CS-E4075 - Special Course in Machine Learning, Data Science and Artificial Intelligence D: Signed graphs: spectral theory and applications

Problem set 1 - Model Solutions
2021

Mathematical exercises

Exercise 1

Credit: Yan Xia

1. For any vector $x \in \mathbb{R}^n$, simple unsigned graph $G = (V, E)$ and its graph Laplacian L,

$$\begin{aligned}
x^T L x &= \sum_{i=1}^{n} x_i^2 L_{ii} + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} x_i x_j L_{ij} \\
&= \sum_{i=1}^{n} x_i^2 d_i - 2 \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j > i}}^{n} x_i x_j A_{ij} \\
&= \sum_{\substack{(i,j) \in E \\ i<j}} (x_i^2 + x_j^2 - 2x_i x_j) \\
&= \sum_{\substack{(i,j) \in E \\ i<j}} (x_i - x_j)^2 \\
&\geq 0
\end{aligned}$$

   thus L is positive semidefinite.

2. For any vector $x \in \mathbb{R}^n$, simple signed graph $G = (V, E^+, E^-)$ and its signed Laplacian L,

$$\begin{aligned}
x^T L x &= \sum_{i=1}^{n} x_i^2 L_{ii} + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} x_i x_j L_{ij} \\
&= \sum_{i=1}^{n} x_i^2 d_i - 2 \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j > i}}^{n} x_i x_j A_{ij} \\
&= \sum_{\substack{(i,j) \in E^+ \\ i<j}} (x_i^2 + x_j^2 - 2x_i x_j) + \sum_{\substack{(i,j) \in E^- \\ i<j}} (x_i^2 + x_j^2 + 2x_i x_j) \\
&= \sum_{\substack{(i,j) \in E^+ \\ i<j}} (x_i - x_j)^2 + \sum_{\substack{(i,j) \in E^- \\ i<j}} (x_i + x_j)^2 \\
&\geq 0
\end{aligned}$$

   thus L is positive semidefinite.

Exercise 2

Credit: Ameet Gadekar

The Laplacian $\mathcal{L}'$ of star graph G with first vertex as the center and only second vertex having negative edge looks like

$$\begin{bmatrix}
n-1 & 1 & -1 & \cdots & \cdots & -1 \\
1 & 1 & 0 & \cdots & \cdots & 0 \\
-1 & 0 & 1 & 0 & \cdots & 0 \\
\vdots & & \ddots & & & \\
-1 & 0 & \cdots & \cdots & \cdots & 1
\end{bmatrix}$$

Since the graph is balanced (as there are no cycles), we know that the spectrum of this signed star graph G is same as the spectrum of unsigned graph $|G|$. Hence, it is sufficient to calculate

all the eigenvalues of $|G|$. Let $\mathcal{L}$ be the Laplacian of $|G|$. Let $V = \{v_1, \cdots, v_n\}$ be eigen vectors of $\mathcal{L}$ corresponding to eigen values $\Lambda = \{\lambda_1, \cdots, \lambda_n\}$ From the lecture, we know that all ones vector $\vec{1}$ is an eigen vector corresponding to eigen value 0 since $\mathcal{L}\vec{1} = 0\vec{1}$. Hence, $v_1 = \vec{1}$ and $\lambda_1 = 0$.
Next consider the vector $v_n = (-(n-1), 1, \cdots, 1)^T$,i.e, $v_n = -(n-1)e_1 + e_2 + \cdots + e_n$, where e_is are the standard basis vectors of $\mathbb{R}^n$. Then,

$$\mathcal{L}v_n = \begin{bmatrix} -(n-1)^2 - (n-1) \\ n \\ \vdots \\ n \end{bmatrix} = \begin{bmatrix} -n(n-1) \\ n \\ \vdots \\ n \end{bmatrix} = n \begin{bmatrix} -(n-1) \\ 1 \\ \vdots \\ 1 \end{bmatrix} = nv_n$$

Thus, v_n is an eigen vector corresponding to the eigen value $\lambda_n = n$. Now, we will show that there are precisely $(n-2)$ linearly independent eigen vectors corresponding to the eigen value 1. This concludes the proof that $\mathcal{L}$ has exactly 3 distinct eigen values.
Now eigen vectors corresponding to eigen value 1 are precisely the vectors in the null space of $A = \mathcal{L} - I$, where $I \in \mathbb{R}^{n \times n}$ is the identity matrix. But then, A has only two linearly independent columns, the first column $(n-2, -1, \cdots, -1)^T$ and the second column $-e_1$, and the rest columns are all $-e_1$. Hence, $Rank(A) = 2$, which implies that $dim(Nullspace(A)) = n - 2$.

Exercise 3

Credit: Ameet Gadekar

1. Let A be the adjacency matrix of a graph G on n vertices. Let λ_{max} be the largest eigenvalue of A. From Rayleigh quotient, we have

$$\lambda_{max} = \max_{x \neq 0} \frac{x^T A x}{x^T x}$$

Let $d_{avg} := \frac{\sum_{i=1}^n d_i}{n}$, where d_i is the degree of i^{th} vertex. In the above equation, we use $x = \frac{1}{\sqrt{n}}\vec{1}$, and noting $x^T x = 1/n \cdot n = 1$,

$$\lambda_{max} \geq \frac{1}{\sqrt{n}}\vec{1}^T A \frac{1}{\sqrt{n}}\vec{1} = \frac{1}{n}\vec{1}^T \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} = \frac{1}{n}\sum_{i=1}^n d_i = d_{avg}$$

2. Let $G = (V, E)$ be the given graph. Let λ refers to eigenvalue(s) of the adjacency matrix A, and α refers to eigenvalue(s) of the Laplacian $\mathcal{L}$. Let $k \in V$ be the maximum degree vertex, and let $d_k = d_{max}$. Now, consider the matrix

$$D_{max} := d_{max} I$$

where $I \in R^{n \times n}$ is the identity matrix. Let $B \in \mathbb{R}^{n \times n}$ be such that

$$B := D_{max} - D$$

where D is the degree matrix of G. Note that,

$$B_{ii} = d_{max} - d_i$$

and $B_{ij} = 0$. Also, note that $B_{kk} = 0$. Let β be eigenvalue(s) of B. Then, observe that since B is a diagonal matrix with $B_{kk} = 0$, we have

$$\beta_{min} = 0$$

Further, we have that

$$A = D - \mathcal{L} = D_{max} - B - \mathcal{L}.$$

Consider any $y \neq 0 \in \mathbb{R}^n$, then

$$\frac{y^T \mathcal{L} y}{y^T y} \geq \alpha_{min}$$

using Rayleigh quotient. Now consider

$$\begin{aligned}
\frac{y^T A y}{y^T y} &= \frac{y^T D_{max} y}{y^T y} - \frac{y^T B y}{y^T y} - \frac{y^T \mathcal{L} y}{y^T y} \\
&= d_{max} \frac{y^T I y}{y^T y} - \frac{y^T B y}{y^T y} - \frac{y^T \mathcal{L} y}{y^T y} \\
&= d_{max} - \frac{y^T B y}{y^T y} - \frac{y^T \mathcal{L} y}{y^T y} \\
&\leq d_{max} - \beta_{min} - \alpha_{min} \\
&= d_{max} \qquad \text{since } \alpha_{min} = 0
\end{aligned}$$

Since the above holds for every non-zero y, it also holds for eigenvector $z \in \mathbb{R}^n$ of A corresponding to λ_{max}. Hence,

$$\lambda_{max} = \frac{z^T A z}{z^T z} \leq d_{max}.$$

Exercise 4

Credit: Ameet Gadekar

$\Longrightarrow$ Assume $G = (V, E)$ has no negative cycles. We show an algorithm to find the partition $V = V_1 \dot\cup V_2$ such that every negative edge $(i, j) \in E(V_1, V_2)$, and every positive edge $(i, j) \in G[V_1]$ or $(i, j) \in G[V_2]$. Assume G is connected (or else repeat the algorithm and argument for each connected component).

1. Choose any vertex $v \in V$.
2. Put v in V_1.
3. Do BFS from v and parallelly construct V_1 and V_2 as follows:
    (a) if (v_i, v_j) is a negative edge, then put v_j in different partition than that of v_i.
    (b) if (v_i, v_j) is a positive edge, then put v_j in the same partition of v_i.

For correctness of the algorithm, we show two claims. Let $T = (V, E_T)$ be the BFS tree discovered by the algorithm. Then, note that any edge $e \notin E_T$ is either a same level edge or a cross edge in the adjacent level, but never a backedge. Further, also observe that for $e \in E_T$, the algorithm correctly classifies e, i.e., if e is a negative edge, then $e \in E(V_1, V_2)$, whereas if e is a positive edge, then $e \in G[V_1]$ or $e \in G[V_2]$. So, we have to prove the correctness for $e \notin E_T$.
Claim: There are no negative edges within the partitions.
Proof. Suppose there is a negative edge $e = (v_i, v_j) \in E \setminus E_T$ in partition V_1 (w.l.o.g.). Let v_c be the least common ancestor of v_i and v_j in T. Note that v_c has to be different than v_i and v_j. Let

P_i and P_j be the paths from v_c to v_i and v_j respectively. Let n_i and n_j the number of negative edges in the path P_i and P_j respectively. Now, if $v_c \in V_1$, then

$$n_i \mod 2 = n_j \mod 2 = 0.$$

Then, consider the cycle $C = (v_c, P_i, v_i, v_j, P_j, v_c)$. C is a negative cycle since the number of negative edges is odd because

$$n_i \mod 2 + n_j \mod 2 + 1 = 1$$

which is a contradiction. Now assume $v_c \in V_2$. Since, $v_i, v_j \in V_1$, we have that

$$n_i \mod 2 = n_j \mod 2 = 1.$$

Hence, the cycle $C = (v_c, P_i, v_i, v_j, P_j)$ is negative because,

$$n_i \mod 2 + n_j \mod 2 + 1 = 1.$$

□

Claim: There are no positive edges crossing the partition.
Proof. Suppose there is a positive edge $e = (v_i, v_j) \in E \setminus E_T$ crossing the partition. Let v_c be the lowest common ancestor of v_i and v_j. Note that v_c has to be different than v_i and v_j. Let n_i be the number of negative edges in the path P_i from v_c to v_i in T. Similarly, let n_j be the number of negative edges in the path P_j from v_j to v_c in T. Now consider the cycle $C = (v_c, P_i, v_i, v_j, P_j, v_c)$ in G. Since v_i and v_j are in different partitions, we have that

$$n_i \mod 2 \neq n_j \mod 2$$

This implies that $n_i + n_j$ is odd. Since (v_i, v_j) is a positive edge, it means that C is a negative cycle, which is a contradiction. □

$\Leftarrow$ Assume there is a partition $V = V_1 \dot\cup V_2$ such that every negative edge $(i, j) \in E(V_1, V_2)$, and every positive edge $(i, j) \in G[V_1]$ or $(i, j) \in G[V_2]$. Then, consider any cycle $C = (v_1, v_2, \cdots, v_k, v_1)$. If C has no negative edges, then we are done since C is not a negative cycle. Similarly, if C has even number of negative edges, then also C is not a negative cycle. Suppose there are $2s + 1$ negative edges for some positive integer s. Also, assume without loss of generality that $v_1 \in V_1$. Now, we trace cycle C starting from v_1, which is in V_1. Then, note that since all the negative edges are cut by the partition (V_1, V_2), every negative edge of C makes us switch the partition. But then there are $2s + 1$ negative edges in C, which implies that $v_1 \in V_2$, which is a contradiction.

Exercise 5

Credit: Nidia Acosta Obscura
We show that when the graph is balanced, connected and regular, then the adjacency matrix and the Laplacian have the same eigenvectors and we can obtain the same information from the eigenvectors of the adjacency matrix. First, we can rewrite $L = D - A$, call this equation 1. Take any eigenvector v_i of L and multiply it on the left by equation 1. This gives us $(v_i)L = (v_i)(D - A) = (v_i)D - (v_i)A$. Rewriting and using the fact that v_i is an eigenvector with eigenvalue λ for L we get: $(v_i)D - \lambda(v_i) = (v_i)A$. Now, since D is the degree matrix of a regular graph, we can rewrite $D = dI$ where d is the degree of the graph and I is the identity matrix. Hence, $(v_i)D = (v_i)(dI) = d(v_i)$ and rewriting everything gives us $(d - \lambda)v_i = (v_i)A$. This implies that v_i is also an eigenvector for A with eigenvalue $d - \lambda$. Finally, in order to find the sign-compliant partition we know that the smallest eigenvector v_i in L is now going to be the largest eigenvector in A as the eigenvalue changes from λ to $d - \lambda$.

Exercise 6

Credit: Nidia Acosta Obscura
Let us first note that switching a set of nodes S all at once gives us the same result as switching each node one at a time. This is because whenever we switch an edge (s, t) where $s \in S$ and $t \notin S$, we do this only once (when switching the node s, as we switch it only once and the other endpoint $t \notin S$ never gets switched), and whenever we switch an edge (s_1, s_2) with $s_1, s_2 \in S$ we switch it always exactly two times (once when switching the node s_1 and once when switching the node s_2). This implies that edges between the set S are switched twice and hence remained unchanged and edges in the cut are switched exactly once, as we wanted.
Now, we can restrict ourselves to considering single node switching operations which would look as follows: for switching node i, we need to change the sign of all the edges adjacent to it, so in an adjacency matrix would correspond to changing the sign of all the entries in row i and all the entries in column i (except for the diagonal which is zero). To do this, we can simply multiply by the matrix P equal to the identity matrix with a -1 at position $I_{i,i}$, both from the left (to switch the row i of A) and from the right (to switch the column i of A). Since P is its own inverse, the above procedure would be to apply the similarity transformation of $PAP^{-1} = A'$. By the hint, we know that A and A' have the same spectra and hence applying one node switching step does not change the eigenvalues of A. For L note that we can use the same analysis, as the switching operation does not change the degrees of vertices and the entry at the diagonal on row i is equal to the entry at the diagonal on column i so hence the entry in L changes sign twice for PLP^{-1} and so it remains unchanged.

Exercise 7

Credit: Ameet Gadekar

1. Let $\mathcal{L}$ and A be the Laplacian and adjacency matrix of C_n respectively, where n is even. Then, letting $D := 2I \in \mathbb{R}^{n\times n}$, we note that

$$\mathcal{L} = D - A$$

Hence, $\mathcal{L}$ and A has same set of eigenvectors. Suppose v is an eigenvector of $\mathcal{L}$ corresponding to eigenvalue λ, then

$$\lambda v = \mathcal{L} v = Dv - Av = (2 - \beta_i)v$$

where β is the eigenvalue of A corresponding to eigenvector v. Hence $\mathcal{L}$ has eigenvalue 4 if and only if A has eigenvalue -2. So, now we show that A, in fact, has an eigenvalue -2.
Consider the following vector $w \in \mathbb{R}^n$:

$$w_i = \begin{cases} -1 & \text{if } i \text{ is even} \\ 1 & \text{if } i \text{ is odd} \end{cases}$$

Let $Aw = u$. The following claim says that $u = (-2)w$, which implies $Aw = (-2)w$. **Claim:** $u_i = -2w_i$.
Proof. Fix $i \in [n]$. Let A_i be the i^{th} row of A_i. Since n is even, we have that the two neighbours of i are[1] - $(i-1)$ and $(i+1)$. Hence, $A_{i,i-1} = A_{i,i+1} = 1$, and rest entries of A_i are zeroes. Suppose

[1] $(i + 1)$ is defined as 1, when $i = n$. Similarly, $(i - 1)$ is defined as n, when $i = 1$.

i is even. Then, note that $w_i = -1$. Further, since $(i-1)$ and $(i+1)$ are odd, we have that $w_{i-1} = w_{i+1} = 1$. Hence,

$$u_i = A_i w = A_{i,i-1} w_{i-1} + A_{i,i+1} w_{i+1} = 1(1) + 1(1) = 2 = (-2)w_i.$$

Now, when i is odd, we have that $w_i = 1$. Also, since $(i-1)$ and $(i+1)$ is even, we have $w_{i-1} = w_{i+1} = -1$. Thus,

$$u_i = A_i w = A_{i,i-1} w_{i-1} + A_{i,i+1} w_{i+1} = 1(-1) + 1(-1) = -2 = (-2)w_i.$$

□

2. From Gershgorin theorem, we know that every eigenvalue of $\mathcal{L}$ lies within one of the Gershgorin disc $D(\mathcal{L}_{ii}, R_i)$, where

$$R_i := \sum_{j \neq i} |\mathcal{L}_{ij}|$$

Note for cycle, $\mathcal{L}_{ii} = 2$, for all $i \in [n]$, and

$$R_i = 2$$

since each row has preciely two -1s corresponding to its two neighbours. So, Gershgorin theorem says that all eigenvalues of $\mathcal{L}$ lie in the disc $D(2,2)$. Hence,

$$\lambda_{max} \leq 4.$$

So, $\lambda_{max} = 4$, when n is even.

Exercise 8

Credit: Ameet Gadekar

We show a combinatorial algorithm to find if G is balanced or not. The algorithm is same as that we used in the first part of exercise 4.

```
1. Repeat for every connected component of $G$
   (a) Choose any vertex $v \in V$ in the connected component.
   (b) Put $v$ in $V_1$.
   (c) Do BFS from $v$ and parallelly construct $V_1$ and $V_2$ as follows:
       i. if $(v_i, v_j)$ is a negative edge, then put $v_j$ in different partition than that of $v_i$.
       ii. if $(v_i, v_j)$ is a positive edge, then put $v_j$ in the same partition of $v_i$.
2. For every edge $e = (v_i, v_j)$ in $G$
   • If $e$ is a positive edge
     – If $v_i$ and $v_j$ are in different partition
       * Return FAIL
   • If $e$ is a negative edge
     – If $v_i$ and $v_j$ are in same partition
       * Return FAIL
3. Return $(V_1, V_2)$
```

Running time: BFS runs in linear time in the number of edges. Further, the sanity check Step 2 considers every edge, and figures out the partitions of its endpoints, which is linear in the number of vertices. Hence the total running time of the algorithm is cubic in the of number of vertices.[2]

[2]Can be improved to quadratic by using array to store the partition.

Correctness: First, assume G is balanced. Then, by definition there are no negative cycles in G. From the two claims in the solution to exercise 4, we note that the algorithm finds a partition such that every negative edge cuts the partition and every positive edge is within a partition.
Assume $V = V_1 \dot\cup V_2$ is the partition returned by the algorithm. Then, by sanity check of Step 2, every negative edge cuts the partition and every positive edge is within a partition. Hence, by definition, this implies that G is balanced.

Exercise 9

Credit: Yan Xia
Let us consider a 2-partitioning $V = (V_1, V_2)$ of graph G that has the minimum number of "incorrect edges", namely within-partition negative edges or cross-partition positive edges. According to what we have proved in Exercise 4, the graph will be balanced after removing those incorrect edges, and thus the number of incorrect edges under this partition is exactly the edge frustration f_e of G. Let E_i denote the set of incorrect edges, and E_c denote the set of correct edges under this partition.
Then we define a vector $x \in \mathbb{R}^n$ so that $x_i = 1$ if vertex $i \in V_1$, and $x_i = -1$ if $i \in V_2$. Then

$$\begin{aligned}\frac{x^TLx}{x^Tx} &= \frac{\sum_{i=1}^n x_i^2 L_{ii} + \sum_{i=1}^n \sum_{\substack{j=1\\j\neq i}}^n x_i x_j L_{ij}}{n} \\ &= \frac{\sum_{i=1}^n d_i - \sum_{i=1}^n \sum_{\substack{j=1\\j\neq i}}^n x_i x_j A_{ij}}{n} \\ &= \frac{2|E| - \sum_{i=1}^n \sum_{\substack{j=1\\j\neq i}}^n x_i x_j A_{ij}}{n}\end{aligned}$$

We further observe that $x_i x_j A_{ij} = 1$ if (i, j) is a "correct edge" (within-partition positive edge / cross-partition negative edge), and $x_i x_j A_{ij} = -1$ if (i, j) is an "incorrect edge" (within-partition negative edge / cross-partition positive edge). Thus

$$\begin{aligned}\frac{x^TLx}{x^Tx} &= \frac{2|E| - 2|E_c| + 2|E_i|}{n} \\ &= \frac{4|E_i|}{n} \\ &= \frac{4}{n} f_e\end{aligned}$$

Since we know $\frac{x^TLx}{x^Tx}$ takes minimum value λ_{min},

$$\frac{4}{n} f_e = \frac{x^TLx}{x^Tx} \geq \lambda_{min}$$

$$\lambda_{min} \leq \frac{n}{4} f_e$$

When $n \geq 4$, it is clear that $\lambda_{min} \leq f_e$ holds.
When $n = 2$, it is easy to see that $f_e = 0$ and $\lambda_{min} = 0$ because the graph is always balanced, and thus $\lambda_{min} \leq f_e$ holds.
When $n = 3$, $f_e = 0$, $\lambda_{min} = 0$ and $\lambda_{min} \leq f_e$ obviously holds when the graph is balanced. When it is not (i.e. when it contains a cycle of one negative edge and two positive edges, or a cycle of three negative edges), through simple calculation it is easy to see that $f_e = 1$ and $\lambda_{min} = 1$, thus $\lambda_{min} \leq f_e$ also holds.

Exercise 10

Credit: Ameet Gadekar

Given a signed graph $G = (V, E)$, $f_e(G)$ is minimum number of edges to be removed from G to make it balanced. Here is a brute force algorithm to find $f_e(G)$.

```
1. $a \longleftarrow 0$.
2. For every subset $F \subseteq E$
      (a) Let $G' = (V, E \setminus F)$.
      (b) If ISBALANCED($G'$) == True
            • If $|F| < a$
                 – $a \longleftarrow |F|$
3. RETURN $a$
```

`ISBALANCED` subroutine is the algorithm of Exercise 8, which is modified to return `True` in Step 3, instead of returning the partition.

Running time: Let $m := |E|$ and $n := |V|$. Since the algorithm goes over all the subsets of E, Step 2 is repeated 2^m times. Step $2a$ takes polynomial time in m, and Step $2b$ also takes polynomial time in m from Exercise 8. Hence the total running time is $O(2^m poly(m))$.

Correctness Let $F^* \subseteq E$ be a edge set corresponding to the optimal value f_e, i.e., $|F^*| = f_e$, and $G'' = (V, E \setminus F^*)$ is balanced. Then, note that Step $2b$ is satisfied since `ISBALANCED`(G'') returns true. Hence, $a \leq f_e$. Further, there is no set $W \subseteq E$ such that $|W| < f_e$ and $G' = (V, E \setminus W)$ is balanced, by definition of f_e. Hence, $a = f_e$.

Programming exercises

Exercise 11

Credit: Yan Xia

1. The visualization of the graph before the transformation is shown in Figure 1.

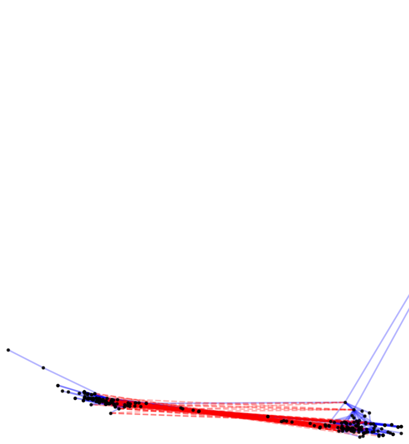

Figure 1: Visualization of the graph before the transformation.

2. To spread out the vertex positions, we take the square root of the coordinates (for negative values, take the negative square root of the absolute value). The visualization of the graph after the transformation is shown in Figure 2.

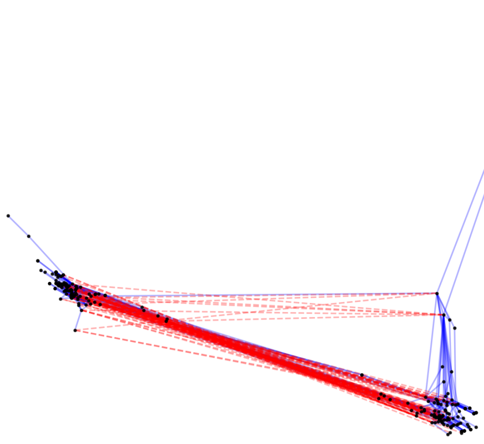

Figure 2: Visualization of the graph after the transformation.

Exercise 12

Credit: Yan Xia
The algorithm proceeds as in Algorithm 1. It runs in $O(|V|(|V| + |E|))$ time.

Algorithm 1 The MaximalBalancedSubgraph algorithm.

Input: Signed graph $G = (V, E^+, E^-)$. Function Subgraph(G, S) that returns the subgraph of G induced by vertex set S. Function CheckBalanced(G) as described in Algorithm **??**.
Output: The maximal balanced subgraph of G.

```
1: $S \leftarrow \emptyset$
2: for $v$ in $V$ do
3:     $S' \leftarrow S \cup \{v\}$
4:     $H \leftarrow$ Subgraph$(G, S')$
5:     if CheckBalanced$(H)$ = True then
6:         $S \leftarrow S'$
7:     end if
8: end for
9: return Subgraph$(G, S)$
```

The vertices corresponding to the located balanced subgraph are:
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 70, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 141, 143, 144, 145, 146, 147, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218]