NP-hardness of two vertex ordering optimizations

Fabrice Lécuyer
Sorbonne Université
CNRS, LIP6
F-75005 Paris, France
fabrice.lecuyer@lip6.fr

Louis Jachiet
LTCI
Télécom Paris
Institut Polytechnique de Paris

Clémence Magnien
Sorbonne Université
CNRS, LIP6
F-75005 Paris, France

Lionel Tabourier
Sorbonne Université
CNRS, LIP6
F-75005 Paris, France

***Abstract*—State-of-the-art triangle listing algorithms use vertex ordering to accelerate their execution and to bound their time complexity. We prove that it is NP-hard to find an ordering that minimizes two quantities involved in the complexity of these algorithms.**

***Index Terms*—graph algorithm, NP-hardness, vertex ordering, pattern mining**

I. Introduction

A. Notations

We consider an unweighted undirected simple graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges. The set of neighbors of a vertex u is denoted $N_u = \{v, \{u, v\} \in E\}$, and its degree is $d_u = |N_u|$. An ordering π is a permutation over the vertices that gives a distinct index $\pi_u \in [\![1, n]\!]$ to each vertex u. In the directed acyclic graph (DAG) $G_\pi = (V, E_\pi)$, for $\{u, v\} \in E$, E_π contains (u, v) if $\pi_u < \pi_v$, and (v, u) otherwise. In such a directed graph, the set N_u of neighbors of u is partitioned into its predecessors N_u^- and successors N_u^+. We define the indegree $d_u^- = |N_u^-|$ and the outdegree $d_u^+ = |N_u^+|$; their sum is $d_u^- + d_u^+ = d_u$. A triangle of G is a set of vertices $\{u, v, w\}$ such that $\{u, v\}, \{v, w\}, \{u, w\} \in E$.

B. Context

Common algorithms used to list triangles in an undirected graph use vertex ordering [1]–[4]. The ordering creates an orientation of the edges, which partitions neighbors into successors and predecessors. The complexity thus depends on the ordering π and is given by either of the following costs:

Definition 1 (Cost induced by an ordering): Given an undirected graph G, the costs C^{++} and C^{+-} induced by a vertex ordering π are defined by:

$$C^{++}(\pi) = \sum_{u \in V} d_u^+ d_u^+ \qquad C^{+-}(\pi) = \sum_{u \in V} d_u^+ d_u^-$$

Yet, optimizing C^{+-} is known to be NP-hard [5]; as far as we know, the proof of this result has not been published. We give a new simpler proof in Section II. We prove in Section III that this is also true for C^{++}, thus leading to Theorem 1:

Theorem 1 (NP-hardness): Given a graph G, it is NP-hard to find an ordering π that minimizes $C^{+-}(\pi)$ or that minimizes $C^{++}(\pi)$.

II. NP-hardness of the C^{+-} problem

Given a graph G and an order $\prec$ on the vertices of G, we define $succ_\prec(u)$ (respectively $pred_\prec(u)$) as the set of neighbors v of u such that $u \prec v$ (resp. $v \prec u$). For any subset of vertices W, we note $C_\prec^{+-}(W) = \sum_{u \in W} |succ_\prec(u)| \cdot |pred_\prec(u)|$. Using this definition we formalize the following problem:

Problem 1 (C^{+-}): Given an undirected graph $G = (V, E)$ and an integer K, is there an order $\prec$ on the vertices such that $C_\prec^{+-}(V) \leq K$?

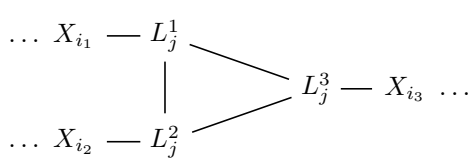
Problem 2 (NAE3SAT+ or not-all-equal positive three-satisfiability): Given a formula $\phi = c_1 \wedge \cdots \wedge c_m$ in conjunctive normal form where each clause consists in three positive literals, is there an assignment to the variables satisfying ϕ such that in no clause all three literals have the same truth value?

The NAE3SAT+ problem is known to be NP-complete by Schaefer's dichotomy theorem [6]. We will show that this problem can be reduced to the C^{+-} problem, thus proving that C^{+-} is NP-hard. Note that a proof was given in-hard [5] but, as far as we know, it has never been published. We give a new simpler proof of the following theorem:

Theorem 2: C^{+-} is NP-hard.

Definition 2: Let ϕ be an instance of NAE3SAT+ with variables $x_1, \ldots, x_n$ and clauses $c_1, \ldots, c_m$, where clause c_j is of the form $l_j^1 \vee l_j^2 \vee l_j^3$. We define a graph G_ϕ by creating three connected vertices L_j^1, L_j^2, L_j^3 representing the literals of each clause c_j; additionally, a vertex X_i is created for each variable x_i and connected to all the L_j^a such that $l_j^a = x_i$. More formally, $G_\phi = (V_\phi, E_\phi)$ with:

- $V_\phi = \{X_i \mid i \in [\![1, n]\!]\} \cup \{L_j^1, L_j^2, L_j^3 \mid j \in [\![1, m]\!]\}$
- $E_\phi = \big\{\{L_j^1, L_j^2\}, \{L_j^1, L_j^3\}, \{L_j^2, L_j^3\} \mid j \in [\![1, m]\!]\big\} \cup \big\{\{X_i, L_j^a\} \mid x_i = l_j^a\big\}$



Proposition 1 ($\Longrightarrow$): Given an instance ϕ of NAE3SAT+ with m clauses and the associated graph G_ϕ, if ϕ is satisfiable then there exists an order $\prec$ on V_ϕ such that $C_\prec^{+-}(V_\phi) \leq 2m$.

Proof: Let ϕ be a satisfiable instance of NAE3SAT+ with the above notations. Take a valid assignment and let us note k the number of variables set to true. There exist indices $i_1, \dots, i_n$ such that $x_{i_1}, \dots, x_{i_k} = true$ and $x_{i_{k+1}}, \dots, x_{i_n} = false$, and for each clause c_j, there are indices $t_j, a_j, f_j \in \{1, 2, 3\}$ such that $l_j^{t_j} = true$, $l_j^{f_j} = false$ and $l_j^{a_j}$ has any value. Now construct the following order on V_ϕ, so that true variables come first, then in each clause the false literal comes before the true one, and the false variables are at the end:

$$
\begin{array}{ll}
X_1 \prec \cdots \prec X_k & \text{True variables} \\
\prec L_1^{f_1} \prec \cdots \prec L_m^{f_m} & \text{False literals} \\
\prec L_1^{a_1} \prec \cdots \prec L_m^{a_m} & \text{Other literals} \\
\prec L_1^{t_1} \prec \cdots \prec L_m^{t_m} & \text{True literals} \\
\prec X_{k+1} \prec \cdots \prec X_n & \text{False variables}
\end{array}
$$

If a given variable x_i is true, the associated vertex X_i has only successors, if it is false it has only predecessors, so in both cases $C_\prec^{+-}(\{X_i\}) = 0$. For a given clause c_j, the variable $l_j^{f_j}$ is false so the corresponding X_i is a successor of $L_j^{f_j}$, which also has successors $L_j^{a_j}$ and $L_j^{t_j}$, but no predecessor. Similarly, $L_j^{t_j}$ has no successor; thus $C_\prec^{+-}(\{L_j^{f_j}, L_j^{t_j}\}) = 0$. Now $L_j^{a_j}$ has one predecessor $L_j^{f_j}$, one successor $L_j^{t_j}$, and one neighbor X_i that is a predecessor if x_i is true, otherwise a successor; in both cases, $C_\prec^{+-}(\{L_j^{a_j}\}) = 2$. The only vertices with a non-negative cost are the $L_j^{a_j}$, so the sum over all m clauses gives $C_\prec^{+-}(V_\phi) = 2m$. ∎

Proposition 2 ($\Longleftarrow$): Given an instance ϕ of NAE3SAT+ with m clauses and the associated graph G_ϕ, if there exists an order $\prec$ on V_ϕ such that $C_\prec^{+-}(V_\phi) \leq 2m$ then ϕ is satisfiable.

Proof: Conversely, consider an order $\prec$ on V_ϕ such that $C_\prec^{+-}(V_\phi) \leq 2m$. For all j, define $f_j, a_j, t_j \in \{1, 2, 3\}$ such that $L_j^{f_j} \prec L_j^{a_j} \prec L_j^{t_j}$; then $L_j^{a_j}$ has one successor, one predecessor, and one other neighbor X_i, so its cost is 2. As G_ϕ contains m such independent triangles, $C_\prec^{+-}(\{L_1^{a_1}, \dots, L_m^{a_m}\}) = 2m$. To ensure $C_\prec^{+-}(V_\phi) \leq 2m$, all the other vertices must have either only predecessors or only successors. If vertex X_i has successors only, assign x_i to true; if X_i has predecessors only, assign x_i to false. For all j, $L_j^{f_j}$ has at least 2 successors ($L_j^{a_j}$ and $L_j^{t_j}$) so its corresponding X_i has to be a successor, which means $x_i = l_j^{f_j}$ is false; similarly, $l_j^{t_j}$ is true. Each clause thus has one true and one false literal, so ϕ is satisfied. ∎

III. NP-HARDNESS OF THE C^{++} PROBLEM

a) Order of elimination: In the main text of the paper, we search for a permutation π but the only important aspect of the permutation is that it defines an order on the vertices. For this NP-hardness proof, it will help think of the following equivalent but more "intuitive" formulation of the problem: we are looking for an order $\prec$ minimizing some cost. We can think of the order as an order in which we eliminate vertices and each time we eliminate a vertex with an outdegree d we pay a cost of d^2 and the cost of an order is the cost of eliminating all vertices.

For the formulation using orders it will help to look at the set of neighbors of u appearing after u in the order $\prec$, which we denote $succ_\prec(u)$ for an order $\prec$. Therefore $|succ_\prec(u)|^2$ is the cost that we pay when we remove u, which allows us to reformulate C^{++} as:

Problem 3 (C^{++}): For a given undirected graph $G = (V, E)$ and an integer K, does there exist an order $\prec$ of the vertices such that $\sum_{u \in V} |succ_\prec(u)|^2 \leq K$?

b) The weighted-C^{++} problem: For the sake of simplicity our proof of completeness will rely on a second novel problem, the weighted-C^{++} problem, and we will show that C^{++} is NP-complete by exhibiting first a reduction between C^{++} and weighted-C^{++} and then a second reduction between weighted-C^{++} and the Set-Cover problem (a well-known NP-complete problem). We now present the weighted-C^{++} problem:

Problem 4 (weighted-C^{++}): Given an undirected graph $G = (V, E)$, a vertex-weighting function $w : V \to \mathbb{N}$ and an integer K, does there exist an order $\prec$ of the vertices such that $\sum_{u \in V} (|succ_\prec(u)| + w(u))^2 \leq K$?

c) Terminology: Given a graph G with the vertex weighting function w and an order $\prec$, the *cost* is the function $\sum_{u \in V} (|succ_\prec(u)| + w(u))^2$ applied to the graph with that order. The *optimal cost* of a graph is the minimal cost achievable by any order. Notice that an instance of the weightless problem can be viewed as an instance of the weighted problem where all weights are 0.

A. Optimality criteria for orders

One difficulty of the reduction proofs is to show that an order necessarily behaves in a controlled way. We see in this section several criteria that ensures that some order has an optimal cost.

One tool that we will use for our optimality criteria is the multiset of cost. Given a graph G and an order $\prec$, the multiset of costs $MC(G, \prec)$ is the multiset composed of the $|succ_\prec(u)|$ for each vertex u in G. The *cost* (or *squared cost*) of a multiset M is $\sum_{c \in M} c^2$. The *linear cost* of a multiset M is the sum of elements in the multiset, i.e. $\sum_{c \in M} c$.

Property 1: For a graph G (weighted or not) the size and the linear sum of the multiset $MC(G, \prec)$ does not depend on the order $\prec$.

Proof: The size of $MC(G, \prec)$ is the number of vertices in G, the linear cost corresponds the number of edges plus the sum of costs. ∎

Note that this allows us to talk about the *linear cost* of a graph G as the linear cost of any multiset of cost for any of its order.

Property 2: When there exists $d \in \mathbb{N}$ such that $MC(G, \prec)$ contains only the values d and $d+1$ then the order $\prec$ is optimal.

Proof: Let us consider an order $\prec$ as described above and let us consider any optimal order $\prec'$ of G. Suppose that the multiset $MC(G, \prec')$ contains e and f such that $e < f - 1$. Then replacing them by $e+1$ and $f-1$ reduces the cost because $(e^2 + f^2) - \big((e+1)^2 + (f-1)^2\big) = 2(f - 1 - e) > 0$.

By repetitively applying this operation we end up with a multiset M that has the same size and the same linear size as $MC(G, \prec')$ with a lower cost but that contains a' times d' and b' times $d'+1$ for some d'.

Without loss of generality we can suppose that there is at least one d in $MC(G, \prec)$ which means that d is the linear cost divided by the size of $MC(G, \prec)$. For the same reason we can ask that d' is the linear cost of $MC(G, \prec)$ divided by its size. Because the size and the linear cost of $MC(G, \prec)$ does not depend on $\prec$ this proves that $d' = d$. By using, once again, the fact that the linear cost and size are the same, we have that the two are the same which means that they have the same cost. ∎

While the property above is true for any graph (weighted or not) it is not really useful for weightless graphs because, in a weightless graph, the last vertex u that we eliminate in the order $\prec$ has $|succ_{\prec}(u)| = 0$, and more generally the vertex u_i which is ordered in the i-th position from the end, has $|succ_{\prec}(u_i)| < i$. The following property handles this case:

Property 3: For a weightless graph, when there exists $d \in \mathbb{N}$ such that $MC(G, \prec)$ contains all integers from 0 to $d+1$ and at most once the integers 0 to $d-1$, then $\prec$ is an optimal order.

Proof: The proof of optimality is similar to the proof of property 2: when the property does not hold, we can find two elements v_i and v_j with $v_i + 2 \leq v_j$ and we can diminish the cost by setting $v_i = v_i + 1$ and $v_j = v_j - 1$. ∎

Finally let us introduce the notion of *marginal cost* for a multiset.

Definition 3 (Marginal cost): We introduce the *marginal cost* to measure how much a multiset deviates from the optimal repartition (as given in property 3). Formally, given a multiset M of size n, we can compute d such that the linear cost of M is $n \times d + v$ where $1 \leq v \leq n$. The *marginal cost* of M is then:

$$\sum_{u \in M} \max\Big(0, u - (d+1)\Big)$$

We know from property 2 that the multiset M' that minimizes the cost with the same linear cost and the same size only contains d and $d+1$ (with at least one $d+1$ since $v > 0$). In other words the marginal cost counts the number of elements bigger than needed and how much they go over the average cost: if we have a $d+2$ it counts for 1, if we have a $d+5$ it counts for 4, etc.

Note that the marginal cost cannot be used directly to decide if an order is optimal. Indeed, consider the two following multisets: M composed of nine times the value 10 and one time the value 11 and M' composed of nine times the values 11 and one time the value 2. They have the same size, the same linear cost and the same marginal cost (which is 0) but M has a lower squared cost than M'.

The following property describes the minimal squared cost among all the multisets with the same size, the same linear cost and the same marginal cost:

Property 4: Among all the multisets that have a size n, a linear cost of $d \times n + v$ with $1 \leq v \leq n$ and a marginal cost of at least k (with $2k < v$), then the ones achieving the minimal squared cost are composed of k times the value $d+2$, $v - 2k$ times the value $d+1$ and $(n - v + k)$ times the value d.

Proof: Take a multiset M and let us note $d+1$ the average value rounded up. Suppose that M contains a value $d-i$ with $i > 0$. Because the linear cost of M is strictly bigger than $d \times |M|$ we can find at least one value $d+j$ with $j > 0$ such that diminishing $d+j$ to $d+j-1$ keeps the marginal cost above k.

Such a $d+j$ can be found because either we have at least one value $d+1$ in M (which does not count in the marginal cost) or the marginal cost was strictly higher than k. Indeed, let us note c for the marginal cost and consider the sum $\sum_{u \in M} min(d+1, u)$. In this sum we know that there are at most c elements at $d+1$, and all other elements are at most d with one at $d-i < d$. Thus $\sum_{u \in M} min(d+1, u) < nd + c$. Now the marginal cost is always equal to $\sum_{u \in M} u - \sum_{u \in M} min(d+1, u) = nd + v - \sum_{u \in M} min(d+1, u) > nd + v - nd - c = v - c$. Overall we have $c > v - c$ or, equivalently $2c > v$, but we supposed $2k < v$ therefore $c > k$.

In all cases we can form M' from M by replacing one $d-i$ with $d-i+1$ and $d+j$ by $d+j-1$. The cost of M' is strictly lower than the cost of M while still having a marginal cost higher than k. ∎

This property can then be used to compare multisets and is summarized by the following property:

Property 5: When M has a marginal cost of k then the cost of M is at least $2k$ higher than the balanced distribution (as given by property 2). This $2k$ bound is reached for the optimality criterion of property 4.

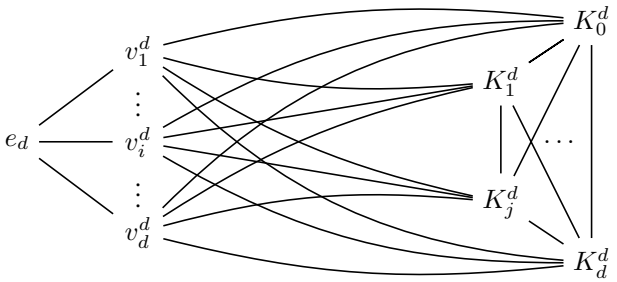
Proof: As seen before the optimal can be reached by taking two values $i, j \in M$ with $i+2 \leq j$ and changing them to $i+1$ and $j-1$. This balancing operation can reduce by at most 1 the marginal cost but reduces the cost by $i^2 + j^2 - (i+1)^2 - (j-1)^2 = 2(j-i) - 2$ and since $j - i \geq 2$ this means the reduction is at least 2 and exactly 2 when $i + 2 = j$. Since we need at least k balancing operations to reach the optimal, this gives us at least a $2k$ reduction of the cost to reach the optimal. Notice that when dealing with an optimal multiset in the sense of property 4 we only combine a d with a $d+2$ which gives us the exact bound. Conversely if we are not in the case of 4 we will have to combine something below (or equal to) d with something higher than $d+3$ OR do a combination that

does not diminish the marginal cost (such as combining $d+1$ and $d+3$). ∎

B. Reduction between weighted-C^{++} and C^{++}

Any instance of C^{++} can be seen as an instance of weighted-C^{++} where the weights are set to 0. For that purpose, the idea is to take a vertex u with some non null weight $w(u)$, link u to $w(u)$ vertices $v_1, \dots, v_{w(u)}$ and make sure that we can guarantee that u appears before all the $v_1, \dots, v_{w(u)}$ in any optimal order. We will thus exhibit a family of graphs to create such v_i vertices before showing that these v_i vertices can always appear after u in the order. Finally we will prove the full reduction.

1) The L_d family of graphs: Let us consider the graph L_d parameterized by $d \in \mathbb{N}$ that contains a $(d+1)$-clique composed of the vertices $K_0^d, \dots, K_d^d$, one vertex e_d that has d neighbors $v_1^d \dots v_d^d$ and such that there is an edge between each v_i^d and each vertex of K_d. Here is a depiction of L_d:



a) Best cost C_d for L_d: The best cost C_d for L_d is induced by the order that starts with e_d then continues with the v_i^d and finishes with the K_i^d, because in that case the cost is d^2 for e_d, $(d+1)^2$ for each v_i^d and i^2 for K_i^d (supposing we start with K_d^d and end with K_0^d). This is optimal by virtue of property 3.

b) Best cost for L_d with a weight 1 on e_d: If we add a weight 1 on e_d then the best cost can still be achieved with the same order but this time the cost of e_d is increased from d^2 to $(d+1)^2$ which means an increase of $2d+1$. Note that here, the optimality cannot be deduced directly from property 3 as the property only applies to weightless graphs. However we can prove that there is an optimal order starting with e_d.

For that, consider any order $\prec$ and let us show that $\prec$ can always can be improved to an order that places the vertex e_d at the beginning. The order $\prec$ ranks three types of vertices: the vertex e_d, the vertices of type V (the $(v_i^d)_i$) and the vertices of type K (the $(K_i^d)_i$).

Let us first suppose that there is a vertex of type K before a vertex of type V before the vertex e_d. In that case the first i vertices are of type V (we can have $i=0$), then we have $j+1$ vertices of type K and then one vertex of type V. Let us consider how the cost changes by exchanging this last K with this last V, i.e. to change from V^iK^jKV to V^iK^jVK. It is clear that the cost changes only for the exchanged V and K. Before the exchange the cost of V was $(d-j)^2$ and after it is $(d-j+1)^2$ whereas for K it was $(2d-i-j)^2$ and after it is $(2d-i-j-1)^2$ overall we get a cost that diminishes by:

$$\begin{aligned}(2d-i-j)^2 - (2d-i-j-1)^2 + (d-j)^2 - (d-j+1)^2 \\ = 2d-2i-4 \\ = 2(d-i-2)\end{aligned}$$

Therefore, unless $i+1=d$ we know that the cost decreases which means that we can always move the V vertices at the beginning except for maybe one. In the end we have that the beginning of an optimal sequence can be restricted to the form $V^iK^le_d$ or $V^{d-1}K^lVe_d$. In this first case moving e_d at the beginning decreases the score by (i^2+i) and in the second case transforming $V^{d-1}K^lVe_d$ into $e_dV^{d-1}K^lV$ decreases the score by d^2+d-2l (which is ≥ 0 because $l \leq d$).

In all cases we can move e_d at the beginning and still improve the order. Now that we know that the best cost can be achieved by placing e_d at the beginning we know that the best cost is $C_d + 2d + 1$ as the best cost for L_d can also be achieved by eliminating e_d first. Placing a weight on e_d has no effect on the rest of the graph once it is eliminated and since e_d can be placed first in both cases.

2) Partitioned graphs: Let us consider a graph G composed of two subgraphs G_1 and G_2 plus exactly one edge $\{e_1, e_2\}$ with $e_1 \in V_1$ and $e_2 \in V_2$. Any order $\prec$ on V induces an order on V_1, an order on V_2 and an order between e_1 and e_2. If another order $\prec'$ induces the same order on V_1, the same order on V_2 and the same order between e_1 and e_2, it realizes the same cost as $\prec$. Therefore an optimal order for G can be found as either an optimal for G_1 and an optimal order for G_2 but where we add a weight of 1 on e_2 OR an optimal order for G_2 and an optimal order for G_1 but where we add a weight of 1 on e_1.

As a result, we obtain the following property:

Property 6: If adding a weight 1 on e_1 in G_1 increases the best cost of G_1 by M and if adding a weight 1 on e_2 increases the best cost of G_2 by at most M, then the best cost of G is equal to the best cost of G_1 plus the best of G_2 where we add a weight of 1 on e_2.

3) Finishing the reduction:

Property 7: Let (G, K) be an instance of the weighted problem, we can compute an equivalent instance of the weightless problem in a time polynomial in the number of edges and vertices in G plus the sum of weights in G.

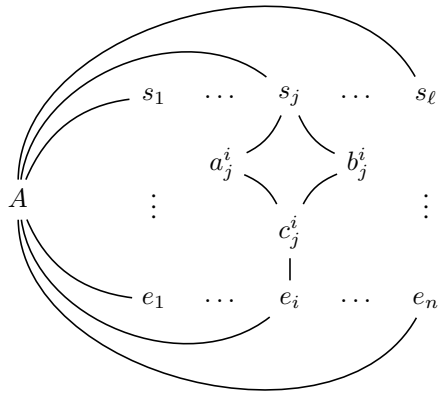
Proof: If all the weights in G are zeros, the proof is trivial. Let us suppose that there is a vertex u with a weight $w(u) > 0$ and a degree $d - w(u)$. Let us consider the graph G' composed of G but where the weight of u is reduced by 1 plus a fresh copy of L_d and an edge between u and e_d.

We claim that the best cost of G' is lower than $K + C_d$ if and only if the best cost of G is lower than K. Indeed, we know that the graph L_d is such that adding a weight 1 on e_d increases the best cost from C_d to $C_d + 2d + 1$. We also know that the sum of the degree of u plus its cost is d therefore for

any order $\prec$ adding a weight 1 on u increases the cost of $\prec$ of, at most, $2d+1$. By applying property 6 we do obtain the expected result.

By applying $\sum_u w(u)$ times this property we obtain an instance (G', K') equivalent to (G, K). The resulting instance is larger with $\sum_u w(u) \times |L_{deg(u)+w(u)}|$ more vertices but this stays polynomial in the size of G plus the sum of weights and the resulting instance can be computed in polynomial time. ■

This proves that if the weighted-C^{++} problem is strongly NP-hard, the C^{++} problem is also NP-hard.



C. Reduction between the weighted-C^{++} and Set Cover

Our reduction for the weighted case will be *strong reductions*, meaning the version of the problem where the weights are polynomial in the size of the graph is still NP-hard. Our reduction will be based on the *Set cover problem*. We recall here the definition of this problem and invite the reader to check the literature for a proof of its NP-completeness:

Problem 5 (Set cover): Given two integers n, k and a set of sets P does there exist a subset $P' \subset P$ of size k such that $\cup_{S \in P'} S = [\![1, n]\!]$?

Let us fix an instance (P, n, k) of the Set Cover problem asking whether we can find k sets $S_1, \ldots, S_k$ in P such that $S_1 \cup \cdots \cup S_k = [\![1, n]\!]$. We suppose, without loss of generality, that the instance is not trivial in the sense that $|P| \geq k$ (there are at least k sets in P), $\cup_{S \in P} S = [\![1, n]\!]$ (each integer in $[\![1, n]\!]$ is contained in at least one $S \in P$) and that all sets $S \in P$ are such that $S \subseteq [\![1, n]\!]$.

Let us exhibit a weighted graph G and a value V such that best cost for G is less than V if and only if $[\![1, n]\!]$ can be covered with k sets from P.

1) The order minimization with weight instance for a set-cover instance: Our reduction will provide a graph G with a weight function w depending on a parameter d such that the set cover instance has a solution if and only if the best order has a multiset of cost containing at most k values $d+2$ and other values are at d and $d+1$ (we will explicit later the values of V and d).

a) Vertices of G: In G the vertices are: a special vertex A, n vertices $e_1, \ldots, e_n$ one for each $i \in [\![1, n]\!]$, ℓ vertices, $s_1, \ldots, s_\ell$ one vertex s_j for each set $S_j \in P$ and finally three vertices a_j^i, b_j^i, c_j^i for each $i \in S_j$.

b) Edges of G: The vertex A has an edge with all vertices of the form s_j or e_i. For a pair (i, j) with $i \in S_j$, both a_j^i and b_j^i have an edge with s_j and c_j^i; in turn c_j^i has an edge with e_i. Overall the graph looks like this:

c) Weights of vertices in G: Recall that the cost of a vertex is the sum of its degree and its weight. In G, we set the weights so that each vertex has a cost of $d+2$, except for the c_j^i which have a cost of $d+3$ and the vertex A which has a cost of $d+1+n+k$. Parameter d just needs to be big enough so that all weights are positive (hence $d > k$ and $d > 2 \times |S|$ for all $S \in P$ suffices).

d) The value of V: As we will show, when there is a set-cover with k sets then we have an order $\prec$ for G such that $MC(G, \prec)$ contains k times the value $d+2$ (corresponding to the k selected sets), $\sum_{S \in P} |S| - n$ times the value d and all the other values are $d+1$. Overall we have $V = k(d+2)^2 + (\sum_{S \in P} |S| - n)d^2 + r(d+1)^2$, where r is the number of vertices in G minus k and minus $(\sum_{S \in P} |S| - n)$.

Note that, per property 4, this value V corresponds to the minimal cost for an order that has a marginal cost of k. Conversely we will show that if there is a solution with a marginal cost of k or less then there is a set-cover with k set, proving that it is a reduction. Note that this converse direction is stronger than what is needed as it exists multisets with a marginal cost of k that do not match the minimal cost.

2) Proof that a solution to set-cover implies a solution to C^{++}: Suppose that we have a solution to set-cover with the sets $S_{j_1}, \ldots, S_{j_k}$. Let us prove that our graph G has an elimination order where the cost of each vertex is d or $d+1$ or $d+2$ but with only k vertices at $d+2$.

The elimination order can be built by iterating j over $j_1, \ldots, j_k$. For each j we start by eliminating s_j for a cost of $d+2$, then we iterate $i \in S_j$ and we eliminate the a_j^i and b_j^i (both at cost $d+1$ once s_j has been removed). Then we eliminate c_j^i (for a cost of d if e_i is already eliminated and $d+1$ otherwise). Finally if e_i is not yet eliminated we eliminate it for a cost of $d+1$.

Once we have done this, the vertex A has lost $k+n$ neighbors: all the e_i and the k vertices s_j that we have selected. Its remaining cost is $d+1$ so we eliminate it, which in turn means that all the remaining s_j have a cost of $d+1$ and we can eliminate them all (with their a_j^i, b_j^i and c_j^i attached).

Overall the cost of this elimination order is exactly V.

3) Proof that a solution to weighted-C^{++} implies a solution to set-cover: Suppose that we have an order $\prec$ such that the total cost is below V. Since V is the optimal cost for a marginal cost of k, the order $\prec$ cannot have a marginal cost higher than k otherwise its cost would be higher than V (see property 5). Knowing that $\prec$ has a marginal cost of at most k, we will extract a solution to set-cover.

First we notice that when A is eliminated, its cost is to $d+1+k-E_s+E_n+R_n$ where E_s the number of s_i eliminated, and R_n is the number of e_i remaining. However, as long as A is not eliminated, E_s is less than (or equal to) the marginal cost of all the vertices eliminated before A. Indeed, if s_j is eliminated while A is still present it is because we paid a marginal cost at least 1 to eliminate directly one of s_j or a^i_i or b^i_j or c^i_j for a $i \in S_j$. That is true because if A is present, then all those vertices have a cost of $d+2$ except c^i_j that ha a cost of $d+3$ or $d+2$ depending on whether e_i is eliminated.

Overall when we eliminate A we pay a marginal cost of $(k-E_s)+R_n$ where E_s is the number of sets eliminated and R_n is the number of integers not yet eliminated. The marginal cost of the order $\prec$ is at least the marginal cost of all vertices eliminated before A plus the marginal cost for A. Because the marginal cost of vertices removed before A is at least E_s, by adding the marginal cost of A we get a marginal cost bigger than $E_s + (k - E_s) + R_n = k + R_n$ which can be equal to k only if $R_n = 0$ which means all vertices corresponding to integers have been eliminated. Note that a vertex e_i can be eliminated because $i \in S_j$ for a vertex s_j eliminated or because e_i is directly eliminated but in that case we have to add a marginal cost of 1 specifically for this vertex. But in that case, it means that the marginal cost of all vertices before A includes the cost of removing this e_i which means that we cannot have a total marginal cost of k. Combining everything we get that if we have an order that has a marginal cost of k then we have k sets $S_{i_1}, \ldots, S_{i_k}$ covering all integers in $[\![1, n]\!]$.

REFERENCES

[1] N. Chiba and T. Nishizeki, "Arboricity and subgraph listing algorithms," *SIAM Journal on Computing*, 1985.

[2] T. Schank and D. Wagner, "Finding, counting and listing all triangles in large graphs, an experimental study," in *WEA*. Springer, 2005.

[3] M. Ortmann and U. Brandes, "Triangle listing algorithms: Back from the diversion," in *Proc. ALENEX*. SIAM, 2014.

[4] M. Danisch, O. D. Balalau, and M. Sozio, "Listing k-cliques in sparse real-world graphs," in *WWW*, 2018.

[5] M. Rudoy, "https://cstheory.stackexchange.com/q/38274," 2017.

[6] T. J. Schaefer, "The complexity of satisfiability problems," ser. STOC. ACM, 1978.