Finding the maximum-weight induced k-partite subgraph of an i-triangulated graph

Louigi Addario-Berry, William S. Kennedy, Andrew D. King, Zhentao Li, and Bruce Reed

September 24, 2006

Abstract

An *i-triangulated graph* is a graph in which every odd cycle has two non-crossing chords; i-triangulated graphs form a subfamily of perfect graphs. A slightly more general family of perfect graphs are *clique-separable* graphs. A graph is clique-separable precisely if every induced subgraph either has a clique cutset, or is a complete multipartite graph or a clique joined to an arbitrary bipartite graph. We exhibit a polynomial time algorithm for finding the maximum-weight induced k-partite subgraph of an i-triangulated graph, and show that the same problem is NP-complete for clique-separable graphs, even in the unweighted case when $k = 2$.

1 Introduction

Given a graph $G = (V, E)$ with a weight function $w : V \to (0, 1]$, the subgraph *induced* by $V' \subseteq V$ consists of V' together with all edges of E with both endpoints in V'. We denote this graph $G[V']$; it has weight $w(V') = \sum_{v \in V'} w(v)$. Finding a maximum-weight induced k-partite graph of a given graph is NP-complete in general, even when all weights are equal and $k = 2$ – this amounts to finding a maximum-size induced bipartite subgraph.

This latter problem is easily seen to be equivalent to the problem of finding a maximum clique in terms of approximability [6], and is therefore not approximable to within $n^{1-\epsilon}$ for any $\epsilon > 0$ unless $NP = ZPP$ or within $n^{1/2-\epsilon}$ unless $P = NP$ [5]. We say that two chords of a cycle are *non-crossing* precisely if there is a straight-line embedding of the cycle and the two chords such that the edges of the cycle form a convex polygon and the interiors of the chords do not intersect. We consider the general problem of finding a maximum-weight induced k-partite subgraph (MWIKS) for the family of *i-triangulated graphs*, graphs in which every odd cycle has two non-crossing chords. We prove:

Theorem 1. *Given any i-triangulated graph $G = (V, E)$, a maximum-weight induced k-partite subgraph of G can be found in polynomial time.*

To do this we exhibit a polynomial-time algorithm that involves first decomposing G in a manner that satisfies our needs, then applying dynamic programming to the decomposition. Note that k is part of the problem, not part of the input.

The family of i-triangulated graphs is a subfamily of perfect graphs. It is well-known that the weighted stable set problem (i.e. MWIKS for $k = 1$) is solvable in polynomial time on all perfect graphs. As the structure of perfect graphs is now well-understood (c.f. [2]), it may be thought that as a matter of fact MWIKS can be efficiently solved for *all* perfect graphs and all k. We show that this is not the case. There is a closely related superset of i-triangulated graphs called *clique-separable* graphs. A graph G is clique-separable if every induced subgraph H of G either has a clique cutset, or is a complete multipartite graph or is a clique joined to a bipartite graph (the *join* of two graphs consists of adding all possible edges between them). Clique-separable graphs are also perfect graphs. We show:

Theorem 2. *The maximum induced bipartite subgraph problem is NP-complete for clique-separable graphs.*

In Section 2 we present the basic concepts, notation, and results we will use throughout the paper, including some structural properties of i-triangulated graphs. In Section 3 we use this method to show that i-triangulated graphs have *subtree decompositions* with certain useful properties. In Section 4 we show how such subtree decompositions can be used to efficiently find maximum-weight induced k-partite subgraphs via dynamic programming, establishing Theorem 1. Finally, in Section 5 we prove Theorem 2.

2 Structural Basics of i-triangulated Graphs

Unless otherwise specified, a graph G will always have vertex set V and edge set E. We say that a vertex v *sees* a vertex u if u and v are adjacent. The *join* of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ has vertex set $V_1 \cup V_2$ and edge set $E_1 \cup E_2 \cup \{v_1v_2 \mid v_1 \in V_1, v_2 \in V_2\}$. The *neighbourhood of* $v \in V$ is $\{w \mid vw \in E\}$ and is denoted $N(v)$. Given a set of vertices $S \subseteq V$, the *neighbourhood of* v *in* S is $N(v) \cap S$ and is sometimes written $N_S(v)$. A *block* of G is a maximal 2-connected subgraph of G, and a *universal vertex* of G is a vertex that is adjacent to every vertex other than itself.

A *hole* (resp. *k-hole*) of G is an induced subgraph of G which is a chordless cycle of length 4 or more (resp. a chordless cycle of length $k \geq 4$). A *cap* is a cycle C of length at least 4 with exactly one chord vx which forms a triangle with two consecutive edges vw, wx of the cycle; the vertex w is called the *tip* of the cap. C is a k-cap if it is a cap with k vertices. An odd hole or cap is a k-hole or k-cap with k odd. The following fact is immediate from the definition of an i-triangulated graph:

Fact 3. *If a graph is i-triangulated then it contains no odd cap or odd hole.*

The following lemma will help us build our structural decomposition in the next section:

Lemma 4. *Let G be an i-triangulated graph. If $H = u_1 \dots u_k u_1$ is an even hole in G then for any $v \in G$ adjacent to u_1 and u_k, $H \subset N(v)$.*

Proof. Suppose v does not see all of H. Then there must be two consecutive (in the cycle order of H) vertices of H that see v, followed by a maximal set of ≥ 1 consecutive vertices of H that do not see v. If this set has odd size we have an even hole and therefore an odd cap. If the set has even size we have an odd hole. Either case yields a contradiction. $\square$

3 Subtree Decompositions

We define a *base graph* to be a graph that is either a complete multipartite graph with no clique cutset or a clique joined to a bipartite graph. Thus a graph G is clique separable precisely if for every induced subgraph H of G, G has a clique cutset or is a base graph.

Before giving our structural decomposition of i-triangulated graphs, we consider a decomposition of a related class. A *subtree intersection representation* (SIR) of G consists of a tree $T = (N, A)$ and a collection $\mathcal{S} = \{S_v | v \in V(G)\}$ of non-empty subtrees of T such that for any two vertices u and v of G, u and v are adjacent precisely if S_u and S_v intersect. See Figure 1 for an example, and observe that associated with each node $s \in N$ we have the set $W_s = \{v \in V(G) | s \in S_v\}$; we denote the set $\{W_s | s \in N\}$ by $\mathcal{W}$ and we denote the SIR by $[T, \mathcal{W}]$. A graph G is *chordal* if it contains no hole; G has a subtree intersection representation precisely if it is chordal [4]. Observe that in an SIR, for any node $s \in N$, W_s induces a clique, and for any arc $ts \in A$, $W_s \cap W_t$ is a clique cutset so long as neither W_s nor W_t is a subset of the other.

As we need to consider graphs that may not be chordal, we use the more general idea of a *subtree decomposition.* A subtree decomposition of G consists of a tree $T = (N, A)$ and a collection $\mathcal{S} = \{S_v | v \in V(G)\}$ of non-empty subtrees of T such that for any two adjacent vertices u and v of G, S_u and S_v intersect. We define W_s and $\mathcal{W}$ as before and denote the decomposition by $[T, \mathcal{W}]$. A subtree decomposition is *standard* if $W_s \subseteq W_t$ for no arc ts.

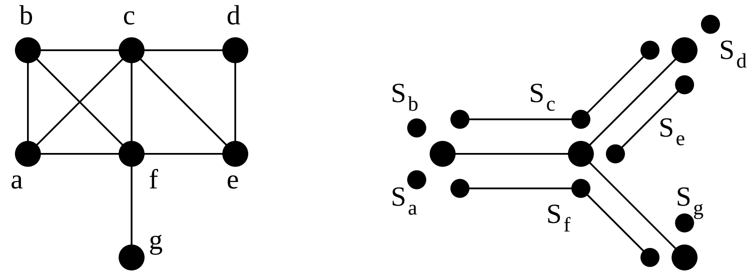

Figure 1: A chordal graph with a subtree intersection representation

Returning now to our given i-triangulated graph G, we wish to build a subtree decomposition $[T, \mathcal{W}]$ of G with certain properties. We will abuse notation and sometimes write W_s to denote $G[W_s]$ for a node s of T. We write U_s for the set of universal vertices in W_s and H_s for $W_s \setminus U_s$.

We insist that our subtree decomposition satisfy the following properties:

(C1) For each node $t \in T$, the graph induced by W_t is a base graph.

(C2) For every arc ts of T, $W_s \cap W_t$ is a clique.

(C3) For each node $t \in T$, at least one of the following must be true:

- (a) $H_t = \emptyset$
- (b) H_t is a 2-connected bipartite graph.
- (c) H_t is a complete multipartite graph with more than 2 parts and each part has more than one vertex.

(C4) For all arcs ts of T, $|(W_s \cap W_t) \setminus U_s| \leq 1$

We call a subtree decomposition satisfying the above requirements *happy*. The aim of this section is to prove the following theorem.

Theorem 5. *Every i-triangulated graph has a happy standard subtree decomposition. Furthermore, such a decomposition can be found in polynomial time.*

First we provide two elementary structural results. The following is a well-known property of sets of subtrees.

Helly Property for Trees. *If $T_1, T_2, \ldots, T_k$ are connected subgraphs of a tree T such that for all $1 \leq i \leq j \leq k$, $T_i \cap T_j \neq \emptyset$, then $\bigcap_{i=1}^{k} T_i \neq \emptyset$.*

We omit the straightforward proof of the Helly Property for Trees, which is by induction on the size of T. An easy consequence is the following:

Lemma 6. *Let G be a graph, C a clique in G and $[T, \mathcal{W}]$ a subtree decomposition of G. Then there exists $t \in T$ such that $C \subseteq W_t$.*

Proof. By definition of a subtree decomposition, if $v_1v_2 \in E(G)$, $T_{v_1} \cap T_{v_2} \neq \emptyset$. Since C is a clique, $\forall v_1, v_2 \in C$, $T_{v_1} \cap T_{v_2} \neq \emptyset$. By the Helly Propery for Trees, $\bigcap_{v \in C} T_v \neq \emptyset$. For any node t in $\bigcap_{v \in C} T_v$, $C \subseteq W_t$. $\square$

To prove Theorem 5 we will construct, in polynomial time, a standard happy subtree decomposition for G, then prove the correctness of our construction. We begin by building a subtree decomposition $[T, \mathcal{W}]$ which satisfies properties C1 and C2.

We do this recursively. First find a minimal clique cutset C in G, or if there is none, let T be an isolated node t with $W_t = V(G)$. Now let V_1 be the vertex set of a component of $G - C$, let $G_1 = G[C \cup V_1]$, and let $G_2 = G[V \setminus V_1]$. Now recursively construct happy standard subtree decompositions $[T_1, \mathcal{W}_1]$ and $[T_2, \mathcal{W}_2]$ of G_1 and G_2 respectively. By Lemma 6 there are two nodes $t_1 \in T_1$ and $t_2 \in T_2$ such that both W_{t_1} and W_{t_2} contain C. Note that $W_{t_1} \cap W_{t_2} = C$. Construct T by joining T_1 and T_2 with an arc between t_1 and t_2, and let $\mathcal{W} = \mathcal{W}_1 \cup \mathcal{W}_2$. If both W_{t_1} and W_{t_2} are the join of a bipartite graph and a clique, $U_{t_1} = U_{t_2}$, and $|H_{t_1} \cap H_{t_2}| > 1$, then contract the arc t_1t_2, replacing W_{t_1} and W_{t_2} with $W_{t_1t_2} = W_{t_1} \cup W_{t_2}$.

It remains to be shown that $[T, \mathcal{W}]$ satisfies (C1) to (C4) and is a standard subtree decomposition. We do this by induction, assuming that $[T_1, \mathcal{W}_1]$ and $[T_2, \mathcal{W}_2]$ are happy standard subtree decompositions. As a basis, $[T, \mathcal{W}]$ is clearly a happy standard subtree decomposition if T is an isolated node since G is clique-separable.

Proof of Theorem 5. If we do not contract the arc t_1t_2 then W_t is a base graph for every $t \in N(T)$, since $\mathcal{W} = \mathcal{W}_1 \cup \mathcal{W}_2$. If we *do* contract t_1t_2, then $[T, \mathcal{W}]$ satisfies (C1) precisely if $W_{t_1t_2}$ is a base graph. This is the case, because we know that $G[H_{t_1} \cap H_{t_2}]$ is a 2-connected bipartite graph since it consists of two 2-connected bipartite graphs pasted together on an edge. Therefore $[T, \mathcal{W}]$ satisfies (C1). Note that for the same reason, G satisfies (C3) if the arc t_1t_2 is contracted, and clearly satisfies (C3) if it is not contracted.

To see that our subtree decomposition satisfies (C2), first note that there is nothing to show if we contract t_1t_2, since for any node s, $W_s \cap W_{t_1t_2}$ is equal to either $W_s \cap W_{t_1}$ or $W_s \cap W_{t_2}$, so the result follows by induction. We only need to show that if t_1t_2 is not contracted then $W_{t_1} \cap W_{t_2}$ is a clique, but we know that $W_{t_1} \cap W_{t_2} = C$, our clique cutset, so $[T, \mathcal{W}]$ satisfies (C2).

Since C is a clique cutset, it is easy to see by induction that $[T, \mathcal{W}]$ is a standard subtree decomposition; we can similarly see that when t_1t_2 is contracted, (C4) is satisfied. All that remains is to prove that $|(W_{t_1} \cap W_{t_2}) \setminus U_{t_1}| \leq 1$ if t_1t_2 is not contracted; it will follow that $|(W_{t_1} \cap W_{t_2}) \setminus U_{t_2}| \leq 1$ by a symmetric argument.

Note that H_{t_1} is 2-connected but not a clique, so if $H_{t_1} \neq \emptyset$ then $|H_{t_1}| \geq 4$ and furthermore it follows from (C3) that for any edge e in H_{t_1} there is an even hole in H_{t_1} containing e. The same is true of H_{t_2}. If $H_{t_1} = \emptyset$ then (C4) is clearly satisfied. Suppose that $|(W_{t_1} \cap W_{t_2}) \setminus U_{t_2}| < 1$ is not satisfied and let u and v be two vertices in $(W_{t_1} \cap W_{t_2}) \setminus U_{t_1}$. If $H_{t_2} = \emptyset$ then since C is a clique cutset there is a vertex $x \in U_{t_2} \setminus W_{t_1}$. This vertex x, along with an even hole in H_{t_1} containing uv, forms an odd cap. Note that x sees only u and v in the even hole because $C = W_{t_1} \cap W_{t_2}$ is a clique cutset. This contradicts Fact 3, so if $|(W_{t_1} \cap W_{t_2}) \setminus U_{t_2}| < 1$ neither H_{t_1} nor H_{t_2} is empty.

Assume, then, that H_{t_2} is a complete multipartite graph containing at least three parts, none of which is a singleton. Since H_{t_1} is not empty there must be an even hole in H_{t_1} containing uv, and since $W_{t_1} \cap W_{t_2}$ is a clique there is a vertex $x \in H_{t_2} \setminus W_{t_1}$ that sees both u and v, forming an odd cap, again contradicting Fact 3. Therefore H_{t_2} must be bipartite. We will show that in this case $U_{t_2} \subseteq U_{t_1}$.

First assume there is a vertex $x \in U_{t_2} \setminus U_{t_1}$; once again uv is contained in some even hole in H_{t_1}. We know x sees u and v, so it follows from Lemma 4 that x sees all of every even hole containing uv. In fact, by repeating this argument we can see that x sees all of the block of H_{t_1} containing uv, which is all of H_{t_1} since H_{t_1} is 2-connected. Since $W_{t_1} \cap W_{t_2}$ is a clique cutset, it must contain x, therefore $x \in W_{t_1}$ and everything in U_{t_1} sees x, so x is universal in W_{t_1}, a contradiction since $x \in U_{t_2} \setminus U_{t_1}$. Therefore no such x can exist and $U_{t_2} \subseteq U_{t_1}$. Observe now that $(W_{t_1} \cap W_{t_2}) \setminus U_{t_1} \subseteq H_{t_1} \cap H_{t_2}$; since in this case $\{u, v\} \subseteq H_{t_2}$ we can deduce that H_{t_1} is also bipartite by applying an argument symmetric to the one used to show that H_{t_2} is bipartite. Similarly, we can show that $U_{t_1} \subseteq U_{t_2}$ by applying an argument symmetric to the one used to show that $U_{t_2} \subseteq U_{t_1}$, so $U_{t_1} = U_{t_2}$. But this contradicts the construction of $[T, \mathcal{W}]$ because t_1t_2 was not contracted, therefore $[T, \mathcal{W}]$ satisfies (C4) and the proof is complete. $\square$

4 The Dynamic Programming

Suppose we have a family of graphs $\mathcal{F}$ for which we can solve the maximum-weight stable set problem efficiently. If $\mathcal{G}$ is the family of graphs which can be constructed by pasting together members of $\mathcal{F}$ on cliques, we can efficiently solve the same problem for any graph

$G \in \mathcal{G}$ through a simple application of dynamic programming similar to that outlined in [1]. This method involves constructing, via decomposition on clique cutsets, a subtree decomposition in which $W_t \in \mathcal{F}$ for every node t of the tree, then working up through the tree from the leaves, constructing partial solutions along the way.

Noting that the maximum-weight stable set problem is precisely MWIKS for $k = 1$, we seek to know whether or not this general method extends to larger k. We show in this section that we can solve MWIKS efficiently for complete multipartite graphs and bipartite graphs joined to cliques, for any fixed k. Therefore the NP-completeness of MIBS for clique-separable graphs, which we prove in the next section, illustrates that the method does not extend in general. However, by adding extra restrictions to the kind of subtree decomposition we can get and doing a little more work, we reach a partial extension of the method in the form of a polynomial-time algorithm for MWIKS on i-triangulated graphs.

In the previous section we described how to construct a happy standard subtree decomposition of an i-triangulated graph G in polynomial time – we now wish to make it a rooted tree, so we choose an arbitrary node as the root and denote it r. Call the resulting rooted happy standard subtree decomposition $[(T, r), \mathcal{W}]$. Given a node t of T, let T_t be the subtree rooted at t and let G_t be the graph induced by $\bigcup_{s \in T_t} W_s$. With these preliminaries in hand we can proceed to a description of our dynamic programming method.

For each node t of T with parent u and each feasible set $S \subseteq W_t \cap W_u$, we wish to compute $B_S(G_t)$, which we define to be an arbitrary MWIKS B of G_t subject to the constraint that $B \cap (W_t \cap W_u) = S$. For a node t of T, we define the set $\mathcal{B}_t$ of k-partite induced subgraphs of G_t as follows:

$$\mathcal{B}_t = \{B_S(G_t) \mid S \subseteq W_t \cap W_u, 0 \leq |S| \leq k\}.$$

We can think of $\mathcal{B}_t$ as the part of the dynamic programming table corresponding to the node t since any k-partite induced subgraph of G contains at most k vertices in any clique. With this in mind, we find $\mathcal{B}_t$ for each node t, starting with the leaves and working our way up the tree. Suppose we are given $\mathcal{B}_{s_i}$ for every child s_i of t and a fixed subset S of $W_u \cap W_t$ of size at most k. As the set $\mathcal{B}_t$ contains at most $|W_t|^k + 1 = O(n^k)$ elements, to compute $\mathcal{B}_t$ in polynomial time it is necessary and sufficient to compute each $B_S(G_t)$ in polynomial time. First consider the case in which t is a leaf.

Lemma 7. *Given a leaf t of $[(T, r), \mathcal{W}]$ with parent u and a subset S of G_t such that $|S \cap H_t| \leq 1$, we can find $B_S(G_t)$ in polynomial time.*

Proof. If $|S \cap H_t| = 0$, the problem is equivalent to that of finding a maximum-weight $(k - |S|)$-partite subgraph of $G_t \setminus S$, which is also a base graph. If $G_t \setminus S$ is a complete multipartite graph (note that this includes the case where $G_t \setminus S$ is a clique), then we take the heaviest $k - |S|$ parts of $G_t \setminus S$. If $G_t \setminus S$ is a 2-connected bipartite subgraph H joined to a clique U, then for $0 \leq l \leq 2$, we consider a maximum-weight l-partite subgraph of H along with the $k - |S| - l$ heaviest vertices in U, then take the heaviest graph over our choices of l. This can be done in polynomial time for $l = 1$ [7]; the cases $l = 0$ and $l = 2$ are trivial.

If $S \cap H_t$ is a vertex v, we can solve the problem easily when $G_t \setminus (S - v)$ is a complete multipartite graph by taking the $k - |S|$ heaviest parts that do not intersect with S, along with all parts that do intersect with S. If $G_t \setminus (S - v)$ is a 2-connected bipartite graph H joined to a clique U, we can solve the problem as before by finding a maximum-weight l-partite subgraph of H subject to the constraint that the subgraph contains v. Clearly this makes the case $l = 0$ irrelevant, and the case $l = 2$ is still trivial (take all of H). When $l = 1$ we are looking for a maximum-weight stable set of H containing v, which is a maximum-weight stable set of $H - v - N(v)$ together with v. □

We now consider the case where t is not a leaf. We will reduce the problem to solving MWIKS on a base graph by reweighting the vertices.

Lemma 8. *Given an internal node t of $[(T, r), \mathcal{W}]$ with parent u and a subset S of G_t such that $|S \cap H_t| \leq 1$, we can find $B_S(G_t)$ in polynomial time.*

Proof. Given S, let $S' \supseteq S$ be a given subset of $S \cup U_t$ of size at most k whose intersection with $W_t \cap W_u$ is exactly S. Note that S' induces a clique in G_t. We wish to find $B_{S'}(G_t)$; clearly $w(B_S(G_t))$ is the maximum over all such S' of $w(B_{S'}(G_t))$. To this end, for any vertex v in $H_t \setminus S'$ we define the weight of v with respect to S', written $w_{S'}(v)$, as

$$w_{S'}(v) = w(v) + \sum_{W_{s_i} \ni v} \left(w(B_{(S' \cap W_{s_i}) \cup \{v\}}(G_{s_i}) - v) - w(B_{S' \cap W_{s_i}}(G_{s_i})) \right).$$

Thus the weight of v with respect to S' represents the weight advantage we gain in $\bigcup_i G_{s_i}$ by putting v in an IKS of G_t whose intersection with $S \cup U_t$ is exactly S'. Since we want to maximize this advantage given S', intuition suggests that $B_{S'}(G_t) \cap W_t$ will be a MWIKS of W_t, considering weights with respect to S' instead of the standard weights. We will show that this is indeed the case.

To see this, consider an IKS B of G_t whose intersection with $H_t \setminus S'$ is X and which has maximum weight subject to this constraint. The weight of B is exactly

$$
\begin{aligned}
w(S') \;&+\; \sum_{v\in X}\left(w(v)+\sum_{W_{s_i}\ni v} w\left(B_{(S'\cap W_{s_i})\cup\{v\}}(G_{s_i})-((S'\cap W_{s_i})\cup\{v\})\right)\right)\\
&+\sum_{i \text{ s.t. } X\cap W_{s_i}=\emptyset} w(B_{S'\cap W_{s_i}}(G_{s_i})-(S'\cap W_{s_i})).
\end{aligned}
$$

Now consider a vertex $u \in W_t \setminus (U_t \cup S' \cup X)$ such that $X' = X \cup \{u\}$ is k-partite, along with an IKS B' of G_t whose intersection with $W_t \setminus (U_t \cup S')$ is X' and which has maximum weight subject to this constraint. The weight of B' is

$$
\begin{aligned}
& w(S')+\sum_{v\in X'}\left(w(v)+\sum_{W_{s_i}\ni v} w\left(B_{(S'\cap W_{s_i})\cup\{v\}}(G_{s_i})-((S'\cap W_{s_i})\cup\{v\})\right)\right)\\
&\qquad +\sum_{i \text{ s.t. } X'\cap W_{s_i}=\emptyset} w(B_{S'\cap W_{s_i}}(G_{s_i})-(S'\cap W_{s_i}))\\
=\;& w(B)+w(u)+\sum_{W_{s_i}\ni u}\left(w(B_{(S'\cap W_{s_i})\cup\{u\}}(G_{s_i})-u)-w(B_{S'\cap W_{s_i}}(G_{s_i}))\right)\\
=\;& w(B)+w_{S'}(u).
\end{aligned}
$$

Define $G_{t,S'}$, as the subgraph of G induced by W_t with each vertex $v \in H_t \setminus S'$ given weight $w_{S'}(v)$ and each vertex $v \in S' \cup U_t$ given weight $w(v)$. It is now clear that we can find $B_{S'}(G_t)$ by fixing its intersection with W_t to be a MWIKS of $G_{t,S'}$ whose intersection with $S \cup U_t$ is S'. It is easy to construct $G_{t,S'}$ in polynomial time, and once we have done so we can find a MWIKS of it in polynomial time as per Lemma 7. After that, we need only consult $\mathcal{B}_{s_i}$ for each child s_i of t to construct $B_{S'}(G_t)$. Since $|S'| \leq k$ there are $O(n^k)$ choices of S', so we can construct $B_S(G_t)$ in polynomial time. □

We are now equipped to prove Theorem 1.

Proof of Theorem 1. By the previous two lemmas, we can efficiently construct the set $\mathcal{B}_t$ for any node $t \neq r$ provided that we have already constructed $\mathcal{B}_{s_i}$ for each child s_i of t. By making a postorder traversal of T we can do this for *all* $t \neq r$ in polynomial time as we go: since $[(T, r), \mathcal{W}]$ is a standard subtree decomposition, it follows that T contains at most n nodes. Once this is done we can construct $\mathcal{B}_r$ by making a dummy parent r' of r such that $W_{r'} = \emptyset$. The maximum-weight (in fact, the only) element of $\mathcal{B}_r$ is a MWIKS of G. □

5 Clique-separable graphs

In this section we prove Theorem 2 via a polynomial-time reduction to the maximum stable set problem – recall that Theorem 2 states that the problem of finding a maximum-size induced bipartite subgraph (MIBS) of a clique-separable graph is NP-complete. Let us formally state the two decision problems that we consider.

Max-CS-MIBS
INSTANCE: A clique-separable graph G and an integer k.
QUESTION: Does G have an induced bipartite subgraph containing at least k vertices?

Max-SS
INSTANCE: A simple graph G and an integer k.
QUESTION: Does G have a stable set of size at least k?

The problem Max-SS is well-known to be NP-complete [3], so we need only show that given a graph G and an integer k, we can construct, in polynomial time, a clique-separable graph H' with an associated integer k' such that H' has an induced bipartite subgraph on at least k' vertices if and only if G has a stable set of size at least k. Forthwith the details.

Proof of Theorem 2. Consider a graph G. Create an auxiliary bipartite graph H whose vertex set is $\{v_A, v_B | v \in V(G)\}$ and which has edge $v_A v_B$ corresponding to each $v \in V(G)$, and two edges $u_A v_B$ and $v_A u_B$ corresponding to each edge uv of G.

Extend H to a graph H' by

1. adding for each edge e of H corresponding to an edge of G, a vertex x_e adjacent to both endpoints of e,
2. adding for each vertex v of G, three vertices x_v, y_v, and z_v and the edges $x_v y_v$, $y_v z_v$, $x_v v_A$, $z_v v_B$, $y_v v_A$, and $y_v v_B$.

Note that H' is clique-separable, as it arises from the bipartite graph H by repeatedly pasting triangles onto edges.

We claim that the largest induced bipartite subgraph in H' has size $3|V(G)|+2|E(G)|+\alpha(G)$, where $\alpha(G)$ is the size of the largest stable set in G. It follows that letting $k' = 3|V(G)|+2|E(G)|+k$, H' has an induced bipartite subgraph on at least k' vertices if and only if G has a stable set of size at least k.

To prove our claim, we choose a largest induced bipartite subgraph F of H' that minimizes the number of vertices of F in H.

Suppose that $x_e \notin V(F)$ for some edge $e \in E(H)$ with endpoints a and b. Since F is maximum, both a and b must be in $V(F)$. But $(V(F) \setminus a) \cup x_e$ induces a bipartite subgraph in H' since in this subgraph x_e must have degree one. This contradicts our choice of F, so $x_e \in V(F)$ for every edge $e \in E(H)$. Furthermore, at most one endpoint of e is in $V(F)$, since the two endpoints form induce a triangle with x_e.

Consider a vertex v of G such that v_A is in $V(F)$. At most one of x_v, y_v can be in $V(F)$. Also, $V(F) \setminus v_A \setminus v_B + x_v + y_v$ is bipartite as x_v, y_v, and z_v induce a bipartite component of $H' \setminus v_A \setminus v_B$. Since F is maximum, we can conclude that v_B is in $V(F)$. By a symmetric argument, we see that either both or neither of v_A and v_B are in $V(F)$ for every $v \in V(G)$.

By the above remarks, $V(F) \cap V(H)$ is $\{v_A, v_B | v \in S\}$ for some set $S \subset V(G)$, and S must be stable. It follows that $V(F)$ is a subset of

$$
\begin{aligned}
& \{x_v, z_v, v_A, v_B | v \in S\} \\
\cup \;& \{x_v, y_v, z_v | v \notin S\} \\
\cup \;& \{x_e | e \in E(H), \nexists v \in V(G) \text{ s.t. } e = v_A v_B\}.
\end{aligned}
$$

Hence $|V(F)| \leq 4|S| + 3|V(G) \setminus S| + 2|E(G)| \leq \alpha + 3|V(G)| + 2|E(G)|$.

On the other hand, for any stable set S of G, letting Z be

$$
\begin{aligned}
& \{x_v, z_v, v_A, v_B | v \in S\} \\
\cup \;& \{x_v, y_v, z_v | v \notin S\} \\
\cup \;& \{x_e | e \in E(H), \nexists v \in V(G) \text{ s.t. } e = v_A v_B\},
\end{aligned}
$$

we have that Z induces a bipartite subgraph in H'. Indeed it is a forest, all of whose vertices are leaves except for y_v for $v \in V(G) \setminus S$ and v_A, v_B for $v \in S$. Letting S be a maximum stable set of G, we reach an induced bipartite subgraph of H' containing $3|V(G)| + 2|E(G)| + \alpha(G)$ vertices. This completes the proof of our claim. □

References

[1] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11(1–2):1–22, 1993.

[2] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas. The strong perfect graph theorem. *Annals of Mathematics*. To appear.

[3] M. R. Garey and D. S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[4] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16:47–56, 1974.

[5] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999.

[6] A. Panconesi and D. Ranjan. Quantifiers and approximation. *Theoretical Computer Science*, 107:145–163, 1993.

[7] Sue Whitesides. A method for solving certain graph recognition and optimization problems, with applications to perfect graphs. *Annals of Discrete Math*, 21, 1984.