Linear Time Recognition of Bipartite $Star_{123}$-Free Graphs

Ruzayn Quaddoura
Faculty of Science and Information Technology, Zarqa Private University, Jordan

Abstract: *In this paper, we present a linear time recognition algorithm for recognizing bipartite graphs without induced subgraphs isomorph to $star_{123}$. Bipartite $star_{123}$-free graphs are a natural generalization of both weak bisplit and $Star_{123}$, Sun_4-free bipartite graphs, both further generalizing bicographs.*

Keywords: *Bipartite graph, decomposition of graphs, complexity.*

Received October 17, 2004; accepted July 252005

1. Introduction

Bipartite graphs underlie suitable models for such a broad spectrum of real-life problems, that adapting results on general graphs to the bipartite case and even attacking combinatorial problems separately for bipartite graphs has been a sufficiently motivated activity since the early years of graph theory; see for example [12] where a bipartite translation" for various graph concepts is given.

This is also the case for the definition and characterization of many special graph classes; for instance, Frost *et al.* in [5] propose several bipartite analogues of split graphs. Giakoumakis and Vanherpe defined in [6] the class of bicographs as a bipartite equivalent of cographs and showed that the bicographs are exactly the class of $Star_{123}$, Sun_4, P_7-free graphs.

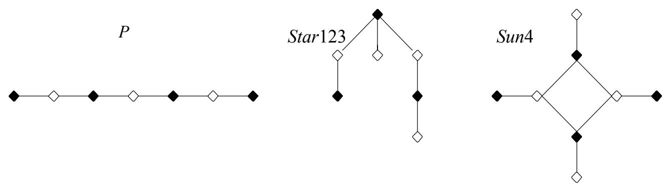

Figure 1. The forbidden configurations for the bicographs.

Fouquet *et al.* in [4] defined the weak bisplit graphs that turned out to be a generalization of bicographs. More precisely, in [4] a general decomposition scheme for bipartite graphs (called canonical decomposition) is given, under which the weak-bisplit graphs are totally decomposable; it is finally shown that these graphs are exactly the bipartite graphs with no induced $Star_{123}$ nor P_7. Another generalization of bicographs, the $Star_{123}$, Sun_4-free bipartite graphs, have been studied in [10], but no recognition algorithm is given.

In this paper, we present a linear time algorithm for a further generalization of both weak-bisplit and $Star_{123}$, Sun_4-free bipartites, namely, the $Star_{123}$-free bipartite graphs. To this end, our algorithm extends the recognition algorithm for $Star_{123}$, P_7-free bipartite graphs given in [7], by making use of some structural properties of the $Star_{123}$-free bipartite graphs, first discussed by Lozin in [9]. For simplicity and abbreviation, we will present all the theorems without proofs, the reader can find these proofs in [13].

The paper is organized as follows. In section 2, we are giving the basic concepts and notations to be used throughout this paper. In section 3, we define the extended canonical decomposition for bipartite graphs and we present Lozin's theorem on the structure of the $Star_{123}$-free bipartite graphs. In section 4, the main ideas of the recognition algorithm are presented in the form of procedures and necessary conditions concerning the decomposition tree. In section 5, the final algorithm is given and it is shown that, using suitable data structures, its execution time is linear on the input size. Section 6 concludes the paper.

2. Notation and Terminology

For terms not defined in this paper the reader can refer to [1]. The graphs considered in this paper are finite, without multiple edges or loops. As usual, for any graph G, we denote by $V(G)$ the set of its vertices and by $E(G)$ the set of its edges (or simply by V and E if there is no risk of confusion) by n and m their respective cardinalities. A bipartite graph $G = (B \cup W, E)$ is defined by two disjoint vertex subsets B - the *black* vertices and W - the *white* ones, and a set of edges $E \subseteq B \times W$.

If the color classes B or W are both non empty the graph will be called *bichromatic*, *monochromatic* otherwise. The *bicomplement* of a bipartite graph $G = (B \cup W, E)$ is the bipartite graph defined by $\overline{G}^{bip} = (B \cup W, B \times W - E)$. For a vertex x, the set of its neighbors in G is denoted by $N(x)$, the cardinality of $N(x)$ is called the degree of x in G and is denoted by $d_G(x)$ (or simply $d(x)$ if no confusion can arise). A chordless chain on k vertices is denoted by P_k and a chordless cycle on k vertices is denoted by C_k. Given a subset X of the vertex set $V(G)$, the subgraph induced by X will be denoted by $G[X]$. A graph G is called Z-free where Z is a set of graphs, when G does not contain un induced subgraph isomorphic to a graph in Z.

Let $G = (V, E)$ be a graph. A (non-empty, proper) subset M of V is a (proper) *module* of G if every vertex in $V-M$ is either adjacent to all vertices in M or to none of them. A module of G is said to be strong if it does not overlap with any other module.

Modular decomposition is a form of decomposition of G that associates with G a unique decomposition tree whose leaves are the vertices of G while the set of leaves associated to the subtree rooted on an internal node corresponds to the strong modules of G. An internal node is labeled P (resp. S, N) for parallel (resp. series, neighbourhood) modules. The module corresponding to a P node induces in G a non connected graph, that of a S node induces a connected graph whose complement is non connected and that of a N node induces a connected graph whose complement is also connected.

Given an internal node of the modular decomposition tree of G, say α, we denote by $M(\alpha)$ the corresponding strong module of G. The representative graph of α is the subgraph of G induced by the set of vertices containing precisely one vertex from each proper maximal strong module of $G[M(\alpha)]$. When G is prime with respect to modular decomposition the representative graph of G will be denoted G^*.

3. Extended Canonical Decomposition of Bipartite Graphs

Definition 1: Given a bipartite graph $G = (B \cup W, E)$ of order at least 2, G is a K + S graph if and only if G contains an isolated vertex or its vertex set can be decomposed into two sets K and S such that K induces a complete bipartite graph while S is a stable set [4].

Property 1: Let $G = (B \cup W, E)$ be a bipartite graph of order at least 2. G is a K + S graph if and only if there exists a partition of its vertex set into two non empty classes V_1 and V_2 such that all possible edges exist between the black vertices of V_1 and the white vertices of V_2 while there is no edge connecting a white vertex of V_1 with a black vertex of V_2 [4].

Such a partition will be referred as an associated partition of G and will be denoted by the ordered pair (V_1, V_2).

Theorem 1 below provides a decomposition scheme for K + S graph.

Theorem 1: A bipartite graph G is a K + S graph if and only if G admits a unique (up to isomorphism) partition of its vertex set into non empty sets $(V_1, ..., V_k)$ satisfying the following conditions [4]:

1. $\forall$ i = 1, ..., k − 1, $(V_1 \cup ... \cup V_i, V_{i+1} \cup ... \cup V_k)$ is an associated partition to the graph G.
2. $\forall$ i = 1, ..., k, G [V_i] is not a K + S graph.

The partition $(V_1,...,V_k)$ of the above theorem will be called K + S decomposition while a set Vi said to be a K + S-component of the graph.

From K + S decomposition together with the decomposition of a bipartite graph G into its connected components (parallel decomposition) or those of $\overline{G}^{bip}$ (series decomposition in the bipartite sense) yield a new decomposition scheme for G, called canonical decomposition.

It is shown in [4] that whatever the order in which the decomposition operators are applied (K + S decomposition, series decomposition or parallel decomposition), a unique set of indecomposable (or prime) graphs with respect to canonical decomposition is obtained. Obviously, a tree is associated to this decomposition. The nodes of this tree are subsets of the vertex set. The internal nodes are labeled by the type of decomposition applied, while its leaves correspond to indecomposable graphs.

A bipartite graph that is indecomposable with respect to canonical decomposition can be decomposed using modular decomposition. In this case, since proper modules of a (connected) bipartite graph are all monochromatic, the root of the modular decomposition tree has label N while its non trivial children consist of monochromatic stable sets and will be labeled M.

In the following, the decomposition process consisting of applying canonical decomposition followed by modular decomposition on a bipartite graph will be denoted extended canonical decomposition; this process is associated to a unique decomposition tree, namely the extended canonical decomposition tree.

By convention, the set of vertices corresponding to the set of leaves having an internal node α as their least common ancestor will be denoted simply by α. Thus the graph that is induced by the set of vertices α is denoted by G [α].

Following is a description of the Extended Canonical Decomposition Tree:

Let G be a bipartite graph and T its extended canonical decomposition tree.

1. There are 5 different types of internal nodes: Parallel (labeled 0), Series (labeled 1), K + S (labeled 2), Neighborhood (labeled N) or Monochromatic (labeled M). The leaves correspond to the vertices of G. An internal node having label i, i $\in$ {0, 1, 2, N, M} will be called a i-node.
2. Two internal nodes having the same label cannot be neighbors.

3. The father of a M-node has label N, the father of a i-node, i ∈{0, 1, 2, N} has label j, j ∈{0, 1, 2}.
4. If G is bi-chromatic, an internal node whose label is distinct from M is bichromatic.
5. The children of a 2-node are ordered following the order given by the K + S decomposition. (Theorem 1).
6. Internal nodes labeled 0 or 1 cannot have a leaf as child. Such node would contain either a universal or an isolated vertex.
7. Let δ be a 2-node, if its father is labeled 0 then:
   a. The first $K + S$ component of δ cannot be a white leaf.
   b. The last $K + S$-component of δ cannot be a black leaf.

   Otherwise the father of δ would have an isolated vertex and would induce a $K + S$ graph.
8. Let δ be a 2-node, if its father is labeled 1 then:
   a. The first $K + S$-component of δ cannot be a black leaf.
   b. The last $K + S$-component of δ cannot be a white leaf.

In [9] V.V Lozin gives the following characterization for bipartite *Star*$_{123}$-free graphs.

Theorem 2: Let G be a bipartite Star$_{123}$-free graph. One of the following hold:

1. G is K + S graph.
2. G and $\overline{G}$ bip aren't both connected.
3. G* or the bicomplement of G* is a path P_k or a cycle C_k with k ≥ 7.

Thus the following Corollary is immediate.

Corollary 1: A bipartite Star$_{123}$-free graph G is prime with respect to canonical decomposition if and only G* or the bicomplement of G* is a path P_k or a cycle C_k and k ≥ 7.

4. Bipartite *Star*$_{123}$-Free Graphs Recognition

A bipartite graph is *Star*$_{123}$-free if and only if its indecomposable subgraphs with respect to extended canonical decomposition are either single vertices or P_k s or C_k or their bicomplement. Thus our recognition algorithm constructs an extended canonical decomposition tree whose prime graphs verify this condition.

As a matter of fact, our algorithm extends the recognition algorithm for weak-bisplit graphs that was presented in [7]. This construction is incremental modulo a preprocessing on the vertices of the graph and follows the ideas developed by Corneil, Perl and Stewart in [2] for cograph recognition.

The main step of our algorithm accepts as input an extended canonical decomposition tree T of a bipartite *Star*$_{123}$-free graph $G = (B \cup W, E)$, a new vertex x and a set of edges E_x connecting x to some vertices of $B \cup W$. We assume that the vertex x is of maximum degree in G. The algorithm first takes into account the adjacencies of x with respect to the vertices of G by using a marking process on T described below. Next, it produces the extended canonical decomposition tree of $G' = (B \cup W \cup \{x\}, E \cup E_x)$ if G' is *Star*$_{123}$-free graph or stops otherwise.

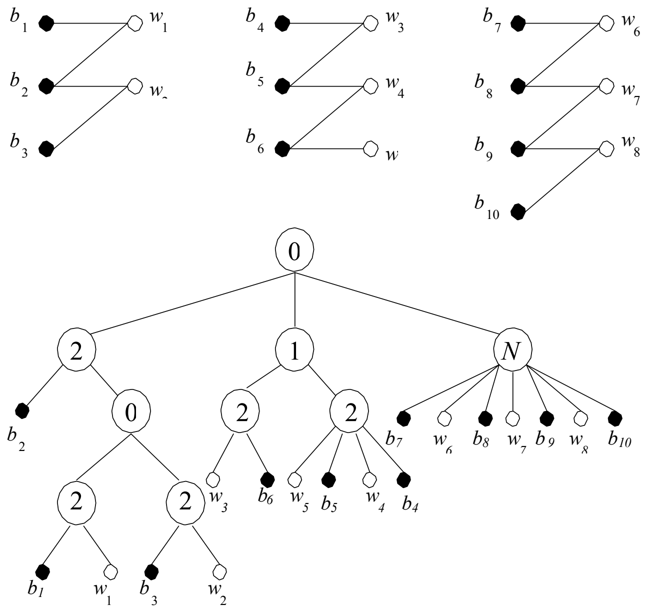

Figure 2. A bipartite graph *Star*$_{123}$-free and its extended canonical decomposition tree.

Henceforth, G, T, x, E_x denote the inputs of the algorithm while T' denotes the extended canonical decomposition tree of the bipartite graph $G' = (B \cup W \cup \{x\}, E \cup E_x)$. We may assume that G is bi-chromatic, since otherwise G could not contain a *Star*$_{123}$. We also assume w.l.o.g. that x is a white vertex.

```
Algorithm Mark
Input: The decomposition tree T of G and the vertex x
Output: The marked tree T.
  Mark and unmark the leaves of T that are neighbors
    of x;
  Let α be a node on a bottom-up traversal of T:

  If there is a child of α that is marked and unmarked
    then mark α
  End If;
  If all of the children of α that are not reduced to
    white leaves are marked and unmarked then
    unmark α
  End If;
```

Algorithm Mark

Input: The decomposition tree T of G and the vertex x
Output: The marked tree T

```
    Mark and unmark the leaves of T that are neighbors
        of x;
    Let α be a node on a bottom-up traversal of T:
    If there is a child of α that is marked and unmarked
        then mark α
    End If;
    If all of the children of α that are not reduced to
        white leaves are marked and unmarked then
        unmark α
    End If;
```

Once the tree *T* has been marked, there is at most three different possible states for a node of *T*. A node of *T* is either marked (denoted by *) or not marked (denoted by ()) or marked and unmarked (denoted (*)). Moreover the black vertices of an internal marked and unmarked node of *T* are all adjacent to *x*, which is not the case of marked or not marked nodes. The vertex *x* has both neighbors and non-neighbors among the vertices belonging to a marked node. Finally, a leaf of *T* is either not marked or marked and unmarked.

We may assume henceforth that the marking process on *T* is done and that the set of marked nodes is not empty, otherwise the black vertices of *G* would be uniform with respect to *x* and the construction of *T* would be obvious.

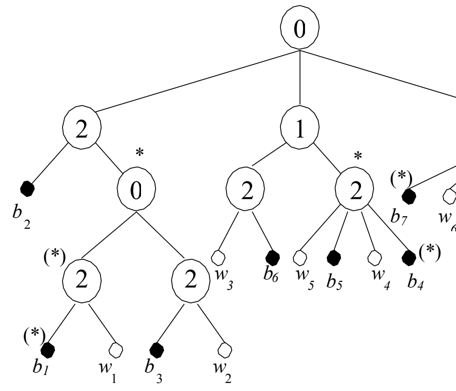

Figure 3. The marked tree for the illustrated graph in the Figure 2 when x is adjacent to b_1, b_4 and b_7.

4.1 Necessary Conditions

If *G'* is a bipartite *Star*$_{123}$-free graph, *G'* must verify some necessary conditions. We assume in this section that *G'* is always a bipartite *Star*$_{123}$-free graph.

Definition 2: Let α be an internal marked node of T. The node α is said to be a well-marked path of type 1 (resp. of type 2) and of length k if there is a partition of the vertex set of α into monochromatic sets, namely (V_1, V_2, ..., V_k) such that:

1. The edge set of the subgraph induced by α is $\bigcup_{i=1}^{k-1} V_i \times V_{i-1}$.
2. V_1 (resp. $V_1 \cup V_k$) is the set of neighbours of x in α.

Theorem 3: Let α be an internal marked N-node of T. Let G* [α] be the representative graph of G [α] and let {v_1, v_2,..., v_k} be the vertex set of G* [α]. One of the following holds:

1. M (v_i) ∪ {x} is a module of G [α ∪ {x}] for some I ∈2 {1, 2, ..., k}.
2. α is a well-marked path or a bicomplement of a well marked path of type 1 or 2 and of length k ≥ 7.

Definition 3: An internal marked nodeα is called a lowest marked node if every descendant of α is not a marked node i. e., if every descendant of α is either marced and unmarced or unmarked.

Corollary 2: Let α be an internal marked node. Then the following conditions hold:

1. α cannot be labeled M.
2. If α is a N-node then α is a lowest marked node.

Theorem 4: Let α and α' be two internal marked nodes of T. One of the following conditions holds:

1. One of these two nodes is an ancestor of the other.
2. Let β be the least common ancestor to α and α', let δ (resp. δ') be the child of β that contains α' (resp. α'). Then:
   a. β has label 0 or 1.
   b. When β has label 0 (resp. 1), x is independent of (resp. total for) all the children of β except δ and δ.
   c. Let γ be an ancestor to β and ς be the child of γ that contains β then:
      1. If γ is labeled 0 (resp. 1) then x is independent of (resp. total for) all the children of γ except ς.
      2. If γ is labeled 2 then x is independent of (resp. total for) all the children of γ that succeed (resp. precede) ς.

Corollary 3: There are at most two lowest marked nodes (say α and α'). Morever the least common ancestor to α' and α' is labeled 0 or 1.

For example, since there is three lowest marked nodes in the marked tree of Figure 3, the graph *G* ∪ {*x*} is not a *Star*$_{123}$-free; while if *x* is not adjacent to one of b_1, b_4, b_7 then the obtained graph is a *Star*$_{123}$-free since there is only two lowest marked nodes and their least common ancestor is a 0-node.

The necessary conditions of Corollary 3 have to be checked. The below Algorithm determines the set *S* of lowest marked nodes whose size must be 1 or 2 within *O* (*d* (*x*)) time complexity. As a matter of fact this algorithm is inspired from [2] with some simple change.

Algorithm Find-lowest-marked-nodes
Input: *T* The extended canonical decomposition tree of *G* marked by *x* and the set *M* of marked nodes (*M* ≠ *Ø*)

Output: The set S of lowest marked nodes in case of success, otherwise the message "failure".

```
Let $\alpha_1$ be an element of M. Mark $\alpha_1$ and all of its ancestors as "visited"
While $M \neq \emptyset$ DO
  Let $\gamma$ be an element of M. $M \leftarrow M - \{\gamma\}$
  Let $\gamma'$ be the least ancestor visited of $\gamma$
  ($\gamma$ and all of its ancestors are now marked as "visited")
  If $\alpha_2$ has not been found then
    If $\gamma' = \alpha_1$ then $\alpha_1 \leftarrow \gamma$
    Else ($\alpha_2$ is found for the first time) $\alpha_2 \leftarrow \gamma$
    End If
  Else ($\alpha_2$ has been found)
    If ($\gamma' = \alpha_1$ or $\gamma' = \alpha_2$) then
      $\alpha_1 \leftarrow \gamma$ or $\alpha_2 \leftarrow \gamma$ according to
      whether $\gamma' = \alpha_1$ or $\gamma' = \alpha_2$
    Else
      Exit with the message "failure"
    End If
  End If
End While
$S \leftarrow \{\alpha_1\}$
If $\alpha_2$ has been found then
$S \leftarrow S \cup \{\alpha_2\}$
End If
```

When the above Algorithm terminates with two lowest marked nodes α_1 and α_2 it is easy to check whether their least common ancestor has label 0 or 1. In this case we denote by α_1, α_2 of these two lowest marked nodes and by β_0 their least common ancestor. We denote also by α to one of the two nodes α_1 and α_2. At this point, we introduce some new notations in order to simplify the presentation.

Notation 1:

1. Let δ, δ' be two internal nodes such that δ is an ancestor of δ', we denote *child* (δ, δ') the (unique) child of δ containing δ'.
2. Let δ be an internal node of T chosen among α or one of its ancestors that is distinct from β_0. We denote by *children* $^{(*)}$ (δ) (resp. *children*^{0} (δ)) the set of marked and unmarked (resp. not marked) children of δ.
3. When δ has label 2, considering the ordering of the children of δ, we denote by *children*$_1^{(*)}$ (δ) (resp. *children*$_2^{(*)}$ (δ)) the set of marked and unmarked children of δ that precede (resp. succeed) *child* (δ, α) and we denote by *children*$_1^0$ (δ)(resp. *children*$_2^0$ (δ)) the set of not marked children of δ that precede (resp. succeed) *child (δ, α)*.

We remark that the behavior with respect to x of the children of β_0 (if exist) that are distinct from *child* (β_0, α) is not compatible with the label of β_0. Every ancestor to α satisfying this property is called misconfigured. More precisely:

Definition 4: Let δ be an ancestor of α. Then δ is said to be misconfigured if it belongs to one of the following cases:

- Is a 0-node and ($\delta = \beta_0$ or *children* $^{(*)}$ (δ) ≠ Ø).
- δ is a 1-node and (δ = β0 or *children* ^{0} (δ) ≠ Ø).
- δ is a 2-node, and *children*$_1$ ^{0} (δ) ≠Ø (in that case δ is said to be misconfigured before α), or *children*$_2$ $^{(*)}$ (δ) ≠ Ø (in that case δis said to be misconfigured after α).

Remark that if β_0 exist then by Theorem 4, β_0 is the highest misconfigured ancestor of α and α'. We denote henceforth by β the highest misconfigured ancestor of α.

Proposition 1: Let δ be a 2-misconfigured node after (resp. before) α. The set *children*$_2^{(*)}$ (δ) (resp. *children*$_1$ ^{0} (δ)) is reduced to a set of consecutive black leaves located just after (resp. before) α.

Proposition 2: Let β be a 2 misconfigured node after (resp. before) α and let γ be the father of α. Then the following conditions hold:

1. Any node δ on the path between β and γ is either labeled 0 (resp. 1) or 2, and when γ is not misconfigured, γ is labeled 0 (resp. 1) or 2.
2. For every 2-node δ on the path between β and α, *children* (δ, α) is the last child (resp. the first child) of δ.
3. There is no misconfigured node on the path between β and γ.
4. If γ is a misconfigured node then γ is labeled 1 (resp. 0) or 2, and when γ is labeled 2, γ is a misconfigured node before (resp. after) α.

Corollary 4: If β is a 2-misconfigured node before and after α then β is the father of α.

Proposition 3: Assume that β is a 0-node (resp. a 1-node). Let γ be the father of α. The following conditions hold:

1. γ is labeled 1 (resp. 0) or is labeled 2 such that α is the last child (resp. the first child) of γ.
2. When γ is a 1-node (resp. a 0-node), *children*$^{(*)}$ (γ) = Ø (resp. *children*^{0} (γ) = Ø).
3. When γ is a 2-node, *children*$_1^0$(γ) = Ø (resp. *children*$_2^0$ (γ) = Ø).

By Propositions 2 and 3, there is at most two misconfigured ancestors of α that are β and the father of α. We denote henceforth by γ the father of α.

4.2. Construction of T'

In order to construct *T'*, we distinguish the different cases that may arise according to the existence of β and to the labels of the nodes α and β. We distinguish two cases, first, no ancestor of α is misconfigured and second, there is a misconfigured ancestor of α. For each case we will present the necessary and sufficient conditions to be *G'* Star_{123}-free graph. The constructions of *T'* are exactly the proof for the sufficient conditions. As we refer already, the reader can find the remaining proofs in [13].

4.2.1. No Ancestor of α is Misconfigured

In this case, the construction of *T'* from T will be reduced to modifications at the subtree rooted on α.

Case 1 α is labeled 0 or 1.

The construction of *T'* when α is labeled 0 is described in Figure 4. In a symmetrical manner, we construct *T'* when α is labeled 1.

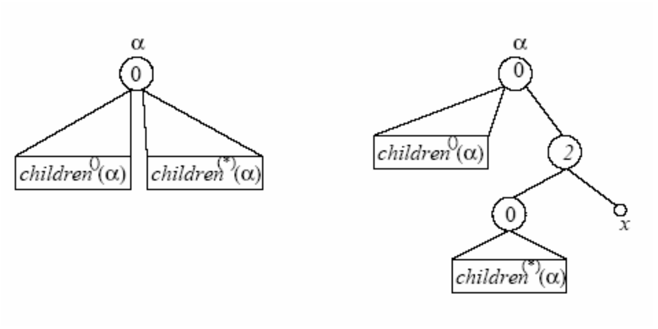

Figure 4. Construction of *T'* when α has label 0, case 1.

Case 2 α is labeled N.

Let $G^*[\alpha]$ be the representative graph of $G[\alpha]$ and let $\{v_1, v_2, ..., v_k\}$ be the vertex set of $G^*[\alpha]$. By Theorem 3, if $M(v_i) \cup \{x\}$ is a module of $G[\alpha \cup \{x\}]$ for some $i \in \{1, 2, ..., k\}$, then *x* will be inserted as a new child of the *M*-node that represents the module $M(v_i)$. If α is a well-marked path or a bicomplement of a well-marked path of type 1 or 2, then *x* will be inserted as a new child of α.

Case 3. α is labeled 2.

Recall that the children of a 2-node are ordered and the adjacencies concerning the vertices of different children are given by this order (see Theorem 1). We define four subsets of consecutive children of α, namely $A^{(*)}$, B^0, $C^{(*)}$ and *D*, as follows:

- $A^{(*)}$ contains the first consecutive children of α that are either a set of white leaves or total for *x*.
- B^0 denotes the set of the first consecutive children of α that are not members of $A^{(*)}$ and that are either a set of white leaves or independent of *x*.
- $C^{(*)}$ denotes the set of the first consecutive children of α that are not members of B^0 nor of $A^{(*)}$ and that they are either a set of white leaves or total for *x*. *D* denotes the set of the remaining children of α.

Theorem 4: G' is a bipartite Star_{123}-free graph if and only if D is independent of x and one of the following conditions holds:

1. $C^{(*)}$ is empty.
2. Induces a complete bipartite or a monochromatic graph.
3. $C^{(*)}$ induces a complete bipartite or a monochromatic graph.

The construction of *T'* for this case is as follows: If $C^{(*)}$ is empty, then *x* will be obviously inserted as a new child of α. Figure 5-a (resp. 5-b) shows the construction of *T'* when condition 2 (resp. condition 3) holds. If B^0 or $C^{(*)}$ is a monochromatic graph then $W_\alpha = \emptyset$.

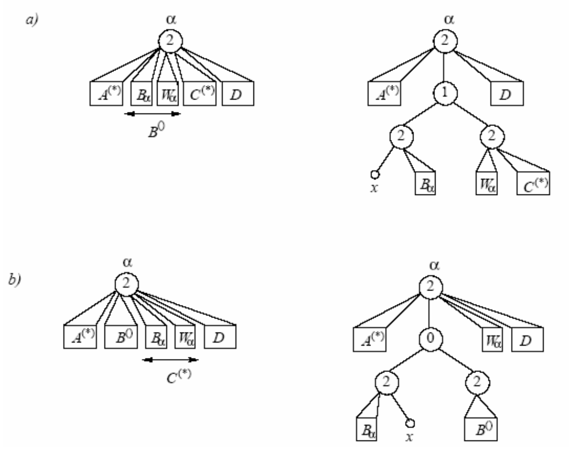

Figure 5. Construction of T' when α has label 2, case 3.

4.2.2. There Exist Misconfigured Ancestors of α

Recall that β is the highest misconfigured ancestor of α, γ is the father of α and S is the set of the lowest marked nodes of T whose size is 1 or 2, and that in the last case, β is labeled 0 or 1.

Case 1. β is labeled 0.

Theorem 5: *G'* is a bipartite Star_{123}-free graph iff the following conditions hold:

1. If S contains one element then the set $children^{(*)}(\beta)$ induces a complete bipartite graph.
2. Either there is $\alpha \in$ S such that $child(\beta, \alpha)$ be a well-marked path of type 1 and of length k, $3 \leq k \neq 6$, or there is $\alpha \in$ S such that $child(\beta, \alpha)$ be a bicomplement of a well-marked path of type 1 and of length 6.

Remark that since $children^{(*)}(\beta)$ induces a complete bipartite $children^{(*)}(\beta)$ is a well-marked path of type 1 and of length 2. Figure 6-a shows the construction of *T* when S = $\{\alpha_1, \alpha_2\}$ and condition 2 holds, or when S =

$\{\alpha\}$ and *child* (β, α) is a well-marked path of type 1 and of length $6 \neq k \geq 4$ or even when *child* (β, α) is a bicomplement of a well-marked path of type 1 and of length 6. In this figure, α represents *child* (β, α_1) and δ' represents*child* (β, α_2) *children*$^{(*)}(\beta)$. For abbreviation, the children of δ and δ' are represented as a list of alternate sets of black vertices and white vertices.

Figure 6-b shows the construction of T' when $S = \{\alpha\}$ and α is a well-marked path of type 1 and of length 3.

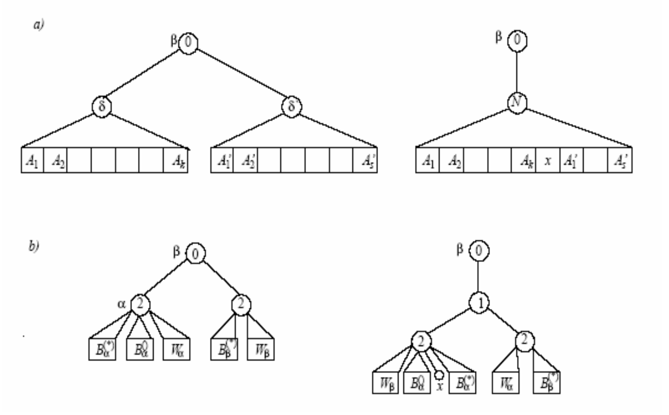

Figure 6. Construction of T' when β has label 0.

Case 2 β is labeled 1.

Theorem 6: G' is a bipartite Star$_{123}$-free graph iff the following conditions hold:

1. If S has one element, then the set *children*^{0} (β) induces an edgeless graph.
2. There is $\alpha \in S$ such that
   a. *Child* (β, α) is a bicomplement of a well-marked path of type 1 and of length k, $3 \leq k \neq 6$, or
   b. *Child* (β, α) is a well-marked path of type 1 and of length 6, or
   c. *Child* (β, α) has exactly two children, both of them inducing complete bipartite graphs.

The construction of T' when condition (a). or condition (b). holds is analogous to the construction of T' in Theorem 8. Also Figure 6-a shows the construction of T' when $S = \{\alpha_1, \alpha_2\}$ and condition C holds. Figure 7 describes the construction of T' when $S = \{\alpha\}$ and condition c holds (the set *children*^{0} (β) that induces an edgeless graph is denoted by W^0_β, B^0_β). In Figure 7-a *child* (β, α) is the node α, the set *children*$^{(*)}$ (β) (resp. *children*^{0} (α)) is denoted by $W_\alpha{}^{(*)}$, $B_\alpha{}^{(*)}$ (resp. $W_\alpha{}^0$, $B_\alpha{}^0$). In Figure 7-b *children* (β, α) is the node γ. The child that is distinct from α and (by Proposition 3) is total for x, is denoted here by $B^{(*)}, W^{(*)}$.

Case 3: β is labeled 2.

In this case the size of S is one.

Case 3.1: γ is misconfigured.

Theorem 7: Assume that β is 2-misconfigured node after α. G' is a bipartite Star$_{123}$-free graph iff one of the following holds:

1. γ is a 1-node and *children*^{0} $(\gamma) \cup \alpha$ is a bicomplement of a well-marked path of type 1 and of length 6.
2. γ is a 2-node and *children*$_1{}^0(\gamma) \cup \alpha$ is a well-marked path of type 1 and of length 5.

Since *children*$_2{}^{(*)}(\beta)$ is reduced to a set of consecutive black leaves just after *child*(β, α), we can consider the set *children*$_2{}^{(*)}(\beta)$as a well-marked path of type 1 and of length 1. Figure 6-a also shows the construction of T' when condition 1 or condition 2 holds: here, β is labeled 2, δ represents the node γ and δ' is the set *children*$_2{}^{(*)}(\beta)$. The node β here is not necessarily the father of δ.

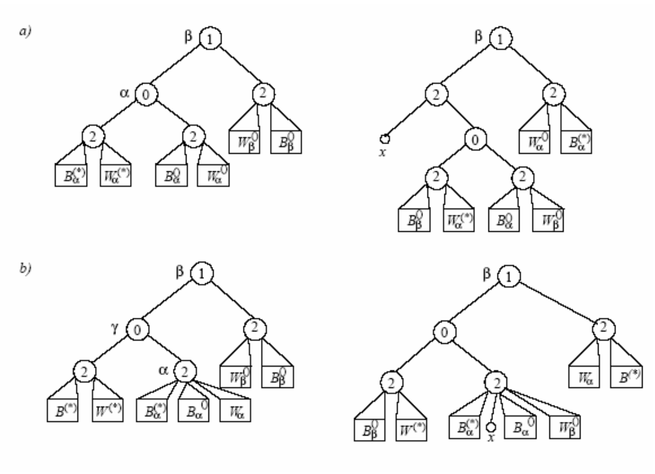

Figure 7. Construction of T when β has label 1, $S = \{\alpha\}$ and condition *c* of Theorem 6 holds.

Theorem 8: Assume that β is a 2-misconfigured node before α. G' is a bipartite Star$_{123}$-free graph iff one of the following holds:

1. γ is a 0-node and
   a. $Children^0(\gamma) \cup \alpha$ is a well-marked path of type 1 and of length 6, or
   b. γ has two children, both of them inducing complete bipartite graphs.
   c. γ is a 2-node and $children_2{}^{(*)}(\gamma) \cup \alpha$ is abicomplement of a well-marked path of type 1 and of length 5.

The construction of T' when condition 1-a or condition 2 holds is analogous to the construction of T' in Theorem 7. Figure 8 describes the construction of T' when condition 1.b holds.

Case 3.2 γ is not misconfigured.

Let's now consider the sets $A^{(*)}$, B^0, $C^{(*)}$, and D previously defined.

Theorem 9: Assume that β is misconfigured after α. G' is a bipartite Star$_{123}$-free graph iff one of the following holds:

1. α is a 0-node.
2. α is a N-node that induces a well-marked path of type 1 and of length $k \geq 7$.
3. α is a 2-node and $C^{(*)}$ is empty or D is empty and B^{0} induces a complete bipartite or a monochromatic graph.

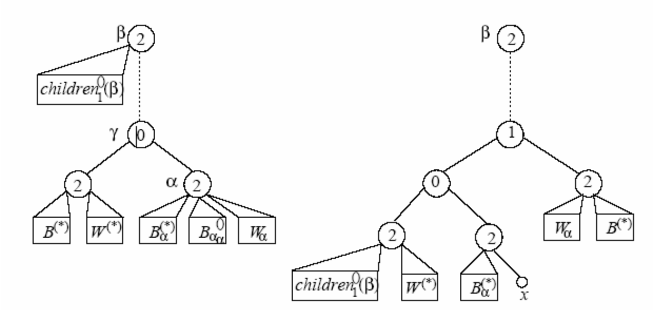

Figure 8. Construction of *T'* when β is a 2-misconfigured node before α, γ is misconfigured and condition 1.*b* oft heorem 8 holds.

Figure 9-*a* shows the construction of *T'* when α is a 0-node. In this case, if the set *children*$^{(*)}$*(α)* is a singleton, δ_1 is deleted and if this unique element of *children*$^{(*)}$*(α)* has label 2, then this element and δ_2 are gathered together. Figure 6-a shows also the construction of *T'* when α is a *N*-node; here δ' represents the set *children*$_2^{(*)}$*(β)* that is considered as a well-marked path of type 1 and of length 1 and the node δ is α. Remark that β here is not necessarily the father of δ. Figure 9-b shows the construction of *T'* when condition 3 holds with $C^{(*)}$ is empty. In this figure, if δ_2 is a singleton, then the node δ_2 is deleted and if this unique element of δ_2 has label 0, then this element and δ_1 are gathered together. Finally, Figure 9-c shows the construction of *T'* when condition 3 holds with $C^{(*)}$ is not empty; here B^{0} is denoted by B_α, W_α. If B^{0} is a monochromatic then $W_\alpha = \emptyset$.

Theorem 10: Assume that β is a 2 misconfigured before α. *G'* is a bipartite Star$_{123}$-free graph iff one of the following conditions holds:

1. α is labeled 1.
2. α is labeled 0 having exactly two children, both of them inducing complete bipartite graphs.
3. α is labeled N and is a bicomplement of a well marked path of type 1 and of length $k \geq 7$.
4. α is labeled 2 and either $C^{(*)}$ is empty, or $A^{(*)}$ is empty and $C^{(*)}$ induces a complete bipartite graph or a monochromatic graph and D is independent of x.

The construction of *T'* when condition 1, 3 or 4 holds with $C^{(*)}$ is empty is analogous to the construction of *T'* in Theorem 9 when condition 1, 2 or 3 holds with $C^{(*)}$ is empty, respectively. Figure 10-a (10-b) shows the construction of *T'* when condition 2 (4 with $C^{(*)}$ is not empty) holds.

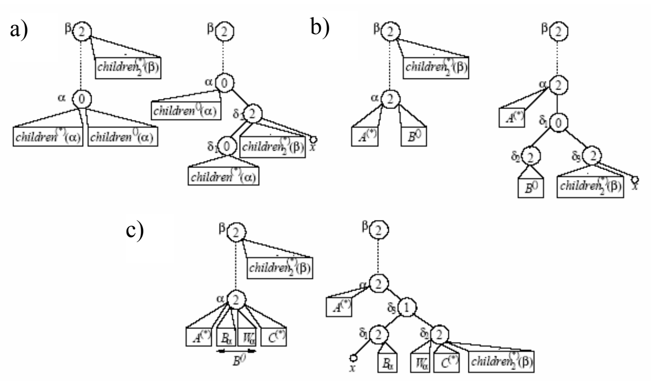

Figure 9. Construction of *T'* when β is a 2 misconfigured after α and γ is not misconfigured.

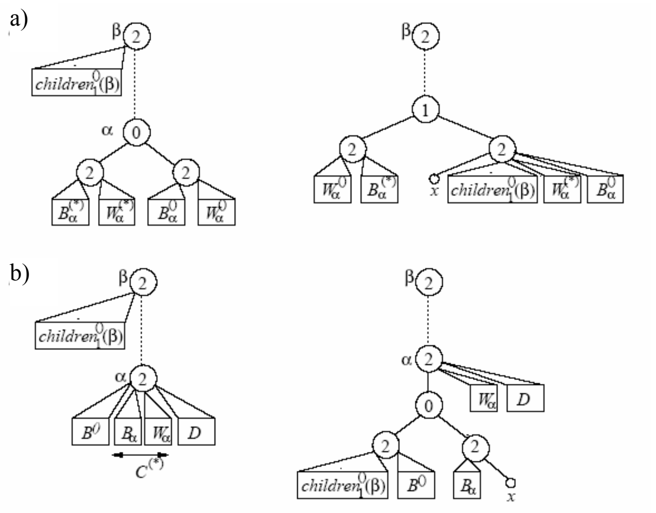

Figure 10. Construction of *T'* when β is a 2 misconfigured before α and γ is not misconfigured.

5. Complexity

The recognition algorithm for bipartite *Star*$_{123}$-free graphs can be written as follows:

```
Algorithm Star_123-free graphs recognition
Input: A bipartite graph G = (B ∪ W, E).
Output: An extended canonical decomposition tree T (G) if G is Star_123-free, otherwise failure message "G is not Star_123-free".

Initialization step: Create a list L of all the vertices of G sorted by degrees in descending order
T ← newvertex;
G' ← Ø;
Construct-tree (G', T, head (L)).

Procedure Construct-tree (G', T, head (L))
Mark (T, x)
Find-lowest-marked-nodes (T, M)
If S = Ø Then T ← insert (x, T)
(If x is an isolated (universal) vertex, then add a new root with x as left (right) child and the root of T as right (left) child)
```

Else
Find the highest misconfigured ancestor β of some $\alpha \in S$ by computing the necessary sets mentionned in

Definition 5:

If β violates the conditions of Theorem 4 then Exit with the message "failure".

Notice that there is always such a β when S has two elements.

```
If there is no such β (and in that case S has only one
    element) then
        T ← insert (x, T) (according to the Cases 1, 2, 3
            of 3.3.1)
Else
  T ← insert (x, T)
   (distinguish the Cases 1, 2, 3 of 3.3.2:
  Case 1 - according to 3.14
  Case 2 - according to 3.15
  Case 3 - check first 3.9, 3.10, then
    Subcase 3.1 - according to 3.16 and 3.17
    Subcase 3.2 - according to 3.18 and 3.19)
End If
End If
G' ←G [V (G') ∪ {x}];
If L = Ø Then exit
Else
 L ←L − {x}, x ← head (L)
 Construct-tree (G', T, x)
End If
```

Let's show now that recognition of bipartite $Star_{123}$-free graph can be done within linear time complexity on the size of G. Our goal is to show that the initialization step requires only $O(d_{G'}(x))$ where $d_{G'}(x)$ is the degree of x in G'. For this we will assume that x is of maximal degree in G'. We assume also that each node in T can access to its father and that the set of children of a node are stored as doubly linked lists. Moreover, we assume that sequences of consecutive leaves having the same color are stored as monochromatic sets, thus visiting a such sequence can be done in one step.

5.1. Marking Procedure

Obviously the marking procedure runs within $O(d_{G'}(x))$ time since at most $O(d_{G'}(x))$ nodes have been marked. We may also suppose that for each marked node on the tree, the set of its marked and unmarked children has been computed.

5.2. Finding the Lowest Marked Nodes

By Corollary 3, there are at most two lowest marked nodes. For every lowest marked node α, let b be a black vertex and w be a white vertex belonging to $G[\alpha]$. Let δ be an ancestor of α, let δ' be the father of δ (if any) and δ'' be the father of δ' (if any). The number of ancestors of α can be shown to be linear in the degree of x in the initial graph, by the following simple argument (taken from [4]): we can associate to $(\delta, \delta', \delta'')$ a private neighbor of b or w. This is obvious if either δ or δ' or δ'' is a 1-node, or if δ is a 0-node, δ' is a 2-node and δ'' is a 0-node: since δ is connected there must be a neighbor of b belonging to a graph located after child *child* (δ,α) or a neighbor of w belonging to a graph located before *child* (δ,α). The same holds when δ, δ' and δ'' are labeled 2, 0 and 2, respectively. Thus the number of ancestors of α is at most $3 \times (d_{G'}(b) + d_G(w))$ which is $O(d_{G'}(x))$ since we have assumed that x is of maximal degree in G.

5.3. Computation of all Sets Needed for the Construction of T at Every Call of Constructtree

Let's show that the computation of all sets needed for the construction of T when x is inserted requires $O(d_{G'}(x))$ times. Consider an internal node of T among α or one of its ancestors (say δ). When δ has 0 or 1 node the computation of the sets *children*$^{(*)}(\delta)$ and *children*$^{0}(\delta)$ is obvious.

When δ is a 2-node and is an ancestor of α, the computation of the sets *children*$_1^{(*)}(\delta)$, *children*$_2^{0}(\delta)$ and checking the proposition 1 can be done simultaneously: If *children*$_1^{0}(\delta)$ is not empty then this set must be a monochromatic set of black leaves located just before *child* (δ, α) while all other children of δ that are located before *child* (δ, α) are marked and unmarked. Thus, the computation of those sets for all ancestors of α requires to visit marked and unmarked nodes whose number is $O(d_{G'}(x))$. Hence testing whether δ is misconfigured or not can be done in $O(d_{G'}(x))$ time.

We now examine the computation of the sets $A^{(*)}$, B^{0}, $C^{(*)}$and D when α is a 2-node: The first set that is computed is $A^{(*)}$ by visiting the set of children of α while the visited nodes are either marked and unmarked or monochromatic set of white leaves. By Theorems 4, 9, or 10 the remaining marked and unmarked children of α must belong to $C^{(*)}$.

The computation of $C^{(*)}$ is done as follows: first pick a child of α (say c) among the remaining marked and unmarked nodes and visit the set of children of α from c in both directions using doubly linked lists, as long as the visited nodes are either marked and unmarked or a monochromatic set of white leaves. All marked and unmarked children of α must be visited following this process otherwise D is not independent of x. Once the set $C^{(*)}$ is known, the computation of $B^{0}$and D follows immediately.

We leave to the reader the task of verifying that in all cases, the insertion of x to the existing tree i.e. a call of Construct-tree(G, T, x) takes constant time.

5.4. Complexity

We will show now that our algorithm recognizes if a bipartite graph *G* is a *Star*$_{123}$-free within linear time on the size of *G*. Let us first show that when adding a vertex x in a graph *G*, we can know in O (d_G (x)) whether *x* is of maximal degree in *G*. For this, we use an additional data structure, namely an array *A* such that *A* [*i*] is the list of vertices outside *G* whose degree in *G* is *i*. Initially *A* [0] contains all vertices of *G*. the vertex *x* is chosen into the non empty list of *A* having a lowest index. When adding *x* to *G*, its neighbors that belong to *A* [*i*] move from *A* [*i*] to *A* [*i* + 1]. In order to find in constant time each neighbor *y* of *x* in *A* [*i*], we use an array *B* such that *B* [*y*] contains the address of *y* into the list *A* [*i*]. Hence, the time complexity for finding and moving the neighbors of *x* from *A* [*i*] to *A* [*i* + 1] and for updating *B* is O (d_G (x)). Since testing whether $G' = G \cup \{x\}$ is a bipartite *Star*$_{123}$-free or not can be done within O (d_G (x)) time complexity, it is clear that our recognition algorithm runs in linear time on the size of *G*.

6. Conclusion

This paper presents an optimal algorithm for recognition the bipartite *Star*$_{123}$-free graphs. We think that this study must be prolonged, since, as we signaled, bipartite graphs present in the same time, theoretically and practically interest. In this perspective, the algorithmic method, which we apply for our recognition algorithm of bipartite *Star*$_{123}$-free graphs has been proved already its effectiveness since initially proposed for the recognition of cographs and weak bisplit graphs, Therefore, we are convinced, in spite of a technique appearance where this method is presented, we can extend with success its application in the algorithmic study for other graph classes.

References

[1] Bondy J. A. and Murty U. S. R., *Graph Theory with Applications*, North Holand, New York, 1979.

[2] Corneil D. G., Perl Y., and Stewart L. K., "A Linear Recognition Algorithm for Cographs," *SIAM Journal of Computing*, vol. 14, no. 4, pp. 926-934, November 1985.

[3] Cournier A. and Habib M., "A New Linear Algorithm for Modular Decomposition," *Lecture Notes in Computer Science*, Springer, Berlin, vol. 787, pp. 68-84, 1994.

[4] Fouquet J. L., Giakoumakis V., and Vanherpe J. M., "Bipartite Graphs Totally Decomposable by Canonical Decomposition," *International Journal of Foundations of Computer Science*, vol. 10, no. 4, 1999.

[5] Frost H., Jacobson M., Kabell J., and Mc-Morris A., "Bipartite Analogues of Split Graphs and Related Topics," *ARS Combinatorial*, vol. 29, pp. 283-288, 1990.

[6] Giakoumakis V. and Vanherpe J. M., "Bi-Complement Reducible Graphs," *Advances in Applied Mathematics*, vol. 18, no. 4, 1997.

[7] Giakoumakis V. and Vanherpe J. M., "Linear Time Recognition of Weak Bisplits Graphs," *International Journal of Foundations of Computer Science*, vol. 14, no. 1, pp. 107-136, 2003.

[8] Lerchs H., "On the Cliques and Kernels," *Technical Report*, Department of Computer Science, University of Toronto, 1971.

[9] Lozin V. V., Bipartite Graphs Without a Skew Star, *RUTCOR Research Report* 20-2001, March 2001.

[10] Lozin V. V., "On a Generalization of Bicomplement Reducible Graphs," *Lecture Notes in Computer Science*, Springer, Berlin, vol. 1893 , pp. 528-538, 2000.

[11] McConnel R. M. and Spinrad J., "Linear Time Modular Decomposition and Efficient Transitive Orientation of Comparability Graphs," *Technical Report*, Department of Computer Science, University of Colorado, 1993.

[12] McKee T. A., "Bipartite Analogs of Graph Theory," *Congressus Numerantium*, vol. 60, pp. 261-268, 1987.

[13] Quaddoura R. and Vanhaerpe J. M., "Linear time Recognition of Bipartite Star$_{123}$-free Graphs," *Technical Report*, LaRIA 2002-12, 2002.

Ruzayn Quaddoura received his MSc in computer science from Institute National Polytechnique de Grenoble (INPG), France, and his PhD in computer science from Picardie Jules Verne University, France, in 1997 and 2003, respectively. Currently, he is an assistant professor at the Department of Computer Science, Zarqa Private University, Jordan. His research interests include algorithmic, combinatorial optimization, and graph theory.