Kiran Kumar Bhogadi
Decomposition and Generation of Minimal Strongly Connected Digraphs
(Under the direction of Robert W. Robinson)

A new algorithm for the generation of minimal strongly connected digraphs(MSDs) is implemented. The previous method – implemented by Susan Gardner Mayhorn – is practical for generating all MSDs through 9 vertices. Using the new method, we are able to generate all MSDs through 12 vertices and those that are 2-connected through 13 vertices, and to count the classes of digraphs by number of vertices and number of arcs, labeled and unlabeled. Our method uses Brendan D. McKay's software package *nauty* in essential ways, to avoid counting duplicates and to calculate automorphism group orders for the labeled counting. Using MAPLE, we calculated the number of labeled MSDs on 13 vertices by number of arcs using an identity relating the number of labeled MSDs to those that are 2-connected, and the same identity provides a check on the numbers generated through 12 vertices.

In addition, a characterization of Cunningham's decomposition trees for MSDs under the X-join (substitution) composition is given. The initial plan was to generate the MSDs using Cunningham's decompositon theorem, which implies that any nontrivial MSD has a unique minimal decomposition in which each member is either a prime, a distar or a cycle. This approach was abandoned due to the difficulty of recognizing MSDs which are prime with respect to X-joins. Using our characterization, all the Cunningham prime MSDs through 11 vertices have been generated.

Index words: Digraph Decomposition, Minimal Strongly Connected Digraphs, 2-Connected MSDs, Labeled Digraphs, Unlabeled Digraphs

Decomposition and Generation of Minimal Strongly Connected Digraphs

by

Kiran Kumar Bhogadi

B.E., Osmania University, Hyderabad, India, 1997

A Thesis Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Master of Science

Athens, Georgia

1999

Decomposition and Generation of Minimal Strongly Connected Digraphs

by

Kiran Kumar Bhogadi

Approved:

Major Professor

Date

Approved:

Graduate Dean

Date

Acknowledgements

I am indebted to Dr. Robinson for his support and guidance throughout my thesis. I appreciate the generous amount of time he has spent working with me – once we had a 4 hour meeting and I did not even have an appointment! Without his guidance, I could not have done this thesis. I thank Dr. Canfield for helping me with MAPLE in verifying the numbers generated and in calculating the labeled MSD numbers on 13 vertices. I thank Dr. Everett for serving on my thesis committee and also Dr. B. D. McKay of the Computer Science Department, Australian National University (*nauty* is fast!).

This thesis is dedicated to my father.

Preface

The existing method for constructing MSDs [12] is based on the simple idea of searching through digraphs on a given set of vertices and testing each digraph to see if it is minimally strongly connected. After optimizing to reduce the size of the search space and employing efficient ways to represent and calculate a digraph's reachability, this method can generate all the MSDs through 9 vertices. On 9 vertices this method took about 10 hours on a SUN Ultra SPARC workstation. On 10 vertices, the estimated amount of time required is 365 days.

Our implemention of the new algorithm can generate all the MSDs through 12 vertices in just over an hour on a SUN Ultra SPARC workstation. The new algorithm, based on the fact that every MSD has at least one vertex of indegree and outdegree 1, generates larger MSDs from smaller seed MSDs by adding paths between qualified vertices in a systematic way. In both implementations, *nauty* has been used extensively to test for isomorphism and to count the numbers of labeled digraphs. As a special case of this algorithm, we are also able to generate all the 2-connected MSDs through 13 vertices in less than an hour on a SUN Ultra SPARC. Using MAPLE and the numbers on labeled 2-connected MSDs through 13 vertices, we calculated the number of labeled MSDs on 13 vertices by number of arcs.

Contents

List of Figures

Chapter 1

Introduction

1.1 Cataloguing Graphs

Catalogued data about an interesting subclass of graphs or digraphs can be used to originate new ideas and conjectures or to provide counter examples to disprove new conjectures. Minimal Strongly connected Digraphs (MSDs) are a subclass of strongly connected digraphs that have been recently catalogued through 9 vertices [12]. Sasao and Butler [14, 15] use MSDs to represent sum-of-products expressions for a function in which none of the products can be removed without changing the function. Their work provides motivation for cataloguing and counting MSDs.

The existing method for constructing MSDs [12] is based on the simple idea of searching through digraphs on a given set of vertices and testing each digraph to see if it is minimally strongly connected. To reduce the search field only one digraph is extended from each isomorphism class of small digraphs and to cut down the amount of time taken to test a digraph for minimal strongness an efficient way to represent and calculate a digraph's reachability is used. Using this method, all the MSDs through 9 vertices have been catalogued.

Our new method for constructing MSDs is based on the fact that every MSD has at least one vertex of indegree and outdegree 1. Larger MSDs are generated from smaller MSDs by systematically adding paths between qualified vertices. Using this method we are able to catalogue and count all the MSDs through 12 vertices, and all 2-connected MSDs through 13 vertices. From the labeled 2-connected MSD numbers, using MAPLE we calculated the number of labeled MSDs on 13 vertices by number of arcs; see [10, Eq. 1.3.3]. Using the identity from this equation, the results are checked through 12 vertices.

1.2 Minimal Strongly connected Digraphs

A graph in which every edge is oriented is called a directed graph or *digraph.* An oriented edge is called an *arc.* Unless stated otherwise, we follow the definitions in Harary [9]. A digraph D is an ordered pair $D = (V, X)$ where V is a finite set of vertices, and X is a collection of ordered pairs of distinct vertices (*i.e.,* the arcs). A digraph is *trivial* if it has just one vertex and no edges. An arc joining vertex u to vertex v will be represented as (u, v), where u is said to be *adjacent to* v, and v is *adjacent from* u. Vertex u is the source, and vertex v is the terminating vertex or terminus.

If $a, b \in V(D)$ and there are two or more (a, b) arcs, then we call D a *multigraph.* Unless it is explicitly stated that a digraph is a multigraph, it is assumed that it is not.

An undirected graph is connected if every two distinct vertices are joined by a path. A *path* in a digraph is defined as a directed walk in which the vertices are distinct. Such a walk is represented as a sequence of vertices: $v_0, v_1, \ldots, v_n$ where (v_{i-1}, v_i) is an arc for each $1 \leq i \leq n$. A *semipath* is defined as a directed semiwalk, which is like a directed walk except that arcs can be either (v_{i-1}, v_i) or (v_i, v_{i-1}), in which every vertex is distinct. If a path joins vertex u to vertex v, then v is reachable from u, hence the term *reachability.*

Unlike connected undirected graphs, connected digraphs are characterized as either unilateral, weak, or strong based on the notion of reachability. A *weak* digraph has the property that every two of its vertices are joined by a semipath. A *unilaterally* *connected* digraph is defined as the digraph in which for any pair of vertices, at least one is reachable from the other. This means there exists a path in one direction or the other between any two of its vertices. A *strongly connected* digraph is a digraph in which any two of its vertices are *mutually reachable*, *i.e.,* each is reachable from the other.

A minimal strongly connected digraph is a strong digraph D in which the removal of any arc x will produce the digraph $D - x$ which is not strong. Three examples of minimal strongly connected digraphs are shown in Figure 1.1. Note that a cycle [1] is a MSD. Also, a trivial digraph is an MSD. Unless explicitly stated, we use the term MSD to represent a non-trivial MSD.

The number of vertices adjacent to a vertex u is called the *indegree* of u, denoted by $id(u)$. Similarly, the number of vertices adjacent from a vertex u is called the *outdegree* of u denoted by $od(u)$. A *carrier* is a vertex u such that $id(u) = od(u) = 1$. Any other vertex is a *non-carrier*.

Note: We use the following abbreviations to classify vertices:

NCI denotes a non-carrier u such that $id(u) = 1$,

NCO denotes a non-carrier u such that $od(u) = 1$,

NC denotes a non-carrier u such that $id(u) > 1$ and $od(u) > 1$.

In an MSD, for any vertex v, $id(v) \geq 1$ and $od(v) \geq 1$. Thus, the classification carrier, NCI, NCO, NC covers all vertices for an MSD.

A *distar* G is a digraph such that $|V(G)| \geq 3$ and for some $u \in V(G)$,

$E(G) = \{(u, v) : v \in V(G) - u\} \bigcup \{(v, u) : v \in V(G) - u\}$

Vertex u is called the *center* of the distar. Any vertex other than the center is called an *ear* vertex. Clearly, a distar is a MSD because all its arcs are essential.

Prime MSDs are discussed in the next chapter.

[1] We will use the term cycle to represent a directed cycle (or dicycle) throughout.

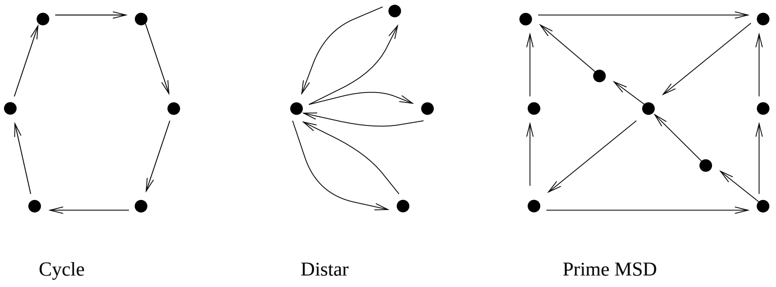

Figure 1.1: Three MSDs: cycle, distar, and a prime MSD on 9 vertices.

Geller [7] defined the term *pseudocycle*, which is a path $W = u_0, u_1 \ldots, u_n$ combined with the arc (u_0, u_n). Geller proved that if a digraph D is minimally strongly connected, then D does not contain a pseudocycle. Hedetneimi [11] characterized MSDs by proving that if a strong digraph contains no pseudocycle then it is minimally strong. In the pseudocycle W, the (u_0, u_n) arc could be described as *non-essential* given that its presence is "non-essential" for u_n to be reachable from u_0. This term will be frequently used throughout the paper. An alternative definition is that $e = (u, v)$ is non-essential in D if an only if v is reachable from u in $D - e$. Hedetneimi's characterization can be restated as follows: A strong digraph D is minimally strong if and only if D does not contain a non-essential arc. This simple characterization of MSDs will be useful to describe various methods for constructing MSDs by avoiding the addition of such non-essential arcs.

Chapter 2

The X-Join Composition

One approach to consider in the construction of MSDs is the use of a decomposition theory in which the members of the decompositions of MSDs are completely characterized. Cunningham's decomposition theory [5, 6] characterizes the members of the decomposition of strongly connected digraphs using the X-join composition of digraphs. This theory could be of further use in characterizing the members of the decomposition of MSDs. If the members of the decomposition of MSDs could be completely characterized, then all the MSDs on a given number of vertices could be constructed by generating all possible compositions of the members.

The X-join, or substitution, composition on which Cunningham's decomposition theory is based, is as follows. Let G_1 and G_2 be non-trivial "marked" digraphs, so that $V(G_1)$ and $V(G_2)$ each contain a marker vertex v (but are otherwise vertex-disjoint). The substitution composition of G_1 with G_2 is the digraph $G = G_1 * G_2$ where $V(G) = (V(G_1) - \{v\}) \cup (V(G_2) - \{v\})$ and $E(G)$ is defined as

$\{(x,y) : (x,y) \in E(G_1) \cup E(G_2), x \neq v \neq y\} \bigcup$

$\{(x,y) : ((x,v) \in E(G_1) \wedge (v,y) \in E(G_2)) \bigvee ((x,v) \in E(G_2) \wedge (v,y) \in E(G_1))\}$.

The marker vertex v is also called the *identifying* vertex. It can be verified that the X-join operation is associative and that it preserves strong connectedness.

Given a digraph D, let $\{V_1, V_2\}$ be a partition of $V(D)$ such that $V(D_1)$ and $V(D_2)$ can be defined as $V_1 \cup \{v\}$ and $V_2 \cup \{v\}$, where $v \notin V(D)$, in such a way that $D = D_1 * D_2$. Cunningham refers to $\{V_1, V_2\}$ as a *split* of D. If $|V_1| \geq 2$ and

$|V_2| \geq 2$, then $\{D_1, D_2\}$ is defined as a *simple decomposition* of D. D_1 and D_2 are the *members* of the decomposition.

A simple decomposition can be "strictly refined" by taking simple decompositions of D_1 and D_2. This is a *general decomposition.* Any decomposition can be depicted using a *decomposition tree.* (A tree is a connected acyclic undirected graph.) The decomposition tree's vertices represent the components of the decomposition, and the edges represent pairs of corresponding markers for the members of the decomposition. Figure 2.1 shows a simple decomposition of D. A general decomposition for D is represented by the decomposition tree T in Figure 2.2. Here, we call D the *underlying* digraph of T.

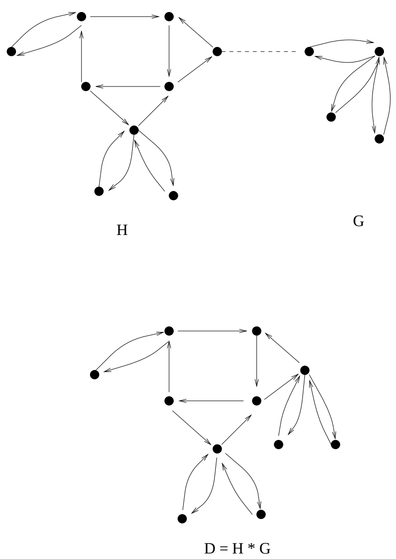

Figure 2.1: The X-join of H and G

A join is said to be *good* if it does not result in non-esential arcs. Otherwise, it is *bad.*

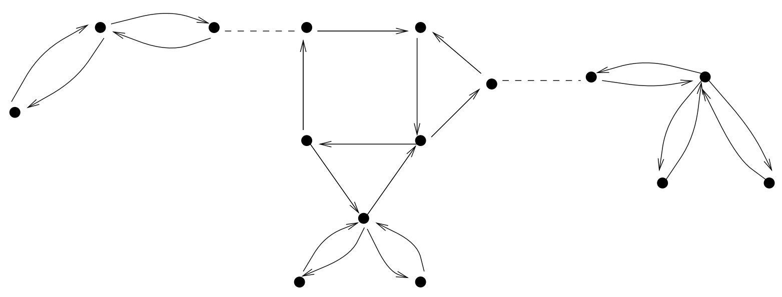

Figure 2.2: A general decomposition of D

2.1 Decomposition Theorem For MSDs

Cunningham observed in [5] that $H * G$ is strongly connected if and only if H and G are strongly connected. Based on this observation, the following theorem is proved in [12].

Theorem 1 *If H and G are non-trivial marked digraphs such that $H * G$ is minimally strongly connected, then H and G are MSDs.*

Corollary 1 *If H or G is not an MSD, then $H * G$ is not an MSD.*

If H and G are two distars joined at their centers, $H * G$ is not minimally strongly connected, as shown in Figure 2.3. Here, (a_1, b_1) is a non-essential arc since we can use the path $a_1 - b_2 - a_2 - b_1$ instead.

Thus, the converse of Theorem 1 is not necessarily true.

A *dicomplete* graph G is a non-trivial digraph in which $E(G)$ contains all the possible ordered pairs of distinct elements, *i.e.,* all possible arcs, of $V(G)$. If a digraph G on n vertices has an ordering $v_1, v_2, \ldots, v_n$ of $V(G)$ in which $E(G) = \{(v_i, v_j) : 1 \leq i < j \leq n\}$, G is called a *transitive tournament.* A sequence of non-trivial transitive tournaments $T_1, T_2, \ldots, T_k$ in which $k \geq 2$ and the last vertex in T_i is identified with the first vertex in T_{i+1}, for $1 \leq i \leq k-1$, and the last vertex of T_k is identified with the first vertex of T_1, forms a circle of transitive tournaments.

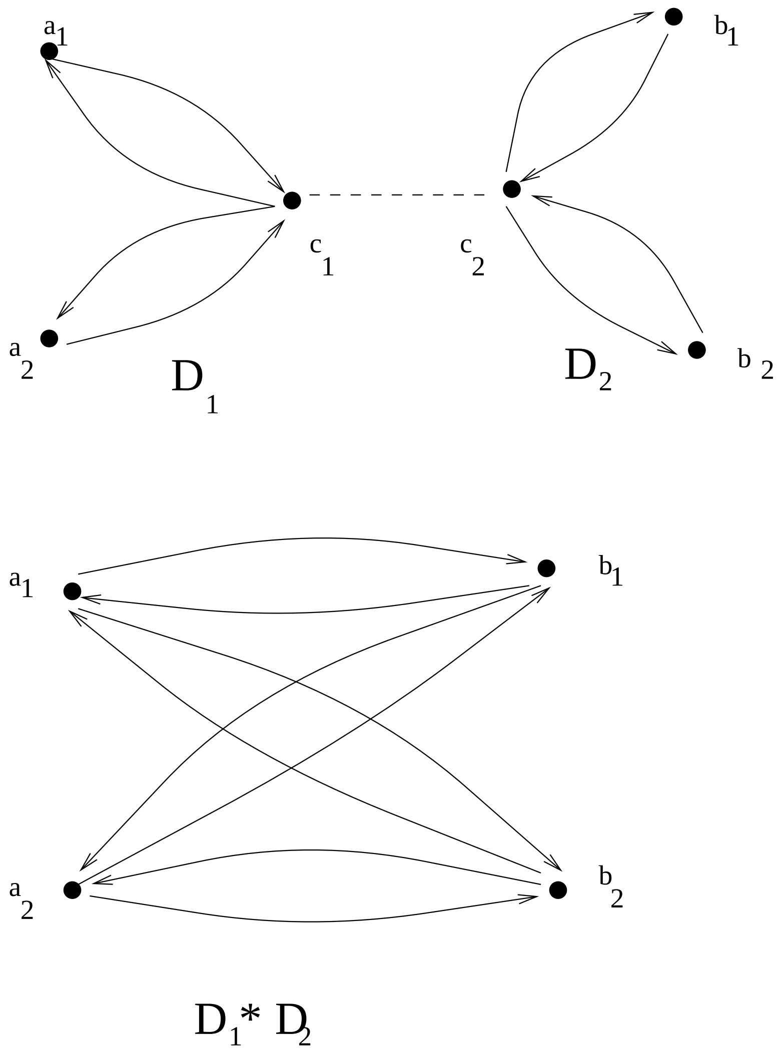

Figure 2.3: Join of two MSDs giving a non-MSD

A circle of transitive tournaments for which $|V(T_i)| = 2$ for $1 \leq i \leq k$ is a cycle. A decomposition X for a digraph D is said to be *minimal* if X is a decomposition of D having the fewest components. A decomposition is *trivial* if there is only one component of the decomposition. A digraph having only the trivial decomposition is called *prime* unless it is dicomplete, a distar, or a circle of transitive tournaments (just a cycle in the case of MSDs). Cunningham's main decomposition theorem states that each strongly connected digraph has a unique minimal decomposition in which each member is either prime, dicomplete, a distar, or a circle of transitive tournaments.

From Theorem 1 when we restrict the above decomposition to MSDs, we have the following theorem, proved in [12].

Theorem 2 *Any non-trivial MSD has a unique minimal decomposition in which each member is either prime, a distar, or a cycle.*

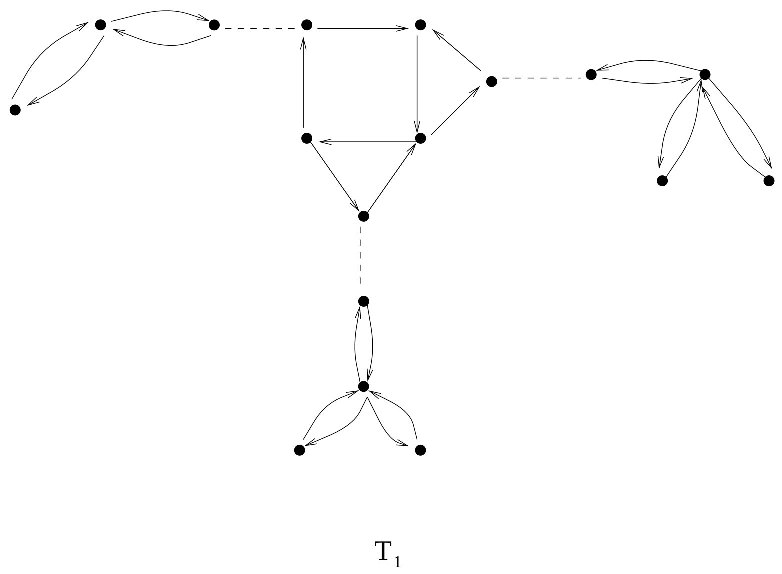

Figure 2.4: T_1 is a minimal decomposition of D

Based on Theorem 2, it is clear that all MSDs can be represented by decomposition trees of prime MSDs, distars, and cycles. Any decomposition tree of an MSD consisting of these components can be transformed into the minimal decomposition tree for this MSD by merging any cycles that are joined in the decomposition tree and by merging any distars which are joined center to ear. This is because the X-join of two cycles is a larger cycle, and the X-join of the center of a distar with an ear vertex of another distar is a larger distar as shown in the Figure 2.5.

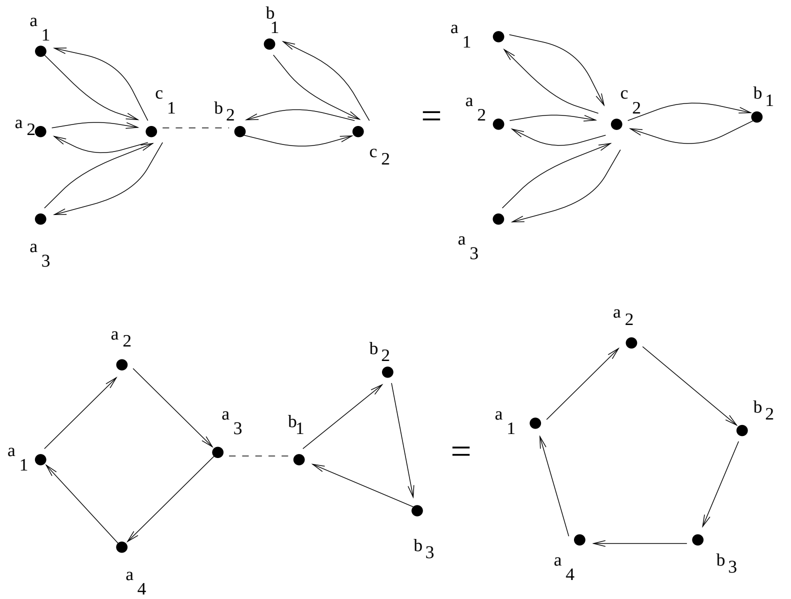

Figure 2.5: distar-distar join resulting in a larger distar (above); cycle-cycle join resulting in a larger cycle (below).

To use Theorem 2 to construct all the MSDs on a given number of vertices, we must develop rules on how to join these components, and we must also characterize the prime MSDs.

2.2 Characterization of the join between two MSDs

In this section, we will characterize the join between 2 MSDs based on the degrees of the identifying vertices.

Theorem 3 *If D_1 and D_2 are two MSDs joined with u and v as identifying vertices then:*

$D_1(u) * D_2(v)$ is bad iff $((od(u) > 1 \wedge id(v) > 1) \bigvee (id(u) > 1 \wedge od(v) > 1))$.

$\Rightarrow$ **direction**:

Case $(od(u) > 1 \wedge id(v) > 1)$: We have the join situation as shown in Figure 2.6. Since D_1 is an MSD, we have a path P_1 connecting u_2 to u_3. (Note that P_1 might pass through u.) Let w_1 be the first vertex on P_1 such that $(w_1, u) \in D_1$ (*i.e.*, $w_1 \neq u_3$ if and only if path P_1 passes through vertex u.)

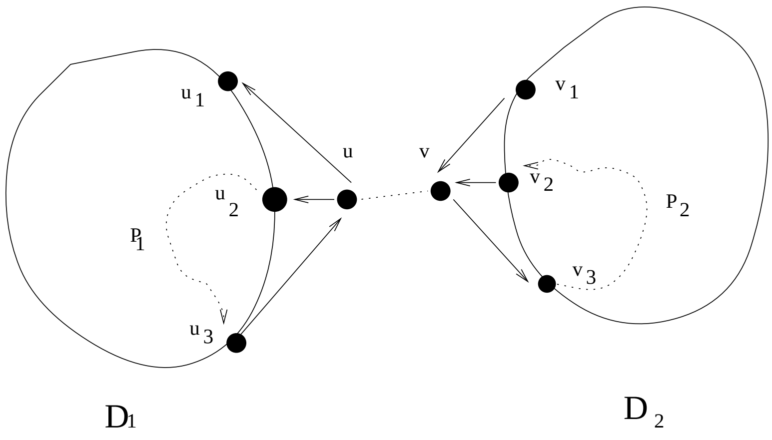

Figure 2.6: Join with $od(u) > 1$ and $id(v) > 1$

Similarly, since D_2 is an MSD, we have a path P_2 from v_3 to v_2. Now, let w_2 be the first vertex on P_2 such that $(w_2, v) \in D_2$ (*i.e.*, $w_2 \neq v_2$, if and only if path P_2 passes through v.)

As shown in Figure 2.7, note that in $D_1 * D_2$ we will have the edges (w_1, v_3) and (w_2, u_1) and thus have arc (v_1, u_1) non-essential because of the path $v_1 - u_2 - P_1{}' - w_1 - v_3 - P_2{}' - w_2 - u_1$ where $P_1{}'$ and $P_2{}'$ are subpaths of P_1 and P_2 from u_2 to w_1 and v_3 to w_2 respectively. Thus, the join is bad.

The proof is similar for the case $(id(u) > 1 \wedge od(v) > 1)$).) Thus, $D_1(u) * D_2(v)$ is bad.

$\Leftarrow$ **direction**:

We prove that

$((id(u) = 1 \vee od(v) = 1) \bigwedge (od(u) = 1 \vee id(v) = 1)) \Rightarrow D_1(u) * D_2(v)$ is good.

Case $(id(u) = 1)$: We prove that all the new arcs formed after the join in $D_1(u) * D_2(v)$, from $w_1 \in V(D_1)$ to $w_2 \in V(D_2)$, are essential.

Figure 2.8 shows this join situation where $id(u) = 1$.

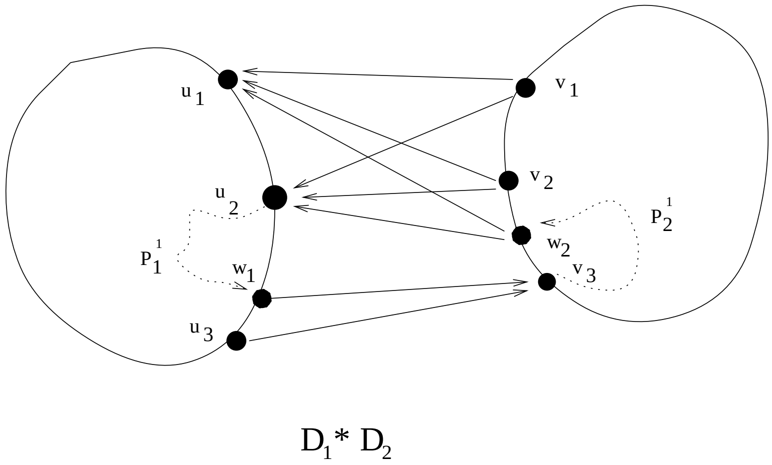

Figure 2.7: Bad join : $od(u) > 1$ and $id(v) > 1$

In D_1, consider the vertex u_1 which is the only vertex adjacent to u since $id(u) = 1$. In $D_1 * D_2$, the only way of reaching of vertices in $V(D_2)$ from vertices in $V(D_1)$ is by using the out-going arcs on u_1 as shown in the Figure 2.9. These are the only arcs going from $V(D_1)$ to $V(D_2)$.

Without loss of generality, consider the arc (u_1, v_1) and suppose that it is non-essential. That is, there is a path $u_1 - x_1 - x_2 \ldots x_p - v_1, p \geq 1$, where $x_i \in V(D_2), 1 \leq i \leq p$ and $x_p \in V_o$. That means, in D_2 we have a path $v - x_1 - x_2 \ldots x_p - v_1$ in addition to the arc (v, v_1). This means that (v, v_1) is non-essential in MSD D_2 which is a contradiction. Hence, (u_1, v_1) is essential in $D_1 * D_2$. Thus, arcs $(u_1, v_i), 1 \leq i \leq k$ are essential in $D_1 * D_2$. The proof is similar for the case $od(v) = 1$ and also for the new arcs going in the other direction. The existing arcs are still essential because no new paths have been added to make them non-essential.

Hence, all arcs in $D_1 * D_2$ are essential and $D_1 * D_2$ is strong and hence an MSD. Thus, the join $D_1(u) * D_2(v)$ is a good join.

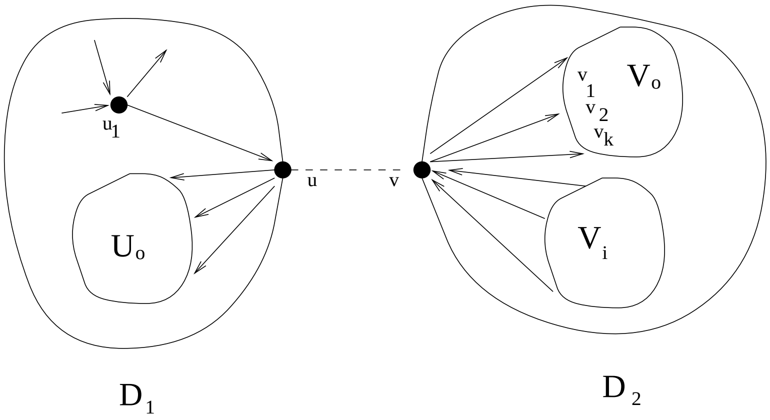

Figure 2.8: Join between $D_1(u)$ and $D_2(v)$ when id(u) = 1.

2.2.1 Summary of the bad joins in the context of the components of a decomposition tree T

- Do not join the centers of distars.
- Do not join the center of a distar to a non-carrier.
- Do not join 2 non-carriers u and v of MSDs D_1 and D_2 where $id(u) > 1$ and $od(v) > 1$.

So, the following is the summary of the only bad joins:

- NC * NC
- NC * NCI
- NC * NCO
- NCI * NCO

Also notice that NCI * NCI and NCO * NCO are good joins.

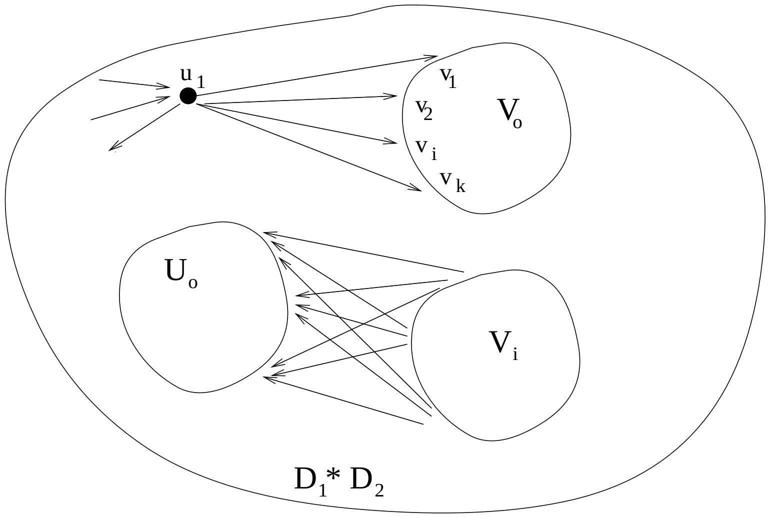

Figure 2.9: $D_1(u) * D_2(v)$ when id(u) = 1.

Chapter 3

Characterization of Cunningham Decomposition Trees For MSDs

In this chapter, given a decomposition tree T, we show how to determine whether or not the underlying digraph [1] D of T is an MSD. Recall that the X-join operation is associative.

3.1 Some terminology for decompositon trees

Cascading join: A series of joins $J_1, J_2, \ldots, J_{k-1}$, say, $D_1(u_1) * D_2(u_2), D_2(u_3) * D_3(u_4), \ldots, D_{k-1}(u_{2k-3}) * D_k(u_{2k-2})$ respectively in a decomposition tree T is said to be a *cascading join* if $(u_{2i-2}, u_{2i-1}) \in E(D_i)$, $1 < i < k$, as shown in Figure 3.1

Note that a join may may affect the degrees of vertices in the neighborhood of the identifying vertices, so as joins are made it may happen that a good join becomes bad!

Figure 3.2 shows a cascading join in which each join is good taken individually, but $D_1 * D_2 * D_3$ contains non-essential arcs.

Real and virtual arcs: In a cascading join, the join "arcs" (represented by dashed lines) are called *virtual* arcs and the remaining arcs (represented by solid lines) are called real arcs. For example: in Figure 3.1 $(u_2, u_3), (u_4, u_5) \ldots$ are real arcs and $(u_1, u_2), (u_3, u_4) \ldots$ are virtual arcs.

[1] Recall that the underlying graph D of a decomposition tree T is the digraph of which T is a decomposition tree.

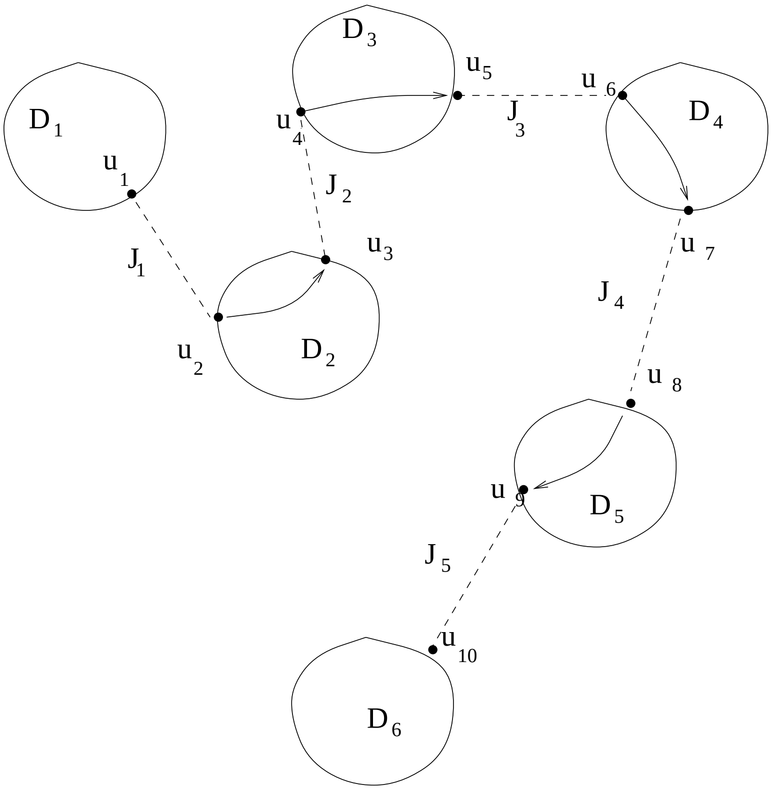

Figure 3.1: A cascading join

Alternating Virtual Path (AVP): In a cascading join, a "path" starting and ending with a virtual arc, and comprising of virtual and real arcs alternately is called an *alternating virtual path.*

In Figure 3.1, the path $u_1 \ldots u_2 - u_3 \ldots u_4 - u_5 \ldots u_6 - u_7 \ldots u_8 - u_9 \ldots u_{10}$ is an alternating virtual path.

Even and odd vertices with respect to a vertex u on an AVP: A vertex v is called *even (odd)* with respect to a vertex u if it is at an even (odd) distance from u on the AVP.

In view of the simple characterization of good and bad joins provided by Theorem 3, we assume that all the joins in T, taken individually, are good.

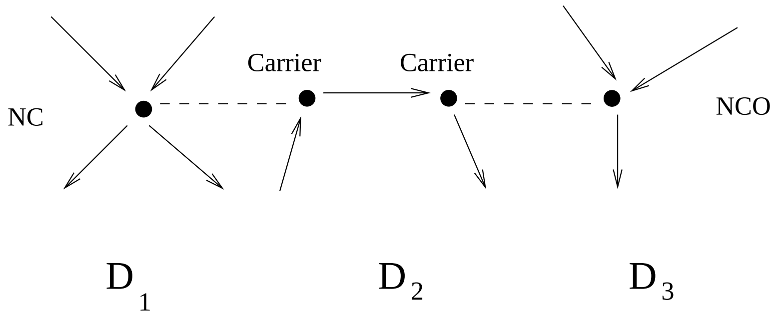

Figure 3.2: Cascading creates a bad join.

We give a procedure for detecting a bad join occurring in a cascading join. We will consider three cases where we can have a bad join after making a series of joins in a cascading join. Here we consider the cases where the search for the bad joins starts with an NCI. But, this approach can be generalized for the cases of NCO and NC in the obvious way.

We show that all the cases of bad joins can be detected by the same search procedure – namely the search procedure used to detect Problem I cases.

3.2 Problem I case

If u is an NCI on a cascading join and there is a vertex v odd with respect to u, such that $id(v) > 1$, *i.e.,* v is either an NCO or NC, we have a bad join as shown in the Figure 3.3. By taking v to be the closest such vertex to u, we may assume that all odd vertices with respect to u, on the AVP P between u and v, have $id = 1$.

Clearly, there will be a bad join here because, after making the joins J_2 and J_3, we have the situation as in Figure 3.4 where $id(u_1) > 1$. Recall that, $id(u_1) = 1$, before making joins J_2 and J_3 (by hypothesis). But, after joins J_2 and J_3, u_1 is either an NCO or NC, so the join J_1 with identifying vertices u and u_1 is bad. The same situation occurs no matter what the distance from u to v is, as long as it is odd.

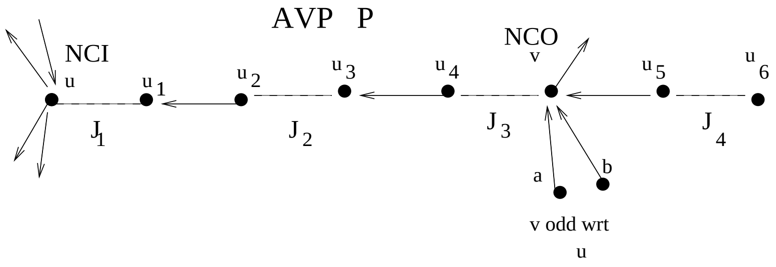

Figure 3.3: Problem I situation

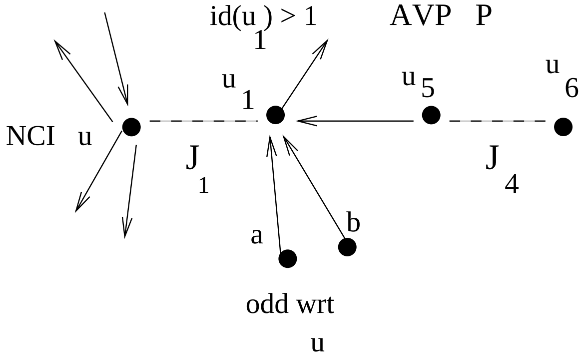

Figure 3.4: Bad join created by other joins, in Problem I case.

3.3 Detecting the Problem I case

For an NCI u, a vertex v odd with respect to u such that $id(v) > 1$ on the AVP P can be found by taking the unique virtual arc incident on u reaching a vertex u_1 (odd with respect to u). If $id(u_1) > 1$, then we are done. Otherwise, $id(u_1) = 1$, and we take the unique real arc coming into u_1 reaching a vertex u_2. Now, from u_2 we will take the unique virtual arc and so on. We continue this way until we find a v odd with respect to u and $id(v) > 1$ as above or we reach the end of the AVP P.

Now, suppose that there is no such v on the AVP P starting at u. That is, all the vertices v which are odd with respect to u on P have $id = 1$, *i.e.,* they are either carriers or NCIs as in Figure 3.5.

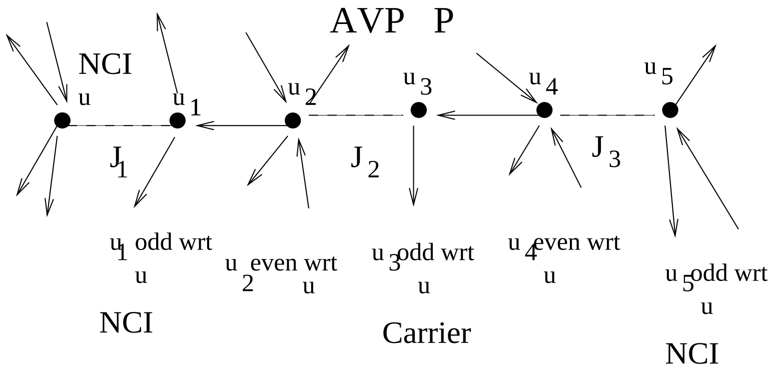

Figure 3.5: No Problem I case on AVP P.

Then, all AVPs starting at the any vertex w that is even with respect to u, also have id(v) = 1 for all odd vertices v with respect to w on P since these paths are sub-paths of P.

Thus, we will not have to check again for Problem I starting at those vertices w which are even with respect to u on P.

3.4 Problem II case

Once we have made sure that there is no Problem I case on an AVP P, let us consider if a potential problem may still exist for any of the odd vertices (with respect to u) on the AVP P.

Let v be an odd vertex with respect to u on P, and let the unique join associated with v be J. Since v is an odd vertex (with respect to u), and it is assumed that Problem I does not arise, we have $id(v) = 1$, *i.e.,* v is either a carrier or an NCI.

Since $od(v) \geq 1$, we consider the situation, where one of the out-going edges from v is on another cascading join, as shown in Figure 3.6.

Different cases for the identifying vertices v and w of J are shown in Figure 3.7

The worst that can happen to the join J after making all the joins in the new alternating path (from w_1 to w_k) is that it can increase the outdegree of v. If v is

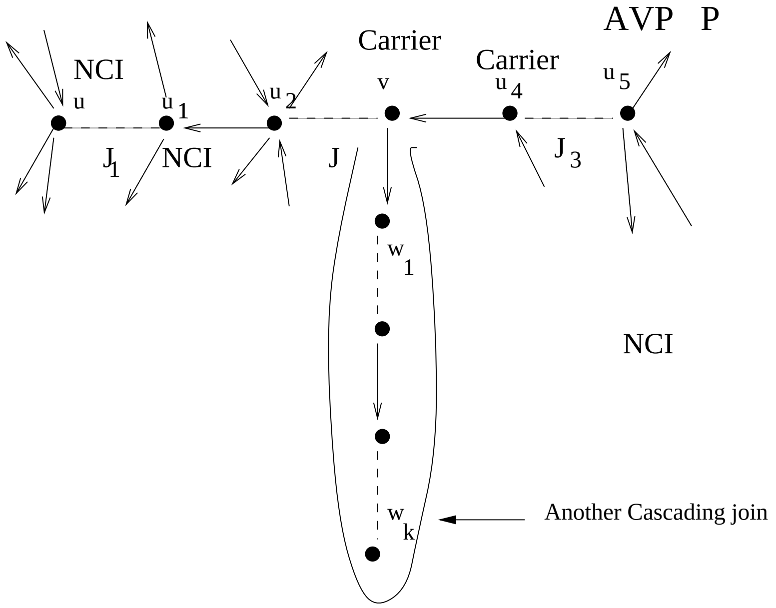

Figure 3.6: Problem II case with v on two AVPs

already an NCI – recall that v is either a carrier or NCI – then no harm can be done to join J, since $od(v) > 1$ already. So, we have a problem only when v is a carrier before making the joins in the new cascade and after making the joins $od(v) > 1$ as shown in Figure 3.8.

In general for the problem in Figure 3.8 to occur, u_2 should be a NC or NCO (cases (3) and (4) in Figure 3.7).

3.5 Detecting the Problem II case

Suppose, tentatively, that after figuring out that Problem I case is non-existent, we marked all odd vertices with respect to u on P as good – indicating that we do not have to worry about searching from these vertices for problem situations. We will now try to verify whether we will miss any Problem II situations because of this greedy approach.

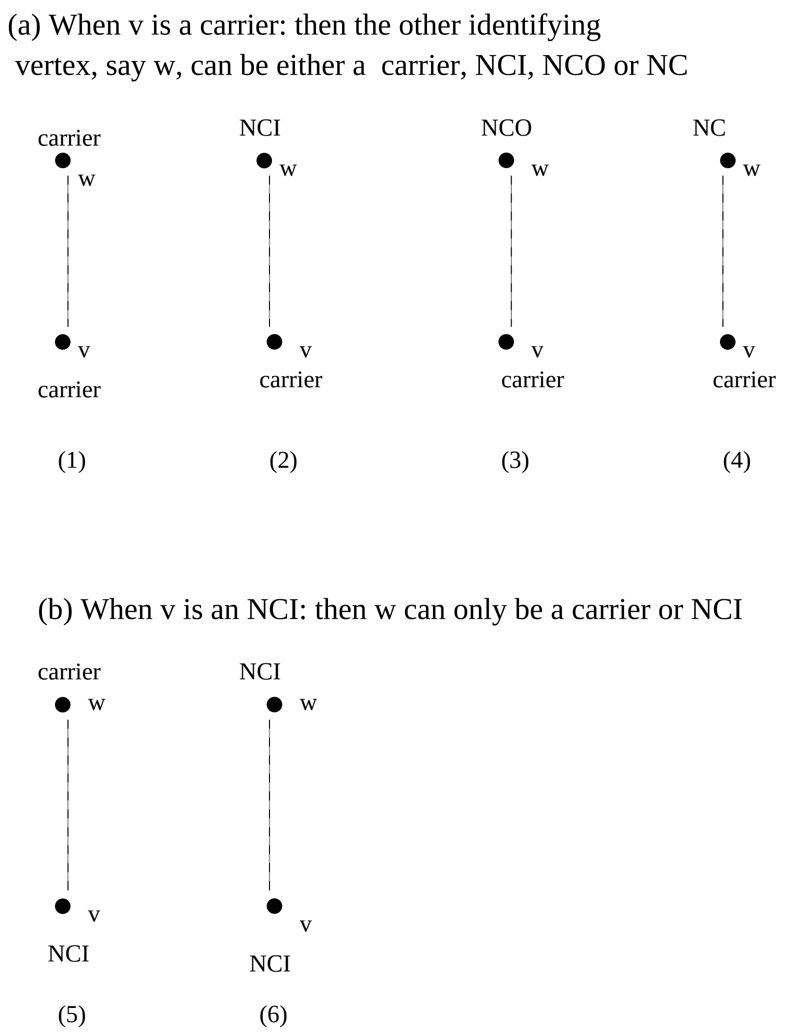

Figure 3.7: In Problem II case, different cases for the identifying vertices v and w

In Figure 3.8, if Problem II were to occur, on AVP P_2 there should be a vertex w, such that $od(w) > 1$ and w is odd with respect to w_1 as shown in Figure 3.9.

But then, while searching for Problem I situations, we will start a search at w looking for odd vertices (with respect to w) that have $id > 1$. During that search, we will detect u_2, which is either a NC or NCO – in both cases $id(u_2) > 1$. Hence, the absence of any Problem I situations also means that this kind of Problem II situations are absent. Hence, the greedy approach mentioned above does not miss any problems of this type.

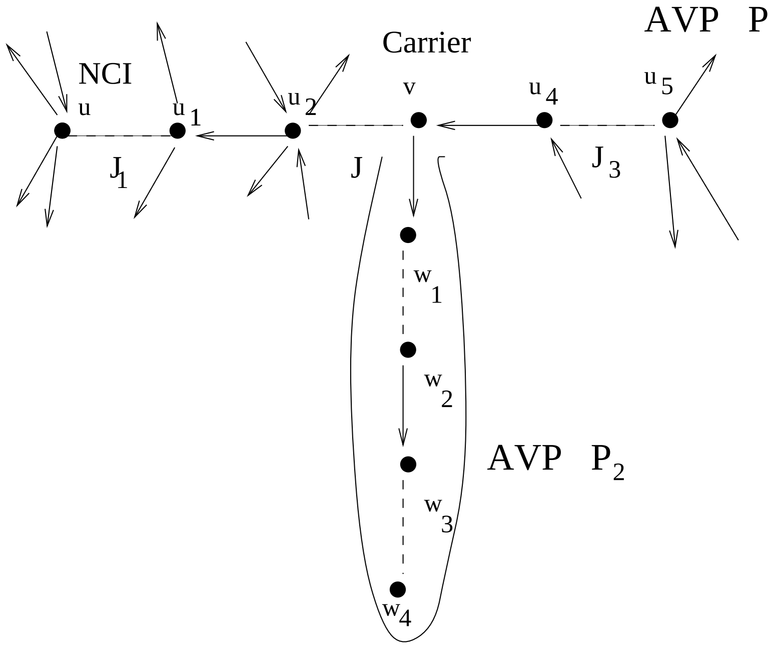

Figure 3.8: Problem II case, $od(v) > 1$ after making joins on P_2 and u_2 is an NC (or NCO). Hence, a bad join.

3.6 Problem III case

Supoose that we have discovered that Problem I case does not occur on P. Now, we will see if we can mark all the even vertices w with respect to u on P as good.

Since, there is no problem I case and if w is joined with vertex y, which is odd with respect to u, vertex y is either a carrier or NCI. Different cases for the join involving w and y are shown in Figure 3.10.

Now, suppose that there is another cascading join that affects the indegree or outdegree of w such that it makes the join involving w and y bad. If y is a carrier, then we will not have any problem. So, the necessary condition for having this problem is that y should be a non-carrier. But, we already know that y can only be a carrier or NCI (since it is odd with respect to u). So, the necessary condition for this problem to occur is y is a NCI. In Figure 3.10, the only situations where y is a NCI is (2) and (4).

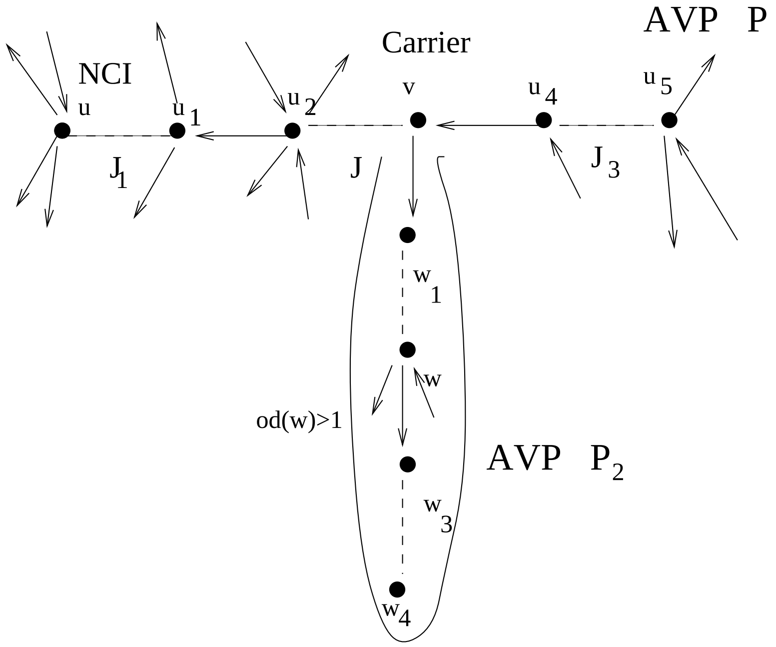

Figure 3.9: Problem II case can be detected using Problem I search procedure

In Figure 3.11, if after making the joins in P_2, only $od(w)$ increases such that after the joins $id(w) = 1$ and $od(w) > 1$ *i.e.,* w is a NCI, it doen not hurt because y is a NCI and NCI*NCI join is good. So, the only problem case occurs if after making some of the joins in P_2, the indegree of w increases such that $id(w) > 1$. (Then, we will have a NCO*NCI or NC*NCI join – which are bad – depending on whether w was a carrier or NCI before those joins.)

3.7 Detecting the Problem III case

If after making all the joins the new cascade increases the $id(w)$, then on AVP P_2 as shown in Figure 3.12, we will have an odd vertex ,say x, (with respect to w_1), with $id(x) > 1$ *i.e.,* x is a NCO or NC. But, then while searching for Problem I situations (NCO or NC version), we will also start a search at x, looking for odd vertices z

with respect to x such that $od(z) > 1$. Here, since y is odd with respect to x, and also since y is a NCI *i.e.,* $od(y) > 1$, we will find that there is a Problem I situation.

So, even if we mark all the even (with respect to u) vertices on P as good – indicating that we do not have to search from these vertices for problem situations – we will not miss this Problem III type situation.

The same procedure can be extended to search for Problem I starting from NCOs and NCs. In the search for Problem I starting at an NCO u, there will be a bad join if there is a odd vertex v with respect to u such that $od(v) > 1$. If u is an NC, then there will be a bad join if $id(v) > 1$ or $od(v) > 1$ *i.e.,* if v is a non-carrier.

3.8 Characterization of Cunningham's decomposition trees

Note: The absence of Problem I cases in a decomposition tree T implies that all the joins, taken individually, are good. (Since individual joins correspond to AVPs of length 1).

Theorem 4 *If T is a decomposition tree with components as MSDs in which there are no Problem I cases, then the underlying graph D of T is an MSD.*

We give the proof by induction on the number of components, n, in T.

Basis: For $n = 2$, the theorem is trivially true (since all the individual joins are good).

Induction: Let the theorem be true for all decomposition trees with $n = k - 1$. Now, consider T with $n = k$ of which D_1 and D_2 are two components. Consider the decomposition tree T' obtained after making the join $D_1 * D_2$. We know that $D_1 * D_2$ is a good join (since all joins taken individually are good). Hence, T' contains $k - 1$ components all of which are MSDs. Also, notice that there will not be any Problem I cases in T'. Hence, by induction the underlying graph D of T is an MSD.

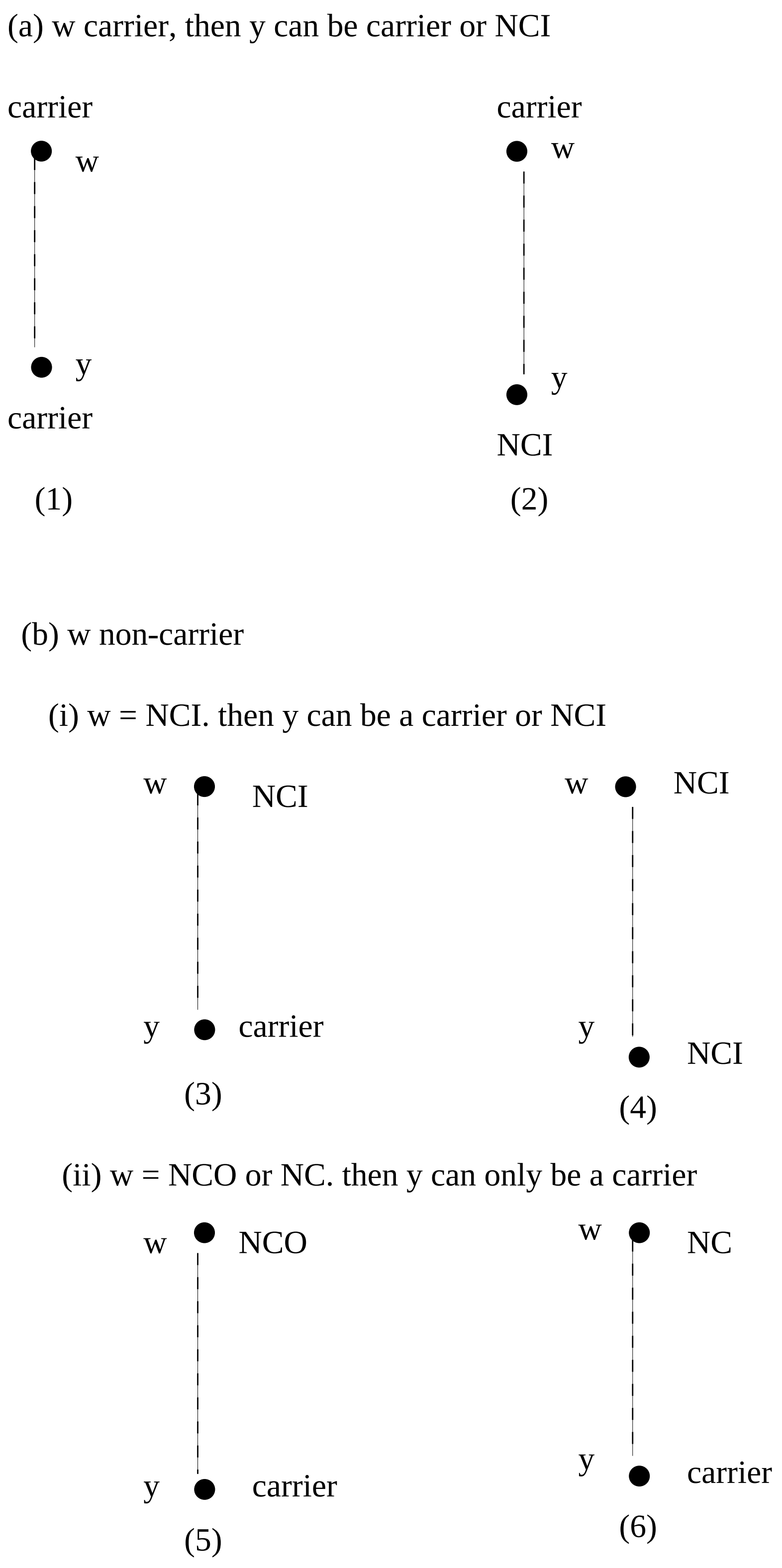

Figure 3.10: Problem III, different cases for the identifying vertices w and y

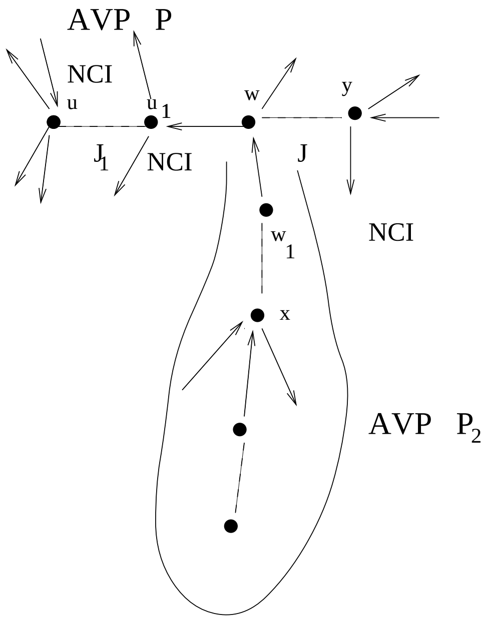

Figure 3.11: Problem III case with w on two AVPs

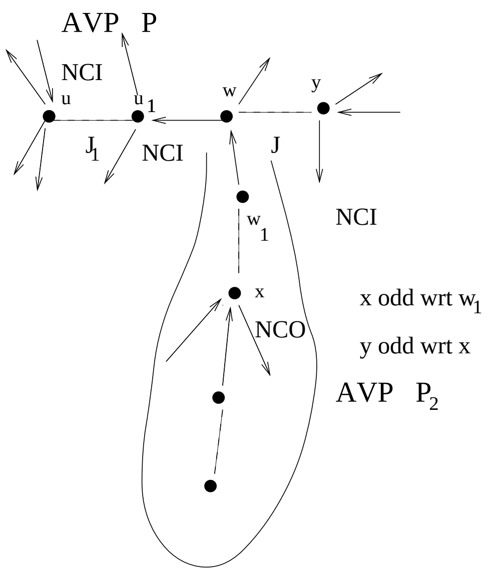

Figure 3.12: Problem III case can be detected using Problem I search procedure

Chapter 4

A New Method for Constructing MSDs

In this chapter we give a new method to generate larger MSDs from smaller MSDs.

Many classes of graphs and digraphs (e.g. connected, 2-connected graphs, strong digraphs, strong blocks) have a constructive characterization called an *ear-decomposition* of the following kind:

Take a "basic" graph or digraph (e.g., tree, cycle) and add "ears" (e.g., edges, paths) repeatedly in a well defined way.

We give an ear decomposition for MSDs where the basic digraphs are smaller MSDs, *i.e.,* MSDs with fewer vertices than the generated MSDs. The ears are paths of length > 1, added systematically between vertices not necessarily distinct, satisfying certain conditions.

Theorem 5 *Every MSD contains at least two carriers.*

See [1, p. 31] or [12] for a proof.

4.1 Generating larger MSDs from smaller MSDs

Consider an MSD D and two vertices $u, v \in V(D)$.

case 1: $(u, v) \in E(D)$.

Since $(u, v) \in E(D)$, adding a path P, $|P| > 1$, between u and v in $D - \{(u, v)\}$ produces another (larger) MSD D' as shown in Figure 4.1. Here, D' is an MSD because it is strongly connected, all of the existing arcs (from D) in D' remain essential and all the newly added arcs on path P are essential as they are incident on at least one carrier.

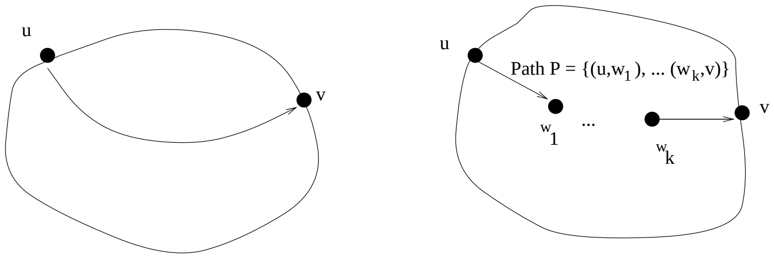

Figure 4.1: Replacing an existing arc in D with a path.

case 2: $(u, v) \notin E(D)$.

If addition of a path P between u and v in D does not make any of the arcs $e \in E(D)$ non-essential, $D \bigcup P$ is a MSD. Notice that $|P| > 1$, because, since D is an MSD, there is a path P' from u to v already existing in D and if $|P| = 1$ we are adding arc (u, v) to D (already containing a u, v path P', where $|P'| > 1$), and hence we will not obtain an MSD. Also, when $|P| > 1$, all the newly added arcs (on path P) are essential in $D \bigcup P$ because they are incident on at least one carrier. Also, all the old arcs remain essential. Hence, $D \bigcup P$ is a (larger) MSD as shown in Figure 4.2.

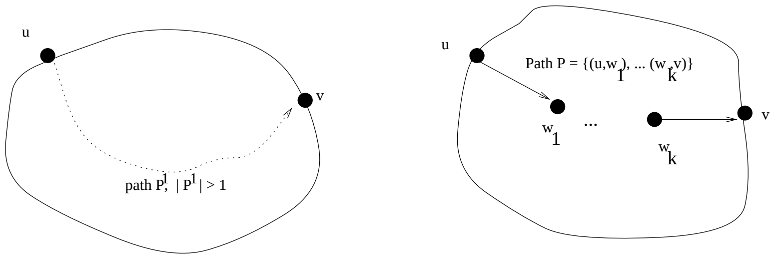

Figure 4.2: Adding a path between non-adjacent vertices.

Theorem 6 *Every 2-connected MSD D' on $n \geq 3$ vertices, can be generated from smaller MSDs by adding paths according to cases 1 or 2 above.*

From Theorem 5, every MSD D' has at least 1 carrier, say u. Now, let v, w be the two vertices in D' such that $(w, u), (u, v) \in E(D')$. Note that $v \neq w$ or else v is a cut-vertex (in view of $n \geq 3$), a contradiction. Now consider the digraph $D'' = D' - \{(w, u), (u, v)\}$.

case (a): No path from w to v exists in D''.

Let $D = D'' \bigcup \{(w, v)\}$. Here, D is an MSD because $D' = D'' \bigcup \{(w, u), (u, v)\}$ is an MSD (by hypothesis) and u is a carrier. Also note that $|V(D)| < |V(D')|$. In this case, we can generate D' from D by using case 1 – by removing the existing (w, v) arc and adding a path of length 2 between w and v in D.

case (b): A path P from w to v exists in D''.

Let $D = D''$. Hence, D is an MSD, $(w, v) \notin E(D)$ and also adding (w, v) to D does not make any existing arc $e \in E(D)$ non-essential – since $D' = D \bigcup \{(w, u), (u, v)\}$ is an MSD (by hypothesis). Notice that $|V(D) < |V(D')|$. Thus, we can generate D' from D by using case 2 – adding a path of length 2 between w and v in D.

Note 1: Case 1 and case 2 generate only MSDs without cut-vertices (*i.e.,* 2-connected MSDs). To generate all MSDs, we have to add another phase, in which we take smaller MSDs and attach cycles (to all vertices) to get larger MSDs – with cut-vertices. (This corresponds to the case of adding a path that begins and ends with the same vertex u.)

Note 2: In the actual generation of larger MSDs from a smaller MSD D, whenever we find two vertices u and v in MSD D satisfying case 1 or case 2, we can add paths of length 2, 3, 4, ... to generate larger MSDs on $|V(D)|+1, |V(D)|+2, |V(D)|+3, \ldots$ vertices.

Note 3: In the actual implementation of case 1, we can let u and v to be non-carriers to increase efficiency. Clearly, this will not exclude any larger MSDs from being generated. Because of this modification, cycles have to be generated explicitly.

4.2 *nauty*

Brendan McKay's *nauty* (no automorphisms, yes?) [13] is a software package that can quickly calculate the automorphism group for a graph or digraph. Many functions are available in *nauty* that make implementing our method for generating MSDs easier. For example, *nauty* can take a digraph as input and output a canonical representation for that digraph. A digraph is stored as an array of *sets* where each set is made up of a number of setwords (actually integers). Each set represents a row of the adjacency matrix bitwise; set i encodes all the vertices adjacent from vertex i. The ability of *nauty* to compute a canonical form of a digraph is used in our generation procedure to test if a digraph has already been generated.

4.3 Counting the labeled MSDs

An *automorphism* of a digraph is an isomorphism of the digraph onto itself. Every automorphism of a digraph D_1 is a permutation of the vertices of D_1 in which all the adjacencies are preserved. The set of vertices of D_1 is given an initial order, and this initial order and a permutation of these vertices form the one-to-one correspondence. If the one-to-one correspondence preserves adjacency, then that permutation is an element of D_1's *automorphism group.* D_1's automorphism group is a subset of all of permutations of D_1's vertices. For example, for any directed cycle C with n vertices, there will be n members of C's automorphism group because there are n permutations of C's vertices that preserve adjacency.

It is possible to calculate the number of labeled graphs on a graph G with n vertices if we know the number of elements of G's automorphism group. The number of different labeled graphs on G is exactly the number of permutations of G's n vertices divided by the number of members of G's automorphism group; see [10, Eq. 1.3.3] for example. Given a graph G as input *nauty* can output the order s of G's automorphism group. Thus, to calculate the number of labeled digraphs for every MSD we simply evaluate $n!/s$ for every unlabeled MSD under consideration.

4.4 Results

We obtained the results shown in the following tables for the numbers of MSDs and those that are 2-connected.

Number of Labeled MSDs								
# arcs	*# vertices*							
	2	3	4	5	6	7	8	9
2	1							
3		2						
4		3	6					
5			36	24				
6			16	360	120			
7				560	3600	720		
8				125	13530	37800	5040	
9					9270	282660	423360	40320
10					1296	435330	5562480	5080320
11						170352	16479120	107533440
12						16807	13527416	552363840
13							3497704	833865984
14							262144	427305312
15								79951104
16								4782969
totals	1	5	58	1069	27816	943669	39757264	2010923289

Number of Labeled MSDs (contd...)				
# arcs	*# vertices*			
	10	11	12	13
10	362880			
11	65318400	3628800		
12	2089281600	898128000	39916800	
13	17238614400	41333846400	13172544000	479001600
14	43947217440	516503433600	839091052800	205491686400
15	39645728640	2085773215680	15156329126400	17558122982400
16	14045175990	3081555338400	92125604838960	441473201769600
17	2021536170	1848678422580	211331781682560	3873428230351680
18	100000000	485585522730	201422926147080	13213679396037120
19		56150407280	86629491927000	19126187109263520
20		2357947691	17740944738132	12703611522012840
21			1702048061340	4138209797044752
22			61917364224	685983500260344
23				55964687223744
24				1792160394037
totals	119153235520	8118839891161	627023347399296	54258093698028037

Number of Unlabeled MSDs									
# arcs	*# vertices*								
	2	3	4	5	6	7	8	9	10
2	1								
3		1							
4		1	1						
5			2	1					
6			2	4	1				
7				7	6	1			
8				3	27	9	1		
9					23	70	12	1	
10					6	131	169	16	1
11						66	559	344	20
12						11	571	1970	662
13							191	3479	5874
14							23	2229	17109
15								541	18509
16								47	8226
17									1514
18									106
totals	1	2	5	15	63	288	1526	8627	52021

Number of Unlabeled MSDs (contd...)		
# arcs	*# vertices*	
	11	12
11	1	
12	25	1
13	1159	30
14	15526	1947
15	69845	37072
16	120582	246971
17	87963	646339
18	28879	732150
19	4217	385484
20	235	98146
21		11724
22		551
totals	328432	2160415

Number of Labeled 2-Connected MSDs									
# arcs	*# vertices*								
	2	3	4	5	6	7	8	9	10
2	1								
3		2							
4			6						
5			12	24					
6				180	120				
7				20	2160	720			
8					2370	25200	5040		
9					30	85260	302400	40320	
10						15750	2236080	3810240	362880
11						42	1829520	51649920	50803200
12							82376	109045440	1132488000
13							56	25686864	4820860800
14								381528	3427129440
15								72	287340480
16									1644390
17									90
18									
totals	1	2	18	224	4680	126972	4455472	190614384	9720629280

Number of Labeled 2-Connected MSDs(contd...)			
# arcs	*# vertices*		
	11	12	13
11	3628800		
12	718502400	39916800	
13	24449040000	10777536000	479001600
14	182080483200	530135020800	171243072000
15	287928527040	6295278528000	11678777510400
16	79528818600	18594847600560	207084246969600
17	2812077180	12373359768000	1021931406175680
18	6760050	1520855779080	1347770842498080
19	110	25297054200	420646459147200
20		26868732	25634292100560
21		132	215210427432
22			104087412
23			156
24			
totals	577527837380	39350618072304	3035133060990120

Number of Unlabeled 2-Connected MSDs									
# arcs	*# vertices*								
	2	3	4	5	6	7	8	9	10
2	1								
3		1							
4			1						
5			1	1					
6				2	1				
7				1	4	1			
8					7	6	1		
9					1	24	9	1	
10						11	74	12	1
11						1	89	173	16
12							18	464	371
13							1	222	1824
14								24	1938
15								1	465
16									33
17									1
18									
totals	1	1	2	4	13	43	192	897	4649

Number of Unlabeled 2-Connected MSDs (contd...)			
# arcs	*# vertices*		
	11	12	13
11	1		
12	20	1	
13	701	25	1
14	5843	1252	30
15	11918	16114	2079
16	6074	56934	39507
17	856	55862	222913
18	41	15881	384327
19	1	1458	206252
20		52	36453
21		1	2328
22			62
23			1
24			
totals	25455	147580	893953

4.5 Checking the results

We can partially verify the results shown in these tables by observing that on p vertices, the fewest number of arcs in an MSD D will equal p and D will be a cycle having $(p-1)!$ labeled versions. Also, on p vertices the maximum number of arcs in an MSD D will equal $2(p-1)$ and D will be a symmetrically directed tree. Then, Cayley's theorem [10, Eq. 1.7.2] that the number of labeled trees having p vertices is p^{p-2} implies that the number of MSDs having $2(p-1)$ arcs is p^{p-2}. Our results are consistent with these predictions. Our results are also consistent for $p \leq 9$ found by Mayhorn using a different method [12].

Apart from the above checks, we have also verified our results by number of arcs using the identity $log(C'(x)) = B'(xC'(x))$; see [10, Eq. 1.3.3]. Here, $C(x)$ is a generating function where the coefficient of $x^i/i!$ is the number of labeled MSDs on i vertices. In order to check the number of labeled MSDs by number of arcs, the coefficient of $x^i/i!$ will be a polynomial in y where the coefficient of y^j in this polynomial is the number of labeled MSDs on i vertices and j arcs. Similarly, $B(x)$ corresponds to the numbers of 2-connected MSDs. After checking the numbers through 12 vertices, we used the numbers of labeled 2-connected MSDs on 13 vertices to calculate the numbers of labeled MSDs on 13.

Chapter 5

Conclusion

We have proposed a new method for generating larger MSDs from smaller seed MSDs by systematically adding paths between qualified vertices. The paths are so added that all the existing arcs are still essential. The newly added arcs are essential because they are incident to or from at least one carrier in the added path. To be more efficient, once a pair of qualifying vertices is found in a seed MSD D for adding a path in between them, we generate all the larger MSDs on $|V(D)+1|, |V(D)+2|, \ldots$ vertices. Also, for efficiency reasons, in case 1 of the generation part, we require that u and v be both non-carriers. Our method is convenient for generating 2-connected MSDs since we have to use only case 1 and case 2 for this, omitting the phase of attaching cycles at the vertices (which allows for cut-vertices). We have catalogued and counted all MSDs through 12 vertices and those that are 2-connected through 13 vertices. After verifying our results through 12 vertices, the number of labeled MSDs on 13 vertices by arc numbers have been calculated using the labeled 2-connected MSD numbers.

We also gave a characterization of Cunningham's decomposition trees for MSDs under the X-join (substitution) composition. To use this characterization to generate all MSDs, the X-join prime MSDs should be characterized. We are not able to characterize the X-join prime MSDs, but using our characterization of Cunningham's decomposition trees for MSDs under the X-join (substitution) composition we are able to catalogue all X-join prime MSDs through 11 vertices.

We also found that a Tutte-style decomposition for 2-connected MSDs exists where the components of the decomposition are directed cycles, strongly oriented bonds and strongly oriented 3-connected graphs. In this composition oppositely-oriented arcs from two components – known as "virtual" arcs – are "joined" such that after the join the ends of these edges are identified and the edges themselves disappear. To generate 2-connected MSDs using this approach, the non-essential arcs have to be virtual so that they will disappear after the join. The apparent difficulty here is to identify which arcs are non-essential and which are not, in a strongly oriented 3-connected graph. For example, in Figure 5.1, in the strongly oriented 3-connected graph D, arc e is essential but it is not obvious as it is not the only incoming arc or the only outgoing arc at either end. This decomposition would be useful if a good characterization of non-essential arcs could be found for strongly oriented 3-connected graphs.

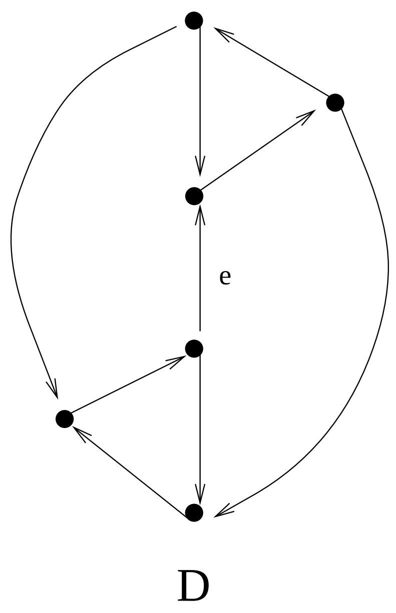

Figure 5.1: An essential arc in a strongly-oriented 3-connected graph.

Bibliography

[1] C. Berge, *Graphs and Hypergraphs*, American Elsevier, New York, 1973.

[2] R. Brualdi and R. Manber, On strong digraphs with a unique minimally strong subdigraph. *Discrete Math.* **71** (1988) , 1–7.

[3] G. Chartrand, D. Erwin, M. Raines, and P. Zhang, Strong distances in strong digraphs. *J. Combin. Math. Combin. Comput.* **31** (1999), 33–44.

[4] Z. Chen and F. Zhang, Bounds of the longest directed cycle length for minimal strong digraphs. *Discrete Math.* **68** (1988), 9–13.

[5] W. H. Cunningham, Decomposition of directed graphs, SIAM *J. Alg. Disc. Meth.* **3** (1982), 214–228.

[6] W. H. Cunningham and J. Edmonds, A Combinatorial Decomposition Theory, *Canad. J. Math.* **32** (1980), 734–765.

[7] D. P. Geller, Minimally strong digraphs, *Proc. Edinburgh Math. Soc.* **17** (1970), (2), 15–22.

[8] M. Grotschel, On minimal strong blocks, *J. Graph Theory* **3** (1979), 213–219.

[9] F. Harary, *Graph Theory*, Addison Wesley, Reading MA, 1969.

[10] F. Harary and E. M. Palmer, *Graphical Enumeration*, Academic Press, New York, 1973.

[11] S. Hedetneimi, Characterizations and constructions of minimally 2-connected graphs and minimally strong digraphs, *Proc. Second Louisiana Conf. on*

Combinatorics, Graph Theory and Computing, Louisiana State Univ., Baton Rouge, La., (1971) 257–282.

[12] Mayhorn, S. G. *Construction of Minimally Strongly Connected Digraphs*, Master's Thesis, Univ. of Georgia, Athens, (1998).

[13] B.D. McKay, *nauty* user's guide (Ver. 1.5), technical report TR-CS-90-02, Computer Science Dept., Australian National University, 1990. http://cs.anu.edu.au/people/bdm

[14] T. Sasao and J.T. Butler, Comparison of the worst and best sum-of-products expressions for multiple-valued functions, *IEEE International Symposium on Multiple-Valued Logic (1997), Nova Scotia, Canada*, 55–60.

[15] T. Sasao and J.T. Butler, Worst and best irredundant sum-of-products expressions, *IEEE Trans. on Computers* (to appear).

Appendix A

The X-join or Cunningham Prime MSDs

In this appendix, all the X-join prime MSDs through 9 vertices are given. The table below gives the number of X-join Prime MSDs (including the converses).

Number of Cunningham or X-join Prime MSDs	
Number of vertices	*Number of Prime MSDs (including the converses)*
3	2
4	0
5	0
6	1
7	0
8	8
9	19
10	96
11	430
totals	556

Here, all the X-join prime MSDs through 9 vertices are given. Excluding the converses – in case of non self-converse digraphs – there are 2 prime MSDs on 3 vertices, 1 on 6 vertices, 5 on 8 vertices and 14 on 9 vertices. Prime MSDs that are self converses are indicated by the legend *sc*.

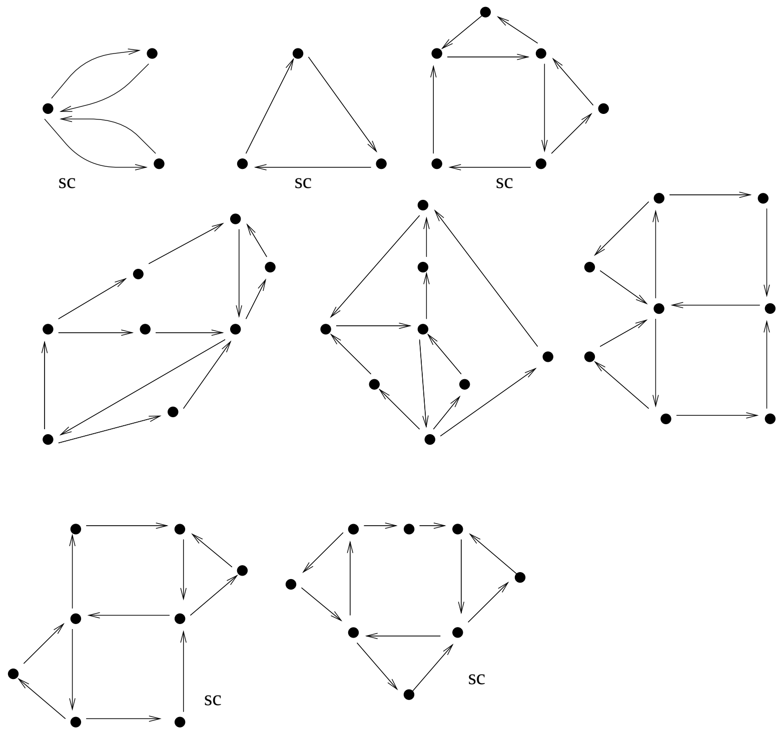

Figure A.1: All X-join prime MSDs through 8 vertices.

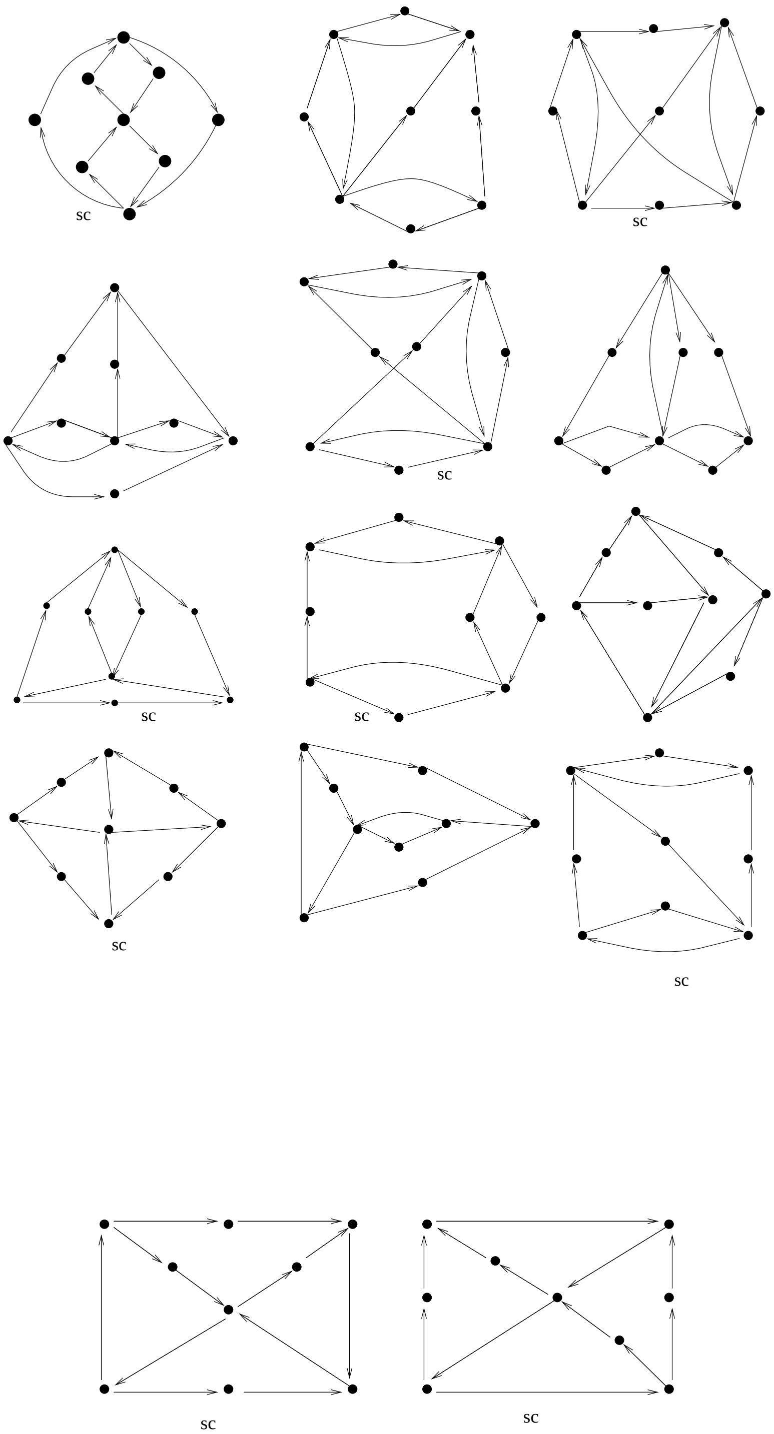

Figure A.2: All X-join prime MSDs on 9 vertices.