Exact Algorithms for Maximum Transitive Subgraph Problem

Sourav Chakraborty[1] and Nitesh Jha[2]

[1]Chennai Mathematical Institute, Chennai, India, e-mail: {sourav, nj}@cmi.ac.in

We study the problem of computing a *Maximum Transitive Subgraph (MTS)* of a given directed graph. This problem is known to be NP-hard. We give an algorithm that runs in time $O(4^{k^2} n^2)$ to output an MTS for a graph with treewidth k.

1. Introduction

Given a directed graph $G = (V, E)$, a subgraph S of G is said to be transitive if for every pair of edges $u \to v$ and $v \to w$ in S, the edge $u \to w$ is also present in S. S is called a *Maximum Transitive Subgraph (MTS)* if it is of the largest size (number of edges) possible. The same problem can also be posed in a weighted setting where edges have weights. Our goal in this article is to compute an MTS of a given graph.

The transitivity structure in a binary relation (directed graph) is a fundamental object that has a rich history in multiple areas of mathematics and computer science. Since transitivity is a desired structure, it is approached in multiple ways. Two most common are transitive closures and transitive subgraphs. The problems can then be posed in the form of an optimal or approximate solution. Problems have also been studied under the notion of *distance* from a transitive structure.

The problem of computing an MTS for a given graph is a well known NP-hard problem [9]. The recent work [2] gives a simple 0.25-approximation algorithm of obtaining an MTS in a general graph. For the case where the underlying undirected graph is triangle free, it gives a 0.874-approximation for the MTS problem. The idea there is to look at the related problem of directed maximum cuts in the same graph. We continue the study of algorithms for computing the MTS under different input restrictions.

Our main goal in this article is to understand the *parameterized complexity* of the MST problem. A parameterization of a problem assigns an integer k to each input instance I and we say that a the problem is *fixed-parameter tractable* if there is an algorithm that solves the problem in time $f(k) \cdot |I|^{O(1)}$. Here, f is any computable function. First systematic study of parameterized complexity was done by Downey and Fellows [5]. More recent account of the field can be found in the texts [6, 7, 4].

Arnborg et al. [1] showed that the problem of MST is fixed parameter tractable. They give an alternate proof of Courcelle's theorem [3] and express the MST problem in *Extended Monadic Second Order*, thus giving a meta-algorithm for the problem. This algorithm is not explicit and f is known to be only a computable function.

In the context of transitivity, [8] shows that the problem of deciding whether a directed graph has a transitive induced subgraph of size k is fixed-parameter tractable.

In Appendix A, we give a poly-time algorithm for computing an MTS in a directed tree. In Appendix B, we give our first generalisation. For a given directed graph with treewidth at most k, we give an algorithm which is runs in time $O(n^{k^2})$. The idea here is to recursively use separators and combine the solutions of the two parts. We improve this algorithm in Section 2, where we give an algorithm that runs in time $O(4^{k^2}n^2)$ to output an MTS for a graph with treewidth k. The main result is stated below.

Theorem 1. *There exists an algorithm that runs in time $O(4^{k^2}n^2)$ to output an MTS for a graph with treewidth k.*

Notation

For any set S and $x \in S$, define $S - x = S \setminus \{x\}$. Let $G = (V, E)$ be a given directed graph. For $v \in V, e \in E$, define $G - v$ to be the graph obtained by removing the vertex v from G and $G \setminus e$ represents the graph obtained by deleting the edge e from G. The notation $F \subseteq G$ defines a subgraph F of G. For any subgraph F of G, $V(F)$ defines the vertex set of F and $E(F)$ defines the edge set of F. For $U \subseteq V$, $G(U)$ defines the induced subgraph on U.

For $A, B \subseteq V$, define $E(A, B) = \{u \to v : u \in A, v \in B\}$ and $\mathcal{E}(A, B) = E(A, B) \cup E(B, A)$. In the context of transitivity, we say that the two-path $u \to v \to w$ is *complete* if $u \to w \in E$. If $u \to w \notin E$, the two-path is called *incomplete*.

2. MTS is FPT Parameterised by Treewidth

We first introduce the basics of tree decomposition of an undirected graph $G = (V, E)$. We borrow the notations from [4]. Let $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ be a tree decomposition, where T is a tree whose every node t is assigned a vertex subset $X_t \subseteq V(G)$, called a bag, such that the following three conditions hold:

1. $\cup_{t \in V(T)} X_t = V(G)$.
2. $\forall$ edge $(u, v) \in E(G), \exists t \in T$ with $u, v \in X_t$.
3. $\forall u \in V(G)$, the set $T_u = \{t \in V(T) : u \in X_t\}$ induces a connected subtree of T.

Further, we use what is called a *nice* tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ of G. Such a decomposition has the following properties.

1. The leaf and the root nodes are empty.
2. For every non-leaf node t is one of the following types:
   a) Introduce: t has exactly one child t' with $X_t = X_{t'} \cup \{u\}$ for some $u \notin X_{t'}$
   b) Forget: t has exactly one child t' with $X_t = X_{t'} \setminus \{u\}$ for some $u \in X_{t'}$
   c) Join: t has two children t' and t'' with $X_t = X_{t'} = X_{t''}$.

We perform a bottom up dynamic programming on $\mathcal{T}$ starting from the leaves and ending at the root. We describe the calculations performed at a node of any type (categorised above) using the computation performed at the children nodes. For any node $t \in \mathcal{T}$, denote by V_t the union of the bags associated with all the nodes in the subtree rooted at t, including X_t. Let G_t define the induced graph on V_t.

We define a table entry $m[t, F, I, O, U, Y]$ for each node $t \in \mathcal{T}$, for each transitive subgraph $F \subseteq E(X_t)$ and for each partition (I, O, U, Y) of X_t. The entry $M = m[t, F, I, O, U, Y]$ contains the MTS on $G(V_t)$ with the restriction that $G(X_t) \cap M = F$ and in M,

- no edges from the set $E(I, V_t \setminus X_t)$ are allowed,
- no edges from the set $E(V_t \setminus X_t, O)$ are allowed,
- any edge from the set $\mathcal{E}(V_t \setminus X_t, U)$ is allowed, and
- no edges from the set $\mathcal{E}(V_t \setminus X_t, Y)$ are allowed.

Vertices in U are said to be *unrestricted*. This partitioning has been defined to support the *Join* operation at a join node in the tree decomposition. The idea is same as before - avoid two-paths across separated partitions and avoid *incomplete* two-paths in the separator.

Leaf node: The only valid cell entry here is $m[t, \phi, \phi, \phi, \phi, \phi] = \phi$.

Introduce node: Suppose node t has child node t' such that $X_t = X_{t'} \cup \{v\}$. For any given partition (I, O, U, Y) of X_t and $F \subseteq G(X_t)$, we need to compute $m[t, F, I, O, U, Y]$.

First notice that the introduced vertex v can have edges to only the vertices in the set $X_{t'}$. Also, by definition, $MTS(G_t) \cap G(X_t) = F$. Applying both these conditions together, if v is not in $V(F)$, no edge incident on v can be included in the MTS of G_t. Hence, for $v \notin V(F)$,

$$m[t, F, I, O, U, Y] = m[t', F, I - v, O - v, U - v, Y - v]$$

Now we consider the case where $v \in V(F)$. For a vertex $r \in V(G)$ and $X \subseteq E(G)$, define the in-neighbours of r in X as $N^i_X(r) = \{s : s \to r \in X\}$ and the out-neighbours of r in X as $N^o_X(r) = \{s : r \to s \in X\}$. Define $N_X(r) = N^i_X(r) \cup N^o_X(r)$.

Consider the set $N_{X_t}(v) \setminus N_F(v)$. These neighbours of v in X_t, wherever they may lie in the partition (I, O, U, Y), can be kept as is in their designated partitions for recursion. The argument is as follows. Consider $u \in N_{X_t}(v) \setminus N_F(v)$. We want to check if any edge through u breaks transitivity. If $u \notin V(F)$, then u does not interact with any other vertex in X_t by definition and hence transitivity is maintained as before. If $u \in V(F)$, any edge in F incident on vertex u is already a part of a transitive set since F is transitive by definition.

We now deal with the set $N_F(v)$. Consider a vertex $u \in N_F(v)$. Following useful cases arise.

1. $u \in I \cap N^i_F(v)$: Here, an edge passing through u may break the transitivity. Such a vertex u must be removed from I and placed in O.
2. $u \in O \cap N^o_F(v)$: This is similar to the last case. We should move u from O to I.
3. $u \in U \cap N_F(v)$: Since the edges $\mathcal{E}(u, V_t \setminus X_t)$ are unrestricted to participate in an MTS, transitivity may break in two ways. Incomplete two-path of the form $r \to u \to v$ or $v \to u \to r$ where $r \in V_t \setminus X_t$ may result. We should disallow these cases.
4. $u \in Y \cap N_F(v)$: This case is fine as $\mathcal{E}(u, V_t \setminus X_t) = \phi$.

We incorporate all these restrictions in the following computation.

$$\begin{aligned}
F' &= F - v \\
I' &= (I \setminus N^i_F(v)) \cup (O \cap N^o_F(v)) \cup (U \cap N^o_F(v)) \\
O' &= (O \setminus N^o_F(v)) \cup (I \cap N^i_F(v)) \cup (U \cap N^i_F(v)) \\
U' &= U \setminus N_F(v) \\
Y' &= Y
\end{aligned}$$

The update method is then $m[t, F, I, O, U, Y] = m[t', F', I', O', U', Y'] \cup F$.

Forget Node: Suppose node t has child t' such that $X_t = X_{t'} \setminus \{v\}$. We update the current entry as follows:

$$m[t, F, I, O, U, Y] = \max m[t', F', I', O', U', Y']$$

where the maximum is over the following conditions:

$$\begin{aligned} F'|_{X_t} &= F \\ I' = I, O' &= O, Y' = Y \\ U' &= U \cup \{v\} \end{aligned}$$

Here, the transitive set F' is allowed to include the vertex v resulting in the condition $F'|_{X_t} = F$. We also allow v to have unrestricted edges since this effectively covers all the cases - only incoming edges on v, or only outgoing edges from v, or the case where v has both incoming and outgoing edges.

Join Node: Suppose node t has children t_1 and t_2 such that $X_t = X_{t_1} = X_{t_2}$. Define arbitrary partitions (to be fixed below) $X_{t_1} = I' \uplus O' \uplus U' \uplus Y'$ and $X_{t_2} = I'' \uplus O'' \uplus U'' \uplus Y''$. We have the following rule for updating the current entry:

$$m[t, F, I, O, U, Y] = \max(m[t_1, F, I', O', U', Y'] \cup m[t_1, F, I'', O'', U'', Y''])$$

under the restriction that: $I' \supseteq I, I'' \supseteq I$ and $O' \supseteq O, O'' \supseteq O$

For each vertex $v \in U$, one of the following is true: $v \in I' \cap I''$, or $v \in O' \cap O''$, or $v \in U' \cap Y''$, or $v \in Y' \cap U''$.

Here, we keep the F same in both t_1 and t_2 as required. In order to join at any vertex v in I, we demand such a vertex must be present in both I' and I'' but we also allow these sets to be larger. This is required as this leaves the possibility of a larger combination while transitivity is still maintained. A similar restriction is employed on O' and O''.

The vertices v in U are unrestricted but we need to be careful while using unrestricted vertices in the join operation. Such a vertex should only be allowed to be unrestricted on one side but completely isolated on the other side. This gives us the possibilities $v \in U' \cap Y''$ or $v \in Y' \cap U''$. But this restriction forbids the possibility of edges being used on both sides of v. Such a case could occur if v uses only incoming (or outgoing) edges on both the sides. To accommodate this, we have the options of $v \in I' \cap I''$ or $v \in O' \cap O''$. Finally, we take the maximum over all the legitimate join operations.

We now estimate the running time of our algorithm. Assuming the input graph has treewidth k, each node X_t is of size at most $k + 1$. The number of partitions of type (I, O, U, Y) of X_t is at most 2^{k+4}. The number of transitive subgraphs F of X_t is at most 2^{k^2}. A single update of $m[\cdot]$ at any node can be done in at most n^2 steps. So a simple upper bound to the time complexity is $4^{k^2} n^2$.

3. Conclusion

In this article, we have continued the systematic study of computing a Maximum Transitive Subgraph of a given directed graph addressed recently in [2]. We show that this problem is fixed-parameter tractable when parameterized by treewidth. In particular, we give an algorithm that runs in time $O(4^{k^2} n^2)$ to output an MTS for a graph with treewidth k. An immediate question that arises is – whether we can reduce the exponent k^2 to $O(k)$.

Another interesting question that we have not addressed here is a lower bound for this problem. It would be interesting to arrive at any lower bound under the standard assumption of ETH.

References

[1] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Problems easy for tree-decomposable graphs (extended abstract). In *Automata, Languages and Programming, 15th International Colloquium, ICALP88, Tampere, Finland, July 11-15, 1988, Proceedings*, pages 38–51, 1988.

[2] Sourav Chakraborty, Shamik Ghosh, Nitesh Jha, and Sasanka Roy. Maximal and maximum transitive relation contained in a given binary relation. In *Computing and Combinatorics - 21st International Conference, COCOON 2015, Beijing, China, August 4-6, 2015, Proceedings*, pages 587–600, 2015.

[3] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

[4] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer, 2015.

[5] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity.* Monographs in Computer Science. Springer, 1999.

[6] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[7] Rolf Niedermeier. *Invitation to fixed-parameter algorithms.* Oxford University Press, 2006.

[8] Venkatesh Raman and Somnath Sikdar. Parameterized complexity of the induced subgraph problem in directed graphs. *Inf. Process. Lett.*, 104(3):79–85, 2007.

[9] Mihalis Yannakakis. Node- and edge-deletion np-complete problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 253–264, 1978.

A. MTS in Trees

In the following discussion, an edge-rooted tree is a rooted tree in which the root has only one child. We identify the root of an edge-rooted tree with a root edge e and denote the tree as T_e.

The problem is to compute an MTS of a given directed tree T (the underlying undirected graph is a tree). We can root the tree at a vertex which has only outgoing edges. Let this root be r and $e_1, \ldots, e_l$ denote the outgoing edges from r. Denote the edge-rooted trees at r by $T_{e_1}, \ldots, T_{e_l}$.

Note that the $T_{e_i}'s$ are completely independent of each other and hence their MTS can be computed independently.

Since there are only outgoing edges from the root r,

$$MTS(T) = \bigcup_{i\in[l]} MTS(T_{e_i})$$

We now describe how to compute $MTS(T_{e_i})$. We can divide the set of transitive subgraphs of T_{e_i} into two subsets and compute their maximums:

1. $MTS^+(T_{e_i})$: maximum over transitive subgraphs that include the edge e_i,
2. $MTS^-(T_{e_i})$: maximum over transitive subgraphs that exclude the edge e_i.

Then,

$$MTS(T_{e_i}) = \max\{MTS^+(T_{e_i}), MTS^-(T_{e_i})\}$$

The maximum here is over the size of the transitive sets.

Computing $MTS^+(T_{e_i})$

This is further divided into two cases.

Case 1: e_i is a 'down' edge

Let e_i be the edge $u \to v$ such that the tree T_{e_i} is rooted at the vertex u. Let the vertex v has outgoing edges $c_1, \ldots, c_s$ and incoming edges $c'_1, \ldots, c'_t$. Then,

$$MTS^+(T_{e_i}) = \left[\cup_{i\in[s]} MTS^-(T_{c_i})\right] \bigcup \left[\cup_{i\in[t]} MST(T_{c'_i})\right]$$

The reason we don't include the edges c_i in the first union in this calculation is because this will lead to a possible inclusion of 2-path (whose first edge is e_i) in the MTS set.

Case 2: e_i is an 'up' edge

Here $e_i = v \to u$ is in 'upward' direction of the rooted tree T_{e_i}. The algorithm is symmetrical to Case 1.

Computing $MTS^-(T_{e_i})$

Here, the direction of e_i is not important. Let the tree T_{e_i} be rooted at u and v be the other vertex of edge e_i. Let the vertex v has outgoing edges $c_1, \ldots, c_s$ and incoming edges $c'_1, \ldots, c'_t$. Let,

$$M_1 = \left[\cup_{i\in[s]} MTS(T_{c_i})\right] \bigcup \left[\cup_{i\in[t]} MST^-(T_{c'_i})\right]$$

$$M_2 = \left[\cup_{i\in[s]} MTS^-(T_{c_i})\right] \bigcup \left[\cup_{i\in[t]} MST(T_{c'_i})\right]$$

Then, we have,

$$MTS^-(T_{e_i}) = \max\{M_1, M_2\}$$

The base cases are defined naturally.

Dynamic programming over edge-rooted trees

Though we defined the solution using a top-down approach, we observe that the subproblems are calculated every time there is a call of type $T(e_i)$. The same subproblem is called many times as part of computation of other subproblems. To avoid this, we do the actual computation in bottom up manner. For each edge-rooted tree T_e, we keep in memory the set $MST(T_e)$. The order of computation of is as follows. We do a breadth-first search at root vertex r. We compute all the $MST(T_e)$ for edges at the largest level first. These sets are just the edges themselves. In next stage, we decrease the level by 1. We go on doing this until we compute the $MST(T_e)$ values for edges at the level 0. This gives us a poly-time algorithm to compute an MST of a directed tree.

B. MTS in Bounded Treewidth Graphs: First Attempt

We try to generalize the idea used in case of trees to compute the MTS for graphs. In particular, we want to compute the MTS for weighted-directed graphs whose underlying undirected graph has bounded treewidth. In the case of trees, at every step, we use a vertex that separates a graph into two (or more) disjoint subgraphs. We could then apply the algorithm recursively on these subgraphs and combine the results to compute the MTS for the current tree. For this, we will need the following lemma which ensures that small treewidth implies small balanced separators.

Lemma 2. *If G is a graph with treewidth at most d, then we can find a $1/2$-balanced separator S of G in polynomial time, such that $|S| \leq d+1$.*

The proof of Lemma 2 can be found in Lemma 7.19 in [4]. Let $G = (V, E)$ be a graph with treewidth at most $d-1$. Then we can find a separator S that separates the graph into vertex sets L and R such that, $V = S \uplus L \uplus R$, $S \leq d$, and $L, R \leq |V|/2$. We need the following subgraph notation to define a useful structure which we use in our algorithm. For $A, S \subseteq V$ such that $A \cap S = \phi$, T be a transitive subgraph of $E(S)$ and (I, O, U, Y) be vertex partitioning of S, define $G(A, S, T, (I, O, U, Y))$ to be a subgraph of G such that,

- all the edges in $E(A)$ are included,
- only the edges in T are included from $E(S)$,
- for the edges between the sets A and S, all the edges in $E(A, I), E(O, A), E(A, U), E(U, A)$ are included and no edges between A an Y are included.

We describe Algorithm 1 in detail now. The high level idea is described in Figure 1. After we compute the balanced-separator S, we work separately on $L \cup S$ and $R \cup S$ and combine the MST calculated from these two subproblems to compute the MST for the main problem. In order to be able to combine the two solutions, we force the transitive set from S to be same in both the solutions. We go over every possible subset T of S and recurse on both left and right side such that the solution from both the sides must contain exactly the set T from the edges in S. To do this, we assign the weight ∞ to the edges of T in the recursive calls.

Notice that any combining a solution from left and right (even with a common set T in the separator) may bring in discrepancies in transitivity. For example, a two-path $a \to b \to c$ with

Algorithm 1: *$MTS(G, w)$*: Maximum Transitive Subgraph of weighted digraph G

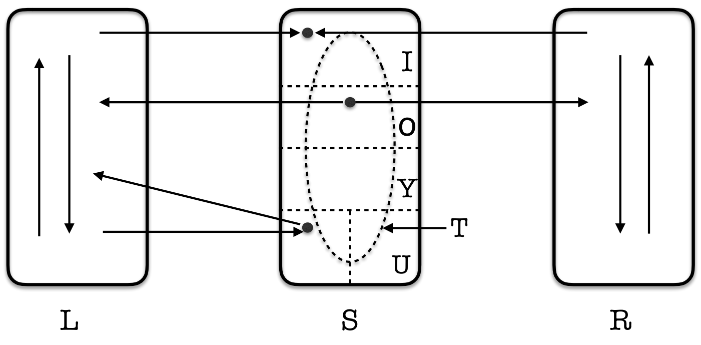
Input : A directed graph $G = (V, E)$ with edge weights $w : E \to \mathbb{N}^+$ such that the underlying undirected graph of G has treewidth at most $d - 1$

Output: An MTS of G

```
1  $max \leftarrow \sum_{e \in E} w(e)$
2  compute a 1/2-separator $S$ of $G$ of size at most $d$, with components $L$ and $R$
3  $M \leftarrow \phi$
4  foreach transitive $T \subseteq E(S)$ do
5      $w' \leftarrow w$
6      foreach $e \in T$ do
7          $w'(e) \leftarrow max$
8      foreach partition $(I, O, U, Y)$ of $S$ do
9          foreach $U' \subseteq U$ do
10             $M' \leftarrow MST(G(L, S, T, (I, O, U', Y \cup (U \setminus U'))), w')$
11                     $\cup\, MST(G(R, S, T, (I, O, U \setminus U', Y \cup U')), w')$
12             if $w(M') > w(M)$ then
13                 $M \leftarrow M'$
14 return $M$
```

$a \in L, b \in S, c \in R$ may creep in. There can be no *completing edge* $a \to c$ since vertices a and c are separated by set S. To take care of this, we partition S into four parts (I, O, U, Y). The subgraph that we pass as argument to the recursive calls has the useful features, such as: only incoming edges are present on vertices in set I. This allows us to combine the results from left and right as no two-paths can have a vertex of I at its center. Similarly we restrict only outgoing edges from the vertices in O. We restrict the vertices in Y to have no edges since a solution may not use all the vertices in S. Finally, we allow a part U' of U to have all the *original* edges on one side and have no edges on the other side. One can think of U' as being *unrestricted* on one side and restricted to have no participation on the other side.

Figure 1: A high level description of Algorithm 1



Theorem 3. *Given (G, w), algorithm $MTS(G, w)$ outputs an MTS of G.*

Proof. We prove it by induction on the number of vertices. For base case, we consider graphs on 3 vertices. It is straightforward to see that the statement is true in this case.

Let H be an MTS of G. Consider any separator S that separates the graph G into L and R (vertex sets). Let $T' = H \cap G(S)$. Notice that $(H \cap G(L)) \cup T'$ is an MTS for $G(L \cup S)$.

Define partition $(I', O', U' = (U'_1, U'_2), Y')$ of S based on the edges in $\mathcal{E}(S, V(H) \cap L)$ and $\mathcal{E}(S, V(H) \cap R)$ and our definition of such a partitioning earlier. Here U'_1 is unrestricted on the left side and U'_2 is unrestricted on the right side.

Now since Algorithm 1 loops over all partitions $(I, O, U = (U_1, U_2), Y)$ of S, it will also hit the particular partition $(I', O', U' = (U'_1, U'_2), Y')$ at one point. At this point a call to $MTS(G(L, S, T, (I', O', U'_1, Y'), w))$ is made. Using induction, $MTS(G(L, S, T, (I', O', U'_1, Y'), w))$ returns the MTS of the graph $G(L, S, T, (I', O', U'_1, Y')$. This would mean that

$$|MTS(G(L, S, T, (I', O', U'_1, Y'), w))| = |(H \cap G(L)) \cup T'|$$

A similar argument can be given for,

$$|MTS(G(R, S, T, (I', O', U'_2, Y'), w))| = |(H \cap G(R)) \cup T'|$$

Since we combine the MTS for $G(L \cup S)$ and $G(R \cup S)$ via the same transitive set T' in the both the cases, we conclude that,

$$|MTS(G, w)| = |H|$$

□

□

We now consider the complexity of Algorithm 1. From Lemma 2, we can assume that each partition is of size at most half of original number of vertices. For a separator of size k, the number of transitive subgraphs inside the separator can be at most 2^{k^2}. The number of 4-partitions of the separator can be at most 2^{k+3}. There two recursive calls, computing their union would need $O(n^2)$ time. This gives us the recurrence $T(n) \leq 2^{k^2} 2^{k+3} k(T(n/2) + O(n^2))$. Solving this, we get a running time of $O(n^{k^2})$.