Linear time recognition algorithms for three variants of vertex series parallel digraphs

Ruzayn Quaddoura
Computer Science Department, Zarqa Private University, Jordan

Abstract *Let S and P be the set of sources and the set of sinks of a digraph G. Tarjan and al in [1] defined the family of vertex series parallel dags (directed asyclic graphs) that is obtained from one vertex by the parallel operation (disjoint union) and the series operation (disjoint union of two dags G_1 and G_2 with adding the arcs of $P_1 \times S_2$. We show in this paper that if we consider the multiplication in the series operation as $S_1 \times P_2$, $P_1 \times P_2$ or $S_1 \times S_2$ then the obtained classes of dags are recognizable in linear time.*

Keywords: *dag's, Bipartite graphs, Complexity*

1 Introduction

The class of series parallel dags (directed asyclic graphs) defined by Tarjan and al in [1] in terms of the subclass of its minimal members. The dags in this subclass are called minimal vertex series parallel (MV SP) and are defined recursively as follows :

1) The dags having a single vertex is MVSP.

2) If G_1 and G_2 are two MVSP, so are the dags constructed by each of the following operations :

i) Parallel composition : $Gp = (V_1 \cup V2, E_1 \cup E_2)$.

ii) Series composition : $Gs = (V_1 \cup V_2, E_1 \cup E_2 \cup (P_1 \times S_2))$ where P_1 is the set of skins of G_1 and S_2 is the set of sources of G_2.

A dag is vertex series parallel (VSP) if and only if its transitive reduction is MVSP. The main result in [1] is a linear time algorithm that determines whether an arbitrary dag is VSP. Series parallel dags have been extensively studied for their application to several theoretical and practical domains, especially to scheduling problems. The class of VSP dags is known as a class for which several scheduling problems are polynomially solvable ([2], [3], [4], [5]) while the same problems are NP-complete for a general graph. In this paper we present linear time recognition algorithms for two variants of the class VSP, namely, the class of series parallel bipartite dags of depth 1 (VSPB1) that is defined as the same as the class of VSP by changing the multiplication in the series operation to be $S_1 \times P_2$ where S_1 is the set of sources of G_1 and P_2 is the set of sinks of G_2. The second class is vertex series parallel dags source-source (VSPs) where the multiplication in the series operation above is $S_1 \times S_2$, S_1, S_2 are the sets of sources of G_1,G_2 respectively. An immediate result of the second algorithm is a linear time recognition for a third class of dags, namely, vertex series parallel dags sink-sink (VSPp) where the multiplication in the series operation is $P_1 \times P_2$.
The paper is oeganized as follows : In section 2 we are giving the basic concepts and notations to be used throughout this paper. In section 3 we present some implements of bipartite graphs that are useful for our recognition algorithm of the class VSPB1 dags, this algorithm is presented in section 4. Sections 5 and 6 are devoted for the recognition algorithms of the classes VSPs and VSPp respectively. Section 7 concludes the paper as a perspective of this work.

2 preliminaries

A directed graph (or a digraph for short) $G = (V,E)$ consists of a set V of vertices and a set E of ordered pairs of vertices called arcs. Let (x, y) be an arc of G then x is a predecessor of y and y is a successor of x. For a vertex x we denote by $N^+(x)$ (resp. $N^-(x)$) the set of its successors (resp. predecessors) and by $N(x)$ the union $N^+(x) \cup N^-(x)$. The positive degree $d^+(x)$ (resp. negative degree $d^-(x)$) of a vertex x is the number $|N^+(x)|$ (resp. $|N^-(x)|$), the degree of a vertex x is the number $|N(x)|$. Given a subset X of the vertex set V, the subgraph induced by X will be denoted by $G[X]$. A vertex x is called source (resp. sink) if x has not any predecessor (resp. any successor). A path of length k is a sequence of vertices x_1, x_2, ..., x_k such that (x_i, x_{i+1}) is an arc for $1 \leq i < k$. If $x_1 = x_k$ and $k \geq 2$ the path is a circuit. A directed graph without circuit is called acyclic, a directed acyclic graph is denoted by dag. A chain of length k is a sequence of vertices x_1, x_2, ..., x_k such that (x_i, x_{i+1}) or (x_{i+1}, x_i) is an arc. If $x_1 = x_k$ and $k \geq 2$ the chain is a cycle. An arc (x, y) is called transitive arc if there is a path from x to y of length at least 2. A bipartite graph $G = (B \cup W,E)$ is given by a set of black vertices B and a set of white vertices W 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 $\overline{G}^{bip} = (B \cup W, B \times W - E)$. A white (resp. black) vertex x of a bipartite graph is B-universal (resp. W-universal) if $N(x) = B$ (resp. $N(x) = W$). A graph G is called Z-free where Z is a set of graphs when G does not contain an induced subgraph isomorphic to a graph of Z.

3 Canonical decomposition of bipartite graphs

Definition 3.1 [6] A bipartite graph $G = (B \cup W,E)$ of order at least 2 is a $K+S$ graph if and only if G contain 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.

Property 3.2 [6] *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. Such a partition will be referred as an associated partition of G and will be denoted by the ordered pair* (V_1, V_2).

Property 3.3 [7] *Let $G = (B \cup W,E)$ be a bipartite graph without universal nor isolated vertex. Exactly one of the following holds*:

1) *G* is a *$K + S$ graph*.
2) *Given any partition, say (X_1,X_2) of the vertex set B* (or *W*), *there exists an induced*
$2K_2$ *of G having vertices in both classes X_1 and X_2*.

Corollary 3.4 [7] *If G is not a $K + S$ graph then any vertex of G belong to a $2K_2$ of G*

Theorem 3.5 below provides a decomposition scheme for $K + S$ graph.

Theorem 3.5 [6] *A bipartite graph G* is a *$K+S$ graph if and only if G admits a unique* (*up to isomorphism*) *partition into non empty sets of its vertex say* $(V_1, ..., V_k)$ *that verifies*:

1) $\forall\, i = 1, ..., k-1$, $(V_1 \cup \ ... \ \cup V_i, V_i{+}1 \cup \ ... \ \cup V\mathrm{k})$ *is an associated partition to the graph G.*
2) $\forall\, i = 1, ..., k$, *the induced supgraph $G[V_i]$ 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 V_i said to be a $K + S$ component of the graph.

Remark 3.6 Theorem 3.5 implies that the order of the components of a $K+S$-decomposition is significant. More precisely, let b be a black vertex belonging to a $K+S$-component, say V_i, then b is adjacent to any white vertex belonging to V_j , $j > i$, and b is independent of any white vertex belonging to V_l, $l < i$; if w denotes a vertex of V_i, we have that w is independent of any black vertex belonging to V_j , $j > i$, w is also adjacent to any black vertex belonging to V_l, $l < i$.

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 [6] 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, and a unique (up to isomorphism) tree is associated to this decomposition, called canonical decomposition tree. 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.

This decomposition process, in particular the order in which $K+S$-decomposition, parallel decomposition and series decomposition are applied, imply some properties of the decomposition tree that we list now. In the following, when no confusion is possible, we will not distinguish between nodes of the tree and their associated subgraphs. In addition the meanings of usual terms like leaf, internal node, child, father, ancestor are as expected.

Properties of the decomposition tree. Let G be a bipartie graph and T be its canonical
decomposition tree.

1) There is 3 different types of internal nodes : Parallel (labelled 0), Series (labelled 1) or $K + S$ labelled 2.

2) Two internal nodes having the same label cannot be consecutive.

3) An internal node labeled 0 or 1 cannot have a trivial leaf. Such node would have either a universal or an isolated vertex and thus would induce a K + S graph.

4) The father of a leaf is labeled 2. (Consequence of 2).

5) If G is bi-chromatic then for any 2-node say α , G[α] is also bi-chromatic. Otherwise the father of a 2 node corresponding to a monochromatic graph would have universal or isolated vertices and would induce a $K + S$ graph, a contradiction with 2.

6) The children of a 2-node are ordered following the order given by the K+S decomposition (see theorem 3.5).

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, a contradiction with 2.

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.

4 Vertex series parallel bipartite digraphs of depth 1 (VSPB1)

Definition 4.1 A dag (VSPB1) is defined recursively as follows :

1)A dag having a single vertex is VSPB1.

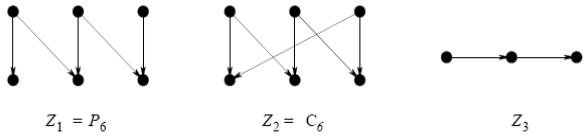
2) If G_1 and G_2 are two VSPB1, the dag constructed by each of the following operations
are also VSPB1 :

i) Parallel composition : $G_p = (V_1 \cup V_2, E_1 \cup E_2)$.
ii) Series composition : $G_s = (V_1 \cup V_2, E_1 \cup E_2 \cup (S_1 \times P_2))$ where S_1 is the set of sources of G_1 and P_2 is the set of sinks of G_2.

We observe that every vertex of a VSPB1 dag is either a source or a sink since the series operation preserve the identity of sources and sinks. Consequently every VSPB1 dag does not contain a path of length 3, consequently, every VSPB1 does not contain any transitive arcs. Thus every VSPB1 is a dag of depth 1. Moreover, we can reduce a VSPB1 to its vertex set by the inverse operations of parallel composition and series composition, that is by the parallel decomposition and series decomposition.
There is a very strong relation between the family of VSPB1 dags and the family of K+S bipartite graphs that is established in theorem 4.2. This relation is the key of our recognition algorithm for the family of VSPB1 dags.

Theorem 4.2 Let G be a dag. The following conditions are equivalent,

1) G is a VSPB1.
2) G is a bipartite graph {Z_1, Z_2, Z_3}-free.
3) G is a bipartite graph of depth 1 and every connected subgraph of G^{no} is $K + S$, where G^{no} is the graph obtained by omitting the orientation of G.



Proof 1 $\Rightarrow$ 2. A P_6 or a C_6 has neither a series nor a parallel decomposition. Thus if G is a VSPB1 dag and G contains a P_6 or a C_6, there is a step during the decomposition of G for which G cannot decompose by series or by parallel decomposition, a contradiction. As we indicated above, G does not contain the configuration Z_3. Let show that G is a bipartite graph. Let $C = x_1, x_2, ..., x_{2k+1}$ be an odd cycle in G. Assume without less of the generality that x_1 is a source of G. Then every vertex x_{2i+1} ($1 \leq i \leq k$) is also a source of G, thus x_{2k+1} is a source of G, a contradiction since x_1 and x_{2k+1} are adjacent.

2 $\Rightarrow$ 3. Assume that G is a bipartite graph {Z_1, Z_2, Z_3}-free. Then G is a bipartite graph of depth 1 and G^{no} is {P_6, C_6}-free. Let H be a subgraph of G^{no} that is non $K + S$. Then H has not an isolated or a universal vertex. By the corollary 3.4, every vertex of H belongs to a $2K_2$ of H. Let ab, cd be a $2K2$ in H. Since H is connected {P_6, C_6}-free, there is a vertex e that is adjacent to a and c or to b and d. Without less of generality assume that e is a white vertex adjacent to a and c. Since e is not universal, there is a black vertex f that is independent of e. Since H is connected there is a chain that connects the vertex f and the $P_5 = a, b, c, d, e$. This chain contains a P_6 or a C_6, a contradiction.

3 $\Rightarrow$ 1. Let $(V_1, ..., V_k)$ be the $K + S$ decomposition of G^{no}. Since G is a bipartite graph of depth 1, then by the definition of $K + S$ decomposition, $(V_1, ..., V_k)$ correspond to a series decomposition of G. If $G[V_i]$ ($1 \leq i \leq k$) is not connected then by supposition, every connected component of $G[V_i]$ is a $K + S$. Thus by considering the $K + S$ decomposition of every connected component of $G[V_i]$, we can reduce G to its vertex set by a parallel and a series decomposition, therefore G is a VSPB1. □

Corollary 4.3 The class of bipartite {P_6,C_6}-free graphs is the smallest class of graphs containing the graph reduced to one vertex and closed under the operations K+S and Parallel.

4.1 Recognition of VSPB1dags

Let G be a dag. We can assume that every vertex of G is either a source or a sink, since this step can be executed by a simple examination of the degree of each vertex. By theorem 4.2, G is a VSPB1 if and only if G^{no} is {P_6, C_6}-free. Consequently the recognition problem of VSPB1 dags

can be reduced to the recognition problem of bipartite $\{P_6,C_6\}$-free graphs. Thus let us consider that G is a bipartite graph.
From the corollary 4.3, G is $\{P_6, C_6\}$-free if and only if its canonical decomposition tree contains only 0-nodes or 2-nodes. Thus, our recognition algorithm constructs a decomposition tree whose its internal nodes are labeled 0 or 2. As a matter of fact, our algorithm is inspired from the two recognition algorithms presented already in [7] and [8] for the weak bisblit graphs and bipartite Star123-free graphs respectively. Both these two algorithms including this new our algorithm follow the ideas developed by Corneil, Perl and Stewart in [9] for cographs recognition. More precisely, given a bipartite graph, we construct a sequence of subgraphs that are all $\{P_6, C_6\}$-free graphs: starting with an empty graph, we add vertices incrementally while the subgraphs obtained remain to be $\{P_6,C_6\}$-free graphs. The initial bipartite graph is $\{P_6, C_6\}$-free graph if and only if all of its vertices have been added successfully by this process. Thus, the main step of our algorithm accepts as input a canonical decomposition tree T of a bipartite $\{P_6, C_6\}$-free graph $G = (B \cup W, E)$, a new vertex x and a set of edges E_x connecting x to some vertices of $B \cup W$. 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 decomposition tree of $G' = (B \cup W \cup \{x\}, E \cup E_x)$ if G' is $\{P_6,C_6\}$-free graph or stops otherwise.
Henceforth G, T, x, E_x denote the inputs of the algorithm while T' denotes the decomposition tree of the bipartite graph $G' = (B \cup W \cup \{x\}, E \cup E_x)$. Since a mono-chromatic bipartite graph is obviously $\{P_6,C_6\}$-free, we may assume w.l.o.g that G is bi-chromatic and that x is a white vertex.

4.1.1 The marking process

The algorithm Mark previously used in [9] takes into account the adjacencies of the vertex x with its neighbors in G by marking the nodes of T.

Algorithm 1 Mark
Input: T the decomposition tree 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
if all of the children of α that are not reduced to white leaves are marked and
unmarked then
unmark α

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, if δ is marked and unmarked then x is adjacent to all black vertices of G[δ] and if δ is marked then x is adjacent to some but not all black vertices of G[δ]. Finally if δ is a leaf then δ 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' is obvious.

Remark 4.4 By the Corollary 4.3, we will take into account only the 0-nodes and the 2-nodes since the 1-nodes are irrelevant.

If G' is $\{P_6, C_6\}$-free bipartite graph, G' must verify the following necessary condition :

Theorem 4.5 Let α and β be two internal marked nodes of T. If G' is $\{P_6,C_6\}$-free bipartite graph then one of the two nodes is an ancestor of the other.

Proof Let δ be the least common ancestor to the marked nodes α and β. We denote by α' (resp. β') the child of δ containing α (resp. β). Since G[α] and G[β] are subgraphs of G[α'] and G[β'] then α' and β' are partial with respect to x. Assume that δ is labeled 0, then G[α'] and G[β'] are connected subgraphs of G[δ]. Thus there is an induced chain b_1,w_1, b_2 in α' (resp. b'_1, w'_1, b'_2 in β') such that x is adjacent to b_1 and not to b_2 (resp. to b'_1 and not to b'_2). The set $\{w'_1, b'_1, x, b_1,w_1, b_2\}$ induces a P_6, a contradiction.
Assume that δ is labeled 2. By corollary 3.4 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 α' (resp. in β' such that x is adjacent to b_1 and not to b_2 (resp. to b'_1 and not to b'_2). Then the set $\{w_2, b_2, w'_2, b_1, x, b'_1, w'_1\}$ induces a P_6, a contradiction. □

Theorem 4.5 implies that the marked nodes are located on a unique chain that begin from the lowest marked node of T to the root. We denote henceforth by α the lowest marked node of T. We assume in the following that the condition of theorem 4.5 is verified and that the node α is known.
Let's now introduce some notations.

Notation 4.6 1) Given two internal nodes of T, say δ and δ', such that δ is an ancestor of δ', we denote by *child*(δ, δ') the unique child of δ that contains δ'.
2) Given an internal node of T, say δ chosen among α or one of its ancestors we denote by $Children^0(\delta)$ (resp.

$Children^{(*)}(\delta)$) the set of children of δ that are not marked (resp. marked and unmarked).
3) When δ has label 2 , considering the ordering of the children of δ, we denote by $Children_1^{(*)}(\delta)$ (resp. $Children_2^{(*)}(\delta)$) the set of marked and unmarked children of δ that precede (resp. succeed) $Child(\delta,\alpha)$ and we denote by $Children_1^{0}(\delta)$ (resp. $Children_2^{0}(\delta)$) the set of not marked children of δ that precede (resp. succeed) $Child(\delta,\alpha)$.

Thus an ancestor of α which is labeled 0 shares its children into at most three classes that are: $Children^{(*)}(\delta)$, $Children^{0}(\delta)$ and $Child(\delta,\alpha)$. If δ is labeled 2, its children can be decomposed into at most 5 sets that are: $Children_1^{(*)}(\delta), Children_2^{(*)}(\delta), Children_1^{0}(\delta)$, $Children_2^{0}(\delta)$ and $Child(\delta,\alpha)$, while the children of α are shared into the two non empty classes $Children^{0}(\alpha)$ and $Children^{(*)}(\alpha)$).

Definition 4.7 Let δ be an ancestor of α in *T*.

When δ is a 0-node, δ is said to be bad marked iff $Children^{(*)}(\delta) \neq \phi$.

When δ is a 2-node, δ is said to be bad marked before α iff $Children_1^{0}(\delta) \neq \phi$.

When δ is a 2-node, δ is said to be bad marked after α iff $Children_2^{(*)}(\delta) \neq \phi$.

4.1.2 Construction of *T'*

In order to construct *T'* we discuss according to the label of α and the existence of bad marked nodes. Recall that *T* has been marked by *x*, all nodes that are marked are known and lie on the path from α to the root of *T*.
When α is labeled 2 and since it is the lowest marked node we can partition its set of children in at most four subsets of consecutive children, namely $A^{(*)}, B^{0}, C^{(*)}, D$ as follows:
$A^{(*)}$ contains the first consecutive children of α that are either a set of white leaves or that are 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 $A^{(*)}$ and that are either a set of white leaves or total for *x*.
D denotes the set of remaining children of α.

Remark 4.8 By construction B^{0} and $C^{(*)}$ cannot be together monochromatic graphs.

Proposition 4.9 Assume that α is a 2-node. If G' is a bipartite {P_6, C_6}-free graph then D is independent of x and one of the following conditions is verified :
1) $C^{(*)}$ is empty.
2) $C^{(*)}$ induce a complete bipartite graph or a monochromatic graph.

Proof Suppose that *D* is not empty, otherwise we are done. Let's show that *x* is independent of *D*. Since *D* is not empty then $C^{(*)}$ and B^{0} are both non empty. Let $b_4 \in D$ such that *x* is adjacent to b_4. Now, *D* contains two adjacent vertices b_4', w_4 such that *x* is independent of b_4', otherwise $b_4 \in C^{(*)}$. Let b_2, b_3 be two black vertices of B^{0} and $C^{(*)}$ respectively. By construction, There is a white vertex *w* such that *w* is adjacent to b_2 and *w* is independent of b_3. But now the set {b_4, *x*, b_3, w_4, b_2,*w*} induces a P_6, a contradiction.
Let's show now that the condition 1 or 2 must be verified. Suppose that $C^{(*)}$ is not empty. Then B^{0} is also non empty.

Claim 1 Every element of $C^{(*)}$ is a leaf.

Proof Suppose that δ is an element of $C^{(*)}$ that is an internal node. Then δ is a 0-node, thus it contains a $2K_2$ say b_1w_1, b_2w_2. Let $b \in B^{0}$, then {*b*,w_1, b_1, *x*, b_2,w_2} induces a C_6, a contradiction. ■

Claim 2 B^{0} contains a white vertex.

Proof Suppose that B^{0} does not contain any white vertex. Then $C^{(*)}$ contains an element that is an internal node, a contradiction with claim 1. ■

Let b_2,w_2 be two adjacent vertices of B^{0}. Suppose that $C^{(*)}$ does not induce a complete bipartite graph nor a monochromatic graph. Then $C^{(*)}$ 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} induces a P_6, a contradiction.

Theorem 4.10 Assume that there is no ancestor of α bad marked. G' is a bipartite {P_6,C_6}- free graph if and only if one of the following condition is verified :

1) α is labeled 0.
2) α is labeled 2 *and the proposition 4.9 is verified.*

Proof The if part of the theorem have been proved above. We will describe the construction of T' for the only if part . The construction of T' when α is labeled 0 is described in the figure 2. If $Children^{(*)}(\alpha)$ is a singleton, the node 0 will be deleted and the element $Children^{(*)}(\alpha)$ will be a child of the node labeled 2. Suppose that α is labeled 2. If $C^{(*)}$ is empty, x will be inserted as a new child of α. The construction of T' when $C^{(*)}$ induces a complete bipartite graph or a monochromatic graph is described in the figure 3. In this case, if $C^{(*)}$ is a monochromatic graph then $W_\alpha = \phi$.

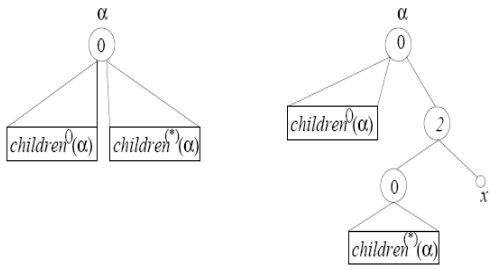


Figure 2: Construction of T' when there is no bad marked ancestor of α and α has label 0

Proposition 4.11 Assume that G' is a bipartite {P_6,C_6}-free graph. If there is a bad marked ancestor β of α then :

1) β is a 2-bad marked node after α.

2) β is the unique bad marked ancestor of α.

3) The set $Children_2^{(*)}(\beta)$ is a set of black leaves located just after $Child(\beta,\alpha)$.

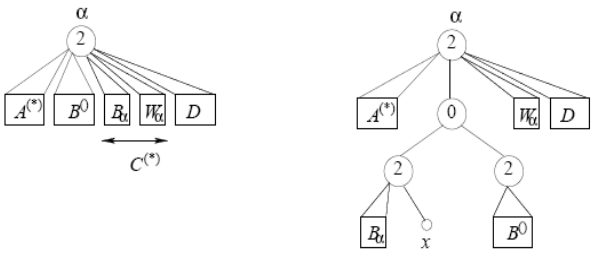


Figure 3: Construction of T' when there is no bad marked ancestor of α , α has label 2.

Proof Suppose that β is a bad marked ancestor of α of type 0. Then there exists two adjacent vertices b_3,w_3 in an element of $Children^{(*)}(\beta)$. Since $Child(\beta,\alpha)$ 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} induces a P_6, a contradiction.

If β is a 2-bad marked node before α, then $Children_1^{0}(\beta)$ contains a black vertex b independent of x. By the corollary 3.4, $Child(\beta,\alpha)$ contains an induced 2K2 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} induces a P_6, a contradiction. Consequently β is a 2-bad marked node after α. Let's consider β the highest bad marked ancestor of α and let $b \in Children_2^{(*)}(\beta)$.

Claim There is no ancestor δ of α containing a white vertex total for $Child(\delta,\alpha)$.

Proof Let δ be an ancestor of α and w is a white vertex of δ total for $Child(\delta,\alpha)$. let b_1w_1, b_2w_2 be an induced 2K2 of $Child(\delta,\alpha)$ 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} induces a P_6, a contradiction. ■

By this claim, if δ is a 2-bad marked node after α then the set $Children_2^{(*)}(\delta)$ is reduced to a set of black leaves located just after $Child(\beta,\alpha)$. Moreover, for every bad marked ancestor δ of α located between α and β, $Child(\delta,\alpha)$ is the last child of δ. Thus β is the unique bad marked ancestor of α. □

Theorem 4.12 Assume that β is the unique 2-bad marked ancestor after α and The set $Children_2^{(*)}(\beta)$ is a set of black leaves located just after $Child(\beta,\alpha)$. G' is a bipartite {P_6,C_6}-free graph if and only if one of the following conditions is verified :

1) α is a 0-node.

2) α is a 2-node such that $C^{(*)}$ is an empty set.

Proof Suppose that α is a 2-node and let $b \in Children_2^{(*)}(\beta)$. Suppose that $C^{(*)}$ is non empty. By proposition 4.9, $C^{(*)}$ induces a complete bipartite graph or a monochromatic graph. In the two cases $B^{()}$ cannot induce a monochromatic graph otherwise $C^{(*)}$ is empty. Thus $B^{()}$ contains two adjacent vertices b_2,w_2. Let b_3 be a black vertex of $C^{(*)}$. Since α induces a connected graph then the last child of α must contain a white vertex say w_3. Now {b, x, b_3,w_3, b_2,w_2} induces a P_6, a contradiction.

For the only if part, we describe the construction of T'. When α is a 0-node, the construction of T' is illustrated in figure 4.a. In this case, if the set $Children^{(*)}(\alpha)$ is a singleton, δ_1 is deleted and if this unique element of

$Children^{(*)}(\alpha)$ is labeled 2, this element and δ_2 are merged. The construction of T' when the condition 2 is verified is illustrated in figure 4.b. If B^0 is a singleton, the node δ_2 is deleted and if this unique element of B^0 is labeled 0, this element and δ_1 are merged.

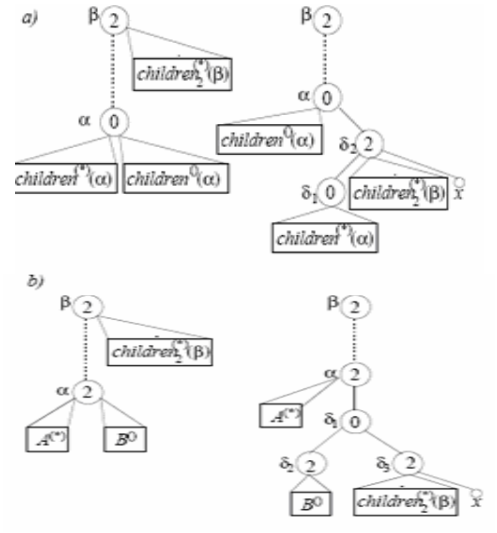


Figure 4: Construction of T' when β, the unique 2-bad marked ancestor after α, exist

4.1.3 Recognition Algorithm

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

Algorithm 2 Recognition of bipartite $\{P_6,C_6\}$-free graphs

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

Output : A canonical decomposition tree $T(G)$ if G is $\{P_6,C_6\}$-free, otherwise failure message "G is not $\{P_6,C_6\}$-free".

Initialization step : Create a list L of all the vertices of G sorted by degrees in descending order

T ← newvertex;

$G' \leftarrow \phi$;

Construct-tree (G', T, *head*(L)).

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

Mark(T, x)

Find-lowest-marked-nodes(T, S)

If $S = \phi$ then T insert(x, T)

(If x is an isolated (resp. universal) vertex, then add a new root having x as left (resp. right) child and the root of T as right (resp. left) child)

Otherwise

If S contains more than one element then Exit with the message "'failure"'.

Find the highest bad marked ancestor β for the element $\alpha \in S$ by computing the necessary sets mentioned in definition 4.7.

If there is no such β then

$T \leftarrow$ insert(x, T) (according to theorem 4.10)

Otherwise

$T \leftarrow$ insert (x, T) (according to theorem 4.12)

End If

End If

$G' \leftarrow$ G[$V(G') \cup$ {x];

If $L = \phi$ then Exit Otherwise

$L \leftarrow$ L − {x}, $x \leftarrow$ head(L)

Construct-tree(G', T, x)

End If

4.1.4 Complexity

Let's show now that recognition of a bipartite $\{P_6,C_6\}$-free graph G 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.

The 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.

Finding the unique lowest marked node

By theorem 4.5, there is a unique lowest marked node, namely α. Corneil Perl and Stewart have shown in [9] that finding α and verifying if all marked nodes lie on the path from α to the root of T takes a time that is proportional to the distance from α to the root of T. This distance is $O(dG'(x))$.

Indeed, 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 [7]): we can associate to $(\delta, \delta', \delta'')$ a private neighbor of b or w. Assume that δ

is a 0-node, then 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 $Child(\delta',\alpha)$ or a neighbor of *w* belonging to a graph located before $Child(\delta',\alpha)$. The same argument holds when δ, δ' and δ'' are respectively labelled 2, 0 and 2. Thus the number of ancestors of α is at most 3×(dG(b)+dG(w)) which is $O(d_G(x))$ since we have assumed that *x* is of maximal degree in *G*.

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 label 0 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 4.11 can be done simultaneously : If $Children_2^{(*)}(\delta)$ is not empty then this set must be a monochromatic set of black leaves located just after $Child(\delta,\alpha)$ while all other children of δ that are located before $Child(\delta,\alpha)$ 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 bad marked 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 proposition 4.9 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 a constant time.

The whole complexity

We will show now that our algorithm recognizes if a bipartite graph $\underline{G}$ is $\{P_6,C_6\}$-free within linear time on the size of $\underline{G}$. Let us first show that when adding a vertex *x* in a graph *G*, we can know in $O(d_{\underline{G}}(x))$ whether *x* is of maximal degree in $\underline{G}$. 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 $\underline{G}$. 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 $\{P_6,C_6\}$-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 $\underline{G}$.

5 Vertex series parallel digraphs source-source (VSPs)

Definition 5.1 A dag VSPs is defined recursively as follows:

1) A dag having a single vertex is VSPs.

2) If $G_1 = (V_1,E_1)$, $G_2 = (V_2,E_2)$ are two VSPs then the dag constructed by each of the following operations are also VSPs :

a) Parallel composition $G_p = (V_1 \cup V_2, E_1 \cup E_2)$.
b) Series composition $G_s = (V_1 \cup V2, E_1 \cup E_2 \cup (S_1 \times S_2))$, where S_i is the set of sources of G_i , $i = 1, 2$.

It is obvious that a VSPs dag is without transitive arcs.

Theorem 5.2 Let G be a connected dag without transitive arcs. G is VSPs if and only if G does not contain a subgraph isomorph to Z_4 or to Z_5.

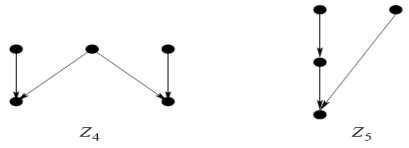


Figure 5

Proof Assume that *G* is a VSPs dag and let's show that *G* is $\{Z_4,Z_5\}$-free.

Claim 1 Let $y_1, y_2 \in V(G)$ *such that* $N^-(y_1) \cap N^-(y_2) \neq \phi$ *then* $N^-(y_1) \subseteq N^-(y_2)$ *or* $N^-(y_2) \subseteq N^-(y_1)$.

Proof By the definition of the series operation, all the arcs $\{(x, y) \mid x \in N^-(y)\}$ are created at the same time. Thus if there is two vertices y_1, y_2 such that $N^-(y_1) \cap N^-(y_2) \neq \phi$ then either all the arcs $\{(x, y_1) \mid x \in N^-(y_1)\}$ and all the arcs $\{(x, y_2) \mid x \in N^-(y_2)\}$ are created at the same time, in this case $N^-(y_1) = N^-(y_2)$, or all the arcs $\{(x, y_1) \mid x \in N^-(y_1)\}$ have been created after the creation of the arcs $\{(x, y_2) \mid x \in N^-(y_2)\}$, in this case $N^-(y_2) \subseteq N^-(y_1)$, or either all the arcs $\{(x, y_2) \mid x \in N^-(y_2)\}$ have been created before all the arcs $\{(x, y_1) \mid x \in N^-(y_1)\}$, in this case $N^-(y_1) \subseteq N^-(y_2)$. ■

By claim 1, G is Z_4-free.

Claim 2 Let y be a vertex of G then for every $x_1, x_2 \in N^-(y)$, $N^-(x_1) = N^-(x_2)$.

Proof Let $x \in N^-(x_1)$. By the definition of the series operation, the arc (x_1, y) has been created before the arcs (x, x_1). Since the arcs (x_1, y), (x_2, y) have been created at the same time then in the moment of creation of the arc (x, x_1) there was as sources x_1 and x_2. Thus $(x, x_2) \in E$, this implies that $N^-(x_1) = N^-(x_2)$. ■

By the claim 2, G is Z_4-free.

Assume now that G is a connected dag without transitive arcs and $\{Z_4, Z_5\}$-free. Let's show that G is VSPs dag.

Let S be the set of all sources of G and $Q = G[V - S]$.

Claim 3 Every vertex of Q adjacent to a vertex of S is a source of Q.

Proof Let y be a vertex of Q that is not a source and adjacent to a vertex $x \in S$. Let z be a source in Q such that z is an ancestor of y. Since G does not contain transitive arcs, $(x, u) \notin E$ for every vertex u located on the path going from z to y. Suppose that z is a predecessor of y. Since S is the set of all sources of G, there is a source $t \in S$ such that $(t,z) \in E$. Since G does not contain transitive arcs, the set $\{t, z, y, x\}$ induces the configuration Z_5, a contradiction. Suppose that z is not a predecessor of y, let u_1 be a predecessor of y in Q and u_2 is a predecessor of u_1. Since G does not contain transitive arcs then $\{x, y, u_1, u_2\}$ induces the configuration Z_5, a contradiction. ■

Let $C_1, \ldots, C_k$ be the connected components of Q and S' is the set of sources of Q. We distinguish two cases :

Case 1 : k = 1. In this case we will prove that the graph induced by $S \cup S'$ is bipartite complete. Suppose the contrary, let $y_1 \in S'$ and $x_1, x_2 \in S$ such that $(x_1, y_1) \in E$ and $(x_2, y_1) \notin$ E. Let $y_2 \in S'$ such that $(x_2, y_2) \in E$. Since $k = 1$ there is a successor y_3 of y_2
and a path from y_1 to y_3. Without less of generality we can suppose that $(y_1, y_3) \in E$. Since y_1, y_2 are sources of Q, (y_1, y_2), $(y_2, y_1) \notin$ E. Thus $\{x_1, y_1, y_3, y_2\}$ induces the configuration Z_5, a contradiction.
It is clear now that G admit a series decomposition into S and $V(G) - S$. Case 2 : k ¸ 2. Suppose that $G[S \cup S']$ is not a bipartite complete. Since $k \geq 2$ and G is connected, $G[S \cup S']$ must be also connected. We claim that there is a vertex $y \in S'$ such that for every $x \in S$, $(x, y) \in E$. Suppose the contrary, then for every vertex $y \in S'$ there is a vertex $x \in S$ such that $(x, y) \notin$ E. Let $y_1, y_2 \in S'$ and $x_1, x_2 \in S$ such that $(x_1, y_1), (x_2, y_2) \in E$ and $(x_1, y_2), (x_2, y_1) \notin$ E. Since $G[S \cup S']$ is connected, there is a chain that connects (x_1, y_1) and (x_2, y_2). Without less of generality, let $x \in S$ such that $(x, y_1), (x, y_2) \in E$, then $\{x, x_1, y_1, x_2, y_2\}$ induces the configuration Z_4, a contradiction.
Let Y be the subset of S' formed from all the universal vertices with rapport to S and $C_1, \ldots, C_r$ be the connected components of Q containing the vertices of Y. Since it is proved in the first case that every source of every connected component C_i $(1 \leq i \leq r)$ is a successor of every source in S, then G admit a series decomposition into $V(G) - (C_1 \cup \ldots \cup C_r)$ and $C_1 \cup \ldots \cup C_r$. It follows that we can always reduce G to its vertex set by a parallel decomposition and a series decomposition, this implies that G is VSPs. □

5.1 Recognition of VSPs dags

We present in this section a linear algorithm to recognize if an arbitrary dag is VSPs or not. We will take into account the following sort of the vertex set for a dag G.

Definition 5.3 Let G be a dag and S is the sources set of G, let $A_1 = S$, $A_2 = N^+(A_1)$, ... ,$A_p = N^+(A_{p-1})$. The sort $\rho = \{A_1, \ldots, A_p\}$ is called a topologically sort of $V(G)$.

Our algorithm use the following result :

Proposition 5.4 Let G be a dag and let $\rho = \{A_1, \ldots, A_p\}$ be the topologically sort of $V(G)$. Then G is a VSPs dag if and only if the following conditions are verified :

1) For every $(x, y) \in E(G)$, $x \in A_i$ and $y \in A_{i+1}$;

2) $G[A_i \cup A_{i-1}]$ is a bipartite Z_4-free graph;

3) Let $C = (C_i, C_{i-1})$ be a connected component of $G[A_i \cup A_{i-1}]$ then $N^-(x) = N^-(y)$ for every $x, y \in C_{i-1}$.

Proof We can remark that $y \in A_j$, $j > i+1$ if and only if G contains a transitive arc or G contains the configuration Z_5. The conditions 1 and 2 assure that G is Z_4-free, the conditions 1 and 3 assure that G is Z_5-free. □

The following theorem provides a simple method for verifying the condition 2 of proposition 5.4.

Theorem 5.5 A directed bipartite graph of depth 1 $G = (B \cup W, E)$ is Z_4-free if and only if for every $y_1, y_2 \in W$, $N^-(y_1) \subseteq N^-(y_2)$ or $N^-(y_1) \cap N^-(y_2) = \phi$.

Proof Obviously if G is Z_4-free then one of the conditions of theorem must be verified. On the contrary, if one of these conditions is verified then every connected component of G contains an isolated vertex or a universal vertex. Therefore we can reduce G to its vertex set by a parallel and series decomposition.

The following algorithm contains the procedures for detecting the conditions of proposition 5.4.

<u>*Algorithm 4 Recognition of VSPs dag*</u>
Input : A dag $G = (V,E)$.
Output : The message "Success" when G is VSPs otherwise "′ Failure"

Step 1
Let $\rho = \{A_1, \ldots, A_p\}$ be the topologically sort of $V(G)$ (cf. 5.3).

For every $(x, y) \in E(G)$ do
If $x \in A_i$ and $y \in A_j$ with $j > i + 1$ then
Exit with the message "Failure"
End If
End For

Step 2
Order the vertices of every A_i, $i = 1, \ldots, p$, in decreasing order according to their negative degrees (i.e. according to the number of their predecessors).
Let $G_1 = G[A_p \cup A_{p-1}], \ldots, G_i = [A_i \cup A_{i-1}], \ldots, G_{p-1} = G[A_2 \cup A_1]$
For $1 \leq i \leq p - 1$ do
Let $\{y_1, \ldots, y_r\}$ be the vertices of A_i such that $d^-(y_1) \geq d^-(y_2) \geq \ldots \geq d^-(y_r)$
For $1 \leq j \leq r$ do
Mark the vertices of $N^-(y_j)$ by j
If there is a vertex $x \in N^-(y_j)$ having a label $k \neq j$ then
If $N^-(y_k) \not\subset N^-(y_j)$ then
Exit with the message "Failure"
End If
Delete the label k
End If
End For
End For

Step 3
Let $C_1, \ldots, C_j = (C_j, C_{j-1}), \ldots C_s$ be the connected component $G_i = G[A_i \cup A_{i-1}]$
For $1 \leq j \leq s$ do
If $G[C_{j-1} \cup N^-(C_{j-1})]$ is not a bipartite complete then
Exit with the message "Failure"
End If
End For
Return "Success"

5.1.1 Complexity

Let's show now that algorithm 4 runs in $O(n + m)$ time. Assume that the graph G is represented by the lists of its successors and predecessors.
The first step of our algorithm requires $O(n + m)$ time. Indeed, it is known that the determination of ρ can be obtained in $O(n + m)$ time, and testing if every arc (x, y) of G is located between two successive levels of ρ requires a time $O(n + m)$ as well.
The second step can be executed in $O(n + m)$ time since to test the inclusions of neighborhoods for every graph G_i, $i = 1, \ldots, p - 1$, can be executed in time $O(E(G_i))$, $i = 1, \ldots, p - 1$. Moreover, the success of the first step guarantees that the edge sets of graphs G_i, $i = 1, \ldots, p - 1$, constitute a partition of E.
The third step can be implemented also in $O(n + m)$ time since finding the connected components of every graph G_i requires $O(E(G_i))$, $i = 1, \ldots, p-1$. To test if the set of sources $\{s_1, \ldots, s_k\}$ of every connected component forms a complete bipartite graph with the set of their predecessors requires $O(d^-(s_1)) + \ldots + O(d^-(s_k)))$ time. This implies that the total time is $O(n + m)$.

6 Vertex series parallel digraphs sink-sink (VSPp)

Definition 5.1 A dag VSPp is defined recursively as follows:
1) A dag having a single vertex is VSPp.

2) If $G_1 = (V_1,E_1)$, $G_2 = (V_2,E_2)$ are two VSPp then the dag constructed by each of the following operations are also VSPp :

a) Parallel composition $G_p = (V_1 \cup V_2, E_1 \cup E_2)$.
b) Series composition $G_s = (V_1 \cup V2, E_1 \cup E_2 \cup (P_1 \times P_2))$, where P_i is the set of sinks of G_i , $i = 1, 2$.

Let $G = (V,E)$ be a dag. The opposite dag G^{op} is defined by $V(G^{op}) = V(G)$ and $E^{op} = \{(x,y): (y, x) \in E\}$.

The following theorem is immediate from the definition.
Theorem 6.2 Let G be a dag. G is VSPp if and only if G^{op} is VSPs.
Thus the recognition algorithm for the family of VSPp is immediate from the recognition algorithm of VSPs.

7 Perspective

A challenging open problem is the two-processor scheduling with UET-UCT: $P2|prec, c_{i,j} =1, p_j = 1|C_{max}$ for which the complexity is unknown. Finta and al. in [5] provided a quadratic algorithm to compute an optimal schedule of this problem when the precedence constraints are represented by a subclass of VSP referred to as series-parallel-1 (SP1) graphs. As perspective, we hope that this problem is polynomial when the task graph belongs to one of our classes defined in this paper.

References

[1] J. Valdes, R. E. Tarjan, E. Lawler, The recognition of series parallel digraphs. *SIAM Journal of Computing*, Vol. 11, No. 2, May 1982, 298-313.

[2] H.M. Abdel-Wahab, T. Kameda, Scheduling to minimize maximum cumulative cost subject to series parallel precedence constraints. *Operation Research*, Vol. 26, 1978, 141-158.

[3] E.L. Lawler, Sequencing Jobs to minimize total weighted completing time subject to precedence constraints. *Annals of Discrete Mathematics*, Vol. 2, 1978, 75-90.

[4] K. Takamizawa, T. Nishizeki, N. Saito, Linear time computability of combinatorial problems on series parallel graphs. *JACM*, Vol. 18, 1982, 623-641.

[5] L. Finta, Z. Liu, I. Milis, E. Bampis, Scheduling UET-UCT series parallel graphs on two processors. *INRIA* Raport de recherch, No. 2566, 1995.

[6] J-L. Fouquet, V. Giakoumakis, J-M. Vanherpe, Bipartite graphs totally decomposable by canonical decomposition. *International Journal of Foundations of Computer Science* Vol. 10, 1999, 513-533.

[7] V. Giakoumakis, J-M. Vanherpe, Linear time recognition of Weak bisplits graphs. *International Journal of Foundations of Computer Science*, Vol. 14, No. 1, 2003, 107-136.

[8] R. Quaddoura, A linear time recognition algorithm of bipartite Star123-free graphs. *IAJIT*, Vol. 3, No 3, July 2006, 193-202.

[9] D.G. Corneil, Y. Perl, L.K. Stewart, A linear recognition algorithm for cographs. *SIAM Journal of Computing*, Vol. 14, No 4, November 1985, 926-934.