A Better Sets Approach Algorithm for the Determination of Component Minimal Complete Subgraph

Silviu Bârză
Faculty of Mathematics and Informatics
Spiru Haret University
Bucharest, Romania
silviu.barza@gmail.com

Abstract
Giving a graph G we wish to determine the component minimal complete subgraph of G. For this goal we present one algorithm in terms of sets representation of graph G. We give too the theorem that characterized this algorithm which is a better algorithm then the two algorithms presented in a previous paper and I show that this it happens.
Keywords: *subgraphs, complete graphs*
ACM/AMS Classification: 05C30

1. General definitions and results

Firstly I wish to give some of the graph theory definitions (see [Bârză and Morogan 2008]) and a few results related to component minimal complete subgraphs (see [Bârză, 2012]).

Definition 1.1 *Let*

$$V = \{x_1, x_2, \ldots, x_n\}$$

be a finite and non-empty set and

$$E = \{\{x, y\} \,|x, y \in X, x \neq y\}\,.$$

*The pair $G = (V, E)$ is a **graph**, elements in V are named **vertices** and elements in E are named **edge**.*

Definition 1.2 *Let $G = (V, E)$ be a graph. A sequence of vertices $y_1, y_2, \ldots, y_k$ is named **walk** in G if $\{y_i, y_{i+1}\} \in E$ for any $i = 1, 2, \ldots, k-1$. If $y_1 = y_k$ the walk is named **loop**. The walk (and the loop) is specified as*

$$L = [y_1, y_2, \ldots, y_k].$$

Definition 1.3 *Let $G = (V, E)$ be a graph. If $W \subset V$ and*

$$F = \{\{x, y\} \in E|x, y \in W\}$$

then the graph $H = (W, F)$ is named ***subgraph*** *of G.*

Definition 1.4 *Let $G = (V, E)$ be a graph. If for any $x, y \in V$ there exists a walk from x to y then G is named* ***connected****. If G is not connected, G is named* ***disconnected****.*

Definition 1.5 *Let $G = (V, E)$ be a graph. A subgraph $H = (W, F)$ of G which is connected and there does not exists a walk from x to y for any $x \in W$ and $y \in V - W$ is named* ***component*** *of G.*

Proposition 1.1 *Let $G = (V, E)$ be a graph. There exists a partition of V with the sets $V_1, V_2, \ldots, V_k$ ($V_1 \cup V_2 \cup \ldots \cup V_k = V$ and $V_i \cap V_j = \oslash$ for any $i, j = 1, 2, \ldots, k$, $i \neq j$) so that subgraphs $G_i = (V_i, F_i)$ are components in G.*

Definition 1.6 *Let $G = (V, E)$ be a graph. If G is connected and it does not contain loops then the graph is named* ***tree****.*

Proposition 1.2 *Let $G = (V, E)$ be a connected graph. G is a tree if and only if $|E| = |V| - 1$*

Definition 1.7 *Let $G = (V, E)$ be a graph with $|V| = p \geq 3$. If for any $x, y \in V$, $x \neq y$, $\{x, y\} \in E$ then G is named* ***complete*** *and we write it as K_p.*

Definition 1.8 *Let $G = (V, E)$ be a graph. If for any $x \in V$ there exist $y, z \in V$, $y \neq z$ so that $\{\{x, y\}, \{x, z\}, \{y, z\}\} \subset E$ we say that G is a* ***minimal complete graph****.*

Definition 1.9 *Let $G = (V, E)$ be a graph. If there exists $W \subseteq V$ so that the subgraph $H = (W, F)$ is a complete graph, with $|W| = p$ and for any $x \in V - W$ the subgraph $H' = (W - \{x\}.F')$ is not a complete graph, then G is named K_p-****maximal complete graph****.*

Definition 1.10 *Let $G = (V, E)$ be a graph. If any component in G is a minimal complete graph then G is named* ***component minimal complete graph****.*

Definition 1.11 *Let $G = (V, E)$ be a graph and $x \in V$. If we consider the set $C = \{y \in V | \{x, y\} \in E\}$ then the number $\omega(x) = |C|$ is named the* ***degree*** *of x.*

Let CMC_G designate the component minimal complete subgraph for the graph G, if such a subgraph exists. If G has not CMC_G we write

$$CMC_G = \emptyset.$$

Proposition 1.3 *Let $G = (V, E)$ be a graph and*

$$X = \{x \in V | \omega(x) = 0\}$$

with $V - X \neq \oslash$. Let consider the subgraph $H = (V - X, F)$ of G. Then, G has a component minimal complete subgraph if and only if H has a component minimal complete subgraph.
In addition, we have

$$CMC_G = CMC_H.$$

Proposition 1.4 *Let $G = (V, E)$ be a connected graph and*

$$X = \{x \in V | \omega(x) = 1\}.$$

Let us consider the subgraph $H = (V - X, F)$ of G. Then, G gas a component minimal complete subgraph if and only if H has a component minimal complete subgraph and

$$CMC_G = CMC_H.$$

.

Proposition 1.5 *Let $G = (V, E)$ be a graph and let $CMC_G = (U, F)$ be the component minimal complete subgraph for G. Then for any $x \in U$ in G we have $\omega(x) \geq 2$.*

Proposition 1.6 *Let $G = (V, E)$ be a graph. If G ia a tree then $CMC_G = \emptyset$.*

Proposition 1.6 is not presented in [Bârză 2012] but it is obvious because if G is a tree, it means that G does not contain loops, and so G does not contains neither loops formed with three edges. It results that G has not CMC_G.

Proposition 1.7 *Let $G = (V, E)$ be a graph formed with the components $G_1, G_2, \ldots, G_k$. Then*

$$CMC_G = \bigcup_{i=1}^{k} CMC_{G_i}.$$

2. **Existing algorithms**

In [Bârză, 2013] I have presented two algorithms for the determination of component minimal complete subgraph, namely, an algorithm for the determination of CMC_G without eliminations, and an algorithm for the determination of CMC_G with vertexes and edges elimination. In this paragraph we remember these algorithms.

Let $G = (V, E)$ be a graph $k = |E|$ and so, $E = \{m_1, m_2, \ldots, m_k\}$. We have considered the algorithm for determination of CMC_G without vertexes and edges elimination which it is specified as the following process:

Step 1. Let $U = \oslash$, $F = \oslash$ and $i = 1$.
Step 2. If $i = k-2$ then STOP otherwise continue.
Step 3. Let $j = i + 1$.
Step 4. If $j = k - 1$ then $i \leftarrow i + 1$ and go to Step 2, otherwise continue.
Step 5. If $m_i \cap m_j = \oslash$ then go to Step 9, otherwise continue.
Step 6. Let $p = (m_i \cup m_j) \backslash (m_i \cap m_j)$.
Step 7. If $p \notin \{m_{j+1}, \ldots, m_k\}$ then go to Step 9, otherwise continue.
Step 8. Let $U \leftarrow U \cup m_i \cup m_j$ and $F \leftarrow F \cup \{m_i, m_j, p\}$.
Step 9. Let $j \leftarrow j + 1$ and go to Step 4.

For this algorithm we have shown that:

Theorem 2.1. *Let $G = (V, E)$ be a graph with $|E| = k$. Then by applying the algorithm given above we obtain CMC_G and the algorithm has the*

complexity $O(k^3)$.

The second algorithm I have presented in [Bârză, 2013], the algorithm for the determination of CMC_G with vertexes and edges elimination, it was given by the following process applied to $H = (X, T)$ as original given graph:

Step 1. Let $Y = \{x \in X | \omega(x) = 0\} \cup \{x \in X | \omega(x) = 1\}$.
Step 2. If $Y = \oslash$ then go to Step 4, otherwise continue.
Step 3. Let $X \leftarrow X/Y$, $T \leftarrow T/\{m \in T | m \cap Y \neq \oslash\}$ and go to Step 1.
Step 4. Let $V = X$ $E = T$, $k = |E|$ and the graph $G = (V, E)$ for which we consider that $E = \{m_1, m_2, \ldots, m_k\}$.
Step 5. Let $U = \oslash$, $F = \oslash$ and $i = 1$.
Step 6. If $i = k-2$ then STOP otherwise continue.
Step 7. Let $j = i + 1$.
Step 8. If $j = k - 1$ then $i \leftarrow i+1$ and go to Step 2, otherwise continue.
Step 9. If $m_i \cap m_j = \oslash$ then go to Step 9, otherwise continue.
Step 10. Let $p = (m_i \cup m_j) \backslash (m_i \cap m_j)$.
Step 11. If $p \notin \{m_{j+1}, \ldots, m_k\}$ then go to Step 9, otherwise continue.
Step 12. Let $U \leftarrow U \cup m_i \cup m_j$ and $F \leftarrow F \cup \{m_i, m_j, p\}$.
Step 13. Let $j \leftarrow j + 1$ and go to Step 4.

For this algortihm I have shown that:

Theorem 2.2. *Let $H = (X, T)$ be a graph. Then by applying the algorithm given above we obtain CMC_H and the algorithm has the complexity $O(k^3)$, where $k \leq |T| - |\{x \in X | \omega(x) = 1\}|$ and it represents the number of remaining edges after the elimination process made in the first three steps of the algorithm.*

I have shown also that the complexity for this algortihm is better than the complexity for the algorithm for the determination of CMC_G if in elimination process there exist erased edges, and that if no edges are eliminated, then the two algorithms had the same complexity.

3. Algorithm for the determination of CMC_G working on components

In this section we wish to specify a process for the determination of CMC_G which to be faster than the *algorithm for the determination of* CMC_G *with elimination*. To speed up the existing algorithm we have to consider the result indicated by proposition 1.7 specified in first section of this paper which are firstly given in [Bârzã, 2012]. It is also important where we introduce working on component process to avoid repeating operations which are not neccessary.

Firstly, we wish to analyze the elimination process from the second algorithm presented in section 2.

In the first step we detect the vertexes x with $\omega_G(x) = 0$ (isolated vertexes) and vertexes x with $\omega_G(x) = 1$, and we eliminate this vertexes. So, mainly, in the first application of steps 1, 2 and 3 we eliminate the component of G formed with one vertex.

The question to be asked is when, for the resulting graph G' we obtain new vertex x with $\omega_{G'}(x) = 0$. Because x remains as vertex in G', it means that, in an application of steps 1, 2 and 3, there exist the vertexes $y_1, y_2, \ldots, y_s$ which are eliminated because $\omega(y_i) = 1$ $(1 \leq i \leq s)$ and for any i $(1 \leq i \leq s)$ $\{x, y_i\}$ is an edge in G.

Continuing this analysis, in a previous application of steps 1, 2 and 3, any of the vertexes y_i $(1 \leq i \leq s)$ are not eliminated because $\omega(y_i) \geq 2$ and so there exist the vertexes $z_{i1}, z_{i2}, \ldots, z_{it_i}$ with $\omega(z_{ij}) = 1$ $(1 \leq j \leq t_i)$ and $\{y_i, z_{ij}\}$ is an edge in G for any j $(1 \leq j \leq t_i)$ and so y_i does not belong to a loop in G.

When we return to G in this manner, we obtain a component of G (and so a connected subgraph G_i of G) which does not contain loops. In this way we identify a component G_i of G which is a tree (see definition 1.6) and we can eliminate this component using proposition 1.2.

This analyze indicates us to separate the elimination process so that, firstly to eliminate only components of G formed with isolated vertexes.

After the above specification it results that the first step of the new algorithm is the elimination of isolated vertexes and then we must initialize the sets for storing CMC and the working sets. We also consider that is not useful to determine all the components of G, but to find the component one by one and then, to already determined component, to apply the rest of the algorithm if component is not a tree.

In this approach the algorithm we propose has the following form:

Step 1. Let $Y \leftarrow \{x \in V | \omega_G(x) = 0\}$. Let $V \leftarrow V - Y$.
Step 2. Let $U \leftarrow \emptyset$, $F \leftarrow \emptyset$, $T \leftarrow \emptyset$, $A \leftarrow V$, and $X \leftarrow E$.
Step 3. If $A = \emptyset$ then STOP; $CMC_G = (U, F)$, otherwise continue.
Step 4. Take $x \in A$, let $B \leftarrow \{x\}$, $C \leftarrow \emptyset$, and $H \leftarrow \emptyset$.
Step 5. If $B = \emptyset$ then goto Step 9, otherwise continue.
Step 6. Take $x \in B$.
Step 7. For any $m \in X$, if $x \in m$ then let $X \leftarrow X - \{m\}$, $H \leftarrow H \bigcup \{m\}$ and $B \leftarrow B \bigcup m$, otherwise continue.
Step 8. Let $B \leftarrow B - \{x\}$, $C \leftarrow C \bigcup \{x\}$ and goto Step 5.
Step 9. Let $A \leftarrow A - C$.
Step 10. If $|H| = |C| - 1$ then goto Step 3, otherwise continue.
Step 11. Let $Y \leftarrow \left\{x \in C | \omega_{(C,H)}(x) = 1\right\}$.
Step 12. If $Y = \emptyset$ then goto Step 14 otherwise continue.
Step 13. Let $Z \leftarrow \{m \in H | m \bigcap Y \neq \emptyset\}$. Let $H \leftarrow H - Z$, $C \leftarrow C - Y$ and goto Step 11.
Step 14. Consider that $H = \{m_1, m_2, \ldots, m_k\}$. Let $i \leftarrow 1$.
Step 15. If $i = k - 1$ then goto Step 3, otherwise continue.
Step 16. Let $j \leftarrow i + 1$
Step 17. If $j = k$ then $i \leftarrow i + 1$ and goto Step 15, otherwise continue.

Step 18. If $m_i \bigcap m_j = \emptyset$ then goto Step 22, otherwise continue.

Step 19. Let $p \leftarrow (m_i \bigcup m_j) - (m_i \bigcap m_j)$

Step 20. If $p \notin \{m_{j+1}, \ldots, m_m\}$ then goto Step 22, otherwise continue.

Step 21. Let $U \leftarrow U \bigcup m_i \bigcup m_j$, $F \leftarrow F \bigcup \{m_i, m_j, p\}$ and $T \leftarrow T \bigcup \{m_i \bigcup m_j\}$.

Step 22. Let $j \leftarrow j + 1$ and goto Step 17.

In Step 1 we eliminate from G all isolated vertexes.

In Step 2 we initialize the sets U and F used to identify CMC_G ($CMC_G = (U, F)$), the set T to store loops with three vertexes from G

$$T = \{\{x, y, z\} \subset V | \{\{x, y\}, \{x, z\}, \{y, z\}\} \subset E\},$$

and working sets A and X.

Step 3 represents the test for terminating the algorithm which it ends when we finish to process all component of G.

Steps from 4 to 9 are used to determinine the next non used component of G. This component forms the subgraph (C, H)

Step 10 verifies if current component is a tree in which case looking for loops with three edges is without sense (it can not exist such a loop).

Steps from 11 to 14 realize repeatly the elimininations of vertexes with degree equal with 1 (and the corresponding edges). In fact, the part of component which is eliminated in this subprocess represent itself a subgraph which is a tree. The rest o the steps will be so applied to the part in which every vertex belongs to a loop.

The steps from 14 to 22 determine all loops with three edges from current component of G and this is the part of algorithm which is determined for the algorithm complexity.

It is obvous that, if the graph G has only one component (is a connected graph) and this component is not a tree, then the new algorithm has the same complexity as *the algorithm for the determination of CMC_G with elimination.*

To determine a formula for the complexity of the new algortihm we must consider that the graph G has the component $G_1, G_2, \ldots, G_p$. Now, let consider that $I = \{i_1, i_2, \ldots, i_t\}$ is the set of indexes so that for any $j \in I$ the component G_j contains loops. From the presentation of the algorithm results that the steps from 14 to 22 will be not applied to the component G_j for any $j \in \{1, 2, \ldots, p\} - I$.

Now we can consider that for any $j \in I$ the component G_j has, after the elimination from steps from 11 to 14, a set of edges with k_j elements so that the complexity related to G_j is $O(k_j^3)$ and so, the complexity of the new algorithm is the sum of the complexity when we apply the algorithm on the component of G.

On this basis we can give now the following result:

Theorem 3.1. *Let $G = (V, E)$ be a graph and $m = |E|$. We consider that the graph G is formed with p components $G_1, G_2, \ldots, G_p$ and for each component $G_i = (V_i, E_i)$, $(1 \leq i \leq p)$ the number of edges is $k_i = |E_i|$.*

The algorithm presented realizes the determination of CMC_G and has the complexity given by:

$$O(\sum_{1\leq i\leq p,\, k_i\neq 0,\, k_i\neq |V_i|-1} k_i^3).$$

In the future I will call the algorithm presented in section 3 as ***the component approach algorithm for the determination of*** CMC_G.

We already observe that if G is a connected graph, but not a tree, *the component approach algorithm for the determination of* CMC_G and *the algorithm for the determination of* CMC_G *with elimination* have the same complexity.

To compare the complexity in general case we will use the next results from algebra which appears as a consequence of the well known formula:

$$(x+y)^3 = x^3 + 3x^2y + 3xy^2 + y^3.$$

Based on this formula we can say that: *If x and y are two non negative numbers then*

$$x^3 + y^3 \leq (x+y)^3.$$

Generalizing the above result we can write that: *If $x_1, x_2, \ldots, x_n$ is a finite sequence on non negative numbers, then*

$$\sum_{i=1}^{n} x_i^3 \leq (\sum_{i=1}^{n} x_i)^3.$$

Because the numbers $k_1, k_2, \ldots, k_p$ represent a finite sequence of non negative integer, we can apply the last result to this sequence of numbers and so it results that

$$\sum_{i=1}^{p} k_i^3 \leq (\sum_{i=1}^{p} k_i)^3.$$

The numbers $k_1, k_2, \ldots, k_p$ are nu numbers of edges in components $G_1, G_2, \ldots, G_p$ ans so we have

$$m = \sum_{i=1}^{p} k_i.$$

We observe also that

$$\sum_{1\leq i\leq p,\, k_i\neq 0,\, k_i\neq |V_i|-1} k_i^3$$

is more restrictive then

$$\sum_{i=1}^{p} k_i^3$$

and so we can write that:

$$\sum_{1\leq i\leq p,\, k_i\neq 0,\, k_i\neq |V_i|-1} k_i^3 \leq \sum_{i=1}^{p} k_i^3.$$

In conclusion we have

$$\sum_{1\leq i\leq p,\, k_i\neq 0,\, k_i\neq |V_i|-1} k_i^3 \leq m^3$$

which shows that *the component approach algorithm for the determination of* CMC_G is a better algorithm than *the algorithm for the determination of* CMC_G *with elimination* for which we already know that is a better algorithm than *the algorithm for the determination of* CMC_G *without elimination.*

References

1. Bamdy, J.A., Murty U.S.R., *Graph Theory*, "Springer-Verlag", 2007.
2. Bârză, S., and Morogan, L.M., *Algoritmica grafurilor*, "Editura Fundatiei România de Mâine", Bucureşti, 2008.
3. Bârză, S., *Some Consideration about Component Minimal Complete Subgraph*, Analele Universităţii Spiru Haret, Seria Matematică-Informatică, anul VII, nr. 2, Editura Fundaţiei România de Mâine, Bucureşti 2012.
4. Bârză, S., *Set Approach Algorithms for Determination of Component Minimal Complete Subgraph*, Analele Universităţii Spiru Haret, Seria Matematică-Informatică, anul VIII, nr. 1, pp. 65-71, Editura Fundaţiei România de Mâine, Bucureşti 2013.
5. Berge, C., *Graph Theory and Application*, romanian edition, "Editura Tehnicâ", Bucureşti, 1971.
6. Popescu, D.R., *Combinatorică şi teoria grafurilor*, "Editura Societăţii de Ştiinţe Matematice din România", Bucureşti, 2005.
7. Tomescu, I., *Combinatorică şi teoria grafurilor*, "Editura Universităţii Bucureşti", Bucureşti, 1990.