Approximating the Sparsest k-Subgraph in Chordal Graphs*

R. Watrigant, M. Bougeret, and R. 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 tight 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 [11]), 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.

In the minimum Partial Vertex Cover (PVC) problem [12], 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

* 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) $v \in S$.

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 $\subseteq$ interval graphs $\subseteq$ chordal graphs $\subseteq$ perfect graphs, as well as split graphs $\subseteq$ chordal and bipartite, cographs $\subseteq$ 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 [12] 2-approx.[6] exact $O^*(1, 4^C)$ [14]
chordal	$\mathcal{NP}$-h [9] 3-approx [15]	$\mathcal{NP}$-h [this paper] 2-approx [this paper]	$\mathcal{NP}$-h (*c.f.* SkS)
interval	OPEN PTAS [16]	OPEN, $FPT(C)$ [17]	OPEN $FPT(n\text{-}k)$ (*c.f.* SkS)
proper interval	OPEN PTAS [16]	OPEN, *PTAS* [this paper]	OPEN
bipartite	$\mathcal{NP}$-h [9]	$\mathcal{NP}$-h (*c.f.* PVC)	$\mathcal{NP}$-h [13]
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	$\mathcal{P}$ [9]	$\mathcal{P}$ [5]	$\mathcal{P}$ (*c.f.* SkS)
max. degree 2	$\mathcal{P}$ [9]	$\mathcal{P}$ [5]	$\mathcal{P}$ (*c.f.* SkS)
max. degree 3	$\mathcal{NP}$-h [9]	OPEN	OPEN

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 [9]) 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 [16] (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, 13] for PVC in bipartite graphs. Due to space constraints, some details of the $\mathcal{NP}$-hardness proof and the *PTAS* in proper interval graphs were omitted and placed in the appendix.

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 usually $n = |V|$, $m = |E|$.
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 [10]. 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 [15] for Densest k-Subgraph in chordal graphs (computing maximum cliques instead of maximum independent sets). Unfortunately, as shown in Figure 2 on the left, 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 [17].
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.

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 2 (on the right), we see that the degree of a vertex

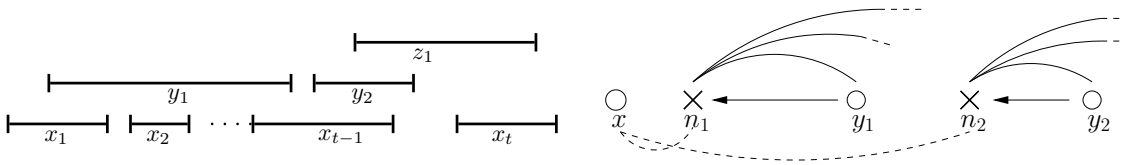

Fig. 2: On the left: example where picking successive independent sets gives an unbounded ratio. For $k = t + 2$ the algorithm would take intervals $\{x_1, ..., x_t, y_1, y_2\}$ of cost t whereas the solution $\{x_1, ..., x_t, y_2, z_1\}$ has cost 4. On the right: idea of the restructuration of a solution 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.

$x \in S^*$ ($x \notin L$) 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 increases (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.

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 are non decreasing:

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.

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

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

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 $M_i = L_i \cap S_{i-1}^*$.
The restructuration consists in adding all vertices of N_i to S_{i-1}^*, and removing a carefully chosen (see Definition 1) subset $D_i \subseteq S_{i-1}^* \backslash (L_0 \cup ... \cup L_i)$ (with $|D_i| = |N_i|$). Then, we will define $S_i^* = (S_{i-1}^* \backslash D_i) \cup N_i$, $R_i = S_{i-1}^* \backslash (D_i \cup L_0 \cup ... \cup L_i)$ and $T_i = M_i \cup D_i$. Figure 3 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. *For $i \in \{0, \ldots, t\}$, let D_i be defined as follows:*

Let us suppose we are given a set $S_{i-1}^* \supseteq L_0 \cup ... \cup L_{i-1}$, and show how to choose properly the set $D_i \subseteq S_{i-1}^* \setminus (L_0 \cup ... \cup L_i)$.
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 $y_j \in S_{i-1}^* \backslash (L_0 \cup ... \cup L_i)$ as follows:

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

with $Q_j = \{y \in S_{i-1}^* \backslash (L_0 \cup ... \cup L_i)$ such that $n_j < y$, and $\{n_j, y\} \in E\}$ (see Figure 3). Finally, we define $D_i = \{y_j : 1 \leq j \leq p_i\}$.

Now that D_i is defined, recall that we have $T_i = M_i \cap D_i$, and the "surviving vertices" $R_i = R_{i-1} \setminus T_i$. Let us now upper bound the degree of vertices of R_i.

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:*

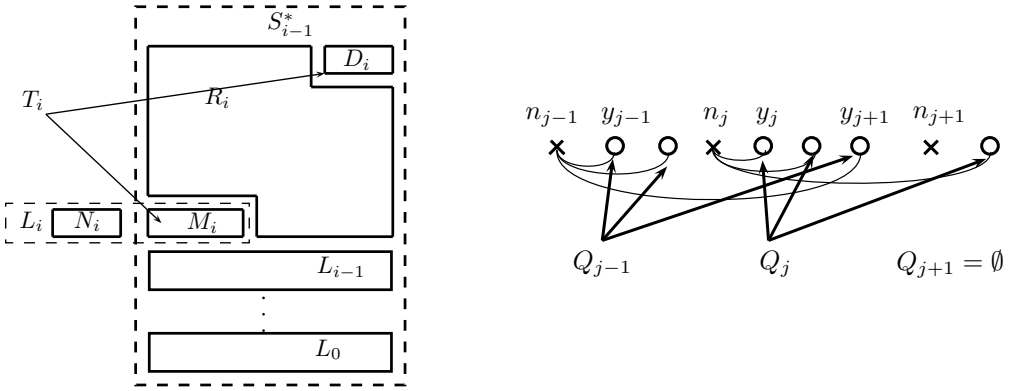

Fig. 3: On the left: description of set S^*_{i-1}. We obtain S^*_i from S^*_{i-1} by removing D_i and adding N_i. Notice that $R_{i-1} = R_i \cup T_i$, and $R_i \cap T_i = \emptyset$. On the right: example of sets Q_j, together with y_j. Circles represent vertices of the considered optimal solution, 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.

- *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 immediatly 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 chordal graphs, 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 $N_i^- = N_i \cap \{y \in N_i : y < x\}$ 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 chordal graphs 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 $\zeta(S_{t+1}^*) = \sum_{x \in S} w_t(x) = W$.

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 4. 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 4, 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 4.
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.*

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. See Appendix A for a more rigorous proof.

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

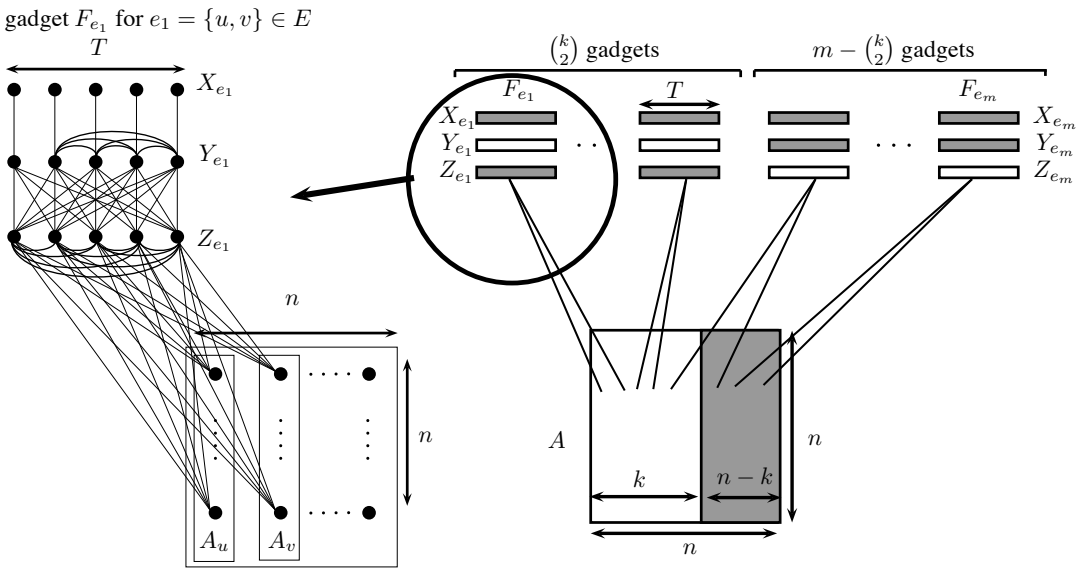

Fig. 4: Schema of the reduction, with an example of a gadget F_{e_1} on the left and its relations to A. Grey rectangles represent vertices of the solution.

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

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.

G contains a k-clique $\Leftarrow$ G' contains k' vertices inducing at most C' edges.

Suppose now that K' is a set of k' vertices of G' which induces at most C' edges. We re-define the sets E_0 and E_1 as follows: $E_0 = \{\{v_p, v_q\} \in E$ such that for all $j \in \{1, ..., n\}$ we have $a_p^j \notin K'$ and $a_q^j \notin K'\}$, and $E_1 = E \backslash E_0$. Roughly speaking, E_0 denotes the set of gadgets *not* adjacent to vertices of the solution among A, and E_1 the set of gadgets adjacent to at least one vertex of the solution among A.

For all $R \subseteq V'$, let $tr(R) = K' \cap R$ be the trace of K' on R, and for all $v \in V'$, let $\mu(v) = |tr(N(v))|$ be the number of neighbours of v belonging to K'.

Let $u \in K'$ and $v \in V' \backslash K'$. We say that $(K' \backslash \{u\}) \cup \{v\}$ is a *safe replacement* if we have $\mu(v) \leq \mu(u)$ if $\{u, v\} \notin E'$ and $\mu(v) - 1 \leq \mu(u)$ if $\{u, v\} \in E'$. For sake of readability, we will keep and update the definitions of E_0 and E_1 when replacing vertices of A (*e.g.* if we remove a vertex $u \in A$ from K' such that there exists $e \in E_1$ such that vertices of Z_e were only adjacent to u among all vertices of $tr(A)$, then e now belongs to E_0).

The proof consists in replacing some vertices of K' by other vertices not in K' without increasing the number of induced edges, in order to obtain a solution that has the same structure as previously. We call such a replacement a *safe modification* or a *safe replacement*.

The core of the proof is based on the three following lemmas.

Lemma 5. *Without loss of generality (and optimality of K'), we can suppose that for all $e \in E$ we have $X_e \subseteq K'$.*

Proof. Let $S = \bigcup_{e \in E} X_e$. Since we have $k' > |S|$, there always exists $u \in K' \backslash S$. Suppose that there exists $e \in E$ and $i \in \{1, ..., T\}$ such that $x_i^e \notin K'$. If $y_i^e \notin K'$, then we have $\mu(x_i^e) = 0$ and we can thus safely replace any other vertex of $K' \backslash S$ by x_i^e. Now, if $y_1^e \in K'$, then $\mu(x_i^e) = 1$. Since $\{x_i^e, y_i^e\} \in E'$, $(K' \backslash \{y_1^e\}) \cup \{x_i^e\}$ is a safe replacement. □

Lemma 6. *(See App. A.2) K' can be safely modified such that one of the two following holds:*
Case A1: for all $e \in E_0$ we have $tr(Z_e) = Z_e$.
Case A2: for all $e \in E_0$ we have $tr(Y_e) = \emptyset$.

Lemma 7. *(See App. A.2) K' can be safely modified such that one of the two following holds:*
Case B1: for all $e \in E_1$ we have $tr(Y_e) = Y_e$.
Case B2: for all $e \in E_1$ we have $tr(Z_e) = \emptyset$.

Let us now define for each case and each $e \in E$ the set of vertices $D_e \subseteq Y_e \cup Z_e$ that have to be replaced (see Figure 5 in Appendix A.3):

- case A1: for all $e \in E_0$, $D_e = Y_e \cap K'$
- case A2: for all $e \in E_0$, $D_e = Z_e \setminus K'$
- case B1: for all $e \in E_1$, $D_e = Z_e \cap K'$
- case B2: for all $e \in E_1$, $D_e = Y_e \setminus K'$

Notice that if $D_e = \emptyset$ for all $e \in E_0$ (resp. $e \in E_1$), then cases A1 and A2 (resp. B1 and B2) collapse. If such a case happen for all $e \in E$, we can immediately conclude, as shown by the following lemma:

Lemma 8. *If $D_e = \emptyset$ for all $e \in E$, then G contains a clique of size k.*

Proof. By construction, we have $|tr(A)| = T$ and $|tr(F_e)| = 2T$ for all $e \in E$. Thus, $cost(tr(A)) = \binom{T}{2}$ and $cost(tr(F_e)) = \binom{T}{2} + 1$ if $Y_e \subseteq K'$, and $cost(tr(F_e)) = \binom{T}{2}$ if $Z_e \subseteq K'$. By construction, $Y_e \subseteq K'$ if and only if $e \in E_1$. Thus, since $cost(K') \leq \binom{T}{2} + m\binom{T}{2} + m - \binom{k}{2}$, we must have $|E_1| \leq m - \binom{k}{2}$ which is equivalent to $|E_0| \geq \binom{k}{2}$. Hence, there exists at most $\lfloor \frac{|A|-T}{n} \rfloor = k$ vertices in G inducing at least $\binom{k}{2}$ edges, *i.e.* G contains a clique of size k. □

We now have to analyse the four cases of Lemma 6 and 7. We only detail here the first one (case A1 and B1), and refer the reader to Appendix A.3 for the other ones.

Let us consider case A1 and B1. To summarize the situation, the solution K' can be partitioned in $K'_A = K' \cap A$, and $K'_F = K' \setminus K'_A$, the vertices selected in the gadgets. Let $\Delta_0 = \sum_{e \in E_0} |D_e|$ be

the number of extra vertices allocated in all the gadgets F_e, $e \in E_0$, and $\Delta_1 = \sum_{e \in E_1} |D_e|$ be the number of extra vertices allocated in all the gadgets F_e, $e \in E_1$. Let $\Delta = \Delta_0 + \Delta_1$. Notice that we have $|K'_A| = T - \Delta$, as a "regular" solution that does not select any extra vertex in a gadget has to pick T vertices in A. Moreover,

- vertices of K' selected in gadgets of E_0 are not adjacent to K'_A (by definition of E_0)
- each gadget of E_0 induces at least $\binom{T}{2}$ edges (as we are in case A1)
- each gadget of E_1 induces at least $\binom{T}{2} + 1$ edges (as we are in case B1)
- each of the Δ_0 vertices is adjacent to at least T vertices in K' (such a vertex is in a set Y_e, and thus is connected to the T vertcies of Z_e)
- each of the Δ_1 vertices is adjacent to at least $T+1$ vertices in K' (such a vertex is in a set Z_e, and thus is connected to at least 1 vertex of K'_A and to the T vertices of Y_e)

Let us now lower bound the total cost of K'. We have:

$$\begin{aligned} cost(K') &\geq |E_0|\binom{T}{2} + |E_1|(\binom{T}{2} + 1) + \Delta_0 T + \Delta_1(T+1) + \binom{T-\Delta}{2} \\ &= m\binom{T}{2} + |E_1| + \Delta T + \Delta_1 + \binom{T}{2} - \binom{\Delta}{2} - \Delta(T-\Delta) \\ &\geq m\binom{T}{2} + (m - |E_0|) + \binom{T}{2} + \frac{\Delta^2}{2} \end{aligned}$$

Notice that in a bad structured solution, a large Δ allows to select only a few vertices in A ($T - \Delta$ instead of T), and thus to have many gadgets (more than $\binom{k}{2}$) in E_0. Let us now consider the contrapositive, *i.e.* we consider that G does not contain a k-clique, and show that K' induces more than C' edges.

Let q and r such that $\Delta = qn + r$, $r < n$. Let us upper bound $|E_0|$. As there are $T - \Delta$ vertices in A, the number of empty "columns" (column u is empty iff none of the a^t_u is selected) is at most $n - \frac{T-\Delta}{n} \leq k + q$.

As G does not contain a k-clique, the $k+q$ vertices corresponding to these $k+q$ columns cannot induce a clique of size $k+q$, and thus $|E_0| < \binom{k+q}{2}$. Thus, we get:

$$\begin{aligned} cost(K') &> m\binom{T}{2} + (m - \binom{k+q}{2}) + \binom{T}{2} + \frac{\Delta^2}{2} \\ &= C' - (\binom{q}{2} + kq) + \frac{\Delta^2}{2} \end{aligned}$$

Thus, as $\frac{\Delta^2}{2} > \binom{q}{2} + kq$, we get the desired inequality. We refer now the reader to Appendix A.3 for the three other cases. □

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 [9] 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 [9]. Finally, as in [16] where the author design a $PTAS$ for Densest k-Subgraph on interval graph (despite the unknown complexity status), we show in Appendix B 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 [16]: restructuring an optimal solution in each "block" of consecutive intervals, and using dynamic programing on these restructured blocks.

References

1. N. Apollonio and A. Sebő. Minconvex factors of prescribed size in graphs. SIAM Journal of Discrete Mathematics, 23(3):1297–1310, 2009.
2. N. Apollonio and B. Simeone. The maximum vertex coverage problem on bipartite graphs. preprint, 2013.
3. A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, and A. Vijayaraghavan. 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, pages 201–210. ACM, 2010.
4. N. Bourgeois, A. Giannakos, G. Lucarelli, I. Milis, V. Th. Paschos, and O. Pottié. The max quasi-independent set problem. Journal of Combinatorial Optimization, 23(1):94–117, 2012.
5. H. Broersma, P. A. Golovach, and V. Patel. Tight complexity bounds for FPT subgraph problems parameterized by clique-width. In Proceedings of the 6th international conference on Parameterized and Exact Computation, IPEC'11, pages 207–218, Berlin, Heidelberg, 2012. Springer-Verlag.
6. N. Bshouty and L. Burroughs. Massaging a linear programming solution to give a 2-approximation for a generalization of the vertex cover problem. In Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science, pages 298–308. Springer, 1998.
7. Leizhen Cai. Parameterized complexity of cardinality constrained optimization problems. Computer Journal, 51(1):102–121, 2008.
8. D. Chen, R. Fleischer, and J. Li. Densest k-subgraph approximation on intersection graphs. In Proceedings of the 8th international conference on Approximation and online algorithms, pages 83–93. Springer, 2011.
9. D.G. Corneil and Y. Perl. Clustering and domination in perfect graphs. Discrete Applied Mathematics, 9(1):27 – 39, 1984.
10. D. Fulkerson and O. Gross. Incidence matrices and interval graphs. Pacific J. Math., 15:835–855, 1965.
11. O. Goldschmidt and D. S. Hochbaum. k-edge subgraph problems. Discrete Applied Mathematics, 74(2):159–169, 1997.
12. J. Guo, R. Niedermeier, and S. Wernicke. Parameterized complexity of vertex cover variants. Theory of Computing Systems, 41(3):501–520, 2007.
13. G. Joret and A. Vetta. Reducing the rank of a matroid. CoRR, abs/1211.4853, 2012.
14. J. Kneis, A. Langer, and P. Rossmanith. In Proceedings of the 34th Workshop of Graph Theoretic Concepts in Computer Science, pages 240–251. Springer, 2008.
15. M. Liazi, I. Milis, and V. Zissimopoulos. A constant approximation algorithm for the densest k-subgraph problem on chordal graphs. Information Processing Letters, 108(1):29–32, 2008.
16. T. Nonner. PTAS for densest k-subgraph in interval graphs. In Proceedings of the 12th international conference on Algorithms and Data Structures, pages 631–641. Springer, 2011.
17. R. Watrigant, M. Bougeret, and R. Giroudeau. The k-sparsest subgraph problem. Technical Report RR-12019, LIRMM, 2012.

A Missing Proofs of the $\mathcal{NP}$-hardness

A.1 Proof of claim "G' is a chordal graph"

We have the following simplicial elimination scheme:

- For all $e \in E$, we can remove X^e since for all $j \in \{1, ..., T\}$, x^e_j is only connected to y^e_j.
- For all $e \in E$, we can remove Y^e. Indeed the remaining neighbourhood of y^e_1 is Z^e which is a clique. And the remaining neighbourhood of y^e_j with $j \geq 2$ is a subset $Y^e \cup Z^e \setminus \{y^e_1\}$ which induces a clique.
- For all $e \in E$, we can remove Z^e since the remaining neighbourhood of z^e_j is a subset of Z^e and vertices of A which induce a clique.
- The remaining vertices induce a clique on A and can thus be eliminated.

□

A.2 Proof of Lemma 6 and 7 (safe replacements)

Proof (Proof of Lemma 6). Let us first restructure each gadget of E_0 separately. For all $e \in E_0$ such that $tr(Y_e) \neq \emptyset$ and $tr(Z_e) \neq Z_e$, let $j_0 = \max\{j \in \{1, ..., T\} : y^e_j \in tr(Y_e)\}$ and let j_1 be such that $z^e_{j_1} \notin tr(Z_e)$. Recall that Lemma 5 ensures that $x^e_{j_0}$ is in K'. If $j_0 \neq 1$, then $\mu(y^e_{j_0}) = y + z + 1$, where $y = |N(y^e_{j_0}) \cap tr(Y_e)|$ and $z = |N(y^e_{j_0}) \cap tr(Z_e)|$. On the other side, we have $\mu(z^e_{j_1}) \leq y + z + 1$ (more precisely, $\mu(z^e_{j_1}) = y + z + 1$ if $y^e_1 \in K'$, and $\mu(z^e_{j_1}) = y + z$ if $y^e_1 \notin K'$). Roughly speaking, this switch ensures that we necessarily "loose" the edge due to the vertex of X^e and we gain at most one edge due to y^e_1. Hence $\mu(z^e_{j_1}) \leq \mu(y^e_{j_0})$ and $(K' \setminus \{y^e_{j_0}\}) \cup \{z^e_{j_1}\}$ is a safe replacement. If $j_0 = 1$, then it means that $tr(Y_e) = \{y^e_1\}$. Suppose that there exists j_1 such that $z^e_{j_1} \notin tr(Z_e)$. We have $\mu(y^e_1) = z + 1$ where $z = |N(y^e_1) \cap tr(Z_e)|$, and $\mu(z^e_{j_1}) = z + 1$. Here again $(K' \setminus \{y^e_1\}) \cup \{z^e_{j_1}\}$ is a safe replacement. After all these replacements, given any $e \in E_0$, $tr(Y_e) \neq \emptyset$ implies that $tr(Z_e) = Z_e$.
Then, we proceed to replacements between gadgets F_e, $e \in E_0$. If one can find $a, b \in E_0$ such that $tr(Y_a) \neq \emptyset$ and $tr(Z_b) \neq Z_b$, then let j_0 be such that $y^a_{j_0} \in tr(Y_a)$ and let j_1 be such that $z^b_{j_1} \notin tr(Z_b)$. We have $\mu(y^a_{j_0}) \geq T + 1$ and $\mu(z^b_{j_1}) \leq T - 1$. Thus, $(K' \setminus \{y^a_{j_0}\}) \cup \{z^b_{j_1}\}$ is a safe replacement.
Theses replacements end either when all the Y_e are empty for all $e \in E_0$ or when all the Z_e are full for all $e \in E_0$, which achieves the proof of Lemma 6. □

Proof (Proof of Lemma 7). The proof is roughly based on the fact that replacing a vertex of Z_e by a vertex of Y_e permits to "loose" at least one edge with vertices A and "gain" one edge with a vertex of X_e. Let us formally prove Lemma 7. Similarly to the proof of Lemma 6, we first restructure each gadget of E_1 separately: for all $e \in E_1$ such that $tr(Z_e) \neq \emptyset$ and $tr(Y_e) \neq Y_e$, let $j_0 = \max\{j \in \{1, ..., T\} : y^e_j \notin K'\}$ and let j_1 be such that $z^e_{j_1} \in tr(Z_e)$. Recall that by definition of E_1, there exists $i, j \in \{1, ..., n\}$ such that $z^e_{j_1}$ is adjacent to a^j_i. We have $\mu(z^e_{j_1}) \geq y + z + 1$, where $y = |N(z^e_{j_1}) \cap Y_e|$ and $z = |N(z^e_{j_1}) \cap Z_e|$. On the other side, we have $\mu(y^e_{j_0}) \leq z + y + 2$ (indeed, $|N(y^e_{j_0}) \cap Z_e| = z + 1$, $|N(y^e_{j_0}) \cap Y_e| \leq y$ and $|N(y^e_{j_0}) \cap X_e| = 1$). Since $\{y^e_{j_0}, z^e_{j_1}\} \in E'$, it holds that $(K' \setminus \{z_{j_1}\}) \cup \{y_{j_0}\}$ is a safe replacement. After all these replacements, given any $e \in E_1$, $tr(Z_e) \neq \emptyset$ implies that $tr(Y_e) = Y_e$.
We now proceed to replacements between gadgets F_e, $e \in E_1$. If one can find $a, b \in E_1$ such that $tr(Z_a) \neq \emptyset$ and $tr(Y_b) \neq Y_b$, then let j_0 be such that $y^b_{j_0} \notin tr(Y_b)$ and let j_1 be such that $z^a_{j_1} \in tr(Z_a)$. We have $\mu(z^a_{j_1}) \geq T + 1$ and $\mu(y^b_{j_0}) \leq T - 1$. Thus $(K' \setminus \{z_{j_1}\}) \cup \{y_{j_1}\}$ is a safe replacement. □

A.3 End of $\mathcal{NP}$-hardness proof: the three other cases

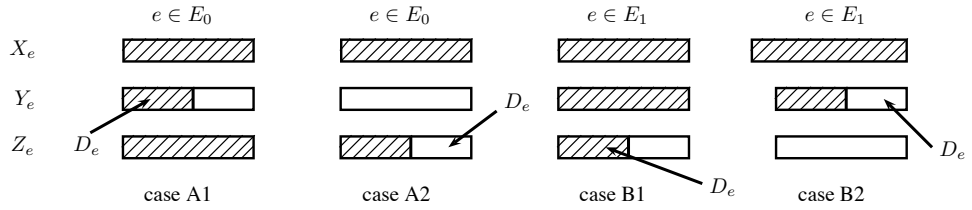

Fig. 5: Schema of different cases. Shaded rectangles represent parts of K'.

Case A2 and B2
Let $\Delta_0 = \sum_{e \in E_0} |D_e|$, $\Delta_1 = \sum_{e \in E_1} |D_e|$ and $\Delta = \Delta_0 + \Delta_1$ (recall that in this case, $D_e \not\subset K'$ for all $e \in E$). Here again we suppose $\Delta > 0$. Let us notice that for all $u \in tr(A)$, $\mu(u) \geq T$. On the other hand, for all $e \in E$ such that there exists $v \in D_e$, we have $\mu(v) \leq T$ (remark that if $e \in E_1$, then $D_e \subseteq Y_e$, and if $e \in E_0$, then v is not adjacent to $tr(A)$ by definition of E_0). Thus $(K' \backslash \{u\}) \cup \{v\}$ is a safe replacement. Since before this replacement we had $tr(A) = T + \Delta$, it is clear that we can repeat this replacement (*i.e.* $K' \backslash \{u\} \cup \{v\}$ where $u \in tr(A)$ and $v \in D_e$ for some $e \in E$) Δ times safely. At this point, the updated value of Δ is 0, *i.e.* $D_e = \emptyset$ for all $e \in E$. By Lemma 8, we must have a clique of size k in G.

Case A2 and B1
If there exists $e \in E_0$ such that there exists $u \in D_e$, then $\mu(u) < T$. If such a vertex exists, then either $|tr(A)| > T$ or there exists $e' \in E_1$ such that there exists $v \in D_{e'}$. In the first case for all $x \in tr(A)$ we have $\mu(x) \geq T$, and $(K' \setminus \{x\}) \cup \{u\}$ is a safe replacement. In the second case we have $\mu(v) > T$ and here again $(K' \setminus \{v\}) \cup \{u\}$ is a safe replacement.
After these replacements we must have $D_e = \emptyset$ for all $e \in E_0$, and we can apply the same arguments as for case A1 and B1.

Case A1 and B2
If there exists $e \in E_1$ such that there exists $u \in D_e$, then $\mu(u) < T$. If such a vertex exists, then either $|tr(A)| > T$ or there exists $e' \in E_0$ such that there exists $v \in D_{e'}$. In the first case for all $x \in tr(A)$ we have $\mu(x) \geq T$, and $(K' \setminus \{x\}) \cup \{u\}$ is a safe replacement. In the second case we have $\mu(v) > T$ and here again $(K' \setminus \{v\}) \cup \{u\}$ is a safe replacement.
After these replacements we must have $D_e = \emptyset$ for all $e \in E_1$, and we can apply the same arguments as for case A1 and B1.

B PTAS for Proper Intervals Graphs

In this section we design a PTAS for Sparsest k-Subgraph in proper interval graphs. We first assume that the instance has one connected component. We prove that we can re-structure an

optimal solution Opt into a near optimal solution Opt' such that the pattern used in Opt' in each "block" (a block corresponds to a separator in the input graph) is simple enough to be enumerated in polynomial time. Then, a dynamic programming algorithm will process the graph blocks by blocks from left to right and enumerate for each one all the possible patterns.

Definitions Let us define some notations that will be used in the algorithm. Recall that we are given a set of proper intervals $\mathcal{I} = \{I_1, ..., I_n\}$ sorted by their right endpoints (and by their left endpoints equivalently).

First, we define by induction the following decomposition of the input graph (see Figure 6). Let $I_{m_1} = I_1$, $L_1 = I_{m_1}$, $R_1 = \{I_j, j > m_1, I_j \text{ overlaps } I_{m_1}\}$. Then, given any $i \geq 1$ we define (while there remains some intervals after R_i):

- $I_{m_{i+1}}$ is the rightmost interval of the set $X = \{I \notin R_i, \exists I' \in R_i \text{ s.t. } I \text{ overlaps } I'\}$ (X is well defined as the instance has a unique connected component)
- $L_{i+1} = \{I_j, j \leq m_{i+1}, I_j \text{ overlaps } I_{m_{i+1}} \text{ and } I_j \notin R_i\}$
- $R_{i+1} = \{I_j, j > m_{i+1}, I_j \text{ overlaps } I_{m_{i+1}}\}$.

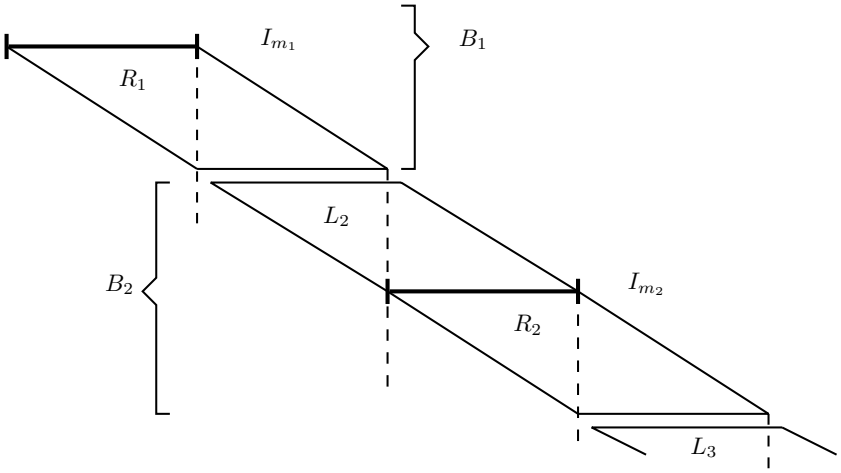

Fig. 6: Schema of the decomposition used in the $PTAS$.

Let a denote the maximum i such that I_{m_i} is defined. Notice that R_a may be empty, and that $I_{m_i} \in L_i$ for all $i \in \{1, ..., a\}$.

For any $i \in \{1, ..., a\}$ we define the block i as $B_i = L_i \bigcup R_i$. Thus, the set of intervals is partitioned into blocs B_i for $1 \leq i \leq a$.

For any $1 \leq i \leq a$ and any solution S (a subset of k intervals), let $L_i^S = L_i \bigcap S$, $R_i^S = R_i \bigcap S$, and $B_i^S = B_i \bigcap S$.

Notice that for any S and i, intervals of R_i^S do not intersect intervals of R_{i-1}^S, and intervals of L_i^S do not intersect $I_{m_{i-1}}$ nor intervals of L_{i-1}^S.

We can now write the cost of a solution S as the sum of the costs inside the blocks and the costs between the blocks. Thus, we have $cost(S) = \sum_{i=1}^{a} cost(B_i^S) + \sum_{i=1}^{a-1} cost(R_i^S, L_{i+1}^S)$. Indeed, by definition, the only edges between blocks B_i and B_{i+1} are edges between R_i and L_{i+1}.

Compacting blocks Let $Comp$ be an injective function from $\mathcal{I}$ to $\mathcal{I}$. For any $S \subseteq \mathcal{I}$, we define $Comp(S) = \bigcup_{I \in S} Comp(I)$. The function $Comp$ is called a *compaction* if for any $S \subseteq \mathcal{I}$ and any $1 \leq i \leq a$ the following holds:

- for all $I \in R_i^S$ we have $Comp(I) \in R_i$ and $r(Comp(I)) \leq r(I)$.
- for all $I \in L_i^S$ we have $Comp(I) \in L_i$ and $r(I) \leq r(Comp(I))$.

Roughly speaking, a compaction "pushes" intervals of B_i^S toward the center I_{m_i}. The idea is that a compaction may increase the cost of a solution inside the blocks, but cannot increase the costs between the blocks. Thus, let us define a ρ-compaction as a compaction $Comp$ such that for any $S \subseteq \mathcal{I}$ and for all $i \in \{1, ..., a\}$ we have $cost(Comp(B_i^S)) \leq \rho.cost(B_i^S)$.

Lemma 9. *If $Comp$ is a ρ-compaction, then for any solution S, $cost(Comp(S)) \leq \rho.cost(S)$.*

Proof. By definition of the decomposition, we have

$$\begin{aligned} cost(Comp(S)) &= \sum_{i=1}^{a} cost(Comp(B_i^S)) + \sum_{i=1}^{a-1} cost(Comp(R_i^S), Comp(L_{i+1}^S)) \\ &\leq \sum_{i=1}^{a} \rho.cost(B_i^S) + \sum_{i=1}^{a-1} cost(Comp(R_i^S), Comp(L_{i+1}^S)) \end{aligned}$$

We now prove that

$$\sum_{i=1}^{a-1} cost(Comp(R_i^S), Comp(L_{i+1}^S)) \leq \sum_{i=1}^{a-1} cost(R_i^S, L_{i+1}^S)$$

Indeed, let $I_R \in R_i^S$ and $I_L \in L_{i+1}^S$ such that I_R and I_L do not overlap. Then by definition of a compaction, we have $r(Comp(I_R)) \leq r(I_R)$ and $l(I_L) \leq l(Comp(I_L))$. Thus, intervals $Comp(I_R)$ and $Comp(I_L)$ do not overlap as well, which proves the result. □

According to the previous lemma, we only have now to find compactions that preserve costs inside the blocks. Given a fixed ϵ, the objective is now to define a $(1 + \epsilon)$-compaction that has a simple structure.

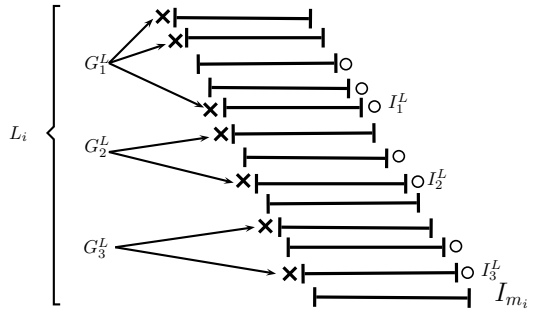

Fig. 7: Example of a compaction of a set X for a block L_i, with $P = 3$, and $x_L = 7$. Intervals marked with a cross represent X. Intervals marked with a circle represent $Comp(X)$.

Lemma 10. *For any fixed $P \in \mathbb{N}$, there is a $(1 + \frac{4}{P})$-compaction such that for any X, $Comp(X)$ can be described by $(2P + 4)$ variables ranging in $\{0, \ldots, n\}$.*

Proof. According to Lemma 9, we only describe $Comp(X)$ for $X \subseteq B_i$, given any $1 \leq i \leq a$. Let $X = X_L \cup X_R$ with $X_L \subseteq L_i$ and $X_R \subseteq R_i$. We define $x_L = |X_L|$, $x_R = |X_R|$. Moreover, we set $x_L = q_L P + r_L$ (with $r_L < P$) and $x_R = q_R P + r_R$ (with $r_R < P$).

Let us split X_L into P subsets $(G_t^L)_{1 \leq t \leq P}$ of consecutive intervals (in the ordering of their right endpoints), with $|G_t^L| = q_L + 1$ for $t \in \{1, ..., r_L\}$ and $|G_t^L| = q_L$ for $t \in \{(r_L + 1), ..., P\}$ (see Figure 7). Similarly, we split X_R into P subsets $(G_t^R)_{1 \leq t \leq P}$ of consecutive intervals, with $|G_t^R| = q_R + 1$ for $t \in \{1, ..., r_R\}$ and $|G_t^R| = q_R$ for $t \in \{(r_R + 1), ..., P\}$.
For all $t \in \{1, ..., P\}$, let I_t^L (resp. I_t^R) be the rightmost (resp. leftmost) interval of G_t^L (resp. G_t^R). The principle of the compaction is to flush every intervals of G_t^L (resp. G_t^R) to the right (resp. left). Thus, for $t \in \{1, ..., r_L\}$, $Comp(G_t^L)$ is defined as the $(q_L + 1)$-rightmost intervals I such that $r(I) \leq r(I_t^L)$, and for $t \in \{(r_L + 1), ..., P\}$, $Comp(G_t^L)$ is defined as the q_L-rightmost intervals I such that $r(I) \leq r(I_t^L)$.
Similarly, for $t \in \{1, ..., r_R\}$, $Comp(G_t^R)$ is defined as the $(q_R + 1)$-leftmost intervals I such that $r(I_t^R) \leq r(I)$, and for $t \in \{(r_R + 1), ..., P\}$, $Comp(G_t^R)$ is defined as the q_R-leftmost intervals I such that $r(I_t^R) \leq r(I)$. The construction for a block L_i is depicted in Figure 7. It is clear that the mapping $Comp$ described above is a compaction. Moreover, given x_L, r_L, x_R, r_R and I_t^L (resp. I_t^R) for all $1 \leq t \leq P$, we are clearly able to construct $Comp(X)$ in polynomial time. Thus, it remains to prove that $Comp$ is a $(1 + \frac{4}{P})$-compaction.

One can easily show the two following key arguments:

(i) all intervals of L_i form a clique, as well as all intervals of R_i.
(ii) for any $t_1, t_2 \in \{1, ..., P\}$ with $t_1 \neq P$ and $t_2 \neq 1$, if an interval of $Comp(G_{t_1}^L)$ overlaps an interval of $Comp(G_{t_2}^R)$, then for any $s_1 \in \{(t_1 + 1), ..., P\}$ and any $s_2 \in \{1, ..., (t_2 - 1)\}$, all intervals of $G_{s_1}^L$ overlap all intervals of $G_{s_2}^R$.

For all $t \in \{1, ..., P\}$, we define $x_t^L = |G_t^L| = |Comp(G_t^L)|$, $x_t^R = |G_t^R| = |Comp(G_t^R)|$. By our construction and (i), we have

$$cost(Comp(X)) \leq \binom{x_L}{2} + \binom{x_R}{2} + \sum_{t=1}^{P} cost(Comp(G_t^L), Comp(X) \cap R_i)$$

$$cost(X) \geq \binom{x_L}{2} + \binom{x_R}{2} + \sum_{t=1}^{P} cost(G_t^L, X \cap R_i)$$

Then, for all $t \in \{1, ..., P\}$, let $\lambda_t \in \{0, 1, ..., P\}$ be the maximum s such that an interval of $Comp(G_t^L)$ overlaps an interval of $Comp(G_s^R)$ (we set $\lambda_t = 0$ if no interval of $Comp(G_t^L)$ overlaps an interval of $Comp(G_1^R)$). By (ii), for all $t \in \{1, ..., P\}$, we have $cost(Comp(G_t^L), Comp(X) \cap R_i) \leq x_t^L \sum_{u=1}^{\lambda_t} x_u^R$ and for all $t \in \{2, ..., P\}$, we have $cost(G_t^L, X \cap R_i) \geq x_t^L \sum_{u=1}^{\lambda_{t-1}-1} x_u^R$ (since some intervals of G_{t-1}^L overlap some intervals of $G_{\lambda_{t-1}}^R$, it implies that all intervals of G_t^L overlap all intervals of $G_{\lambda_{t-1}-1}^R$).

Combining the previous inequalities, we now have

$$cost(Comp(X)) \leq \binom{x_L}{2} + \binom{x_R}{2} + \sum_{t=1}^{P} x_t^L \sum_{u=1}^{\lambda_t} x_u^R$$

$$cost(X) \geq \binom{x_L}{2} + \binom{x_R}{2} + \sum_{t=2}^{P} x_t^L \sum_{u=1}^{\lambda_{t-1}-1} x_u^R$$

Thus, we have

$$\Delta = cost(Comp(X)) - cost(X) \leq x_1^L \sum_{u=1}^{\lambda_1} x_u^R + \sum_{t=2}^{P} x_t^L \sum_{u=\lambda_{t-1}}^{\lambda_t} x_u^R$$

As in our case we have $x_t^L \leq (q_L + 1)$, we get $\Delta \leq (q_L + 1)(\sum_{u=1}^{\lambda_P} x_u^R + \sum_{u=1}^{P} x_{\lambda_u}^R) \leq 2(q_L + 1)x_R \leq 2(\frac{x_L}{P} + 1)x_R$.

It remains now to handle particular cases, according to the values of x_L and x_R.

- If $x_L \geq P$, then $2(\frac{x_L}{P} + 1)x_R \leq \frac{4}{P} x_L x_R$, and $\frac{\Delta}{cost(X)} \leq \frac{\frac{4}{P} x_L x_R}{(x_L - 1)x_L + (x_R - 1)x_R} \leq \frac{\frac{4}{P} x_L x_R}{\frac{1}{2}(x_L^2 + x_R^2)} \leq \frac{4}{P}$ (we lower bounded $(x_R - 1)$ by $\frac{x_R}{2}$ as cases with $x_R \leq 1$ lead to even better ratio).
- If $x_L < P$, then we set $Comp(X \cap L_i) = X_L$ (*i.e.* we keep the left part unchanged). If $x_R < P+1$, then we set $Comp(X) = X$ and we get a 1-compaction. Notice that in these cases we are still able to construct $Comp(X)$ in polynomial time. Suppose now that $x_R \geq P+1$. One can improve the previous lower bound and write $cost(X) \geq \frac{(x_L - 1)x_L}{2} + \frac{(x_R - 1)x_R}{2} + \sum_{t=1}^{P} x_t^L (\sum_{u=1}^{\lambda_t - 1} x_u^R)$. Indeed, for all $t \in \{1, ..., x_L\}$ the set G_t^L is a singleton (and $G_t^L = \emptyset$ for $t \in \{x_L + 1, ..., P\}$), and thus the interval of G_t^L overlaps some intervals of $G_{\lambda_t}^R$, which implies that it overlaps all intervals of $G_{\lambda_t - 1}^R$. Thus, we get $\Delta \leq \sum_{t=1}^{P} x_t^L x_{\lambda_t}^R \leq \sum_{t=1}^{x_L} x_{\lambda_t}^R \leq x_R$, and $\frac{\Delta}{cost(X)} \leq \frac{2x_R}{(x_L - 1)x_L + (x_R - 1)x_R} \leq \frac{2}{P}$, which terminates the proof of the lemma.

□

Algorithm Let us now write a dynamic programming algorithm for the instances that have a unique connected component (we will drop this hypothesis after). Let *Opt* be an optimal solution, P a fixed integer and *Comp* the previous $(1 + \frac{4}{P})$-compaction. The algorithm constructs a solution which is at least as good as $Comp(Opt)$ by enumerating for all blocks all the possible compacted patterns (*i.e.* all the possible $Comp(X)$).

Let us now define more formally the algorithm, starting with the parameters. The first parameter $k' \leq k$ is the number of interval to choose. i is the starting block, meaning that the k' interval must be chosen in $\bigcup_{l=i}^{a} B_l$. Finally, B_{i-1}^S represents the set of $2P+4$ variables that encode the set of intervals X_{i-1} chosen in block $(i-1)$. Since we can construct X_{i-1} from B_{i-1}^S in polynomial time, we will directly use B_{i-1}^S to denote X_{i-1}, for the sake of readability.

Algorithm 2 $DP(k', i, B_{i-1}^S)$

// For the sake of clarity we drop the classical operations related to the "marking
// table" that avoid multiple computations with same arguments
// We also drop the base case $i = a+1$ (*i.e.* there are no more remaining intervals in the instance)
$\Omega \leftarrow$ all possible patterns for block i using less or equal than k' intervals
return $\arg\min_{B \in \Omega} cost(B_{i-1}^S \cup B \cup DP(k' - |B|, i+1, B))$

Lemma 11. *For any P, $DP(k, 1, \emptyset)$ outputs a $(1+\frac{4}{P})$-approximation for the* Sparsest k-Subgraph *in $n^{\mathcal{O}(P)}$.*

Proof. The objective is to prove that $cost(DP(k,1,\emptyset)) \leq cost(Comp(Opt))$, where $Comp$ is the previous $(1+\frac{4}{p})$-compaction. According to Lemma 10, it is sufficient to get a $(1+\frac{4}{p})$-approximation.

For sake of readability, for all $i \in \{1, ..., a\}$, we define $B_i^* = Comp(Opt) \cap B_i$ and $k_i^* = |\bigcup_{l=i}^{a} B_i^*|$.

We prove by induction on i (starting from $i = a+1$) that $cost(B_{i-1}^* \cup DP(k_i^*, i, B_{i-1}^*)) \leq cost(Comp(Opt) \cap \bigcup_{l=i-1}^{a} B_l)$.

Let us suppose that the hypothesis is true for $i+1$ and prove it for i. Considering the iteration where DP chooses $B = B_i^*$.

$$cost(B_{i-1}^* \cup DP(k_i^*, i, B_{i-1}^*)) \leq cost(B_{i-1}^*) + cost(B_{i-1}^*, B_i^*) + DP(k_i^* - |B_i^*|, i+1, B_i^*)$$

(recall that $cost(X_1, X_2) = |\{(I_l, I_{l'}) \in E, I_l \in X_1, I_{l'} \in X_2\}|$). Using the induction hypothesis we get the desired result.

The dependency in P in the running time is due to the $n^{2P+\mathcal{O}(1)}$ possible values for the set of parameters and the branching time in $n^{2P+\mathcal{O}(1)}$ when enumerating sets B_i^S. □

Finally, let us extend the previous result to instances having several connected component. We only sketch briefly the algorithm as it follows the same idea as [8] for the k-densest subgraph problem.

Let us suppose that for any $k' \leq k$ we have an algorithm $A(k', X)$ which is a ρ-approximation for k'-sparsest subgraph on a instance X having one connected component.

Let $(C_i)_{1 \leq i \leq x}$ denote the connected components of a (general) instance of Sparsest k-Subgraph. It is sufficient to define a dynamic programming algorithm $DP(k', i)$ that computes a ρ approximation of the k'-sparsest subgraph on $\bigcup_{t=i}^{x}(C_t)$ by keeping the best of all the $\mathcal{A}(l, C_i) + DP(k' - l, i+1)$, for $1 \leq l \leq k'$. Thus, we get the following result:

Theorem 4. *There is a PTAS for* Sparsest k-Subgraph *on proper interval graphs running in $n^{\mathcal{O}(\frac{1}{\epsilon})}$*