Symmetric Classes of Series-Parallel Digraphs

Ruzayn Quaddoura
Faculty of Information Technology, Zarqa University, Jordan
Ruzayn@zu.edu.jo

Abstract: *Let S and T be the set of sources and sinks of a digraph G. Valdes et al. [16] defined the Series-Parallel digraph (SP-dags) as a digraph whose reduction transitive is a Minimal Series-Parallel-digraph (MSP-dags). An MSP digraph is any digraph that can be constructed starting with one vertex by applying two composition operators, a parallel composition which is the disjoint union of two MSP-dags, and a series composition which is the disjoint union of two MSP-dags G_1 and G_2 with adding the arcs of $T_1 \times S_2$. This famous class of digraphs has numerous theoretical and applied information technology applications. We show in this paper that if we consider the multiplication in the series operation as $S_1 \times T_2$, $T_1 \times T_2$, or $S_1 \times S_2$ then the obtained symmetric classes of series-parallel digraphs are recognizable in linear time.*

Keywords: *Digraphs, series-parallel digraphs, design and analysis of algorithms, complexity.*

Received September 22, 2024; accepted December 19, 2024
https://doi.org/10.34028/iajit/22/2/1

1. Introduction

Series-Parallel digraphs (SP-dags) for short serve as powerful tools for modeling, analyzing, and optimizing complex systems across various disciplines, making them indispensable in both theoretical research and practical applications [3]. The class of SP-dags was defined by Valdes *et al*. [16] in terms of minimal series-parallel digraphs (MSP-dags) as follows:

Definition 1: *An MSP-dag is defined recursively as follows:*

a) A dag containing only one vertex is an MSP-dag.
b) If $G_1=(V_1, E_1)$ and $G_2=(V_2, E_2)$ are two MSP-dags then the dag constructed by each of the following operations is also an MSP-dag:

- *Parallel composition: $G=G_1PG_2=(V_1 \cup V_2, E_1 \cup E_2)$.*
- *Series composition: $G=G_1SG_2=(V_1 \cup V_2, E1 \cup E_2 \cup T_1 \times S_2)$ where T_1 is the set of sinks of G_1 and S_2 is the set of sources of G_2.*

A dag is an SP if and only if its transitive reduction is an MSP-dag. It is proved by Valdes *et al*. [16] that a dag is an MSP if and only if it doesn't contain a sub-graph isomorphic to the configuration in Figure 1. In addition, a linear time recognition algorithm for *SP* dags has been presented by Valdes *et al*. [16]. This algorithm effectively helps to find solutions to many problems related to this type of dag, for example [2, 10, 12].

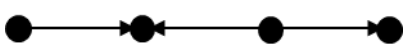

Figure 1. The forbidden configuration of an MSP-dag.

We propose in this work to study three symmetric classes of SP-dags by considering the multiplication in the series operation of the minimal members to be one of $S_1 \times T_2$, $S_1 \times S_2$, or $T_1 \times T_2$. The first class is called Series-Parallel Sources Sinks and is denoted by (SP-ST), the multiplication in the series operation of the minimal members in this class is $S_1 \times T_2$. The second class is called Series-Parallel Sources Sources and is denoted by (SP-SS), the multiplication in the series operation of the minimal members in this class is $S_1 \times S_2$. The third class is called series-parallel sinks sinks and is denoted by SP-TT, the multiplication in the series operation of the minimal members in this class is $T_1 \times T_2$. For each case, we will show that the minimal members of the corresponding class of dags can be defined by some of the forbidden configurations. Using this result, we will show that all these classes can be recognized in linear time. The motivation of this study is currently a purely mathematical point of view, hoping that it will find light in practical and theoretical applications. The paper is organized as follows: The fundamental ideas and notations that will be utilized throughout this study are provided in section 2. The class SP-ST dags is presented in section 3. The class SP-SS dags and the class SP-TT dags are presented in section 4 and section 5 respectively. We present in section 6, as a conclusion, two potential uses of these classes.

2. Preliminaries

A directed graph (or a digraph for short) $G=(V, E)$ is defined by two sets, $V(G)$ or simply V is the vertex set, and $E(G)$ or simply E is the arc set. Every arc of E is an ordered pair of vertices of V. The number n indicates the number of vertices of G and the number m indicates the number of edges of G. If $(x, y) \in E$ then x is called a predecessor of y, and y is called a successor of x. The set of all predecessors of a vertex x is denoted by $N^+(x)$, and the set of all successors of x is denoted by $N^-(x)$. The set of neighbors of x is the set $N(x) = N^+(x) \cup N^-(x)$. The

number $|N^+(x)|$ is called the positive degree of x and denoted by $d^+(x)$, and the number $|N^-(x)|$ is called the negative degree of x and denoted by $d^-(x)$, the degree of a vertex x is the number $d(x)=|N(x)|$. A vertex x is called a source if $d^+(x)=0$ and is called a sink if $d^-(x)=0$. Given a subset X of the vertices set V, the sub-graph induced by X will be denoted by $G[X]$. The set $N^+(X)$ is the set of successors of all elements of X, and the set $N^-(X)$ is the set of predecessors of all elements of X. A path of length k is a sequence of vertices $x_1, x_2, ..., x_k$ such that any two consecutive vertices form an arc. A path $x_1, x_2, ..., x_k$ is called a circuit if $x_1=x_k$ and $k \geq 2$. A directed acyclic graph, denoted by dag, is a digraph with no circuit. 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 called a cycle. An arc (x, y) is called a transitive arc if there is a path from x to y of length at least 3. A dag G is minimal if any arc of G is not transitive. The transitive reduction of a dag G is obtained by removing from G all transitive arcs. 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 $E \subseteq B \times W$. A subset S of vertices of $V(G)$ is called an independent set if there is no edge between any two vertices of S. A bi-clique is the complement in the bipartite sense of an independent set. A graph G is called F-free where F is a set of graphs when G does not contain an induced sub-graph isomorphic to a graph of F.

3. Series-Parallel Sources Sinks Dags

We define the class of series-parallel sources sinks dags in terms of minimal series-parallel sources sinks dags as follows:

Definition 2: (*Minimal series-parallel sources sinks*)

a) A dag containing only one vertex is an MSP-ST dag.
b) If $G_1=(V_1, E_1)$ and $G_2=(V_2, E_2)$ are two MSP-ST dags then the dag constructed by each of the following operations is also an MSP-ST dag :

- *Parallel composition: $G=G_1PG_2=(V_1 \cup V_2, E_1 \cup E_2)$.*
- *Series composition: $G=G_1SG_2=(V_1 \cup V_2, E_1 \cup E_2 \cup S_1 \times T_2)$ where S_1 is the set of sources of G_1 and T_2 is the set of sinks of G_2.*

Definition 3: *A dag is an SP-ST dag if and only if its transitive reduction is an MSP-ST dag.*

As MSP-dags, an MSP-ST dag G can be represented by a binary decomposition tree $T(G)$ that reflects the construction of G starting of its vertices using series and parallel operations as follows:

- The leaves correspond to the vertices of G.
- Let α be an internal node and α_1, and α_2 are respectively the left and right child of α, then α is labeled by P (resp. S) if $G[\alpha]=G[\alpha_1]PG[\alpha_2]$ (resp. $G[\alpha]=G[\alpha_1]SG[\alpha_2]$) where $G[\alpha_i]$, i=1, 2 is the sub-graph of G induced by the set of vertices having α_i as their least common ancestor.

Figure 2 illustrates an example of an MSP-ST dag and its binary decomposition tree. It is worth mentioning that the two children of an S-node are ordered according to the series operation of that node.

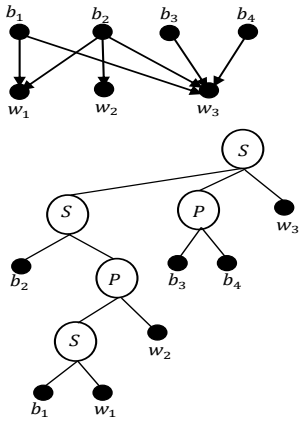

Figure 2. An MSP-ST dag and its binary decomposition tree.

There is a strong relationship between an MSP-ST dag and the notion of a $K \oplus S$-bipartite graph defined in [11] as follows:

Definition 4: *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.*

This relation is established by the following property:

Property 1: *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 every white vertex $w \in V_2$, $bw \in E$, and for every white vertex $w \in V_1$ and every black vertex $b \in V_2$, $bw \notin E$.*

Remark: Let $G=(B \cup W, E)$ be a $K \oplus S$-graph without isolated vertices, let (V_1, V_2) be the partition of $V(G)$ defined as in Property 1 that we called a K⊕S-decomposition of G. For i=1, 2, if we consider the black vertices of V_i as the set of sources of $G[V_i]$ and the set of white vertices of V_i as the set of sinks of $G[V_i]$ then, we can translate the partition (V_1, V_2) of G as a series decomposition of G to $G[V_1]$ and $G[V_2]$. This remark and Theorem 1 which is proved by Quaddoura and Al-Qerem [15] help us to characterize the MSP-ST dags as it shows in Theorem 2.

Theorem 1: *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.*

Theorem 2: *Let G be a dag. The following statements are equivalent,*

1. *G is an MSP-ST dag.*
2. *G is a bipartite $\{Z_1, Z_2, Z_3\}$-free digraph.*
3. *G is a bipartite digraph of depth 1 and every connected sub-graph of G^{n0} is a $K \oplus S$-graph, where*

G^{n0} is the graph obtained by omitting the orientation of G.

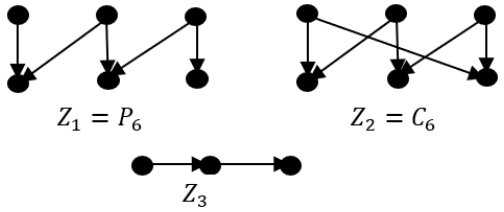

Figure 3. The forbidden configurations of an MSP-ST dag.

Proof 1⇒2: We observe that every vertex of an MSP-ST dag is either a source or a sink, the reason for this fact is that the series operation does not change the identity of a source vertex or a sink vertex. Consequently, an MSP-ST dag does not contain a path of length 3. This means that an MSP-ST dag is a bipartite dag of depth one. So, an MSP-ST dag is Z_3-free. From the other part, we can check that a P_6 or a C_6 has neither a series nor a parallel decomposition. Thus, if G is an MSP-ST dag and contains a P_6 or a C_6 then, there is a step during the construction of the binary decomposition tree $T(G)$ for which the decomposition can no longer continue. So, G must be $\{Z_1, Z_2\}$-free.

Proof 2⇒3: Assume that G is a bipartite $\{Z_1, Z_2, Z_3\}$-free digraph. Then G is a bipartite graph of depth 1 and G^{n0} is a $\{P_6, C_6\}$-free. By Theorem 1, every connected sub-graph of G^{n0} is a $K \oplus S$- graph.

Proof 3⇒1: To prove that G is an MSP-ST dag it is sufficient to prove that G can be reduced to its vertex set by a parallel and a series decomposition. We can suppose, without loss of generality, that G^{no} is connected, otherwise, since by supposition every connected sub-graph of G^{no} is a $K \oplus S$-graph, we can apply this treatment for every connected component of G^{no}. Let (V_1, V_2) be the $K \oplus S$-decomposition of G^{no}. Since G is a bipartite dag of depth 1 then, by the above Remark, (V_1, V_2) is a series decomposition of G to $G[V_1]$ and $G[V_2]$. Thus, by considering the $K \oplus S$-decomposition of every connected component of G^{no} $[V_1]$ and those of G^{no} $[V_2]$, we deduce that G can be reduced to its vertex set by a parallel and a series decomposition, therefore G is an MSP-ST.

The following Corollary is immediate since an MSP-ST dag is Z_3-free.

Corollary 1: *The class of SP-ST dags and MSP-ST dags are identical.*

To recognize that an arbitrary dag G is an SP-ST dag, we check first that G is a Z_3-free, this can be done in $O(n)$ time complexity by checking that every vertex of G is a source or a sink. Then we check that the bipartite graph G^{no} is $\{P_6, C_6\}$-free, this can be done by the $O(n+m)$ time recognition algorithm of the $\{P_6, C_6\}$-free bipartite graphs presented by Quaddoura and Al-Qerem [15].

Corollary 2: *The class of SP-ST dags can be recognized in O(n+m) time complexity.*

4. Series-Parallel Sources Sources Dags

We define the class of series-parallel sources sources dags in terms of minimal series-parallel sources sources dags as follows:

Definition 5: (*Minimal series-parallel sources sources*)

a) A dag having a single vertex is an MSP-SS dag.
b) If $G_1=(V_1, E_1)$ and $G_2=(V_2, E_2)$ are two MSP-SS dags then the dag constructed by each of the following operations is also an MSP-SS dag:

- *Parallel composition: $G=G_1PG_2=(V_1 \cup V_2, E_1 \cup E_2)$.*
- *Series composition: $G=G_1SG_2=(V_1 \cup V_2, E_1 \cup E_2 \cup S_1 \times S_2)$ where S_i is the set of sources of G_i, i=1, 2.*

Definition 6: *A dag is an SP-SS dag if and only if its transitive reduction is an MSP-SS dag.*

In the same way, as in the class of MSP dags, an MSP-SS dag G can be represented by a binary decomposition tree $T(G)$ that reflects the construction of G starting of its vertices using series and parallel operations. Also, as in the binary decomposition tree of an MSP dag, the two children of an S-node are ordered according to the series operation of that node. Figure 4 represents an SP-SS dag G, the transitive reduction of G which is the MSP-SS dag G', and the binary decomposition tree $T(G')$. The following theorem is the key to our recognition algorithm of SP-SS dags, it characterizes the MSP-SS dags by two forbidden configurations.

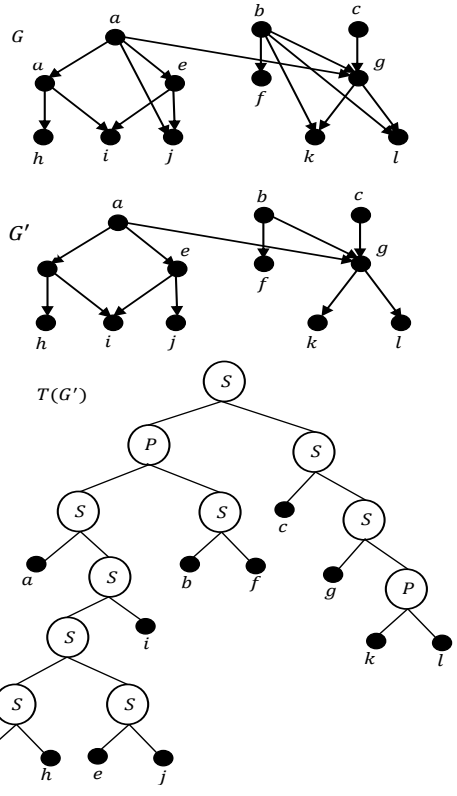

Figure 4. An SP-SS dag G, the transitive reduction of G which is the MSP-SS dag G', and the binary decomposition tree $T(G')$.

Theorem 3: *Let G be a connected dag without transitive arcs. G is an MSP-SS dag if and only if G is* $\{F_1, F_2\}$*-free, as shown in Figure 5.*

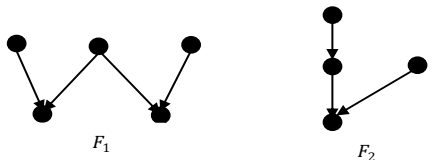

Figure 5. The forbidden configurations of an MSP-SS dag.

Proof: Suppose that G is an MSP-SS dag and let's show that G is $\{F_1, F_2\}$-free.

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

Proof: Since G is an MSP-SS, every arc in G is created by a series operation. According to the series operation, for any vertex y, all the arcs $\{(x, y) \mid x \in N^-(y)\}$ are created by the same series operation. So, if there is two vertices y_1, y_2 such that $N^-(y_1) \cap N^-(y_2) \neq \emptyset$ then, if the two sets of arcs $\{(x, y_1) \mid x \in N^-(y_1)\}$ and $\{(x, y_2) \mid x \in N^-(y_2)\}$ are created by the same series operation then $N^-(y_1) = N^-(y_2)$. Suppose that the set of arcs $\{(x, y_1) \mid x \in N^-(y_1)\}$ is created by a series operation S_1 and the set of arcs $\{(x, y_2) \mid x \in N^-(y_2)\}$ is created by a series operation S_2 where S_1 precedes S_2. Since $N^-(y_1) \cap N^-(y_2) \neq \emptyset$, the vertex y_2 was a source during the operation S_2, so $N^-(y_1) \subseteq N^-(y_2)$. If S_2 precedes S_1 then $N^-(y_2) \subseteq N^-(y_1)$.

By Claim 1, G is $F1$-free.

Claim 2: Let $y \in V(G)$, 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) is created by a series operation S_1 that precedes the series operation S_2 for which the arc (x, x_1) has been created. Since the arcs (x_1, y), (x_2, y) were created by the same series operation S_1 then, during the series operation S_2 there exist as sources x_1 and x_2, therefore $(x, x_2) \in E$, this implies that $N^-(x_1)=N^-(x_2)$.

By Claim 2, G is F_2-free.

Suppose now that G is a connected dag without transitive arcs and $\{F_1, F_2\}$-free. Let's show that G is an MSP-SS dag. Let S be the set of all sources of G and $Q = G[V - S]$.

Claim 3: Every vertex of Q that is a successor to a vertex of S is a source of Q.

Proof: Let y be a vertex of Q that is not a source and a successor 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 F_2, a contradiction. Suppose that z is not a predecessor of y, let u_1 and u_2 be two vertices of the path going from z to y such that u_1 is a predecessor of y and u_2 is a predecessor of u_1. Since G does not contain transitive arcs, the *set* $\{x, y, u_1, u_2\}$ induces the configuration F_2, a contradiction.

Let $C_1, ..., C_k$ be the connected components of Q and S' is the set of sources of Q.

Claim 4: If a source $x \in S$ is a predecessor to a source y of some connected component C_i, $1 \leq I \leq k$ then x is a predecessor to every source of C_i.

Proof: Suppose the contrary then, there is a source y' in C_i such that $(x, y') \notin E$. The vertices y and y' have a common successor in C_i. Otherwise, let z be a successor of y in C_i and z' is a successor of y' in C_i, since $G[C_i]$ is connected there is a chain in $G[C_i]$ that connects (y, z) and (y', z'), this chain contains the configuration F_1, a contradiction. Now, let $z \in C_i$ such that $(y, z), (y', z) \in E$, the set $\{x, y', y, z\}$ induces the configuration F_2, a contradiction.

If $k = 1$ then by Claim 4, $G[S \cup S']$ is a bipartite complete. By Claim 3, G admits a series decomposition into S and $V(G) - S$.
Suppose $k \geq 2$. If $G[S \cup S']$ is a bipartite complete then as above G admits a series decomposition into S and $V(G) - S$. So, suppose that $G[S \cup S']$ is not a bipartite complete. Since G is connected, $G[S \cup S']$ must be also connected.

Claim 5: There is a vertex $y \in S'$ such that for every $x \in S$, $(x, y) \in E$.

Proof: 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 in $G[S \cup S']$ that connects (x_1, y_1) and (x_2, y_2). Without loss 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 F_1, a contradiction.

Let $Y = \{y \in S' : \forall x \in S, (x, y) \in E\}$ and $C_1, ..., C_r$ are the connected components of Q that contain the vertices of Y. It is proven in Claim 4 that every source of every connected component C_i $(1 \leq I \leq r)$ is a successor of every source in S. Therefore, by Claim 5, G admits a series decomposition into $V(G) - (C_1, ..., C_r)$ and $C_1, ..., C_r$. Now, by applying this treatment on $V(G) - C_1, ..., C_r)$ and $C_1, ..., C_r$, 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 an MSP-SS dag.

4.1. Recognition of SP-SS Dags

We present in this section a linear algorithm to recognize if an arbitrary dag is an SP-SS dag or not. We will take into account the topological sort of a dag G defined to be a linear ordering of all vertices, such that if G has an arc (x, y), then x appears before y in the ordering. It is

known that the topological sort of a dag can be obtained in $O(n + m)$ time complexity [4]. We can define the topological sort of a dag to be suitable for our algorithm as follows:

Definition 7: *Let G be a dag and S is the set of sources of G, let* $A_1=S$, *and* $A_i=\{x$: *there is a source s such that the length of the longest path from s to x is equal to* $i\}$. *The sort* $\rho=(A_1, ..., A_p)$ *is called the topological sort of* $V(G)$.

For example, the topological sort of the dag G in Figure 4 is $\rho=(A_1, A_2, A_3)$ where, $A_1=\{a, b, c\}$, $A_2=\{d, e, f, g\}$ and $A_3=\{h, i, j, k, l\}$.

Our algorithm uses the following result:

Lemma 1: *Let G be a dag and let* $\rho=(A_1, ..., A_p)$ *be the topological sort of* $V(G)$. *Then G is an MSP-SS dag if and only if the following conditions are verified:*

a) For every $(x, y) \in E(G)$ *there is* $1 \leq i \leq p$-1 *such that* $x \in A_i$ *and* $y \in A_{i+)}$;
b) For every $2 \leq i \leq p$, $G[A_{i-1} \cup A_i]$ *is a bipartite* F_1*-free graph;*
c) Let $C = (C_{i-1}, C_i)$ *be a connected component of* $G[A_{i-1} \cup A_i]$, $3 \leq I \leq p$ *then for every* $x, y \in C_{i-1}$, $N^-(x)=N^-(y)$.

Proof: Let $(x, y) \in E(G)$ where $x \in A_i$ and $y \in A_j$. We can remark that $j > i + 1$ if and only if G contains a transitive arc or G contains the configuration F_2. Conditions a and b assure that G is a F_1-free, and conditions a and c assure that G is a F_2-free.

The following Lemma provides a simple method for verifying the condition b in Lemma 1.

Lemma 2: *Let* $G=(B \cup W, E)$ *be a bipartite dag of depth one. G is* F_1*-free if and only if for every* $x, y \in W$, $N^-(x) \subseteq N^-(y)$ or $N^-(x) \cap N^-(y)=\emptyset$.

Proof: Obviously if G is F_1-free then the only if conditions of this Lemma must be verified. On the contrary, if one of these conditions is verified then every connected component of G contains a universal vertex. Therefore, we can reduce G to its vertex set by a parallel and a series decomposition, so G is an MSP-SS dag, hence G is F_1-free.

We need a tool that characterizes the transitive arcs in an SP-SS dag in linear time since it's improbable that a linear time transitive reduction algorithm exists for all dags [1]. By Lemma 1, the transitive arcs in an SP-SS dag G are only those arcs that are not located between two consecutive levels of the topological sort of G. This condition is necessary but is not sufficient, since it is possible after removing the arcs of this type from some dag G then, the resulting graph may be an MSP-SS dag even though G is not an SP-SS dag. Lemma 3 presents a sufficient condition to be an arc of this type transitive.

Lemma 3: *Let G be a dag such that the graph G', obtained by removing every arc of G that is not located between two consecutive levels of the topological sort* ρ *of G, is an MSP-SS dag. If for every* arc $(x, y) \in E(G) - E(G')$ there is a common descendant sink in G' to both x and y then G is an SP-SS dag.

Proof: Let $(x, y) \in E(G) - E(G')$, and let t be a descendant sink in G' to both x and y. To prove the Lemma it is sufficient to prove that (x, y) is a transitive arc. Suppose the contrary. Let $\rho=(A_1, ..., A_p)$ be the topological sort of G. Since $(x, y) \in E(G) - E(G')$ then by supposition, there are $1 \leq i < j \leq p$ and $j \geq i + 2$ such that $x \in A_i$ and $y \in A_j$. Since t is a common descendant sink in G' to both x and y, there is $j \leq r \leq p$ such that $t \in A_r$ (in case $j = r$ then, $y = t$). Since $y \in A_j$ and $j \geq i + 2$ then, there is a path in G' from some vertex $yi \in A_i$ to y say $y_i, y_{(i+1)}, \ldots, y_j$ where $y_j = y$. Since t is a descendant sink of y then, there is a path from y to t in G' say $y_j, y_{(j+1)}, \ldots, t$. Let $P_1 = y_i, y_{i+1}, \ldots, y_j, y_{j+1}, \ldots, t$. Similarly, since t is a descendant sink of x, there is a path in G' from x to t say $P_2=x_i, x_{i+1}, \ldots, t$ where $x_i=x$. Since t is a common descendant sink in G' to both x and y, we can suppose that starting of some $i \leq k \leq r$, the sub-path $y_k, y_{k+1}, \ldots, t$ of P_1 is exactly the sub-path $x_k, x_{k+1}, \ldots, t$ of P_2. If $i \leq k \leq j$ then, (x, y) is a transitive arc, a contradiction, so $k > j$. Since $k > j \geq i+2$, the vertices $x_{k-1}, x_{k-2}, y_{k-1}$ are existed. Without loss of generality, we can suppose that $y_{k-1} = y_j = y$. Since, by the construction of ρ, every A_i, $1 \leq i \leq p$ is a stable set, and $x_{k-1}, y_{k-1} \in A_{k-1}$, we have (x_{k-1}, y_{k-1}), $(y_{k-1}, x_{k-1}) \notin E(G')$. Also $(x_{k-2}, y_{k-1}) \notin E(G')$, otherwise (x, y) is a transitive arc. But now the set $\{x_k, x_{k-1}, x_{k-2}, y_{k-1}\}$ induces the configuration F_2, a contradiction.

We can now translate the results of Lemma 1, Lemma 2, and Lemma 3 into the algorithm "Recognition of SP-SS dags." Algorithm (1) contains the necessary procedures for detecting if an arbitrary dag is an SP-SS dag or not based on the above Lemmas. The input of this algorithm is the topological sort $\rho = (A_1, ..., A_p)$ of a dag $G = (V, E)$.

- *Step* 1: Computes the transitive reduction G' of G, according to Lemma 1 (if G is an SP-SS dag), by striping every arc of E not located between two consecutive levels of ρ. The dag G becomes the dag G' in the rest of the steps, so we referred to the set of successors or the set of predecessors in G' for some vertex x to be simply $N^+(x)$ and $N^-(x)$ respectively.
- *Step* 2: Tests whether G' contains the configuration F_1 or not according to Lemma 2.
- *Step* 3: Tests whether G' contains between two consecutive levels of ρ the configuration F_2 or not according to condition c in Lemma 1. The success of step 2 and step3, means that G' is an MSP-SS dag.
- *Step* 4: Computes the descendants sinks $\delta(y)$ in G' for every vertex $y \in V(G)$.
- *Step* 5: The input of this step is the set of arcs D resulting from step 1. This final step checks whether

every arc in D is transitive according to the output of step 4 and based on Lemma 3.

Algorithm 1: Recognition of SP-SS Dag.

Input: The topological sort $\rho = (A_1,..., A_p)$ of a dag $G = (V, E)$.
Output: The message "Success" if G is an SP-SS dag, otherwise "'Failure message"

Step 1
$D = \emptyset$
For every $(x, y) \in E(G)$ *do*
If $x \in A_i$ *and* $y \in A_j$ *with* $j > i+1$ *then* $D = D \cup \{(x, y)\}$, $E = E-\{(x, y)\}$
End If
End For

Step 2
Let $G_1 = G[A_1 \cup A_2], ..., G_{p-1} = G[A_{p-1} \cup A_p]$
For i = p-1 down to 1
Let $A_{i+1} = \{y_1, ..., y_r\}$ *such that* $d^-(y_1) \geq \cdots \geq d^-(yr)$
Let $C_1 = \cdots = C_r = \emptyset$
For $1 \leq j \leq r$
If there is a vertex $x \in N^-(y_j)$ *that is marked by* $k \neq j$ *then*
If there is a vertex $x \in N^-(y_j)$ *that is not marked by k then*
Exit with a failure message
End If
Else Mark every vertex in $N^-(y_j)$ *by j;* $C_j = C_j \cup N(y_j)$
End If
End For

Step 3
Let $C_1, \ldots C_k$ *be the non-empty sets produced in step 2*
For $1 \leq i \leq k$
Let $C_i = \{x_1, \ldots, x_s\}$
For $2 \leq j \leq s$
If $N^-(x_j) \neq N^-(x_1)$ *then Exit with a failure message*
End For
End For

Step 4
For every sink t let $\delta(t) = \{t\}$
For i=p-1 down to 1
Let $A_i = \{y_1, \ldots, y_r\}$
For i=1 to r
$\delta(y_i) = \bigcup_{x \in N^+(y_i)} \delta(x)$
End for
End for

Step 5
For every $(x, y) \in D$
If $\delta(x) \cap \delta(y) = \emptyset$ *then exit with a failure message*
Return success message

4.2. Complexity

Let's show that the time complexity of this algorithm is $O(n + m)$. The computation of G' according to step 1 runs in $O(m)$ time, since this step considers only the set of arcs E. The output of step 1 is the input of step 2. According to step 1, the sets of arcs $E(G_i)$, i=1, ..., p-1, constitute a partition of $E(G)$. Therefore, testing the inclusion relation, according to Lemma 2, of the vertices of $N^-(y_j)$, j=1, …, r for every $A_{i+1} = \{y_1, \ldots, y_r\}$, i=1, ..., p-1, using the marking procedure described in step 2, can be executed in time $O(|V(G_i)| + |E(G_i)|)$. The non-empty sets C_1, …, C_k produced in step 2 for every G_i, i=1, ..., p-1 are the input of step 3. Indeed $C_i \cup \{N^+(x): x \in C_i\}$, $1 \leq i \leq k$, are the connected components of G_i, i = 1, ..., p-1. To test the condition c of Lemma 1, it is enough to compare for every $x \in C_i$, the set $N^-(x)$ with $N^-(x_1)$ where x_1 is an arbitrary vertex of C_i. This can be done in $O(|V(G_i)| + |E(Gi)|)$ time. So, the total time complexity of step 2 and step 3 requires $O(n + m)$ time. The set of descendant sinks $\delta(y)$ in G' for a vertex y is equal to the union of all the sets of descendant sinks $\delta(x)$ in G' where $x \in N^+(y)$. By step 1, if $y \in A_i$, p-1 $\leq i \leq$ 1 then, $N^+(y) \subseteq A_{i+1}$. So the computation of the descendants sinks for all the vertices $y \in A_i$, p-1$\leq i \leq$ 1 can be executed in time $O(|V(G_i)| + |E(G_i)|)$. Therefore, the total time complexity of step 4 is $O(n + m)$ time. The input of step 4 is the set D produced by step 1, so this step can be executed in $O(|D|)$. Hence, the total time complexity of the whole algorithm is $O(n + m)$.

5. Series-Parallel Sinks Sinks Dags

We define the class of series-parallel sinks sinks (SP-TT) dags in terms of minimal series-parallel sinks sinks (MSP-TT) dags as follows:

Definition 8: (*Minimal series-parallel sinks sinks*)

c) A dag having a single vertex is an MSP-TT dag.
d) If $G_1=(V_1, E_1)$ *and* $G_2=(V_2, E_2)$ *are two MSP-SS dags then the dag constructed by each of the following operations is also an MSP-SS dag:*

- *Parallel composition:* $G=G_1PG_2=(V_1 \cup V_2, E_1 \cup E_2)$.
- *Series composition:* $G=G_1SG_2=(V_1 \cup V_2, E_1 \cup E_2 \cup T_1 \times T_2)$ *where* T_i *is the set of sinks of* G_i, $i=1, 2$.

Definition 9: *A dag is an SP-TT dag if and only if its transitive reduction is an MSP-TT dag.*

Definition 10: *Let* $G=(V, E)$ *be a dag. The opposite dag of G is the dag* $G^{op}=(V, E^{op})$ where $E^{op}=\{(x, y): (y, x) \in E\}$.

It is clear that a dag G is an SP-TT if and only if G^{op} is an SP-SS. So, to recognize whether a dag G is an SP-TT or not, it is sufficient to recognize whether G^{op} is a SP-SS or not.

Corollary 3: *Let G be a connected dag without transitive arcs. G is an MSP-TT dag if and only G is* $\{H_1, H_2\}$*-free, as shown in Figure 6.*

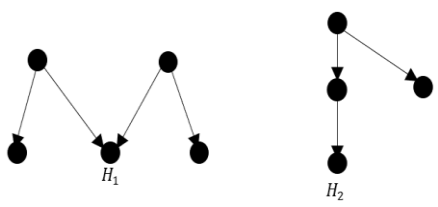

Figure 6. The forbidden configurations of an MSP-TT dag.

6. Conclusions

We think there are some significant algorithmic and combinatorial implications for the symmetric classes of series-parallel digraphs proposed in this study. In this final section, we talk about a few potential uses.

- *Oriented chromatic number*. Courcelle in [5] established the concept of oriented colorings on oriented graphs as follows: An oriented r-coloring for oriented graph $G=(V, E)$ is a partition of the vertex set V into r independent sets, such that all arcs connecting two of these subsets have the same direction. The oriented chromatic number of a graph G is the smallest integer r for which G has an oriented r-coloring. Oriented colorings are useful in scheduling models where incompatibilities are oriented [9]. Computing the oriented chromatic number of an oriented graph is generally NP-complete [9]. It was proved by Gurski *et al*. [12] that this number is at most 7 for the class of MSP dags, and a linear time algorithm for computing the oriented chromatic number of an MSP dag is proposed. We conjecture that the computation of the oriented chromatic number of an MSP-ST dag, MSP-SS dag, or MSP-TT dag can be done in efficient time.
- *Clique-width number*. Courcelle *et al*. in [6] introduce the notion of clique-width of a graph G to be the smallest number of labels required to construct G using the four operations listed below:
  - The operation $i(v)$ to create a new vertex v has the label i.
  - The operation $G \oplus H$ to make a union of two disjoint labeled graphs G and H.
  - The operation $\eta_{i,j}(G)$ to add the labeled graph G an edge (or an arc in case of digraphs) from each vertex with label i to each vertex with label j ($i \neq j$).
  - The operation $\rho_{i\rightarrow j}(G)$ to change in the labeled graph G every label i to label j.

If the clique-width of a given graph or digraph is bounded then many problems that are NP-hard in general admit polynomial-time solutions when restricted to this graph (see for example [7, 8, 13, 14]. We conjecture that the computation of the clique-width of an MSP-ST dag, MSP-SS dag, or MSP-TT dag can be done in efficient time.

References

[1] Aho A., Garey M., and Ullman J., "The Transitive Reduction of a Directed Graph," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 131-137, 1972. https://www.cs.tufts.edu/comp/150FP/archive/al-aho/transitive-reduction.pdf

[2] Baffi L. and Petreschi R., "Parallel Maximal Matching on Minimal Vertex Series-Parallel Digraphs," *in Proceedings of the Asian Computing Science Conference on Algorithms, Concurrency and Knowledge*, Thailand, pp. 34-47, 1995. https://doi.org/10.1007/3-540-60688-2_33

[3] Bang-Jensen J. and Gutin G., *Digraphs Theory, Algorithms and Applications*, Springer, 2009. https://link.springer.com/book/10.1007/978-1-84800-998-1

[4] Cormen T., Leiserson C., Rivest R., and Stein C., *Introduction to Algorithms*, The MIT Press, 2009. https://cdn.manesht.ir/19908___Introduction%20to%20Algorithms.pdf

[5] Courcelle B., "The Monadic Second-Order Logic of Graphs VI: On Several Representations of Graphs by Relational Structures," *Discrete Applied* Mathematics, vol. 54 no. 2-3, pp. 117-149, 1994. https://doi.org/10.1016/0166-218X(94)90019-1

[6] Courcelle B., Engelfriet J., and Rozenberg G., "Handle-Rewriting Hypergraph Grammars," *Journal of Computer and System Sciences*, vol. 46, no. 2, pp. 218-270, 1993. https://doi.org/10.1016/0022-0000(93)90004-G

[7] Courcelle B., *Handbook of Graph Grammars and Computing by Graph Transformation*, 1997. https://doi.org/10.1142/9789812384720_0005

[8] Courcelle B., Makowsky J., and Rotics U., "Linear Time Solvable Optimization Problems on Graphs of Bounded Clique Width," *Theory of Computing Systems*, vol. 33, pp. 125-150, 2000. https://doi.org/10.1007/s002249910009

[9] Culus J. and Demange M., "Oriented Coloring: Complexity and Approximation," *in Proceedings of the 32nd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, Merin, pp. 226-236, 2006. https://link.springer.com/chapter/10.1007/11611257_20

[10] Finta L., Liu Z., Mills I., and Bampis E., "Scheduling UET-UCT Series-parallel Graphs on Two Processors," *Theoretical Computer Science*, vol. 162, no. 2, pp. 323-340, 1996. https://doi.org/10.1016/0304-3975(96)00035-7

[11] Fouquet J., Giakoumakis V., and Vanherpe J., "Bipartite Graphs Totally Decomposable by Canonical Decomposition," *International Journal of Foundations of Computer Science*, vol. 10, no. 4, pp. 513-533, 1999. https://doi.org/10.1142/S0129054199000368

[12] Gurski F., Komander D., and Lindemann M., "Efficient Computation of the Oriented Chromatic Number of Recursively Defined Digraphs," *Theoretical Computer Science*, vol. 890, pp. 16-35, 2021. https://doi.org/10.1016/j.tcs.2021.08.013

[13] Kurt M., Berberler M., and Ugurlu O., "A New Algorithm for Finding Vertex-Disjoint Paths," *The International Arab Journal of Information*

Technology, vol. 12, no. 6, pp. 550-555, 2015. https://iajit.org/PDF/vol.12,no.6/7418.pdf

[14] Oum S. and Seymour P., "Approximating Clique Width and Branch-Width," *Journal of Combinatorial Theory Series B*, vol. 94, no. 4, pp. 514-528, 2006. https://doi.org/10.1016/j.jctb.2005.10.006

[15] Quaddoura R. and Al-Qerem A., "Bipartite (P_6, C_6)-Free Graphs: Recognition and Optimization Problems," *Symmetry*, vol. 16, no. 4, pp. 1-14, 2024. https://doi.org/10.3390/sym16040447

[16] Valdes J., Tarjan R., and Lawler E., "The Recognition of Series-Parallel Digraphs," *SIAM Journal on Computing,* vol. 11, no. 2, pp. 298-313, 1982. https://doi.org/10.1137/0211023

Ruzayn Quaddoura received his MSc in Theoretical Computer Science from Institute National Polytechnique de Grenoble (INPG, France), and his PhD in Theoretical Computer Science from Picardie Jules Verne University (Amiens, France). Currently, he is an assistant professor at Zarqa University, Faculty of Information Technology, Department of Computer Science. His research interests include Algorithmic, Combinatorial Optimization, and Graph Theory.