DIMACS Technical Report 2006-12
July 2006

Generating 3-Vertex Connected Spanning Subgraphs

by

Endre Boros
RUTCOR, Rutgers University
640 Bartholomew Road, Piscataway NJ 08854

Konrad Borys
RUTCOR, Rutgers University Piscataway NJ 08854

Vladimir Gurvich
RUTCOR, Rutgers University

Gabor Rudolf
RUTCOR, Rutgers University Piscataway NJ 08854

DIMACS is a collaborative project of Rutgers University, Princeton University, AT&T Labs–Research, Bell Labs, NEC Laboratories America and Telcordia Technologies, as well as affiliate members Avaya Labs, HP Labs, IBM Research, Microsoft Research, Stevens Institute of Technology, Georgia Institute of Technology and Rensselaer Polytechnic Institute. DIMACS was founded as an NSF Science and Technology Center.

ABSTRACT

In this paper we present an algorithm to generate all minimal 3-vertex connected spanning subgraphs of an undirected graph with n vertices and m edges in incremental polynomial time, i.e., for every K we can generate K (or all) minimal 3-vertex connected spanning subgraphs of a given graph in $O(K^2 log(K)m^2 + K^2m^3)$ time, where n and m are the number of vertices and edges of the input graph, respectively. This is an improvement over what was previously available and is the same as the best known running time for generating 2-vertex connected spanning subgraphs. Our result is obtained by applying the decomposition theory of 2-vertex connected graphs to the graphs obtained from minimal 3-vertex connected graphs by removing a single edge.

1 Introduction

Vertex connectivity is a fundamental concept in network reliability theory. While in the simplest case only the connectedness of an undirected graph, that is, the presence of a spanning tree, is required, in practical applications higher levels of connectivity are often desirable. Given the possibility that the edges of the network can randomly fail the reliability of the network is defined as the probability that the operating edges provide a certain level of connectivity. Most methods computing network reliability depend on the efficient generation of all minimal subsets of network edges which guarantee the required connectivity [17, 4].

In this paper we consider the problem of generating minimal 3-vertex connected spanning subgraphs. An undirected graph G on at least $k+1$ vertices is *k-vertex connected* if every subgraph of G obtained by removing at most $k-1$ vertices is connected. A subgraph of a graph G is *spanning* if it has the same vertex set as G. We define the problem of generating minimal 3-vertex connected spanning subgraphs as follows:

Minimal 3-Vertex Connected Spanning Subgraphs Generation Problem

Input: A 3-vertex connected undirected graph G

Output: The list of all minimal 3-vertex connected spanning subgraphs of G

Note that the number of all minimal 3-vertex connected spanning subgraphs of a graph G may be exponential in the number of vertices and edges. Therefore we measure the running time of generation algorithms in both the input and output size. A generation algorithm may output a minimal 3-vertex connected spanning subgraph any time during its execution. A generation algorithm runs in *incremental polynomial time* if it outputs K subgraphs (or all, if the number of minimal 3-vertex connected subgraphs is less than K) in time polynomial in n, m and K. A generation algorithm runs with *polynomial delay* if it outputs K subgraphs (or all) in time polynomial in n and m and linear in K (see e.g., [17, 11, 8]).

It was recently shown that for every fixed value of k we can generate K (or all, if the number of minimal k-vertex connected spanning subgraphs is less than K) minimal k-vertex connected spanning subgraphs of a graph with n vertices and m edges in $O(K^3nm^3 + K^2n^4m^5 + Kn^km^2)$ time [2]. For small values of k this can be improved upon: numerous research articles consider the problem of efficiently generating spanning trees in connected graphs ($k = 1$) [14, 6, 13, 1], with the best known running time being $O(Kn+m)$ [6, 13]. K minimal 2-vertex connected spanning subgraphs can be generated in time $O(K^2log(K)m^2 + K^2m^3)$ [9]. This improvement over the $O(K^3nm^3 + K^2n^4m^5)$ guaranteed by the general algorithm is achieved by exploiting the block decomposition of connected graphs [5, Chapter 3]. A similar decomposition theory exists for 2-vertex connected graphs [7]. In this paper we shall utilize this fact to achieve a similar improvement for the generation of minimal 3-vertex connected spanning subgraphs.

We remark that minimal strongly connected subgraphs of strongly connected digraphs can also be efficiently generated [3].

The remainder of the paper is organized as follows. In Section 1.1 we state our main result and in Section 1.2 we recall a technique from [10] used to prove the main result. The proof of our theorem is in Section 2.

1.1 Main Result

We show that the minimal 3-vertex connected spanning subgraphs problem can be solved in incremental polynomial time.

For every K we can generate K (or all, if the number of minimal 3-vertex connected spanning subgraphs is less than K) minimal 3-vertex connected spanning subgraphs of a given graph in $O(K^2 log(K) m^2 + K^2 m^3)$ time, where n and m are the number of vertices and edges of the input graph, respectively.
This is an improvement over the $O(K^3 nm^3 + K^2 n^4 m^5)$ guaranteed by the general algorithm applied for $k = 3$ [2] and it is the same running time as for the minimal 2-vertex connected spanning subgraphs generation problem.

1.2 The $X - e + Y$ method

In this section we recall a technique from [10], which is a variant of the supergraph approach introduced by [15]. Let $\mathcal{C}$ be a class of finite sets and for every $E \in \mathcal{C}$ let $\pi_E : 2^E \to \{0, 1\}$ be a monotone Boolean function, i.e., one for which $X \subseteq Y$ implies $\pi_E(X) \leq \pi_E(Y)$. We assume that $\pi_E(\emptyset) = 0$ and $\pi_E(E) = 1$. Let

$$\mathcal{F} = \{X \mid X \subseteq E \text{ is a minimal set satisfying } \pi_E(X) = 1\}.$$

Our goal is to generate all sets belonging to $\mathcal{F}$.

First we can fix an arbitrary linear order $\prec$ on elements of E and define a mapping $\mathit{Project} : \{X \subseteq E \mid \pi_E(X) = 1\} \to \mathcal{F}$ by

$$\mathit{Project}(X) = X \setminus Z,$$

where Z is the lexicographically first subset of X, with respect to $\prec$, such that $\pi_E(X \setminus Z) = 1$ and $\pi_E(X \setminus (Z \cup e)) = 0$ for every $e \in X \setminus Z$. We can compute $\mathit{Project}(X)$ by deleting one by one, from the smallest to the largest, elements of X whose removal does not change the value of π_E to 0. This requires evaluating π_E exactly $|X|$ times.

We next introduce a directed graph $\mathcal{G} = (\mathcal{F}, \mathcal{E})$ on vertex set $\mathcal{F}$. We define the neighborhood $N(X)$ of a vertex $X \in \mathcal{F}$ as follows

$$N(X) = \{\mathit{Project}((X \setminus e) \cup Y) \mid e \in X, Y \in \mathcal{Y}_{X,e}\},$$

where $\mathcal{Y}_{X,e}$ is defined by

$$\mathcal{Y}_{X,e} = \{Y \mid Y \text{ is a minimal subset of } E \setminus X \text{ satisfying } \pi_E((X \setminus e) \cup Y) = 1\}.$$

In other words, for every set $X \in \mathcal{F}$ and for every element $e \in X$ we extend $X \setminus e$ in all possible minimal ways to a set $X' = (X \setminus e) \cup Y$ for which $\pi_E(X') = 1$ (since $X \in \mathcal{F}$, we have $\pi_E(X \setminus e) = 0$), and introduce each time a directed arc from X to $Project(X')$. We call the obtained directed graph $\mathcal{G}$ the *supergraph* of our generation problem.

[[10]] The supergraph $\mathcal{G} = (\mathcal{F}, \mathcal{E})$ is strongly connected.

Since $\mathcal{G}$ is strongly connected, by performing a breadth-first search in $\mathcal{G}$ we can generate all elements of $\mathcal{F}$. Thus, given a procedure that generates all elements of $\mathcal{Y}_{X,e}$ for every $X \in \mathcal{F}$ and $e \in X$, the procedure $Transversal(\mathcal{G})$, defined below, generates all elements of $\mathcal{F}$.

```
                    $Traversal(\mathcal{G})$
Find an initial vertex $X^0 \leftarrow Project(E)$, initialize a queue $\mathcal{Q} = \emptyset$
    and a dictionary of output vertices $\mathcal{D} = \emptyset$.

Perform a breadth-first search of $\mathcal{G}$ starting from $X^o$:

    1 output $X^0$ and insert it to $\mathcal{Q}$ and to $\mathcal{D}$
    2 while $\mathcal{Q} \neq \emptyset$ do
    3   take the first vertex $X$ out of the queue $\mathcal{Q}$
    4   for every $e \in X$ do
    5     for every $\mathcal{Y} \in \mathcal{Y}_{X,e}$
    6       compute the neighbor $X' \leftarrow Project((X \setminus e) \cup Y)$
    7       if $X' \notin \mathcal{D}$ then output $X'$ and insert it to $\mathcal{Q}$ and to
          $\mathcal{D}$
```

Assume that there is a procedure that outputs K elements of $\mathcal{Y}_{X,e}$ in time $\phi(K, E)$ and there is an algorithm evaluating π_E in time $O(\gamma(E))$. Then $Traversal(\mathcal{G})$ outputs K elements of $\mathcal{F}$ in time $O(K^2|E|^2\gamma(E) + K^2 log(K)|E|^2 + K|E|\phi(K, E))$. Let $X \in \mathcal{F}$ and $e \in X$.

Claim 1 *If Y and Y' are distinct elements of $\mathcal{Y}_{X,e}$, then they produce different neighbors of X in $\mathcal{G}$ in line* **7**.

First we observe that for every $Y \in \mathcal{Y}_{X,e}$ we have $Project((X \setminus e) \cup Y) = ((X \setminus (Z \cup e)) \cup Y$, where Z is the lexicographically first subset of $X \setminus e$, with respect to $\prec$, such that $\pi_E((X \setminus (Z \cup e)) \cup Y) = 1$ and $\pi_E((X \setminus (Z \cup e \cup f)) \cup Y) = 0$ for every $f \in X \setminus (Z \cup e)$. By the minimality of Y, we have $\pi_E((X \setminus e) \cup (Y \setminus y)) = 0$ for every $y \in Y$. Thus $Project((X \setminus e) \cup Y)$ must contain Y. Also note that by minimality of Y, we obtain $X \setminus e$ and Y are disjoint.

Hence for Y and Y', distinct elements of $\mathcal{Y}_{X,e}$, we have $Project((X \setminus e) \cup Y) = ((X \setminus (Z \cup e)) \cup Y$ and $Project((X \setminus e) \cup Y') = ((X \setminus (Z' \cup e)) \cup Y'$. Since $Project((X \setminus e) \cup Y)$

contains Y, $Project((X \setminus e) \cup Y')$ contains Y', $X \setminus e$ and Y are disjoint, $X \setminus e$ and Y' are disjoint and $Y \neq Y'$ we obtain $Project((X \setminus e) \cup Y) \neq Project((X \setminus e) \cup Y')$.

Note that we output a vertex of the supergraph $\mathcal{G}$ every time we insert it to the queue $\mathcal{Q}$ and each vertex of $\mathcal{G}$ is inserted to the queue $\mathcal{Q}$ and removed from $\mathcal{Q}$ only once. Thus to generate K elements we repeat the while loop of lines **2-7** at most K times. As $|X| < |E|$ we repeat the for loop of lines **4-7** at most $|E|$ times. By Claim 1 we repeat the for loop of lines **5-7** at most K times (otherwise we generate more than K distinct neighbors). Generating K elements of $\mathcal{Y}_{X,e}$ takes $O(\phi(K, E))$ time.

We repeat lines **6**,**7** at most $K^2|E|$ times. Recall that evaluating $Project$ takes $O(|E|\gamma(E))$ time. We can implement the dictionary $\mathcal{D}$ as a red-black tree. Then the operations FIND and INSERT in $\mathcal{D}$ require at most a logarithmic number of comparisons, where each comparison takes $O(|E|)$ time. This implies that executing lines **6**,**7** a single time takes $O(|E|\gamma(E) + log(K)|E|)$ time.

Thus the time $Traversal(\mathcal{G})$ needs to output K elements is $O(K^2|E|^2\gamma(E)+ K^2log(K)|E|^2 +K|E|\phi(K, E))$.

2 Proof of Theorem 1.1

In this section we apply the $X - e + Y$ method to the generation of all minimal 3-vertex connected spanning subgraphs.

For a given 3-vertex connected graph (V, E) we define a Boolean function π_E as follows: for a subset $X \subseteq E$ let

$$\pi_E(X) = \begin{cases} 1, & \text{if (V,X) is 3-vertex connected;} \\ 0, & \text{otherwise.} \end{cases}$$

Clearly π_E is monotone, $\pi_E(\emptyset) = 0$ and $\pi_E(E) = 1$. Then $\mathcal{F} = \{X \mid X \subseteq E$ is a minimal set satisfying $\pi_E(X) = 1\}$ is the family of edge sets of all minimal k-vertex connected spanning subgraphs of (V, E).

Before we describe an algorithm of generating elements of $\mathcal{Y}_{X,e}$ we recall in Section 2.1 the decomposition theory for 2-vertex connected graphs presented in [7]. Then in Section 2.2 we prove that the decomposition of a graph $(V, X \setminus e)$ has a special structure, when (V, X) is a minimal 3-vertex connected subgraph. In Section 2.3 we introduce a minimal forward a-b extensions generation problem and recall an algorithm from [3] which solves it, then in Section 2.4 we reduce the problem of generating elements of $\mathcal{Y}_{X,e}$ to solving the minimal forward a-b extensions problem. Finally, in Section 2.5 we analyze the complexity of the procedure $Traversal$.

2.1 Dividing a Graph Into Triconnected Components

In this section we closely follow the exposition from [7].

Let $G = (V, E)$ be a 2-vertex connected multigraph with at least four vertices. A pair of vertices $\{x, y\}$ is called a *separation pair* of G if there is a partition E_1, E_2 of the edge set E such that

- $|E_1| \geq 2$, $|E_2| \geq 2$,
- the subgraphs induced by E_1, E_2 are connected,
- $\{x, y\} = V(E_1) \cap V(E_2)$, where $V(E_1)$ and $V(E_2)$ denote the sets of vertices of G incident to E_1 and E_2, respectively.

Note that if G has no separation pairs then G is 3-vertex connected.

For a separation pair $\{x, y\}$ and a corresponding partition E_1, E_2, we define $G_1 = (V(E_1), E_1 \cup xy)$ and $G_2 = (V(E_2), E_2 \cup xy)$. We call the multigraphs G_1, G_2 *split graphs* of G with respect to $\{x, y\}$. Replacing a multigraph G by two split graphs is called *splitting* G. There may be many possible ways to split a multigraph, even with respect to a fixed separation pair $\{x, y\}$. We denote a splitting operation by $s(x, y, i)$, where i is a label distinguishing this split operation from other splits. The new edges of G_1 and G_2 are called *virtual edges*. We label them (xy, i) so they are associated with the split $s(x, y, i)$.

Suppose a multigraph G is split, the split graphs are split, and so on, until no more splits are possible. We call the graphs constructed this way *split components* of G. The split components of a multigraph are of three types:

- triple bonds, where a *bond* is a multigraph having exactly two vertices u, v and one or more edges uv,
- triangles, where a *triangle* is a cycle of length 3,
- 3-vertex connected graphs.

To every decomposition of G into split components we associate a graph T as follows. The vertices of T are the split components. Two split components are connected if they both contain a virtual edge (xy, i). Therefore each edge of T corresponds to exactly one separation pair (though a separation pair can correspond to several edges of T or to none). Clearly T is a tree. We call T the *split components tree.*

The split components are not necessarily unique. In order to get unique components we must partially reassemble the split components. Suppose $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are two split components, both containing a virtual edge (xy, i). Let $G = (V_1 \cup V_2, (E_1 \setminus xy) \cup (E_2 \setminus xy))$. We call G a *merge graph* of G_1 and G_2 and denote it by $G = Merge(G_1, G_2)$. Merging is the inverse operation of splitting. If we perform a sufficient number of merges we recreate the original multigraph.

In order to find unique components we need to merge adjacent triple bonds as much as possible to obtain bonds and to merge adjacent triangles as much as possible to obtain cycles. We call these unique components *triconnected components.* Let T' be a tree obtained from T by contracting edges between triple bonds and between triangles. Notice that vertices of

T' are in one to one correspondence with triconnected components. Therefore we call T' the *tree of triconnected components*.

[[12, 16, 7]] G has a unique decomposition into triconnected components, each of which is 3-vertex connected graph, a cycle or a bond.

2.2 Structure of the Subgraph $(V, X \setminus e)$.

In this section we describe a structure of the graph obtained by removing an edge from a minimal 3-vertex connected subgraph of G.

Let (V, X) be a minimal 3-vertex connected subgraph of G and let $e \in X$ (see Figure 1). Consider a decomposition of $(V, X \setminus e)$ into split components $B_1, \ldots, B_l$. Let T denote the

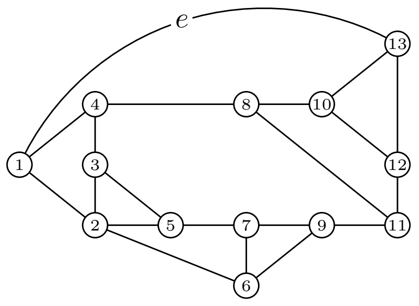

Figure 1: A minimal 3-vertex connected graph (V, X).

split components tree of $(V, X \setminus e)$ corresponding to this decomposition.

Claim 2 *T is a path such that both ends contain an endpoint of e.*

Suppose that one endpoint of e belongs to a separation pair. Then the removal of this separation pair disconnects (V, X), a contradiction with (V, X) being 3-vertex connected.

Thus neither endpoint of e belongs to any separation pair. Observe that vertices that do not belong to separation pairs occur in exactly one split component. Therefore it suffices to show that every leaf of T contains an endpoint of e. Suppose on the contrary that there is a leaf of T that does not contain an endpoint of e. Then removing the separation pair corresponding to the edge of T incident to this leaf disconnects (V, X), a contradiction with (V, X) being 3-vertex connected.

The above claim implies that we can assume the split components are indexed in a way such that T is the path $B_1 \ldots B_l$ (see Figure 2).

Claim 3 *There are no two subsequent split components B_i, B_{i+1} such that both are triple bonds.*

Suppose B_i and B_{i+1} are triple bonds. If B_i consists of three virtual edges then B_i has three neighbors in T, a contradiction with T being a path. Therefore both B_i and B_{i+1} have at least one nonvirtual edge each. Observe that the vertex set of B_i is the same pair of vertices

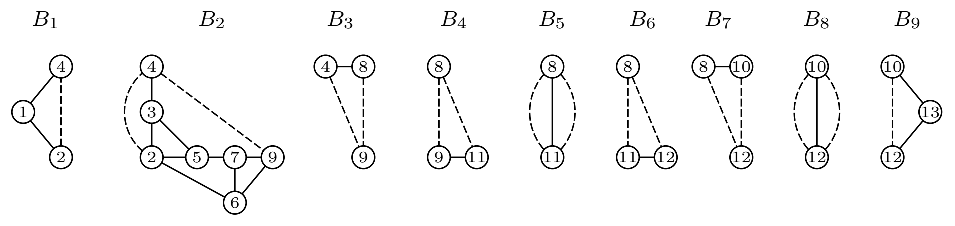

Figure 2: Decomposition of $(V, X \setminus e)$ into split components.

as that of B_{i+1}. Also, recall that split components partition the edge set X. Thus B_i and B_{i+1} contain two different edges of X connecting the same two vertices, a contradiction with the minimality of (V, X).

Claim 4 *The first two split components B_1 and B_2 are not both triangles. Similarly, the last two split components B_{l-1} and B_l are not both triangles.*

Suppose B_1 is a triangle on vertices $\{v_1, v_2, v_4\}$ and B_2 is a triangle on vertices $\{v_2, v_3, v_4\}$. Without loss of generality we assume that B_1 has the virtual edge v_2v_4 and B_2 has two virtual edges v_2v_4, v_2v_3. Consequently the vertices v_1 and v_4 both have degree two in $(V, X \setminus e)$. After adding the edge e, one of them still has degree two. Thus removing its neighbors disconnects (V, X), a contradiction with (V, X) being 3-vertex connected.

Claim 5 *There are no three subsequent split components B_i, B_{i+1}, B_{i+2} such that all of them are triangles.*

Suppose B_i is a triangle on vertices $\{v_1, v_4, v_5\}$, B_{i+1} is a triangle on vertices $\{v_1, v_2, v_4\}$ and B_{i+2} is a triangle on vertices $\{v_2, v_3, v_4\}$. By Claim 4, the triangles B_i, B_{i+1}, B_{i+2} cannot be the first or the last three vertices of the path T, thus each triangle has two virtual edges. Without loss of generality assume that the edges v_1v_4 and v_2v_4 are virtual.

We show that one of the vertices v_1, v_2, v_3, v_4, v_5 has degree two in $(V, X \setminus e)$. We have following four cases:

Case 1: The edges v_1v_5 and v_2v_3 are virtual. Then v_4 has degree two.
Case 2: The edges v_1v_5 and v_3v_4 are virtual. Then v_2 has degree two.
Case 3: The edges v_4v_5 and v_2v_3 are virtual. Then v_1 has degree two.
Case 4: The edges v_4v_5 and v_3v_4 are virtual. Then v_1 and v_2 have degree two.

Since none of these triangles is the end of the path T, endpoints of e do not belong to any of them. Thus any vertex of degree two in $(V, X \setminus e)$ has the same degree in (V, X). Removing the two neighbors of that vertex disconnects (V, X), a contradiction with (V, X) being 3-vertex connected.

Let $T(a, b, c)$ denote the triangle on vertices a, b, c with a nonvirtual edge ab and virtual edges ac, bc (see Figure 3). We consider two subsequent triangles B_i and B_{i+1}. By Claim 4 each triangle has two virtual edges. Without loss of generality we can assume that B_i is the triangle $T(v_1, v_2, v_4)$. Let B_{i+1} be a triangle on vertices $\{v_2, v_3, v_4\}$ with a virtual edge v_2v_4.

Claim 6 *Nonvirtual edges of B_i and B_{i+1} cannot have a common endpoint (see for example B_3 and B_4 in Figure 2). Thus B_{i+1} is the triangle $T(v_3, v_4, v_2)$.*

Suppose v_3v_4 is a virtual edge, consequently v_1v_2, v_2v_3 are not virtual. Then v_2 is a vertex of degree two in (V, X), a contradiction. Thus the edge v_2v_3 must be virtual.

A *square* $Q(a, b, c, d)$ is a cycle of length 4 on vertices a, b, c, d with nonvirtual edge ab, cd and virtual edges ad, bc (see Figure 3).

Now we consider the unique decomposition of $(V, X \setminus e)$ into triconnected components $C_1, \ldots, C_k$ obtained by merging pairs of triple bonds and pairs of triangles containing the same virtual edge, as described in Theorem 2.1.

If (V, X) is minimal 3-vertex connected graph then each triconnected component is one of the following four types:

- a triple bond,
- a triangle,
- a square of the form $Q(a, b, c, d)$,
- a 3-vertex connected graph.

Moreover, the tree T' of triconnected components is a path of length at most $2|V| + 1$. Consider the decomposition into the split components $B_1, \ldots, B_l$. By Claim 3, there are no two triple bonds containing the same virtual edge, consequently every triple bond B_i is a triconnected component. By Claim 5 there are no three consecutive triangles in the sequence $B_1, \ldots, B_l$. Let B_i and B_{i+1} be a pair of subsequent triangles. By Claim 6, they are of the form $B_i = T(a, b, c)$ and $B_{i+1} = T(c, d, b)$. After merging they form a square $Q(a, b, c, d)$ (see Figure 3).

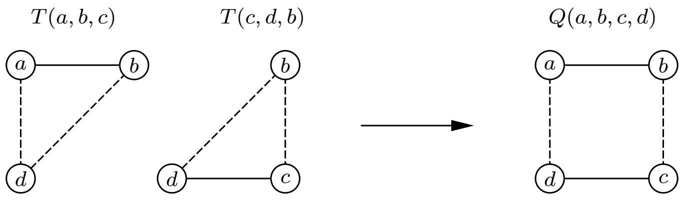

Figure 3: Triangles $T(a, b, d)$, $T(c, d, b)$ and their merge graph $Q(a, b, c, d)$.

Since the tree T' is obtained from the path T by contracting some of its edges, T' is also a path. Let $\{x_1, y_1\}, \ldots, \{x_{k-1}, y_{k-1}\}$ be the separation pairs corresponding to edges of T', where $\{x_{i-1}, y_{i-1}\}$ belongs to both C_{i-1} and C_i for each $i = 2, \ldots, k$. We call a a separation pair $\{x_i, y_i\}$ a *new pair* if $\{x_i, y_i\} \not\subseteq \{x_1, \ldots, x_{i-1}, y_1, \ldots, y_{i-1}\}$. Observe that C_i is a triple bond if and only if $\{x_i, y_i\} = \{x_{i-1}, y_{i-1}\}$.

Now we show that if C_i is not a triple bond then $\{x_i, y_i\}$ is a new pair. Suppose that $\{x_i, y_i\} \subseteq \{x_1, \ldots, x_{i-1},\ y_1, \ldots, y_{i-1}\}$. Since C_i is not a triple bond, we have $\{x_i, y_i\} \neq$

$\{x_{i-1}, y_{i-1}\}$. Without loss of generality we can assume that $x_i \notin \{x_{i-1}, y_{i-1}\}$, implying $x_i \in \{x_1, \ldots, x_{i-2}, y_1, \ldots, y_{i-2}\}$. Consequently $x_i \in V(G_1)$ and $x_i \in V(G_2)$, where G_1 and G_2 are the split graphs with respect to $\{x_{i-1}, y_{i-1}\}$. So $x_i = V(G_1) \cap V(G_2)$, a contradiction with $x_i \notin \{x_{i-1}, y_{i-1}\}$. Together with Claim 3, this implies that at least half of separation pairs are new and since the number of new pairs is at most $|V|$, we obtain that $k-1 \leq 2|V|$.

2.3 Minimal Forward a-b Extensions

In this section we present an algorithm from [3] which generates all minimal forward a-b extensions.

We consider a directed graph $G = (V, F \cup B)$ whose arcs are partitioned into a set of forward arcs F and a set of backward arcs B, and two distinguished vertices $a, b \in V$. A *forward a-b extension X of G* is a subset of forward arcs such that b is reachable from a in $(V, B \cup X)$. We define the problem of generating minimal forward a-b extensions as follows:

Minimal Forward a-b Extensions Generation Problem

Input: A directed graph $G = (V, F \cup B)$ and two distinguished vertices $a, b \in V$

Output: The list of all minimal forward a-b extensions of G

The following algorithm generates all minimal forward a-b extensions.

```
            Extend($z, W_1, W_2$)
1 $B(z) \leftarrow z \cup \{$ all vertices that $z$ can be reached from using
    only backward arcs $\}$
2 if $a \in B(z)$ then output $W_1$
3 else
4   $Z \leftarrow \{uv \in F \setminus (W_1 \cup W_2) : u \notin B(z), v \in B(z)\}$
5   for every $uv \in Z$ do
6     if $u$ reachable from $a$ in $(V, B \cup F \setminus (W_1 \cup W_2 \cup Z))$
7         then Extend($u, W_1 \cup \{uv\}, W_2 \cup (Z \setminus \{uv\})$)
```

[3] For every K the procedure *Extend*$(b, \emptyset, \emptyset)$ generates K (or all) minimal forward a-b extensions of a graph $G = (V, F \cup B)$ in $O(K|V|(|V| + |F| + |B|))$ time.

2.4 Generating Elements of $\mathcal{Y}_{X,e}$

For a minimal 3-vertex connected spanning subgraph (V, X) of $G = (V, E)$ and an edge $e \in X$ (see Figure 4), $\mathcal{Y}_{X,e}$ is the collection of minimal subsets of $E \setminus X$ restoring 3-vertex connectivity to $(V, X \setminus e)$. In this section we reduce the problem of generating elements of $\mathcal{Y}_{X,e}$ to an instance of the minimal forward a-b extensions generation problem.

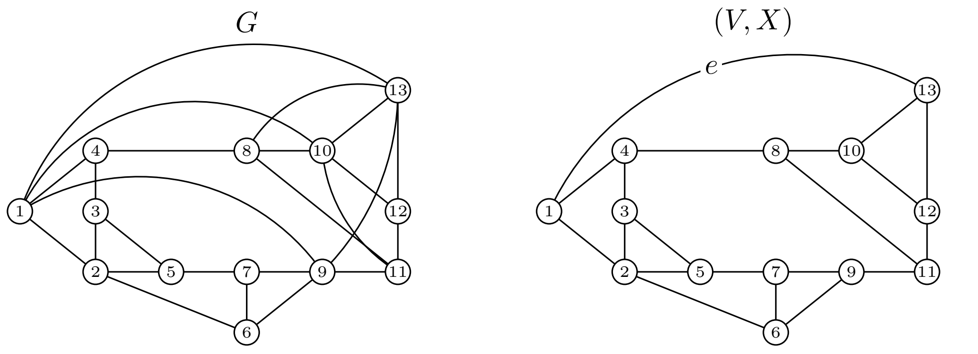

Figure 4: 3-vertex connected graph $G = (V, E)$ and a minimal 3-connected spanning subgraph (V, X) of G.

Consider a decomposition of $(V, X \setminus e)$ into triconnected components $C_1, \dots, C_k$. Let T be the tree of triconnected components. By Theorem 2.1 and Proposition 2.2 the decomposition is unique, the triconnected components are of four types: 3-vertex connected graphs, triple bonds, triangles and squares; furthermore, we can assume that the triconnected components are indexed in a way such that T is the path $C_1, \dots, C_k$ (see Figure 5).

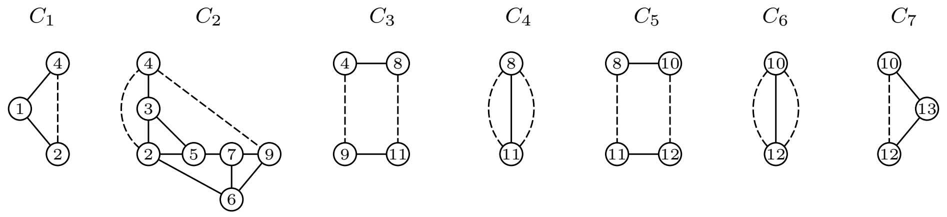

Figure 5: Decomposition of $(V, X \setminus e)$ into triconnected components.

Next we construct a directed graph H as follows:

- for each 3-vertex connected graph or triangle C_i, we add a vertex D_i,
- for each square C_i, we add three vertices D_i, E_i, F_i with arcs D_iE_i, D_iF_i,
- for each 3-vertex connected graph or triangle C_i, $i \geq 2$, we add an arc D_iD_j, where $j = \begin{cases} i-2, & \text{if } C_{i-1} \text{ is a triple bond;} \\ i-1, & \text{otherwise,} \end{cases}$

- for each square C_i, we add arcs E_iD_j, F_iD_j, where $j = \begin{cases} i-2, & \text{if } C_{i-1} \text{ is a triple bond;} \\ i-1, & \text{otherwise} \end{cases}$

  (see Figure 6).

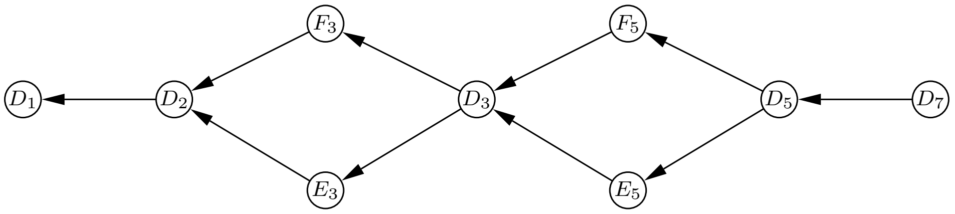

Figure 6: Directed graph H.

We call arcs of the digraph H *backward arcs*. A *segment* is a backward arc D_iD_j. A *diamond* consists of four arcs D_iE_i, D_iF_i, E_iD_j, F_iD_j. Note that the backward arcs of H can be uniquely partitioned into segments and diamonds.

Observation 1 *Let $A_j \in \{D_j, E_j, F_j\}$ and let $A_i \in \{D_i, E_i, F_i\}$, where $j < i$. Then A_j is reachable from A_i using the backward arcs.*

For a vertex u, we define $l(u) = \min\{\ i \mid u \in C_i\}$ and $r(u) = \max\{\ i \mid u \in C_i\}$ to be the indices of the leftmost and rightmost triconnected component containing u. Obviously $l(u) \leq r(u)$. Note that if u does not belong to any separation pair, then u belongs to exactly one triconnected component, therefore $l(u) = r(u)$. Also note that for every vertex u neither $C_{l(u)}$ nor $C_{r(u)}$ is a triple bond.

Let $uv \in E \setminus X$ be an edge such that $l(u) \leq l(v)$. We define

$$R(u) = \begin{cases} F_{r(u)}, & \text{if } C_{r(u)} \text{ is a square } Q(a,b,c,d) \text{ and } u = a; \\ E_{r(u)}, & \text{if } C_{r(u)} \text{ is a square } Q(a,b,c,d) \text{ and } u = d; \\ D_{r(u)}, & \text{otherwise,} \end{cases}$$

$$L(v) = \begin{cases} F_{l(v)}, & \text{if } C_{l(v)} \text{ is a square } Q(a,b,c,d) \text{ and } v = b; \\ E_{l(v)}, & \text{if } C_{l(v)} \text{ is a square } Q(a,b,c,d) \text{ and } v = c; \\ D_{l(v)}, & \text{otherwise.} \end{cases}$$

Let H' be the directed multigraph obtained from H by adding the arc $R(u)L(v)$ to D for every edge $uv \in E \setminus X$ such that $r(u) \leq l(v)$ (see Figure 7). We call the new arcs *forward arcs*.

The graph H' corresponding to the minimal 3-vertex connected subgraph (V, X) of $G = (V, E)$ has at most $6|V| + 3$ vertices and at most $|E| + 12|V| + 6$ arcs. Furthermore, we can construct H' in $O(|V| + |E|)$ time. By Proposition 2.2 the number of triconnected components is at most $2|V| + 1$, and since at most three vertices of H' correspond to a single triconnected component, the number of vertices of H' is at most $6|V|$. We add at most two backward arcs coming out of a vertex and at most $|E|$ forward arcs. The bound on the

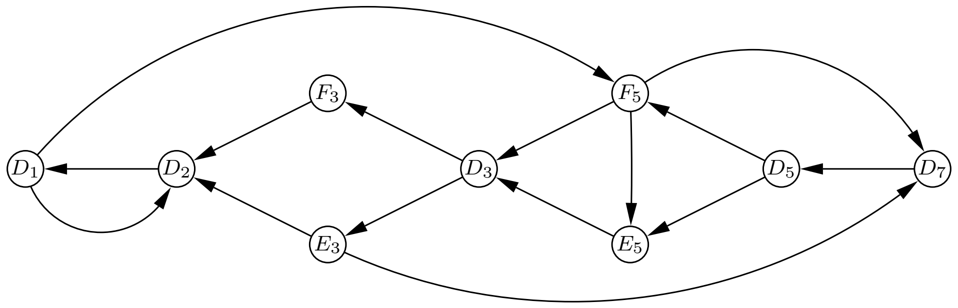

Figure 7: Directed multigraph H'.

complexity of the constructing H' follows from the fact that we can find all triconnected components of the graph $(V, X \setminus e)$ in $O(|V| + |X|)$ time [7].

We next show that generating elements of $\mathcal{Y}_{X,e}$ is equivalent to the minimal forward D_1-D_k extensions generation problem in H' (see Section 2.3).

Observation 2 *Let $\{x, y\}$ be a separation pair of $(V, X \setminus e)$ with split graphs G_1, G_2 and let $f \in E \setminus X$. Then $\{x, y\}$ is not a separation pair of $(V, X \setminus e \cup f)$ if and only if $f = uv$, where $u \in V(G_1) \setminus \{x, y\}$ and $v \in V(G_2) \setminus \{x, y\}$. In this case we say that f* annihilates *$\{x, y\}$.*

Let $Z \subseteq E \setminus X$ be a set such that $(V, X \setminus e \cup Z)$ is 3-vertex connected. We define $A_Z = \{\ R(u)L(v) \mid uv \in Z,\ r(u) \le l(v)\}$ to be the subset of forward arcs corresponding to the edges of Z.

A_Z is a forward D_1-D_k extension of H'. Since the backward arcs of H' can be partitioned into segments and diamonds it is sufficient to show that for every segment or diamond we can reach its right end from its left end using only the forward arcs in A_Z and the backward arcs.

Claim 7 *Let $S = D_iD_j$ be a segment. D_i is reachable from D_j using the arcs in A_Z and the backward arcs.*

Let $\{x, y\}$ be a separation pair of $(V, X \setminus e)$ corresponding to the edge $C_{i-1}C_i$ with split graphs $G_1 = Merge(C_1, \dots, C_{i-1})$, $G_2 = Merge(C_i, \dots, C_k)$. By Observation 2, there is an edge $uv \in Z$ such that u belongs to G_1, v belongs to G_2 and $u, v \notin \{x, y\}$ (see Figure 8).

Since $u \in G_1$, $v \in G_2$, we have $r(u) \le i-1$ and $l(v) \ge i$. Hence the forward arc $R(u)L(v)$ belongs to A_z.

Since $l(v) \ge i$, by Observation 1 D_i is reachable from $L(v)$ using some backward arcs.

Recall that

$$j = \begin{cases} i-2, & \text{if } C_{i-1} \text{ is a triple bond;} \\ i-1, & \text{otherwise.} \end{cases}$$

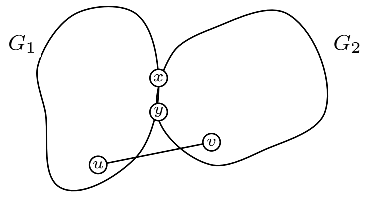

Figure 8: Edge uv annihilating $\{x, y\}$.

Since $C_{r(u)}$ cannot be a triple bond and $r(u) \leq i-1$, we obtain $r(u) \leq j$. By Observation 1, $R(u)$ is reachable from D_j using some backward arcs.

Therefore D_i is reachable from D_j using the forward arc $R(u)L(v)$ and some backward arcs.

Claim 8 *Let $S = \{D_iE_i,\ D_iF_i,\ E_iD_j,\ F_iD_j\}$ be a diamond. D_i is reachable from D_j using the arcs in A_Z and the backward arcs.*

Since S is a diamond, C_i is a square $Q(a, b, c, d)$. Then $\{\{a, d\},\ \{b, c\},\ \{a, c\},\ \{b, d\}\}$ are the four separation pairs in C_i.

Let $uv \in Z$ be an edge annihilating $\{a, d\}$. Observe that $r(u) \leq j$. Depending on v we have two cases:

Case 1: $v \notin \{b, c\}$. Then $l(v) \geq i+1$ and $R(u)L(v)$ belongs to A_Z. Thus D_i is reachable from D_j using the forward arc $R(u)L(v)$ and some backward arcs.

Case 2: $v \in \{b, c\}$. Consequently $l(v) = i + 1$. Without loss of generality assume $v = b$ (see Figure 9). Then $R(u)F_i$ belongs to $A(Z)$. Let $u'v' \in Z$ be an edge annihilating $\{b, d\}$.

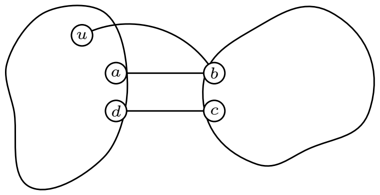

Figure 9: Edge ub annihilating $\{a, d\}$.

Observe that $r(u') = i$ if $u' = a$ and $r(u') \leq i - 1$ otherwise. Hence either $R(u') = F_i$ or $R(u') \in \{D_j, E_j, F_j\}$, where $j \leq i-1$. In the latter case, by Observation 1 $R(u')$ is reachable from F_i using the backward arcs. Depending on v' we have two subcases:

Case 2.1: $v' \neq c$. Consequently $l(v') \geq i + 1$ and $R(u')L(v') \in A_Z$. By Observation 1, D_i is reachable from $L(v')$ using some backward arcs. Thus D_i is reachable from D_j using one or two forward arcs from $\{R(u)F_i, R(u')L(v')\}$ and some backward arcs.

Case 2.2: $v' = c$. Consequently $L(v') = E_i$ and $R(u')E_i \in A_Z$. Let $u''v'' \in Z$ be an edge annihilating $\{b, c\}$. Then $l(v'') \geq i + 1$, $r(u'') = i$ if $u'' \in \{a, d\}$ and $r(u'') \leq i - 1$ otherwise. Thus $R(u'')L(v'') \in A_Z$. By Observation 1, D_i is reachable from $L(v'')$ using some backward arcs and $R(u'')$ is reachable from either E_i or F_i using some backward arcs. Hence D_i is reachable from D_j using two or three forward arcs from $\{R(u)F_i, R(u')E_i,\ R(u'')L(v'')\}$ and some backward arcs.

Now we consider a forward D_1-D_k extension A of H'. Let Z_A be the set of edges corresponding to arcs of A.

$(V, X \setminus e \cup Z_A)$ is 3-vertex connected. We define a mapping μ between segments and diamonds of H and sets of separation pairs as follows:

- for each segment $S = D_iD_j$ let $\mu(S) = \{\{x, y\}\}$, where $\{x, y\}$ is the separation pair corresponding to the edge C_iC_{i-1} of T,
- for each diamond $S = \{D_iE_i, D_iF_i, E_iD_j, F_iD_j\}$, corresponding to some square $Q(a, b, c, d)$, let $\mu(S) = \{\{a, d\},\ \{a, c\},\ \{b, d\}\}$.

Claim 9 *Let $\mathcal{S}$ be the set of all segments and diamonds of H. Then $\bigcup_{S\in\mathcal{S}} \mu(S)$ is the set of all separation pairs of $(V, X \setminus e)$.*

Let $\{x, y\}$ be a separation pair of $(V, X \setminus e)$ with split graphs G_1, G_2. We continue splitting these graphs until we obtain a decomposition into split components where xy is a virtual edge belonging to at least two split components.

If xy belongs to two subsequent triangles, then after merging these triangles we obtain a square $Q(a, b, c, d)$, where $\{x, y\} \in \{\{a, c\}, \{b, d\}\}$. Otherwise xy belongs to two or three split components (in the latter case the middle split component is a triple bond) and, after the merging of bonds and triangles into triconnected components, xy is still the virtual edge of two or three triconnected components.

Thus each separation pair $\{x, y\}$ is one of the following five types:

- there are two virtual edges xy, belonging to C_{i-1} and C_i, where C_i is not a square,
- there are four virtual edges xy, belonging to C_{i-2}, C_{i-1} and C_i, where C_i is not a square and C_{i-1} is a triple bond,
- there are two virtual edges xy, belonging to C_{i-1} and C_i, where $C_i = Q(a, b, c, d)$ and $xy = ad$,
- there are four virtual edges xy, belonging to C_{i-2}, C_{i-1} and C_i, where $C_i = Q(a, b, c, d)$, $xy = ad$ and C_{i-1} is a triple bond,
- $\{x, y\}$ is one of the two separation pairs $\{a, c\}, \{b, d\}$ of a square $C_i = Q(a, b, c, d)$.

Let

$$j = \begin{cases} i-2, & \text{if } C_{i-1} \text{ is a triple bond;} \\ i-1, & \text{otherwise.} \end{cases}$$

In the first two cases $\{x, y\} \in \mu(D_iD_j)$. In the remaining cases $\{x, y\} \in \mu(S)$, where S is the diamond $\{D_iE_i,\ D_iF_i,\ E_iD_j,\ F_iD_j\}$.

By Observation 2, to prove Lemma 11 it suffices to show that for every separation pair $\{x, y\}$ there exists an edge in Z_A, neither endpoint of which is a vertex of the separation pair, connecting the split graphs of $(V, X \setminus e)$ with respect $\{x, y\}$. By Claim 9, all separation pairs correspond to segments and diamonds.

First, consider a segment D_iD_j with $\mu(D_iD_j) = \{x, y\}$. Let $G_1 = Merge(C_1, \ldots, C_{i-1})$, $G_2 = Merge(C_i, \ldots, C_k)$ be split graphs of $(V, X \setminus e)$ with respect to $\{x, y\}$. The extension A must contain an arc R_pL_q, where $R_p \in \{D_p, E_p, F_p\}$, $L_q \in \{D_q, E_q, F_q\}$ $p \leq j$, $q \geq i$ (see Figure 10).

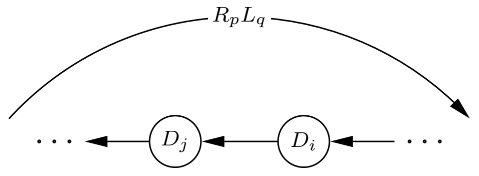

Figure 10: Arc R_pL_q of A.

Let uv be an edge of Z corresponding to R_pL_q. Since $r(u) = p$, $l(v) = q$, we obtain $u \in G_1$ and $v \in G_2$. Note that $r(x), r(y) \geq i$. Since $r(u) \leq j \leq i - 1$, we have $u \notin \{x, y\}$. Similarly, $v \notin \{x, y\}$. Thus uv annihilates $\{x, y\}$.

Next, consider a diamond $S = \{D_iE_i,\ D_iF_i,\ E_iD_j,\ F_iD_j\}$ with $C_i = Q(a, b, c, d)$ and $\mu(S) = \{\{a, d\}, \{a, c\}, \{b, d\}\}$. Consider the following three pairs of split graphs:

$$G'_{a,d} = Merge(C_1, \ldots, C_{i-1}), \quad G''_{a,d} = Merge(C_i, \ldots, C_k),$$

$$G'_{a,c} = Merge(C_1, \ldots, C_{i-1}, T(a, c, d)), \quad G''_{a,c} = Merge(T(a, b, c), C_{i+1}, \ldots, C_k),$$

$$G'_{b,d} = Merge(C_1, \ldots, C_{i-1}, T(a, b, d)), \quad G''_{b,d} = Merge(T(c, d, b), C_{i+1}, \ldots, C_k).$$

where $T(a, c, d), T(a, b, c), T(a, b, d), T(c, d, b)$ are triangles as introduced in Section 2.2.

Since A is a forward D_1-D_k extension by Observation 1 the vertex D_i is reachable from D_j using arcs of A and backward arcs. We have following three cases:

Case 1: $R_pL_q \in A$, where $p \leq j$, $q \geq i$. Let uv be an edge of Z corresponding to R_pL_q (see Figure 11). Observe that $u, v \notin \{a, b, c, d\}$. Thus uv annihilates all three separation

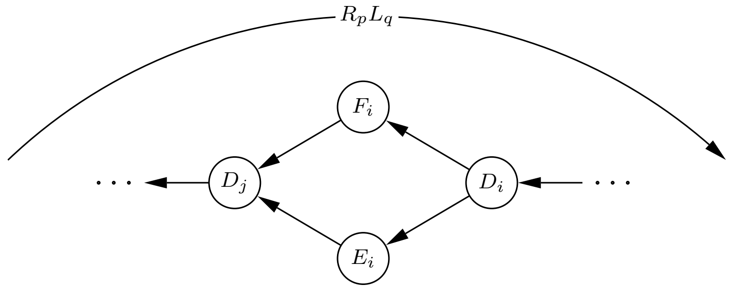

Figure 11: Arc R_pL_q corresponding to edge uv annihilating all three separation pairs of $\mu(S)$.

pairs of $\mu(S)$, since u belongs to the split graphs $G'_{a,d}$, $G'_{a,c}$, $G'_{b,d}$ and v belongs to $G''_{a,d}$, $G''_{a,c}$, $G''_{b,d}$.

Case 2: $\{R_pE_i, E_iL_q\} \in A$ or $\{R_pF_i, F_iL_q\} \in A$, where $p \leq j$, $q \geq i$. Without loss of generality suppose that A contains the arcs R_pF_i, F_iL_q. Let uv, $u'v'$ be edges of Z_A corresponding to R_pF_i, F_iL_q, respectively. Observe that $v = b$, $u' = a$ and $u, v' \notin \{a, b, c, d\}$ (see Figure 12). The edge ub annihilates the separation pairs $\{a, d\}$ and $\{a, c\}$, since u

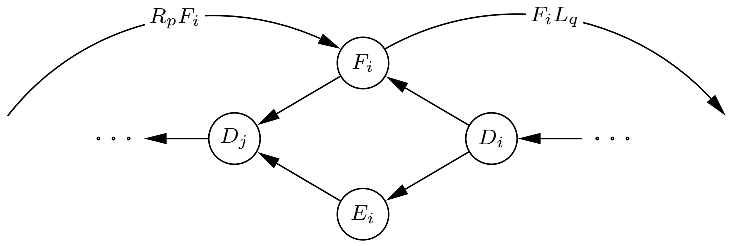

Figure 12: Arcs R_pE_i, E_iL_q corresponding to edges uc, dv', respectively.

belongs to G'_{ad}, G'_{ac} and b belongs to G''_{ad}, G''_{ac}. The edge av' annihilates $\{b, d\}$, since a belongs to $G'_{b,d}$ and v' belongs to $G''_{b,d}$.

Case 3: $\{R_pE_i, E_iF_i, F_iL_q\} \in A$ or $\{R_pF_i, F_iE_i, E_iL_q\} \in A$, where $p \leq j$, $q \geq i$. Without loss of generality suppose that A contains the arcs R_pF_i, F_iE_i, E_iL_q. Then edges $ub, ac, dv' \in Z_A$ correspond to R_pF_i, F_iE_i and E_iL_q, respectively, where $u, v' \notin \{a, b, c, d\}$ (see Figure 13).

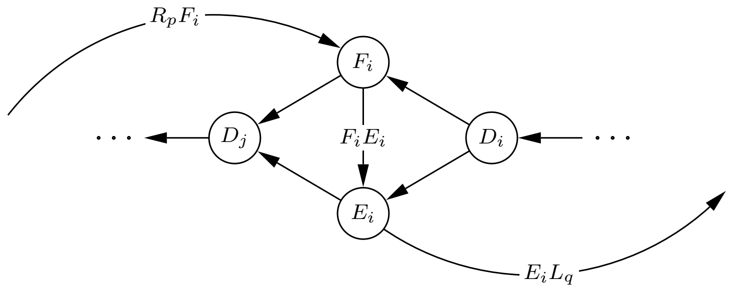

Figure 13: Arcs R_pF_i, F_iE_i and E_iL_q corresponding to edges ub, ac, dv', respectively.

The edge ub annihilates $\{a, d\}$, $\{a, c\}$, ac annihilates $\{b, d\}$ and dv' annihilates $\{a, c\}$.

2.5 Complexity

In this section we utilize Proposition 1.2 to analyze the total running time of the procedure $Transversal(\mathcal{G})$. Let $n = |V|$, $m = |E|$. As G is 3-vertex connected, $m \geq n$.

Since one can test if a graph is 3-vertex connected in $O(n+m)$ time [7], we have $\gamma(E) = m$. By Lemma 2.4 the graph H' has $O(n)$ vertices and $O(m)$ arcs. Thus by Theorem 2.3 we obtain $\phi(K, E) = Knm$.

By Proposition 1.2 the procedure $Transversal(\mathcal{G})$ generates K minimal 3-vertex connected subgraphs in $O(K^2 log(K) m^2 + K^2 m^3)$ time.

References

[1] A. Tamura A. Shioura and T. Uno. An optimal algorithm for scanning all spanning trees of undirected graphs. *SIAM Journal on Computing*, 26(3):678–692, 1997.

[2] E. Boros, K. Borys, K. Elbassioni, V. Gurvich, K. Makino, and G.Rudolf. Generating k-vertex connected spanning subgraphs and k-edge connected spanning subgraphs. Manuscript.

[3] E. Boros, K. Elbassioni, V. Gurvich, and L. Khachiyan. Enumerating minimal dicuts and strongly connected subgraphs and related geometric problems. In D. Bienstock and G. Nemhauser, editors, *Integer Programming and Combinatorial Optimization, 10th International IPCO Conference, New York, NY, USA*, volume 3064 of *Lecture Notes in Computer Science*, pages 152–162, Berlin, Heidelberg, New York, June 7-11 2004. Springer. (RRR 36-2003).

[4] C.J. Coulbourn. *The Combinatorics of Network Reliability.* Oxford University Press, 1987.

[5] R. Diestel. *Graph theory.* Springer-Verlag, 2000.

[6] H.N. Gabow and E.W. Myers. Finding all spanning trees of directed and undirected trees. *SIAM Journal on Computing*, 117:280–287, 1978.

[7] J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(3):135–157, 1973.

[8] D.S. Johnson and Ch. H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27:119–123, 1988.

[9] L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, V. Gurvich, and K. Makino. Enumerating spanning and connected subsets in graphs and matroids. Manuscript.

[10] L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, V. Gurvich, and K. Makino. Generating cut conjunctions and bridge avoiding extensions in graphs. In *Algorithms and Computation: 16th International Symposium, ISAAC 2005, Sanya, Hainan, China, December 19-21, 2005*, pages 156–165, 2005, full version to appear in Algorithmica.

[11] E. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Generating all maximal independent sets: NP-hardness and polynomial-time algorithms. *SIAM Journal on Computing*, 9:558–565, 1980.

[12] S. Maclaine. A structural characterization of planar combinatorial graphs. *Duke Math Journal*, 3:460472, 1937.

[13] T. Matsui. Algorithms for finding all the spanning trees in undirected graphs. Technical report, Department of Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo, 1993. Report METR93-08.

[14] R. C. Read and R. E. Tarjan. Bounds on backtrack algorithms for listing cycles, paths, and spanning trees. *Networks*, 5:237–252, 1975.

[15] B. Schwikowski and E. Speckenmeyer. On enumerating all minimal solutions of feedback problems. *Discrete Applied Mathematics*, 117:253–265, 2002.

[16] W.T. Tutte. *Connectivity in graphs.* Univ. Toronto Press, 1966.

[17] L.G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8:410–421, 1979.