Turk. J. Math. Comput. Sci.
13(1)(2021) 182–191

DOI : 10.47000/tjmcs.748563

A Maximum Cardinality Cut Algorithm for Co-bipartite and Split Graphs Using Bimodular Decomposition

Arman Boyaci[1], Mordechai Shalom[2,*]

[1]*Department of Industrial Engineering, Faculty of Engineering, Boğaziçi University, 34440 Rumelihisari, Istanbul, Turkey.*
[2]*Department of Computer Science, Faculty of Science, Tel-Hai Academic College, 12210, Upper Gallilee, Israel.*

Received: 05-06-2020 • Accepted: 10-06-2021

Abstract. The maximum cardinality cut (MaxCut) problem remains NP-complete for co-bipartite graphs and for split graphs. Based on modular decomposition, in [3] it is shown that MaxCut is polynomial-time solvable for graphs that are factorable to bounded treewidth graphs. However, this approach fails in co-bipartite graphs because the modules of a co-bipartite graph are exactly twins and connected components. In this work we extend the result of [3] to co-bipartite graphs using the concept of bimodules and bimodular decomposition proposed in [6]. Namely, we show that MaxCut is polynomial-time solvable for co-bipartite graphs that can be factored to bounded treewidth graphs using bimodular decomposition.

2010 AMS Classification: 05C85

Keywords: Maximum cardinality cut, bimodular decomposition, split graphs.

1. Introduction

A *cut* of a graph $G = (V(G), E(G))$ is a partition of $V(G)$ into two subsets $S, \bar{S}$ where $\bar{S} = V \setminus S$. The *cut-set* of $(S, \bar{S})$ is the set of edges of G with exactly one endpoint in S. The maximum cardinality cut (MaxCut) problem is to find a cut of a given graph whose cut-set has the biggest number of edges possible. MaxCut has applications in statistical physics and circuit layout design [1].

MaxCut is a widely studied problem, it is in fact one of the 21 NP-hard problems of Karp [11]. It is shown that MaxCut remains NP-hard when restricted to the following graph classes: chordal graphs, undirected path graphs, split graphs, tripartite graphs, co-bipartite graphs [3], unit disk graphs [4] and total graphs [8]. On the positive side, it was shown that MaxCut can be solved in polynomial time in planar graphs [9], in line graphs [8] and in co-bipartite chain graphs [5].

Modular decomposition is the process of recursively partitioning a graph into its so called modules. Many optimization problems can be solved using dynamic programming, when the modular decomposition of the input graph satisfies specific conditions. Based on modular decomposition, in [3] it is shown that MaxCut is polynomial-time solvable in the class of factorable to bounded treewidth graphs. However, this approach fails in co-bipartite graphs because the modules of a co-bipartite graph (also a split graph) are exactly its twins and its connected components. The concept of

*Corresponding Author
Email addresses: armanboyaci@gmail.com (A. Boyacı), cmshalom@gmail.com (M. Shalom)

bimodules and bimodular decomposition is proposed in [6]. In this work we extend the result of [3] using the bimodular decomposition presented in [6]. This result also generalizes our recent result in [5]. The technique is applicable to other optimization problems.

In Section 2 we start with definitions and notation and summarize related work. In Section 3 we begin with a basic case that we term bi-cographs, which is analogous to cographs in bimodular decomposition, and introduce the technique. In Section 4, we generalize the technique to graphs factorable (using bimodular decompositions) to bounded treewidth graphs. In Section 5 we mention work in progress and future research directions.

2. Preliminaries

General Notation: For two integers n, m such that $n \leq m$, we denote by $[n, m]$ the set of integers between n and m, inclusive and $[n] \stackrel{def}{=} [1, n]$. For a set X and a singleton $\{x\}$ we denote $X \cup \{x\}$ and $X \setminus \{x\}$ by $X + x$ and $X - x$, respectively. We use $\triangle$ as the symmetric difference operator, i.e. $X \triangle Y \stackrel{def}{=} (X \setminus Y) \cup (Y \setminus X)$. Bold characters denote vectors. We denote by $\mathbf{0}$ and by $\mathbf{1}$ the vectors with all entries 0 and 1 respectively. The i-th component of a vector $\mathbf{v}$ is v_i, i.e. $\mathbf{v} = (v_1, \ldots, v_k)$, and $\sum \mathbf{v} \stackrel{def}{=} \sum_{i=1}^{k} v_i$. For a vector $\mathbf{v}$ with index set I and $I' \subset I$, we denote by $\mathbf{v}_{I'}$ the projection of $\mathbf{v}$ on I'.

Graph Notation and Terminology: Given a simple graph (no loops or parallel edges) $G = (V(G), E(G))$ and a vertex v of G, uv denotes an edge between two vertices u, v of G. We denote by $N_G(v)$ the set of neighbors (or the *open neighborhood*) of v in G, and by $d_G(v) = |N_G(v)|$ the degree of v in G. Whenever there is no ambiguity we omit the subscript G and write $d(v)$ and $N(v)$. Two adjacent (resp. non-adjacent) vertices u, v of G are *twins* (resp. *false twins*) if $N_G(u) \setminus \{v\} = N_G(v) \setminus \{u\}$. A vertex having degree zero is termed *isolated*, and a vertex adjacent to all other vertices is termed *universal*. A *clique* (resp. *independent set*) of a graph is a set of pairwise adjacent (resp. non-adjacent) vertices. A clique of k vertices is denoted by K_k. For a graph G and $U \subseteq V(G)$, we denote by $G[U]$ the subgraph of G induced by U, and $G \setminus U \stackrel{def}{=} G[V(G) \setminus U]$. The complement $\overline{G}$ of a graph $G = (V, E)$ is the graph $(V, V \times V \setminus E)$. The union of two graphs G, G' is $G \cup G' \stackrel{def}{=} (V(G) \cup V(G'), E(G) \cup E(G'))$. For two vertex disjoint graphs G_1, G_2, $G_1 \times G_2$ is the graph $G_1 \cup G_2$ where all the possible edges between the vertices of G_1 and G_2 are added. Formally, $G_1 \times G_2 \stackrel{def}{=} (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2) \cup V(G_1) \times V(G_2))$.

A graph is *bipartite* (resp. *co-bipartite*, *split*) if its vertex set can be partitioned into two independent sets (resp. two cliques , resp. a clique and an independent set) V, V'. We denote such a graph as $B(V, V', E)$ (resp. $C(V, V', E)$, $S(V, V', E)$) where E is the set of edges that have exactly one endpoint in V. The *bipartite complement* $\bar{G}^{bip}$ of a bipartite graph $G = B(V, V', E)$ is the bipartite graph $B(V, V', (V \times V') \setminus E)$.

Cuts: A *cut* of a graph is a partition of $V(G)$ into two sets S and $\bar{S} = V(G) \setminus S$. We denote a cut by one of the subsets of the partition. $E(S, \bar{S})$ denotes the *cut-set* of S, i.e. the set of the edges of G with exactly one endpoint in S, and $\text{cs}(S) \stackrel{def}{=} \left|E(S, \bar{S})\right|$ is termed the *cut size* of S. A *maximum cut* of G is one having the biggest cut size among all cuts of G. We refer to this size as the *maximum cut size* of G.

Definition 2.1. Given a cut S of a graph G and two disjoint subsets X, Y of vertices of G, we denote by $E(S, X, Y)$ the set of edges in the cut-set of S that have one endpoint in X and the other in Y, i.e. $E(S, X, Y) \stackrel{def}{=} E(S, \bar{S}) \cap X \times Y$. We observe that whenever $X \times Y \subseteq E(G)$ the value $E(S, X, Y)$ depends only on $|S \cap X|$ and $|S \cap Y|$. Therefore, in this case we define $\text{cs}(X, Y, |S \cap X|, |S \cup Y|) \stackrel{def}{=} |E(S, X, Y)|$.

Clearly,

$$\text{cs}(X, Y, i, j) = i(|Y| - j) + j(|X| - i).$$

Definition 2.2. For a graph G, we denote by $maxc_G(i)$ the biggest cut size among the cuts that partition $V(G)$ into a set of size i and a set of size $|V(G)| - i$. Formally,

$$maxc_G(i) \stackrel{def}{=} \max \{\text{cs}(S) : S \subseteq V(G), |S| = i\}.$$

Clearly, the maximum cut size of G is

$$\max \{maxc_G(i) : 0 \leq i \leq \lfloor |V(G)| / 2 \rfloor\}.$$

Definition 2.3. For a co-bipartite (resp. split) graph $G = C(V, V', E)$ (resp. $S(V, V', E)$), we denote as G_B the corresponding bipartite graph $B(V, V', E)$. We also denote by $maxc_{G_B}(i, i')$ the maximum size of a cut of G_B that partitions V (resp. V') into a set of size i (resp. i') and a set of size $|V| - i$ (resp. $|V'| - i'$). Formally,

$$maxc_{G_B}(i, i') \stackrel{def}{=} \max\left\{\text{cs}(S) : S \subseteq V \cup V', |S \cap V| = i, \left|S \cap V'\right| = i'\right\}.$$

Clearly, the maximum cut size of G is

$$\max\left\{maxc_{G_B}(i, i') + i \cdot (|V| - i) + i' \cdot (\left|V'\right| - i') : \mathbf{0} \leq (i, i') \leq (|V|, \left|V'\right|)\right\},$$

if G is co-bipartite, and

$$\max\left\{maxc_{G_B}(i, i') + i \cdot (|V| - i) : \mathbf{0} \leq (i, i') \leq (|V|, \left|V'\right|)\right\},$$

if G is a split graph. Moreover, it can be computed in polynomial time once the values $maxc_{G_B}(i, i')$ are given. In this work we compute these values, and therefore focus on bipartite graphs.

Tree Decomposition and Treewidth: Given a graph G, a tree decomposition of G is a tree $\mathcal{T} = (\mathcal{B}, F)$, where $\mathcal{B} = \{B_1, B_2, ...\}$ is a set of subsets of $V(G)$ termed *bags*, such that the following three conditions are met:

- $\bigcup \mathcal{B} = V(G)$.
- For every edge $uv \in E(G)$, $u, v \in B_i$ for some $B_i \in \mathcal{B}$.
- If $v \in B_i \cap B_j$ then $v \in B_k$ for every bag B_k on the path between B_i and B_j in $\mathcal{T}$.

The *width* $\omega(\mathcal{T})$ of a tree decomposition $\mathcal{T} = (\mathcal{B}, F)$ is defined as the size of its largest bag minus one, i.e., $\omega(\mathcal{T}) = \max\{|B| : B \in \mathcal{B}\} - 1$. The treewidth of a graph G, denoted as $tw(G)$, is the width of the minimum-width tree decomposition of G. For an integer $k > 0$, we denote by $\mathcal{T}(k)$ the class of graphs having treewidth at most k. For any fixed $k > 0$ there exists an algorithm that finds a tree decomposition of minimum width of a given graph in $\mathcal{T}(k)$ (see [2]). Many graph problems that are NP-hard are polynomial-time solvable when restricted to $\mathcal{T}(k)$ for some fixed k.

It is easy to show that every bag B_i corresponding to an internal vertex of $\mathcal{T}$ is a separator of G.

Definition 2.4. A tree decomposition $\mathcal{T}$ is *canonical* if

- $\mathcal{T}$ is a binary tree.
- If a bag B_i has two children B_j, B_k (in $\mathcal{T}$) then $B_i = B_j = B_k$.
- If a bag B_i has exactly one child B_j then $\left|B_i \triangle B_j\right| = 1$.

It is well known that every tree decomposition can be transformed to a canonical one with the same width (see for example [3]). In this work we consider only canonical tree decompositions $\mathcal{T} = (\mathcal{B}, F)$ rooted at an arbitrary vertex. For a bag $B \in \mathcal{B}$ we denote by B^* the set of vertices in B and all the descendants of B in $\mathcal{T}$.

Cographs: The class of *cographs* (shorthand for *complement-reducible* graphs) is defined recursively as follows: a) a graph with one vertex is a cograph, and b) if G_1 and G_2 are cographs then so are $G_1 \cup G_2$ and $G_1 \times G_2$. Thus a cograph is a graph that can be generated from single-vertex graphs K_1 by repeatedly applying the binary operations $\cup$ and $\times$. A *cotree* is a binary tree whose leaves represent single vertices and the internal (non-leaf) vertices are labeled with $\cap$ or $\times$. Every subtree of a cotree represents a cograph.

If G is a cograph with at least two vertices then its root is labeled either with $\cup$ in which case G is not connected, or with $\times$ in which case $\overline{G}$ is not connected. This observation implies a polynomial-time algorithm to construct a cotree of a given cograph. Many graph problems (maximum independent set, maximum clique, minimum vertex coloring) can be solved in linear time in cographs by constructing its cotree and performing a bottom-up traversal of it.

MaxCut in Cographs: The maximum cut size of a cograph G can be computed using dynamic programming via a bottom up traversal of its cotree and using Lemma 2.5 at each internal vertex of the cotree.

Lemma 2.5. *[3] Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be vertex disjoint graphs. Then*

$$\begin{aligned} maxc_{G_1 \cup G_2}(i) = \max\{&maxc_{G_1}(j) + maxc_{G_2}(i - j) : \\ &j \in [0, i], j \leq |V_1|, i - j \leq |V_2|\}, \\ maxc_{G_1 \times G_2}(i) = \max\{&maxc_{G_1}(j) + maxc_{G_2}(i - j) + \\ &j(|V_2| - (i - j)) + |V_1 - j|(i - j) : \\ &j \in [0, i], j \leq |V_1|, i - j \leq |V_2|\}. \end{aligned}$$

Modular decomposition: A *module* of a graph $G = (V, E)$ is a subset $M \subseteq V$ of its vertices such that any vertex in $V \setminus M$ is either adjacent to all vertices of M or to none of them. In other words, the vertices of M are indistinguishable by vertices not in M. It follows from this definition that M is a module of G if and only if it is a module of $\overline{G}$. Clearly, $V, \emptyset$ and all the singleton subsets of V are modules of G. These modules are termed as *trivial*. A graph containing only trivial modules is termed *prime*. A module M is *maximal* if no proper superset of M except V is a module. Gallai [7] showed that if both G and its complement are connected, then the maximal modules of G do not intersect, i.e. they form a partition of $V(G)$. In this case G can be decomposed to its maximal modules unless G is prime. One can recursively decompose a graph to its connected components, or into the connected components of its complement, or its maximal modules until prime graphs are obtained. This defines the *modular decomposition* tree of a given graph. Such a decomposition can be found in linear time [12].

In the sequel we use the terminology and notations from [3]. Let $H = (V, E)$ be a graph with $r > 1$ vertices $\{v_1, \dots v_r\}$ and $H_1, H_2, \dots H_r$ be r vertex disjoint graphs. The *factor graph* $H[H_1, H_2, \dots H_r]$ is the graph obtained by taking the disjoint union of $H_1, H_2, \dots H_r$ and adding all edges $vw \in V_i \times V_j$ whenever $ij \in E$. Every graph G is a factor graph since $G = G[G_1, G_2, \dots, G_n]$ where G_i is the graph consisting of the single vertex $v_i \in V(G)$. A factorization $H[H_1, \dots, H_r]$ of G such that r is minimum is equivalent to finding the maximal modules of G. The *factor tree* of a graph G is the decomposition tree of G where each (internal) vertex v of the tree with r children, corresponding to the factorization operation $H[H_1, \dots, H_r]$ is labeled by the graph H, termed the *label graph* of v. Note that a cotree is a special case of a factor tree in which the label graphs are either K_2 or $\overline{K_2}$.

Definition 2.6. For a class C, $\mathcal{F}(C)$ denotes the class of graphs with factor tree labeled with graphs from C. The factor-treewidth $fw(G)$ of a graph G is the smallest number k such that $G \in \mathcal{F}(\mathcal{T}(k))$.

For instance, $\mathcal{F}(\{K_2, \overline{K_2}\})$ is the class of cographs and the factor-treewidth of a cograph is 1. We note that a factor tree can be computed in linear time from a modular decomposition tree as follows. For an internal vertex v of the tree with r children the label graph of v contains r vertices, and the adjacency relation between two vertices can be determined by inspecting the subgraphs corresponding to the leaves and checking the adjacency between two arbitrary vertices, one from each graph.

MaxCut and Modular Decomposition: Lemma 2.5 can be extended to any factorization in the following way. Let $G = H[H_1, \dots, H_r]$, let S be a cut of G, let $S_i \stackrel{def}{=} S \cap V(H_i)$ for $i \in [r]$ and let $\mathbf{v}$, $\mathbf{s}$ be the vectors with r entries $v_i = |V(H_i)|$, $s_i = |S_i|$. By considering the inter-module and intra-module edges of G separately and recalling that the inter-module edges exist whenever the corresponding vertices are adjacent in H, we have

$$\begin{aligned} \mathrm{cs}(S) &= \sum_{j=1}^{r} \mathrm{cs}(S_j) + \sum_{\ell m \in E(H)} |S_\ell| \cdot \left|\bar{S}_m\right| + |S_m| \cdot \left|\bar{S}_\ell\right| \\ &= \sum_{j=1}^{r} \mathrm{cs}(S_j) + \sum_{\ell m \in E(H)} s_\ell(v_m - s_m) + s_m(v_\ell - s_\ell) \end{aligned}$$

and observing that $|S| = \sum \mathbf{s}$ we conclude

$$maxc_G(i) = \max \left\{ \sum_{j=1}^{r} maxc_{H_j}(x_j) + \sum_{\ell m \in E(H)} s_\ell(v_m - s_m) + s_m(v_\ell - s_\ell) : \sum \mathbf{s} = i, \mathbf{0} \le \mathbf{s} \le \mathbf{v} \right\}.$$

Whenever the order of H is bounded by some constant, $maxc_G(i)$ can be computed in polynomial time since in this case, the number of vectors $\mathbf{s} \le \mathbf{v}$ is polynomial in the order of H. This implies a polynomial-time algorithm for the class of graphs $\mathcal{F}(\mathcal{G}_k)$ for any constant $k > 0$ where $\mathcal{G}_k$ is the class of graphs of order at most k. Exploiting the facts that a) given a tree decomposition of H the ℓm edges of H are within the bags of H and b) a separator of H induces a separator of G, the idea is extended to the following theorem:

Theorem 2.7. *[3] For every constant k,* MaxCut *is solvable in polynomial time for graphs in* $\mathcal{F}(\mathcal{T}(k))$.

Bimodular decomposition: Bipartite graphs do not fit well within the framework of modular decomposition in the following sense: the only non-trivial modules of a bipartite graph $B(V, V', E)$ are its connected components and sets

consisting of twins. In other words, a connected twin-free bipartite graph contains only trivial modules. The concept of bimodule is defined in [6] via which non-trivial decompositions can be obtained and calculated in polynomial time. In the sequel we give a brief description of this decomposition.

Let $G = B(V, V', E)$ be a bipartite graph. A *bimodule* of G is a subset M of its vertices such that any vertex in $V \setminus M$ (resp. $V' \setminus M$) is either adjacent to all vertices of $M \cap V'$ (resp. $M \cap V$) or to none of them. In other words, the vertices of $M \cap V$ (resp. $M \cap V'$) are indistinguishable by vertices of $V' \setminus M$ (resp. $V \setminus M$). $V \cup V'$, V, V', $\emptyset$, every singleton, and every pair of vertices from $V \times V'$ are *trivial bimodules* of G.

Unlike maximal modules, maximal bimodules may overlap. Nevertheless some bimodules termed *canonical strong bimodules* have the desired property, based on which the *bimodular decomposition* tree $T(G)$ of a bipartite graph G can be computed in polynomial time.

Let H be the graph represented by an internal vertex v of a decomposition tree $T(G)$. The vertex v is of one of the following four types:

- *parallel*: the children of v represent the connected components of H.
- *series*: the children of v represent the connected components of $\bar{H}^{bip}$.
- $K + S$: the children of v represent induced subgraphs $H_1, H_2, \ldots, H_r$ such that for every pair $i, j \in [r]$ with $i < j$ a) there is an edge between every vertex of $V(H_i) \cap V$ and every vertex of $V(H_j) \cap V'$, b) there is no edge between a vertex of $V(H_i) \cap V'$ and a vertex of $V(H_j) \cap V$.
- *prime*: the children of v represent canonical strong bimodules of H.

We note that we can modify the tree (at the expense of increasing the number of its vertices) so that in the first three cases v has two children. The tree obtained in this way is unique (up to the order of the children of vertices). We now note that given two bimodules M_1, M_2 of G, every vertex of $M_1 \cap V$ (resp. $M_1 \cap V'$) is either adjacent to every vertex of $M_2 \cap V'$ (resp. $M_2 \cap V$) or to none of them. Therefore, we can capture the relations between M_1 and M_2 by the existence or absence of two potential edges. Formally,

Definition 2.8. Given the bipartite graphs $H_1 = B(V_1, V_1', E_1), \ldots, H_r = B(V_r, V_r', E_r)$ and two graphs H, H' with $V(H) = V(H') = [r]$ the *bi-factor graph* $(H, H')[H_1, \ldots, H_r]$ is the bipartite graph $B(V, V', E)$ such that $V = \bigcup_{i=1}^{r} V_i$, $V' = \bigcup_{i=1}^{r} V_i'$ and

$$E = \bigcup_{i=1}^{r} E_i \ \cup \bigcup_{\ell m \in E(H), \ell < m} V_\ell \times V_m' \ \cup \bigcup_{\ell m \in E(H'), \ell > m} V_\ell' \times V_m.$$

We define the *bifactor tree* of G in a way similar to the factor tree, with the following differences: a) each internal vertex is labeled by an ordered pair of graphs (H, H'), b) the tree is ordered, i.e. the children of a vertex are numbered.

Definition 2.9. Given a class $\mathcal{C}$ of graphs, $\mathcal{BF}(\mathcal{C})$ denotes the class of bipartite graphs G such that for every vertex v of $T(G)$ the union of the two label graphs of v is a graph in $\mathcal{C}$. The bifactor treewidth $bfw(G)$ of a bipartite G is the smallest number k such that $G \in \mathcal{BF}(\mathcal{T}(k))$.

Max Cut in Co-bipartite Chain graphs: A *co-bipartite chain graph* is a co-bipartite graph $G = (V, V', E)$ where V has a nested neighborhood ordering, i.e. its vertices can be ordered as $v_1, v_2, \ldots$ such that $N_G(v_1) \subseteq N_G(v_2) \subseteq \cdots$ [10].

In [5] we showed that the twin-free co-bipartite chain graphs do not have a factorization of bounded treewidth, implying that Theorem 2.7 is not applicable to this graph class.

Theorem 2.10. *[5] For every $k > 0$ there exists a twin-free co-bipartite graph G such that $G \notin \mathcal{F}(\mathcal{T}(m))$.*

On the other hand, using a dynamic programming algorithm similar to the ones presented in [3] we showed the following:

Theorem 2.11. *[5]* MaxCut *can be solved in polynomial time for co-bipartite chain graphs.*

Lemma 2.12. *Let G be a twin-free chain co-bipartite graph. Then $G \in \mathcal{BF}(\mathcal{T}(1))$.*

Proof. $\bar{G} = B(V, V', E)$ is a bipartite chain graph which is twin-free, except possibly two isolated vertices. The vertices of V and V' can be numbered as $v_1, v_2, \ldots, v_n$ and $v_1', v_2', \ldots, v_n'$ respectively, according to the nested neighborhood ordering. Then $V_1 = \left\{v_1, v_1'\right\}, \ldots, V_n = \{v_n, v_n'\}$ is a $K + S$ decomposition of $\bar{G}$. Therefore, the bifactor tree of $\bar{G}$ (thus of G) is a binary tree with labels $(K_2, \bar{K}_2)$. The result follows from the fact that K_2 is a tree. □

In this work we generalize Theorem 2.11 to the class $\mathcal{BF}(\mathcal{T}(k))$ for every constant k.

3. Bi-CoGraphs

In this section we define the analogous of cographs for the bipartite/cobipartite case, as an intermediate step to describe the results in the next section. We recall that the class of cographs is $\mathcal{F}(\{K_2, \bar{K}_2\})$. We define analogously the class of *bi-cographs* as $\mathcal{BF}(\{K_2, \bar{K}_2\})$. We define the following binary graph operations on bipartite graphs that correspond to the label pairs $(K_2, \bar{K}_2)$ and (K_2, K_2) respectively. Let $G_1 = B(V_1, V_1', E_1)$ and $G_2 = B(V_2, V_2', E_2)$. Then

$$\begin{aligned} G_1 /^{bip} G_2 &\stackrel{def}{=} B(V_1 \cup V_2, V_1' \cup V_2', E_1 \cup E_2 \cup V_1 \times V_2'), \\ G_1 \times^{bip} G_2 &\stackrel{def}{=} B(V_1 \cup V_2, V_1' \cup V_2', E_1 \cup E_2 \cup V_1 \times V_2' \cup V_1' \times V_2). \end{aligned}$$

A *bi-cotree* is an ordered binary tree whose leaves represent single vertices and the internal (non-leaf) vertices are labeled with $\cup$, $/^{bip}$ or $\times^{bip}$.

If G is a bi-cograph then $maxc_G(i, i')$ can be computed using dynamic programming via a bottom up traversal of its bi-cotree and using Lemma 3.1 at each internal vertex of the bi-cotree.

Lemma 3.1. *Let $G_1 = B(V_1, V_1', E_1)$ and $G_2 = B(V_2, V_2', E_2)$ be vertex disjoint bipartite graphs. Then*

$$\begin{aligned} maxc_{G_1 \cup G_2}(i, i') &= \max\{maxc_{G_1}(i_1, i_1') + maxc_{G_2}(i_2, i_2') : \ (i_1, i_1', i_2, i_2') \in I\}, \\ maxc_{G_1 /^{bip} G_2}(i, i') &= \max\{maxc_{G_1}(i_1, i_1') + maxc_{G_2}(i_2, i_2') + \\ &\qquad \mathrm{cs}(V_1, V_2', i_1, i_2') : \ (i_1, i_1', i_2, i_2') \in I\}, \\ maxc_{G_1 \times^{bip} G_2}(i, i') &= \max\{maxc_{G_1}(i_1, i_1') + maxc_{G_2}(i_2, i_2') + \\ &\qquad \mathrm{cs}(V_1, V_2', i_1, i_2') + \mathrm{cs}(V_1', V_2, i_1', i_2) : (i_1, i_1', i_2, i_2') \in I\} \end{aligned}$$

where

$$\begin{aligned} I = \{(i_1, i_1', i_2, i_2') : & i_1 + i_2 = i, \mathbf{0} \le (i_1, i_2) \le (|V_1|, |V_2|), \\ & i_1' + i_2' = i', \mathbf{0} \le (i_1', i_2') \le (\left|V_1'\right|, \left|V_2'\right|)\}. \end{aligned} \tag{3.1}$$

Proof. Let $G = B(V, V', E) \in \left\{G_1 \cup G_2, G_1 /^{bip} G_2, G_1 \times^{bip} G_2\right\}$ where $V = V_1 \cup V_2$, $V' = V_1' \cup V_2'$. Let S be a cut of G and let S_k be the cut induced by S on G_k for $k \in \{1, 2\}$. Then, the edges of $E(S, \bar{S})$ are partitioned into three sets, a) the edges of G_1, b) the edges of G_2, and c) the edges that join vertices from both graphs. The edges joining vertices from two graphs are further partitioned into two sets, namely those joining vertices of V_1 with vertices of V_2' and vertices of V_2 with vertices of V_1'. Therefore,

$$\mathrm{cs}(S) = \mathrm{cs}(S_1) + \mathrm{cs}(S_2) + \left|E(S, V_1, V_2')\right| + \left|E(S, V_1', V_2)\right|.$$

We now focus on cuts S such that $|S \cap V| = i$, $|S \cap V'| = i'$. For such a cut S, let $|S \cap V_k| = |S_k \cap V_k| = i_k$ and $\left|S \cap V_k'\right| = \left|S_k \cap V_k'\right| = i_k'$ for $k \in \{1, 2\}$. Clearly, $(i, i') = (i_1 + i_2) + (i_1' + i_2')$. We have

$$\left|E(S, V_1, V_2')\right| = \begin{cases} 0 & \text{if } G = G_1 \cup G_2 \\ \mathrm{cs}(V_1, V_2', i_1, i_2') & \text{otherwise} \end{cases}$$

and

$$\left|E(S, V_1', V_2)\right| = \begin{cases} \mathrm{cs}(V_1', V_2, i_1', i_2) & \text{if } G = G_1 \times^{bip} G_2 \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, $\mathrm{cs}(S) = \mathrm{cs}(S_1) + \mathrm{cs}(S_2) + f(i_1, i_1', i_2, i_2')$ where f is a function that depends on G. Then

$$\begin{aligned} maxc_G(i, i') = \max\{ & \mathrm{cs}(S_1) + \mathrm{cs}(S_2) + f(i_1, i_1', i_2, i_2') : (i_1, i_1', i_2, i_2') \in I, \\ & \forall k \in \{1, 2\} (|S \cap V_k| = i_k \wedge \left|S \cap V_k'\right| = i_k')\} \end{aligned}$$

For fixed values of (i_1, i_1', i_2, i_2') the terms are independent. Then

$$\begin{aligned} maxc_G(i, i') = \max\{ & \max\left\{\mathrm{cs}(S_1) : |S \cap V_1| = i_1, \left|S \cap V_1'\right| = i_1'\right\} + \\ & \max\left\{\mathrm{cs}(S_2) : |S \cap V_2| = i_2, \left|S \cap V_2'\right| = i_2'\right\} + \\ & f(i_1, i_1', i_2, i_2') : (i_1, i_1', i_2, i_2') \in I\} \end{aligned}$$

The lemma follows by the definition of $maxc_G(i, i')$. □

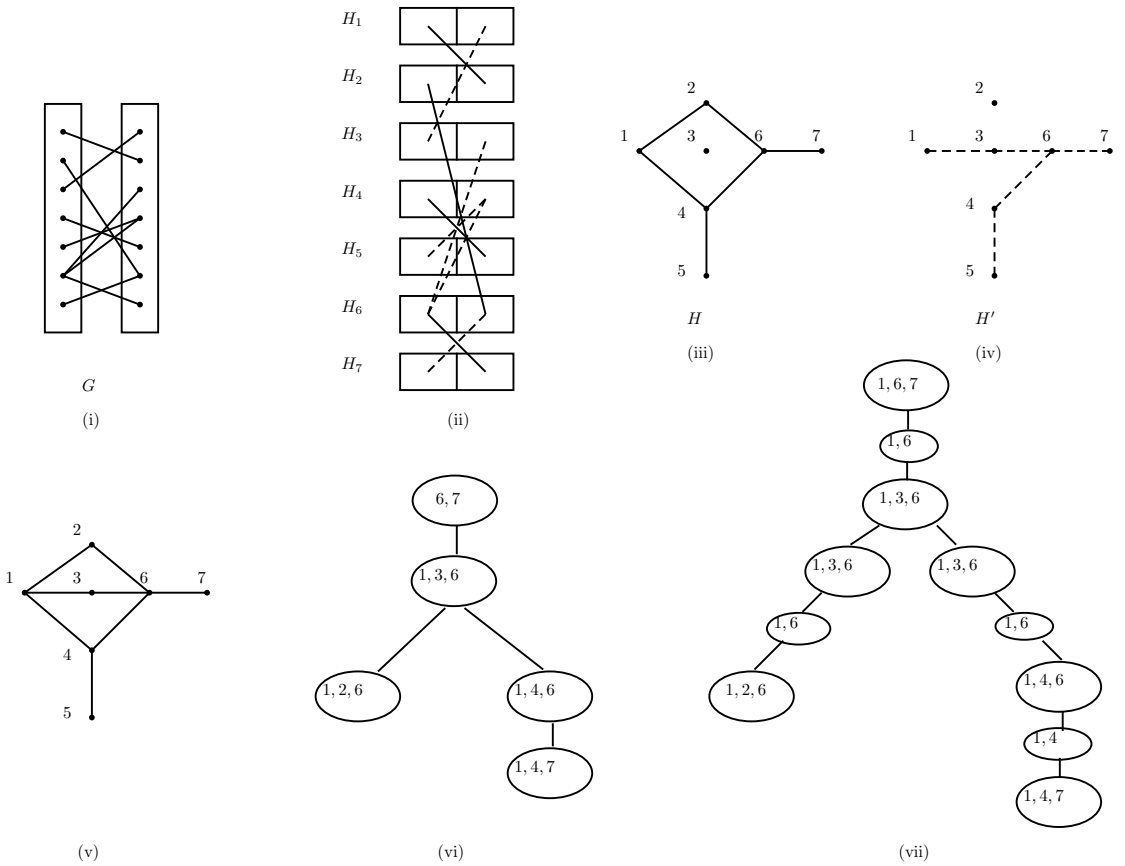

FIGURE 1. (i)A bipartite graph G, (ii) the bimodular decomposition of G into modules $H_1, H_2, H_3, H_4, H_5, H_6, H_7$. $G = (H, H')[H_1, H_2, H_3, H_4, H_5, H_6, H_7]$, (iii, iv) the label graphs H, H', (v) the graph $H \cup H'$, (vi) a tree decomposition of $H \cup H'$, (vii) a canonical tree decomposition of $H \cup H'$ with same width.

Note that $maxc_G(i, i')$ can be computed by trying all the possible values (i_1, i'_1, i_2, i'_2) that are given by (3.1) and choosing the best value.

We conclude this section by noting that the class of co-bipartite chain graphs is included in the class of bi-cographs. All the internal vertices of the bi-cotrees of a co-bipartite chain graph are labeled with $/^{bip}$. Therefore, Lemma 3.1 extends Theorem 2.11 to bi-cographs.

4. Graphs With Bounded Bi-factor Width

We now consider bipartite graphs having a bimodular decomposition with labeled graph pairs such that the union of each pair has bounded treewidth, i.e. the graphs $G \in \mathcal{BF}(\mathcal{T}(k))$ for some constant k. Recall that our goal is to compute the values $maxc_G(i, i')$. For this purpose, we first compute the bi-factor tree $T(G)$ and then make a bottom-up traversal of it. At each vertex v of $T(G)$ we compute the values $maxc_{G_v}(i, i')$ where G_v is the graph defined by the subtree of $T(G)$ rooted at v. Whenever v is a leaf, v represents a trivial bimodule, i.e. consists of at most two vertices, one from V and one from V'. In this case $i, i' \in \{0, 1\}$ and the values $maxc_{G_v}(i, i')$ can be computed trivially. In the sequel we consider the root of $T(G)$ and assume that all the subtrees are traversed recursively. In the rest of this section, $G = (H, H')[H_1, \ldots, H_r]$ where $H_1, \ldots, H_r$ are bipartite graphs with bipartititons $(V_1, V'_1), \ldots, (V_r, V'_r)$, and $\mathbf{v}$ (resp. $\mathbf{v}'$) is a vector with r entries such that $v_i = |V_i|$ (resp. $v'_i = \left|V'_i\right|$).

Definition 4.1. For every pair of vectors $\mathbf{x}, \mathbf{x}'$ such that $\mathbf{0} \leq \mathbf{x} \leq \mathbf{v}$ and $\mathbf{0} \leq \mathbf{x}' \leq \mathbf{v}'$ we denote by $maxc_G(\mathbf{x}, \mathbf{x}')$ the maximum cut size of among all cuts S of G such that $|S \cap V_i| = x_i$ and $\left|S \cap V'_i\right| = x'_i$ for every $i \in [r]$.

Clearly,

$$maxc_G(i,i') = max\left\{max_G(\mathbf{x},\mathbf{x}') : \sum \mathbf{x} = i, \sum \mathbf{x}' = i'\right\}. \tag{4.1}$$

Lemma 4.2.

$$maxc_G(\mathbf{x},\mathbf{x}') = \sum_{j=1}^{r} maxc_{H_j}(x_j,x'_j) + \sum_{\ell m\in E(H),\ell<m} \mathrm{cs}(V_\ell,V'_m,x_\ell,x'_m) + \sum_{\ell m\in E(H'),\ell>m} \mathrm{cs}(V'_\ell,V_m,x'_\ell,x_m). \tag{4.2}$$

Proof. From the definition of the bi-factor graph and using the same arguments as in the proof of Lemma 3.1, for any cut S of G we have

$$\mathrm{cs}(S) = \sum_{j=1}^{r} \mathrm{cs}(S_j) + \sum_{\ell m\in E(H),\ell<m} \left|E(S,V_\ell,V'_m)\right| + \sum_{\ell m\in E(H'),\ell>m} \left|E(S,V'_\ell,V_m)\right|$$

where S_j is the cut induced by S on H_j. Moreover, whenever $|S\cap V_i| = x_i$ and $\left|S\cap V'_i\right| = x'_i$ and $\ell m \in E(H)$ with $\ell < m$ (resp. $\ell m \in E(H')$ with $\ell > m$) we have $\left|E(S,V_\ell,V'_m)\right| = \mathrm{cs}(V_\ell,V'_m,x_\ell,x'_m)$ and $\left|E(S,V'_\ell,V_m)\right| = \mathrm{cs}(V'_\ell,V_m,x'_\ell,x_m)$. Therefore, for all cuts S with these properties, the last two sums are fixed. Observing that the terms of the first summation are independent, the lemma follows. □

Note that Equations (4.1) and (4.2) do not lead to an efficient algorithm since the number of possible vector pairs $(\mathbf{x},\mathbf{x}')$ is exponential. However, in the sequel we use these equations locally in individual bags of a tree decomposition. Since the bags contain at most $k+1$ vertices, this computation can be carried out in polynomial time.

Consider a canonical tree decomposition $\mathcal{T}$ of $H\cup H'$ and a bag B of $\mathcal{T}$. Recall that B^* is the set of vertices in $V(H\cup H') = V(H) = V(H')$ contained in the subtree of $\mathcal{T}$ rooted at B. For a subset $U = \{h_1,h_2\dots h_k\}$ of $V(H)$ we introduce the following shorthand notation for the induced subgraph of G corresponding to U:

$$G_{[U]} = (H[U],H'[U])[H_{h_1},\dots H_{h_k}].$$

Definition 4.3. For a pair of vectors $\mathbf{x},\mathbf{x}'$ such that $\mathbf{0}\le\mathbf{x}\le\mathbf{v}$ and $\mathbf{0}\le\mathbf{x}'\le\mathbf{v}'$ we denote by $maxc_{G_{[B^*]}}(B,\mathbf{x},\mathbf{x}')$ the maximum size of a cut S of $G_{[B^*]}$ such that $\left|V_{h_i}\cap S\right| = x_i$ and $\left|V'_{h_i}\cap S\right| = x'_i$ for every $i\in[k]$.

Since $G = G_{[B^*_r]}$, we have $maxc_G(\mathbf{x},\mathbf{x}') = maxc_{G_{[B^*_r]}}(B_r,\mathbf{x},\mathbf{x}')$. The following lemma describes the computation of these values for a bag B, from the values for its children.

Lemma 4.4.

$$maxc_{G_{[B^*]}}(B,\mathbf{x},\mathbf{x}') =$$

$$maxc_{G_{[B]}}(\mathbf{x},\mathbf{x}') \quad \textit{if } B \textit{ is a leaf},$$

$$maxc_{G_{[B^*_1]}}(B_1,\mathbf{x},\mathbf{x}') + maxc_{G_{[B^*_2]}}(B_2,\mathbf{x},\mathbf{x}') - maxc_{G_{[B]}}(\mathbf{x},\mathbf{x}')$$
$$\textit{if } B \textit{ has two children } B_1,B_2,$$

$$maxc_{G_{[B^*_1]}}(B_1,\mathbf{x}_{B-m},\mathbf{x}'_{B-m}) + maxc_{H_m}(x_m,x'_m) + \sum_{\ell m\in E(H)} \mathrm{cs}(V_\ell,V'_m,x_\ell,x'_m) + \sum_{\ell m\in E(H')} \mathrm{cs}(V'_\ell,V_m,x'_\ell,x_m)$$
$$\textit{if } B \textit{ has one child } B_1 \textit{ such that } B\setminus B_1 = \{m\},$$

$$\max\left\{maxc_{G_{[B^*_1]}}(B_1,\mathbf{y},\mathbf{y}') : \mathbf{y}_{B_1-m} = \mathbf{x}, \mathbf{y}'_{B_1-m} = \mathbf{x}'\right\}$$
$$\textit{if } B \textit{ has one child } B_1 \textit{ such that } B_1\setminus B = \{m\}.$$

Proof. Since $\mathcal{T}$ is a canonical decomposition, B satisfies exactly one of the below conditions. We prove each case separately.

- B is a leaf: In this case $B^* = B$ and the definitions of both sides of the equation coincide.
- B has two children B_1, B_2: In this case $B_1 = B_2 = B$. Then every vertex of B^* is either in exactly one of B_1^* and B_2^* or in both. In the latter case such a vertex is also in B by the third property of a tree decomposition. The same holds for every edge that connects two vertices of B^*. Consider the summand in 4.2 corresponding to a vertex of B^* or to an edge of $E(H)$ (resp. $E(H')$) in B^*. The proposed formula guarantees that every such summand is accounted exactly once.
- B has one child B_1 such that $B \setminus B_1 = \{m\}$: In this case $B^* \setminus B_1^* = \{m\}$. As for the edges, the edges of B^* are exactly the edges of B_1^* with the addition of all the edges incident to m in H or H'. The first term accounts for all the edges and vertices of B_1^*, the second term accounts for all the intra-module edges of H_m and the last two terms account for all the edges incident to m in H and H' respectively.
- B has one child B_1 such that $B_1 \setminus B = \{m\}$: In this case $B^* = B_1^*$. However, since the entry corresponding to m is unspecified in $\mathbf{x}$ and $\mathbf{x}'$ one has to consider each possible extension $\mathbf{y}$ (resp. $\mathbf{x}$) of the vector $\mathbf{x}$ (resp. $\mathbf{x}'$) with this entry and return the maximum value.

□

Summarizing, the above results imply the following algorithm: a) find a bi-factor tree $T(G)$ of G, b) for every vertex v of $T(G)$ find a tree decomposition $\mathcal{T}_v$ of the pair of graphs corresponding to v, c) perform a bottom-up traversal of $T(G)$ and at each vertex v make a bottom-up traversal of $\mathcal{T}_v$ using Lemma 4.4.

Theorem 4.5. *For every constant k,* MaxCut *is solvable in polynomial time for co-bipartite graphs in $\mathcal{BF}(\mathcal{T}(k))$.*

5. Extensions and Future Work

In this work we combined the decomposition techniques presented in [3] and [6]. We used these techniques to solve the maximum cardinality cut problem in a new family of graphs. This work can be further extended in the following directions.

New problems: To apply this technique to other problems is an important line of research. The application to maximum independent set problem is work in progress. Our goal is to find a unified framework that can be used to apply the technique to other problems.

Graphs with small chromatic number: The concept of bimodule and the corresponding decomposition algorithms seem to be extendable to any graph of bounded chromatic number. However, since as opposed to bipartitions, to find a χ-coloring of a graph with chromatic number χ is NP-complete, in order to compute such a decomposition, a coloring of the graph must be provided.

Other decomposition types: In this work we replaced the modular decomposition part of the technique presented in [3] with bi-modular decomposition. Alternatively, one can replace the tree-decomposition part with another decomposition technique, or apply other decomposition techniques in addition to the techniques mentioned here.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

References

[1] Barahona, F., Grötschel, M., Jünger, M., Reinelt, G., *An application of combinatorial optimization to statistical physics and circuit layout design*, Operations Research, **36**(3)(1988), 493–513.

[2] Bodlaender, H.L., *A linear-time algorithm for finding tree-decompositions of small treewidth*, SIAM J. Comput., **25**(6)(1996), 1305–1317.

[3] Bodlaender, H.L., Jansen, K., *On the complexity of the maximum cut problem*, Nord. J. Comput., **7**(2000), 14–31.

[4] Diaz,J., Kaminski, M., *Max-cut and max-bisection are NP-hard on unit disk graphs*, Theoretical Computer Science, **377**(1–3)(2007), 271–276.

[5] Ekim, T., Boyacı, A., Shalom, M., *The maximum cardinality cut problem in co-bipartite chain graphs*, Journal of Combinatorial Optimization, **35**(2018), 250–265.

[6] Fouquet, J-L., Habib, M., De Montgolfier, F., Vanherpe, J-M., *Bimodular decomposition of bipartite graphs*, In 30th International Workshop on Graph Theoretic Concepts in Computer Science, WG, 117–128, 2004.

[7] Gallai, T., *Transitiv orientierbare graphen*, Acta Mathematica Academiae Scientiarum Hungarica, **18**(1)(1967), 25–66.

[8] Guruswami, V., *Maximum cut on line and total graphs*, Discrete Applied Mathematics, **92**(2-3)(1999), 217–221.

[9] Hadlock, F., *Finding a maximum cut of a planar graph in polynomial time*, SIAM J. Comput., **4**(3):221–225, 1975.
[10] Heggernes, P., Kratsch, D., *Linear-time certifying recognition algorithms and forbidden induced subgraphs*, Nord. J. Comput., **14**(1-2)(2007), 87–108.
[11] Karp, R.M., *Reducibility among combinatorial problems*, In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds), Complexity of Computer Computations, The IBM Research Symposia Series, Springer, 85–103, 1972.
[12] Tedder, M., Corneil, D., Habib, M., Paul, C., *Simpler linear-time modular decomposition via recursive factorizing permutations*, In: Aceto, L., Damgard, I., Goldberg, L., Halldorsson, M.M., Ingolfsdattir, A., Walukiewicz, I. (eds), Automata, Languages and Programming, volume 5125 of Lecture Notes in Computer Science, Springer, 634–645, 2008.