Into the Square: On the Complexity of Some Quadratic-time Solvable Problems

Michele Borassi, Pierluigi Crescenzi, Michel Habib

Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 322 (2016) 51–67

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Into the Square
On the Complexity of Some Quadratic-time Solvable Problems

Michele Borassi[1]

IMT Institute for Advanced Studies
Lucca, Italy

Pierluigi Crescenzi[2]

Dipartimento di Ingegneria dell'Informazione
University of Florence
Florence, Italy

Michel Habib[3]

LIAFA
University Paris Diderot
Paris, France

Abstract

We analyze several quadratic-time solvable problems, and we show that these problems are not solvable in *truly subquadratic* time (that is, in time $\mathcal{O}(n^{2-\epsilon})$ for some $\epsilon > 0$), unless the well known Strong Exponential Time Hypothesis (in short, SETH) is false. In particular, we start from an artificial quadratic-time solvable variation of the k-SAT problem (already introduced and used in the literature) and we will construct a web of Karp reductions, proving that a truly subquadratic-time algorithm for any of the problems in the web falsifies SETH. Some of these results were already known, while others are, as far as we know, new. The new problems considered are: computing the betweenness centrality of a vertex (the same result was proved independently by Abboud et al.), computing the minimum closeness centrality in a graph, computing the hyperbolicity of a graph, and computing the subset graph of a collection of sets. On the other hand, we will show that testing if a directed graph is transitive and testing if a graph is a comparability graph are subquadratic-time solvable (our algorithm is practical, since it is not based on intricate matrix multiplication algorithms).

Keywords: Quadratic-time algorithms, reductions, Strong Exponential Time Hypothesis, transitive closure

[1] Email: michele.borassi@imtlucca.it

[2] Email: pierluigi.crescenzi@unifi.it

[3] Email: habib@liafa.univ-paris-diderot.fr

http://dx.doi.org/10.1016/j.entcs.2016.03.005

1 Introduction

Since the very beginning of theoretical computer science and until recent years, the duality between NP-hard problems and polynomial-time solvable problems has been considered the threshold distinguishing "easy" from "hard" problems. However, polynomial-time algorithms might not be as efficient as one expects: for instance, in real-world networks with millions or billions of nodes, also quadratic-time algorithms might turn out to be too slow in practice, and a *truly subquadratic* algorithm would be a significant improvement, where an algorithm is said to be truly subquadratic if its time-complexity is $\mathcal{O}(n^{2-\epsilon})$ for some $\epsilon > 0$, where n is the input size. Following the main ideas behind the theory of NP-completeness, and not being able to show that a specific polynomial-time solvable problem might or might not admit a faster algorithm, researchers have recently started to prove that the existence of such an algorithm would imply faster solutions for other well-known and widely studied problems. As an example, a huge amount of work started from the analysis of the 3Sum problem, which consists of, given three sets A, B, and C of integers, deciding whether there exists $a \in A, b \in B$, and $c \in C$ such that $a + b + c = 0$. This problem has been widely studied and, as far as we know, the best known algorithms are subquadratic [6] (that is, their time complexity is $o(n^2)$), but not truly subquadratic (that is, their time complexity is *not* $\mathcal{O}(n^{2-\epsilon})$ for any $\epsilon > 0$). Recently, in [25], it was also proved that the decision tree complexity of the 3Sum problem is truly subquadratic, but this result is not sufficient to provide truly subquadratic algorithms. Hence, the 3Sum problem has become a starting point for proving the "hardness" of many other problems, especially in algebraic geometry (we refer the interested reader to [30]). Note that all these results do not deal with the notion of completeness, but they simply prove that "a problem is harder than another", using a kind of reduction between problems, relying on the fact that the easiest problem has been studied for years and no efficient algorithm has been found. A more recent development along this research direction is based on the Strong Exponential Time Hypothesis (SETH), used as a tool to prove the hardness of polynomial-time solvable problems. This hypothesis says that there is no algorithm for solving the k-Sat problem in time $\mathcal{O}((2-\epsilon)^n)$, where $\epsilon > 0$ does not depend on k [27]. Researchers have used this hypothesis in order to prove hardness results, starting from [38] where, among many other results, it is proved that the TwoDisjointSets problem is not solvable in truly subquadratic time, unless SETH is false (in [38], the TwoDisjointSets problem is named CooperativeSubsetQuery). In [40], instead, it is proved, for several problems, that either they all have truly subcubic algorithms, or none of them do. SETH is also used in [35], where it is proved that the hypothesis is falsified by the existence of algorithms for some problems with a specific polynomial-time complexity. More general results were published in [1], where betweenness centrality, reach centrality, radius, median, and diameter are considered, and where the parameter considered is the number of nodes and not the input size (for instance, the negative triangle problem, which is a starting point for several reductions, is solvable in $\mathcal{O}(m^{\frac{3}{2}})$, that is, in truly subquadratic time). Among the other results in this area, we find [36,37,2,10].

1.1 Our Contribution

Although many hardness results have been published so far, there is no survey containing them all (existing papers start from the k-Sat problem, instead of passing through simpler problems, like the TwoDisjointSets problem). As shown in Figure 1, in this paper we collect many reductions proved until now, we put them into a unified framework, and, most importantly, we use this analysis in order to prove *new* reductions. Note that, differently from other works [40,1], the parameter we consider is the input size and not the number of nodes: this analysis is more suited if sparse graphs are considered (for instance, most real-world complex networks). More specifically, the new reductions deal with the following problems.

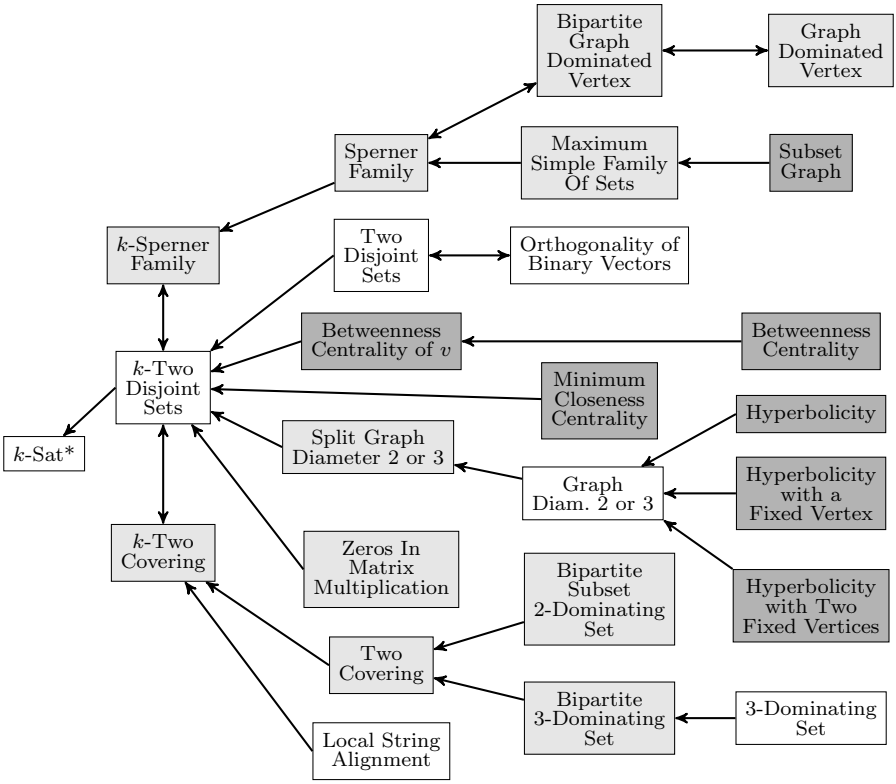

Fig. 1. The web of reductions provided by this paper. Gray problems indicate original results, white problems indicate already-known results, and light-gray problems indicate intermediate steps, which nevertheless can be interesting in their own rights, like the SplitGraphDiameter2Or3 problem.

BetweennessCentrality: the betweenness centrality is a widely used graph parameter related to community structures. In particular, the betweenness centrality of a vertex v is defined as the sum, over all pairs of nodes s and t different from v, of the ratios between the number of shortest paths from s to t passing through v and the number of shortest paths from s to t (for more information, see [22,33] and the references therein). Despite numerous works like [9,4,17], no truly subquadratic algorithm computing the betweenness centrality is known, even of a single vertex (BetweennessCentralityV). Moreover, in [4], it is

said that finding better results for approximating the betweenness centrality of all vertices is a "challenging open problem". Our analysis does not only prove that computing the betweenness centrality of all vertices in subquadratic time is against SETH, but it also presents the same result for computing the betweenness of a single vertex. The same result was proved independently in [1], through a reduction from the diameter computation problem.

MinimumClosenessCentrality: another fundamental parameter in graph analysis is closeness centrality, defined for the first time in 1950 [7] and recently reconsidered when analyzing real-world networks (for the interested reader, we refer to [31] and the references therein). The closeness centrality of a vertex v is defined as the inverse of the sum of all distances $d(v, w)$ from v to any other vertex w. This parameter has also raised algorithmic interest, and the most recent result is a very fast algorithm to approximate the closeness centrality of all vertices [13]. In this paper, we will prove the hardness of finding the "least central" vertex with respect to this measure. Simple consequences of this results are the hardness of computing in truly subquadratic time the closeness centrality of all vertices, or of extracting a "small enough" set containing all peripheral vertices.

Hyperbolicity: the Gromov hyperbolicity of a graph [24] (also known as δ-hyperbolicity [12]) recently got the attention of researchers in the field of network analysis [19], and has relations with the chordality of a graph [41], and with diameter and radius computation [12,16]. The hyperbolicity of a 4-tuple of vertices x, y, v, w is equal to $\delta(x, y, v, w) = \frac{1}{2}(S_1 - S_2)$, where S_1 is the maximum sum among $d(x, y) + d(v, w), d(x, v) + d(y, w), d(x, w) + d(y, v)$, and S_2 is the second maximum sum. The hyperbolicity of a graph is the maximum hyperbolicity over all 4-tuples. Hence, a naive algorithm needs time $\mathcal{O}(n^4)$. In [14], a better algorithm is provided: the time-complexity is still $\mathcal{O}(n^4)$, but it is much more efficient on real-world networks. Finally, this bound was improved in [21], where the authors describe an algorithm with time-complexity $\mathcal{O}(n^{3.69})$, which is a consequence of an algorithm that computes in time $\mathcal{O}(n^{2.69})$ the maximum hyperbolicity of a 4-tuple of the form (v, x_1, x_2, x_3), for a fixed vertex v. This paper provides also a hardness result for this problem, namely, that an algorithm running in time smaller than $\mathcal{O}(n^{2.05})$ for this latter problem would imply faster algorithms for (max,min) matrix product. Here, we will use another approach: we will prove that recognizing graphs of hyperbolicity 1 is not solvable in truly subquadratic time, unless SETH is false. Furthermore, the same result holds if we fix one vertex v, and even if we fix two vertices v, w in the 4-tuple (note that this latter problem is quadratic-time solvable).

SubsetGraph: given a collection $\mathcal{C}$ of subsets of a given ground set X, this problem asks to compute the subset graph of $\mathcal{C}$. The subset graph is defined as a graph whose vertices are the elements of $\mathcal{C}$, and which contains an edge (C, C') if $C \subseteq C'$. For this problem, the first subquadratic algorithm was proposed in [42], and in [34,18] matching lower bounds are proved. However, these lower bounds are based on the number of edges in the subset graph, which might be

quadratic with respect to the input size, apart from logarithmic factors. Our results show that the complexity of computing the subset graph is not due to the output size only, but it is intrinsic: in particular, we will prove that even deciding whether the subset graph has no edge is hard. This excludes the existence of a truly subquadratic algorithm to check if a solution is correct, or a truly subquadratic algorithm for instances where the output is sparse.

We will include in our web of reductions several "intermediate" problems. Among them, we have some variations of finding dominating sets in a graph (which is one of the 21 Karp's NP-complete problems [29]), detecting subsets with particular characteristics in a given collection (together with their "translations" in terms of graph properties), and distinguishing split graphs of diameter 2 and 3. This latter result is significant because it implies that it is hard to exactly compute the diameter of graphs in a class that contains split graphs: for instance, chordal graphs, where it is possible to approximate the diameter with an additive error of at most 1 [12]. The hardness proof of these results are already known, either as "hidden" steps in proofs already appeared in the literature, or as simple corollaries of previous work. However, we think that it is important to highlight these intermediate results, both because they might be interesting in their own right, and because they might be useful in the simplification of existing proofs and in the design of new proofs.

On the positive side, we prove that two problems admit subquadratic-time algorithms: testing if a graph is transitive and testing if it is a comparability graph. These two results will follow from a new analysis, that will prove that a known purely combinatorial algorithm [23] that computes the transitive closure has running-time $\mathcal{O}(m^{\frac{3}{2}} \log n)$. The same result can be obtained using rectangular matrix multiplication [39], together with some tricks: however, we focused on the combinatorial counterpart, because it does not contain big constants hidden in the $\mathcal{O}$ notation, and it is much more practical and easy to implement.

1.2 Structure of the Paper

In Section 2 we will define some of the problems we will analyze and we will introduce the notion of quasilinear reducibility, while in Section 3 we will prove the new hardness results (the proof of all other reductions are included in the extended version of this paper [8]). Section 4 provides positive results for the recognition of transitive graphs and of comparability graphs. Section 5 concludes the paper and provides some open problems.

2 Preliminary Definitions and Results

The starting point of our reductions is an "artificial" variation of k-Sat which is quadratic-time solvable, but not solvable in time $\mathcal{O}(n^{2-\epsilon})$ unless SETH is false.

Problem: k-Sat*.
Input: two sets of variables $X = \{x_i\}$, $Y = \{y_j\}$ of the same size, a set C of

clauses over these variables, such that each clause has at most size k, and the two power sets $\mathcal{P}(X), \mathcal{P}(Y)$ (used to change the input size).

Output: **True** if there is an evaluation of all variables that satisfies all clauses, **False** otherwise.

It should be noticed that this problem differs from the classic one only by the input size. This way, a quadratic-time algorithm exists (trying all possible evaluations). However, if m is the number of variables and $n = 2^{\frac{m}{2}}$ is the input size, since both X and Y have size $\frac{m}{2}$, an algorithm running in time $\mathcal{O}(n^{2-\epsilon})$ with ϵ not depending on k would imply an algorithm solving k-SAT in time $\mathcal{O}(2^{\frac{m}{2}(2-\epsilon)}) = \mathcal{O}\left(\left(2^{\frac{2-\epsilon}{2}}\right)^m\right)$, where m is the number of variables. This latter result contradicts SETH. After defining the "starting" problem, we need to define the reducibility notion we are going to use.

Definition 2.1 A quasilinear Karp reduction from a problem $\mathcal{P}$ to a problem $\mathcal{Q}$ is a function Φ from instances of $\mathcal{P}$ to instances of $\mathcal{Q}$ satisfying, for every instance I of $\mathcal{P}$, the following two properties.

- $\Phi(I)$ can be computed in time $\tilde{\mathcal{O}}(s(I))$, where $s(I)$ is the size of input I. [4]
- I and $\Phi(I)$ have the same output (if the output is not boolean, we will also require a linear-time computable function that transforms the output of $\Phi(I)$ into the output of I).

If there exists a quasilinear reduction from $\mathcal{P}$ to $\mathcal{Q}$, we will write $\mathcal{P} \leq_{ql} \mathcal{Q}$.

Remark 2.2 If $\mathcal{P} \leq_{ql} \mathcal{Q}$ and there is an algorithm solving $\mathcal{Q}$ in time $\tilde{\mathcal{O}}(n^{2-\epsilon})$ for some ϵ, then $\mathcal{P}$ can be solved in time $\tilde{\mathcal{O}}(n^{2-\epsilon})$.

The reductions that we will provide are summarized in Figure 1 (the definition of the problems is either in the text, or in Appendix A). The previous remark implies that an $\mathcal{O}(n^{2-\epsilon})$-time algorithm for any of these problems would falsify SETH. Although many reductions in the figure were already known, some reductions are new, and they deal with the following graph parameters, widely used in the field of network analysis.

Definition 2.3 Given a graph $G = (V, E)$, the betweenness centrality of a vertex v is,

$$\sum_{s,t \in V, s \neq v \neq t} \frac{\text{number of shortest paths from } s \text{ to } t \text{ through } v}{\text{number of shortest paths from } s \text{ to } t}.$$

Definition 2.4 Given a graph $G = (V, E)$, the closeness centrality of a vertex v is defined as $\frac{1}{\sum_{w \in V} d(v,w)}$, where $d(v, w)$ is the distance between v and w.

Definition 2.5 Given a graph $G = (V, E)$, the hyperbolicity of a 4-tuple of vertices x, y, v, w is $\delta(x, y, v, w) = \frac{1}{2}(S_1 - S_2)$, where S_1 is the maximum sum among $d(x, y) +$

[4] By $\tilde{\mathcal{O}}(f(n))$ we mean $\mathcal{O}(f(n) \log^k n)$ for some fixed k.

$d(v,w), d(x,v)+d(y,w), d(x,w)+d(y,v)$, and S_2 is the second maximum sum. The hyperbolicity of a graph is the maximum hyperbolicity over all 4-tuples.

In particular, we will consider the following problems.

Problem: BetweennessCentralityV.
Input: a graph $G = (V, E)$ and a vertex $v \in V$.
Output: the betweenness centrality of v.

Problem: BetweennessCentrality.
Input: a graph $G = (V, E)$.
Output: the betweenness centrality of each vertex $v \in V$.

Problem: MinimumClosenessCentrality.
Input: a graph $G = (V, E)$ and a threshold σ.
Output: **True** if there exists a vertex with closeness centrality smaller than σ, **False** otherwise.

Problem: Hyperbolicity.
Input: a graph $G = (V, E)$.
Output: the hyperbolicity of G.

Problem: HyperWithAFixedVertex.
Input: a graph $G = (V, E)$ and a vertex x.
Output: the maximum over each triple of vertices y, v, w of $\delta(x, y, v, w)$.

Problem: HyperWith2FixedVertices.
Input: a graph $G = (V, E)$ and two vertices x, y.
Output: the maximum over each pair of vertices v, w of $\delta(x, y, v, w)$.

Problem: SubsetGraph.
Input: a set X and a collection $\mathcal{C}$ of subsets of X.
Output: the graph $G = (\mathcal{C}, E)$, where, for each $C, C' \in \mathcal{C}$, $(C, C') \in E$ if and only if $C \subseteq C'$.

3 The New Hardness Results

In this section we prove the new (that is, gray) reductions included in Figure 1, apart from betweenness centrality, which was independently proved also in [1]. All the other proofs are available in the extended version of this paper [8]. All these reductions start from the k-TwoDisjointSets problem.

Problem: k-TwoDisjointSets.
Input: a set X and a collection $\mathcal{C}$ of subsets of X such that $|X| < \log^k(|\mathcal{C}|)$.

Output: **True** if there are two disjoint sets $C, C' \in \mathcal{C}$, **False** otherwise.

The hardness of this problem is proved by the following theorem, whose proof also appears in the extended version of this paper [8].

Theorem 3.1 ([38]) *For each k, k-Sat$^* \leq_{ql} k$-TwoDisjointSets.*

3.1 Hardness of Closeness

Theorem 3.2 *k-TwoDisjointSets $\leq_{ql}$ MinimumClosenessCentrality.*

Proof. Instead of minimizing the closeness centrality, we will try to maximize the *farness*, which is the inverse of the closeness centrality, that is, the sum of all distances from v to another vertex. We will build a graph where the vertices with biggest farness correspond to sets in $\mathcal{C}$, and the value of the farness does not depend on the corresponding set, if this set is not disjoint to any other set. If this latter condition is not satisfied, then the farness of the vertex is bigger. In particular, let us consider an instance $I = (X, \mathcal{C})$ of k-TwoDisjointSets, let us assume $X \notin \mathcal{C}$, and let us build a graph in the following way:

- $V = V_1 \cup V_1' \cup V_2 \cup V_3$, where V_1 and V_1' are two disjoint copies of X, $V_2 = \mathcal{C}$ and $V_3 = \{(x, C) \in X \times \mathcal{C} : x \notin C\}$;
- $V_1 \cup V_1'$ is a clique;
- for $x \in V_1 \cup V_1'$ and $C \in V_2$, there is an edge from x to C if and only if $x \in C$;
- for each $(x, C) \in V_3$ and $C' \in V_2$, there is a link between these vertices if and only if $C = C'$.

Claim: the vertex with maximum farness is in V_3.

Proof. [Proof of claim] For each vertex $v \in V_2$, consider a vertex $w \in V_3$ linked to v. It is clear that all shortest paths from w to any other vertex must pass through v (which is the only vertex linked to w). This means that the farness of w is bigger than the farness of v.

For each vertex $v \in V_1 \cup V_1'$, let us consider a vertex $w \in V_2$ linked to v. The only vertices which are closer to w than to v are the vertices in V_3 attached to w, because each other neighbor of w is a neighbor of v. These vertices influence the farness of w by $|X| - |C|$, where C is the set corresponding to w. However, there are $2(|X| - |C|)$ vertices in $V_1 \cup V_1'$ linked to v and not to w (the elements not in C): this proves that the farness of w is bigger than the farness of v. □

At this point, let us consider the farness of vertices in V_3. In particular, let (x, C) be an element of V_3 such that $C \cap C' \neq \emptyset$ for each C': the farness of (x, C) can be exactly computed by considering the classes of vertices in Table 1.

Before computing the farness of (x, C), we compute $\sum_{C' \neq C} |X| - |C'| = (|\mathcal{C}| - 1)|X| - \sum_{C' \neq C} |C'| = (|\mathcal{C}| - 1)|X| - \sum_{C' \in \mathcal{C}} |C'| + |C|$. The farness of (x, C) is then:

Table 1
The distance from (x, C) to another vertex

Set	Kind of vertex	Number	distance from (x, C)
$V_1 \cup V_1'$	vertex in C	$2\|C\|$	2
$V_1 \cup V_1'$	vertex outside C	$2(\|X\| - \|C\|)$	3
V_2	C	1	1
V_2	$C' \neq C$	$\|\mathcal{C}\| - 1$	3
V_3	(x', C)	$\|X\| - \|C\|$	2
V_3	$(x', C'), C' \neq C$	$\sum_{C' \neq C} \|X\| - \|C'\|$	4

$$4|C| + 6(|X| - |C|) + 1 + 3(|\mathcal{C}| - 1) + 2(|X| - |C|) + 4\left(\sum_{C' \neq C} |X| - |C'|\right) =$$

$$= 4|C| + 8|X| - 8|C| + 1 + 3|\mathcal{C}| - 3 + 4(|\mathcal{C}| - 1)|X| - 4\sum_{C' \in \mathcal{C}} |C'| + 4|C| =$$

$$= 4|\mathcal{C}||X| - 4\left(\sum_{C' \in \mathcal{C}} |C'|\right) + 3|\mathcal{C}| + 4|X| - 2.$$

Note that this value does not depend on the particular (x, C) chosen (this was the main goal of our construction). It is clear that if $C \cap C' = \emptyset$, then the farness of each vertex (x, C) and (x, C') is bigger than the value previously computed.

As a consequence, there are two disjoint sets if and only if in the whole graph there is a vertex with farness bigger than $4|\mathcal{C}||X| - 4(\sum_{C' \in \mathcal{C}} |C'|) + 3|\mathcal{C}| + 4|X| - 2$, and both this value and the underlying graph can be computed in linear time. □

3.2 Hardness of Hyperbolicity

In order to prove the hardness of the hyperbolicity problems, we will "pass through" the following problem, related to the computation of the diameter of a graph (that is, the maximum distance between two vertices).

Problem: GraphDiameter2Or3.
Input: a graph G.
Output: **True** if G has diameter 2, **False** otherwise.

Theorem 3.3 ([36]) k-TwoDisjointSets $\leq_{ql}$ GraphDiameter2Or3.

The proof of the above theorem in [36] contains all the steps from the k-Sat problem to the GraphDiameter2Or3 problem. These steps are, instead, separated in the proofs appearing in the extended version of this paper [8], in order to show the hardness of an interesting intermediate problem, that is, the computation of the diameter in the case of split graphs (and, therefore, in the case of chordal graphs). The remaining part of this section is devoted to the proof that all hyperbolicity problems are harder than GraphDiameter2Or3. Let us start by stating two lemmas.

Lemma 3.4 ([14], Lemma 5) *Let x, y, v, w be a 4-tuple, and let $S_1 = d(x, y) + d(v, w)$. Then $\delta(x, y, v, w) \leq \frac{d(x,y)}{2}$ and $\delta(x, y, v, w) \leq \frac{d(v,w)}{2}$.*

Lemma 3.5 ([15], Lemma 3.1) *For each quadruple x, y, v, w, the hyperbolicity $\delta(x, y, v, w)$ is smaller than or equal to the minimum distance between two vertices in the quadruple.*

The construction used in the proof of the hardness of all hyperbolicity problems is the same. Given an input graph $G = (V, E)$ for the GraphDiameter2Or3 problem, the corresponding graph for the hyperbolicity problem is $H = (V', E')$, defined as follows. The set V' is $\{x\} \cup V_x \cup \tilde{V} \cup \{z\} \cup V_y \cup \{y\}$, where V_x, $\tilde{V}$ and V_y are disjoint copies of V, x, y, and z are new vertices. The set E' is defined as follows:

- x is connected to every vertex in V_x, y is connected to every vertex in V_y, and z is connected to every vertex in V_x and to every vertex in V_y;
- corresponding vertices in V_x and $\tilde{V}$ and corresponding vertices in $\tilde{V}$ and V_y are connected;
- if (v, w) is an edge of G, then the copies of v and w in $\tilde{V}$ are linked.

By this construction, if x, y are the aforementioned vertices and v, w is a pair of vertices in $\tilde{V}$, then $2\delta(x, y, v, w) = d(x, y) + d(v, w) - \max(d(x, v) + d(y, w), d(x, w) + d(y, v)) = 4 + d(v, w) - 4 = d(v, w)$, and $d(x, y) + d(v, w)$ is the biggest sum. Furthermore, by construction, the shortest paths between two vertices in $\tilde{V}$ remain in $\tilde{V}$, or they have length at least 4. This means that $\max_{v,w \in \tilde{V}} \delta(x, y, v, w) = \max_{v,w \in \tilde{V}} \frac{d(v,w)}{2}$ is at most 1 if and only if the maximum distance between two vertices in $\tilde{V}$ is at most 2, if and only if the diameter of G is at most 2. In order to conclude our reduction, we prove that other 4-tuples in H have smaller hyperbolicity, so that the hyperbolicity of H is exactly half the diameter of G.

Lemma 3.6 *The hyperbolicity of each 4-tuple in H which is not of the form x, y, v, w with $v, w \in \tilde{V}$ is at most 1.*

Proof. Let us consider a 4-tuple v, w, v', w' having hyperbolicity bigger than 1, and let us first prove that x and y belong to this 4-tuple. This happens because, by Lemma 3.5, the distance among any pair of these vertices is at least 2, and $2\delta(v, w, v', w') \leq S_1 - 4$. If we want the hyperbolicity to be bigger than 1, we need S_1 to be bigger then 6 and a distance in S_1 must be at least 4. Since the only such distance in H is $d(x, y)$, x and y must be in the 4-tuple v, w, v', w', as we wanted to prove. We conclude the proof by showing that the hyperbolicity of a 4-tuple x, y, v, w with $v \notin \tilde{V}$ is at most 1. If $v = z$, this holds because $d(z, w) \leq 2$ for each w, and hence $\delta(x, y, z, w) \leq 1$ by Lemma 3.4. If $v \in V_x$ (resp. $v \in V_y$), then $\delta(x, y, v, w) \leq 1$ because $d(x, v) = 1$ (resp. $d(y, v) = 1$), and we may use Lemma 3.5. □

Thanks to this lemma, we may conclude our reductions, because the hyperbolicity of H is bigger than 1 if and only if the diameter of G is bigger than 2. Since we

know that x and y belong to the 4-tuple with maximum hyperbolicity, the hardness of all hyperbolicity problems follows from the hardness of the diameter computation.

Theorem 3.7 GraphDiameter2Or3 $\leq_{ql}$ HyperWith2FixedVertices.
GraphDiameter2Or3 $\leq_{ql}$ HyperWithAFixedVertex.
GraphDiameter2Or3 $\leq_{ql}$ Hyperbolicity.

3.3 Hardness of Computing the Subset Graph

In order to prove the hardness of computing the subset graph, we will show that it is hard to decide whether the subset graph is empty. This happens if and only if, given a ground set X and a collection $\mathcal{C}$ of subsets of X, there is no pair of elements $C, C' \in \mathcal{C}$ such that $C \subseteq C'$. If we restrict ourselves to the case when X is small with respect to $\mathcal{C}$, we have reduced the SubsetGraph problem to the following problem.

Problem:	k-SpernerFamily.
Input:	a set X and a collection $\mathcal{C}$ of subsets of X such that $\lvert X\rvert < \log^k(\lvert\mathcal{C}\rvert)$.
Output:	**True** if there are two sets $C, C' \in \mathcal{C}$ such that $C \subseteq C'$, **False** otherwise.

We will conclude the proof by showing the hardness of this problem, through the following theorem.

Theorem 3.8 k-TwoDisjointSets $\leq_{ql}$ k-SpernerFamily.

Proof. Consider an instance $I = (X, \mathcal{C})$ of k-TwoDisjointSets. First of all, we define $\Phi'(I) = (X, \mathcal{C}')$, where $\mathcal{C}' = \mathcal{C} \cup \bar{\mathcal{C}}$, and $\bar{\mathcal{C}} := \{X - C : C \in \mathcal{C}\}$ (which is not the correct definition, but we will see how to adapt it). If we find two sets $C \in \mathcal{C}, C' \in \bar{\mathcal{C}}$ such that $C \subseteq C'$, we know that C and $X - C'$ are disjoint and in $\mathcal{C}$, so we have found a solution. However, when we solve the k-SpernerFamily problem, we are not sure that $C \in \mathcal{C}$ and $C' \in \bar{\mathcal{C}}$: we will solve this issue by slightly modifying X' and $\mathcal{C}'$. In order to avoid the existence of $C, C' \in \mathcal{C}$ such that $C \subseteq C'$, we define $k := \lceil \log_2(\lvert\mathcal{C}\rvert) \rceil$, and we add two sets $Y = \{y_1, \dots, y_k\}$ and $Z = \{z_1, \dots, z_k\}$ to X. Then, we add Y and Z to each set in $\bar{\mathcal{C}}$ and we add to each element $C \in \mathcal{C}$ some y_i and some z_j, so that no element of $\mathcal{C}$ can dominate another element in $\mathcal{C}$ (for example, we may associate each set C with a unique binary number with k bits, and code this number using y_i as zeros and z_j as ones). This way, we preserve dominations between sets in $\bar{\mathcal{C}}$ and sets in $\mathcal{C}$, and we also avoid that a set in $\bar{\mathcal{C}}$ dominates a set in $\bar{\mathcal{C}}$. Finally, we need to avoid that two sets in $\bar{\mathcal{C}}$ dominate each other: it is enough to make the same construction adding new sets Y' and Z' of logarithmic size, and use them to uniquely code any element in $\bar{\mathcal{C}}$. In order to preserve dominations between $\mathcal{C}$ and $\bar{\mathcal{C}}$, none of the elements in Y' and Z' is added to subsets in $\mathcal{C}$. □

4 Transitive Closure and Comparability Graph Test

In this section, we provide a new analysis of an old algorithm that computes the transitive closure of a graph: we will prove that its time-complexity is $\mathcal{O}(m_{tc}^{\frac{3}{2}} \log n)$, where m_{tc} is the number of edges in the transitive closure. Consequently, we will be able to test if a graph is transitive in time $\mathcal{O}(m^{\frac{3}{2}} \log n)$, and to recognize comparability graphs in the same amount of time, combining the transitive closure algorithm with a result in [32]. The definitions of the problems follow.

Problem: TransitiveClosure.
Input: a directed acyclic graph $G = (V, E)$.
Output: the transitive closure of G, that is, the minimum subgraph of G such that if there is a path from a vertex v to a vertex w, $E(v, w)$ holds.

Problem: ComparabilityRecognition.
Input: an undirected graph G.
Output: **True** if there is a transitive orientation of G, **False** otherwise.

First of all, let us restrict to the acyclic case: if the graph is not acyclic, the transitive closure can be easily deduced from the transitive closure of its strongly connected component graph, which is acyclic (see [5], Section 4.3 for more information).

Algorithm 1: computing the transitive closure of a graph.

Input: a directed acyclic graph $G = (V, N)$, stored as a list of out-neighbors $N(v_i)$ for each vertex $v_i \in V$.
Output: the transitive closure $G' = (V, N')$ of G, stored as a list of out-neighbors $N'(v_i)$ for each vertex $v_i \in V$.

```
1 Find a topological ordering (v_1, ..., v_n) of V;
2 for i = n to 1 do
3 |   N'(v_i) = N(v_i);
4 |   for w in N(v_i) do
5 |   |   N'(v_i) = N'(v_i) ∪ N'(w);
6 return (V, N');
```

Algorithm 1 provides the pseudo-code of the algorithm, published for the first time in [23]. The correctness of the algorithm follows from the fact that vertices are analyzed in reverse topological order: when the algorithm processes v_i in the outer `for` loop, $N'(w)$ is already computed for each $w \in N'(v_i)$.

The running-time analysis follows from the following theorem.

Theorem 4.1 *It is possible to implement Algorithm 1 with time-complexity $\mathcal{O}(m_{tc}^{\frac{3}{2}} \log n)$, where m_{tc} is the number of edges in the transitive closure.*

Proof. All steps of Algorithm 1 apart from Line 5 need at most time $\mathcal{O}(m)$. For

the analysis of Line 5, we observe that, if $n_{out}(w)$ is the out-degree of w in the transitive closure, a single step can be done in time $\mathcal{O}(|N'(w)| \log(|N'(v_i)|)) = \mathcal{O}(n_{out}(w) \log n)$, if we implement $N'(v_i)$ as a self-balanced binary search tree (the computation of these trees and their update has no impact on the asymptotic running-time).

Moreover, a vertex w is processed in the inner `for` loop at most $n_{in}(w)$ times, where $n_{in}(w)$ is the in-degree of w in the transitive closure (actually, it is at most the in-degree of w in the original graph, or even in its transitive reduction). We conclude that the time needed to perform Line 5 is at most $\mathcal{O}(\sum_{w \in V} n_{in}(w) n_{out}(w) \log n)$.

We claim that $\sum_{w \in V} n_{in}(w) n_{out}(w)$ is at most the number of triangles in the transitive closure (seen as an undirected graph). Indeed, if x is an in-neighbor of w and y is an out-neighbor of w, xy must be an edge (the graph is transitive), and xwy forms a triangle. We are not counting twice the same triangle when analyzing different vertices because w is always between x and y in a topological order. The claim follows.

Since the number of triangles in a graph with m edges is at most $\mathcal{O}(m^{\frac{3}{2}})$ (see for instance [28] for a proof), the theorem follows. □

Remark 4.2 In the analysis, the bound $\sum_{w \in V} n_{in}(w) n_{out}(w) = \mathcal{O}(m^{\frac{3}{2}})$ is tight: if we consider a directed acyclic complete graph, $\sum_{w \in V} n_{in}(w) n_{out}(w) = \sum_{i=1}^{n} i(n - i) = \mathcal{O}(n^3) = \mathcal{O}(m^{\frac{3}{2}})$.

Corollary 4.3 *It is possible to check if a graph is transitive in time* $\mathcal{O}(m^{\frac{3}{2}} \log n)$.

Proof. To test if a graph is transitive, we run the previous algorithm for $\mathcal{O}(m^{\frac{3}{2}} \log n)$ time, and stopping if an edge is added. If after $\mathcal{O}(m^{\frac{3}{2}} \log n)$ the algorithm does not stop, m_{tc} must be bigger than m, and the graph is not transitive.□

Corollary 4.4 *It is possible to check if a graph is a comparability graph in time* $\mathcal{O}\left(m^{\frac{3}{2}} \log n\right)$.

Proof. It is known that, if a graph G is a comparability graph, then it is possible to compute a transitive orientation of G in linear time [32] (for a simpler algorithm running in $\mathcal{O}(n + m \log n)$ see [26]). Let H be the output of this algorithm, and let us apply the previous corollary on H: if it is transitive, G is clearly a comparability graph, otherwise the graph G is not a comparability graph, since the transitive orientation algorithm did not provide a transitive orientation. The running time of comparability graph recognition coincides with the running time of the transitive closure algorithm, since all other steps are faster. This proves the theorem. □

Finally, we observe that our transitive closure algorithm yields a boolean matrix multiplication algorithm with running-time $\mathcal{O}(m^{1.5})$, where m is the number of ones in the result, since computing the transitive closure is equivalent to computing boolean matrix multiplication [20].

5 Conclusions and Open Problems

In this paper, we have analyzed several hardness results for quadratic-time solvable problems, and we have proved that some problems are subquadratic-time solvable. This work can be seen as a starting point to develop many more reductions and to include inside its framework several new problems. Among these new problems, the computation of three widely studied graph parameters is included. It would be really interesting to connect these results with the existing reductions related to the 3Sum problem: until now, we have not been able to connect this problem with SETH. However, it is possible to connect it with other problems: for example, it is known that the local alignment of strings problem is harder than 3Sum [3]. Another open problem deals with the computation of the diameter in the case of planar graphs and with the computation of the radius of a graph. This latter measure is similar to the diameter, but it looks somehow "easier" to compute: for example, the radius of chordal graphs is linear-time computable [11]. The question is whether also these problems can be inserted in our framework of quadratic-time hard problems, or a truly subquadratic algorithm exists for them. Among other problems that are not in our framework, but have no truly subquadratic time algorithm, we find the computation of the transitive reduction of a directed graph (a "converse" of the transitive closure), finding maximum flows in networks or finding maximum matchings in bipartite weighted graphs. Finally, it would be nice to check if it is possible to remove the $\log n$ in the running time of the transitive closure algorithm (also improving the polynomial part would be interesting, but we think it is much more difficult).

References

[1] Abboud, A., F. Grandoni and V. V. Williams, *Subcubic equivalences between graph centrality problems, APSP and diameter*, 26th ACM/SIAM Symposium on Discrete Algorithms (2015), pp. 1681–1697.

[2] Abboud, A. and V. V. Williams, *Popular conjectures imply strong lower bounds for dynamic problems*, Proceedings of the 55th Annual Symposium on Foundations of Computer Science (FOCS) (2014).

[3] Abboud, A., V. V. Williams and O. Weimann, *Consequences of Faster Alignment of Sequences*, ICALP (2014).

[4] Bader, D. A., S. Kintali, K. Madduri and M. Mihail, *Approximating betweenness centrality*, The 5th Workshop on Algorithms and Models for the Web-Graph (2007).

[5] Bang-Jensen, J. and G. Gutin, "Digraphs Theory, Algorithms and Applications," August, Springer, 2008.

[6] Baran, I., E. D. Demaine and M. Patrascu, *Subquadratic algorithms for 3SUM*, Algorithmica **50** (2008), pp. 584–596.

[7] Bavelas, A., *Communication patterns in task-oriented groups*, Journal of the Acoustical Society of America **22** (1950), pp. 725–730.

[8] Borassi, M., P. Crescenzi and M. Habib, *Into the Square - On the Complexity of Some Quadratic-Time Solvable Problems*, Preprint on arXiv (2014).

[9] Brandes, U., *A faster algorithm for betweenness centrality*, The Journal of Mathematical Sociology **25** (2001), pp. 163–177.

[10] Bringmann, K., *Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails*, Proceedings of the 55th Annual Symposium on Foundations of Computer Science (FOCS) (2014), pp. 661–670.

[11] Chepoi, V. and F. F. Dragan, *A Linear-Time Algorithm for Finding a Central Vertex of a Chordal Graph*, in: *ESA*, 1994, pp. 159–170.

[12] Chepoi, V., F. F. Dragan, B. Estellon, M. Habib and Y. Vaxès, *Diameters, centers, and approximating trees of delta-hyperbolicgeodesic spaces and graphs*, Proceedings of the twenty . . . (2008), pp. 59–68.

[13] Cohen, E., D. Delling, T. Pajor and R. F. Werneck, *Computing classic closeness centrality, at scale*, in: *Proceedings of the Second ACM Conference on Online Social Networks*, COSN '14 (2014), pp. 37–50.

[14] Cohen, N., D. Coudert and A. Lancin, *Exact and approximate algorithms for computing the hyperbolicity of large-scale graphs*, Technical Report [Research Report] RR-8074, INRIA (2012).

[15] Cohen, N., D. Coudert and A. Lancin, *On computing the Gromov hyperbolicity*, ACM Journal of Experimental Algorithms (2015).

[16] Crescenzi, P., R. Grossi, M. Habib, L. Lanzi and A. Marino, *On computing the diameter of real-world undirected graphs*, Theoretical Computer Science **514** (2013), pp. 84–95.

[17] Edmonds, N., T. Hoefler and A. Lumsdaine, *A space-efficient parallel algorithm for computing betweenness centrality in distributed memory*, 2010 International Conference on High Performance Computing (2010), pp. 1–10.

[18] Elmasry, A., *The Subset Partial Order: Computing and Combinatorics.*, ANALCO (2010), pp. 27–33.

[19] Fang, W., *On Hyperbolic Geometry Structure of Complex Networks* (2011).

[20] Fischer, M. J. and A. R. Meyer, *Boolean matrix multiplication and transitive closure*, Switching and Automata Theory (1971), pp. 2–4.

[21] Fournier, H., A. Ismail and A. Vigneron, *Computing the Gromov hyperbolicity of a discrete metric space*, arXiv preprint arXiv:1210.3323 (2012), pp. 1–6.

[22] Freeman, L. C., *A set of measures of centrality based on betweenness*, Sociometry (1977).

[23] Goralcíkova, A. and V. Koubek, *A reduct-and-closure algorithm for graphs*, Mathematical Foundations of Computer Science 1979 (1979), pp. 301–307.

[24] Gromov, M., "Hyperbolic groups," Springer, 1987.

[25] Gronlund, A. and S. Pettie, *Threesomes, Degenerates, and Love Triangles*, Proceedings of the 55th Annual Symposium on Foundations of Computer Science (FOCS) (2014), pp. 621–630.

[26] Habib, M., R. McConnell, C. Paul and L. Viennot, *Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing*, Theoretical Computer Science **234** (2000), pp. 59–84.

[27] Impagliazzo, R., R. Paturi and F. Zane, *Which Problems Have Strongly Exponential Complexity?*, Journal of Computer and System Sciences **63** (2001), pp. 512–530.

[28] Itai, A. and M. Rodeh, *Finding a minimum circuit in a graph*, SIAM Journal on Computing (1978).

[29] Karp, R. M., *Reducibility among combinatorial problems*, in: *Complexity of Computer Computations*, Springer, 1972 pp. 85–103.

[30] King, J., *A survey of 3SUM-hard problems* (2004).

[31] Latora, V. and M. Marchiori, *A measure of centrality based on network efficiency*, New Journal of Physics (2007).

[32] McConnell, R. M. and J. Spinrad, *Modular decomposition and transitive orientation*, Discrete Mathematics **201** (1999), pp. 189–241.

[33] Newman, M. E. J., "Networks: An Introduction," OUP Oxford, 2010.

[34] Pritchard, P., *On computing the subset graph of a collection of sets*, Journal of Algorithms (1999), pp. 1–14.

[35] Pătraşcu, M. and R. Williams, *On the possibility of faster SAT algorithms*, ACM-SIAM Symposium on Discrete Algorithms (2010).

[36] Roditty, L. and V. V. Williams, *Fast approximation algorithms for the diameter and radius of sparse graphs*, Proceedings of the 45th annual ACM symposium on Symposium on theory of computing - STOC '13 (2013), p. 515.

[37] Roditty, L. and V. V. Williams, *Approximating the diameter of a graph*, Preprint on arXiv (2014).

[38] Williams, R., *A new algorithm for optimal 2-constraint satisfaction and its implications*, Theoretical Computer Science **348** (2005), pp. 357–365.

[39] Williams, R. and H. Yu, *Finding orthogonal vectors in discrete structures.*, SODA (2014).

[40] Williams, V. V. and R. Williams, *Subcubic Equivalences between Path, Matrix and Triangle Problems*, 2010 IEEE 51st Annual Symposium on Foundations of Computer Science (2010), pp. 645–654.

[41] Wu, Y. and C. Zhang, *Hyperbolicity and chordality of a graph*, the electronic journal of combinatorics **18** (2011), pp. 1–22.

[42] Yellin, D. M. and C. S. Jutla, *Finding Extremal Sets in Less Than Quadratic Time*, Information Processing Letters **48** (1993), pp. 29–34.

A Problem Definitions

In this appendix, we will precisely define all the problems that ar not defined in the text.

Problem: Bipartite3DominatingSet.
Input: a bipartite graph $G = (V, E)$.
Output: a triple v, w, x such that $V = N(v) \cup N(w) \cup N(x) \cup \{v, w, x\}$, if it exists.

Problem: BipartiteSubset2DominatingSet.
Input: a graph $G = (V, E)$ and a subset $V' \subseteq V$.
Output: a pair v, w such that $V' = N(v) \cup N(w) \cup \{v, w\}$, if it exists.

Problem: BipGDominatedVertex.
Input: a bipartite graph (V_1, V_2, E).
Output: **True** if there are vertices v, w such that $N(v) \supseteq N(w)$, **False** otherwise.

Problem: 3DominatingSet.
Input: a graph $G = (V, E)$.
Output: a triple v, w, x such that $V = N(v) \cup N(w) \cup N(x) \cup \{v, w, x\}$, if it exists.

Problem: GraphDiameter2Or3.
Input: a graph G.
Output: **True** if G has diameter 2, **False** otherwise.

Problem: GraphDominatedVertex.

Input: a graph (V, E).
Output: **True** if there are vertices v, w such that $N(v) \supseteq N(w)$, **False** otherwise.

Problem: k-TwoCovering.
Input: a set X and a collection $\mathcal{C}$ of subsets of X such that $|X| < \log^k(|\mathcal{C}|)$.
Output: **True** if there are two sets $C, C' \in \mathcal{C}$ such that $X = C \cup C'$, **False** otherwise.

Problem: LocalStringAlign.
Input: two binary strings with a symbol $*$ that may replace any character.
Output: The longest common substring of the two strings.

Problem: MaximalElementsFamily.
Input: a set X and a collection $\mathcal{C}$ of subsets of X.
Output: the maximum simple family of subsets of X (simple means without inclusion).

Problem: OrthogonalityBinaryVectors.
Input: a collection of binary vectors.
Output: **True**, if there are two orthogonal vectors, **False** otherwise.

Problem: SpernerFamily.
Input: a set X and a collection $\mathcal{C}$ of subsets of X.
Output: **True** if there are two sets $C, C' \in \mathcal{C}$ such that $C \subseteq C'$, **False** otherwise.

Problem: TwoCovering.
Input: a set X and a collection $\mathcal{C}$ of subsets of X.
Output: **True** if there are two sets $C, C' \in \mathcal{C}$ such that $X = C \cup C'$, **False** otherwise.

Problem: TwoDisjointSets.
Input: a set X and a collection $\mathcal{C}$ of subsets of X.
Output: **True** if there are two disjoint sets $C, C' \in \mathcal{C}$, **False** otherwise.

Problem: ZerosMatrixMultiplication.
Input: two $(0-1)$-matrices M, M' implemented as adjacency lists.
Output: **True** if MM' contains a 0, **False** otherwise.