Dynamically maintaining minimal integral separator for Threshold and Difference Graphs *

Tiziana Calamoneri, Angelo Monti, and Rossella Petreschi

Computer Science Department,
“Sapienza” University of Rome, Italy
calamo/monti/petreschi@di.uniroma1.it

Abstract. This paper deals with the well known classes of threshold and difference graphs, both characterized by separators, i.e node weight functions and thresholds. The interest is in maintaining minimum the value of the separator when graph operations are dynamically done on these classes of graphs. Results are presented for operations like addition/deletion of either an edge or a node. Moreover, the disjoint union and the join of two threshold graphs are considered.

Keywords: threshold graphs, difference graphs, chain graphs, threshold signed graphs, graph operations.

1 Introduction

Threshold graphs constitute a very important and well studied graph class, since they find applications in several fields, such as psychology, parallel processing, scheduling, and graph theory. For this reason, threshold graphs have been defined many times in the literature (see, e.g. [4, 6]), and have been widely studied. Difference graphs (also known as chain graphs) – that are strictly related to threshold graphs, though incomparable – had similar destiny and have been independently introduced (see, e.g. [5, 9]). For a comprehensive survey on threshold graphs, difference graphs and related topics, see [7].

Among all the numerous equivalent definitions of threshold and difference graphs, many of them exploit a node weight function and a threshold. Orlin [8] presented an algorithm for minimizing the threshold and weights w.r.t. one of these definitions in linear time in the number of nodes. In this paper we consider a different (equivalent) definition based on a threshold and a node weight function and for it we find a minimum separator in linear time w.r.t. the number of different degrees of the graph. Moreover we prove that it is possible to directly modify the node weight function and the threshold for defining a new graph obtained by dynamically operating on these classes of graphs. In many cases we maintain the minimality of the separator. We consider the operations of addition/deletion of either an edge or a node with all its incident edges. Moreover, we study the binary operations disjoint union and join of two threshold graphs.

* Partially supported by the Italian Ministry of Education and University, PRIN project ”AMANDA: Algorithmics for MAssive and Networked DAta”
2 Preliminaries

In this section, we list some definitions and properties from [7]. For the sake of clarity, we reorganized them in order to optimize the presentation.

**Definition 1.** A graph \( G = (V, E) \) is a **threshold graph** if there is a mapping \( a : V \to \mathbb{R}^+ \) and a positive real number \( S \) such that

\[
a(v) < S \text{ for all } v \in V \tag{1}
\]

\[
(v, w) \in E \text{ if and only if } a(v) + a(w) \geq S \tag{2}
\]

The pair \((a, S)\) will be called **separator** for graph \( G \).

In Figure 1.a a threshold graph with one of its separators is depicted.

**Definition 2.** A graph \( G = (V, E) \) is a **difference graph** if there is mapping \( a : V \to \mathbb{R} \) and a positive real number \( T \) such that

\[
|a(v)| < T \text{ for all } v \in V \tag{3}
\]

\[
(v, w) \in E \text{ if and only if } |a(v) - a(w)| \geq T \tag{4}
\]

The pair \((a, T)\) will be called **separator** for graph \( G \).

The node set of a difference graph \( G \) can be partitioned as \( V = U \cup W \), where \( U = \{v \in V : a(v) \geq 0\} \) and \( W = \{v \in V : a(v) < 0\} \); both \( U \) and \( W \) induce a stable set and hence \( G \) is bipartite with bipartition \((U, W)\). A difference graph with one of its separators is shown in Figure 1.b.

Although Definitions 1 and 2 are very similar, the two defined classes are incomparable. Nevertheless they are strictly related, as shown by the following theorem:

**Theorem 1.** A bipartite graph \( G = (U \cup W, E) \) is a difference graph if and only if adding to \( G \) all possible edges with both ends in \( U \) yields a threshold graph.
Definition 3. Let $G = (V, E)$ be a graph whose distinct node-degrees are $\delta_1 < \ldots < \delta_m$, and let $\delta_0 = 0$ (even if no node of degree 0 exists). Let $D_i = \{v \in V \text{ s.t. } \deg(v) = \delta_i\}$ for $i = 0, \ldots, m$; $D_i$ is called $i$-th box; the sequence $D_0, \ldots, D_m$ is called the degree partition of $G$.

Definition 4. Let $G = (U \cup V, E)$ be a bipartite graph whose distinct node-degrees for the nodes in the partition $X \in \{U, W\}$ are $\delta_1^X < \ldots < \delta_t^X$ and let $\delta_0^X = 0$ (even if no node of degree 0 exists in the partition $X$). Let $D_i^X = \{v \in X \text{ s.t. } \deg(v) = \delta_i^X\}$ for $0 \leq i \leq t_X$; the sequence $D_0^U, \ldots, D_{t_U}^U, D_0^W, \ldots, D_{t_W}^W$ is called the bipartite degree partition of $G$.

Lemma 1. Let $G = (V, E)$ be a threshold graph with degree partition $D_0, \ldots, D_m$ and whose node set is partitioned into a clique $K$ and a stable set $I$; let $x \in D_i$ and $y \in D_j$ be two distinct nodes.

1. $D_0 \cup \ldots \cup D_{(m/2)} = I$ and $D_{(m/2)+1} \cup \ldots \cup D_m = K$;
2. If $e = (x, y) \notin E$, the graph $G' = (V, E \cup \{e\})$ is a threshold graph if and only if $i + j = m$;
3. If $e = (x, y) \in E$, the graph $G' = (V, E \setminus \{e\})$ is a threshold graph if and only if $i + j = m + 1$;
4. $e = (x, y) \in E$ if and only if $i + j \geq m + 1$.

Lemma 2. Let $G = (U \cup W, E)$ be a difference graph with degree partition $D_0^U, \ldots, D_{t_U}^U, D_0^W, \ldots, D_{t_W}^W$, and let $x \in D_i^U$ and $y \in D_j^W$ be two distinct nodes.

1. $t_U = t_W = t$;
2. If $e = (x, y) \notin E$, the graph $G' = (U \cup W, E \cup \{e\})$ is a difference graph if and only if $i + j = t$;
3. If $e = (x, y) \in E$, the graph $G' = (U \cup W, E \setminus \{e\})$ is a difference graph if and only if $i + j = t + 1$;
4. $e = (x, y) \in E$ if and only if $i + j \geq t + 1$.

Given a graph $G$, we recall that the neighborhood $N(v)$ of a node $v$ is the set of all neighbors of $v$, and its closed neighborhood is $N[v] = N(v) \cup \{v\}$.

Lemma 3. Given a graph $G = (V, E)$, the vicinal preorder is a binary relation on the nodes of $V$ such that $u \geq v \iff N[u] \supseteq N(v)$. The vicinal preorder is total on $V$ if $G$ is a threshold graph $G$ and on $U$ and $V$ if $G$ is a difference graph.

We now introduce a superclass of both threshold and difference graphs.

Definition 5. A graph $G = (V, E)$ is a threshold signed graph if there is a mapping $a : V \to \mathbb{R}$ and two positive real numbers $S$ and $T$ such that

\[ |a(v)| < \min\{S, T\} \quad (5) \]

and

\[ (v, w) \in E \iff \text{either } |a(v) + a(w)| \geq S \text{ or } |a(v) - a(w)| \geq T. \quad (6) \]

The triple $(a, S, T)$ will be called separator for graph $G$. 

Given a graph $G = (V, E)$, we denote by $\deg(v)$ the degree of node $v$. 

Given a graph $G = (V, E)$, we denote by $\deg(v)$ the degree of node $v$. 

"
Consider \( X = \{ x \in V \text{ s.t. } a(x) < 0 \} \) and \( Y = \{ x \in V \text{ s.t. } a(x) \geq 0 \} \).

As highlighted in Figure 1.c, we can see a threshold signed graph as constituted by two threshold graphs, \( G^- \) and \( G^+ \) respectively induced by \( X \) and \( Y \), that are connected by a difference graph \( D \). Notice that for \( X \) we consider the opposite of the \( a \)'s values.

### 3 A data structure for computing minimal integral separator for threshold or difference graphs

Although in Definition 1 a threshold graph \( G = (V,E) \) is a graph having a separator with non-negative real values, it is common to equivalently require the separator to have non-negative integral values (i.e. an integral separator) [7]. We say that an integral separator \((a,S)\) for \( G \) is minimum if for any other integral separator \((a',S')\) for \( G \) we have \( S \leq S' \). In the following theorem we show that the value of \( S \) of a minimum integral separator \((a,S)\) of \( G \) is given by the cardinality of the degree partition of \( G \) plus 1.

**Theorem 2.** Let \( G = (V,E) \) be a threshold graph with degree partition \( D_0, \ldots, D_m \). The pair \((a,S)\), where \( S = m + 1 \) and for each node \( v \in V \), \( a(v) = i \) if \( v \in D_i \), is a minimal integral separator of \( G \).

**Proof.** First of all, we prove that \((a,S)\) is a separator, i.e. that satisfies the two inequalities of Definition 1. Note that for each \( v \in V \) it holds \( 0 \leq a(v) \leq m < S \) thus the pair \((a,S)\) satisfies Inequality 1. Moreover, Inequality 2 follows from Item 4 in Lemma 1. Trivially, \((a,S)\) is integral.

Let us now prove that \((a,S)\) is minimal. By contradiction, let \((a,S)\) be not minimal, and let \((a',S')\) be an integral separator for \( G \) such that \( S' < S \). Observe that only isolated nodes can have weight equal to zero (indeed, if \((u,v) \in E \) and \( a'(u) = 0 \) then \( a'(u) + a'(v) = a'(v) < S' \) from Inequality 1, but this contradicts Inequality 2). Moreover, notice that two nodes \( u \) and \( v \) having the same weight necessarily behave in the same way (i.e. for any other node \( w \in V \), it holds that \((u,w) \in E \) if and only if \((v,w) \in E \)), so nodes having different degree cannot have the same weight. All this implies that the function \( a' \) on the non isolated nodes assumes at least \( m \) different strictly positive weights. Thus Inequality 1 implies that \( S' \geq m+1 \). The chain \( m+1 = S > S' \geq m+1 \) proves the minimality of \((a,S)\). \( \square \)

From the same reasonings, we deduce the following theorem for a difference graph \( G \).

**Theorem 3.** Let \( G = (U \cup W, E) \) be a difference graph with bipartite degree partition \( D_U^1, \ldots, D_U^{t}, D_W^1, \ldots, D_W^{t} \). The pair \((a,S)\), where \( S = 2t + 1 \) and, for each node \( v \in U \), \( a(v) = -i \) if \( v \in D_U^i \) and \( a(v) = i \) if \( v \in D_W^i \), is a minimal integral separator of \( G \).

Now we present two data structures for representing threshold and difference graphs allowing us to compute in a natural way minimal integral separators for
these graph classes, according to Theorems 2 and 3. Since a threshold graph $G = (V, E)$ is univocally determined by its degree partition $D_0, \ldots, D_m$ [2]; we may store $G$ using two arrays $\delta[0..m]$ and $\mu[0..m]$, where $\delta[i]$ represents the degree $\delta_i$ of the nodes in the $i$-th box $D_i$, and $\mu[i]$ represents its cardinality, $|D_i|$. Similarly, let $G = (U \cup W, E)$ be a difference graph with bipartite degree partition $D_U^1, \ldots, D_U^t, D_W^1, \ldots, D_W^t$. This partition univocally determines $G$ and $G$ may be represented using the arrays $\delta_X[0..t]$ and $\mu_X[0..t]$, where $\delta_X[i]$ represents the degree of the nodes in the $i$-th box $D_X^i$, and $\mu_X[i]$ represents its cardinality $|D_X^i|$, $X \in \{U, W\}$. If $G = (V, E)$ is a threshold graph, $i$ is the weight associated to all the $\mu[i]$ nodes belonging to the box $D_i$ of degree $\delta_i$. Similarly, If $G = (U \cup W, E)$ is a difference graph, $i$ is the weight associated to all the $\mu_X[i]$ nodes belonging to box $D_X^i$ of degree $\delta_X[i]$, with $X \in \{U, W\}$.

Exploiting these two data structures, the following theorem holds:

**Theorem 4.** Given a threshold (difference) graph $G$ by means of its (bipartite) degree partition, its minimal integral separator can be found in time linear w.r. t. the number of different degrees in $G$.

## 4 Adding/deleting an edge to threshold/difference graphs

In this section we study how to get a new graph, obtained by adding/deleting an edge from a graph that is either a threshold or a difference graph. In order to make easier the exposition, preliminarily we consider two functions, operating on the data structures introduced in Section 3.

By $\text{IncreaseDeg}(\delta, \mu, i, \text{dim})$ we denote the operation of updating arrays $\delta[0..\text{dim}]$ and $\mu[0..\text{dim}]$ when the degree of a node in box $D_i$, $0 \leq i \leq \text{dim}$, is increased by one. $\text{IncreaseDeg}$ can have as consequence the appearance of a new box (if $i = \text{dim}$ or if the degree of the nodes in $D_{i+1}$ is different from the degree of nodes in $D_i$ plus one). On the other hand, this increment can also have as consequence the disappearance of box $D_i$ (if $i \neq 0$, $|D_i| = 1$ and the degree of the nodes in $D_{i+1}$ is equal to the degree of the nodes in $D_i$ plus one). Symmetrically, we may consider the operation $\text{DecreaseDeg}(\delta, \mu, i, \text{dim})$ of updating arrays $\delta[0..\text{dim}]$ and $\mu[0..\text{dim}]$ when the degree of a node in box $D_i$, $0 < i \leq \text{dim}$, is decreased by one. In the Appendix the details of these functions are reported.

Let us now consider a threshold graph $G$. Let $(x, y)$, $x \in D_i$ and $y \in D_j$, the edge to add/delete to/from $G$. Items 2 and 3 of Lemma 1 give a characterization of the indices $i$ and $j$ to ensure that the modified graph is still a threshold graph; namely, $i + j = m$ in case of insertion, and $i + j = m + 1$ in case of deletion.

We present two operations, $\text{InsertEdge}(\delta, \mu, i, m)$ and $\text{DeleteEdge}(\delta, \mu, i, m)$, that update the data structure when an edge is added between a node in box $D_i$ and a node in box $D_{m-i}$ and when an edge is deleted between a node in box $D_i$ and a node in box $D_{m+1-i}$, respectively. Observe that with the insertion of an edge, the degrees of its endpoints are increased by one. Thus we can call twice subroutine $\text{IncreaseDeg}$, once on a node in box $D_i$ and once on a node in
Fig. 2. Examples proving that all 5 cases in algorithm InsertEdge are possible. Grey edges represent the edges that are going to be added. (In order to consider DeleteEdge, figures must be read from right to left.)

Since after the execution of IncreaseDeg (DecreaseDeg), $m$ may potentially vary from $m$ to $m \pm 1$, with the execution of InsertEdge (DeleteEdge) the number of boxes can potentially vary from $m$ to $m \pm 2$. In Figure 2 we show that all the five possibilities may actually occur.

Assume now that $G$ is a difference graph. The algorithms for adding/eliminating an edge in $G$ are based on the same idea presented for the algorithms on
threshold graphs, but they are even simpler because the data structure used for representing these graphs keeps separated the bipartition (and so, adding a new box after the first call of \texttt{IncreaseDeg} does not affect the index of the other endpoint). Notice that Item 1 of Lemma 2 ensures that the number of boxes in the two classes is the same $t$. So, for difference graphs, $t$ can either remain unaltered or to change to $t \pm 1$ and all the three possibilities may actually occur. The two algorithms for inserting and deleting an edge in a difference graph follow:

\begin{verbatim}
D - InsertEdge($\delta_U, \mu_U, \delta_W, \mu_W, i, t$)  D - DeleteEdge($\delta_U, \mu_U, \delta_W, \mu_W, i, t$)
  $j \leftarrow t - i$;  $j \leftarrow t + 1 - i$;
  IncreaseDeg($\delta_U, \mu_U, i, t$);  DecreaseDeg($\delta_U, \mu_U, i, t$);
  IncreaseDeg($\delta_W, \mu_W, j, t$);  DecreaseDeg($\delta_W, \mu_W, j, t$);
\end{verbatim}

\textbf{Note.} All the algorithms described in this section are correct and maintain the minimality of the integral separators.

5 Adding/deleting a node to threshold/difference graphs

In this section we study how to get a new graph by adding/deleting a node from a graph that is either a threshold or a difference graph. In order to make easier the exposition, preliminarily we consider four functions, operating on the data structures introduced in Section 3.

By \texttt{+Node($\delta, \mu, d, \text{dim}$)} we denote the operation of giving space to a new node of degree $d$ either in a threshold graph or in a partition of a difference graph, without caring about the update of its neighbors (that will be done with another subroutine). This subroutine looks for the box where the new node must be inserted: if there exists a box $D_i$ with degree $d$, $\mu_i$ is simply increased by one; otherwise a new box for the new node is created.

By \texttt{IncreaseDegreeOfSetNode($\delta, \mu, d, \text{dim}$)} we denote the operation of augmenting by one the degree of the $d$ nodes of highest degree either in a threshold graph or in a partition of a difference graph. This subroutine increases by one the degree of all the boxes $D_i$ s.t. $d - \sum_{s=j+1}^{m} |D_s| \geq 0$, while nodes of boxes $D_1, \ldots, D_{j-1}$ remain unchanged. For what concerns $D_j$, it is in general split into two boxes (precisely $d - \sum_{s=j+1}^{m} |D_s|$ nodes leave $D_j$ to form a new box with degree augmented by one). We can define even the symmetric functions: by \texttt{-Node($\delta, \mu, i, \text{dim}$)} we denote the operation eliminating from the data structure storing either a threshold or a difference graph a node in box $D_i$, $0 \leq i \leq \text{dim}$, regardless of its neighbors (whose degree will be updated with another subroutine). By \texttt{DecreaseDegreeOfSetNode($\delta, \mu, d, \text{dim}$)} we denote the operation of decreasing by one the degree of the $d$ nodes of highest degree either in a threshold graph or in a partition of a difference graph. In the Appendix the details of these functions are reported.

Let now $G = (V, E)$ be a threshold graph. Adding a new node of degree $d$ to $G$ yields a threshold graph if and only if the $d$ neighbors of the new node are the $d$ nodes with highest degrees (this can be easily deduced from Item 4
of Lemma 1). So, we can call IncreaseDegreeOfSetNode and observe that, after its execution, $m$ could be increased by one. Then, we have to update the data structure by inserting the new node by means of $\text{+Node}$, and even in this case $m$ could be increased by one. So, the number of the different degrees can potentially vary from $m$ to $m + 2$. Figure 3 shows that all three possibilities can occur.

By $\text{InsertNode}(\delta, \mu, d, m)$ we denote the operation of updating the data structure storing threshold graph $G$ when a node of degree $d$ is added to the graph. The previous reasonings can be repeated when $G$ is a difference graph (assuming, w.l.o.g., that the new node is inserted in partition $U$), so giving rise to $\text{D-InsertNode}(\delta_U, \mu_U, \delta_W, \mu_W, i, t)$, that is the operation of updating the data structure when a node of degree $d$ is added to the difference graph.

$\text{InsertNode}(\delta, \mu, d, m)$ \quad $\text{D-InsertNode}(\delta_U, \mu_U, \delta_W, \mu_W, d, t)$
$\text{+Node}(\delta, \mu, d, m)$ \quad $\text{+Node}(\delta_U, \mu_U, d, t)$
$\text{IncreaseDegreeOfSetNode}(\delta, \mu, d, m)$ \quad $\text{IncreaseDegreeOfSetNode}(\delta_W, \mu_W, d, t)$

Now we consider the problem of deleting nodes to a threshold graph. Any node-induced subgraph $G'$ of $G$ is a threshold graph (indeed for the graph $G'$ use the mapping $a$ restricted to the nodes of $G'$ and the same value $S$). Thus the class of threshold graphs is closed under the deletion of an arbitrary node. Given a threshold graph, by $\text{DeleteNode}(\delta, \mu, i, m)$ we denote the operation of updating the data structure when a node is deleted from box $D_i$, $0 \leq i \leq m$. This deletion is performed by $\text{-Node}$ that can have as consequence the disappearance of box $D_i$ (if $i \neq 0$ and $|D_i| = 1$). Thus $m$ can decrease by one. Moreover, the $\delta[i]$ nodes with highest degree must have their degrees decreased by one. These nodes belong to boxes $D_m, \ldots, D_{m+1-i}$. It can occur that the degree of nodes in box $m + 1 - i$ becomes equal to the degree of the nodes in box $m - i$ and, in this case, the two boxes merge and the number of boxes further decrease by one.
Hence after deleting a node, the number of boxes in the degree partition can potentially vary from \( m \) to \( m - 2 \) and all cases can occur, as shown in Figure 3.

Analogous reasonings can be done when \( G \) is a difference graph, and define \( \text{D-DeleteNode}(\delta_U, \mu_U, \delta_W, \mu_W, i, t) \) as the operation of updating the data structure when a node is deleted (assuming w.l.o.g. that the new node is deleted from partition \( U \)).

\[
\begin{align*}
\text{DeleteNode}(\delta, \mu, i, m) & \quad \text{D - deleteNode}(\delta_U, \mu_U, \delta_W, \mu_W, i, t) \\
\quad d & \leftarrow \delta[i] \\
\quad \text{Node}(\delta, \mu, i, m) & \quad \text{Node}(\delta_U, \mu_U, i, t) \\
\text{DecreaseDegreeOfSetNode}(\delta, \mu, d, m) & \quad \text{DecreaseDegreeOfSetNode}(\delta_W, \mu_W, d, t)
\end{align*}
\]

Note. All the algorithms described in this section are correct and maintain the minimality of the integral separators.

6 Disjoint union and join of two threshold graphs

Given two graphs with disjoint node sets \( G_1 = (V_1, E_1) \) and \( G_2 = (V_2, E_2) \), their disjoint union is the graph \( G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2) \); their join is the graph \( G_1 + G_2 \) obtained adding to their disjoint union all the edges that connect the nodes of the first graph with the nodes of the second graph. We observe that, if \( G_1 \) and \( G_2 \) are both threshold graphs, \( G_1 \cup G_2 \) and \( G_1 + G_2 \) are threshold signed graphs, where the difference graph connecting the two threshold graphs is either the null graph (in \( G_1 \cup G_2 \)) or the complete bipartite graph (in \( G_1 + G_2 \)).

Let us now introduce a data structure for representing a threshold signed graph. Given a threshold signed graph \( G = (X \cup Y, E) \), two nodes \( u \) and \( w \) are false twins if they have the same neighborhood, i.e. \( N(u) = N(w) \); they are true twins if they have the same closed neighborhood, i.e. \( N[u] = N[w] \). We say that \( u \) and \( w \) are simply twins if they are either true or false twins and they belong to the same set, \( X \) or \( Y \). Let us consider the partition of the node set into equivalence classes, \( B_1, \ldots, B_{\Delta} \), induced by the relation of being twins. Even though there is not a tie between the degree partition of a threshold signed graph and its structure, as in the case of threshold and difference graphs, it is possible to extend the reasonings done in the proof of Theorem 2 to this class of graphs. Indeed, if \( v \) is an isolated node it is not restrictive to assume \( a(v) = 0 \) and, obviously, if \( a(v) = 0 \) then \( v \) is an isolated node. Moreover, it is easy to see that two nodes having the same value of \( a \) are necessarily twins. From the other hand, if there are two twins \( u \) and \( w \) having \( a(u) \neq a(w) \) (w.l.o.g. let \( a(u) < a(w) \)), we can easily modify function \( a \) in order to assign them the same value (that is \( a(w) \) if \( u \) and \( w \) are connected and \( a(u) \) otherwise). So, from now on, we consider only node weight functions assigning value 0 to each isolated node and the same value to each set of twins.

As consequence of all these reasonings, we may store a threshold signed graphs by means of two arrays \( \alpha[0..\Delta] \) and \( \mu[0..\Delta] \): in \( \alpha[i] \) there is the value of the weight assigned to the \( \mu[i] \) nodes of \( B_i \), \( 0 \leq i \leq \Delta \); if there are no isolated nodes \( \alpha[0] \) and \( \mu[0] \) are set to 0. Variables \( S \) and \( T \) store the two thresholds.
Let us now go back to consider the operations of disjoint union and join of two threshold graphs \( G_1 \) and \( G_2 \) stored as \( \delta_1[0..m_1] \), \( \mu_1[0..m_1] \) and \( \delta_2[0..m_2] \), \( \mu_2[0..m_2] \), respectively. Assume first that \( G_1 \) and \( G_2 \) have the same threshold \( S \) (i.e. \( m_1 = m_2 = m \)). Informally, the array \( \alpha[1..\Delta] \) of both \( G_1 \cup G_2 \) and \( G_1 + G_2 \) is obtained by opportune transcribing the values of the node weight function of the single threshold graphs (deduced through Theorem 2), the array \( \mu[1..\Delta] \) is obtained by copying the values of \( \mu_1 \) and \( \mu_2 \), while threshold \( S \) is kept unaltered. For what concerns threshold \( T \), in the case of \( G_1 \cup G_2 \) it is set to a sufficiently large value in order to guarantee that no edges are in the difference subgraph, in the case of \( G_1 + G_2 \) it is set to a sufficiently small value in order to guarantee that the difference subgraph is a complete bipartite graph. In this latter case, \( T \) assumes a too small value, contradicting Property 5 of Definition 5, so we need to modify the values of the node weight function and of the two thresholds in order to restore the property.

The following lemmas formalize the operations of disjoint union and join of two threshold graphs:

**Lemma 4.** Let be given two threshold graphs \( G_1 \) and \( G_2 \) by means of \( \delta_1[0..m] \), \( \mu_1[0..m] \) and \( \delta_2[0..m] \), \( \mu_2[0..m] \) and let \( S = m + 1 \) be their common threshold. The following function determines the threshold signed graph \( G_1 \cup G_2 \).

\[
\text{DisjointUnion}(\delta_1, \mu_1, \delta_2, \mu_2, m)
\begin{align*}
\mu[0] &\leftarrow \mu_1[0] + \mu_2[0]; \alpha[0] \leftarrow 0; \\
\text{FOR } i = 1 \text{ TO } m \text{ DO} \\
& \alpha[i] \leftarrow -i; \mu[i] \leftarrow \mu_1[i]; \\
& \text{FOR } i = 1 \text{ TO } m \text{ DO} \\
& \alpha[m+i] \leftarrow i; \mu[m+i] \leftarrow \mu_2[i]; \\
\Delta &\leftarrow 2m; \\
S &\leftarrow m + 1; \\
T &\leftarrow 2m + 1; \\
\text{RETURN } (\alpha, \mu, S, T, \Delta).
\end{align*}
\]

*Proof.* Both \( S \) and \( T \) determined by function \( \text{DisjointUnion} \) are greater than the modulo of each \( \alpha[i], i = 0, \ldots, \Delta \) as far as they are defined.

Moreover, the two threshold subgraphs \( G^- \) and \( G^+ \) of \( G_1 \cup G_2 \) are exactly the same as \( G_1 \) and \( G_2 \), respectively. Finally, no edge can satisfy the condition \( \alpha[u] + \alpha[v] \geq T \) in view of the definition of \( T \), so the difference subgraph \( D \) is empty. It follows that \( \alpha, \mu, S \) and \( T \) correctly define \( G_1 \cup G_2 \).

\[\square\]

**Lemma 5.** Let be given two threshold graphs \( G_1 \) and \( G_2 \) by means of \( \delta_1[0..m] \), \( \mu_1[0..m] \) and \( \delta_2[0..m] \), \( \mu_2[0..m] \) and let \( S = m + 1 \) be their common threshold. The following function determines the threshold signed graph \( G_1 \cup G_2 \).

\[
\text{Join}(\delta_1, \mu_1, \delta_2, \mu_2, m)
\begin{align*}
\alpha[0] &\leftarrow 0; \mu[0] \leftarrow 0; \\
\text{IF } \mu_1[0] \neq 0 \\
& \text{THEN } \text{flag}1 \leftarrow 0 \\
& \text{ELSE } \text{flag}1 \leftarrow 1 \\
\text{FOR } i =\text{flag}1 \text{ TO } m \text{ DO}
\end{align*}
\]
Proof. Preliminarily, observe that added: we do this exploiting the two boolean variables flag1 and flag2.

\[ \alpha[i + 1 - \text{flag1}] \leftarrow -i; \mu[i + 1 - \text{flag1}] \leftarrow \mu_1[i]; \]

IF \( \mu_2[0] \neq 0 \)
THEN flag2 \( \leftarrow 0 \)
ELSE flag2 \( \leftarrow 1 \)
FOR \( i = \text{flag2} \) TO \( m \) DO
\[ \alpha[m + i + 2 - \text{flag1} - \text{flag2}] \leftarrow i; \mu[m + i + 2 - \text{flag1} - \text{flag2}] \leftarrow \mu_2[i]; \]
\[ \Delta \leftarrow 2m + 2 - \text{flag1} - \text{flag2}; \]
\[ S \leftarrow m + 1; \]
\[ T \leftarrow \min_{i \leq \Delta} \{|\alpha[i]|\}; \]
\[ k \leftarrow m - T + 1; \]
FOR \( i = 1 \) TO \( m + 1 - \text{flag1} \) DO
\[ \alpha[i] \leftarrow \alpha[i] - k; \]
FOR \( i = m + 2 - \text{flag1} \) TO \( 2m + 2 - \text{flag1} - \text{flag2} \) DO
\[ \alpha[i] \leftarrow \alpha[i] + k; \]
\[ S \leftarrow S + 2k; \]
\[ T \leftarrow T + 2k; \]
RETURN \( (\alpha, \mu, S, T, \Delta) \).

Proof. Preliminarily, observe that \( G_1 + G_2 \) cannot have isolated nodes, so we set \( \mu[0] \) to 0; moreover, if either \( G_1 \) or \( G_2 \) contain isolated nodes, a box needs to be added: we do this exploiting the two boolean variables flag1 and flag2.

So, even in this case, the two threshold graphs \( G^- \) and \( G^+ \) of \( G_1 + G_2 \) are exactly the same as \( G_1 \) and \( G_2 \), respectively. \( T \) is set to the modulo of the smallest node weight in order to guarantee that the difference graph \( D \) is a complete bipartite graph. In this way, \( T \) results in a value that contradicts Property 5 of Definition 5. By incrementing the modulo of each \( \alpha[i] \) of an opportune value \( k \) and \( S \) and \( T \) by \( 2k \), we are able to restore the inequality.

It remains to handle the case in which \( G_1 \) and \( G_2 \) have different thresholds. In this case, we propose to the functions described in the proofs of Lemmas 4 and 5 a preprocessing phase that equalize their thresholds as detailed in the following lemma, where with the notation \( \alpha' = xa + y \) (where \( x \) and \( y \) are integer values and \( a \) is a node weight function) we compactly mean that, for each node \( v \), \( \alpha'(v) = xa(v) + y \). We want to underline that now on we represent a threshold graph in terms of its separator, instead of in terms of our data structure, because the description of the equalization appears more comprehensive.

**Lemma 6.** Let be given two thresholds graphs \( G_1 \) and \( G_2 \) and let \((a_1, S_1)\) and \((a_2, S_2)\) be their integral separators with \( S_1 < S_2 \). Then \((a'_1, S'_1) = (2a_1 + S_2 - S_1, 2S_2)\) is an integral separator for \( G_1 \) and \((a'_1, a'_2) = (2a_2, 2S_2)\) is an integral separator for \( G_2 \).

Proof. Let \((v, w)\) be an edge in \( G_1 \), i.e. \( a_1(v) + a_1(w) \geq S_1 \); then \( a'_1(v) + a'_1(w) = 2a_1(v) + S_2 - S_1 + 2a_1(w) + S_2 - S_1 \geq 2S_2 = S'_1 \). In the same way, let \( v \) and \( w \) be not connected in \( G_1 \), i.e. \( a_1(v) + a_1(w) < S_1 \); then \( a'_1(v) + a'_1(w) = 2a_1 + S_2 - S_1 < 2S_2 = S'_1 \). Finally, the pair \((a'_1, S'_1)\) is a feasible integral separator since, for any node \( v \), \( a_1(v) < S_1 \) implies \( a'_1(v) < S'_1 \). Analogous reasonings lead to prove that \((a'_2, S'_2)\) is an integral separator for \( G_2 \). Function **EqualizeThresholds** in the Appendix details the proof. □
Notice that the values of the node weight function and of the two thresholds $S$ and $T$ of the resulting threshold signed graphs will be integral but not necessarily minimal.

References

Appendix

IncreaseDeg($\delta, \mu, i, \text{dim}$)

IF $(i \neq \text{dim} \text{ AND } \delta[i] + 1 = \delta[i + 1])$ THEN
\[
\begin{align*}
\mu[i] &\leftarrow \mu[i] - 1; \\
\mu[i + 1] &\leftarrow \mu[i + 1] + 1;
\end{align*}
\]
ELSE
IF $(i \neq 0 \text{ AND } \mu[i] = 0)$ THEN
\[
\begin{align*}
\text{FOR } k = i \text{ TO } \text{dim} - 1 \text{ DO} & \\
\delta[k] &\leftarrow \delta[k + 1]; \\
\mu[k] &\leftarrow \mu[k + 1]; \\
\delta[i + 1] &\leftarrow \delta[i] + 1; \\
\mu[i + 1] &\leftarrow 1; \\
\text{dim} &\leftarrow \text{dim} + 1
\end{align*}
\]
ELSE
IF $(i \neq 0 \text{ AND } \mu[i] = 1)$ THEN
\[
\begin{align*}
\delta[i] &\leftarrow \delta[i] + 1;
\end{align*}
\]
ELSE
\[
\begin{align*}
\mu[i] &\leftarrow \mu[i] - 1; \\
\text{FOR } k = \text{dim} \text{ DOWNTO } i + 1 \text{ DO} & \\
\delta[k + 1] &\leftarrow \delta[k]; \\
\mu[k + 1] &\leftarrow \mu[k]; \\
\delta[i + 1] &\leftarrow \delta[i] + 1; \\
\mu[i + 1] &\leftarrow 1; \\
\text{dim} &\leftarrow \text{dim} + 1
\end{align*}
\]
RETURN($\delta, \mu, \text{dim}$).

DecreaseDeg($\delta, \mu, i, \text{dim}$)

IF $(\delta[i] - 1 = \delta[i - 1])$ THEN
\[
\begin{align*}
\mu[i] &\leftarrow \mu[i] - 1; \\
\mu[i - 1] &\leftarrow \mu[i - 1] + 1;
\end{align*}
\]
ELSE
IF $(\mu[i] = 0)$ THEN
\[
\begin{align*}
\text{FOR } k = i + 1 \text{ TO } \text{dim} \text{ DO} & \\
\delta[k - 1] &\leftarrow \delta[k]; \\
\mu[k - 1] &\leftarrow \mu[k]; \\
\text{dim} &\leftarrow \text{dim} - 1
\end{align*}
\]
ELSE
IF $(\mu[i] = 1)$ THEN
\[
\begin{align*}
\delta[i] &\leftarrow \delta[i] - 1;
\end{align*}
\]
ELSE
\[
\begin{align*}
\mu[i] &\leftarrow \mu[i] - 1; \\
\text{FOR } k = \text{dim} \text{ DOWNTO } i \text{ DO} & \\
\delta[k + 1] &\leftarrow \delta[k]; \\
\mu[k + 1] &\leftarrow \mu[k]; \\
\delta[i + 1] &\leftarrow \delta[i] - 1; \\
\mu[i] &\leftarrow 1; \\
\text{dim} &\leftarrow \text{dim} + 1
\end{align*}
\]
RETURN($\delta, \mu, \text{dim}$).

IncreaseDegreeOfSetNode($\delta, \mu, d, \text{dim}$)

\[
i \leftarrow \text{dim}; a \leftarrow d;
\]
WHILE $(a \geq \mu[i])$ DO
\[
\begin{align*}
a &\leftarrow a - \mu[i]; \\
\delta[i] &\leftarrow \delta[i] + 1; \\
i &\leftarrow i - 1;
\end{align*}
\]
IF $(a \neq 0)$ THEN
\[
\begin{align*}
\text{dim} &\leftarrow \text{dim} + 1; \\
\text{FOR } j = \text{dim} \text{ DOWNTO } i + 2 & \\
\mu[j] &\leftarrow \mu[j - 1]; \\
\delta[j] &\leftarrow \delta[j - 1]; \\
\delta[i + 1] &\leftarrow \delta[i] + 1; \\
\mu[i + 1] &\leftarrow a; \\
\mu[i] &\leftarrow \mu[i] - a;
\end{align*}
\]
RETURN($\delta, \mu, \text{dim}$).

DecreaseDegreeOfSetNode($\delta, \mu, d, \text{dim}$)

\[
i \leftarrow \text{dim}; a \leftarrow d;
\]
WHILE $(a > 0)$ DO
\[
\begin{align*}
a &\leftarrow a - \mu[i]; \\
\delta[i] &\leftarrow \delta[i] - 1; \\
i &\leftarrow i - 1;
\end{align*}
\]
IF $(i \neq \text{dim} \text{ AND } \delta[i] = \delta[i + 1])$ THEN
\[
\begin{align*}
\mu[i] &\leftarrow \mu[i] + \mu[i + 1]; \\
\text{FOR } k = i + 1 \text{ TO } \text{dim} - 1 \text{ DO} & \\
\delta[k] &\leftarrow \delta[k + 1]; \\
\mu[k] &\leftarrow \mu[k + 1]; \\
\text{dim} &\leftarrow \text{dim} - 1;
\end{align*}
\]
RETURN($\delta, \mu, \text{dim}$).
+Node(δ, µ, d, dim))
i ← 0;
WHILE(i ≤ dim AND d > δ[i]) DO i = i + 1;
IF (i ≤ dim AND d = δ[i]) THEN µ[i] ← µ[i] + 1;
ELSE
   dim ← dim + 1;
   FOR k = dim DOWNTO i + 1 DO
      δ[k] ← δ[k − 1];
      µ[k] ← µ[k − 1];
   ENDFOR
   δ[i] ← d;
   µ[i] ← 1;
RETURN(δ, µ, dim).

−Node(δ, µ, i, dim)
µ[i] ← µ[i] − 1;
IF (i ≠ 0 AND µ[i] ≠ 0) THEN
   FOR k = i TO dim − 1 DO
      δ[k] ← δ[k + 1];
      µ[k] ← µ[k + 1];
   ENDFOR
   dim ← dim − 1;
RETURN(δ, µ, dim).

In order not to overburden the pseudocode, we present only the case in which S1 ≤ S2

EqualizeThresholds(a, n1, n2, S1, S2)
IF S1 = S2 THEN RETURN
ELSE
   IF S1 and S2 have different parity THEN
      S1 ← 2S1;
      S2 ← 2S2;
      FOR i = 1 TO n1 + n2 DO
         a[i] ← 2a[i];
      ENDFOR
      k ← (S2 − S1)/2;
      FOR i = 0 TO n1 DO
         a[i] ← a[i] + k;
      S1 ← S1 + 2k;
   ELSE
      RETURN.