Article

Bipartite (P_6,C_6)-Free Graphs, Recognition and Optimization Problems

Ruzayn Quaddoura * and Ahmad Al-qerem

Posted Date: 27 February 2024

doi: 10.20944/preprints202402.1561.v1

Keywords: Bipartite graphs; Graphs Decomposition; Complexity; Optimization Problems

Article

Bipartite (P_6, C_6)-Free Graphs, Recognition and Optimization Problems

Ruzayn Quaddoura * and Ahmad Al-qerem

Department of Computer Science, Zarqa University, Jordan
* Correspondence: ruzayn@zu.edu.jo

Abstract: The canonical decomposition of a bipartite graph is a new decompositionmethod that involves three operators, parallel, series and $K + S$. The class of weak bisblit graphs is totally decomposable with respect of these operators, and the class of bicographs is totally decomposable with respect of parallel and series operators. We prove in this paper that the class of bipartite (P_6, C_6)-free graphs is exactly the class of bipartite graphs that are totally decomposable with respect of parallel and $K + S$ operators. We simplify the recognition algorithm of weak-bisplit graphs to adepte with bipartite (P_6, C_6)-free graphs. As a result of this adapted algorithm, we prsent efficient solutions in this class of graphs for two optimization graph problems, the first is the maximm balanced bi-clique problem and the the second is the maximum independent set problem.

Keywords: bipartite graphs; graphs decomposition; complexity; optimization problems

1. Introduction

All graphs under consideration are undirected and simple. A graph $G = (V, E)$ is called bipartite if the vertex set V can be partitioned into two sets B that called the black vertices and W that called the white vertices such that $E \subseteq B \times W$. So a bipartite graph will be referred as $G = (B \cup W, E)$. This property makes bipartite graphs useful in various practical applications, including: recommender systems [18], social networks [13], and information retrieval [10]. A bipartite graph $G = (B \cup W, E)$ is called complete or bi-clique if $E = B \times W$. A complete bipartite graph with $|B| = n$ and $|W| = m$ is referred as $K_{n,m}$. A $Star_{i,j,k}$ is a tree for which there is only one vertex v of degree three and three other vertices of degree one such that the distance from v to those vertices are respectively i, j and k. for example $K_{1,3}$ is $Star_{1,1,1}$. We can remark that every connected component in a bipartite $Star_{1,1,1}$-free graph is either a chordless path or a chordless cycle. Fouquet et al. in [5] presented a decomposition method concerning bipartite graphs, called canonical decomposition, which is based on three operators, series, parallel and $K + S$, and proved that the class of graphs that are totally decomposable with respect of this three operators of decomposition is the class of bipartite $(Star_{1,2,3}, P_7)$-free graphs which is called the class of weak bisplit graphs since it is considered as bipartite analogue of split graphs. Obviously, the class of weak bisplit graphs is a natural generalization of the class of bipartite (P_6, C_6)-free graphs and this last is a natural generalizationof the class of bipartite $Star_{1,2,2}$-free graphs which is studied by Lozin in [17]. Giakoumakis et. al. in [15] defined the class of bicographs as bipartite analogue of cographs. It is proved in [15] that the class of bicographs is exactly the class of bipartite $(Star_{1,2,3}, P_7, Sun_4)$-free graphs and is the class of totally decomposable graphs with respect of parallel and series decompositions.We prove in this work that the class of bipartite (P_6, C_6)-free graphs is the class of totally decomposable graphs with respect of parallel and $K + S$ decompositions. As a result of this fact, the class of bipartite (P_6, C_6)-free graphs can be recognized in linear time using the recognition algorithm of weak bisplit graphs presented in [16] or the recognition algorithm of $Star_{1,2,3}$-free graphs presented in [12]. But since these two algorithms contain several redundant cases when projected to bipartite (P_6, C_6)-free graphs, we propose a simplification of these two algorithms to adapt only the class of bipartite (P_6, C_6)-free graphs. As a result of this adapted algorithm, we prsent

 doi:10.20944/preprints202402.1561.v1

efficient solutions in this class of graphs for two optimization graph problems, the first is the maximm balanced bi-clique problem and the the second is the maximum independent set problem.

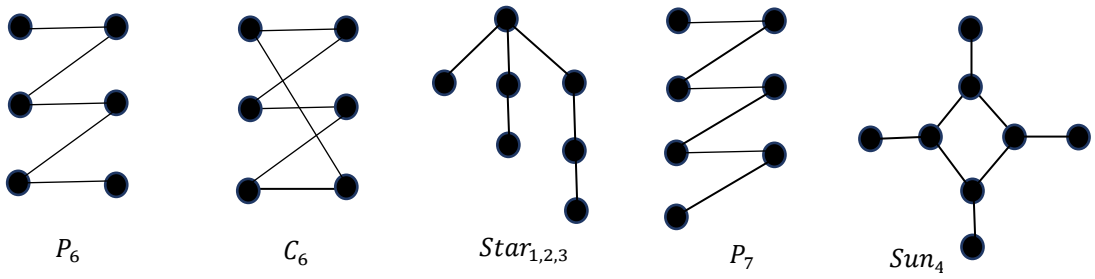


Figure 1. The configuration$P_6, C_6, Star_{123}$and Sun_4.

2. Notation and Terminology

For a graph G, we use $V(G)$ to represent its vertices and $E(G)$ to represent its edges. The number $|V(G)|$ is represented by n and the number $|E(G)|$ is represented by m. If the two vertex sets B and W in a bipartite graph $G = (B \cup W, E)$ are both non-empty, the graph is referred to as bi-chromatic, otherwise it is monochromatic. The bi-complement of a bipartite graph $G = (B \cup W, E)$ is defined as $\bar{G}^{bip} = (B \cup W, B \times W - E)$. The set of neighbors of a vertex v in G is represented by $N(v)$ and the number $|N(v)|$ is referred to as the degree of v and represented by $d(v)$. A vertex v is considered isolated if $d(v) = 0$ and is considered universal if $d(v) = |B|$ when$v \in W$ or $d(v) = |W|$ when $v \in B$. A $2K_2$ is the complement of C_4. A subset S of vertices of $V(G)$ is called independent set if there is no edge between any two vertices of S. The sub-graph induced by a subset X of vertices of $V(G)$ is represented by $G[X]$. A graph G is considered Z-free, where Z is a set of graphs, if for any subset $X \subseteq V(G)$, $G[X] \notin Z$, that is there is no sub-graph of G isomorphic to a graph in Z. Decomposition of graphs according to predefined operators is a powerful method for obtaining efficient solutions of a big number of graph problems. The reader can found a survey on graph decomposition methods and their uses in [11]. In this direction, Fouquet et al. in [5] presented a decomposition method concerning bipartite graphs, which is based on three operators: decomposing a bipartite graph into connected components, decomposing the bi-complement of a bipartite graph into connected components, and decomposing a bipartite graph into $K \oplus S$ components. Our recognition algorithm of bipartite (P_6, C_6)-free graphs depends mainly on this method of decomposition, so to introduce our algorithm we need to present an overview of this method.

Definition 1 [5]. *A bipartite graph $G = (B \cup W, E)$such that $n \geq 2$ is a $K \oplus S$ graph if the vertex set $V(G)$ contains an isolated vertex or there is a partition of $V(G)$ into two sets: a bi-clique K and an independent set S.*

Property 1 [5]. *A bipartite graph $G = (B \cup W, E)$ with $n \geq 2$ is a $K \oplus S$-graph if and only if $V(G)$ can be partitioned into two sets V_1 and V_2, such that for every black vertex $b \in V_1$ and for every white vertex $w \in V_2, bw \in E$ and for every white vertex $w \in V_1$ and for every black vertex $b \in V_2, bw \notin E$.*

We denote by the ordered pair (V_1, V_2) to represent the partition of the vertex set $V(G)$ of a $K \oplus S$-graph G and we called it a $K \oplus S$-partition of G.

Property 2 [5]. *In a bipartite graph $G = (B \cup W, E)$ that does not contain universal or isolated vertices, either G is a $K \oplus S$-graph or, for any partition of the black vertices B or white vertices W into two sets X_1 and X_2, there is an induced $2K_2$ in G with vertices in both X_1 and X_2.*

Corollary 1[5]. *If a graph G is not a $K \oplus S$-graph, then every vertex $v \in V(G), v$ is a vertex of some $2K_2$.*

Theorem 1 provides a method for decomposing a graph of type $K \oplus S$.

Theorem 1 [5]. *A bipartite graph G is a $K \oplus S$-graph if and only if there exists a unique partition $(V_1, \ldots, V_r)$ of $V(G)$, such that the following conditions are hold:*

- *For every $i = 1, \ldots, r$, $V_i \neq \emptyset$.*
- *For every $i = 1, \ldots, r-1$, the sets $V_1 \cup \ldots \cup V_i$ and $V_{i+1} \cup \ldots \cup V_r$ form a $K \oplus S$-*partition *of G.*

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 February 2024 doi:10.20944/preprints202402.1561.v1

- *For every* $i = 1,\dots,r$, *the sub-graph* $G[V_i]$ *is not a* $K \oplus S$*-graph.*

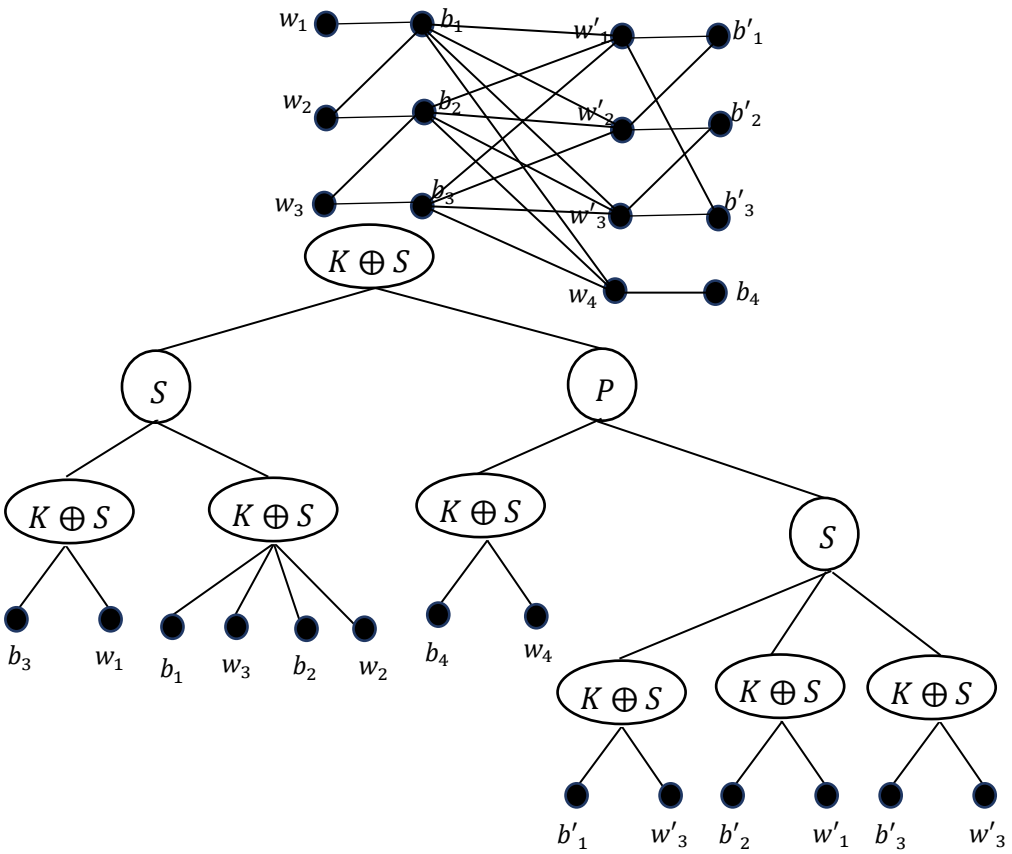
The partition $(V_1,\dots,V_r)$ of the vertex set $V(G)$ is called the $K + S$-decomposition of the graph G, and each set V_i is referred to as a $K \oplus S$-component of G.

According to Theorem 1, the arrangement of components in a $K \oplus S$-decomposition is important. Specifically, let $(V_1,\dots,V_r)$ be the $K \oplus S$-decomposition of G, if a black vertex $b \in V_i$, then for every white vertex $w \in V_j$ where $j > i$, $bw \in E(G)$, and for every white vertex $w \in V_k$ where $k < i$, $bw \notin E(G)$. Similarly, if a white vertex $w \in V_i$, then for every black vertex $b \in V_j$ where $j > i$, $bw \notin E(G)$, and for every black vertex $b \in V_k$ where $k < i$, $bw \in E(G)$.

The canonical decomposition of a bipartite graph G is a new decomposition method defined in [5] as following:

- Decompose G into its $K \oplus S$-components if G is a $K \oplus S$-graph, this decomposition is called $K \oplus S$-decomposition and is denoted by $K \oplus S$.
- Decompose G into the connected components of G if G is not connected graph, this decomposition is called parallel decomposition and is denoted by P.
- Decompose G into the connected components of $\bar{G}^{bip}$ if $\bar{G}^{bip}$ is not connected graph, this decomposition is called series decomposition and is denoted by S.
- If G cannot be decomposed in $K \oplus S$, parallel or series decomposition then G is called indecomposable graph or prime graph.

It has been proven in [5] that no matter the order in which the operator of decomposition is applied (series decomposition, parallel decomposition, or $K \oplus S$-decomposition), the set of indecomposable graphs obtained is unique. This also creates a unique tree (up to isomorphism) associated with this decomposition known as the canonical decomposition tree. The internal nodes of the tree are labeled by the type of decomposition applied, and the leaves correspond to indecomposable graphs. Figure 2 shows an illustration of a bipartite graph and its canonical decomposition tree.



 doi:10.20944/preprints202402.1561.v1

Figure 2. An example of bipartite graph$G$ and the canonical decomposition tree$T(G)$.

The canonical decomposition tree $T(G)$ of a bipartite graph G resulting in order from the $K \oplus S$-decomposition, parallel decomposition, and series decomposition procedure has several properties outlined below. The terms vertex node, son, parent, and grandparent are used in their conventional sense. If α is an internal node then $G[\alpha]$ is the sub-graph induced by the set of vertices nodes having α as their least common grandparent.

1) The tree $T(G)$ consists of 3 types of internal nodes: parallel denoted by P, series nodes denoted by S, and$K \oplus S$-nodes.
2) Two consecutive internal nodes cannot have the same label.
3) An internal node δ labeled P or S cannot have a son that is a vertex node v. Otherwise, v would be either an isolated or a universal vertex in $G[\delta]$. By the definition of a $K \oplus S$-graph, $G[\delta]$ would be a $K \oplus S$-graph.
4) The parent of a vertex node is always labeled $K \oplus S$ (consequence of 3).
5) If G is a bi-chromatic graph, then for any $K \oplus S$-node δ, $G[\delta]$ must also be bi-chromatic. Otherwise, if there is a node δ labelled $K \oplus S$ and $G[\delta]$ is a monochromatic graph then the parent of δ say γ would have isolated or universal vertices so $G[\gamma]$ would be a $K \oplus S$-graph, a contradiction with 2.
6) The sons of a $K \oplus S$-node are ordered according to the $K \oplus S$-decomposition.
7) Let δ be a$K \oplus S$-node and δ_1, δ_2 are respectively the first and last sons of δ. If the parent of δ say γ is a P-node then, δ_1 cannot be a white vertex node and δ_2 cannot be a black vertex node. Otherwise, by Property 1, δ_1 and δ_2 are isolated vertices in $G[\delta]$, since γ is a P-node, δ_1 and δ_2 are also isolated vertices in $G[\gamma]$, so γ must be a $K \oplus S$-node, a contradiction with 2.
8) If the parent of a $K \oplus S$-nodeδ is labeled S and δ_1, δ_2 are respectively the first and last sons of δ then δ_1 cannot be a black vertex node and δ_2 cannot be a white vertex node. Similar to 7.

3. Recognition Algorithm of Bipartite (P_6, C_6)-free Graphs

The following Theorem is the key of our recognition algorithm for bipartite (P_6, C_6)-free graphs.

Theorem 2. *A bipartite graph G is (P_6, C_6)-free if and only if every connected sub-graph of $G$is a $K \oplus S$-graph.*

Proof. Suppose that G is a bipartite graph (P_6, C_6)-free. Let H be a connected sub-graph of G which is not a $K \oplus S$-graph. By Corollary 1, H contains a $2K_2$ say b_1w_1, b_2w_2. Since H is a connected sub-graph of G and G is (P_6, C_6)-free graph, there is a vertex in $V(H)$ that connects b_1w_1 and b_2w_2. Suppose without loss of generality that b is a black vertex such that $bw_1, , bw_2 \in E(H)$. Since H is not $K \oplus S$-graph, the vertex b is not universal, so there is a white vertex w such that $bw \notin E(H)$. Since H is connected there is a path in H that connects the vertex w and the path $P_5 = b_1, w_1, b, w_2, b_2$. But now the set $\{b_1, w_1, b, w_2, b_2, w\}$ forms a P_6 or a C_6, a contradiction.

The inverse is clear since a P_6 or a C_6 is connected and contains a $2K_2$, so it is not a $K \oplus S$-graph. □

Theorem 2 states that the class of bipartite (P_6, C_6)-free graphs is the smallest class closed under parallel and $K \oplus S$-decomposition. So the canonical decomposition tree of a bipartite (P_6, C_6)-free graph consists only of P-nodes or $K \oplus S$-nodes. Our recognition algorithm builds a decomposition tree with P or $K \oplus S$ labeled internal nodes if the input graph is (P_6, C_6)-free otherwise a failure message.This building was influenced by the cographs recognition method proposed by Corneil, et al in [2]. Moreover, this algorithm greatly simplifies two recognition algorithms when projected on bipartite (P_6, C_6)-free graphs. The first is for weak bisplit graphs presented in [16] and the second for bipartite $Star_{123}$-free graphs presented in [12], where both these two algorithms need to examine more than twenty cases in order to confirm that the input graph is (P_6, C_6)-free or not, while, as we will see, our algorithm needs to examine only two cases that are presented below in Theorem 3 and

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 February 2024 doi:10.20944/preprints202402.1561.v1

in Theorem 4. The algorithm begins with an empty graph and gradually adds vertices, ensuring that the resulting sub-graphs remain (P_6, C_6)-free. The initial bipartite graph is considered (P_6, C_6)-free if all vertices can be added successfully in this manner. The principal step of the algorithm takes into considerationthe decomposition tree $T$of a (P_6, C_6)-free bipartite graph $G = (B \cup W, E)$, a vertex $x \notin B \cup W$, and a set of edges denoted by $E(x) = \{xv: v \in B \cup W \text{ and } v \in N(x)\}$, and produces the decomposition tree T' of the resulting graph $G' = (B \cup W \cup \{x\}, E \cup E(x))$ if it remains (P_6, C_6)-free, or stops otherwise. The algorithm considers the connections of x to other vertices in G using a marking procedure. We can assume without loss of generality that x is a white vertex and the graph G is a bi-chromatic graph.

3.1. Marking Procedure

The Marking procedure, used in [2], takes into consideration the neighbors of the vertex x in the graph G to mark the nodes of T, the decomposition tree of G.

Algorithm 1: Marking

Input: The canonical decomposition tree T of G, and the white vertex x.

Output: The marking tree T.

For every black vertex node v of T.

If v is a neighbor of x mark v by (t), if v is not a neighbor of x, mark v by $()$.

Traverse T on a bottom-up traversal, let α be an internal node of T:

If every son of α which is distinct of white vertex node is marked by $()$ then mark α by $()$.

If there is a son of α marked by (t) and a son marked by $()$ then mark α by (p).

If every son of α which is distinct of white vertex node is marked by (t) then mark α by (t).

At the end of the marking procedure on tree T, a node can have three possible states: marked by(t), marked by (p), or marked by $()$. If a node δ is marked by (t), it means that x is total for $G[\delta]$, that is, x is connected to all black vertices in $G[\delta]$. If it is marked by (p), it means that x is partial for $G[\delta]$, that is, x is connected to some but not all black vertices in $G[\delta]$. If it is marked by $()$, it means that x is independent of $G[\delta]$, that is, it x is not connected to any black vertex in $G[\delta]$. If a node is a vertex node, it can either be marked by $()$ or marked by (t). By Theorem 2, the marking procedure focuses only on P-nodes and $K \oplus S$-nodes, ignoring the S-nodes that must be unavailable. For the graph G' to be considered bipartite (P_6, C_6)-free, it must meet a necessary condition.

Lemma 1. *If two internal nodes in the tree $T$are marked by(p), and G' is a (P_6, C_6)-free bipartite graph, then one of these two nodes must be a grandparent of the other.*

Proof. Suppose that α and β are two internal nodes marked by (p) then α and β are partial with respect to x. Let δ be the least common grandparent of $\alpha$and β. We denote by $\alpha'$and β' to be respectively the son of δ containing α and the son of δ containingβ. Since $G[\alpha]$ and $G[\beta]$ are sub-graphs of $G[\alpha']$ and $G[\beta']$ then $\alpha'$and $\beta'$are partial with respect to x. Assume that δ is labeled P, then $G[\alpha']$ and $G[\beta']$ are connected sub-graphs of $G[\delta]$. Thus there is an induced path b_1, w_1, b_2 in $G[\alpha']$ (resp. b'_1, w'_1, b'_2 in $G[\beta']$) such that x is adjacent to b_1 and not adjacent to b_2 (resp. to b'_1 and not to b'_2). The set $\{w'_1, b'_1, x, b_1, w_1, b_2\}$ forms a P_6, a contradiction.

Assume that δ is labeled $K \oplus S$. By Corollary 1,α' (resp.β') contains a $2K_2$ that is partial with respect to x. Let b_1w_1, b_2w_2 (resp. $b'_1w'_1, b'_2w'_2$) be a $2K_2$ in $G[\alpha']$ (resp. in $G[\beta']$ such that x is adjacent to b_1 and not adjacent to b_2 (resp. to b'_1 and not to b'_2). Then the set $\{w_2, b_2, w'_2, b_1, x, b'_1, w'_1\}$ forms a P_6, a contradiction. □

By lemma 1, the nodes in T that are marked by (p) are arranged in a single path that starts from the lowest node marked by (p) and goes up to the root. The lowest node marked by (p) is referred to as α. It is assumed that the conditions of Lemma 1 are met and that α is known. The following notations are introduced:

- Given two internal nodes δ and δ' such that δ is a grandparent of δ', the unique son of δ

 doi:10.20944/preprints202402.1561.v1

that contains δ' is denoted as $son(\delta, \delta')$.

- For an internal node δ, which is eitherα or one of its grandparents, the set of sons of δ that are marked by $()$ is denoted as $sons^{()}(\delta)$, and the set of sons that are marked by (t) is denoted as $sons^{(t)}(\delta)$.
- If δ has a label $K \oplus S$, considering the ordering of the sons of δ, the set of sons of δ marked by (t) and located before $son(\delta, \alpha)$ is denoted as $sons_1^{(t)}(\delta)$, and the set of sons marked by (t) and located after $son(\delta, \alpha)$ is denoted as $sons_2^{(t)}(\delta)$. The set of sons of δ marked by $()$ and located before $son(\delta, \alpha)$ is denoted as $sons_1^{()}(\delta)$, and the set of sons marked by $()$ and located after $son(\delta, \alpha)$ is denoted as $sons_2^{()}(\delta)$.

So, a node δ labeled P which is a grandparent of α splits its sons into at most three categories: $sons^{(t)}(\delta)$, $sons^{()}(\delta)$, and $son\,(\delta, \alpha)$ the son that containsα,. If δ *is* labeled $K \oplus S$, its sons can be divided into at most five sets: $son(\delta, \alpha)$, $sons_1^{(t)}(\delta)$, $sons_1^{()}(\delta)$, $sons_2^{(t)}(\delta)$, and $sons_2^{()}(\delta)$. Meanwhile, the sons of α are split into two non-empty categories: $son^{()}(\alpha)$, and $sons^{(t)}(\alpha)$.

Definition 2. *If δ is a grandparent of αin T, then δ is considered incompatible P-node if it has at least one son marked by (t) (i.e. $sons^{(t)}(\delta) \neq \emptyset$). If δ is a $K + S$-node, it is considered incompatible before α if it has at least one son marked by $()$ located before $son(\delta, \alpha)$ (i.e. $sons_1^{()}(\delta) \neq \emptyset$) . If δ is a $K \oplus S$-node, it is considered incompatible after α if it has at least one son marked by (t) located after $son(\delta, \alpha)$ (i.e. $sons_2^{(t)}(\alpha) \neq \emptyset$).*

3.2. Building the tree T'

We assume that the necessary condition of Lemma 1 has been verified and all nodes that are marked by (p) are known and are arranged in a single path that starts from the lowest marked node α and goes up to the root of T. In order to build T', we examine the label of α and the presence of incompatible marked nodes.When α is $K \oplus S$-node and since it is the lowest node marked by (p), we can divide its group of sons into a maximum of four consecutive subsets, namely $X_1^{(t)}$, $X_2^{()}$, $X_3^{(t)}$, X_4 where:

$X_1^{(t)}$includes the first group of consecutive sons of α that are either a group of white vertices nodes or total with respect to x.

$X_2^{()}$includes the first group of consecutive sons of α that are not part of $X_1^{(t)}$ and either a group of white vertices nodes or not related to x.

$X_3^{(t)}$includes the first group of consecutive sons of α that are not part of $X_2^{()}$ nor of $X_1^{(t)}$ and either a group of white vertices nodes or total with respect to x.

X_4represents the remaining sons of α.

Note that, as a result of this division of the sons of α when its label is $K \oplus S$, $X_2^{()}$ and $X_3^{(t)}$cannot be together monochromatic graphs.

Lemma 2. *Assume that α is a $K \oplus S$-node. If G' is a bipartite (P_6, C_6)-free graph then the following conditions are hold:*

1) x has no neighbor in X_4.

2) If $X_3^{(t)}$is empty then $G[X_3^{(t)}]$ is a monochromatic graph or a complete bipartite graph.

Proof. Suppose that X_4 is not empty, otherwise we are done. Let's show that x has no neighbor in X_4. Since X_4 is not empty then $X_3^{(t)}$ and $X_2^{()}$ are both non empty. Let $b_4 \in X_4$ such that x is adjacent to b_4. Now, X_4 contains two adjacent vertices b'_4, w'_4 such that x is not adjacent to b'_4 otherwise $b_4 \in X_3^{(t)}$. Let b_2, b_3 be two black vertices of $X_2^{()}$and $X_3^{(t)}$respectively. By construction, there is a white vertex w such that w is adjacent to b_2 and w is not adjacent to b_3. But now the set $\{b_4, x, b_3, w_4, b_2, w\}$ forms a P_6, a contradiction.

Let's show now that the condition 2 must be hold. Suppose that $X_3^{(t)}$is not empty then $X_2^{()}$is also non empty.

 doi:10.20944/preprints202402.1561.v1

Claim 1. *Every element of* $X_3^{(t)}$ *is a vertex node.*

Proof. Suppose that δ is an element of $X_3^{(t)}$ that is an internal node. Then δ is a P-node, thus it contains a $2K_2$ say b_1w_1, b_2w_2. Let $b \in X_2^{()}$, then the set $\{b, w_1, b_1, x, b_2, w_2\}$ forms a C_6, a contradiction. ■

Claim 2. $X_2^{()}$*contains a white vertex.*

Proof. Suppose that $X_2^{()}$ does not contain any white vertex then $X_3^{(t)}$contains an element that is an internal node, a contradiction with claim 1. ■

Let b_2, w_2 be two vertices of $X_2^{()}$ such that $b_2w_2 \in E(G)$. Suppose that $G[X_3^{(t)}]$ is neither a monochromatic graph nor a complete bipartite graph. Then $X_3^{(t)}$contains the vertices b_3, b'_3, w_3 such that b_3 is adjacent to w_3 and b'_3 is independent of w_3. Consequently $\{b'_3, x, b_3, w_3, b_2, w_2\}$ forms a P_6, a contradiction.□

Theorem 3. *Assume that there is no incompatible grandparent of* α. G'*is a bipartite* (P_6, C_6)*- free graph if and only if one of either*

1) α *is a* P*-node or*

2) α *is a* $K \oplus S$*-node and Lemma 2 is hold.*

Proof. The if part of the Theorem has been proved in lemma 2. We will describe the building of T' for the only if part. The building of T' when αi s aP-node is described in Figure 3.a. If $sons^{(t)}(\alpha)$ consists of a unique son then this son will be a son of the node labeled $K \oplus S$. Suppose that α is a $K \oplus S$-node. If $X_3^{(t)}$is empty, then we insert x in T as a new son of α. The building of T' when $G[X_3^{(t)}]$ is a monochromatic graph or a complete bipartite graph is described also in Figure 3.b. In this case, if $G[X_3^{(t)}]$ is a monochromatic graph then $W_\alpha = \emptyset$. □

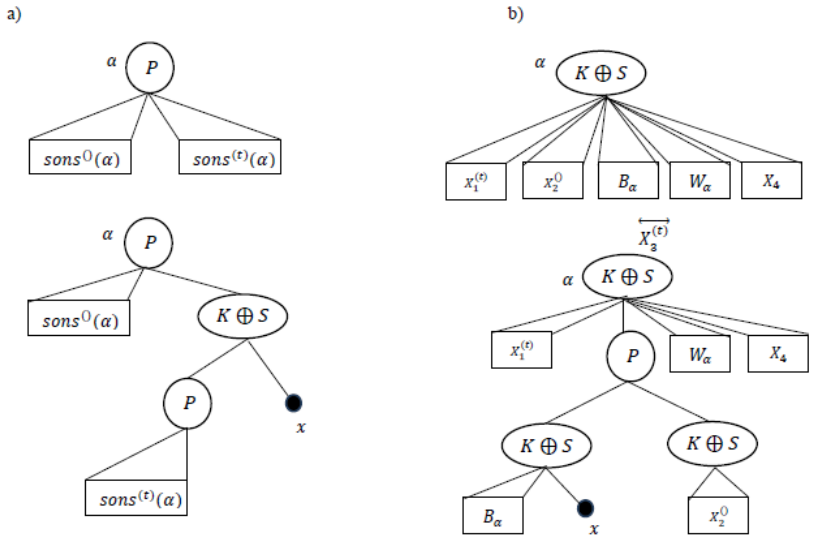


Figure 3. Building of T' when there is no incompatible grandparent of α.

Lemma 3. *Assume that* G' *is a bipartite* (P_6, C_6)*-free graph. If there is an incompatible grandparent* β *of* α *then the following condition are holds:*

1) β *is a* $K \oplus S$*-incompatible node after* α.

2) β *is the unique incompatible grandparent of* α.

3) The set $sons_2^{(t)}(\beta)$ *consists of black vertices nodes located exactly after* $son(\beta, \alpha)$.

Proof. Suppose that β is an incompatible grandparent of α of type P. Then there exists two adjacent vertices b_3, w_3 in an element of $sons^{(t)}(\beta)$. Since $son(\beta, \alpha)$ induces a connected graph, it contains an induced P_3 say b_1, w_1, b_2 such that b_1 is adjacent to x and b_2 is independent of x. But now $\{x, b_1, w_1, b_2, b_3, w_3\}$ forms a P_6, a contradiction.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 February 2024 doi:10.20944/preprints202402.1561.v1

If β is a $K \oplus S$-incompatible node before α, then $sons_1^0(\beta)$ contains a black vertex b independent of x. By Corollary 1, $son(\beta, \alpha)$ contains a $2K_2$ say b_1w_1, b_2w_2 such that x is adjacent to b_1 and x is independent of b_2. The set $\{x, b_1, w_1, b, w_2, b_2\}$ forms a P_6, a contradiction. Consequently β is a $K \oplus S$-incompatible node after α. Let's consider β to be the highest incompatible grandparent of $\alpha$and let $b \in sons_2^{(t)}(\beta)$.

Claim. *There is no grandparent δ of αcontaining a white vertex total for $son(\delta, \alpha)$.*

Proof. Let δ be a grandparent of α and w is a white vertex of δ total for $son(\delta, \alpha)$. Let b_1w_1, b_2w_2 be an induced $2K_2$ of $son(\delta, \alpha)$ such that x is adjacent to b_1 and independent of b_2. Then the set $\{b, x, b_1, w, b_2, w_2\}$ forms a P_6, a contradiction. ■

By this claim, if δ is a $K \oplus S$-incompatible node after α then the set $sons_2^{(t)}(\delta)$ consists of black vertices nodes located exactly after $son(\delta, \alpha)$. Moreover, for every grandparent δ of α labeled $K \oplus S$ and located between α and β, $son(\delta, \alpha)$ is the last son of δ, otherwise δ contains a white vertex total for $son(\delta, \alpha)$, a contradiction, thus δ cannot be incompatible. Therefore, $\beta$is the unique incompatible grandparent of α. □

Theorem 4. *Assume that $\beta$is the unique $K \oplus S$-incompatible grandparent after α and the set $sons_2^{(t)}(\beta)$ consists of black vertices nodes located exactly after $son(\beta, \alpha)$. G' is a bipartite (P_6, C_6)-free graph if and only if one of the following conditions is hold:*

1) α is a P-node.

2) α is a $K \oplus S$-node such that $X_3^{(t)}$ is an empty set.

Proof. Suppose that α is a $K \oplus S$-node and let $b \in sons_2^{(t)}(\beta)$. Suppose that $X_3^{(t)}$ is non empty. By lemma 2, $G[X_3^{(t)}]$ is a monochromatic graph or a complete bipartite graph. In the two cases $G[X_2^0]$ cannot be a monochromatic graph otherwise $X_3^{(t)}$ is empty. Thus X_2^0 contains two adjacent vertices b_2, w_2. Let b_3 be a black vertex of $X_3^{(t)}$. Since $G[\alpha]$ is a connected graph then there is a white vertex say w_3 in the last son of α, but now $\{b, x, b_3, w_3, b_2, w_2\}$ forms a P_6, a contradiction.

For the only if part, we describe the building of T'.When α is a P-node, the building of T' is illustrated in Figure 4.a. If the set $sons^{(t)}(\alpha)$ is a unique son then this son must be labeled $K \oplus S$. In this case, we delete the node δ_1 and the element $sons^{(t)}(\alpha)$ will be a son of δ_2. The building of $T'$when the condition 2 is hold is illustrated in Figure 4.b. If X_2^0 is a unique son, then this son is either a node labeled P or a black vertex node. In this case, we delete the node δ_2, and the element X_2^0will be a son of δ_1.□

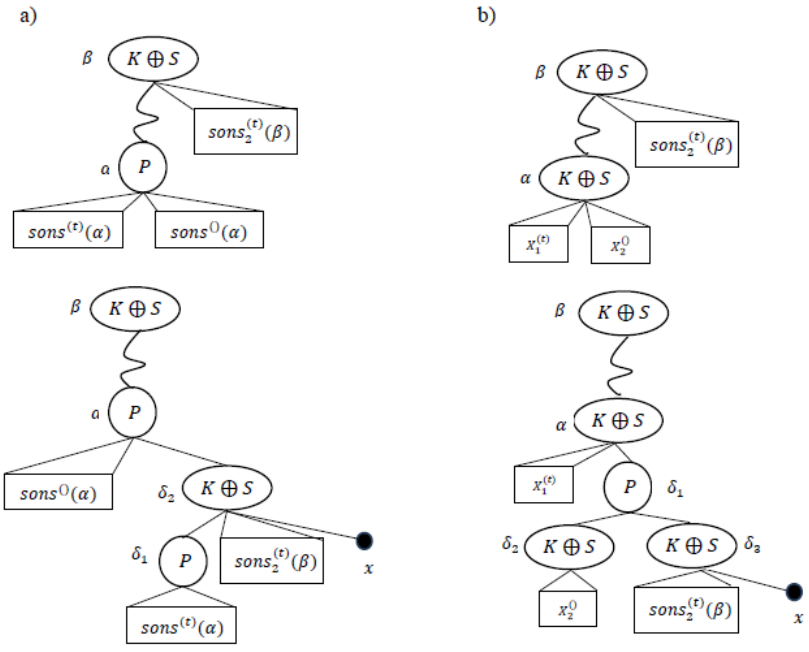


Figure 4. Building of $T'$when β the unique $K + S$-incompatible grandparent of α exist.

 doi:10.20944/preprints202402.1561.v1

3.3. Recognition Algorithm

The recognition algorithm of bipartite (P_6, C_6)-free graphs is given by Algorithm 2, where the procedure of the step Build-tree $(G', T, head(L))$ is presented in Algorithm 3.

Algorithm 2 Recognition of bipartite (P_6, C_6)-free graph

Input: a bipartite graph $G = (B \cup W, E)$.

Output: if G is (P_6, C_6)-free graph then the canonical decomposition tree $T(G)$, otherwise a failure message "G is not (P_6, C_6)-free graph".

Initialization step: Let L be the list of all the vertices of G sorted in descending order according to their degrees.

T =new-vertex;

$G' = \emptyset$;

Build-tree $(G', T, head(L))$.

Algorithm 3 Procedure Build-tree $(G', T, head(L))$

1) Marking(T, x)
2) Find the set $S = \{\delta: \delta$ is an internal node marked by $(p)\}$
3) If $S = \emptyset$ then $T = insert(x, T)$
   (If x is independent of T (resp. total for T) then create a new root δ of T labeled $K \oplus S$ such that x is the left (resp. right) son of δ and the root of T is the right (resp. left) son of δ)
4) Else if $|S| > 2$ than Exit with the message "failure".
5) Else if $|S| = 1$ then $T = insert(x, T)$ (according to Theorem 3)
6) Else if one of the two nodes of S is not a grandparent of the other then Exit with the message "failure".
   Else let $S = \{\alpha, \beta\}$ and β is the grandparent of α
7) If β is a P-node then Exit with the message "failure".
8) Else $T = insert\ (x, T)$ (according to Theorem 4)
9) $G' = G[V(G') \cup \{x\}]$;
10) If $L = \emptyset$then Exit else $L = L - \{x\}; x = head(L)$; Build-tree$(G', T, x)$

3.4. Complexity

Our aim is to demonstrate that recognizing a bipartite graph G, which does not contain P_6 or C_6, can be accomplished in $O(n + m)$ time complexity. Since the principal step of our algorithm is the step Build-tree(G', T, x), we will demonstrate the linearity of our algorithmby showing that this step requires only$O(d_{G'}(x))$ operations, where $d_{G'}(x)$ is the degree of the node x in G'.

It is evident that the step 1 runs within $O(d_{G'}(x))$ time, as only a maximum of $O(d_{G'}(x))$ nodes are marked. Furthermore, we can assume that for every node in the tree T, the set of its sons that are marked by (t), by (p) or by () has been calculated. So find the set S requires also $O(d_{G'}(x))$ operations. Suppose that $S = |2|$. We can check whether one of the two nodes of S is a grandparent of the other as following: Choose an element of S and start to mark the parent of this element, then mark the parent of parent and so on until the other element of S is marked or until the root of T is marked. For the last case, that is, if the root of T has been marked, we repeat this process for the other element of S. By this manner, we can also determine the node α, the lowest node marked by (p) and the grandparent β. Obviously, this process can be done in $O(d_{G'}(x))$ mark operations.

It remain analysis the time complexity of the function $insert\ (x, T)$. This requires to verify the necessary conditions in Theorem 3 or in Theorem 4. Therefore we need to compute all required sets

 doi:10.20944/preprints202402.1561.v1

for building the tree T'. If α has label P, then the computation of the set $sons^{(t)}(\alpha)$ and the set $sons^{()}(\alpha)$ is straightforward. Suppose α has label $K \oplus S$. We can compute the sets $X_1^{(t)}$, $X_2^{()}$, $X_3^{(t)}$,and X_4 as following: First, we compute $X_1^{(t)}$ by traversing the set of sons of α from left to right. In this manner, $X_1^{(t)}$ will be the first nodes that are either a set of white vertices or nodes marked by (t), we continue by this traversing until a son of α marked by $()$ has been found. The remaining sons of α marked by (t) must belong to $X_3^{(t)}$ since X_4 is independent of x according to Lemma 2. To compute $X_3^{(t)}$, we choose a son of α (let's call it c) from the remaining nodes marked by (t) and we traverse the set of sons of α starting from c in left and right directions, from left until a son of α marked by $()$ has been found and from right also until a son of α marked by $()$ has been found or until the last son from right has been found. We continue by this traversing as long as every son is either a white vertex node or is a node marked by (t). As long as the set $X_3^{(t)}$ has been computed, the sets $X_2^{()}$ can be computed immediately. The remaining sons of α form the set X_4 which must be independent of x This computation requires $O(d_{G'}(x))$ time complexity.

Finally, we need to determine if the node β, which its label is $K \oplus S$, is incompatible after α or not and check whether the set $sons_2^{(t)}(\beta)$ is a set of black vertices nodes located exactly after $son(\beta, \alpha)$. This two conditions can be achieved together as following: Since β is known as a grandparent of α, the son $son(\beta, \alpha)$ is identified. Now, we can traverse the sons of β starting from the first son located exactly after $son(\beta, \alpha)$ and determine whether any one of these sons is a white vertex node or not. In addition, we must traverse the sons of β starting from the first son located exactly before $son(\beta, \alpha)$ to determine if the set $sons_1^{()}(\delta)$ is empty or not. This traverse also requires $O(d_{G'}(x))$ time complexity.

we leave to the reader to verify that the step Build-tree$(G, T, x))$ that correspond to insert x in the tree T takes a constant time in all cases.

Since testing whether $G' = G \cup \{x\}$ is a bipartite (P_6, C_6)-free or not can be done within $O(d_{G'}(x))$ time complexity, it is clear that recognition of bipartite (P_6, C_6)-free graph algorithm runs in $O(n + m)$ time complexity.

4. Optimization Problems

We believe that the canonical decomposition tree for bipartite (P_6, C_6)-free graph can be used to find efficient solutions for several optimization graph problems because of the simple structure of this tree.In this paper, we limit ourselves to show that the canonical decomposition tree of a bipartite graph (P_6, C_6)-free can be used to solve in polynomial time the maximal balanced bi-clique problem and in linear time the maximum independent set problem. In the conclusion part of this paper, we talk about some potential use for this result and consider it as a subject for further study.

Let $T(G)$ be the canonical decomposition tree for a bipartite (P_6, C_6)-free graph G. To present our solutions of the above two problems, we need to covert $T(G)$ to a binary tree as following:

Visit the nodes of $T(G)$ in depth first search

Let S be an internal visited node and $S_1, \dots, S_k$ are the sons of S.If $k > 2$then the left son of S is S_1 and the right son becomes a new son S' that has the same label as S with sons $S_2, \dots, S_k$.

4.1. Maximum balanced bi-clique problem

A sub-graph $F = G[X \cup Y]$ of a bipartite graph G is called balanced bi-clique if F is a bi-clique and $|X| = |Y|$. The balanced bi-clique problem is to compute a balanced bi-clique in G of maximum size. This problem is important in many different fields of study. It has numerous practical uses in very large-scale integration (VLSI), such as the design of defect-tolerant devices [1,7], programmable logic array folding [14]. Balanced bi-clique problem is NP-complete for a general bipartite graph [9], and there are very few works dedicated to obtaining an exact maximum balanced bi-clique, aside from the work [6] where it is proposed two exact algorithms to find a maximum balanced bi-clique for small dense and large sparse bipartite graphs respectively. The majority of known techniques for determining a maximum balanced bi-clique are heuristic algorithms, see for example, [8,19].

 doi:10.20944/preprints202402.1561.v1

We propose in this work an $O(n^3)$ time complexity algorithm to compute a maximum balanced bi-clique in a bipartite (P_6, C_6)-free graph G using its canonical decomposition binary tree $T(G)$. The idea of our solution is to compute all possible bi-cliques in G, that are maximal with respect to set inclusion, then find among them the one that contains a maximum balanced bi-clique. A bi-clique F is maximal with respect to set inclusion if there is no bi-clique in G that contains F. The structure of $T(G)$ when G is a bipartite (P_6, C_6)-free graph and the definition of $K \oplus S$ operation allow us to achieve this computation by a post order traversal of $T(G)$ and associating for each internal node α all possible maximal bi-cliques in $G[\alpha]$ (with respect to set inclusion) through the two sets ofmaximal bi-cliques associated with the left son α_1 and the right son α_2 of α.The set of maximal bi-cliques associated with α_1 denoted by $L(\alpha_1) = \{F_i^1 = G[X_i^1 \cup Y_i^1] : i = 1, \dots, r\}$ and the set of maximal bi-cliques associated with α_2 denoted by $L(\alpha_2) = \{F_i^2 = G[X_i^2 \cup Y_i^2] : i = 1, \dots, k\}$. We suppose that for every bi-clique $F_i^j = G[X_i^j \cup Y_i^j]$, X_i^j is a set of black vertices and Y_i^j is a set of white vertices. In addition, we suppose that the members of $L(\alpha_1)$ are arranged from left to right according to their appearance in the sub-tree $T(\alpha_1)$. Likewise, we suppose that the members of $L(\alpha_2)$ are arranged from left to right according to their appearance in the sub-tree $T(\alpha_2)$. This supposition is done directly according to the arrangement of sons for every $K \oplus S$-node in $T(G)$. The reader can verify simply the truth of computation used in the algorithm Balanced Bi-clique for the set of maximal bi-cliques $L(\alpha)$ according to the definition of $K \oplus S$-node and the definition of P-node. Figure 5 can help to imagine this computation.

Algorithm Balanced Bi-clique

Input: A binary canonical decomposition tree $T(G)$ of a bipartite (P_6, C_6)-free graph $G = (B \cup W, E)$.

Output: A maximal balanced bi-clique $F = G[X \cup Y]$ for G

Let α be a node on a post order traversal of $T(G)$

If α is a black vertex node b (resp. a white vertex node w) then $L(\alpha) = \{G[\{b\} \cup \emptyset]$ (resp.$G[\emptyset \cup \{w\}]$

Else let α_1 and α_2 be the left and right son of α respectively and let $(\alpha_1) = \{F_i^1 = G[X_i^1 \cup Y_i^1] : i = 1, \dots, r\}$, $L(\alpha_2) = \{F_i^2 = G[X_i^2 \cup Y_i^2] : i = 1, \dots, k\}$.

If α is a $K \oplus S$-node then

Let

$$L_1 = \{G[(X_1^1 \cup \dots \cup X_r^1) \cup (Y_1^2 \cup \dots \cup Y_k^2)]\}$$

$$L_2 = \{G[X_i^1 \cup (Y_i^1 \cup Y_1^2 \cup \dots \cup Y_k^2)] : i = 1, \dots r\}$$

$$L_3 = \{G[(X_i^2 \cup X_1^1 \cup \dots \cup X_r^1) \cup Y_i^2] : i = 1, \dots k\}$$

If $r \neq 1$ or $k \neq 1$ then $L(\alpha) = L_1 \cup L_2 \cup L_3$ else $L(\alpha) = L_2 \cup L_3$

Else// α is a P-node //$L(\alpha) = L(\alpha_1) \cup L(\alpha_2)$

If α is the root of $T(G)$ then let $L(\alpha) = \{F_i = G[X_i \cup Y_i], i = 1, \dots, s\}$

Let $s_t = \max\{\min(|X_1|, |Y_1|), \dots, \min(|X_s|, |Y_s|)\}$ return F_t

---

The number of bi-cliques computed for each internal node is at most $O(n^2)$. Since $T(G)$ contains $O(n)$ node, the algorithm Balanced Bi-clique has a time complexity $O(n^3)$.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 February 2024 doi:10.20944/preprints202402.1561.v1

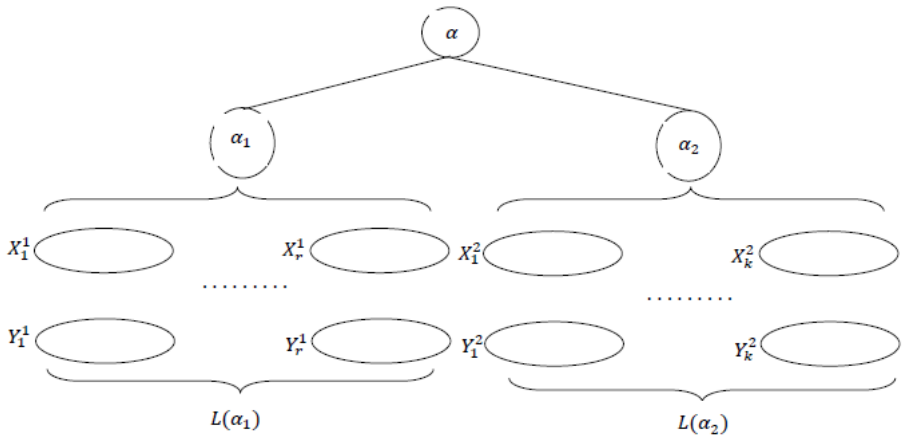


Figure 5. A node α and the sets of all maximal bi-cliques associated with its sons.

4.2. Maximum independent set problem

A subset S of the vertex set $V(G)$ in a graph G is called independent set if any two vertices in S are not adjacent. The maximum independent set problem is to compute an independent set in G of maximum size. This problem is NP-complete for general graphs [9], but it can be solved in $O(n^{1.5}\sqrt{m/\log n})$ time complexity for a general bipartite graph [4]. This time complexity can improved to $O(n)$ for bipartite (P_6, C_6)-free graph using its canonical binary decomposition tree.. The idea of our solution results from the structure of $K \oplus S$-graph G as following: Let (V_1, V_2) be a $K \oplus S$-partition of the vertex set $V(G)$. By Property 1, every black vertex of V_1 is connected to every white vertex of V_2 and every white vertex of V_1 is independent to every black vertex of V_2. So, the maximum independent set in G is either the maximum independent set in $G[V_1]$ or the maximum independent set in $G[V_2]$ or the independent set formed by the union of white vertices of $G[V_1]$ and black vertices of $G[V_2]$. This remark proves the correctness of the following algorithm. Note that if G is not connected then the maximum independent set in G is equal to the union of the maximum independent sets in its connected components.

Algorithm Maximum Independent Set

Input: A binary canonical decomposition tree $T(G)$ of a bipartite (P_6, C_6)-free graph $G = (B \cup W, E)$.

Output: A maximum independent set S for G

Let α be a node on a post order traversal of $T(G)$.

If α is a black vertex node b (resp. a white vertex node w) then

$S(\alpha) = \{b\},\ B(\alpha) = \{b\},\ W(\alpha) = \emptyset$ (resp.$S(\alpha) = \{w\},\ B(\alpha) = \emptyset,\ W(\alpha) = \{w\}$)

Else let α_1 and α_2 be respectively the left and right son of α and let $S(\alpha_1)$ be a maximum independent set of $G[\alpha_1]$ and $S(\alpha_2)$ is a maximum independent set of $G[\alpha_2]$, let $W(\alpha_1), B(\alpha_1)$ be respectively the white and black vertices of $G[\alpha_1]$ and $W(\alpha_2), B(\alpha_2)$ are respectively the white and black vertices of $G[\alpha_2]$.

If α is a $K \oplus S$-node then

$s = \max\{|S(\alpha_1)|, |S(\alpha_2)|, |W(\alpha_1) \cup B(\alpha_2)|\}$

$S(\alpha) = S$ where $|S| = s,\ W(\alpha) = W(\alpha_1) \cup W(\alpha_2)$ and $B(\alpha) = B(\alpha_1) \cup B(\alpha_2)$

else// α is a P-node

$S(\alpha) = S(\alpha_1) \cup S(\alpha_2),\ W(\alpha) = W(\alpha_1) \cup W(\alpha_2)$ and $B(\alpha) = B(\alpha_1) \cup B(\alpha_2)$

Since $T(G)$ contains $O(n)$ node, the algorithm maximum independent set has a time complexity $O(n)$.

 doi:10.20944/preprints202402.1561.v1

Conclusion

we have showed in this paper that bipartite (P_6, C_6)-free graphs can be recognized in linear time. Using this result, we solved two optimization graph problems in this class of graphs, the first is the maximm balanced bi-clique problem and the the second is the maximum independent set problem. An additional potential using of the canonical decomposition tree of bipartite (P_6, C_6)-free graph is to solve the problem P| prec. $p_j = 1$| C_{max} **:** Suppose there are n tasks with a unit execution time, and their order is constrained by a directed acyclic graph. Additionally, there are m machines of the same type. The goal is to discover a schedule that minimizes the makespan, which is the time when the final task in the graph finishes its execution. It is proved in [3] that this problem is NP-complete even if the precedence constraints form a bipartite graph of depth one. We conjecture that this problem can be solved in polynomial time for bipartite (P_6, C_6)-free graph.

References

1. A. Al-Yamani, S. Ramsundar, and D. K. Pradhan. A defect tolerance scheme for nanotechnology circuits. IEEE Transactions on Circuits and Systems I: Regular Papers, 54(11):2402–2409, 2007.
2. D. G. Corneil, Y. Perl, L.K. Stewart, A linear recognition algorithm for cographs. SIAM Journal of Computing, Vol.14, No 4, November, 926-934. 1985.
3. E. BAMPIS, The Complexity of short schedules for UET bipartite graphs, RAIRO Oper. Res. 33 367-370. (1999).
4. H. Alt, N. Blum, K. Mehlhorn, M. Paul, Computing a maximum cardinality matching in bipartite graphs in time $n^{1.5}\sqrt{m/\log n}$, Inform. Process. Lett. 37, 237–240 (1991).
5. J.L. Fouquet, V. Giakoumakis, J.M. Vanherpe, Bipartite graphs totally decomposable by canonical decomposition, Internat. J. Foundations Comput. Sci. 10(4) 513–533. (1999).
6. L. Chen, C. Liu, R. Zhou, J. Xu, J. Li. Efficient Exact Algorithms for Maximum Balanced Biclique Search in Bipartite Graphs, SIGMOD '21: Proceedings of the 2021 International Conference on Management of Data, June 2021Pages 248–260 18
7. M. B. Tahoori. Application-independent defect tolerance of reconfigurable Nano architectures. ACM Journal on Emerging Technologies in Computing Systems (JETC), 2(3):197–218, 2006.
8. M. Li, J.-K. Hao, and Q. Wu. General swap based multiple neighborhood adaptive search for the maximum balanced biclique problem. Computers & Operations Research, 119:104922, 2020.
9. M.R. GAREY and D. S. JOHNSON, Computers and Intractability, A Guide to the Theory of NP-completeness, Mathematical Series. Freeman, San Francisco, CA, 1979.
10. P. Biró, & Fleiner, T..Recent advances in matching algorithms. European Journal of Operational Research, 294(3), 745-769. (2022).
11. R. Quaddoura, K. Mansour, Classical graphs decomposition and their totally decomposable graphs, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.1, January 2010.
12. R. Quaddoura, Linear time recognition of bipartite $Star_{123}$-free graphs, The international Arab Journal of Information Technology, Vol. 3, No. 3, July (2006).
13. R. Wang, Y. Wu, X. Hu, & J. Liu, Bipartite graph neural networks for social recommendation. Information Sciences, 572, 396-409 (2021).
14. S. Ravi and E. L. Lloyd. The complexity of near-optimal programmable logic array folding. SIAM Journal on Computing, 17(4):696–710, 1988.
15. V. Giakoumakis, J.-M. Vanherpe, Bi-complement reducible graphs, Adv. Appl. Math. 18 (1997) 389– 402.

 doi:10.20944/preprints202402.1561.v1

16. V. Giakoumakis, J.M. Vanherpe, Linear time recognition and optimization for weak bisplit graphs, bi—cographs and bipartite P_6-free graphs, International Journal of Foundation of Computer Science, Vol. 14, No. 1,107-136(2003).
17. V.V. Lozin, E-free bipartite graphs, Discrete Anal. Oper. Res. Ser. 1 7(1) (2000) 49–66 (in Russian).
18. X. Chakraborty, A Kumar. & G. Tomar, A survey on bipartite graph based recommender systems. Information, Processing & Management, 58(6), 102536. (2021).
19. Y. Zhou and J.-K. Hao. Tabu search with graph reduction for finding maximum balanced bicliques in bipartite graphs. Engineering Applications of Artificial Intelligence, 77:86 – 97, 2019.