Approximating the Sparsest k-Subgraph in Chordal Graphs*

Rémi Watrigant, Marin Bougeret, and Rodolphe Giroudeau

LIRMM, Université Montpellier 2, France

Abstract. Given a simple undirected graph $G = (V, E)$ and an integer $k < |V|$, the Sparsest k-Subgraph problem asks for a set of k vertices which induces the minimum number of edges. As a generalization of the classical independent set problem, Sparsest k-Subgraph is $\mathcal{NP}$-hard and even not approximable unless $\mathcal{P} = \mathcal{NP}$ in general graphs. Thus, we investigate Sparsest k-Subgraph in graph classes where independent set is polynomial-time solvable, such as subclasses of perfect graphs. Our two main results are the $\mathcal{NP}$-hardness of Sparsest k-Subgraph on chordal graphs, and a greedy 2-approximation algorithm. Finally, we also show how to derive a $PTAS$ for Sparsest k-Subgraph on proper interval graphs.

1 Introduction

1.1 Related Problems

Given a simple undirected graph $G = (V, E)$ and an integer $k < |V|$, the Sparsest k-Subgraph problem asks for a set of k vertices which induces[1] the minimum number of edges. It appears that this problem falls into the family of *cardinality constrained optimization problems*, introduced by [7], and is more precisely a generalization of the so-called independent set problem. This observation immediately implies that Sparsest k-Subgraph is $\mathcal{NP}$-hard and even not approximable in general graphs unless $\mathcal{P} = \mathcal{NP}$, as the optimal value is 0 whenever there is an independent set of size k. Thus, we only consider Sparsest k-Subgraph in graph classes where independent set is polynomial-time solvable. Let us first present some related problems, and then discuss their relation to Sparsest k-Subgraph. Actually, the following three problems can all be considered as *cardinality constrained* versions of other well-known combinatorial optimization problems, namely vertex cover and max clique, both very close to independent set.

In the maximum Quasi-Independent Set (QIS) problem [4] (also called k-edge-in in [10]), we are given a graph G and an integer C, and we ask for a set of vertices S of maximum size inducing at most C edges.

* This work has been funded by grant ANR **2010** BLAN **021902**.

[1] An edge $\{u, v\} \in E$ is said to be induced (resp. covered) by a set S if $u \in S$ and (resp. or) $v \in S$.

C. Kaklamanis and K. Pruhs (Eds.): WAOA 2013, LNCS 8447, pp. 73–84, 2014.

In the MINIMUM PARTIAL VERTEX COVER (PVC) problem [11], we are given a graph G and an integer C, and we ask for a set of vertices S of minimum size which covers[1] at least C edges.

Finally, we can mention the corresponding maximization problem of SPARSEST k-SUBGRAPH, namely DENSEST k-SUBGRAPH, which consists in finding a subset S of exactly k vertices inducing the maximum number of edges.

The decision versions of QIS, PVC, and SPARSEST k-SUBGRAPH are polynomially equivalent. Indeed, QIS could be considered as a dual version of SPARSEST k-SUBGRAPH where the budget (the number of edges in the solution of SPARSEST k-SUBGRAPH) is fixed. PVC and SPARSEST k-SUBGRAPH are also polynomially equivalent as for any S, the number of edges induced by S plus the number of edges covered by $V \backslash S$ equals $|E|$. Then, exact results for DENSEST k-SUBGRAPH on a graph class implies the same result for SPARSEST k-SUBGRAPH on the corresponding complementary class, and conversely. Unlike exact results, approximation algorithms do not transfer directly between any of these problems.

Considering these remarks and previous studies on these problems, Figure 1 presents known results and open problems about SPARSEST k-SUBGRAPH (SkS), DENSEST k-SUBGRAPH (DkS) and PVC in restricted graph classes. In each cell, the first line generally describes the general complexity ($\mathcal{NP}$-hard *versus* Polynomial), whereas other lines present some results concerning approximation or parameterized complexity. We recall that PROPER INTERVAL GRAPHS $\subset$ INTERVAL GRAPHS $\subset$ CHORDAL GRAPHS $\subset$ PERFECT GRAPHS, as well as SPLIT GRAPHS $\subset$ CHORDAL GRAPHS and BIPARTITE GRAPHS, COGRAPHS $\subset$ PERFECT GRAPHS.

Graphs classes	DkS	SkS	PVC
general	$\mathcal{NP}$-h (*c.f.* MAX CLIQUE) $n^{\frac{1}{4}+\epsilon}$-approx. [3]	$\mathcal{NP}$-h, not approx. (*c.f.* INDEP. SET)	$\mathcal{NP}$-h, $W[1]$-h [11] 2-approx.[6] exact $O^*(1, 4^C)$ [13]
chordal	$\mathcal{NP}$-h [8] 3-approx [14]	$\mathcal{NP}$-h [this paper] 2-approx [this paper]	$\mathcal{NP}$-h (*c.f.* SkS)
interval	OPEN, PTAS [15]	OPEN	OPEN
proper interval	OPEN	OPEN $PTAS$ [this paper]	OPEN
bipartite	$\mathcal{NP}$-h [8]	$\mathcal{NP}$-h (*c.f.* PVC)	$\mathcal{NP}$-h [12]
line	OPEN	$\mathcal{P}$ (*c.f.* PVC)	$\mathcal{P}$ [1]
planar	OPEN	$\mathcal{NP}$-h (*c.f.* INDEP. SET)	$\mathcal{NP}$-h (*c.f.* SkS)
cographs, split, bounded treewidth	$\mathcal{P}$ [8]	$\mathcal{P}$ [5]	$\mathcal{P}$ (*c.f.* SkS)
max. degree 2	$\mathcal{P}$ [8]	$\mathcal{P}$ [5]	$\mathcal{P}$ (*c.f.* SkS)
max. degree 3	$\mathcal{NP}$-h [8]	$\mathcal{NP}$-h (*c.f.* INDEP. SET)	$\mathcal{NP}$-h (*c.f.* SkS)

Fig. 1. Main results for DkS, SkS and PVC in some restricted graph classes

1.2 Contributions and Organization of the Paper

According to Figure 1, Densest k-Subgraph was already known to be $\mathcal{NP}$-hard on chordal graphs. However, as the complement of a chordal graph (and in particular the graph used in the reduction of [8]) is a perfect graph and not necessarily a chordal graph, this result only provides the $\mathcal{NP}$-hardness of Sparsest k-Subgraph on perfect graphs.

Thus, our motivation is to study Sparsest k-Subgraph on a classical subclass of perfect graphs. The main results of the paper are the $\mathcal{NP}$-hardness of Sparsest k-Subgraph in chordal graphs (Section 3), and a tight 2-approximation greedy algorithm (Section 2). Finally, we show in Section 4 how the arguments of [15] (which provides a PTAS for DkS in interval graphs) can be adapted to SkS in proper interval graphs. Notice that our $\mathcal{NP}$-hardness result implies the $\mathcal{NP}$-hardness of PVC in chordal graphs, which supplements the recent $\mathcal{NP}$-hardness of [2,12] for PVC in bipartite graphs. Due to space constraints, the proof of Lemma 6 for the NP-hardness in chordal graphs as well as the $PTAS$ in proper interval graphs were omitted. They can be found in the long version of this paper available in [17].

1.3 Notations and Definitions

All graphs studied in this paper are simple and without loop. For the remaining, $G = (V, E)$ will denote the input graph of the problem, and we define as usual $n = |V|$, $m = |E|$.

Chordal graphs are graphs with no induced cycle of length four or more. They may also be defined equivalently in terms of simplicial elimination order [9]. A vertex $v \in V$ is called simplicial if its neighbourhood $N(v)$ is a clique. A *simplicial elimination order* of G is an ordering $v_1, ..., v_n$ of V such that for all $i \in \{1, ..., n\}$, v_i is simplicial in $G[v_i, ..., v_n]$. It is known that a graph G is chordal if and only if it admits a simplicial elimination order. In addition, such an ordering can be found in polynomial time for a chordal graph. Hence, we will suppose in the following that $V = \{v_1, ..., v_n\}$ is sorted according to a simplicial elimination order of G. Similarly, for a subset of vertices $S \subseteq V$, we will denote by $\min(S)$ (resp $\max(S)$) the first (resp. last) vertex of S in the simplicial elimination order chosen for the graph. Finally, since we have a total ordering on the vertices, we will use the notations $x < y$ and $x > y$ for two vertices $x, y \in V$.

Given two sets $S_1, S_2 \subseteq V$, we denote by $cost(S_1)$ the number of edges in the graph induced by vertices of S_1, and $cost(S_1, S_2) = |\{\{v, v'\} \in E, v \in S_1, v' \in S_2\}|$. Given a set $S \subseteq V$ and $x \in V$, we denote by $d(x, S)$ the degree of x in S.

Finally, we refer the reader to the classical literature for definitions of approximation algorithms.

2 2-Approximation in Chordal Graphs

2.1 Idea of the Algorithm

We now present a tight 2-approximation algorithm for chordal graphs. First, notice that any approximation algorithm for Sparsest k-Subgraph must output

a maximum independent set of size k if such a set exists, as in this case the optimal value is 0. Hence, a natural idea for computing a solution to Sparsest k-Subgraph is to choose first a maximum independent set S (this can be done in polynomial time in chordal graphs). If k vertices or more were picked, then the algorithm stops. Otherwise, several ideas may come up.

A first idea would be to remove this independent set from the graph, and iteratively pick another one, until we get k vertices. This approach is the same as the 3-approximation of [14] for Densest k-Subgraph in chordal graphs (computing maximum cliques instead of maximum independent sets). Unfortunately, as shown in Figure 2, this algorithm has an unbounded approximation ratio for Sparsest k-Subgraph even in interval graphs (a subclass of chordal graphs). It still provides a 2-approximation in proper interval graphs [16].

Thus, after picking the first maximum independent set, our idea is to assign weights on remaining vertices according to the size of their neighbourhood in the constructed solution. At each step, the algorithm just picks an independent set (called a *layer*) among the vertices of minimum weight, and then updates the weights of remaining vertices. The algorithm is more formally defined in the next subsection. In the next paragraph, we describe the key idea of the analysis.

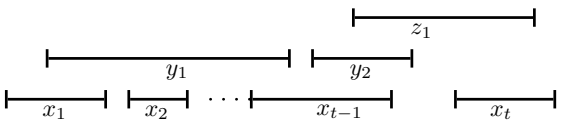

Fig. 2. In this case picking successive independent sets gives an unbounded ratio: for $k = t + 2$ the algorithm will take intervals $\{x_1, ..., x_t, y_1, y_2\}$ of cost t whereas the solution $\{x_1, ..., x_t, y_2, z_1\}$ is of cost 4

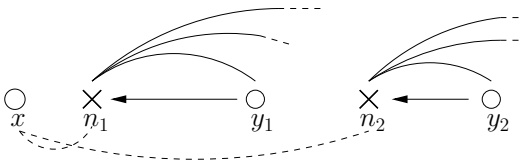

Fig. 3. Idea of the restructuration of a solution S^*. Circles denote vertices of S^*, and crosses denote vertex of L (chosen by the algorithm). When replacing y_1 by n_1 and y_2 by n_2, the degree of x can only increase by one. Indeed, x cannot be connected to n_1 and n_2, as L is an independent set.

The idea of the proof of the 2-approximation ratio is to restructure an optimal solution S^* until we get S (the output of the algorithm), while bounding the cost variation during the restructurations. Let us show what makes this restructuration work for the first layer. Let L be the independent set chosen by the

algorithm at the first step. Roughly speaking, for each $n_j \in L$ which is not in S^*, we restructure S^* by removing y_j, the "first" neighbour of n_j which is after n_j and in S^*, and adding n_j instead. As depicted in Figure 3, we see that the degree of a vertex $x \in S^*$ ($x \notin L$) will increase by at most 1. Concerning future layers, the analysis will become more complex, as we will have to take weights into account.

2.2 Algorithm and Analysis

Presentation of the Algorithm. As described previously, Algorithm 1 picks successively an independent set among the vertices of lower weights. It also updates the weights according to the picked vertices. For technical reasons, the weights are not exactly equal to their degree in the constructed solution. Indeed, when restructuring an optimal solution to match L_i we will see that the degree of almost all "surviving" vertices in the optimal solution increases by at most 1 (this is why we add a "bonus" of -1 in the updated weight Line 13), and even sometimes cannot increase (this is why there is no "bonus" Line 11). This modification will allow us to show that at the end of the algorithm, the value W returned by the algorithm is a lower bound of the optimal value (Lemma 3). We will then show that the real value of the returned solution $cost(S)$ is less than two times W (Lemma 4), and thus is a 2-approximation.

Algorithm 1. A 2-approximation for SPARSEST k-SUBGRAPH in chordal graphs

1: $S \leftarrow \emptyset$, $W \leftarrow 0$, $i \leftarrow 0$, $w_0(x) = 0\ \forall x \in V$
2: **while** $|S| \leq k$ **do**
3: $L_i \leftarrow$ a maximum independent set of the graph induced by $\{x \in V \backslash (L_0 \cup ... \cup L_{i-1}) : w_i(x) = i\}$
4: $S \leftarrow S \cup L_i$ // or the $(k - |S \cup L_i|)$ leftmost vertices of L_i if $|S \cup L_i| > k$
5: $W \leftarrow W + i|L_i|$ // we update the cost computed by the algorithm
6: **for** $x \in V$ **do**
7: **if** $x \in (L_0 \cup ... \cup L_i)$ **then**
8: $w_{i+1}(x) = w_i(x)$
9: **else**
10: **if** $d(x, L_i) = 0$ OR $(d(x, L_i) = 1$ AND $w_i(x) = i)$ **then**
11: $w_{i+1}(x) = w_i(x) + d(x, L_i)$
12: **else**
13: $w_{i+1}(x) = w_i(x) + d(x, L_i) - 1$
14: $i \leftarrow i + 1$
15: $t \leftarrow i - 1$ // L_t is the last "layer" of the algorithm
16: **return** (S, W)

Remark 1. The maximum independent set of line 3 is greedily constructed as follows: pick the first vertex of the simplicial elimination order in the independent set, delete its neighbourhood, and repeat the operation until the graph becomes empty.

Even if we sometimes add -1 when updating the weights, we can observe that for a fixed $x \in V$, its successive weights can be lower bounded as follows:

Lemma 1. *For all $i \in \{0, ..., t\}$, $\forall x \in V \setminus (L_0 \cup \cdots \cup L_i)$, $w_{i+1}(x) \geq i + 1$.*

Proof. Let i and x be as in the statement. Suppose by induction that $w_i(x) \geq i$ (notice that $w_0(x) = 0$). If $w_i(x) \geq i + 1$ then the results follows. Otherwise $w_i(x) = i$, and by construction of the algorithm (Line 11), if $d(x, L_i) \geq 1$, then $w_{i+1}(x) \geq i + 1$. Finally, if $d(x, L_i) = 0$, then x must belong to L_i which contradicts the definition of x. □

Restructuration of Solutions. Let S^* be an optimal solution for the Sparsest k-Subgraph problem in chordal graphs. We will now show that we can modify this solution in order to obtain the output of the algorithm, while bounding the cost variation.

Let us define by induction a sequence $(S_i^*)_{i=-1,0,...,t}$ with $S_{-1}^* = S^*$ and $S_t^* = S$ (the solution returned by the algorithm), such that $S_i^* \subseteq V$ and $|S_i^*| = k$ for all $i = -1...t$. We also assure that $(L_0 \cup ... \cup L_i) \subseteq S_i^*$ for all $i = 0...t$. To that end, given $i \in \{0, ..., t\}$, we show how to restructure the set S_{i-1}^* into a new set S_i^*. Let us first introduce some notations.

We partition the set L_i (defined in the algorithm) into two sets of vertices, whether they belong to S_{i-1}^* or not: $L_i = M_i \cup N_i$, with $M_i = L_i \cap S_{i-1}^*$ (and thus $N_i = L_i \setminus S_{i-1}^*$).

The restructuration consists in adding all vertices of N_i to S_{i-1}^*, and removing a carefully chosen (see Definition 1) subset $D_i \subseteq S_{i-1}^* \setminus (L_0 \cup ... \cup L_i)$ (with $|D_i| = |N_i|$). Then, we will define $S_i^* = (S_{i-1}^* \setminus D_i) \cup N_i$, $R_i = S_{i-1}^* \setminus (D_i \cup L_0 \cup ... \cup L_i)$ and $T_i = M_i \cup D_i$. Figure 4 summarizes the situation.

To bound the cost variation, we show in Lemma 2 that the degree of "surviving" vertices (*i.e.* vertices in R_i) increases by at most one. The next definition shows how to choose properly the set D_i.

Definition 1. *Let $i \in \{0, \ldots, t\}$, and let us suppose we are given a set $S_{i-1}^* \supseteq L_0 \cup ... \cup L_{i-1}$. Let $N_i = \{n_1, ..., n_{p_i}\}$ and suppose that $n_1 < ... < n_{p_i}$ defines an ordering of N_i according to the simplicial elimination order of the graph. For all $j = 1, ..., p_i$ successively, we pick a vertex $y_j \in S_{i-1}^* \setminus (L_0 \cup ... \cup L_i)$ as follows:*

$$y_j = \begin{cases} \min(Q_j) & \text{if } Q_j \neq \emptyset \\ \max(S_{i-1}^* \setminus (L_0 \cup ... \cup L_i \cup \{y_1, ..., y_{j-1}\})) & \text{if } Q_j = \emptyset \end{cases} \quad (1)$$

with $Q_j = \{y \in S_{i-1}^* \setminus (L_0 \cup ... \cup L_i \cup \{y_1, ..., y_{j-1}\})$ such that $n_j < y$, and $\{n_j, y\} \in E\}$ (see Figure 4). Finally, we define $D_i = \{y_j : 1 \leq j \leq p_i\}$.

It is easily seen that $|D_i| = |N_i|$ since all y_j are distinct. Now that D_i is defined, recall that we have $T_i = M_i \cup D_i$, and the "surviving vertices" $R_i = R_{i-1} \setminus T_i$. Let us now upper bound the degree of vertices of R_i.

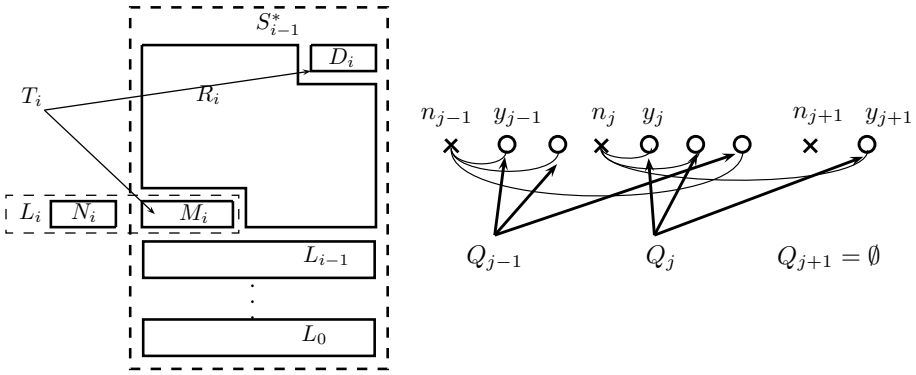

Fig. 4. On the left: description of set S^*_{i-1}. We obtain S^*_i from S^*_{i-1} by removing D_i and adding N_i. Notice that $R_{i-1} = R_i \cup T_i$, and $R_i \cap T_i = \emptyset$. On the right: example of sets Q_j, together with y_j. Circles represent vertices of the considered optimal solution, and crosses represent vertices chosen by the algorithm that are not in the optimal solution. Edges between vertices of the optimal solution have not been drawn for sake of simplicity.

Lemma 2. *Let $R_i = A_i \cup B_i$, with $A_i = \{x \in R_i : d(x, L_i) = 0$ or $(d(x, L_i) = 1$ and $w_i(x) = i)\}$ and $B_i = R_i \backslash A_i$. We have:*

- *if $x \in A_i$, $d(x, L_i) \leq d(x, T_i)$*
- *if $x \in B_i$, $d(x, L_i) \leq d(x, T_i) + 1$*

This immediately implies that for all $x \in R_i$ we have $w_{i+1}(x) \leq d(x, T_i) + w_i(x)$.

Proof. Let us show that if $x \in A_i$, then $d(x, N_i) \leq d(x, D_i)$, and if $x \in B_i$, then $d(x, N_i) \leq d(x, D_i) + 1$. Since $L_i = M_i \cup N_i$ and $T_i = M_i \cup D_i$ (these unions being disjoint), the desired inequalities follow immediatly.

- if $x \in A_i$, then either $d(x, L_i) = 0$, which obviously implies the result, or $d(x, L_i) = 1$ and $w_i(x) = i$. We thus only consider the second case. Here again if $d(x, N_i) = 0$ then the result is straightforward, so let us suppose $d(x, N_i) = 1$, *i.e.* suppose that there exists a vertex of N_i, say n_{j_0}, such that x and n_{j_0} are adjacent. Two cases are possible:
  - First case: $x < n_{j_0}$. Recall that n_{j_0} is the only neighbour of x in L_i. Hence, x is not adjacent to all vertices of L_i that are before n_{j_0} in the simplicial elimination order. In addition, recall that $w_i(x) = i$. Thus, this case cannot happen since by definition of the algorithm, x would have been chosen in L_i instead of n_{j_0}.
  - Second case: $n_{j_0} < x$. It is clear that in this case $Q_{j_0} \neq \emptyset$ (as at least $x \in Q_{j_0}$). As by definition $x \notin D_i$, we have $y_{j_0} < x$. By definition of perfect elimination order, since $\{n_{j_0}, y_{j_0}\} \in E$ and $\{n_{j_0}, x\} \in E$, we must have $\{x, y_{j_0}\} \in E$. Hence $d(x, D_i) = 1$ and the result follows.

– if $x \in B_i$, then let $N_i^- = \{y \in N_i : y < x\}$ and $N_i^+ = N_i \setminus N_i^-$. For all $n_j \in N_i^-$ such that $\{n_j, x\} \in E$, then as previously $Q_j \neq \emptyset$ and by the definition of chordal graphs we have $\{y_j, x\} \in E$ with $y_j \in D_i$. Thus, $d(x, N_i^-) \leq d(x, D_i)$. Finally we claim that $d(x, N_i^+) \leq 1$. Indeed, suppose that there exists $n_{j_1}, n_{j_2} \in N_i^+$ such that $\{x, n_{j_1}\}, \{x, n_{j_2}\} \in E$. By definition of perfect elimination order we must have $\{n_{j_1}, n_{j_2}\} \in E$ which contradicts the definition of L_i which is an independent set. This proves that $d(x, N_i) \leq d(x, D_i) + 1$ □

Let us now define the appropriate ζ function that computes the cost of an intermediate solution S_i^*. For all $i \in \{0, ..., t\}$, let

$$\zeta(S_i^*) = cost(R_i) + \sum_{x \in R_i} w_{i+1}(x) + \sum_{x \in L_0 \cup \cdots \cup L_i} w_{i+1}(x)$$

Notice that $\zeta(S_{-1}^*) = cost(S^*)$ and $\zeta(S_t^*) = \sum_{x \in S} w_t(x) = W$.

Lemma 3. *For all $i \in \{0, ..., t\}$, D_i is such that $\zeta(S_i^*) \leq \zeta(S_{i-1}^*)$.*

Proof. By definition, we have:

$$\begin{aligned}
\zeta(S_i^*) &= cost(R_i) + \textstyle\sum_{x \in R_i} w_{i+1}(x) + \sum_{x \in L_0 \cup \cdots \cup L_i} w_{i+1}(x) \\
&= cost(R_i) + \textstyle\sum_{x \in R_i} w_{i+1}(x) + i|L_i| + \sum_{x \in L_0 \cup \cdots \cup L_{i-1}} w_{i+1}(x) \\
&\leq cost(R_i) + \textstyle\sum_{x \in R_i} (w_i(x) + d(x, T_i)) + i|L_i| + \sum_{x \in L_0 \cup \cdots \cup L_{i-1}} w_i(x) \text{ by Lemma 2}
\end{aligned}$$

In addition, since $R_{i-1} = R_i \cup T_i$ and $|T_i| = |L_i|$, we have:

$$\begin{aligned}
\zeta(S_{i-1}^*) &= cost(R_{i-1}) + \textstyle\sum_{x \in R_{i-1}} w_i(x) + \sum_{x \in L_0 \cup \cdots \cup L_{i-1}} w_i(x) \\
&= cost(R_i) + cost(R_i, T_i) + \textstyle\sum_{x \in R_i} w_i(x) + \sum_{x \in T_i} w_i(x) + \sum_{x \in L_0 \cup \cdots \cup L_{i-1}} w_i(x) \\
&\geq cost(R_i) + cost(R_i, T_i) + \textstyle\sum_{x \in R_i} w_i(x) + i|L_i| + \sum_{x \in L_0 \cup \cdots \cup L_{i-1}} w_i(x)
\end{aligned}$$

which matches the upper bound for $\zeta(S_i^*)$. □

The previous lemma implies that $W = \zeta(S_t^*) \leq \zeta(S_{-1}^*) = cost(S^*)$. Thus, to prove that Algorithm 1 is a 2-approximation we only need the following lemma.

Lemma 4. $cost(S) \leq 2W$.

Proof. Roughly speaking, when creating a layer L_i and updating the cost W, the algorithm adds $i|L_i|$, *i.e.* for all $x \in L_i$ the algorithm only adds i instead of $d(x, L_0 \cup \cdots \cup L_{i-1})$. Thus, we will now prove for any $x \in L_i$, $d(x, L_0 \cup \cdots \cup L_{i-1}) \leq 2i$.

Let x in L_i. For any l, $0 \leq l \leq i$, let $x_l = w_l(x)$ be the weight of x before creating layer L_l. Thus, the successive weights of x is a sequence $(x_0, \ldots, x_i)$ where $x_0 = 0$ and $x_i = i$. Notice that after x is added in L_i its weight will not be changed.

Let us show by induction that for any l, $d(x, L_0 \cup \cdots \cup L_{l-1}) \leq x_l + l$. Let us suppose that the previous statement is true for l and prove it for $l + 1$.

Let $z = d(x, L_l)$. We have $d(x, L_0 \cup \cdots \cup L_l) = d(x, L_0 \cup \cdots \cup L_{l-1}) + z \leq x_l + l + z$. As $x_{l+1} \geq x_l + z - 1$, we get the desired inequality.

Thus, for any $x \in L_i$ we get $d(x, L_0 \cup \cdots \cup L_{i-1}) \leq x_i + i = 2i$, and thus $cost(S) = \sum_{i=1}^{t} \sum_{x \in L_i} d(x, L_0 \cup \cdots \cup L_{i-1}) \leq 2 \sum_{i=1}^{t} i|L_i| = 2W$. □

Theorem 1. *There is a tight polynomial* 2*-approximation algorithm for* SkS *in chordal graphs.*

For the tightness result, consider the instance with $n = 5$, $k = 4$, and edges $\{x_1, x_2\}$, $\{x_2, x_3\}$ and $\{x_4, x_5\}$ (notice that $(x_1, x_2, x_3, x_4, x_5)$ is a simplicial elimination order). The algorithm will first pick x_1, x_3 and x_4. Then, we have $w_1(x_2) = w_1(x_5) = 1$ and the algorithm takes x_2 instead of x_5.

3 $\mathcal{NP}$-Hardness in Chordal Graphs

Main Arguments. The following $\mathcal{NP}$-hardness proof is a reduction from the k-CLIQUE problem in general graphs. Roughly speaking, given an input instance $G = (V, E)$ together with $k \in \mathbb{N}$, we construct the split graph of adjacencies of G, *i.e.* we build a clique on a set A representing the vertices of G, and an independent set F representing the edges of G, connecting A and F with respect to the adjacencies of the graph. Then, we replace each vertex of the independent set (corresponding to an edge $e \in E$) by a gadget F_e represented in Figure 5. Any solution will have to take the same number of vertices among each gadget. The key idea is that there is two ways to take these vertices in a gadget F_e. The first way (choosing X_e and Z_e) encodes that the edge e belongs to the k-clique. It is cheaper than the second way, but is adjacent to the clique A. The second way (choosing X_e and Y_e) encodes that edge e does not belong to the k-clique. It induces more edges, but is not adjacent to the clique A. Thus, as depicted in Figure 5, a k-clique is encoded by *not* picking the corresponding vertices in A, obtaining $\binom{k}{2}$ gadgets of the first type, and $m - \binom{k}{2}$ of the second type. In this way, there is no edge in the solution between any gadget and the clique A. For technical reasons, each vertex of A is duplicated n times.

Gadget. Let us define the gadget F mentioned above. F is composed of three sets X, Y and Z of T vertices each (we will set the value of T later). We define $X = \{x_1, ..., x_T\}, Y = \{y_1, ..., y_T\}$ and $Z = \{z_1, ..., z_T\}$. The set X induces an independent set, while Z induces a clique, and there is a clique of size $(T-1)$ on vertices $\{y_2, ..., y_T\}$. For all $i \in \{1, ..., T\}$, x_i is adjacent to y_i, and y_i is adjacent to all vertices of Z. Such a construction is depicted at the left of Figure 5.

In the following we will force the solution to take $2T$ vertices among each gadget. It is easy to see that the sparsest $2T$-subgraph of F is composed of the sets X and Z, which induces $\binom{T}{2}$ edges. In contrast, notice that choosing X and Y induces $(\binom{T}{2} + 1)$ edges.

Theorem 2. SPARSEST k-SUBGRAPH *remains* $\mathcal{NP}$*-hard in chordal graphs.*

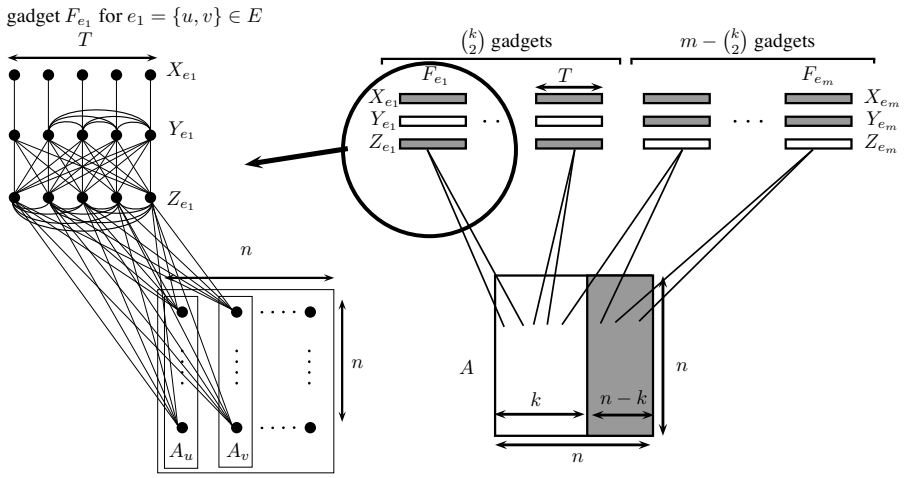

Fig. 5. Schema of the reduction, with an example of a gadget F_{e_1} on the left and its relations to A. Grey rectangles represent vertices of the solution.

Proof. We reduce from the classical k-CLIQUE problem in general graphs. Let $G = (V, E)$ and $k \in \mathbb{N}$. We note $|V| = n$, $V = \{v_1, ..., v_n\}$, $|E| = m$ and $T = n(n-k)$. In the following we will define $G' = (V', E')$ together with $k', C' \in \mathbb{N}$ such that G' is a chordal graphs which can be constructed in polynomial time, and such that G contains a clique of size k if and only if one can find k' vertices in G' which induce C' edges or less.

The Construction. V' is composed of two parts A and F:

- We first define a clique of size n^2 over $A = \{a_i^j : i, j \in \{1, ..., n\}\}$. For each $u \in V$, the "column" $A_u = \{a_u^j : j \in \{1, ..., n\}\}$ represents the vertex u in G.
- For all $e \in E$, we construct a gadget F_e composed of X_e, Y_e and Z_e as defined previously. Let $X_e = \{x_1^e, ..., x_T^e\}$, $Y_e = \{y_1^e, ..., y_T^e\}$ and $Z_e = \{z_1^e, ..., z_T^e\}$. Moreover, for all $e = \{v_p, v_q\} \in E$, all vertices of Z_e are connected to A_p and A_q.
- We define $k' = m2T + T$ and $C' = m\binom{T}{2} + \binom{T}{2} + (m - \binom{k}{2})$.

It is clear that the construction can be carried out in polynomial time. Let us briefly sketch that G' is a chordal graph: for each gadget, X_e, Y_e, Z_e is a simplicial elimination order. Then, the remaining vertices form a clique.

Now we prove that G contains a clique of size k if and only if G' contains k' vertices inducing at most C' edges.

Lemma 5. *G contains a k-clique $\Rightarrow$ G' contains k' vertices inducing at most C' edges.*

Proof. Let us suppose that $K \subseteq V$ is a clique of size k in G. W.l.o.g. we suppose $K = \{v_1, ..., v_k\}$. Moreover, we note $E_0 = \{\{v_p, v_q\} \in E$ such that $v_p, v_q \in K\}$ and $E_1 = \{\{v_p, v_q\} \in E$ such that $v_p \notin K$ or $v_q \notin K\}$. We construct $K' \subseteq V'$ as follows:

- For all $i \in \{(k+1), ..., n\}$ and all $j = \{1, ..., n\}$, we add a_i^j to K'.
- For all $e \in E$, we add all vertices of X_e to K'.
- For all $e \in E_0$, we add all vertices of Z_e to K'.
- For all $e \in E_1$, we add all vertices of Y_e to K'.

One can verify that K' is a set of $k' = 2mT + T$ vertices inducing exactly $C' = \binom{T}{2} + m\binom{T}{2} + (m - \binom{k}{2})$ edges. Indeed, we picked $T = n(n-k)$ vertices from A which is a clique and thus induce $\binom{T}{2}$ edges. Then, for all $e \in E$, we picked $2T$ vertices, which induce $\binom{T}{2}$ edges if $e \in E_0$, and $(\binom{T}{2} + 1)$ edges if $e \in E_1$. Since $|E_0| = \binom{k}{2}$ (and thus $|E_1| = m - \binom{k}{2}$), we have the desired number of edges.

Lemma 6. *G' contains k' vertices inducing at most C' edges $\Rightarrow$ G contains a k-clique.* □

4 Approximation in Proper Interval Graphs

Let us now discuss the status of Sparsest k-Subgraph and Densest k-Subgraph on interval graphs. First, notice that the complexity status ($\mathcal{NP}$-hardness *versus* $\mathcal{P}$) of Sparsest k-Subgraph remains unknown in interval and proper interval graphs. We also recall that this question is a longstanding open problem for DkS, as well as its complexity in planar graphs. Indeed, the former paper [8] proves the $\mathcal{NP}$-hardness of DkS in comparability, chordal graphs, and states the open question of its complexity in planar and (proper) interval graphs. Since then, and despite a lot of effort, no major improvement has been done so far.

As interval graphs are exactly the intersection of chordal graphs and co-comparability graphs, finding out the complexity status of Sparsest k-Subgraph in interval graphs would determine the complexity of Densest k-Subgraph in a subclass of comparability graphs, improving the results of [8]. Finally, as in [15] where the author design a *PTAS* for Densest k-Subgraph on interval graph (despite the unknown complexity status), we are able to show the following theorem.

Theorem 3. *There is a PTAS for* SkS *in proper intervals running in $n^{\mathcal{O}(\frac{1}{\epsilon})}$.*

This result uses the same kind of arguments as in [15]: restructuring an optimal solution in each "block" of consecutive intervals, and using dynamic programing on these restructured blocks.

References

1. Apollonio, N., Sebő, A.: Minconvex factors of prescribed size in graphs. SIAM Journal of Discrete Mathematics 23(3), 1297–1310 (2009)
2. Apollonio, N., Simeone, B.: The maximum vertex coverage problem on bipartite graphs. Preprint (2013)
3. Bhaskara, A., Charikar, M., Chlamtac, E., Feige, U., Vijayaraghavan, A.: Detecting high log-densities: An $\mathcal{O}(n^{1/4})$ approximation for densest k-subgraph. In: Proceedings of the 42nd ACM symposium on Theory of Computing, pp. 201–210. ACM (2010)
4. Bourgeois, N., Giannakos, A., Lucarelli, G., Milis, I., Paschos, V., Pottié, O.: The max quasi-independent set problem. Journal of Combinatorial Optimization 23(1), 94–117 (2012)
5. Broersma, H., Golovach, P.A., Patel, V.: Tight complexity bounds for FPT subgraph problems parameterized by clique-width. In: Marx, D., Rossmanith, P. (eds.) IPEC 2011. LNCS, vol. 7112, pp. 207–218. Springer, Heidelberg (2012)
6. Bshouty, N., Burroughs, L.: Massaging a linear programming solution to give a 2-approximation for a generalization of the vertex cover problem. In: Meinel, C., Morvan, M., Krob, D. (eds.) STACS 1998. LNCS, vol. 1373, pp. 298–308. Springer, Heidelberg (1998)
7. Cai, L.: Parameterized complexity of cardinality constrained optimization problems. Computer Journal 51(1), 102–121 (2008)
8. Corneil, D.G., Perl, Y.: Clustering and domination in perfect graphs. Discrete Applied Mathematics 9(1), 27–39 (1984)
9. Fulkerson, D., Gross, O.: Incidence matrices and interval graphs. Pacific J. Math. 15, 835–855 (1965)
10. Goldschmidt, O., Hochbaum, D.S.: k-edge subgraph problems. Discrete Applied Mathematics 74(2), 159–169 (1997)
11. Guo, J., Niedermeier, R., Wernicke, S.: Parameterized complexity of vertex cover variants. Theory of Computing Systems 41(3), 501–520 (2007)
12. Joret, G., Vetta, A.: Reducing the rank of a matroid. CoRR, abs/1211.4853 (2012)
13. Kneis, J., Langer, A., Rossmanith, P.: Improved upper bounds for partial vertex cover. In: Broersma, H., Erlebach, T., Friedetzky, T., Paulusma, D. (eds.) WG 2008. LNCS, vol. 5344, pp. 240–251. Springer, Heidelberg (2008)
14. Liazi, M., Milis, I., Zissimopoulos, V.: A constant approximation algorithm for the densest k-subgraph problem on chordal graphs. Information Processing Letters 108(1), 29–32 (2008)
15. Nonner, T.: PTAS for densest k-subgraph in interval graphs. In: Dehne, F., Iacono, J., Sack, J.-R. (eds.) WADS 2011. LNCS, vol. 6844, pp. 631–641. Springer, Heidelberg (2011)
16. Watrigant, R., Bougeret, M., Giroudeau, R.: The k-sparsest subgraph problem. Technical Report RR-12019, LIRMM (2012)
17. Watrigant, R., Bougeret, M., Giroudeau, R.: Approximating the sparsest k-subgraph in chordal graphs. Technical Report hal-00868188, LIRMM (2013)