirani showed how to implement Edmond's algorithm in time $O(m\sqrt{n})$ [34].

MAXIMUM MATCHING is considered fundamental to the study of fixed-parameter algorithm for problems in P [26, 33]. In particular Mertzios et al. designed new algorithms to solve maximum matching in $O(\rho^{O(1)}(n+m))$ time for various graph parameters ρ [33]. Coudert et al. gave a $O(\rho^4 n + m)$ time algorithms for solving MM, when parameterized by either the modular-width or the P_4-sparseness of the graph [12]. Recently, Kratsch et al. improved this last result to $O(\mathsf{mw}^2 \log \mathsf{mw} \cdot n + m)$[31].

In the following we present an algorithm parameterized by neighborhood diversity for solving the MAXIMUM MATCHING of a graph $G = (V, E)$. Our algorithm will use the solution of a generalization of MM, namely the MAXIMUM b-MATCHING problem [20].

Let $b : V \to \mathbb{Z}^+$ be a function on the vertices of G (where G may have also self-loops), a *b-matching* for G is a function $x : E \to \mathbb{Z}^+$ such that for each $u \in V$

$$\sum_{v:\ u \neq v \wedge (u,v) \in E} x(u,v) + 2x(u,u) \leq b(u). \tag{1}$$

The value $x(u,v)$ represents the multiplicity of the edge (u,v), that is how many times the edge (u,v) is used by the matching. Note that if $b(u) = 1$, for each $u \in V$, then the function x becomes a classical matching for G.
A maximum b-matching for G is a function x for which $\sum_{(u,v)\in E} x(u,v)$ is maximum. In the following we will call by $|x| = \sum_{(u,v)\in E} x(u,v)$ the *size* of the b-matching x.
Gabow showed how to find a maximum b-matching in time $O(\min\{b(V), n \log n\}\ (m + n \log n))$, where $b(V) = \sum_{v \in V} b(v)$ [20].

4.1 The algorithm

We describe now the proposed algorithm.

1. Let $\mathcal{V} = \{V_1, \ldots, V_t\}$ and $H = (V(H), E(H))$ be the type partition and the type graph of G, respectively. Define the function $b_H : V(H) \to \mathbb{Z}^+$ such that $b_H(i) = |V_i|$.
2. Use Gabow's algorithm to find a maximum b_H-matching for H, let it be $x_H : E \to \mathbb{Z}^+$.
3. Construct the desired maximum matching M for G using x_H.

We show now how to implement the above step 3. To this aim, we first notice that (1) implies

$$\sum_{\substack{j \\ i \neq j \wedge (i,j) \in E}} x_H(i,j) + 2x_H(i,i) \leq b_H(i) \qquad \text{for each } i \in V(H)$$

$$x_H(i,j) \leq \min\{b_H(i), b_H(j)\} \qquad \text{for each } (i,j) \in E(H)$$

Fix any $i \in V(H)$ and let $j_1, j_2, \ldots, j_{a(i)}$ be the neighbors of i in H such that $x_H(i, j_\ell) > 0$, for $\ell = 1, \ldots, a(i)$. Recalling that $x_H(i,j)$ represents the multiplicity of the edge (i,j) in the matching and $b_H(i) = |V_i|$, we select

$$s(i) = \sum_{\substack{j \\ i \neq j \wedge (i,j) \in E(H)}} x_H(i,j) + 2x_H(i,i) = \sum_{\ell=1}^{a(i)} x_H(i, j_\ell) + 2x_H(i,i)$$

vertices in V_i and partition them into $a(i)+1$ sets, $S_{i,j_1}, S_{i,j_2}, \ldots, S_{i,j_{a(i)}}, C_i$ such that

$$|C_i| = \begin{cases} 2\ x_H(i,i) & \text{if } V_i \text{ induces a clique,} \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad |S_{i,j_\ell}| = x_H(i,j_\ell) \quad \text{for } \ell = 1,\ldots,a(i).$$

A matching M for G can be now obtained as

$$M = \left(\bigcup_{\substack{(i,j)\in E(H) \\ i\neq j}} M_{i,j} \right) \bigcup \left(\bigcup_{\substack{i\in V(H) \\ V_i \text{ is a clique}}} M_i \right), \tag{2}$$

where

- $M_{i,j}$ is a perfect matchings connecting the vertices in $S_{i,j}$ with the vertices in $S_{j,i}$ (recall that $|S_{i,j}| = |S_{j,i}| = x_H(i,j)$);
- M_i is a perfect matchings connecting $x_H(i,i)$ pairs of vertices in C_i, if V_i induces a clique, and $M_i = \emptyset$ otherwise.

▶ **Fact 1.** *The size of the matching M equals that of the b_H-matching in H, that is, $|M| = |x_H|$.*

Proof. By definition $|x_H| = \sum_{(i,j)\in E(H)} x_H(i,j) = \sum_{\substack{(i,j)\in E(H) \\ i\neq j}} |M_{i,j}| + \sum_{i\in V(H)} |M_i| = |M|$. ◀

The following result implies the desired optimality of the matching in (2).

▶ **Lemma 1.** *If x_H is a maximum b_H-matching for H then M is a maximum matching for G.*

Proof. Assume by contradiction that there exists a matching M' for G such that $|M'| > |M|$. Let $y(i,j)$ be the number of edges in M' connecting one vertex in V_i to a vertex in V_j, for $i,j \in V(H)$ (clearly, $y(i,j) = 0$ if $(i,j) \notin E(H)$). Recalling that M' is a matching and, therefore, each vertex in V_i can be the end vertex of only one edge in M', we have that

$$\sum_{j:\ i\neq j \wedge (i,j)\in E(H)} y(i,j) + 2y(i,i) \leq |V_i| = b_H(i).$$

By (1), y is a b_H-matching for H. Moreover, its existence contradicts optimality of x_H, since

$$|y| = \sum_{(i,j)\in E(H)} y(i,j) = |M'| > |M| = |x_H|.$$ ◀

Running time. Considering that $b_H(V(H)) = \sum_{i\in V(H)} b_H(i) = \sum_{i\in V(H)} |V_i| = n$ and that $|E(H)| \leq \mathsf{nd}^2$, we have that the algorithm due to Gabow [20] for computing a maximum b_H-matching for H requires time $O(\min\{b_H(V(H)), \mathsf{nd}\log\mathsf{nd}\} \cdot (\mathsf{nd}\log\mathsf{nd} + |E(H)|)) = O(\min\{n, \mathsf{nd}\log\mathsf{nd}\} \cdot \mathsf{nd}^2)$.

Summarizing, we have shown the following result.

▶ **Theorem 2.** *For any graph $G = (V,E)$, the maximum matching problem can be solved in time $O(\mathsf{nd}^3 \log \mathsf{nd} + n + m)$.*

We stress that the above algorithm can be generalized to get a maximum b-matching of G. Hence, we can obtain a linear-time kernelization for b-matching (a linear-time algorithm to compress the input into a small input of size $f(\mathsf{nd})$).

5 Cycles

Finding and counting simple cycles in graphs is a classical well studied problem [4]. In particular, triangle detection, counting and/or enumeration problems have applications in many areas, such as spam detection over complex network analysis [6, 35] and bioinformatics [39]. An extensive annotated list of applications can be found in Kolountzakis et al. [29].

Establishing whether there exists a triangle in a general graph is conjectured not to be solvable in time $O(n^{3-\epsilon})$, for $\epsilon > 0$, with a combinatorial algorithm [38]. Furthermore, in [2] it is also conjectured that TRIANGLE COUNTING is not solvable in time $O(n^{\omega-\epsilon})$, for $\epsilon > 0$, with ω being the exponent for fast matrix multiplication[2]. The fastest known algorithm for TRIANGLE COUNTING in general graphs relies on fast matrix multiplication and runs in time $O(n^{\omega})$ [4]. Coudert et al. [12] presented a fast algorithm parameterized by the clique-width cw of the graph running in time $O(\mathsf{cw}^2(n+m))$. Bentert et al. [8] have studied the problem under various parameters including feedback edge number, distance to d-degenerate graphs, and clique-width. They also presented an algorithm for TRIANGLE LISTING in a graph parameterized by the clique-width, running in time $O(\mathsf{cw}^2 \cdot n + n^2 + \#T)$ where $\#T$ denotes the number of triangles in G.

With respect to general cycles in a graph, the GIRTH problem asks to determine the size of the smallest cycle in a given graph. By an old result of Itai and Rodeh [28], if the girth is even, it is possible to determine it in time $O(n^2)$. If, otherwise, the graph has odd girth then any algorithm would have to be able to detect if the graph has a triangle, requiring time $O(n^{\omega})$. Itai and Rodeh also showed that any algorithm that can find a triangle in dense graphs can also compute the girth, so obtaining an $O(n^{\omega})$ time girth algorithm. However, in case of sparse graphs the best running time for the girth is in general $O(nm)$. In [12] was presented a parameterized algorithms that solve the GIRTH problem in time $O(\rho^2(n+m))$ where ρ is either the modular-width or the split-width.

5.1 Triangle counting and listing

Given a graph $G = (V, E)$, TRIANGLE COUNTING problem asks to determine the number of triangles in G. We present an algorithm that solves the TRIANGLE COUNTING problem, then we extend it to solve the TRIANGLE LISTING problem.

Algorithm. Let $\mathcal{V} = \{V_1, V_2, \ldots, V_t\}$ be the type partition of G and let $H = (V(H), E(H))$ be the type graph. We count the triangles in G by computing three values

$-a_i$: The number of triangles with all the three vertices in V_i; for $i \in V(H)$.

$-b_i$: The number of triangles with exactly two vertices in V_i; for $i \in V(H)$.

$-c$: The number of triangles with each vertex in a different set of the type partition.

It is immediate to see that $a_i = b_i = 0$ whenever V_i induces an independent set in G. If, otherwise, V_i induces a clique in G then each subset of V_i containing three vertices is a triangle. Furthermore, for each neighbor j of i in H, each pair of vertices in V_i forms a triangle with any vertex in V_j. Hence, for $i \in V(H)$

$$a_i = \begin{cases} \binom{|V_i|}{3} & \text{if } V_i \text{ induces a clique and } |V_i| \geq 3, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

[2] It is known that $2 \leq \omega < 2.3728639$ due to Le Gall [23].

$$b_i = \begin{cases} \binom{|V_i|}{2} \sum_{j:(i,j)\in E(H)} |V_j| & \text{if } V_i \text{ induces a clique and } |V_i| \geq 2 \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

To compute c we have to count the number of triangles with each vertex in a distinct set of the type partition. Hence, for each triangle in H involving for instance the vertices $i, j, k \in V(H)$ we have $|V_i||V_j||V_k|$ triangles in G

$$c = \sum |V_i||V_j||V_k|, \tag{5}$$

where the sum is over all $i, j, h \in V(H)$ forming a triangle in H.

We use a result due to Kratsch et al.[31] to compute c in (5). We present it in terms of the type partition of G.

▶ **Lemma 3.** *[31] Let $G = (V, E)$ be a graph with type partition $V_1, V_2, \ldots, V_t$ and type graph H. Consider the weight function $w : E(H) \to R^+$ with $w(i, j) = \sqrt{|V_i||V_j|}$. Let A be the weighted adjacency matrix of H with respect to w. Then, the number of triangles in G with each vertex in a different set of the type partition is*

$$c = \sum_{i,j\in V(H)} \frac{1}{3}(A^2 \circ A)_{i,j} \tag{6}$$

where $A \circ B$ denotes the Hadamard product of the matrices A and B, i.e., $(A \circ B)_{i,j} = A_{i,j}B_{i,j}$.

By (3), (4) and (6), we have that the number of triangles in G is

$$\sum_{\substack{i\in V(H):\\ V_i \text{ induces a clique}\\ |V_i|\geq 3}} \binom{|V_i|}{3} + \sum_{\substack{i\in V(H):\\ V_i \text{ induces a clique}\\ |V_i|\geq 2}} \left(\binom{|V_i|}{2} \sum_{j:(i,j)\in E(H)} |V_j| \right) + \sum_{i,j\in V(H)} \frac{1}{3}(A^2 \circ A)_{i,j} \tag{7}$$

Running time. The first two terms of (7) can be computed in time $O(|E(H)|) \leq O(\mathsf{nd}^2)$. The time to evaluate the last term depends on the time to compute the matrix $A^2 \circ A$. The best algorithm to compute the matrix multiplication A^2 requires time $O(\mathsf{nd}^\omega)$. Finally, multiplying each element of A^2 with the correspondent element in A and summing up all the obtained values takes time $O(\mathsf{nd}^2)$.

▶ **Theorem 4.** *The* TRIANGLE COUNTING *problem can be solved in time $O(\mathsf{nd}^\omega + n + m)$.*

By exploiting the above algorithm and (7) we can easily obtain an algorithm that lists all the triangles of G and so solving the TRIANGLE LISTING problem.

▶ **Theorem 5.** *For any graph $G = (V, E)$, the* TRIANGLE LISTING *problem can be solved in time $O(\mathsf{nd}^\omega + n + m + \#T)$, where $\#T$ denotes the number of triangles in G.*

5.2 Girth

We present an algorithm that finds the girth $\mu(G)$ of any connected graph G.

Algorithm. Let $\mathcal{V} = \{V_1, V_2, \ldots, V_t\}$ be the type partition of G and let $H = (V(H), E(H))$ be the type graph.

One can obtain the girth of G by distinguishing the following cases:

1. If there exists $i \in V(H)$ such that $|V_i| \geq 2$ and V_i induces a clique then $\mu(G) = 3$.
2. If Cases 1. does not hold then compute the girth of H ($h = \mu(H)$):
   - If either there exist two neighbors $i, j \in V(H)$, with $|V_i| \geq 2$ and $|V_j| \geq 2$ or there exists a vertex $i \in V(H)$ with $|V_i| \geq 2$ having at least two neighbors then $\mu(G) = \min\{4, h\}$.
   - Otherwise, $\mu(G) = h$.

The algorithm first checks if there is a triangle involving an edge connecting two nodes in the same type set. Indeed, if there exists at least a type set V_i inducing a clique (recall that $|V_i| \geq 2$ in this case) then any two vertices in V_i with any neighbor inside or outside V_i form a triangle (recall that G is connected). If this is the case the algorithm returns $\mu(G) = 3$.

Otherwise, we know that *all the type sets $V_1, V_2, \ldots, V_t$ induce independent sets.* In this case, any cycle in G must involve only edges between pairs of type sets, and so edges of H. The algorithm computes the girth h of H. Then if there is a cycle of length 4 involving two nodes in the same type set, the algorithm returns the value $\min\{4, h\}$. Cycles of length 4 are identified by the following conditions. If there exist $(i, j) \in E(H)$, with $|V_i| \geq 2$ and $|V_j| \geq 2$, then any two vertices in V_i and any two vertices in V_j induces a cycle of length 4 in G. Analogously, if there exists a vertex $i \in V(H)$ with $|V_i| \geq 2$ having at least two neighbors, say $j, h \in V(H)$, then a vertex in V_j together with a vertex in V_h and any two vertices in V_i induce a cycle of length 4 in G.

If none of the above cases holds, then we know that for each $i \in V(H)$, such that $|V_i| \geq 2$, i has only one neighbor j such that $|V_j| = 1$. Hence, no vertex in V_i may be a vertex of a cycle. Therefore, there is a cycle of length ℓ in G iff there is a cycle of length ℓ in H. Hence, $\mu(G) = \mu(H)$ and the girth of G is obtained by computing the girth of H.

Running time. The worst case in the algorithm is the calculus of the girth of H. Hence, the running time of the algorithm is $O(\mathsf{nd}^\omega + n + m)$.

▶ **Theorem 6.** *For any graph $G = (V, E)$, the* GIRTH *problem can be solved in time $O(\mathsf{nd}^\omega + n + m)$.*

6 Global minimum vertex cut

In this section we consider the GLOBAL MINIMUM VERTEX CUT problem, a generalization to vertex capacities of the vertex connectivity of a graph.

Given a graph $G = (V, E)$, a set $X \subseteq V$ is a *vertex cut* of G if $G - X$ is disconnected. It is then possible to partition $V(G) - X$ into two non empty sets A_X and B_X where each vertex in A_X has only neighbors in $A_X \cup X$ and, each vertex in B_X has only neighbors in $B_X \cup X$. We call the pair (A_X, B_X) the *disconnected partition* of $G - X$. Given a capacity function $c : V \to R^+$ on the vertices of G, a vertex cut X of minimum capacity $c(X) = \sum_{u \in X} c(u)$ is said the *global minimum vertex cut* of G. The GLOBAL MINIMUM VERTEX CUT problem asks to find the global minimum vertex cut of G given a capacity function c.

As highlighted in [31], the global minimum vertex cut in G can be obtained by solving a global edge capacitated cut in a directed graph using standard reductions between flow/cut variants. By using the result of Hao et al.[27], it can be done in time $O(n^3 \log n)$. Kratsch et al.[30, 31] presented an algorithm parameterized by the modular width mw that solves the global minimum vertex cut problem in time $O(\mathsf{mw}^2 \log \mathsf{mw} \cdot n + m)$. We present here an algorithm parameterized by the neighborhood diversity that solves the problem in time $O(\mathsf{nd}^3 \log \mathsf{nd} + n + m)$.

Algorithm. Let $\mathcal{V} = \{V_1, V_2, \ldots, V_t\}$ be the type partition of G and let $H = (V(H), E(H))$ be the type graph. Consider the capacity function $c_H : V(H) \to R^+$ of the type graph H defined as $c_H(i) = \sum_{u \in V_i} c(u)$. The algorithm first finds the global minimum vertex cut X_H of H with capacities c_H, and then returns $X = \bigcup_{i \in X_H} V_i$.

We will prove that the set X returned by the algorithm is a global minimum vertex cut of G with capacity c. To this aim, we first characterize a global minimum vertex cut in terms of the type partition of G.

▶ **Lemma 7.** *Let $G = (V, E)$ be a graph with type partition $V_1, V_2, \ldots, V_t$. Let $c : V \to R^+$ be the capacity function of G and let X be any global minimum vertex cut of G with capacity c.*

a) *For each disconnected partition A_X, B_X of $G - X$, any vertex $u \in X$ must have at least a neighbor in A_X and at least a neighbor in B_X.*

b) *For each $i = 1, \ldots, t$, either $V_i \subseteq X$ or $V_i \cap X = \emptyset$.*

c) *There exists a disconnected partition A_X, B_X of $G - X$ such that either $V_i \subseteq A_X$ or $V_i \cap A_X = \emptyset$ (resp. either $V_i \subseteq B_X$ or $V_i \cap B_X = \emptyset$), $i = 1, \ldots, t$.*

Proof. To prove a) we proceed by contradiction and assume that there exists a vertex $u \in X$ that has only neighbors in $X \cup A_X$ (resp. in $X \cup B_X$). In this case $X - \{u\}$ is a global vertex cut whose capacity is less than that of X, and this contradicts the minimum capacity of X.

The proof of b) is again by contradiction. Suppose that there exists a vertex $u \in X \cap V_i$ and a vertex $v \in V_i - X$. W.l.o.g., suppose that $v \in A_X$. Since $u, v \in V_i$, they share the same neighborhood and by a) vertex u must have at least a neighbor in B_X. Hence, also v must have at least a neighbor in B_X, thus contradicting the assumption that $G - X$ is disconnected.

Finally, we prove c). Let A_X, B_X be any disconnected partition of $G - X$ and let $u \in A_X \cap V_i$ (and $V_i \not\subseteq A_X$). By b) no vertex in V_i is in X. We have only to consider the possibility that $B_X \cap V_i \neq \emptyset$. This is not possible if V_i induces a clique in G since otherwise A_X and B_X should be connected by at least one edge. In case V_i induces an independent set in G, we can move each vertex in $B_X \cap V_i$ from B_X to A_X obtaining again a disconnected partition of $G - X$. Hence, $A_X \cup (B_X \cap V_i)$ and $B_X - V_i$ is a disconnected partition of $G - X$. This proves that it is possible to find a disconnected partition of $G - X$ satisfying c). ◀

▶ **Theorem 8.** *The set X returned by the algorithm is a global minimum vertex cut with capacity c.*

Proof. We first prove that X is a vertex cut of G, that is we prove that $G - X$ is disconnected. By the fact that X_H is a global minimum vertex cut of H with capacity c_H, we can find a disconnected partition A_H, B_H of $H - X_H$ and so we have that each $i \in A_H$ has all its neighbors in $A_H \cup X_H$ and, each $j \in B_H$ has all its neighbors in $B_H \cup X_H$. By the construction of X in the algorithm, i.e., $X = \bigcup_{i \in X_H} V_i$, we can pinpoint the sets $A = \bigcup_{i \in A_H} V_i$ and $B = \bigcup_{i \in B_H} V_i$, that are a partition of $V(G) - X$. Now, we prove that there is no edge between vertices of A and B. Fix $u \in A$. Since $u \in V_i$ for some $i \in A_H$ and using the fact that A_H and B_H are a disconnected partition of $H - X_H$ and so u cannot be neighbor of any vertex in same set V_j for $j \in B_H$, we have that vertex u has all its neighbors in $A \cup X$. In the same way we can prove that each $v \in B$ has all its neighbors in $B \cup X$. This proves that A, B is a disconnected partition of $G - X$.

Finally, we prove that X has minimum capacity, so completing the proof. By contradiction suppose there exists a global minimum vertex cut Y with $c(Y) < c(X)$. By b) in Lemma 7 we can define the set $Y_H = \{i \in V(H) \mid V_i \subseteq Y\}$. Let A_Y and B_Y be the disconnected partition

of $G - Y$ for which c) in Lemma 7 holds. Hence, if we define $A = \{i \in V(H) \mid V_i \subseteq A_Y\}$ and $B = \{j \in V(H) \mid V_j \subseteq B_Y\}$, then each $i \in A$ has all its neighbors in $A \cup Y_H$, and each $j \in B$ has all its neighbors in $B \cup Y_H$ proving that Y_H is a vertex cut of H. Furthermore,

$$\begin{aligned} c(Y_H) &= \sum_{i \in Y_H} c_H(i) = \sum_{i \in Y_H} \sum_{u \in V_i} c(u) = \sum_{u \in Y} c(u) = c(Y) \\ &< c(X) = \sum_{u \in X} c(u) = \sum_{i \in X_H} \sum_{u \in V_i} c(u) = \sum_{i \in X_H} c_H(i) \\ &= c(X_H), \end{aligned}$$

which contradicts the assumption that X_H is a global minimum vertex cut of H respect to the capacity function c_H. ◀

Running time. The running time of our algorithm strongly depends on the time to compute the global minimum vertex cut X_H of H with capacities c_H. By using the best known algorithm to evaluate the global minimum vertex cut, we have that X_H can be computed in time $O(\mathsf{nd}^3 \log \mathsf{nd})$.

▶ **Theorem 9.** *For any graph $G = (V, E)$ and capacity function $c : V \to R^+$, the* GLOBAL MINIMUM VERTEX CUT *problem can be solved in time $O(\mathsf{nd}^3 \log \mathsf{nd} + n + m)$.*

References

1 A. Abboud, F. Grandoni, and V. V. Williams. Subcubic equivalences between graph centrality problems, apsp and diameter. In *ACM-SIAM Symposium on Discrete Algorithms (SODA), 1681–1697*, 2015.

2 A. Abboud and V. V. Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *IEEE Symposium on Foundations of Computer Science (FOCS'14). IEEE, 434–443*, 2014.

3 F. N. Abu-Khzam, S. Li, C. Markarian, F. Meyer auf der Heide, and P. Podlipyan. Modular-Width: An Auxiliary Parameter for Parameterized Parallel Complexity. In *Frontiers in Algorithmics, FAW 2017, LNCS 10336*, 2017.

4 N. Alon, R. Yuster, and U. Zwick. Finding and counting given length cycles. *Algorithmica, 17(3): 209–223*, 1997.

5 S. E. Asch. Studies of independence and conformity: A minority of one against a unanimous majority. *Psychological Monographs, 70*, 1956.

6 L. Becchetti, P. Boldi, C. Castillo, and A. Gionis. Efficient algorithms for large-scale local triangle counting. *ACM Trans. Knowl. Discov. Data 4 (3)*, 2010.

7 R. Belmonte, F. V. Fomin, P. A. Golovach, and M. S. Ramanujan. Metric Dimension of Bounded Width Graphs. *Mathematical Foundations of Computer Science (MFCS '15), LNCS 923*, 2015.

8 M. Bentert, T. Fluschnik, A. Nichterlein, and R. Niedermeier. Parameterized aspects of triangle enumeration. *Journal of Computer and System Sciences, 103, 61–77*, 2019.

9 M. Borassi, P. Crescenzi, and M. Habib. Into the square: On the complexity of some quadratic-time solvable problems. *Electronic Notes in Theoretical Computer Science, 322:51–67*, 2016.

10 G. Cordasco, L. Gargano, A. A. Rescigno, and U. Vaccaro. Optimizing spread of influence in social networks via partial incentives. In *22nd International Colloquium on Structural Information and Communication Complexity, SIROCCO 2015. LNCS 9439, 119–134*, 2015.

11 G. Cordasco, L. Gargano, A. A. Rescigno, and U. Vaccaro. Evangelism in social networks: Algorithms and complexity. *Networks 71(4): 346–357*, 2018.

12 D. Coudert, G. Ducoffe, and A. Popa. Fully polynomial FPT algorithms for some classes of bounded clique-width graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '18), 2765–2784*, 2018.

13 M. Doucha and J. Kratochvíl. Cluster Vertex Deletion: A Parameterization between Vertex Cover and Clique-Width. *MFCS 2012, 348–359*, 2012.

14 R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer, 2012.

15 P. Dvorák, D. Knop, and T. Toufar. Target Set Selection in Dense Graph Classes. In *arXiv:1610.07530*, 2016.

16 D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World.* Cambridge University Press, ISBN:0521195330, 2010.

17 J. Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics, 17(3):449–467*, 1965.

18 J. Fiala, T. Gavenciak, D. Knop, M. Koutecky, and J. Kratochvíl. Fixed parameter complexity of distance constrained labeling and uniform channel assignment problems. In *arXiv:1507.00640*, 2015.

19 F. V. Fomin, M. Liedloff, P. Montealegre, and I. Todinca. Algorithms Parameterized by Vertex Cover and Modular Width, through Potential Maximal Cliques. In *Ravi R., Gørtz I.L. (eds) Algorithm Theory – SWAT 2014, LNCS vol 8503, Springer*, 2014.

20 H. N. Gabow. Data structures for weighted matching and extensions to b-matching and f-factors. *ACM Transactions on Algorithms, Vol. 14 (3), Article 39*, 2018.

21 J. Gajarský, M. Lampis, and S. Ordyniak. Parameterized Algorithms for Modular-Width. In *Gutin G., Szeider S. (eds) Parameterized and Exact Computation, IPEC 2013, LNCS 8246*, 2013.

22 A. Gajentaan and M. H. Overmars. On a class of $o(n^2)$ problems in computational geometry. *Computational Geometry, 5(3):165–185*, 1995.

23 F. Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic ComputationJuly, ISSAC '14, 296–303*, 2014.

24 R. Ganian. Using neighborhood diversity to solve hard problems. In *arXiv:1201.3091*, 2012.

25 L. Gargano and A.A. Rescigno. Complexity of conflict-free colorings of graphs. *Theoretical Computer Science, 566, 39–49*, 2015.

26 A. C. Giannopoulou, G. B. Mertzios, and R. Niedermeier. Polynomial fixed-parameter algorithms: A case study for longest path on interval graphs. *Theoretical Computer Science, 689: 67–95*, 2017.

27 J. Hao and J. B. Orlin. A faster algorithm for finding the minimum cut in a directed graph. *J. Algorithms, 17(3):424–446*, 1994.

28 A. Itai and M. Rodeh. Finding a minimum circuit in a graph. *SIAM Journal on Computing, 7(4):413–423*, 1978.

29 M. N. Kolountzakis, G. L. Miller, R. Peng, and C. E. Tsourakakis. Efficient triangle counting in large graphs via degree-based vertex partitioning. *Internet Math. 8 (1–2), 161–185*, 2012.

30 D. Kratsch and J. P. Spinrad. Between $o(nm)$ and $o(n^\alpha)$. *SIAM Journal on Computing, 36(2):310–325*, 2006.

31 S. Kratsch and F. Nelles. Efficient and adaptive parameterized algorithms on modular decompositions. In *arXiv:1804.10173*, 2018.

32 M. Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica 64, 19–37*, 2012.

33 G. B. Mertzios, A. Nichterlein, , and R. Niedermeier. The power of linear-time data reduction for maximum matching. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science (MFCS'17), 83, 46:1–46:14*, 2017.

34 S. Micali and V. V. Vazirani. An $o(sqrt(|V|)|E|)$ algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science, 17–27*, 1980.

35 M. E. J. Newman. The structure and function of complex networks. *SIAM Rev. 45 (2), 167–256*, 2003.

36 R. Niedermeier. *Invitation to Fixed-Parameter Algorithms.* Oxford University Press, 2006.

37 V. V. Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *International Symposium on Parameterized and Exact Computation (IPEC), 16–28*, 2015.

38 V. V. Williams and R. R. Williams. Subcubic equivalences between path, matrix and triangle problems. In *IEEE Symposium on Foundations of Computer Science (FOCS), 645–654*, 2010.

39 Y. Zhang and S. Parthasarathy. Extracting analyzing and visualizing triangle k-core motifs within networks. In *Proceedings of the 28th International Conference on Data Engineering, ICDE '12, IEEE Computer Society, 1049–1060*, 2012.