

# Creating Minimal Vertex Series Parallel Graphs from Directed Acyclic Graphs.

Margaret Mitchell

Division of Information and Communication Sciences

Macquarie University

Sydney, Australia

margm@ics.mq.edu.au

## Abstract

Visualizing information whose underlying graph is directed acyclic is easier if the graph is Minimal Vertex Series Parallel (MVSP). We present method of transforming any directed acyclic graph into one that is MVSP. This enables an easy visualization of the information contained in the original graph. We illustrate this by an example of a schedule that contains parallelism.

Keywords: Directed acyclic graph, parallelism, graph decomposition, minimal vertex series parallel, preservation of partial order.

## 1 Introduction

Directed acyclic graphs (DAGs) are a natural means of representing information that involves parallelism. When this information can be expressed purely in terms of series relationships (e.g. those that exist in a straight line graph) and parallel relationships (e.g. those that exist in a graph containing  $n$  vertices and no arcs), the corresponding graph is minimal vertex series parallel (MVSP). Information that can be expressed in terms of an MVSP graph can be visualized easily due to the natural way in which MVSP graphs lend themselves to decomposition. We will show this by an example in Section 5. The information represented by the graph in Figure 3 is difficult to describe without merely repeating one by one the relationships represented by each arc in the graph. As a graph becomes large, so this problem with description becomes greater.

In (Valdes & Lawler 1982), Valdes et al defined MVSP graphs and provided a method for recognising and decomposing such graphs. Others, such as (Bodlaender & de Fluiter 1996), (Eppstein 1992) and (Schoenmakers 1995) have provided methods of recognising and decomposing series parallel (SP) graphs, which are MVSP graphs that have the additional property of being planar.

When a graph is not MVSP, however, the graph cannot be decomposed in this way, and the information it contains is more difficult to visualize. For example, in a situation of scheduling tasks for users working in parallel, there is added difficulty in determining whether there are dependencies between any given sets of tasks. Such problems become greater as graphs become large.

We present a method of transforming any given DAG to a graph that is MVSP, without loss of

the constraint information contained in the original graph. It is then possible to apply a decomposition algorithm to the resulting graph. This means that the information in the original graph may now be viewed in a top-down fashion.

Decomposition algorithms such as Valdes' generally produce decomposition trees in a bottom-up fashion, by performing local analysis on the graph then building up a tree by adding parts of the graph to the analysis. The decomposition tree is binary, and therefore not unique for a given graph. That is, for a given graph, there may be several valid decomposition trees. As an alternative, we present a decomposition algorithm that analyses the graph top-down and results in a unique decomposition tree. We also show that decomposing a graph in this fashion can be done without any backtracking.

In Section 2 we give some definitions that will be used in the rest of the paper. In Section 3 we motivate the notion of using the addition of arcs to transform a DAG into an MVSP graph. We also present an overview of our graph transformation algorithm and prove that it can be applied to any directed acyclic graph. In Section 4 we present our decomposition algorithm and prove that this algorithm succeeds on any MVSP graph. In Section 5 we give an example of how our graph transformation can be used to present scheduling information that involves parallelism.

This paper is the continuation of our work presented in (Mitchell 2001).

## 2 Definitions

We begin by defining some concepts we will use throughout the paper. It is assumed that the reader is familiar with the concepts of a graph and a directed acyclic graph. We concern ourselves only with finite graphs. First, we define the notions of series and parallel composition, and the class of VSP graphs.

**Definition 1** Let  $G$  and  $G_1, \dots, G_n$  be disjoint graphs s.t.  $G = G_1 \cup \dots \cup G_n$ .

We denote this by  $G = \text{parallel}(G_1, \dots, G_n)$ . We say that  $G$  is the *parallel composition* of  $G_1, \dots, G_n$ . Note that  $G$  may have more than one parallel decomposition, and that parallel composition is associative.

**Definition 2** Let  $G_1, \dots, G_n$ , where for  $1 \leq k \leq n$ ,  $G_k = (V_k, E_k)$ , are disjoint graphs s.t.  $G = \{V_1 \cup \dots \cup V_n, E_1 \cup \dots \cup E_n \cup E_1' \cup \dots \cup E_{n-1}'\}$ , such that

for every  $e \in E_j'$

- $\text{source}(e) \in T_j$ , where  $T_j$  is the set of sinks of  $G_j$
- $\text{terminal}(e) \in S_{j+1}$ , where  $S_{j+1}$  is the set of sources of  $G_{j+1}$

and for every vertex  $u \in G_j$  the following holds.

- For every vertex  $v \in S_{j+1}$ , arc  $(u, v) \in E_j'$

We denote this by  $G = \text{series}(G_1, \dots, G_n)$ . We say that  $G$  is the *series composition* of  $G_1, \dots, G_n$ . Note that  $G$  may have more than one series decomposition, and that series composition is associative.

**Definition 3** A directed acyclic graph  $G$  is *minimal vertex series parallel (MVSP)* if it satisfies one of the following conditions.

1.  $G = (\{v\}, \emptyset)$
2. There exist disjoint MVSP DAGs  $G_1, \dots, G_n$  s.t.  $G = \text{parallel}(G_1, \dots, G_n)$ .
3. There exist disjoint MVSP DAGs  $G_1, \dots, G_n$  s.t.  $G = \text{series}(G_1, \dots, G_n)$ .

**Definition 4** Given a graph  $G$ , an arc  $e$  is *transitive* if there exists a path from the source vertex of  $e$  to the terminal vertex of  $e$  that avoids  $e$ .

**Definition 5** The transitive reduction of a graph  $G = \{V, E\}$  is  $G' = \{V, E \setminus T\}$  where  $T$  is the set of all arcs that are transitive in  $G$  and  $T \subset E$ .

**Definition 6** A directed acyclic graph  $G$  is *vertex series parallel (VSP)* if its transitive reduction is MVSP.

We now define the type of decomposition tree that our graph decomposition algorithm will produce for any given graph. Unless otherwise specified, any decomposition tree subsequently referred to in this paper is a tree of this type.

**Definition 7** A *graph decomposition tree*  $T$  for a graph  $G$  is a vertex-labelled tree such that

1. If  $G$  consists of a single vertex then  $T$  consists of a single leaf node.
2. If there exist disjoint DAGs  $G_1, \dots, G_n$  s.t.
  - $G = \text{parallel}(G_1, \dots, G_n)$
  - $n$  is the maximum number of disjoint subgraphs of  $G$  such that  $G = \text{parallel}(G_1, \dots, G_n)$

then

- the root node  $r$  of  $T$  has a value of *parallel*
- $r$  has children  $r_1, \dots, r_n$  which are root nodes of graph decomposition trees  $T_1, \dots, T_n$ , such that for all  $1 \leq i \leq n$ ,  $T_i$  is a decomposition tree of  $G_i$ .

3. If there exist disjoint DAGs  $G_1, \dots, G_n$  s.t.

- $G = \text{series}(G_1, \dots, G_n)$
- $n$  is the maximum number of disjoint subgraphs of  $G$  such that  $G = \text{series}(G_1, \dots, G_n)$

then

- the root node  $r$  of  $T$  has a value of *series*
- $r$  has children  $r_1, \dots, r_n$  which are root nodes of graph decomposition trees  $T_1, \dots, T_n$ , such that for all  $1 \leq i \leq n$ ,  $T_i$  is a decomposition tree of  $G_i$ .

For this definition to be useful in constructing an actual decomposition tree, it must be the case for  $G = \text{parallel}(G_1, \dots, G_n)$  or  $G = \text{series}(G_1, \dots, G_n)$  that  $G_1, \dots, G_n$  are MVSP. (It follows from this that  $G$  must be MVSP). This would seem intuitively obvious, what we do is prove this in Section 4.

It would also seem that any graph that is MVSP has the property that for  $G = \text{parallel}(G_1, \dots, G_n)$  or  $G = \text{series}(G_1, \dots, G_n)$  all of  $G_1, \dots, G_n$  are MVSP, thus enabling the decomposition to succeed and eliminating any need for backtracking. However, it is not immediately obvious to prove, especially since an induced subgraph of an MVSP graph is not necessarily MVSP.

Also, for the decomposition tree to be unique, it must be the case for  $G = \text{parallel}(G_1, \dots, G_n)$  or  $G = \text{series}(G_1, \dots, G_n)$  that  $G_1, \dots, G_n$  are uniquely determined.

We will prove all this in Section 4.

### 3 Why Add Arcs to a Directed Acyclic Graph?

It can be seen, intuitively, from the definitions in Section 2, that it is always possible to construct a decomposition tree for an MVSP graph (and therefore also construct a decomposition tree for any of its corresponding VSP graphs without any loss of constraint information by first deleting all transitive arcs in the graph). Since a graph that is MVSP is simply a VSP graph with its transitive arcs removed, and since removal of transitive arcs in a finite graph preserves the prerequisite relationships in the original graph, most of the reasoning that follows in the remainder of this paper will discuss MVSP graphs only.

Now, more needs to be said about how to construct such a decomposition tree for a graph that is not MVSP without any loss of information.

The graph shown in Figure 1 is MVSP. Its decomposition tree is also shown.

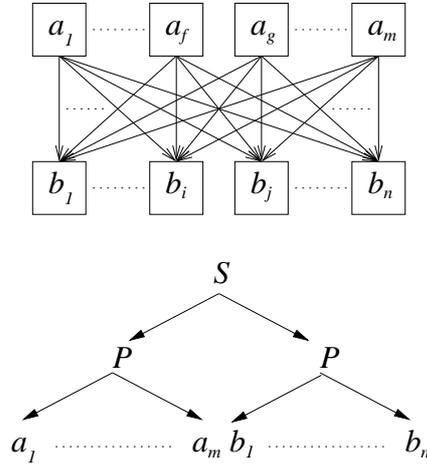


Figure 1: An MVSP graph and its decomposition tree

Now, other than the fact that  $A_g$  is not a prerequisite of  $B_j$ , the graph shown in Figure 2 is identical to the graph shown in Figure 1. If we maintain a list of prerequisite exceptions for a graph, we can represent the graph shown in Figure 2 as the decomposition tree shown the same figure.

Note that a DAG that is not MVSP is transformed into an MVSP graph by an addition, rather than a removal, of arcs. This ensures that all of the constraints present in the original graph are retained in the transformed graph.

An alternative method of representing a non MVSP graph could be to delete arcs. When a graph

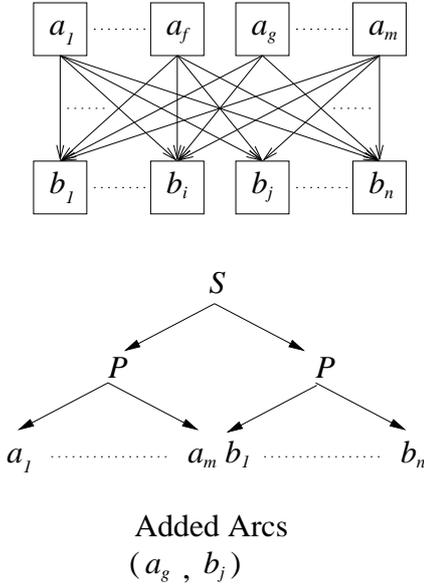


Figure 2: A non MVSP graph and the decomposition tree of its transformed graph.

represents a plan to be executed, this means to remove precedences such that the resulting graph is MVSP, and rather than prerequisite exceptions add corresponding warnings to the user about additional prerequisites. In this situation the user needs to be constantly watching out for additional warnings in order to avoid violating the prerequisites in the plan. With the use of prerequisite exceptions, however, the user can add flexibility to the execution of the plan, but if a prerequisite exception is missed none of the prerequisites of the plan are violated.

The general form of transforming a non MVSP graph into one that is MVSP can be stated as follows.

#### Algorithm 1

1. Let  $G$  be the graph corresponding to a given group of tasks.
2. Find all arcs  $(u,v)$  s.t.  $u \not\prec v, v \not\prec u$  and  $u \neq v$ , where  $u < v$  means that the task corresponding to vertex  $u$  is a prerequisite for the task corresponding to vertex  $v$  in the corresponding plan  $P$  of the graph  $G$ . Call this  $E_C$  (complementary arcs).
3. Find the smallest subset  $C$  of  $E_C$ , s.t. the  $G \cup C$ , where  $G$  is the original graph, is MVSP under transitive reduction.
4. The result of the analysis will be the decomposition tree for  $G \cup C$ , and a list of prerequisite exceptions corresponding to  $C$ .

### 3.1 Arcs Can be Added to Any Non VSP Graph such that the Resulting Graph is VSP

Before applying our graph transformation algorithm to any given directed acyclic graph, we wish to know whether there is any set of arcs at all that will transform the original graph into one that is VSP. We prove now that this is the case for any directed acyclic graph.

**Theorem 1** Given a non-VSP (and therefore non-MVSP) DAG,  $G = \{V, E\}$ , there exists a VSP graph (which can therefore be reduced to an MVSP graph)  $G' = \{V, E \cup E'\}$ , where for every  $e \in E'$ ,  $source(e) \in V$  and  $terminal(e) \in V$ .

**Proof 1** Algorithmic proof — add arcs to a given graph  $G = \{V, E\}$  to make the graph VSP.

Let  $V' = \{v_1..v_n\}$  be the sequence of all vertices in  $V$  under a topological sort.

Let  $E' = \{e_1..e_k\}$  be  $E_1 \setminus E_2$ , where  $E_1 = \{f_1..f_{n-1}\}$  is the set of arcs s.t. for all  $1 \leq j < n$ ,  $source(f_j) = v_j$  and  $terminal(f_j) = v_{j+1}$  (that, the arcs along the spine of the topologically sorted graph).

Let  $E'$  be those arcs in  $E_1$  but not in  $E$ . and  $E_2 = E_1 \cap E$ .

Let  $G' = \{V, E \cup E_1\} = \{V, E \cup E'\}$ . Let  $G_1 = \{V, E_1\}$ . Let  $E_3 = E \setminus E_1$ .

The graph  $G_1$  is a straight line and is MVSP. Every arc in  $E_3$  is a transitive arc in  $G'$ . So the graph  $G'$  is VSP.

## 4 Decomposition Algorithm

Our top-down decomposition algorithm can be stated as follows.

**Algorithm 2** 1. If  $G$  consists of a single vertex then let  $T$  be a single leaf node.

2. If  $G$  consists of disjoint subgraphs  $G_1, \dots, G_n$ , then  $G_1, \dots, G_n$  satisfy case 2 of Definition 7. Construct tree as per case 2 of Definition 7.

3. If  $G$  consists of one connected component, find disjoint DAGs  $G_1, \dots, G_n$  s.t. case 3 of Definition 7 is satisfied and construct tree as per case 3 of Definition 7.

For this Algorithm 2 to succeed, we require that it succeeds on  $G_1, \dots, G_n$  for both case 2 and case 3. (This will actually mean that  $G_1, \dots, G_n$  must be MVSP, and we will prove that Algorithm 2 succeeds for a given graph if and only if that graph is MVSP). In informal terms, we wish to “break up” a graph in series or parallel, and know that the resulting components are MVSP, without any need to check these components. The following subsection gives us this result. After that, we prove that Algorithm 2 succeeds for a given graph if and only if that graph is MVSP and that such a decomposition tree is unique.

### 4.1 Results Regarding Graph Decomposition

When constructing a graph decomposition tree for a graph  $G$ , each step is concerned with finding subgraphs  $G_1, \dots, G_n$  of  $G$  such that  $G$  is either the series composition or parallel composition of  $G_1, \dots, G_n$ . If  $G_1, \dots, G_n$  are all MVSP, we know, from the definition of MVSP, that  $G$  is also MVSP. In this section, we prove that the converse also holds. In informal terms, this means that there is never the possibility of “breaking up” a graph into series or parallel components and finding that this breakup will not permit a decomposition tree.

**Lemma 1** Given an MVSP graph,  $G$ , then any two subgraphs  $G_1, G_2$  of  $G$ , s.t.  $G = parallel(G_1, G_2)$ , are MVSP.

**Proof 2** Proof by contradiction:

Assume that there exists a set  $S$  of MVSP graphs, s.t. for any graph  $G \in S$ , there exist two subgraphs,  $G_1, G_2$  of  $G$ , s.t.  $parallel(G_1, G_2) = G$ , and one of  $G_1, G_2$  is not MVSP.

Let  $X$  be a minimal graph in  $S$ , where size is measured by the number of vertices in the graph. (1)

By (1), let  $X_1$  and  $X_2$  be subgraphs of  $X$  s.t.  $parallel(X_1, X_2) = X$  and  $X_1$  (w.l.o.g.) is not MVSP. (2)

Considering Definition 3 of MVSP;

It follows from (2) that the graph  $X$  is not a single vertex graph. This means that  $X$  does not satisfy case 1 of the definition of MVSP.

It follows from (2) and the definition of parallel composition that  $X$  consists of two or more connected components. By the definition of series composition, which always produces graphs consisting of a single connected component,  $X$  does not satisfy case 3 of the definition of MVSP.

But since  $X$  is MVSP, it must therefore satisfy case 2 of the definition of MVSP.

Therefore, by the definition of MVSP there exist subgraphs  $A, B$  of  $X$ , s.t.  $\text{parallel}(A, B) = X$  and  $A, B$  are MVSP.

One of the following is true (w.l.o.g.):

1.  $A = X_1, B = X_2$
2.  $A \subset X_1, B = \text{series}(X_2, X_1')$  (or, w.l.o.g.,  $B = \text{series}(X_1', X_2)$ ), where  $X_1' = X_1 \setminus A$
3.  $B \subset X_2, A = \text{series}(X_1, X_2')$  (or, w.l.o.g.,  $A = \text{series}(X_2', X_1)$ ), where  $X_2' = X_2 \setminus B$
4.  $A \subset X_1, B = \text{parallel}(X_2, X_1')$ , where  $X_1' = X_1 \setminus A$
5.  $B \subset X_2, A = \text{parallel}(X_1, X_2')$ , where  $X_2' = X_2 \setminus B$

Considering each case

1. Contradiction, by (2), which implies  $A$  is not MVSP.
2. Since  $B = \text{series}(X_2, X_1')$ ,  $X_1'$  and  $X_2$  belong to the same connected component of  $X$ , by definition of series composition. By definition of MVSP,  $X_2$  cannot be related in parallel composition with any superset of  $X_1'$ . Contradiction, by (2).
3. Since  $A = \text{series}(X_1, X_2')$ ,  $X_2'$  and  $X_1$  belong to the same connected component of  $X$ , by definition of series composition. By definition of MVSP,  $X_1$  cannot be related in parallel composition with any superset of  $X_2'$ . Contradiction, by (2).
4. By (1),  $X_1'$  is MVSP. Hence  $\text{parallel}(X_1', A) = X_1$  is MVSP, by the definition of MVSP graph. Contradiction, by (2).
5.  $A$  is an MVSP graph containing subgraphs  $X_1, X_2'$ , s.t.  $\text{parallel}(X_1, X_2') = A$ , one of which ( $X_1$ ) is not MVSP. But  $A$  is smaller than  $X$ , hence contradiction, by (1).

**Lemma 2** Given an MVSP graph,  $G$ , any  $n$  subgraphs  $G_1..G_n$  of  $G$ , s.t.  $\text{parallel}(G_1..G_n) = G$ , are MVSP.

**Proof 3** By repeated application of Lemma 1.

**Corollary 1** Given  $n$  graphs,  $G_1..G_n$ , where one of  $G_1..G_n$  is not MVSP,  $G = \text{parallel}(G_1..G_n)$ , is not MVSP.

**Lemma 3** Given an MVSP graph,  $G$ , then any two subgraphs  $G_1, G_2$  of  $G$ , s.t.  $G = \text{series}(G_1, G_2)$ , are MVSP.

**Proof 4** Proof by contradiction:

Assume that there exists a set  $S$  of MVSP graphs, s.t. for any graph  $G \in S$ , there exist two subgraphs,  $G_1, G_2$  of  $G$ , s.t.  $\text{series}(G_1, G_2) = G$ , and one of  $G_1, G_2$  is not MVSP.

Let  $X$  be the smallest graph, or one of the smallest graphs in  $S$ , where size is measured by the number of vertices in the graph. (3)

By (3), let  $X_1$  and  $X_2$  be subgraphs of  $X$  s.t.  $\text{series}(X_1, X_2) = X$  and  $X_1$  (w.l.o.g.) is not MVSP. (4)

Considering the definition of MVSP;

It follows from (4) that  $X$  is not a single vertex graph. This means that  $X$  is not a single vertex and does not satisfy case 1 of the definition of MVSP.

It follows from (4) and the definition of series composition that  $X$  consists of a single connected component. From the definition of parallel composition, which always produces graphs consisting of two or more connected components,  $X$  does not satisfy case 2 of the definition of MVSP.

But since  $X$  is MVSP, it must therefore satisfy case 3 of the definition of MVSP.

Therefore, by the definition of MVSP, there exist subgraphs  $A, B$  of  $X$ , s.t.  $\text{series}(A, B) = X$  and  $A, B$  are MVSP. (5)

One of the following is true (w.l.o.g.):

1.  $A = X_1, B = X_2$
2.  $A \subset X_1, B = \text{parallel}(X_2, X_1')$ , where  $X_1' = X_1 \setminus A$
3.  $B \subset X_2, A = \text{parallel}(X_1, X_2')$ , where  $X_2' = X_2 \setminus B$
4.  $A \subset X_1, B = \text{series}(X_1', X_2)$  (or, w.l.o.g.,  $B = \text{series}(X_2, X_1')$ ), where  $X_1' = X_1 \setminus A$
5.  $B \subset X_2, A = \text{series}(X_1, X_2')$ , (or, w.l.o.g.,  $A = \text{series}(X_2', X_1)$ ), where  $X_2' = X_2 \setminus B$

Considering each case

1. Contradiction, by (4), which implies  $A$  is not MVSP.
2. By Lemma 1,  $X_1'$  is MVSP. By (5),  $A$  is MVSP. By definition of MVSP graphs,  $\text{parallel}(A, X_1') = X_1$  is MVSP. Contradiction by (4), which says  $X_1$  is not MVSP.
3.  $A$  is an MVSP graph containing subgraphs  $X_1, X_2'$ , s.t.  $A = \text{parallel}(X_1, X_2')$ , one of which ( $X_1$ ), is not MVSP. Contradiction by Lemma 1.
4. By (3),  $X_1'$  is MVSP. Hence  $\text{series}(X_1', A) = X_1$  is MVSP, by definition of MVSP graphs. Contradiction, by (4), which says  $X_1$  is not MVSP.
5.  $A$  is an MVSP graph containing subgraphs  $X_1, X_2'$ , s.t.  $\text{series}(X_1, X_2') = A$ , one of which ( $X_1$ ) is not MVSP. But  $A$  is smaller than  $X$ , hence contradiction, by (3).

**Lemma 4** Given an MVSP graph,  $G$ , any  $n$  subgraphs  $G_1..G_n$  of  $G$ , s.t.  $\text{series}(G_1..G_n) = G$ , are MVSP.

**Proof 5** By repeated application of Lemma 3.

**Corollary 2** Given  $n$  graphs,  $G_1..G_n$ , where one of  $G_1..G_n$  is not MVSP,  $G = series(G_1..G_n)$  is not MVSP

We are now in position to state that an MVSP graph can be “broken up” any way we like to provide a decomposition of a graph and a visualization of the information it contains. This leads to the results in the following subsection regarding the ability to construct a decomposition tree for a given graph if and only if that graph is MVSP. For a decomposition to be unique, however, it must be the case that the graph subcomponents formed as a result of this breakup are uniquely determined. We will prove in the next subsection that our definition of decomposition has this property.

## 4.2 Results Regarding Decomposition Trees for MVSP Graphs

We now prove that, for a given graph, it is possible to construct a decomposition tree of the type we’ve defined if and only if that graph is MVSP, and that when we can do so, the decomposition tree is unique.

**Lemma 5** For a graph  $G$ , Algorithm 2 succeeds  $G$  iff  $G$  is MVSP.

**Proof 6** Assume  $G$  is MVSP. Prove that we can construct a decomposition tree  $T$  of  $G$ . Consider the construction of  $T$  in terms of Definition 7 of a decomposition tree and Definition 3 of an MVSP graph.

- **Base Case** If  $G$  satisfies Case 1 of Definition 7,  $T$  is a single leaf node.
- **Induction Step A** If  $G$  satisfies Case 2 of Definition 7 (i.e. there exist disjoint MVSP digraphs  $G_1, \dots, G_n$  s.t.  $G = parallel(G_1, \dots, G_n)$ , where  $n$  is the maximum number of disjoint subgraphs of  $G$  such that  $G = parallel(G_1, \dots, G_n)$ ), we construct  $T$  as follows.
  - Root node  $r$  has value *parallel*.
  - $r$  has children  $r_1, \dots, r_n$ .
  - $r_1, \dots, r_n$  are root nodes of decomposition trees  $T_1, \dots, T_n$ , such that for all  $1 \leq i \leq j$ ,  $T_i$  is a decomposition tree of  $G_i$ .

By Lemma 2,  $G_1, \dots, G_n$  are MVSP, so we can construct decomposition trees  $T_1, \dots, T_n$  for  $G_1, \dots, G_n$ . So by induction, we can construct a decomposition tree for  $G$ .

- **Induction Step B** By similar reasoning, if  $G$  satisfies Case 3 of Definition 7, we can construct a decomposition tree for  $G$ .

Assume that we can construct a decomposition tree  $T$  of  $G$ . Prove  $G$  is MVSP.

Proof by induction.

- **Base Case** By definition of a decomposition tree, all subtrees of  $T$  that are leaf nodes represent single vertex graphs, which are MVSP by case 1 of the definition of MVSP.
- **Induction Step A** Assume that root node  $r$  of  $T$  has value *parallel*, and its children  $r_1, \dots, r_n$  are the root nodes of subtrees  $T_1, \dots, T_n$  respectively, and that the graphs  $G_1, \dots, G_n$  that they respectively represent are MVSP. By case 2 of the definition of MVSP, the graph  $G$  which  $T$  represents is MVSP.

- **Induction Step B** By similar reasoning for root node  $r$  of  $T$  with value *series*, the graph  $G$  which  $T$  represents is MVSP.

So, for a graph  $G$ , it is possible to construct a decomposition tree of  $G$  iff  $G$  is MVSP.

**Lemma 6** For a graph  $G$ , if Algorithm 2 succeeds then  $T$  is unique.

**Proof 7** We prove this by showing that for a decomposition tree  $T$  of  $G$ , the children  $r_1, \dots, r_n$  of root node  $r$  of  $T$  are uniquely determined. Another way of stating this is as follows.

- If  $G = parallel(G_1, \dots, G_n)$  and  $n$  is the maximum number of disjoint subgraphs of  $G$  such that  $G = parallel(G_1, \dots, G_n)$ , then  $G_1, \dots, G_n$  are uniquely determined.
- If  $G = series(G_1, \dots, G_n)$  and  $n$  is the maximum number of disjoint subgraphs of  $G$  such that  $G = series(G_1, \dots, G_n)$ , then  $G_1, \dots, G_n$  are uniquely determined.

We prove this by contradiction.

- **Parallel Case** Let  $r$  have value *parallel*, that is let  $G = parallel(G_1, \dots, G_n)$ .

Assume also that  $G = parallel(G'_1, \dots, G'_n)$ , where for at least one  $i$  in  $1 \leq i \leq n$ ,  $G'_i$  is not the same graph as  $G_i$ .

By the pigeonhole principle and maximality of  $n$ , and w.l.o.g, this means that one of  $G'_1, \dots, G'_n$ , say  $G'_j$  is a proper subgraph of one of  $G_1, \dots, G_n$ , say  $G_j$ .

Let  $G''_j$  be  $G_j \setminus G'_j$ . By the definition of parallel composition, since  $G_j$  and  $G'_j$  are disjoint subgraphs of  $G$ ,  $G''_j$  is also a disjoint subgraph of  $G$ . So,  $G = parallel(G_1, \dots, G'_j, G''_j, \dots, G_n)$ . But this means that the maximum number of disjoint subgraphs of  $G$  such that  $G$  is the parallel composition of these graphs is  $n + 1$ , which is a contradiction.

So, if  $G = parallel(G_1, \dots, G_n)$  and  $n$  is the maximum number of disjoint subgraphs of  $G$  such that  $G = parallel(G_1, \dots, G_n)$ , then  $G_1, \dots, G_n$  are uniquely determined.

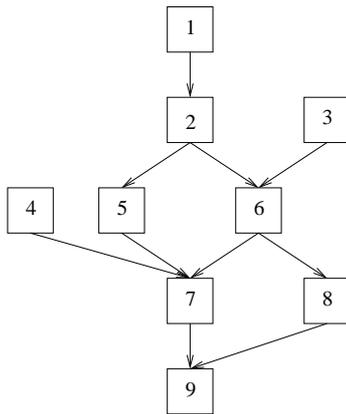
- **Series Case** By similar reasoning for *series*, if  $G = series(G_1, \dots, G_n)$  and  $n$  is the maximum number of disjoint subgraphs of  $G$  such that  $G = series(G_1, \dots, G_n)$ , then  $G_1, \dots, G_n$  are uniquely determined.

So, for a decomposition tree  $T$  of  $G$ , the children  $r_1, \dots, r_n$  of root node  $r$  of  $T$  are uniquely determined. Therefore, for a graph  $G$ , if it is possible to construct a decomposition tree  $T$  of  $G$  then  $T$  is unique.

These results guarantee us the ability to produce a decomposition tree for any MVSP graph without any need for backtracking. They also mean that if a top-down decomposition such as we have described has not succeeded on a given graph, that graph is not MVSP, and the information it contains cannot be visualized purely in terms of series and parallel relationships.

## 5 Example of Applying the Graph Transformation Algorithm to a Scheduling Problem

The following example will show, on a small scale, the changes that take place in a graph under our transformation and how these changes facilitate the visualization of the information contained in that graph. Consider the graph shown in Figure 3, and the tasks corresponding to the nodes in the graph. The tasks are concerned with laying a lawn complete with sprinkler system. The graph is not MVSP, and there is difficulty in stating the constraints within the graph succinctly without simply repeating the relationships in the graph as represented by each arc. Unless a graph is small, it is easy to become lost and violate a constraint in the plan. This difficulty with visualizing information whose underlying graph is non MVSP graph increases as a set of tasks increases in size.



1. Order Topsoil
2. Spread Topsoil
3. Buy Sprinkler Equipment
4. Order Turf
5. Spread Fertilizer
6. Install Sprinkler System
7. Lay Turf
8. Install Sprinkler Controller
9. Water Lawn

Figure 3: A set of tasks corresponding to a non MVSP graph

We modify the graph shown in Figure 3 by adding the arcs (6, 5) and (8, 7). The resulting graph is shown in Figure 4. We then perform a transitive reduction on that graph.

The results are shown in Figure 5. Figure 6 shows a decomposition tree of this graph.

The decomposition tree shown in Figure 6 can now be used to represent the set of tasks shown in Figure 3 in a way that makes clear the series and parallel relationships within those tasks. One of many possible visualizations, other than of course the decomposition tree itself, involves the use of sections, as shown in Figure 7. Another possible representation could be a summarized view of the tasks with hypertext links to show more detailed information.

We can see from this example that visualization of the scheduling information given in Figure 3 is facilitated by transforming the underlying graph to one that is MVSP.

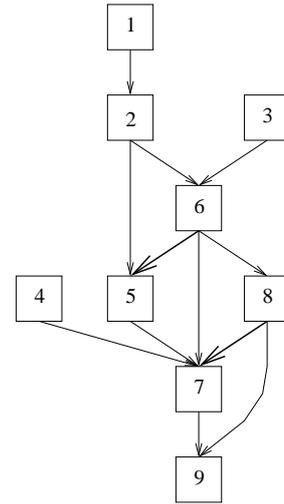


Figure 4: Result of adding arcs (6, 5) and (8, 7) to the graph shown in Figure 3

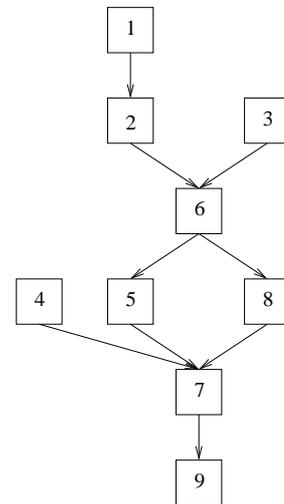


Figure 5: Transitive reduction of graph shown in Figure 4.

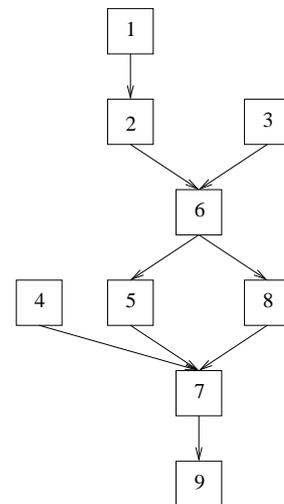


Figure 6: Decomposition tree for graph shown in Figure 5.

Complete sections 1, 2 and 3 in sequence.

- 1 Complete sections 1A and 1B in any order.
  - 1A Order the turf.
  - 1B Complete sections 1B.1-1B.3 in sequence.
    - 1B.1 Complete sections 1B.1A and 1B.1B in any order.
      - 1B.1A Complete items 1B.1A.1 and 1B.1A.2 in sequence.
        - 1B.1A.1 Order the topsoil.
        - 1B.1A.2 Spread the topsoil.
      - 1B.1B Buy the sprinkler equipment.
    - 1B.2 Install the sprinkler system.
    - 1B.3 Complete items 1B.3A and 1B.3B in any order.
      - 1B.3A Spread the fertilizer
      - 1B.3B Install the sprinkler controller
- 2 Lay the turf.
- 3 Water the lawn.

Note: You may begin to spread the fertilizer before finishing installing the sprinkler system. You may also begin to lay the turf before finishing installing the sprinkler controller.

Figure 7: An example of instructions based on the set of tasks in Figure 3 after applying our graph transformation to the underlying graph

## 6 Conclusion

In this paper we have presented the problem of creating an MVSP graph from a directed acyclic graph by the addition of arcs, and shown that this is possible to do for any directed acyclic graph. The resulting graph may then be decomposed. This means that the information contained in a directed acyclic graph can be more easily visualized. We are currently working on an NP-Completeness proof for the associated decision problem, and its complexity for the class  $W[1]$ .

In connection with the fact that decomposing a graph enables easier visualization of the information contained in that graph, we have also presented a method of decomposing an MVSP graph in a top-down fashion and proved that this succeeds if and only if a graph is MVSP. This method of decomposition produces a unique decomposition for a given graph and does so without a need for backtracking. We are working on assessing the complexity of decomposing a graph in the top-down fashion we described.

## References

- Bodlaender, H. L. & de Fluiter, B. (1996), Parallel algorithms for series parallel graphs, in 'Lecture Notes in Computer Science', pp. 277-289.
- Eppstein, D. (1992), 'Parallel recognition of series-parallel graphs', *Information and Computation* **98**(1), 41-55.
- Mitchell, M. (2001), Use of directed acyclic graph analysis in generating instructions for multiple users, in 'Conferences in Research and Practise in Information Technology', Vol. 9, Sydney, Australia.
- Schoenmakers, B. (1995), A new algorithm for the recognition of series parallel graphs, Technical

Report CS-R9504, Centrum voor Wiskunde en Informatica (CWI).

Valdes, J., T. R. & Lawler, E. (1982), 'The recognition of series parallel digraphs', *SIAM Journal of Computing* **11**(2), 298-313.