www.combinatorics.ir

Transactions on Combinatorics
ISSN (print): 2251-8657, ISSN (on-line): 2251-8665
Vol. 9 No. 1 (2020), pp. 1-24.

www.ui.ac.ir

A LINEAR ALGORITHM FOR COMPUTING $\gamma_{[1,2]}$-SET IN GENERALIZED SERIES-PARALLEL GRAPHS

POUYEH SHARIFANI AND MOHAMMAD REZA HOOSHMANDASL*

Communicated by Hamidreza Maimani

Abstract. For a graph $G = (V, E)$, a set $S \subseteq V$ is a $[1, 2]$-set if it is a dominating set for G and each vertex $v \in V \setminus S$ is dominated by at most two vertices of S, i.e. $1 \leq |N(v) \cap S| \leq 2$. Moreover a set $S \subseteq V$ is a total $[1, 2]$-set if for each vertex of V, it is the case that $1 \leq |N(v) \cap S| \leq 2$. The $[1, 2]$-domination number of G, denoted $\gamma_{[1,2]}(G)$, is the minimum number of vertices in a $[1, 2]$-set. Every $[1, 2]$-set with cardinality of $\gamma_{[1,2]}(G)$ is called a $\gamma_{[1,2]}$-set. Total $[1, 2]$-domination number and $\gamma_{t[1,2]}$-sets of G are defined in a similar way. This paper presents a linear time algorithm to find a $\gamma_{[1,2]}$-set and a $\gamma_{t[1,2]}$-set in generalized series-parallel graphs.

1. Introduction

There is a rich theoretical literature around many important topics in graph theory, however, due to their vast new applications, we are in need of efficient algorithms for computing them. Unfortunately, most of these problems are **NP**-hard. Traditionally there are two approaches to handle such problems. One is to use algorithms that yield to approximate solution[10, 11, 21]. The other is to restrict the problem to some special cases and solve the problem efficiently[20, 2, 6].

Finding a minimum dominating set for graphs is such a problem, which is known to be **NP**-complete. However, it is solvable in polynomial time for trees and series parallel graphs [1, 8, 5, 13].

Also, finding a $[1, k]$-set for graphs is an **NP**-complete problem [4]. In [4], it is shown that the problem of finding a $[1, 2]$-set of cardinality at most ℓ, for a given integer ℓ is **NP**-complete, too.

MSC(2010): Primary: 05C15; Secondary: 20D60.
Keywords: Domination, Total Domination, [1,2]-set, Total [1,2]-set, Series-parallel graphs, Generalized series-parallel graph.
Received: 20 July 2017, Accepted: 24 November 2019.
*Corresponding author.

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

.

The concept of total $[1,k]$-set is proposed in [4]. In [7], it has been proved that the problem of checking whether a given graph G have any total $[1,2]$-set is **NP**-complete. For some special families of trees, Yang and Wu computed $[1,2]$-domination number in [19]. Then, Goharshady et al. presented a linear algorithms to compute the minimum cardinality of $[1,2]$-sets and total $[1,2]$-sets in general trees [9].

In this paper, we consider the problem of computing the minimum cardinality of $[1,2]$-sets and total $[1,2]$-sets in generalized series-parallel graphs, or GSPs for short. These graphs belong to the class of decomposable graphs which means that they can be represented by their parse trees. They can also be obtained from a set of single-edges by recursively applying the operations of series, parallel and generalized series. GSP graphs includes series-parallel (SP) graphs, outerplanar graph, trees, unicyclic graphs, C_N-trees, C-trees, 2-trees, cacti and filaments [12, 17] (see Figure 1).

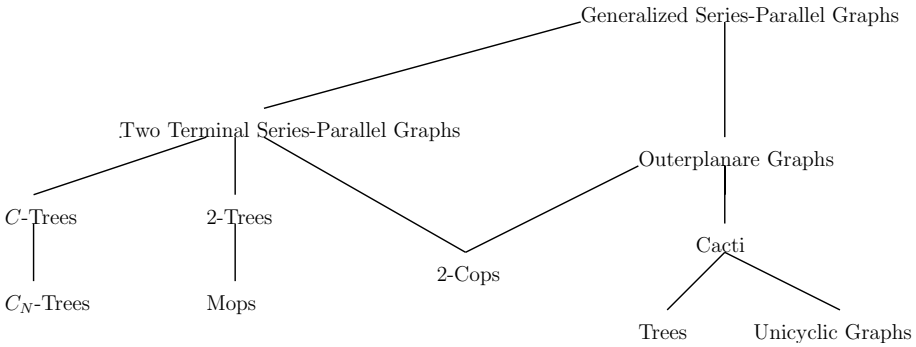


FIGURE 1. Subfamilies of the generalized series-parallel family [12].

Given a parse tree of a GSP graph, there exist linear-time algorithms for solving many graph theoretic problems such as finding the min cut and the maximum cardinality minimal dominating set [12] and minimum dominating set for SPs [12], to name a few see [16]. ?

The main contribution of this paper are presenting polynomial time algorithms to find $[1,2]$-sets and total $[1,2]$-sets with minimum cardinality for GSP graphs.

The paper is organized as follows: In Section 2, we study some notation and terminology of dominations and total dominations. In Section 3, we describe generalized series-parallel graphs and corresponding parse tree. In Section 4, we present a linear time algorithm to compute a $[1,2]$-set with minimum cardinality for generalized series-parallel graphs. In addition in this section we analyze correctness and complexity of the algorithms. In Section 6, we change procedures of the algorithm to find minimum total $[1,2]$-sets in generalized series-parallel graphs.

2. Terminology and notation

In this section, we review some required terminology and notations of graph theory. For other notation which is not defined here, the reader is refer to [18].

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

Let $G = (V, E)$ be a simple graph. The open and closed neighborhoods of a vertex $v \in V(G)$ are defined as

$$N_G(v) = \{u : uv \in E(G)\},$$

and

$$N_G[v] = N_G(v) \cup \{v\},$$

respectively. Note that we omit the subscript G, whenever there is no risk of confusion.

A subset of vertices of G such as D is called a dominating set, if every $v \in V$ is either an element of D or is adjacent to an element of D. In other words, for any $v \in V \backslash D$, it is the case that $|N(v) \cap D| \geq 1$. Similarly, a set D of vertices of G is called a total dominating set, if every $v \in V$ is adjacent to an element of D, i.e. $|N(v) \cap D| \geq 1$ holds for all $v \in V$. A γ-set is a dominating set of G with minimum cardinality. The size of such a set is called domination number of G and is denoted by $\gamma(G)$. Similarly, a γ_t-set is a total dominating set of G with minimum cardinality and total domination number of G is the cardinality of such a set which is denoted by $\gamma_t(G)$.

A set $S \subseteq V$ is called a $[1, 2]$-set of G if for each $v \in V \setminus S$ we have $1 \leq |N(v) \cap S| \leq 2$, i.e. v is adjacent to at least one but not more than two vertices in S. The size of the smallest $[1, 2]$-sets of G is denoted by $\gamma_{[1,2]}(G)$. Any such set is called a $\gamma_{[1,2]}$-set of G.

As a generalization of $[1, 2]$-sets, for two nonnegative integers j and k, where $j \leq k$, a set $S \subseteq V$ is an $[j, k]$-set if for each $v \in V \setminus S$ we have $j \leq |N(v) \cap S| \leq k$. A set $S' \subseteq V$ is called a total $[j, k]$-set of G if for each $v \in V$ we have $j \leq |N(v) \cap S| \leq k$, i.e. if each vertex, no matter in S' or not, has at least j and at most k neighbors in S'. These definitions can be found in [4].

3. Generalized Series-Parallel Graphs and Parse Tree

First we define the class of graphs that we study in this paper. In addition, we review some known results about them.

Definition 3.1 (Generalized Series-Parallel Graphs)**.** [3] *A generalized series-parallel (GSP) graph is any graph $G = (V, E, s, t)$ with two distinguished nodes $s, t \in V$, called terminals, which is defined recursively as follows:*

(1) o_i: *The graph G consisting of two vertices connected by a single edge is a GSP graph.*

(2) o_s: *Given two GSP graphs $G_1 = (V_1, E_1, s_1, t_1)$, $G_2 = (V_2, E_2, s_2, t_2)$, the series operation of G_1 and G_2 creates a new GSP graph $G = G_1 o_s G_2 = (V, E, s_1, t_2)$, where*

$$V = V_1 \cup V_2 \setminus \{s_2\},$$

and

$$E = E_1 \cup E_2 \cup \{t_1 v : v \in N_{G_2}(s_2)\} \setminus \{s_2 v : v \in N_{G_2}(s_2)\}.$$

(3) o_p: *Given two GSP graphs $G_1 = (V_1, E_1, s_1, t_1)$, $G_2 = (V_2, E_2, s_2, t_2)$, the parallel operation of G_1 and G_2 creates a new GSP graph $G = G_1 o_p G_2 = (V, E, s_1, t_1)$, where*

$$V = V_1 \cup V_2 \setminus \{s_2, t_2\}, \text{and}$$

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

$$E = E_1 \cup E_2 \cup \{s_1 v : v \in N_{G_2}(s_2)\} \cup \{t_1 v : v \in N_{G_2}(t_2)\} \setminus (\{s_2 v : v \in N_{G_2}(s_2)\} \cup \{t_2 v : v \in N_{G_2}(t_2)\})$$

(4) o_g: *Given two GSP graphs* $G_1 = (V_1, E_1, s_1, t_1)$, $G_2 = (V_2, E_2, s_2, t_2)$, *the generalized series operation of* G_1 *and* G_2 *creates a new GSP graph* $G = G_1 o_g G_2 = (V, E, s_1, t_1)$, *where*

$$V = V_1 \cup V_2 \setminus \{s_2\}, \text{and}$$

$$E = E_1 \cup E_2 \cup \{t_1 v : v \in N_{G_2}(s_2)\} \setminus \{s_2 v : v \in N_{G_2}(s_2)\}.$$

(5) *Any GSP graph is obtained by finite application of rules* (1) *through* (4).

Figure 2 illustrates the example of these rules to construct a GSP graph.

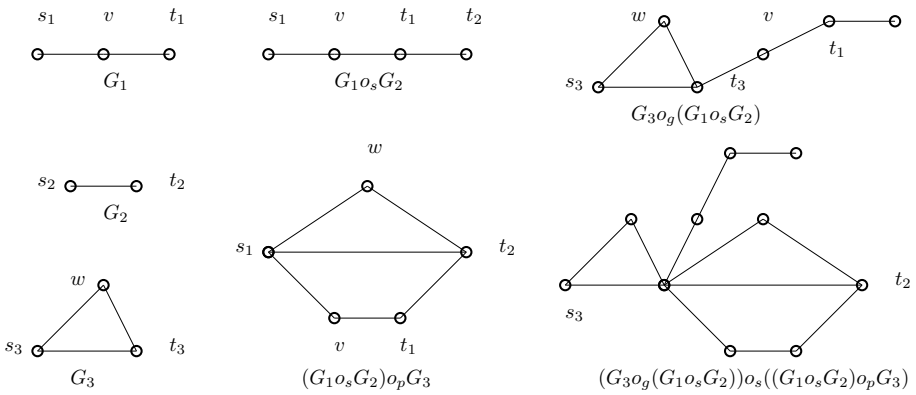


FIGURE 2. Example of operations for constructing GSP graphs

Moreover, the subclass of GSPs obtained by finite application of rules (1) to (3) is called series-parallel or SP class. Note that in Figure 2 the final graph $(\hat{G} o_g (G_1 o_s G_2)) o_s ((G_1 o_s G_2) o_p \hat{G})$ is a GSP graph but it is not SP.

p-graphs are defined as an special kind of subgraphs of GSP graph which are reviewed next.

Definition 3.2 (p-graph)**.** *Let* $G = (V, E, x, y)$ *be a GSP and* $\hat{G} = (\hat{V}, \hat{E}, \hat{x}, \hat{y})$ *subgraph of* G *satisfy the following conditions:*

(1) $\hat{x} = x$ *or there exists an edge* $\{u, v\} \in E \setminus \hat{E}$ *such that* $x \notin \hat{V}$ *and* $v = \hat{x} \in \hat{V}$.

(2) $\hat{y} = y$ *or there exists an edge* $\{w, z\} \in E \setminus \hat{E}$ *such that* $w \notin \hat{V}$ *and* $z = \hat{y} \in \hat{V}$.

then $\hat{G}$ *is called a p-graph of* G.

Remark 3.3. *Note that the concept of p-graph is defined for SPs in* [15]*, we have extended its definition to GSPs.*

A generalized series-parallel graph G can be represented by a binary parse tree T which is defined as follows.

Definition 3.4 (Binary Parse Tree for Generalized Series-Parallel Graphs)**.** [15] *The binary parse tree of GSP* G *is defined recursively as follows:*

(1) *A tree consisting of a single vertex labeled* $(u, v)_i$ *is a binary parse tree of primitive GSP* $G = (\{u, v\}, \{\{u, v\}\})$.

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

(2) *Let $G = (V, E)$ be a GSP by some composition of two other GSPs G_1 and G_2, and T_1 and T_2 be the binary parse trees of them, respectively. Then, the binary parse tree of G is a tree with the root r labeled as either $(u, v)_s$, $(u, v)_p$ or $(u, v)_g$ depending on which operation is applied to generate G. Vertices u and v are terminals of G and roots of T_1 and T_2 are the left and right children of r, respectively.*

Obviously, for any binary parse tree of GSP G, every internal vertex of G has exactly two children and there are $|E|$ leaves.

Note that when we use a label (x, y) , we do not care about the label being either i, s, p and g.

Let t be an internal vertex of T and $\tau(t)$ denote the subtree of T rooted at t . Also the left and right subtree of t are denoted as $\tau_l(t)$ and $\tau_r(t)$, respectively. Then the vertices of T are labeled as follows:

(a) *For each edge $e = \{x, y\} \in E$, there exists exactly one leaf which is labeled by (x, y) in T.*

(b) *For each internal vertex $t \in V_T$ labeled as $(x, y)_s$, the root of $\tau_l(t)$ is labeled as (x, z) and the root of $\tau_r(t)$ is labeled as (z, y), where z is some vertex of V. These vertices are called s-vertices.*

(c) *For each internal vertex $t \in V_T$ is labeled as $(x, y)_p$, the root of $\tau_l(t)$ and $\tau_r(t)$ are labeled as (x, y). These vertices are called p-vertices.*

(d) *For each internal vertex $t \in V_T$ labeled $(x, z)_g$, the root of $\tau_l(t)$ is labeled as (x, z), and the root of $\tau_r(t)$ is labeled as (z, y), where z is a vertex of V. These vertices are called g-vertices.*

Figure 3 illustrates a binary parse tree for a GSP.

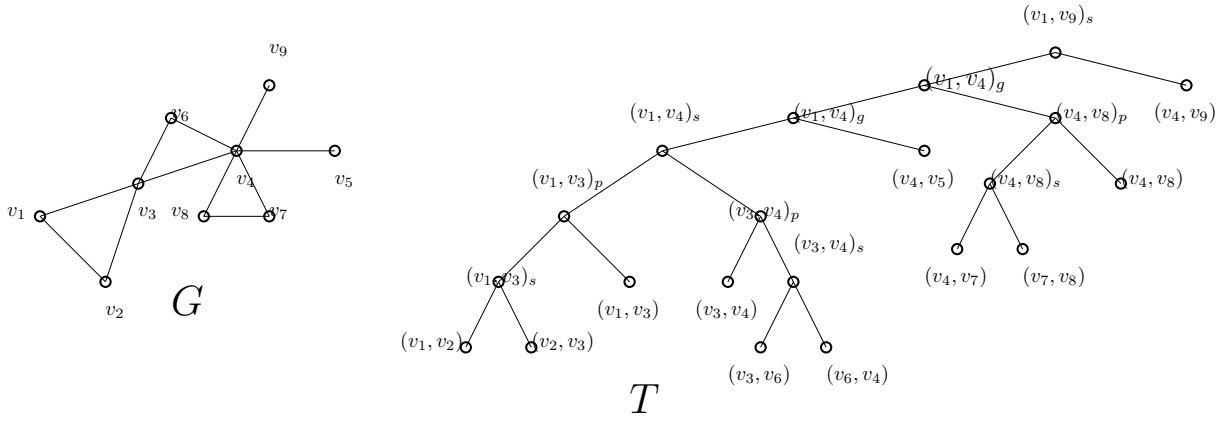


FIGURE 3. A GSP graph and a binary parse tree of it.

Note that the binary parse tree of a GSP graph is not necessarily unique.

Lemma 3.5. [14] *For a given GSP like G, a binary parse tree can be found in linear time.*

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

4. Basic blocks of algorithms for finding minimum $[1,2]$-set

We are given a GSP G, at the first step, this algorithm find a parse tree like T of given G. Each subtree of parse tree is corresponding to a p-graph of G. Four procedures ProcessLeaf, ProcessSvertex, ProcessPvertex and ProcessGvertex form the basic block of algorithm. Our goal is to find a $\gamma_{[1,2]}$-set for a given graph G.

4.1. **Definitions.** Let t is a vertex of T and $\hat{G}$ is a p-graph to subtree with root t of T. We define the following:

- For any vertex v of T, the set $ch(v)$ consist of all children of v. In this parse tree, if v is a leaf, $ch(v)$ will be an empty set and if it is internal vertex, $ch(v)$ will contain two elements.
- Let (x,y) be the label of v, $S(x_{i,j},y_{i',j'})$ is a subset of $V(\hat{G})$ like D, such that the number vertices of $V(\hat{G})\cap D$ and $V(G)\setminus V(\hat{G})\cap D$ which dominate x be i and j, respectively. Similarly i' vertex of $V(\hat{G})\cap D$ and j' vertex of $V(G)\setminus V(\hat{G})\cap D$ dominate y. In this algorithm if $i=j=0$ then $x\in D$ and if $i'=j'=0$ then $y\in D$. In a similar way, we can define $S_\ell(x_{i_\ell,j_\ell},y_{i'_\ell,j'_\ell})$ and $S_r(x_{i_r,j_r},y_{i'_r,j'_r})$ for the root of left and right subtree of v respectively.
- The function $Minsize$ received a number of sets as input and return a set among them with minimum cardinality.

This algorithm in each step of traversing binary parse tree, for each visiting vertex v of T recall one of the procedures ProcessLeaf, ProcessSvertex, ProcessPvertex and ProcessGvertex to find a subset of $S(x_{i,j},y_{i',j'})\subseteq V(\hat{G})$ such that junction of $S(x_{i,j},y_{i',j'})$ and a subset $V(G)\setminus V(\hat{G})$ form a $\gamma_{[1,2]}$-set for a p-graph $\hat{G}$ of G. So the following cases can not occur for every vertex $x\in V(\hat{G})$.

1. x is dominated by one vertex of $V(\hat{G})$ and two vertices of $V(G)\setminus V(\hat{G})$.
2. x is dominated by two vertices of $V(\hat{G})$ and one vertex of $V(G)\setminus V(\hat{G})$.
3. x is dominated by two vertices of $V(\hat{G})$ and two vertices of $V(G)\setminus V(\hat{G})$.

So $S(x_{i,j},y_{i',j'})$, will be computed for $(i,j),(i',j')\in M$ such that $M=\{(0,0),(0,1),(0,2),(1,0)(1,1),(2,0)\}$.

4.2. **Procedure for Leaves of T.** Input of this procedure is a leaf of $v\in V_T$ labeled by $(x,y)_i$ and output $S(x_{i,j},y_{i',j'})$ for the leaf labeled by $(x,y)_i$ and for all $(i,j),(i',j')\in M$. We compute $S(x_{i,j},y_{i',j'})$ for leaves using table 1.

TABLE 1. $S(x_{i,j},y_{i',j'})$ for different values of (i,j) in rows and (i',j') in columns

	$(0,0)$	$(0,1)$	$(0,2)$	$(1,0)$	$(1,1)$	$(2,0)$
$(0,0)$	$\{x,y\}$	$\emptyset$	$\emptyset$	$\{y\}$	$\{y\}$	NaN
$(0,1)$	$\emptyset$	$\emptyset$	$\emptyset$	NaN	NaN	NaN
$(0,2)$	$\emptyset$	$\emptyset$	$\emptyset$	NaN	NaN	NaN
$(1,0)$	$\{x\}$	NaN	NaN	NaN	NaN	NaN
$(1,1)$	$\{x\}$	NaN	NaN	NaN	NaN	NaN
$(2,0)$	NaN	NaN	NaN	NaN	NaN	NaN

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

1: **procedure** PROCESSLEAF(x, y)
2: $S(x_{0,0}, y_{0,0}) \leftarrow \{x, y\}$
3: $S(x_{0,0}, y_{1,0}) \leftarrow \{x\}$
4: $S(x_{0,0}, y_{1,1}) \leftarrow \{x\}$
5: $S(x_{1,0}, y_{0,0}) \leftarrow \{y\}$
6: $S(x_{1,1}, y_{0,0}) \leftarrow \{y\}$
7: **for all** $j = 0, 1, 2$ *and* $j' = 1, 2$ **do**
8: $S(x_{0,j}, y_{0,j'}) \leftarrow \emptyset$
9: **end for**
10: $S(x_{0,1}, y_{0,0}) \leftarrow \emptyset$
11: $S(x_{0,2}, y_{0,0}) \leftarrow \emptyset$
12: **for all** $(i, j) \in M$ **do**
13: $S(x_{i,j}, y_{2,0}) \leftarrow NaN$
14: $S(x_{2,0}, y_{i,j}) \leftarrow NaN$
15: **end for**
16: **for all** $(i, j) \in M \setminus \{(0, 0), (2, 0)\}$ **do**
17: $S(x_{i,j}, y_{1,0}) \leftarrow NaN$
18: $S(x_{i,j}, y_{1,1}) \leftarrow NaN$
19: $S(x_{1,0}, y_{i,j}) \leftarrow NaN$
20: $S(x_{1,1}, y_{i,j}) \leftarrow NaN$
21: **end for**
22: **end procedure**

A leaf of tree which is labeled by (x, y) is corresponding to an edge $\{x, y\}$ of GSP graph G. Following cases will occur to compute $S(x_{i,j}, y_{i',j'})$.

- $i = j = 0$ and $i' = j' = 0$: $x, y \in D$ and $S(x_{i,j}, y_{i',j'}) = \{x, y\}$.
- $i = 1$, $i' = j' = 0$: x is dominated by y and $S(x_{i,j}, y_{i',j'}) = \{y\}$.
- $i' = 1$, $i = j = 0$: y is dominated by x and $S(x_{i,j}, y_{i',j'}) = \{x\}$.
- $i = i' = 0$, $j \neq 0$ and $j' \neq 0$: x and y is dominated by one vertex or two vertices of vertex $V(G) \setminus V(\hat{G})$, so $x, y \notin D$ and $S(x_{i,j}, y_{i',j'})$ is an empty set.
- $i = 0$ and $j \in \{1, 2\}$: It implies that $x \notin D$ and it is dominated by some vertices $V(G) \setminus V(\hat{G})$, so $y \notin D$ and $S(x_{i,j}, y_{0,0})$ is undefinable. Similarly If $i' = 0$, $j' \in \{1, 2\}$ and $i = j = 0$ $S(x_{i,j}, y_{i',j'})$ is undefinable.
- $i = 1$, $i' \neq 0$ or $j' \neq 0$: $S(x_{i,j}, y_{i',j'})$ is undefinable. $i = 1$ implies that x is dominated by exactly one vertex of $\hat{G}$, this vertex is y. So $y \in D$ and $i' = j' = 0$, similarly for $i' = 1$, if $i \neq 0$ or $j \neq 0$, $S(x_{i,j}, y_{i',j'})$ is undefinable.
- $i = 2$, since there is exactly one vertex set of $\hat{G}$ to dominating x, $S(x_{i,j}, y_{i',j'})$ is undefinable and similarly for $i' = 2$, $S(x_{i,j}, y_{i',j'})$ is undefinable.

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

4.3. **Sets for s-vertices of T.** Let t is a vertex of T is labeled by $(x, y)_s$. In this procedure, the set $S(x_{i,j}, y_{i',j'})$ will computed for a given vertex x, y and common vertex z. Assume the sets corresponding to $\tau_l(t)$ and $\tau_r(t)$ are $S_\ell(x_{i_\ell,j_\ell}, z_{i'_\ell,j'_\ell})$ and $S_r(z_{i_r,j_r}, y_{i'_r,j'_r})$ respectively.

1: **procedure** ProcessSvertex(x, z, y)
2: **for all** $(i, j), (i', j') \in M$ **do**
3: $setlist \leftarrow \emptyset$
4: **for all** $(i'_\ell, j'_\ell) \in M$ **do**
5: Add $S_\ell(x_{i,j}, z_{i'_\ell,j'_\ell}) \cup S_r(z_{j'_\ell,i'_\ell}, y_{i',j'})$ to $setlist$
6: **end for**
7: $S(x_{i,j}, y_{i',j'}) \leftarrow Minsize(setlist)$
8: **end for**
9: **end procedure**

Let root of $\tau_l(t)$ and $\tau_r(t)$ labeled by (x, z) and (z, y) respectively, for some $z \in V$. Let $S_\ell(x_{i_\ell,j_\ell}, z_{i'_\ell,j'_\ell})$ and $S_r(z_{i_r,j_r}, y_{i'_r,j'_r})$ be sets associated with the vertex (x, z) and (z, y) of T. Now for each s-vertex of T, we can compute sets $S(x_{i,j}, y_{i',j'})$ for all $(i, j), (i', j') \in M$ as follow,

$$S(x_{i,j}, y_{i',j'}) = Minsize_{(i,j),(i',j')\in M}\{S_\ell(x_{i,j}, z_{i'_\ell,j'_\ell}) \cup s_r(x_{j'_\ell,i'_\ell}, y_{i',j'})\} \tag{4.1}$$

To prove the correctness of formula 4.1, let $\tau_l(t)$, $\tau_r(t)$ and $\tau(t)$ represent vertices of T which is corresponding p-graphs $G_1 = (V_1, E_1, x, z)$, $G_2 = (V_2, E_2, z, y)$ and $\hat{G} = G_1 o_s G_2 = (\hat{V}, \hat{E}, x, y)$ respectively. Figures 5-4 show the computation $S(x_{i,j}, y_{i',j'})$ when z is dominated by vertices of G_1 or vertices of G_2. According to the number of vertices in $D \cap V(G_1)$, $D \cap (V(G) \setminus V(G_1))$, $D \cap V(G_2)$ and $D \cap (V(G) \setminus V(G_2))$ that dominate z, the following cases will occur.

- $z \in D$, in this case $(0, 0) \in M$, see Figure 4.

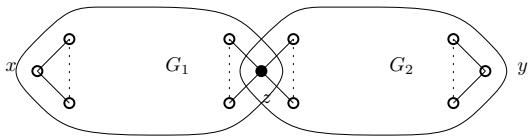


Figure 4. $z \in D$.

- $z \notin D$, if z is dominated by one vertex of G_1 then $(0, 1) \in M$, see Figure 5.a or if it is dominated by one vertex of G_2 then then $(1, 0) \in M$, see Figure 5.b.

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

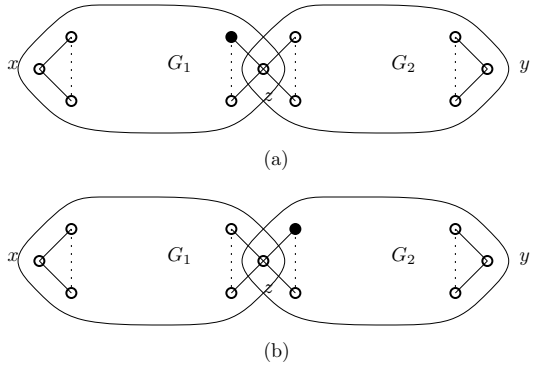


FIGURE 5. z is dominated once.

- $z \notin D$, if it is dominated by two vertices of G_1 then $(0,2) \in M$, see Figure 6.a or if it is dominated by two vertices of G_2 then $(2,0) \in M$, see Figure6.b.

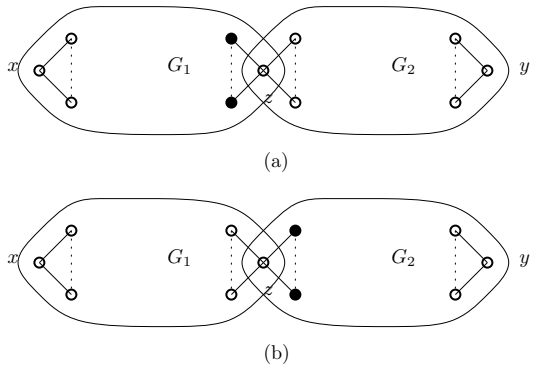


FIGURE 6. z is dominated by two vertices of $V(G_1)$ or $V(G_2)$.

- $z \notin D$, if it is dominated by one vertex of G_1 and it is dominated by one vertex of G_2 then $(1,1) \in M$, see Figure7.

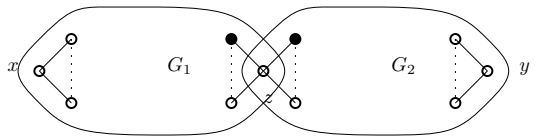


FIGURE 7. z is dominated by one vertex of $V(G_1)$ and the other of $V(G_2)$.

4.4. **Sets for p-vertices of T.** Let t is a vertex of T is labeled by $(x,y)_p$, in this procedure, the set $S(x_{i,j}, y_{i',j'})$ will computed for a given vertex x, y. The sets corresponding to $\tau_l(t)$ and $\tau_r(t)$ are $S_\ell(x_{i_\ell,j_\ell}, y_{i'_\ell,j'_\ell})$ and $S_r(x_{i_r,j_r}, y_{i'_r,j'_r})$ respectively.

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

procedure ProcessPvertex(x, y)
  for all $(i,j),(i',j') \in M$ **do**
    $setlist[(i,j),(i',j')] \leftarrow \emptyset$
  end for
  for all $(i,j) \in \{(0,0),(0,1),(0,2)\}$ **do**
    for all $(i',j') \in \{(0,0),(0,1),(0,2)\}$ **do**
      $setlist[(i,j),(i',j')] \leftarrow S_\ell(x_{i,j}, y_{i',j'}) \cup S_r(x_{i,j}, y_{i',j'})$
    end for
    $Setlist[(i,j),(1,0)] \leftarrow \{S_\ell(x_{i,j}, y_{1,0}) \cup S_r(x_{i,j}, y_{0,1}), S_\ell(x_{i,j}, y_{0,1}) \cup S_r(x_{i,j}, y_{1,0})\}$
    $Setlist[(i,j),(1,1)] \leftarrow \{S_\ell(x_{i,j}, y_{1,1}) \cup S_r(x_{i,j}, y_{0,2}), S_\ell(x_{i,j}, y_{0,2}) \cup S_r(x_{i,j}, y_{1,1})\}$
    $Setlist[(i,j),(2,0)] \leftarrow \{S_\ell(x_{i,j}, y_{2,0}) \cup S_r(x_{i,j}, y_{0,2}), S_\ell(x_{i,j}, y_{0,2}) \cup S_r(x_{i,j}, y_{2,0}),$
    $S_\ell(x_{i,j}, y_{1,1}) \cup S_r(x_{i,j}, y_{1,1})\}$
    $Setlist[(1,0),(i,j)] \leftarrow \{S_\ell(x_{1,0}, y_{i,j}) \cup S_r(x_{0,1}, y_{i,j}), S_\ell(x_{0,1}, y_{i,j}) \cup S_r(x_{1,0}, y_{i,j})\}$
    $Setlist[(1,1),(i,j)] \leftarrow \{S_\ell(x_{1,1}, y_{i,j}) \cup S_r(x_{0,2}, y_{i,j}), S_\ell(x_{0,2}, y_{i,j}) \cup S_r(x_{1,1}, y_{i,j})\}$
    $Setlist[(2,0),(i,j)] \leftarrow \{S_\ell(x_{2,0}, y_{i,j}) \cup S_r(x_{0,2}, y_{i,j}), S_\ell(x_{0,2}, y_{i,j}) \cup S_r(x_{2,0}, y_{i,j}),$
    $S_\ell(x_{1,1}, y_{i,j}) \cup S_r(x_{1,1}, y_{i,j})\}$
  end for
  for all $(i,j),(i',j') \in \{(0,1),(1,0)$ **do**
    Add $S_\ell(x_{i,j}, y_{i',j'}) \cup S_r(x_{j,i}, y_{j',i'})$ to $setlist[(i,j),(i',j')]$
  end for
  for all $(i,j) \in \{(0,1),(1,0)$ **do**
    Add $S_\ell(x_{i,j}, y_{1,1}) \cup S_r(x_{j,i}, y_{0,2})$ to $setlist[(i,j),(1,1)]$
    Add $S_\ell(x_{i,j}, y_{0,2}) \cup S_r(x_{j,i}, y_{1,1})$ to $setlist[(i,j),(1,1)]$
    Add $S_\ell(x_{i,j}, y_{0,2}) \cup S_r(x_{j,i}, y_{2,0})$ to $setlist[(1,0),(2,0)]$
    Add $S_\ell(x_{i,j}, y_{2,0}) \cup S_r(x_{j,i}, y_{0,2})$ to $setlist[(1,0),(2,0)]$
    Add $S_\ell(x_{i,j}, y_{1,1}) \cup S_r(x_{j,i}, y_{1,1})$ to $setlist[(1,0),(2,0)]$
    Add $S_\ell(x_{1,1}, y_{i,j}) \cup S_r(x_{0,2}, y_{j,i})$ to $setlist[(1,1),(1,0)]$
    Add $S_\ell(x_{0,2}, y_{i,j}) \cup S_r(x_{1,1}, y_{j,i})$ to $setlist[(1,1),(1,0)]$
  end for
  $setlist[(1,1),(1,1)] \leftarrow \{S_\ell(x_{1,1}, y_{1,1}) \cup S_r(x_{0,2}, y_{0,2}), S_\ell(x_{1,1}, y_{0,2}) \cup S_r(x_{0,2}, y_{1,1}),$
  $S_\ell(x_{0,2}, y_{1,1}) \cup S_r(x_{1,1}, y_{0,2}), S_\ell(x_{0,2}, y_{0,2}) \cup S_r(x_{1,1}, y_{1,1})\}$

Since the number of vertices of $V(G_1)$ and $V(G_2)$ dominate x make changes in value of i and similarly the number of vertices of $V \setminus V(G_1)$ and $V \setminus V(G_2)$ make changes in value of j, for each $(i,j) \in M$ we describe the method for finding $S(x_{i,j}, y_{i',j'})$. It is enough to find a relation between values of (i,j), (i_ℓ, j_ℓ) and (i_r, j_r) in formula 4.2.

$$S(x_{i,j}, y_{i',j'}) = Minsize\{S_\ell(x_{i_\ell,j_\ell}, y_{i'_\ell,j'_\ell}) \cup s_r(x_{i_r,j_r}, y_{i'_r,j'_r})\}. \tag{4.2}$$

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

for all $(i',j') \in \{(0,2),(2,0),(1,1)\}$ **do**
  Add $S_\ell(x_{1,1}, y_{i',j'}) \cup S_r(x_{0,2}, y_{j',i'})$ to $setlist[(1,1),(2,0)]$
  Add $S_\ell(x_{1,1}, y_{i',j'}) \cup S_r(x_{0,2}, y_{j',i'})$ to $setlist[(1,1),(2,0)]$
end for
for all $(i,j) \in \{(0,2),(2,0),(1,1)\}$ **do**
  Add $S_\ell(x_{i,j}, y_{1,1}) \cup S_r(x_{j,i}, y_{0,2})$ to $setlist[(2,0),(1,1)]$
  Add $S_\ell(x_{i,j}, y_{0,2}) \cup S_r(x_{j,i}, y_{1,1})$ to $setlist[(2,0),(1,1)]$
end for
for all $,(i,j)(i',j') \in \{(0,2),(2,0),(1,1)\}$ **do**
  Add $S_\ell(x_{i,j}, y_{i',j'}) \cup S_r(x_{j,i}, y_{j',i'})$ to $setlist[(2,0),(2,0)]$
end for
for all $(i,j),(i',j') \in M$ **do**
  $S(x_{i,j}, y_{i',j'}) \leftarrow Minsize(setlist[(i,j),(i',j')])$
end for
end procedure

To find this relation, let $\tau_l(t)$, $\tau_r(t)$ and $\tau(t)$ of T corresponding to p-graphs $G_1 = (V_1, E_1, x, y)$, $G_2 = (V_2, E_2, x, y)$ and $\hat{G} = G_1 o_p G_2 = (\hat{V}, \hat{E}, x, y)$ respectively. Now according to (i,j) (resp. (i',j')) values for (i_ℓ, j_ℓ) and (i_r, j_r) (resp. (i'_ℓ, j'_ℓ) and (i'_r, j'_r)) are determined.

- $x \in D$, in formula 4.2, if $i = j = 0$ then $i_\ell = j_\ell = i_r = j_r = 0$. We define:

$$S(x_{0,0}, y_{i',j'}) = Minsize\{s_\ell(x_{0,0}, y_{i'_\ell, j'_\ell}) \cup s_r(x_{0,0}, y_{i'_r, j'_r})\}.$$

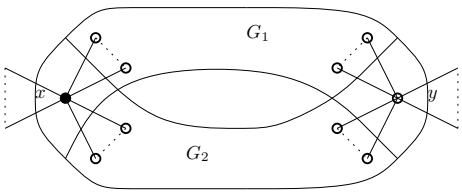


FIGURE 8. $x \in D$

- $x \notin D$, if x is not dominated by any vertex of $V(\hat{G})$ then it is dominated by one vertex or two vertices of $V(G) \setminus V(\hat{G})$, see Figure 9. In the left side of formula 4.2, $i = 0$ and $j \neq 0$, so $i_\ell = i_r = 0$ and $j_\ell = j_r = j$.

  We define $S(x_{0,1}, y_{i',j'}) = Minsize_{i',j' \in \{0,1,2\}}\{S_\ell(x_{0,1}, y_{i'_\ell, j'_\ell}) \cup s_r(x_{0,1}, y_{i'_r, j'_r})\}$, see Figure 9.a, and $S(x_{0,2}, y_{i',j'}) = Minsize\{S_\ell(x_{0,2}, y_{i'_\ell, j'_r}) \cup s_r(x_{0,2}, y_{i'_r, j'_r})\}$, see Figure 9.b.
- $x \notin D$, if x is dominated by exactly one vertex of $V(\hat{G})$ then, it is dominated by a vertex of G_1 that is out of G_2, see Figure 10.a or it is dominated by a vertex of G_2 that is out of G_1, see Figure 10.b.

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

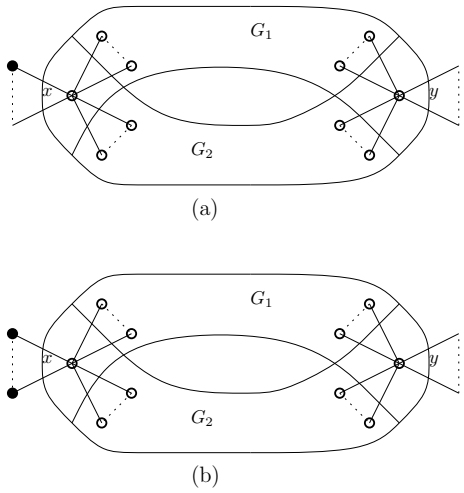


FIGURE 9. x is not dominated one vertex or two vertices of $V(G) \setminus V(\hat{G})$.

It means that in formula 4.2, if $i = 1$ and $j = 0$ then $i_\ell = j_r = 0$ and $j_\ell = i_r = 1$ or $i_\ell = j_r = 1$ and $j_\ell = i_r = 0$. We define: $S(x_{1,0}, y_{i',j'}) = Minsize_{i',j' \in \{0,1,2\}}\{S_\ell(x_{1,0}, y_{i'_\ell,j'_r}) \cup s_r(x_{0,1}, y_{i'_r,j'_r}), S_\ell(x_{0,1}, y_{i'_\ell,j'_r}) \cup s_r(x_{1,0}, y_{i'_r,j'_r})\}$.

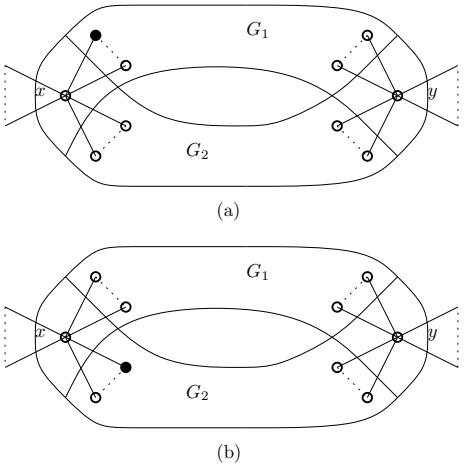


FIGURE 10. x is dominated by exactly one vertex of $V(\hat{G})$.

- $x \notin D$, if x is dominated by exactly two vertices of $V(\hat{G})$, then one of the following cases occurs:
  1. x is dominated by two vertex of G_1 that they are out of G_1, see Figure 11.a.
  2. x is dominated by two vertices of G_2 that they are out of G_1, see Figure 11.b.
  3. x is dominated by exactly one vertex of G_1 and exactly one vertex of G_2, see Figure 11.c.

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

In formula 4.2, if $i = 2$ and $j = 0$ then $(i_\ell, j_\ell (= (0,2), (2,0)$ or $(1,1)$ and $i_r = j_\ell,\ j_r = i_\ell$. So that, we define:

$$\begin{aligned} S(x_{2,0}, y_{i',j'}) = Minsize \quad & \{S_\ell(x_{2,0}, y_{i'_\ell, j'_r}) \cup s_r(x_{0,2}, y_{i'_r, j'_r}) \\ & S_\ell(x_{0,2}, y_{i'_\ell, j'_r}) \cup s_r(x_{2,0}, y_{i'_r, j'_r}), \\ & S_\ell(x_{1,1}, y_{i'_\ell, j'_r}) \cup s_r(x_{1,1}, y_{i'_r, j'_r})\}. \end{aligned}$$

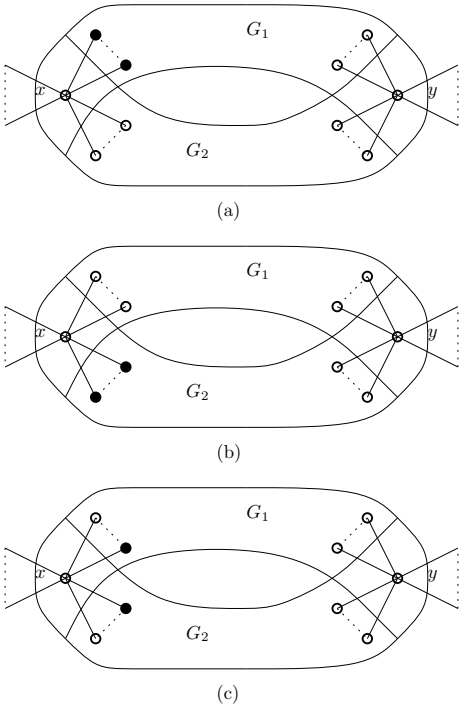


FIGURE 11. x is dominated by exactly two vertices of $V(\hat{G})$.

- $x \notin D$, if x is dominated by exactly one vertex of $\hat{G}$ and exactly one vertex out of $\hat{G}$ then one of the cases hold
  1. x is dominated by one vertex of G_1 and a vertex out of $G_1 o_p G_2$ see Figure **??**.a.
  2. x is dominated by one vertex of G_2 and a vertex out of $G_1 o_p G_2$ see Figure **??**.b.

  In formula 4.2, if $i = 1$ and $j = 1$ then $(i_\ell, j_\ell) = (1,1), (i_r, j_r) = (0,2)$ or $(i_\ell, j_\ell) = (0,2), (i_r, j_r) = (1,1)$. So that we define:

$$S(x_{1,1}, y_{i',j'}) = Minsize\{S_\ell(x_{1,1}, y_{i'_\ell, j'_\ell}) \cup s_r(x_{0,2}, y_{i'_r, j'_r}), S_\ell(x_{0,2}, y_{i'_\ell, j'_\ell}) \cup s_r(x_{1,1}, y_{i'_r, j'_r})\}.$$

??

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

4.5. **Sets for g-vertices of T.** Let t is a vertex of T is labeled by $(x,y)_g$. In this procedure, the set $S(x_{i,j},y_{i',j'})$ will computed for a given vertex x, y. The sets corresponding to $\tau_l(t)$ and $\tau_r(t)$ are $S_\ell(x_{i_\ell,j_\ell},y_{i'_\ell,j'_\ell})$ and $S_r(x_{i_r,j_r},y_{i'_r,j'_r})$ respectively. Let the roots of $\tau_l(t)$ and $\tau_r(t)$ are labeled by (x,y)

---

procedure ProcessGvertex(x,y,z)
  for all $(i,j)\in M$ **do**
    $setlist[i,j]\leftarrow\emptyset$
  end for
  for all $(i,j)\in M$ **do**
    for all $(i'_r,j'_r)\in M$ **do**
      for all $(i',j')\in\{(0,0),(0,1),(0,2)\}$ **do**
        Add $S_\ell(x_{i,j},y_{i',j'})\cup S_r(y_{i',j'},z_{i'_r,j'_r})$ to $setlist[i',j']$
      end for
      Add $S_\ell(x_{i,j},y_{0,1})\cup S_r(y_{1,0},z_{i'_r,j'_r})$ to $setlist[1,0]$
      Add $S_\ell(x_{i,j},y_{1,0})\cup S_r(y_{0,1},z_{i'_r,j'_r})$ to $setlist[1,0]$
      Add $S_\ell(x_{i,j},y_{1,1})\cup S_r(y_{0,2},z_{i'_r,j'_r})$ to $setlist[1,1]$
      Add $S_\ell(x_{i,j},y_{0,2})\cup S_r(y_{1,1},z_{i'_r,j'_r})$ to $setlist[1,1]$
      Add $S_\ell(x_{i,j},y_{2,0})\cup S_r(y_{0,2},z_{i'_r,j'_r})$ to $setlist[2,0]$
      Add $S_\ell(x_{i,j},y_{0,2})\cup S_r(y_{2,0},z_{i'_r,j'_r})$ to $setlist[2,0]$
      Add $S_\ell(x_{i,j},y_{1,1})\cup S_r(y_{1,1},z_{i'_r,j'_r})$ to $setlist[2,0]$
    end for
    $S(x_{i,j},y_{i',j'})\leftarrow Minsize(setlist[i',j'])$
  end for
end procedure

---

and (y,z) respectively, for some $z\in V$. Let $s_\ell(x_{i_\ell,j_\ell},y_{i'_\ell,j'_\ell})$ and $s_r(y_{i_r,j_r},z_{i'_r,j'_r})$ be associated with the vertex (x,y) and (y,z) of T. If a vertex $w\in V(\hat{G})$ (resp. $V(G)\setminus V(\hat{G})$) dominate x then, $w\in V(G_1)$ (resp. $V(G)\setminus V(G_1)$), so $i_\ell=i$ and $j_\ell=j$. In this kind of operation, number of vertices $V(G_1)$, $(V(G)\setminus V(G_1)$, $V(G_2)$ and $(V(G)\setminus V(G_2)$ make changes in i' and j'. So for each $(i,j)\in M$ define:

$$S(x_{i,j},y_{i',j'})=Minsize_{(i'_r,j'_r)\in M}\{s_\ell(x_{i,j},y_{i'_\ell,j'_\ell})\cup s_r(y_{i_r,j_r},z_{i'_r,j'_r})\} \tag{4.3}$$

To find a relation between (i',j'), (i'_ℓ,j'_ℓ) and (i_r,j_r) for different value of (i'_r,j'_r), let $\tau_l(t)$, $\tau_r(t)$ and $\tau(t)$ of T corresponding to p-graphs G_1, G_2 and $\hat{G}=G_1o_gG_2$. According to the number of vertices $D\cap V(G_1)$ and $D\cup V(G_2)$ that dominate y, the following cases occur:

- $y\in D$, see Figure 13.

  In formula 4.3, if $i'=0$ and $j'=0$ then $i'_\ell=j'_\ell=0$, $i_r=j_r=0$ and $j'_r=0$, now we define:

$$S(x_{i,j},y_{0,0})=Minsize_{(i'_r,j'_r)\in M}\{s_\ell(x_{i,j},y_{0,0})\cup s_r(y_{0,0},z_{i'_r,j'_r})\}.$$

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

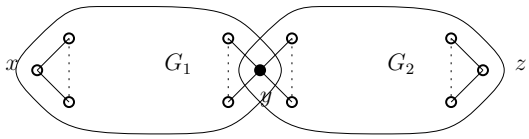


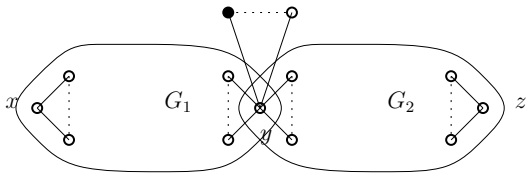
FIGURE 13. $y \in D$

- $y \notin D$, if y is not dominated by any vertex of $\hat{G}$ then, it is dominated by one vertex or two vertices in $V(G) \setminus V(\hat{G})$, see Figure 14.

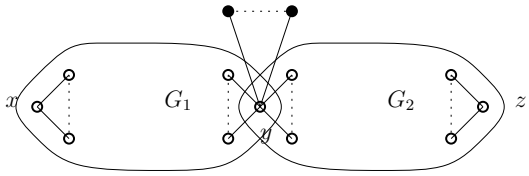
  In the left side of formula 4.3, if $i' = 0$ and $j' \neq 0$ then $i'_\ell = i_r = 0$ and $j'_\ell = j_r = j$. So that we define

$$S(x_{i,j}, y_{0,1}) = Minsize_{(i'_r,j'_r)\in M}\{s_\ell(x_{i,j}, y_{0,1}) \cup s_r(y_{0,1}, z_{i'_r,j'_r})\},$$

$$S(x_{i,j}, y_{0,2}) = Minsize_{(i'_r,j'_r)\in M}\{s_\ell(x_{i,j}, y_{0,2}) \cup s_r(y_{0,2}, z_{i'_r,j'_r})\}.$$



(a)



(b)

FIGURE 14. y is dominated by one vertex or vertices of $V(G) \setminus V(\hat{G})$.

- $y \notin D$, if y is dominated by exactly one vertex $w \in V(\hat{G})$ and it is not dominated by any vertex of $V(G) \setminus V(\hat{G})$, then $w \in V(G_1)$, see Figure 15.a or $w \in V(G_2)$ see Figure 15.b.

  In formula 4.3, if $i' = 1$ and $j' = 0$ then $(i'_\ell, j'_\ell) = (0, 1)$ *or* $(1, 0)$, $i_r = j'_\ell$ and $j_r = i'_\ell$. So that we define:

$$S(x_{i,j}, y_{1,0}) = Minsize_{(i'_r,j'_r)\in M}\{s_\ell(x_{i,j}, y_{0,1}) \cup s_r(y_{1,0}, z_{i'_r,j'_r}), s_\ell(x_{i_\ell,j_\ell}, y_{1,0}) \cup s_r(y_{0,1}, z_{i'_r,j'_r})\}.$$

- $y \notin D$, let y is dominated by exactly two vertices of $V(\hat{G})$ and it is not dominated by any $V(G) \setminus V(\hat{G})$ then one of the following cases occur
  1. y is dominated by two vertices of $V(G_1)$, see Figure 16.a,
  2. y is dominated by two vertices of $V(G_2)$, see Figure 16.b,
  3. y is dominated by one vertex of $V(G_1)$ and one vertex of $V(G_2)$, see Figure 16.c.

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

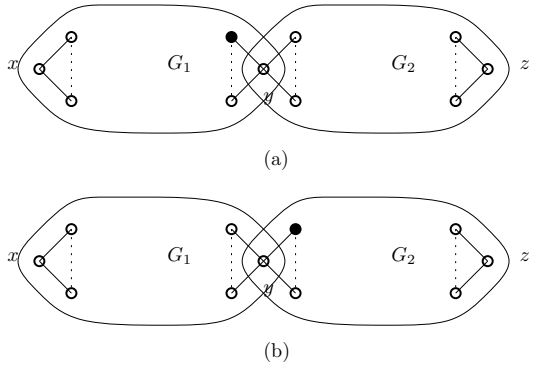


FIGURE 15. y is dominated by one vertex of $V(\hat{G})$.

In formula 4.3, if $i'=2$ and $j'=0$ then $(i'_\ell, j'_\ell)=(0,2),(2,0)$ *or* $(1,1)$ and $i_r=j'_\ell$, $j_r=i'_\ell$. we define:

$$S(x_{i,j},y_{2,0})=Minsize_{(i'_r,j'_r)\in M} \quad \{s_\ell(x_{i,j},y_{2,0})\cup s_r(y_{0,2},z_{i'_r,j'_r}),$$
$$s_\ell(x_{i,j},y_{0,2})\cup s_r(y_{2,0},z_{i'_r,j'_r}),$$
$$s_\ell(x_{i,j},y_{1,1})\cup s_r(y_{1,1},z_{i'_r,j'_r})\}.$$

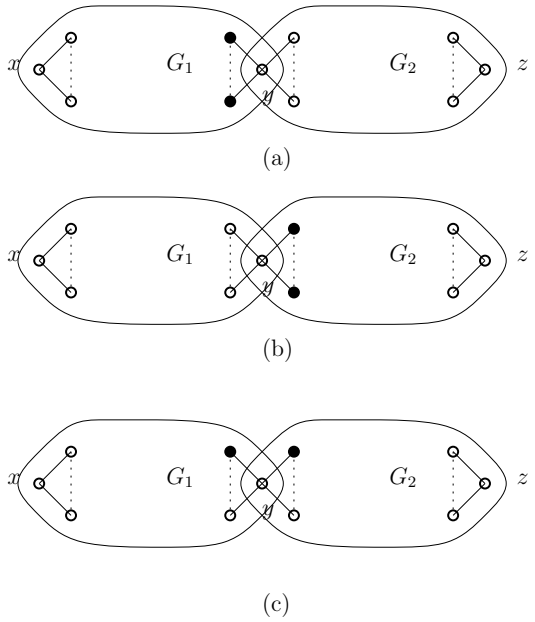


FIGURE 16. y is dominated by two vertices of $V(\hat{G})$.

- $y\notin D$, if y is dominated by one vertex of $\hat{G}$ and one vertex of $V(G)\setminus V(\hat{G})$, then y is dominated by one vertex of $V(G_1)$ and the other out of $V(G_1o_gG_2)$ see Figure 17.a or one vertex of $V(G_2)$ and the other out of $V(G_1o_gG_2)$ see Figure 17.b.

  In formula 4.3, if $i'=1$ and $j'=1$ then $(i'_\ell, j'_\ell)=(1,1)$ and $(i_r,j_r)=(0,2)$ or$(i'_\ell, j'_\ell)=(0,2)$ and $(i_r,j_r)=(1,1)$. So that we define:

$$S(x_{i,j},y_{1,1})=Minsize_{i'_r,j'_r\in\{0,1,2\}}\{s_\ell(x_{i,j},y_{1,1})\cup s_r(y_{0,2},z_{i'_r,j'_r}),s_\ell(x_{i_\ell,j_\ell},y_{0,2})\cup s_r(y_{1,1},z_{i'_r,j'_r})\}.$$

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

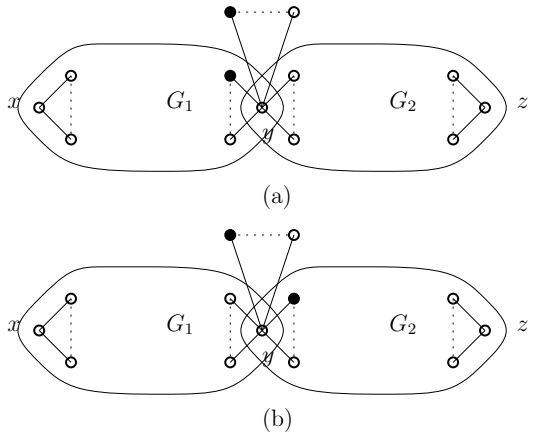


FIGURE 17. y is dominated by one vertex of $V(\hat{G})$ and one vertex of $V(G) \setminus V(\hat{G})$.

In each formula of subprocedures s-vertex, p-vertex and g-vertex, we delete any undefinable subformula of the right side of formula computing $S_i(x_{i,j}, y_{i',j'})$. Furthermore if all subformulas are deleted from the right hand side of some formula, then $S_i(x_{i,j}, y_{i',j'})$ will be undefinable.

5. **Algorithm** $[1,2]$**-MINSET**

At the first step of algorithm $[1,2]$-MINSET finding a parse tree for given GSP like T by some known linear algorithms. Each vertex v of T have a label such that $(x,y)_i$, $(x,y)_s$, $(x,y)_p$ or $(x,y)_g$, such that x and y are terminals of corresponding p-graph to $\tau(v)$. x and y will be inputs of procedures. By traversing parse tree T in bottom-up order and visiting vertices of T, proper procedure will be called. After visiting root of T and computing $S(x_{i,j}, y_{i',j'})$ for root(T), a $[1,2]$-set for G can be found. It is enough to return a $S(x_{i,j}, y_{i',j'})$ with minimum cardinality when $(i,j),(i',j') \in \{(0,0),(0,1),(0,2)\}$.

Theorem 5.1. *For a given generalized series-parallel graph $G = (V,E)$, algorithm $[1,2]$-MINSET find a minimum $[1,2]$-set in time $O(|V|)$.*

Proof. Algorithm $[1,2]$-MINSET, traversing parse tree T in bottom-up order, easily computes at most 36 set for each internal vertex of them. Each initial sets for leaves of tree represents all possible $[1,2]$-dominating sets in a graph containing only one edge. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be the graphs represented by the subtrees $\tau_l(t)$ and $\tau_r(t)$. Assume these are input of procedures ProcessLeaf, ProcessSvertex, ProcessPvertex and ProcessGvertex. It is easy to see that this procedure finds all possible minimum $[1,2]$-sets in graph. Finally, extracts only a valid minimum $[1,2]$-set. These step of algorithm require at most $|O(V_T)|$ operations. Since each binary tree with n leaves has $O(n)$ vertices and the binary parse tree of every GSP graph have has $|E(G)|$ leaves, so $|V_T| \in O(|E(G)|)$. Since every GSP graphs are planar and in planar graph we have $|E| \leq 3|V| - 6$. So in order to prove that algorithm $[1,2]$-MINSET computes a $\gamma_{[1,2]}$-set for a given GSP graph in time $O(|V|)$, it is sufficient to show that the parse tree T can be constructed in $O(|V|)$ operations that it is proved in [15]. □

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

Algorithm 1 :Finding a $\gamma_{[1,2]}$-sets of a GSP

find a parse tree of G like T
$S \leftarrow empty\ stack$ ▷ main stack with nodes from X
$Q \leftarrow empty\ queue$ ▷ auxiliary queue to build S
$Q.add(root\ r)$
$M \leftarrow \{(0,0),(0,1),(0,2),(1,0),(1,1),(2,0)\}$
while $Q \neq \emptyset$ **do**
    $x \leftarrow Q.pop()$
    $Q.add(ch(x))$
    $S.add(x)$
end while
while $S \neq \emptyset$ **do**
    $v \leftarrow S.pop()$ ▷ all children of v are processed
    switch $type\ of\ v$ **do**
        case $Leaf$
            ProcessLeaf(x,y) ▷ $(x,y)_i$ is the label of v
        case $s-vertex$
            ProcessSvertex(x,z,y) ▷ $(x,y)_s$, (x,z) and (z,y) are label of v, left and right child of v respectively.
        case $p-vertex$
            ProcessPvertex(v) ▷ $(x,y)_p$ is the label of v and the label of left and right child of v is (x,y).
        case $g-vertex$
            ProcessGvertex(x,y,z) ▷ $(x,y)_s$, (x,y) and (y,z) are label of v, left and right child of v respectively.
end while
$D \leftarrow \emptyset$
for all $(i,j),(i',j') \in \{(0,0),(0,1),(0,2)\}$ **do**
    Add $S(x_{i,j},y_{i',j'})$ to D
end for
$\gamma_{[1,2]}(G) \leftarrow Minsize(D)$

6. Algorithms for minimum total $[1,2]$-set

In this section, we make some changes in three procedures of $[1,2]-MINSET$ algorithm and define new procedures ProcessLeaf-t, ProcessSvertex-t, ProcessPvertex-t and ProcessGvertex-t to use in *total* $[1,2]-MINSET$ algorithm and compute a $\gamma_{t[1,2]}$-set for a GSP graph G while binary parse tree of G is computed in the first step of algorithm. These procedures compute set $S(x_{i,j,k},y_{i',j',k'})$

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

for leaves labeled by $(x,y)_i$ and internal vertices labeled by $(x,y)_s$, $(x,y)_p$, $(x,y)_g$ of in binary parse tree for all $i,j,i',j' \in \{0,1,2\}$ and $k,k' \in \{0,1\}$.

There is a connection between the sets computed for each vertex of T and the p-graph $\hat{G} = (\hat{V}, \hat{E})$. Let D is a total $[1,2]$-dominating set for $\hat{G}$, then for each vertex $x \in V(\hat{G})$, $x_{i,j,k}$ implies the following assertions with respect to i, j, k and D.

- $x_{i,j,0}$ implies that $x \notin D$, i vertices of $V(\hat{G})$ and j vertices of $V(G) \setminus V(\hat{G})$ that they adjacent to x belong to D.
- $x_{i,j,1}$ implies that $x \in D$, i vertices of $V(\hat{G})$ and j vertices of $V(G) \setminus V(\hat{G})$ that they adjacent to x belong to D.

Since every vertices of graph must be dominated by one vertex or two vertices of total $[1,2]$-set, it is clear, for all vertices of tree $S(x_{i,j,k}, y_{i',j',k'})$ is undefinable, when $(i,j),(i',j') \in \{(0,0),(1,2),(2,1),(2,2)\}$ and we define a new set as follow,

$$M_t = \{(0,1,0),(0,1,1),(1,0,0),(1,0,1),(0,2,0),(0,2,1),(2,0,0),(2,0,1),(1,1,0),(1,1,1)\}.$$

At the end of this algorithm a $\gamma_{t[1,2]}$-set for a GSP graph G with minimum size can be found.

6.1. **Procedure for Leaves of T.** Input of this procedure is a leaf of $v \in V_T$ labeled by $(x,y)_i$ and output $S(x_{i,j,k}, y_{i',j',k'})$ for the leaf labeled by $(x,y)_i$ and for all $(i,j,k),(i',j',k') \in M_t$. We compute $S(x_{i,j,k}, y_{i',j',k'})$ for leaves using table 1.

In this procedure, following cases will occur:

- $k = k' = 1$: $x, y \in D$ and $S(x_{i,j,k}, y_{i',j',k'}) = \{x,y\}$.
- $i = 1$, $k' = 1$: x is dominated by y and $S(x_{i,j,k}, y_{i',j',k'}) = \{y\}$.
- $i' = 1$, $k = 1$: y is dominated by x and $S(x_{i,j,k}, y_{i',j',k'}) = \{x\}$.
- $k = k' = 0$: x and y are dominated by one vertex or two vertices out of this leaf, so $x, y \notin D$ and $S(x_{i,j,0}, y_{i',j',0}) = \emptyset$.
- $i = 0$ and $k' = 1$: $S(x_{i,j,k}, y_{i',j',k'})$ is undefinable. $i = 0$ means that there is not any vertex of $\hat{G} = (\hat{V}, \hat{E})$ to dominate x, so $y \notin D$ and $S(x_{0,j,k}, y_{i',j',k'})$ is definable if $k' = 0$. Similarly if $i' = 0$ and $k = 1$ then, $S(x_{i,j,k}, y_{i',j',k'})$ is undefinable.
- $i = 1$ and $k' = 0$: $S(x_{i,j,k}, y_{i',j',k'})$ is undefinable. $i = 1$ means that there is a vertex in $\hat{G}$ to dominate x, so it is necessary to set $y \in D$ and nd $S(x_{1,j,k}, y_{i',j',k'})$ is definable if $k' = 1$. Similarly if $i' = 1$ and $k = 0$ then $S(x_{i,j,k}, y_{i',j',k'})$ is undefinable.
- $i = 2$: since there is exactly one vertex $y \in V(\hat{G})$ to dominate x, $S(x_{i,j,k}, y_{i',j',k'})$ is undefinable and similarly for $i' = 2$, $S(x_{i,j,k}, y_{i',j',k'})$ is undefinable.

6.2. **Sets for s-vertices of T.** Let v is a vertex of T is labeled by $(x,y)_s$. In this procedure, the set $S(x_{i,j,k}, y_{i',j',k'})$ will computed for a given vertex x, y and common vertex z. Assume the sets corresponding to $\tau_l(t)$ and $\tau_r(t)$ are $S_\ell(x_{i_\ell,j_\ell,k_\ell}, z_{i'_\ell,j'_\ell,k'_\ell})$ and $S_r(z_{i_r,j_r,k_r}, y_{i'_r,j'_r,k'_r})$ respectively.

Now for each $(i,j,k) \in M_t\}$ define:

$$S(x_{i,j,k}, y_{i',j',k'}) = Minsize_{(i'_\ell,j'_\ell,k'_\ell)\in M}\{s_\ell(x_{i,j,k}, z_{i'_\ell,j'_\ell,k'_\ell}) \cup s_r(z_{j'_\ell,i'_\ell,k'_\ell}, y_{i',j',k'})\} \tag{6.1}$$

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

$G_1 = (V_1, E_1, x, z)$, $G_2 = (V_2, E_2, z, y)$ and $\hat{G} = G_1 o_s G_2 = (\hat{V}, \hat{E}, x, y)$, to prove formula 6.1, we describe following cases.

- $z \in D$, if z is dominated by two vertices of $V(G_1) \setminus V(G_2)$, $V(G_2) \setminus V(G_1)$ or one of them of $V(G_1)$ and the other of $V(G_2)$ then, $(2,0,1), (0,2,1), (1,1,1) \in M_t$.
- $z \notin D$, if z is dominated by exactly one vertex of $V(G_1) \setminus V(G_2)$ then, $(1,0,0) \in M$ and if it is dominated by exactly one vertex of $V(G_2) \setminus V(G_1)$ then, $(0,1,0) \in M_t$.
- $z \in D$, if z is dominated by exactly one vertex of $V(G_1) \setminus V(G_2)$ then, $(1,0,1) \in M$ and if it is dominated by exactly one vertex of $V(G_2) \setminus V(G_1)$ then, $(0,1,1) \in M_t$.
- $z \notin D$, if z is dominated by two vertices of $V(G_1) \setminus V(G_2)$, $V(G_2) \setminus V(G_1)$ or one of them of $V(G_1)$ and the other of $V(G_2)$ then, $(2,0,0), (0,2,0), (1,1,0) \in M_t$.

6.3. **Sets for p-vertices of T.** Let v is a vertex of T is labeled by $(x,y)_p$, in this procedure, the set $S(x_{i,j}, y_{i',j'})$ will computed for a given vertex x, y. The sets corresponding to $\tau_l(t)$ and $\tau_r(t)$ are $S_\ell(x_{i_\ell,j_\ell}, y_{i'_\ell,j'_\ell})$ and $S_r(x_{i_r,j_r}, y_{i'_r,j'_r})$ respectively. For $(i,j,k) \in M_t$, we define,

$$S(x_{i,j,k}, y_{i',j',k'}) = Minsize\{s_\ell(x_{i_\ell,j_\ell,k}, y_{i'_\ell,j'_\ell,k'}) \cup s_r(x_{i_r,j_r,k}, y_{i'_r,j'_r,k'})\}. \tag{6.2}$$

Let $\tau_l(t)$, $\tau_r(t)$ and $\tau(t)$ represent p-graphs $G_1 = (V_1, E_1, x, y)$, $G_2 = (V_2, E_2, x, y)$ and $\hat{G} = G_1 o_p G_2 = (\hat{V}, \hat{E}, x, y)$ respectively. According to values of (i,j,k) (resp. (i',j',k')) proper values for (i_ℓ, j_ℓ, k_ℓ) (resp. $(i'_\ell, j'_\ell, k'_\ell)$) and (i_r, j_r, k_r) (resp. (i'_r, j'_r, k'_r)) will determine. Moreover, if $x \in D$ then in formula 6.2 $k = k_\ell = k_r = 1$ and if $x \notin D$, then $k = k_\ell = k_r = 0$.

- If x is dominated by one vertex of $V(G) \setminus V(\hat{G})$ then, in formula 6.2, $i = 0, j = 1$ and we have $i_\ell = i_r = 0$, $j_\ell = j_r = 1$. So that we define, We define:

$$S(x_{0,1,k}, y_{i',j',k'}) = Minsize\{s_\ell(x_{0,1,k}, y_{i'_\ell,j'_\ell,k'_\ell}) \cup s_r(x_{0,1,k}, y_{i'_r,j'_r,k'_r})\}$$

- If x is dominated by two vertices of $V(G) \setminus V(\hat{G})$ then, in formula 6.2, $i = 0, j = 2$ and we have $i_\ell = i_r = 0$, $j_\ell = j_r = 2$. So that we define, We define:

$$S(x_{0,2,k}, y_{i',j',k'}) = Minsize\{s_\ell(x_{0,2,k}, y_{i'_\ell,j'_\ell,k'_\ell}) \cup s_r(x_{0,2,k}, y_{i'_r,j'_r,k'_r})\}$$

- If x is dominated by one vertex of $w \in V(\hat{G})$ then, in formula 6.2, $i = 1$ and $j = 0$. Two following cases occur,
  1. $w \in V(G_1) \setminus V(G_2)$, so that $i_\ell = j_r = 1$ and $i_r = j_\ell = 0$,
  2. $w \in V(G_2) \setminus V(G_1)$, so that $i_\ell = j_r = 0$ and $i_r = j_\ell = 1$.

  We define:

$$\begin{aligned} S(x_{1,0,k}, y_{i',j',k'}) = Minsize \quad & \{s_\ell(x_{1,0,k}, y_{i'_\ell,j'_\ell,k'_\ell}) \cup s_r(x_{0,1,k}, y_{i'_r,j'_r,k'_r}), \\ & s_\ell(x_{0,1,k}, y_{i'_\ell,j'_\ell,k'_\ell}) \cup s_r(x_{1,0,k}, y_{i'_r,j'_r,k'_r})\}. \end{aligned}$$

- If x is dominated by two vertices of $v, w \in V(\hat{G})$ then, in formula 6.2, $i = 2$ and $j = 0$. Following cases occur,
  1. $v, w \in V(G_1) \setminus V(G_2)$, so that $i_\ell = j_r = 2$ and $i_r = j_\ell = 0$,
  2. $v, w \in V(G_2) \setminus V(G_1)$, so that $i_\ell = j_r = 0$ and $i_r = j_\ell = 2$,

2. $w \in V(G_1) \setminus V(G_2)$ and $v \in V(G_2) \setminus V(G_1)$, so that $i_\ell = j_\ell = i_r = j_r = 1$.

We define:

$$\begin{aligned} S(x_{2,0,k}, y_{i',j',k'}) = Minsize \quad & \{s_\ell(x_{2,0,k}, y_{i'_\ell,j'_\ell,k'_\ell}) \cup s_r(x_{0,2,k}, y_{i'_r,j'_r,k'_r}), \\ & s_\ell(x_{0,2,k}, y_{i'_\ell,j'_\ell,k'_\ell}) \cup s_r(x_{2,0,k}, y_{i'_r,j'_r,k'_r}), \\ & s_\ell(x_{1,1,k}, y_{i'_\ell,j'_\ell,k'_\ell}) \cup s_r(x_{1,1,k}, y_{i'_r,j'_r,k'_r})\}. \end{aligned}$$

- If x is dominated by one vertex of $v \in V(\hat{G})$ and one vertex of $w \in V(G) \setminus V(\hat{G})$ then, in formula 6.2, $i = 1$ and $j = 1$. Following cases occur,
  1. $v \in V(G_1)$, so that $i_\ell = j_\ell = 1$, $i_r = 0$ and $j_r = 2$,
  2. $v \in V(G_2))$, so that $i_\ell = 0$, $j_\ell = 2$ and $i_r = j_r = 1$.

$$\begin{aligned} S(x_{1,1,k}, y_{i',j',k'}) = Minsize \quad & \{s_\ell(x_{1,1,k}, y_{i'_\ell,j'_\ell,k'_\ell}) \cup s_r(x_{0,2,0}, y_{i'_r,j'_r,k'_r}), \\ & s_\ell(x_{0,2,0}, y_{i'_\ell,j'_\ell,k'_\ell}) \cup s_r(x_{1,1,0}, y_{i'_r,j'_r,k'_r})\}. \end{aligned}$$

6.4. **Sets for g-vertices of T.** Let v is a vertex of T is labeled by $(x, y)_g$. In this procedure, the set $S(x_{i,j}, y_{i',j'})$ will computed for a given vertex x, y.

Let the roots of $\tau_l(t)$ and $\tau_r(t)$ are labeled by (x, y) and (y, z) respectively, for some $z \in V$. The sets corresponding to $\tau_l(t)$ and $\tau_r(t)$ are $s_\ell(x_{i_\ell,j_\ell,k_\ell}, y_{i'_\ell,j'_\ell,k'_\ell})$ and $S_r(x_{i_r,j_r,k_r}, y_{i'_r,j'_r,k'_r})$ respectively. If a vertex $w \in V(\hat{G})$ (resp. $V(G) \setminus V(\hat{G})$) dominate x then, $w \in V(G_1)$ (resp. $V(G) \setminus V(G_1)$), so $i_\ell = i$ and $j_\ell = j$. In this kind of operation, number of vertices $V(G_1)$, $(V(G) \setminus V(G_1)$, $V(G_2)$ and $(V(G) \setminus V(G_2)$ make changes in $i', j'$and k'. So for each $(i, j, k) \in M_t$ define:

$$S(x_{i,j,k}, y_{i',j',k'}) = Minsize\{s_\ell(x_{i,j,k}, y_{i'_\ell,j'_\ell,k'}) \cup s_r(y_{i_r,j_r,k'}, z_{i'_r,j'_r,k'_r})\} \tag{6.3}$$

To find a relation between (i', j', k'), $(i'_\ell, j'_\ell, k'_\ell)$ and (i_r, j_r, k_r) for different value of (i'_r, j'_r, k'_r), let $\tau_l(t)$, $\tau_r(t)$ and $\tau(t)$ of T corresponding to p-graphs $G_1 = (V_1, E_1, x, y)$, $G_2 = (V_1, E_1, y, z)$ and $\hat{G} = G_1 o_g G_2 = (\hat{V}, \hat{E}, x, y)$. In formula 6.3, if $y \in D$, then $k' = k_\ell = k_r = 1$, if $y \in D$, then $' = k_\ell = k_r = 0$. According to the number of vertices $D \cap V(G_1)$ and $D \cup V(G_2)$ that dominate y, the following cases occur:

- y is dominated by only one vertex of $V(G) \setminus V(\hat{G})$, in the left side of formula 6.3, $i = 0$ and $j = 1$, so $i'_\ell = i_r = 0$ and $j'_\ell = j_r = 1$. We define:

$$S(x_{i,j,k}, y_{0,1,k'}) = Minsize\{s_\ell(x_{i,j,k}, y_{0,1,k'}) \cup s_r(y_{0,1,k'}, z_{i'_r,j'_r,k'_r})\}$$

- y is dominated by only two vertices of $V(G) \setminus V(\hat{G})$ then, in the left side of formula 6.3, $i = 0$ and $j = 2$, so $i'_\ell = i_r = 0$ and $j'_\ell = j_r = 2$. We define:

$$S(x_{i,j,k}, y_{0,2,k'}) = Minsize\{s_\ell(x_{i,j,k}, y_{0,2,k'}) \cup s_r(y_{0,2,k'}, z_{i'_r,j'_r,k'_r})\}$$

- y is dominated by exactly one vertex of $V(\hat{G})$, in formula 6.3, if $i = 1$ and $j = 0$. So two cases occur
  1. $i'_\ell = j_r = 0$ and $i_r = j'_\ell = 1$,
  2. $i'_\ell = j_r = 1$ and $i_r = j'_\ell = 0$.

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

We define:

$$S(x_{i,j,k}, y_{1,0',k'}) = Minsize \quad \{s_\ell(x_{i,j,k}, y_{1,0,k'}) \cup s_r(y_{0,1,k'}, z_{i'_r,j'_r,k'_r}), \\ s_\ell(x_{i_\ell,j_\ell,k_\ell}, y_{1,0,k'}) \cup s_r(z_{0,1,k'}, y_{i'_r,j'_r,k'_r})\}.$$

- y is dominated by only two vertices of $V(\hat{G})$, formula 6.3, $i = 2$ and $j = 0$ so, one of the following cases occur
  1. $i'_\ell = j_r = 0$ and $i_r = j'_\ell = 2$,
  2. $i'_\ell = j_r = 2$ and $i_r = j'_\ell = 0$,
  3. $i'_\ell = j_r = 1$ and $i_r = j'_\ell = 1$.

  We define:

$$S(x_{i,j,k}, y_{2,0,k'}) = Minsize \quad \{s_\ell(x_{i,j,k}, y_{2,0,k'}) \cup s_r(y_{0,2,k'}, z_{i'_r,j'_r,k'_r}), \\ s_\ell(x_{i,j,k}, y_{0,2,k'}) \cup s_r(y_{2,0,k'}, z_{i'_r,j'_r,k'_r}), \\ s_\ell(x_{i,j,k}, y_{1,1,k'}) \cup s_r(y_{1,1,k'}, z_{i'_r,j'_r,k'_r})\}.$$

- y is dominated by exactly one vertex of $V(\hat{G})$ and one vertex of $V(G) \setminus V(\hat{G})$, in formula 6.3 $i = 1$ and $j = 1$ so, two cases occur
  1. $i'_\ell = j'_\ell = 1$, $i_r = 0$ and $j'_r = 1$,
  2. $i'_\ell = 0$, $j'_\ell = 1$ and $i_r = j_r = 1$.

  We define:

$$S(x_{i,j,k}, y_{1,1,k'}) = Minsize \quad \{s_\ell(x_{i,j,k}, y_{1,1,k'}) \cup s_r(y_{0,2,k'}, z_{i'_r,j'_r,k'_r}), \\ s_\ell(x_{i,j,k}, y_{0,2,k'}) \cup s_r(y_{1,1,k'}, z_{i'_r,j'_r,k'_r})\}.$$

Similar to procedure internal-set of algorithm $[1,2]$-MINSET, in each formula of methods s-vertex, p-vertex and g-vertex, we delete any undefinable subformula of the right side of formula computing $S_i(x_{i,j,k}, y_{i',j',k'})$. Furthermore if all subformulas are deleted from the right hand side of some formula, then $S(x_{i,j,k}, y_{i',j',k'})$ will be undefinable.

Algorithm total $[1,2]$-MINSET is similar to algorithm $[1,2]$-MINSET when we use procedures ProcessLeaf-t, ProcessSvertex-t, ProcessPvertex-t and ProcessGvertex-t.

Theorem 6.1. *For a given GSP graph $G = (V, E)$, algorithm total $[1,2]$-MINSET find a minimum $[1,2]$-set in time $O(|V|)$.*

7. Concluding remarks

In this paper, we initiate the study of the $[1,2]$-set and total $[1,2]$-set problems for generalized series-paralle graphs. We have also provided exact polynomial time algorithms for these classes of graphs.

Acknowledgments

The authors are grateful to Dr. A. Shakiba and A. K. Goharshady for their constructive comments and suggestions on improving of our paper.

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

Algorithm 2 :Finding a $\gamma_{t[1,2]}$-sets of a GSP

find a parse tree of G like T
$M_t \leftarrow \{(0,1,0),(0,1,1),(1,0,0),(1,0,1),(0,2,0),(0,2,1),(2,0,0),(2,0,1),(1,1,0),(1,1,1)\}$.
traversing T in a bottom up order and visit vertex v
switch *type of* v **do**
    case $Leaf$
        ProcessLeaf(x,y) ▷ $(x,y)_i$ is the label of v
    case $s-vertex$
        ProcessSvertex(x,z,y) ▷ $(x,y)_s$, (x,z) and (z,y) are label of v, left and right child of v respectively.
    case $p-vertex$
        ProcessPvertex(v) ▷ $(x,y)_p$ is the label of v and the label of left and right child of v is (x,y).
    case $g-vertex$
        ProcessGvertex(x,y,z) ▷ $(x,y)_s$, (x,y) and (y,z) are label of v, left and right child of v respectively.
$D \leftarrow \emptyset$ ▷ (x,y) is the label of root T
for all $(i,j,k),(i',j',k') \in \{(1,0,0),(1,0,1),(2,0,0),(2,0,1)\}$ **do**
    Add $S(x_{i,j,k}, y_{i',j',k'})$ to D
end for
$\gamma_{[1,2]}(G) \leftarrow Minsize(D)$

References

[1] A. V. Aho and J. E. Hopcroft, *Design & Analysis of Computer Algorithms*, Pearson Education India, 1974.

[2] S. C. Chang, J. J. Liu and Y. L. Wang, The weighted independent domination problem in series-parallel graphs, *Intelligent Systems and Applications*, **274** (2015) 77–84.

[3] P. Chebolu, M. Cryan and R. Martin, Exact counting of euler tours for generalized series-parallel graphs, *J. Discrete Algorithms*, **10** (2012) 110–122.

[4] M. Chellali, T. W. Haynes, S. T. Hedetniemi and A. McRae, [1,2]-sets in graphs, *Discrete Appl Math.*, **161** (2013) 2885–2893.

[5] E. J. Cockayne and S. T. Hedetniemi, Towards a theory of domination in graphs, *Networks*, **7** (1977) 247–261.

[6] C. J. Colbourn, P. J. Slater and L. K. Stewart, Locating dominating sets in series parallel networks, *Congr. Numer*, **56** (1987) 135–162.

[7] O. Etesami, N. Ghareghani, M. Habib, M.R. Hooshmandasl, R. Naserasr and P. Sharifani, When an optimal dominating set with given constraints exists, *Theor. Comput. Sci.*, **780** (2019) 54–65.

[8] M. R. Garey, D. S. Johnson and L. Stockmeyer, Some simplified np-complete graph problems, *Theor. Comput. Sci.*, **1** (1976) 237–267.

[9] A. Goharshadi and M. R. Hooshmandasl, M. Alambardar Meybodi, [1,2]-sets and [1,2]-total sets in trees with algorithms, *Discrete Appl. Math.*, **198** (2016) 136–146.

[10] S. Guha and S. Khuller, Approximation algorithms for connected dominating sets, *Algorithmica*, **20** (1998) 374–387.

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509

[11] S. Guha and S. Khuller, Improved methods for approximating node weighted steiner trees and connected dominating sets, *Inform Comput*, **150** (1999) 57–74.

[12] E. O. Hare, S. T. Hedetniemi, R. C. Laskar, K. Peters and T. Wimer, Linear-time computability of combinatorial problems on generalized-series-parallel graphs, *Discrete Algorithms and Complexity*, (1987) 437–457.

[13] S. T. Hedetniemi, R. Laskar and J. Pfaff, A linear algorithm for finding a minimum dominating set in a cactus, *Discrete Appl Math.*, **13** (1986) 287–292.

[14] J. E. Hopcroft and R. E. Tarjan, Dividing a graph into triconnected components, *SIAM J. Comput.*, **2** (1973) 135–158.

[15] T. Kikuno, N. Yoshida and Y. Kakuda, A linear algorithm for the domination number of a series-parallel graph. *Discrete Appl. Math.*, **5** (1983) 299–311.

[16] M. B. Richey and R. G. Parker, Minimum-maximal matching in series-parallel graphs, *European J. Oper. Res.*, **33** (1988) 98–105.

[17] K. Takamizawa, T. Nishizeki and N. Saito, Linear-time computability of combinatorial problems on series-parallel graphs, *J. Assoc. Comput. Mach.*, **29** (1982) 623–641.

[18] D. B. West, *Introduction to graph theory*, **2**, Prentice hall Upper Saddle River, 2001.

[19] X. Yang and B. Wu, [1,2]-domination in graphs, *Discrete Appl. Math.*, **175** (2014) 79–86.

[20] C.-C. Yen and R. Lee, A linear time algorithm to solve the weighted perfect domination problem in series-parallel graphs, *European J. Oper. Res.*, **73** (1994) 192–198.

[21] F. Zou, Y. Wang, X. H. Xu, X. Li, H. Du, P. Wan and W. Wu, New approximations for minimum-weighted dominating sets and minimum-weighted connected dominating sets on unit disk graphs, *Theor. Comput. Sci.*, **412** (2011) 198–208.

Pouyeh Sharifani

Department of Computer Science, Yazd University, P.O.Box 98195-741, Yazd, Iran

Email: pouyeh.sharifani@gmail.com

Mohammad Reza Hooshmandasl

Department of Computer Science, Yazd University, P.O.Box 98195-741, Yazd, Iran

and

Department of Computer Science, University of Mohaghegh Ardabili, P.O.Box 56199-11367, Ardabil, Iran

Email: hooshmandasl@yazd.ac.ir

DOI: http://dx.doi.org/10.22108/toc.2019.105482.1509