The algorithms in graph theory —cyclic vertex (edge) connectivity

Jun Liang

2017. 11. 13

Outline

1. Some algorithms in graph theory

2. The cyclic vertex (edge) connectivity

3. Adaboost

Some algorithms

- The shortest path algorithms——Dijkstra （O（n^2））， Bellman-Ford（ Negative weight edge ） and Floyd （O（n^3））

  The time complexity——Dijkstra

  (1) Adjacency matrix $O(n^2)$

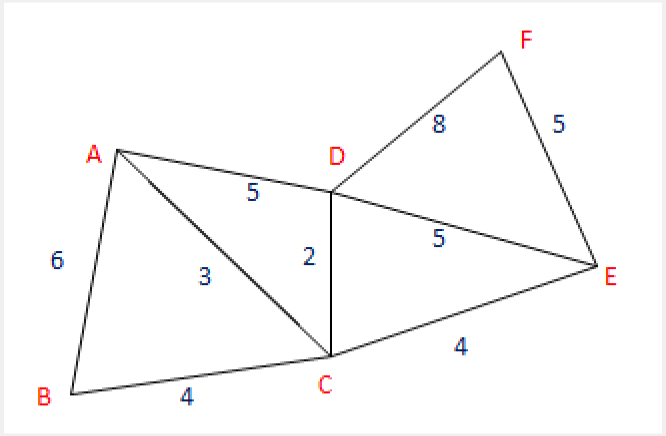
  (2) Adjacency list and binary heap $O((m+n)\log_2 n)$

  (3) Adjacency list and Fibonacci heap $O(m+n\log_2 n)$

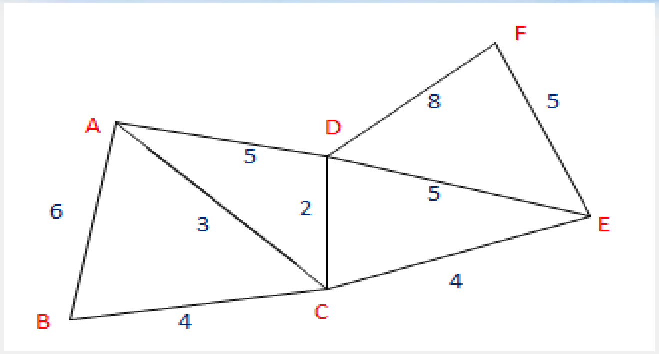
- The Minimum spanning tree algorithms——prim (Adjacency matrix :$O(n^2)$) Adjacency list: $O(m\log_2 n)$)， kruskal and Sollin(Boruvka)

- The matching algorithm - Hungarian algorithm
- The network flow algorithm -———Ford-Fulkerson
- The algorithms on other aspects such as connectivity, coloring, traversal, Clique etc.

An example for Dijkstra：



The key point: the shortest path from the source v to any vertex in S is not greater than the shortest path from v to any vertex in U.



Step	Set S	Set U
1	S={A}	U={B，C，D，E，F} Update the distances from A to the vertices in U A→B＝6　A→C＝3　A→D＝5 A→other vertices=∞ Put the vertex C with the lowest weight into the set S.
2	S={A，C}	U={B，D，E，F} Update the distances from A to the vertices in U A→B＝6　A→D＝5　A→E＝7 A→F=∞ Put the vertex D with the lowest weight into the set S.
3	S={A，C，D}	U={B，E，F} Update the distances from A to the vertices in U A→B＝6　A→E＝7　A→F＝13 Put the vertex B with the lowest weight into the set S.

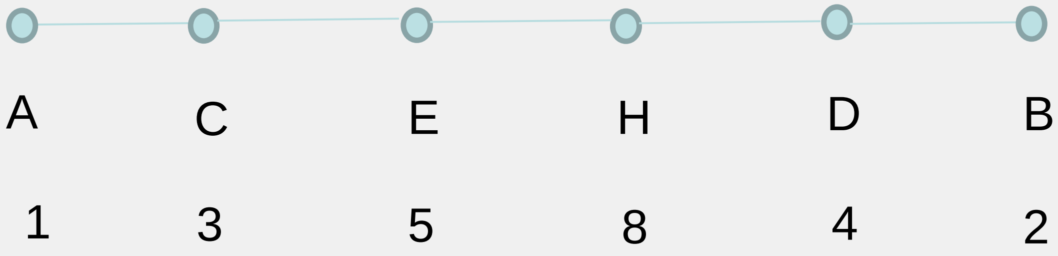
Setp	Set S	Set U
4	S={A，B，C，D}	U={E，F} Update the distances from A to the vertices in U A→E＝７　A→F＝13 Put the vertex E with the lowest weight into the set S.
5	S={A，B，C，D，E}	U={F} Update the distances from A to the vertices in U A→F＝12 Put the vertex F with the lowest weight into the set S.
6	S={A，B，C，D，E，F}	The set U is empty and the algorithm is over.

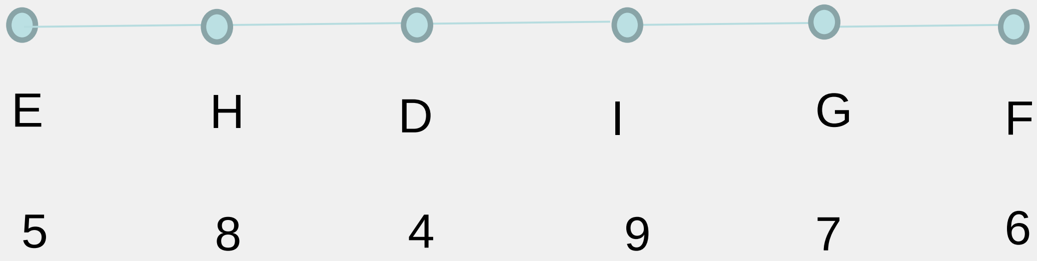
Floyd's Algorithm: For each pair of vertices u and v in G, we see if there is a vertex w such that the sum of the distance from u to w and from w to v is shorter than the distance from u to v.

```
for(k=0;k<n; k++)
{
   for(i=0;i<n;i++)
          for(j=0;j<n;j++)
            if(D[i][j]>(D[i][k]+D[k][j]))
            {
                    D[i][j]=D[i][k]+D[k][j];
            }
}
```

Some shortest paths

- A——B：
- E——F





Introduction to cyclic vertex (edge) connectivity

- The concept of cyclic connectivity was proposed by Tait in 1880.
- It has appeared in some theories developed for solving the Four Color Conjecture.
- The vertex connectivity and edge connectivity in graph theory are often used to measure network reliability.
- The cyclic vertex (edge) connectivity is a kind of conditional connectivity.

The problem on a polynomial time algorithm for determining the cyclic vertex (edge) connectivity of a graph has been standing for many years.

The polynomial algorithms for determining the cyclic edge connectivity of cubic graphs and *k*-regular graphs had not been solved until a few years ago.

It is not known yet whether the problem to determine the cyclic vertex connectivity of a graph is a P-problem.

Some results for cyclic edge connectivity

- In 2004， Dvorak et.al. gave an algorithm for cyclic edge connectivity of cubic graphs, and the time complexity of their algorithm was $O(v^2 \log_2 v)$.

- In 2005， Lou and Wang gave an algorithm determining the cyclic edge connectivity of k-regular graphs, then the time complexity of the algorithm was improved to $O(k^9v^6)$ by Lou and Liang in 2014.

- In 2008, Lou and Wang charactered all graphs with infinite cyclic edge connectivity. ($O(|V||E|)$)

The cyclic edge connectivity of planar graphs

Plummer, 1972

A planar 5-connected graph has a cyclic edge connectivity of at most 13, while the planar 4-connected graph has a cyclic edge connectivity of any integer greater than or equal to 4.

In 2009, Lu Y gave an algorithm for cyclic edge connectivity of planar graphs.($O(|V|^4)$)

Some results for cyclic vertex connectivity

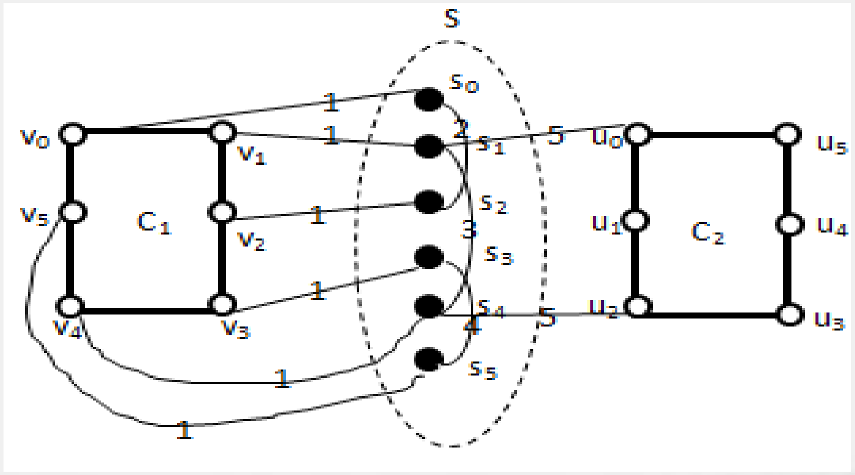
- In 2016, we gave a polynomial algorithm for cyclic vertex connectivity of cubic graphs. $O(v^{15/2})$

- In 2017, we gave a random algorithm for cyclic vertex connectivity of some graphs satisfying some conditions.

Terminology:

A set S of vertices (edges) in G is a cyclic vertex (edge) cutset if G − S is not connected and at least two components contain a cycle respectively.

The cyclic vertex (edge) connectivity ck(G) is the cardinality of a minimum cyclic vertex (edge)cutset in G. If no cyclic vertex (edge) cutset exists, ck(G) (c(λ)) is defined to be ∞.



Proposition 1 ([1]) If k(G) = ck(G), then ck(G)$\leq \delta$ (G) and ck(G)$\neq \infty$. Conversely, if G is a simple graph such that ck(G)$\neq \infty$ and ck(G)$< \delta$ (G) then k(G) = ck(G).

Proposition 2 ([1]) (i) If p is a positive integer, the is a simple graph G' such that ck(G') = ∞ and cλ(G') = p. (ii) If p$\leq$q are positive integers, there is a simple graph H' such that ck(H') = p and cλ(H') = q. (iii) If p $\leq$ q are positive integers, there is a simple graph F' such that ck(F') = q and cλ(F') = p.

Theorem 3 Let G be a connected k-regular graph with girth g. If v(G)$\geq$2g(k-1), then ck(G)$\leq$(k-2)g.

Theorem 4 Let G be a k-regular graph with girth g $\geq$ 7, suppose that C is an induced cycle in a connected component of G, then |N(C)| $\geq$ (k-2)g.

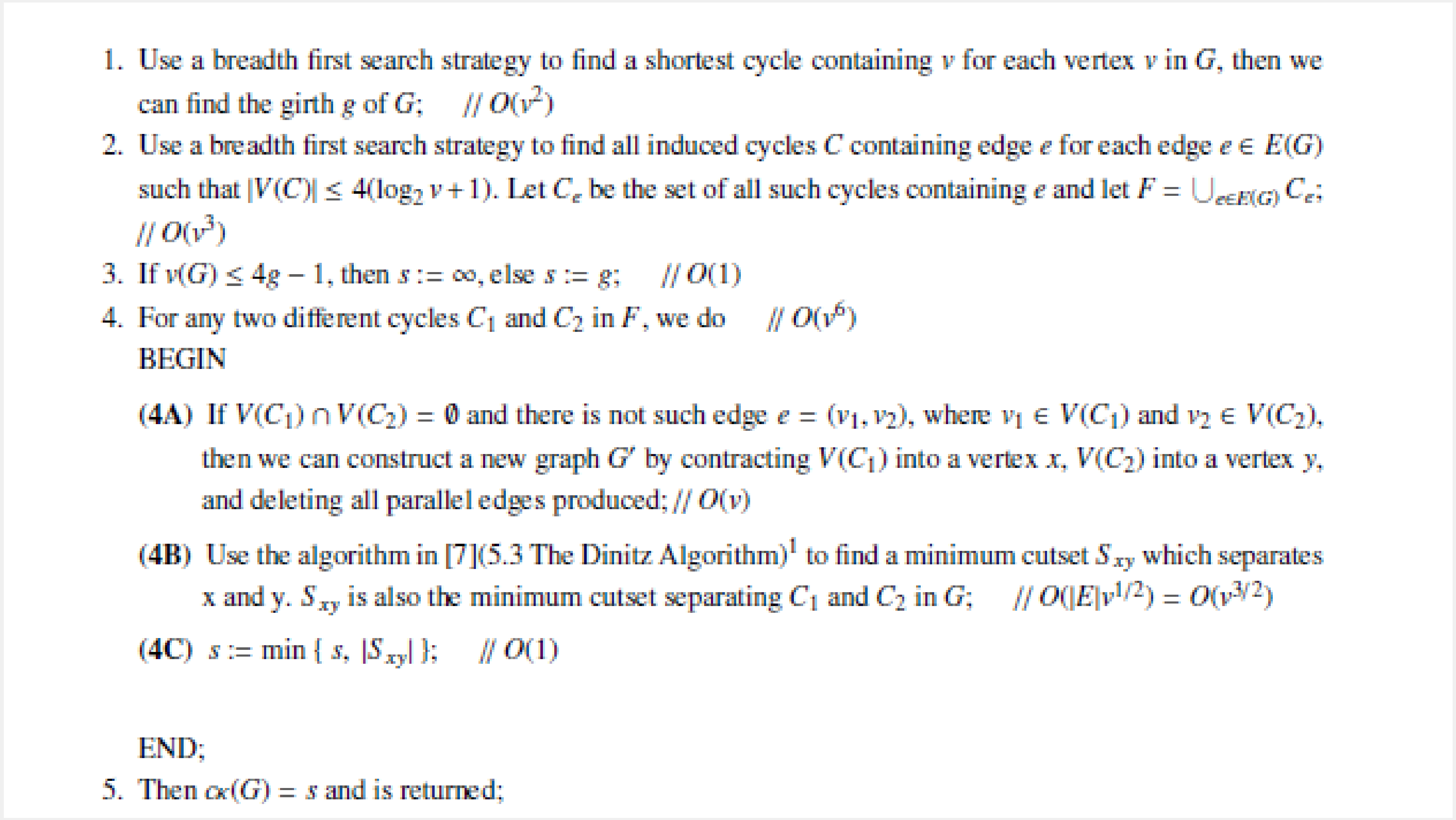
[1] Peroche B. On several sorts of connectivity[J]. Discrete mathematics, 1983, 46(3): 267-277.

Cubic graphs

The main idea of algorithm of cubic graphs is that:

we firstly find all induced cycles of length less than or equal to $4(\log_2 v + 1)$, and put these cycles into a set F. Then for any two disjoint cycles in F, we find a vertex cutset to separate them. Then the minimum vertex cutset is the minimum cyclic vertex cutset, and the cardinality of the minimum cyclic vertex cutset is cyclic vertex connectivity ck(G). In Algorithm 1, the symbol s denotes the initial value of ck(G).

Algorithm 1:



```
1. Use a breadth first search strategy to find a shortest cycle containing $v$ for each vertex $v$ in $G$, then we
   can find the girth $g$ of $G$;   // $O(v^2)$
2. Use a breadth first search strategy to find all induced cycles $C$ containing edge $e$ for each edge $e \in E(G)$
   such that $|V(C)| \le 4(\log_2 v + 1)$. Let $C_e$ be the set of all such cycles containing $e$ and let $F = \bigcup_{e \in E(G)} C_e$;
   // $O(v^3)$
3. If $v(G) \le 4g - 1$, then $s := \infty$, else $s := g$;   // $O(1)$
4. For any two different cycles $C_1$ and $C_2$ in $F$, we do   // $O(v^6)$
   BEGIN
   (4A) If $V(C_1) \cap V(C_2) = \emptyset$ and there is not such edge $e = (v_1, v_2)$, where $v_1 \in V(C_1)$ and $v_2 \in V(C_2)$,
        then we can construct a new graph $G'$ by contracting $V(C_1)$ into a vertex $x$, $V(C_2)$ into a vertex $y$,
        and deleting all parallel edges produced; // $O(v)$
   (4B) Use the algorithm in [7](5.3 The Dinitz Algorithm)[1] to find a minimum cutset $S_{xy}$ which separates
        x and y. $S_{xy}$ is also the minimum cutset separating $C_1$ and $C_2$ in $G$;   // $O(|E|v^{1/2}) = O(v^{3/2})$
   (4C) $s := \min\{ s, |S_{xy}| \}$;   // $O(1)$

   END;
5. Then $c\kappa(G) = s$ and is returned;
```

The notations $N_r(a_i)$ and T (ai) (T-tree) are defined as:

$N_0(a_i) = \{a_i\}$， $N_1(a_i) = \{u | ua_i \in E(D_s)\}$， 和 $N_r(a_i) = \{u |$ $\exists u_1 \in N_{r-1}(a_i)$， $u \notin \bigcup_{j=0}^{r-1} N_j(ai)$， $u_1 \notin S$， $uu_1 \in E(D_s)\}$。

$$T(a_i) = G[\bigcup_{r=0}^{c/4-1} N_r(a_r)] - E_s - E_{T_i} \ (0 \le i \le c-1).$$

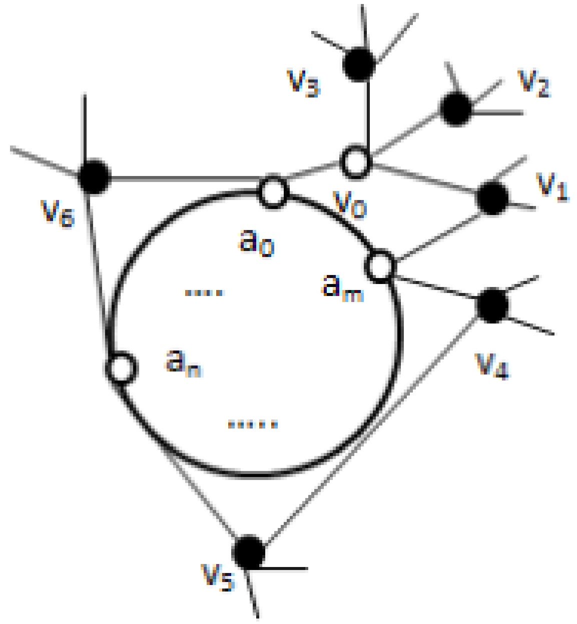
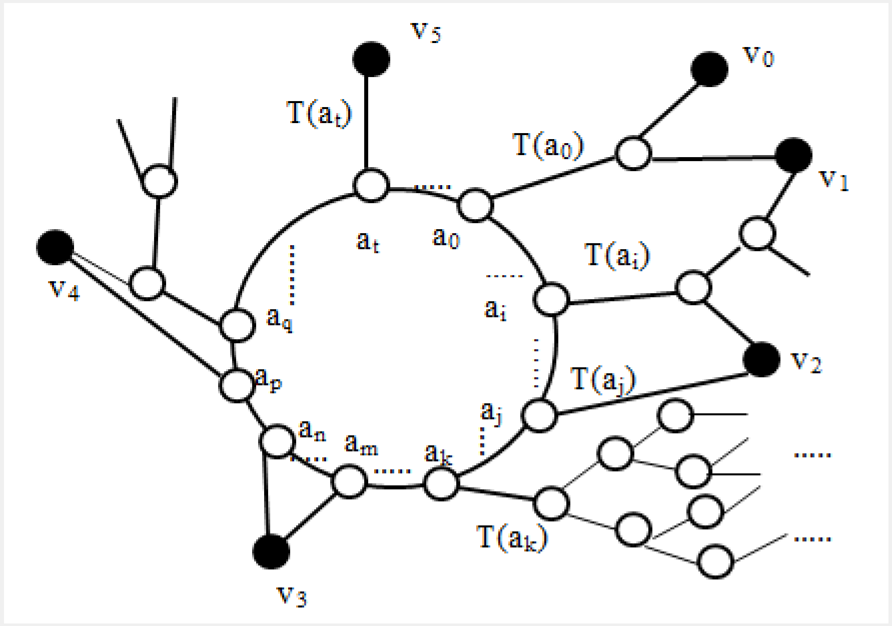


图 2-1 一个 $N_r(a_i)$ 的例子

$N_1(a_0) = \{v_0，v_6\}$，$N_2(a_0) = \{v_1，v_2，v_3\}$，$N_1(a_m)=\{v_1，v_4\}$，$N_2(a_m) = \emptyset$，$N_1(a_n) = \{v_5，v_6\}$，和$N_2(a_n) = \emptyset$。

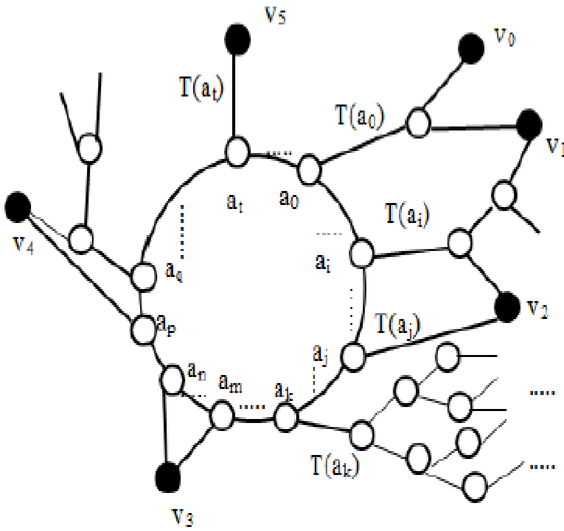
Let $G[\bigcup_{r=0}^{c/4-1} N_r(a_i)]$ be an induced subgraph of G. Let E_s be the set of edges whose ends are both in S. For any $v_0 \in N_{r_1}(a_i), v_1 \in N_{r_2}(a_i)$ $(r_1 \le r_2)$ and $v_0 \ne v_1$, if $v_2 \in N_{r_1+1}(a_i) \cap S, r_1 < r_2$, and $v_2v_1, v_2v_0 \in E(G)$, then we put the edge v_2v_1 into a set E_{T_i} of edges ; if $v_2 \in N_{r_1+1}(a_i) \cap S, r_1 = r_2$, and $v_2v_1, v_2v_0 \in E(G)$, then either the edge v_2v_1 or v_2v_0 belongs to the set E_{T_i} of edges. Then $T(a_i) = G[\bigcup_{r=0}^{c/4-1} N_r(a_i)] - E_s - E_{T_i}$ is a subgraph of G. Then each vertex $v_2 \in N_{r_1+1}(a_i) \cap S$ is adjacent to exactly one $v_0 \in N_{r_1}(a_i)$ such that r_1 is as small as possible. Since C_1 is a minimum cycle in D_1, $T(a_i)$ is a tree called *T-tree*.

Suppose a tree has $2^0 + 2^0 + 2^1 + 2^2 + \cdots + 2^{c/4-2} = 2^{c/4-1}$ vertices and is fully contained in D_1, then the tree is called a *full-tree*. Suppose a tree has $2^0 + 2^0 + 2^0 + 2^1 + 2^2 + \cdots + 2^{c/4-3} = 2^{c/4-2} + 1$ vertices contained in D_1, then the tree is called a *half-tree* .



$if\ v(G) \geq v(D_1) \geq 2^{c/4-1}$, then we have $c \leq 4(\log_2 v + 1)$.

If D_1 contains a full-tree or two vertex-disjoint half-trees, then we can get the conclusion of this theorem.



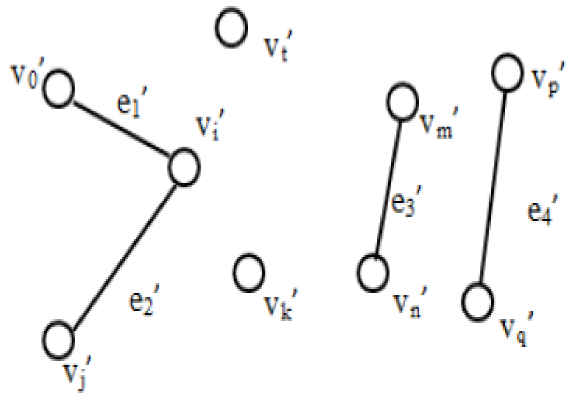
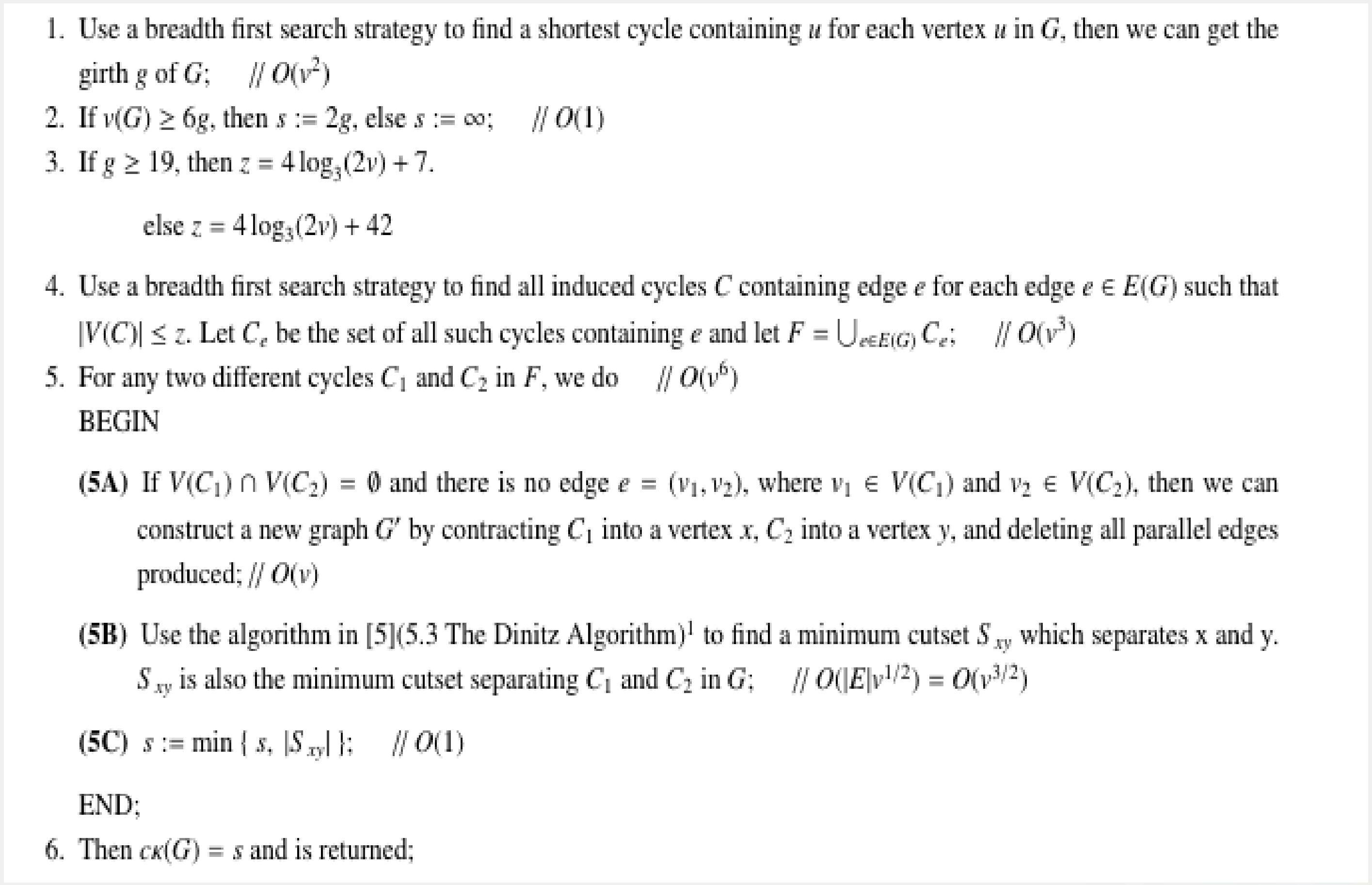


Figure 1: An example for a subgraph in $G[V(D_1) \cup S] - E_s$ Figure 2: A corresponding graph of Figure 1

K-regular graphs

Algorithm 2 (for 4-regular graphs):



```
1. Use a breadth first search strategy to find a shortest cycle containing $u$ for each vertex $u$ in $G$, then we can get the
   girth $g$ of $G$;    // $O(v^2)$
2. If $v(G) \geq 6g$, then $s := 2g$, else $s := \infty$;    // $O(1)$
3. If $g \geq 19$, then $z = 4\log_3(2v) + 7$.
       else $z = 4\log_3(2v) + 42$
4. Use a breadth first search strategy to find all induced cycles $C$ containing edge $e$ for each edge $e \in E(G)$ such that
   $|V(C)| \leq z$. Let $C_e$ be the set of all such cycles containing $e$ and let $F = \bigcup_{e \in E(G)} C_e$;    // $O(v^3)$
5. For any two different cycles $C_1$ and $C_2$ in $F$, we do    // $O(v^6)$
   BEGIN
   (5A) If $V(C_1) \cap V(C_2) = \emptyset$ and there is no edge $e = (v_1, v_2)$, where $v_1 \in V(C_1)$ and $v_2 \in V(C_2)$, then we can
        construct a new graph $G'$ by contracting $C_1$ into a vertex $x$, $C_2$ into a vertex $y$, and deleting all parallel edges
        produced; // $O(v)$
   (5B) Use the algorithm in [5](5.3 The Dinitz Algorithm)[1] to find a minimum cutset $S_{xy}$ which separates x and y.
        $S_{xy}$ is also the minimum cutset separating $C_1$ and $C_2$ in $G$;    // $O(|E|v^{1/2}) = O(v^{3/2})$
   (5C) $s := \min\{s, |S_{xy}|\}$;    // $O(1)$
   END;
6. Then $c\kappa(G) = s$ and is returned;
```

Algorithm 3 (for k-regular graphs (k$\geq$ 5)):

```
1. Use a breadth first search strategy to find a shortest cycle containing $v$ for each vertex $v$ in $G$, then we can find the girth $g$ of $G$;   // $O(kv^2)$
2. Use a breadth first search strategy to find all induced cycles $C$ containing edge $e$ for each $e \in E(G)$ such that $|V(C)| \leq 4\log_{k-1}[(k-2)v] + 9k^2$. Let $C_e$ be the set of all such cycles containing $e$ and let $F = \bigcup_{e \in E(G)} C_e$;   // $O(v^3 k^3 k^{9k^2/2})$
3. If $v(G) \geq 2g(k-1)$, then $s := (k-2)g$, else $s := \infty$;   // $O(1)$
4. For any two different cycles $C_1$ and $C_2$ in $F$ do   // $O(v^6 k^6 k^{9k^2})$
   BEGIN
   (4A) If $V(C_1) \cap V(C_2) = \emptyset$ and there is no such edge $e = (v_1, v_2)$, where $v_1 \in V(C_1)$ and $v_2 \in V(C_2)$, then we can construct a new graph $G'$ by contracting $V(C_1)$ into a vertex $x$, $V(C_2)$ into a vertex $y$, and deleting all parallel edges produced; // $O(v)$
   (4B) Use the algorithm in [4](5.3 The Dinitz Algorithm)[1] to find a minimum cutset $S_{xy}$ which separates x and y. $S_{xy}$ is also the minimum cutset separating $C_1$ and $C_2$ in $G$;   // $O(|E|v^{1/2}) = O(kv^{3/2})$
   (4C) $s := \min\{\, s,\ |S_{xy}|\, \}$;   // $O(1)$

   END;
5. Then $c\kappa(G) = s$ and is returned;
```

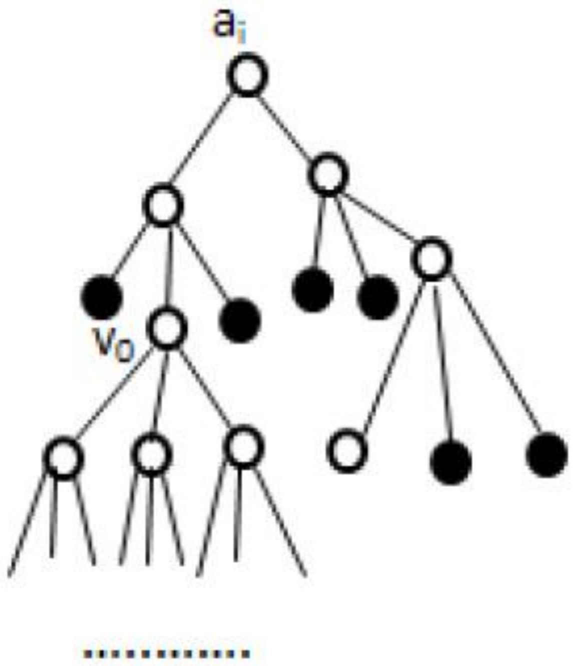


Figure 3: A T-tree $T(a_i)$.

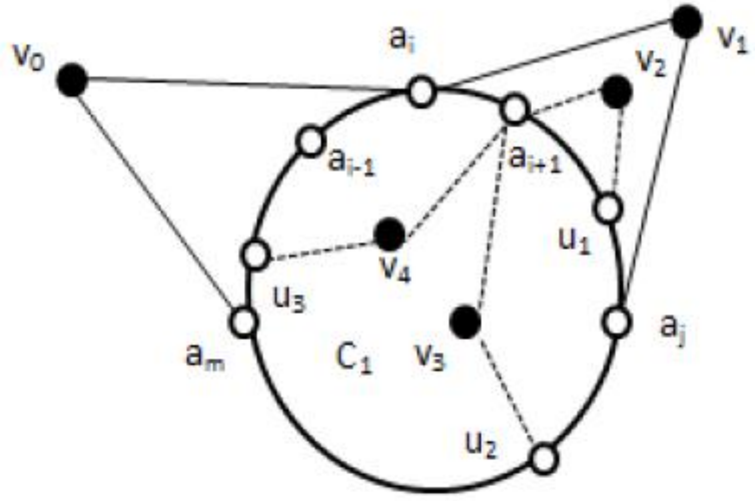
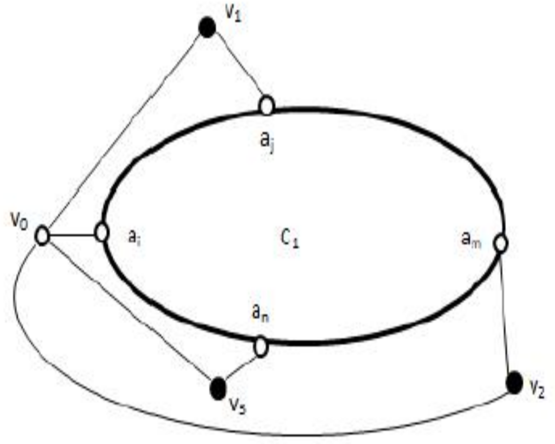
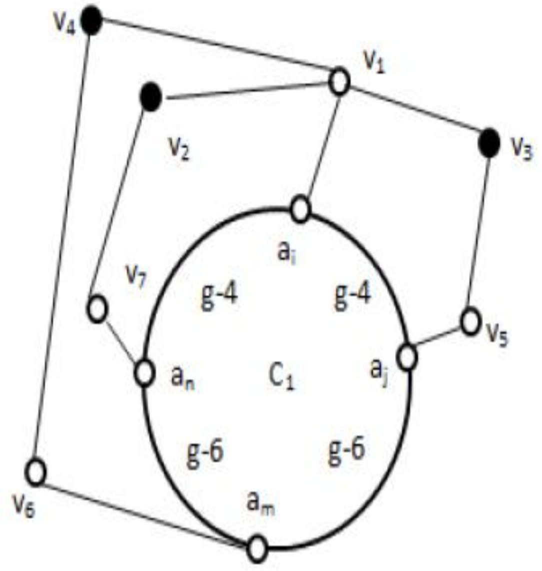


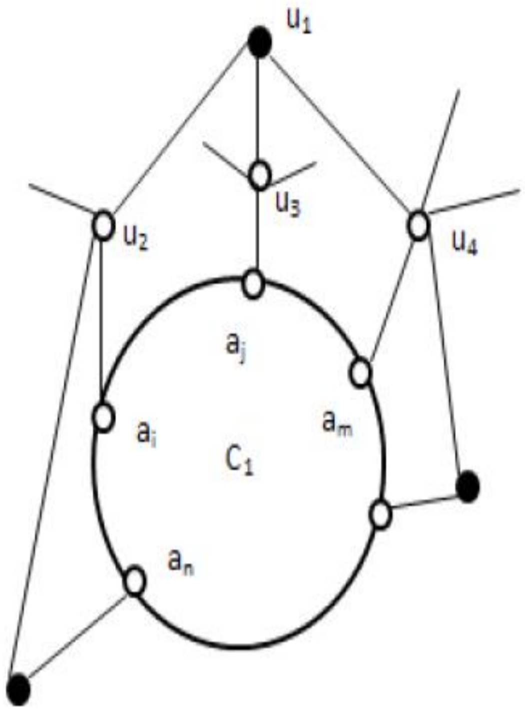
Figure 4: $3g - 8 \le c \le 3g - 2$.



(a) The first structure

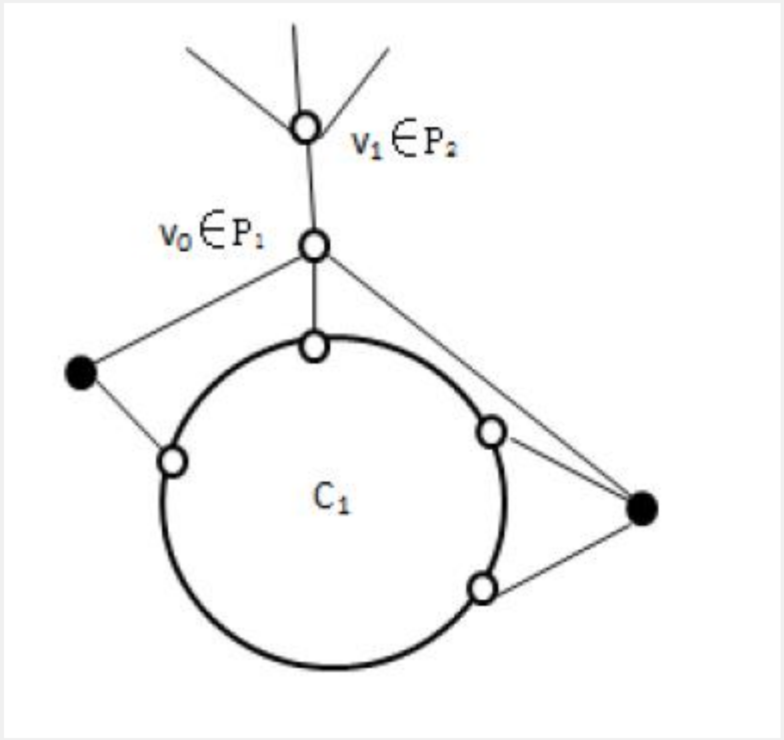


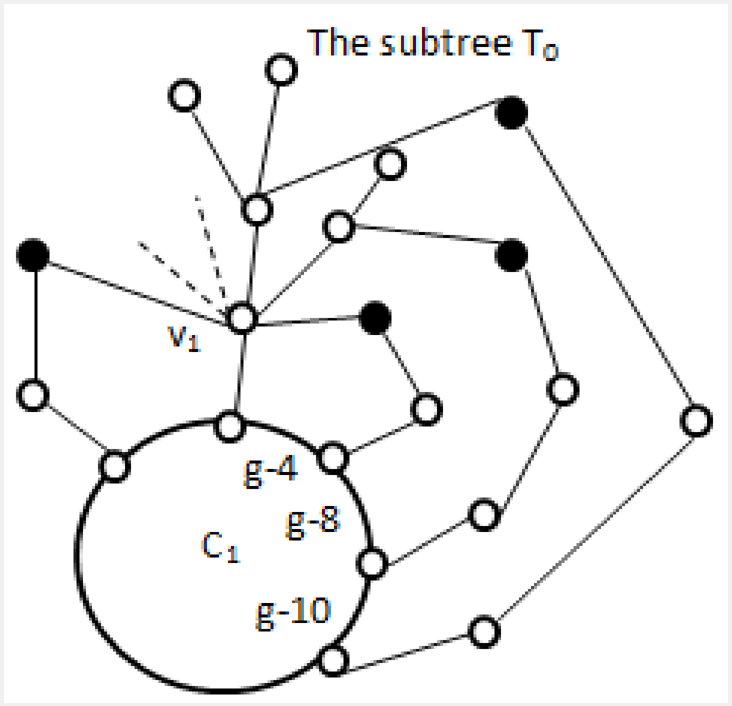
(b) The second structure



(c) The third structure

Figure 7: Three structures that do not exist for $c \leq 3g - 2$ and $g \geq 19$.


$v_1 \in P_2$
$v_0 \in P_1$
C_1



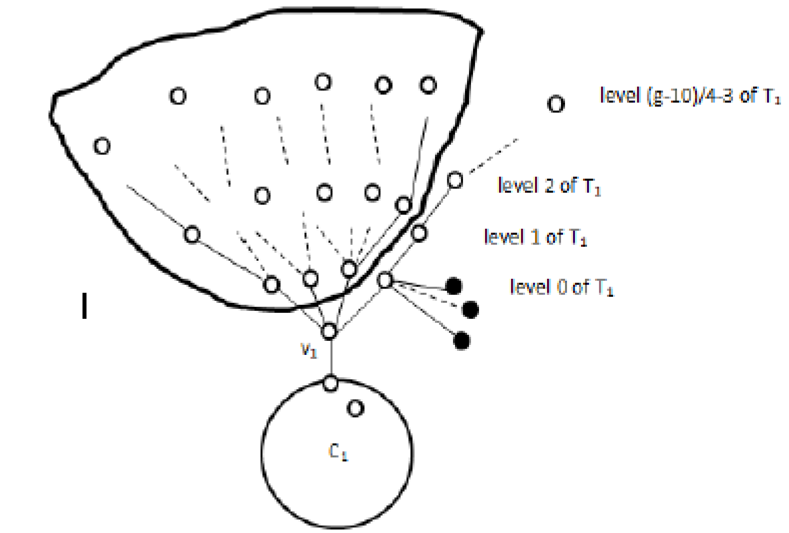
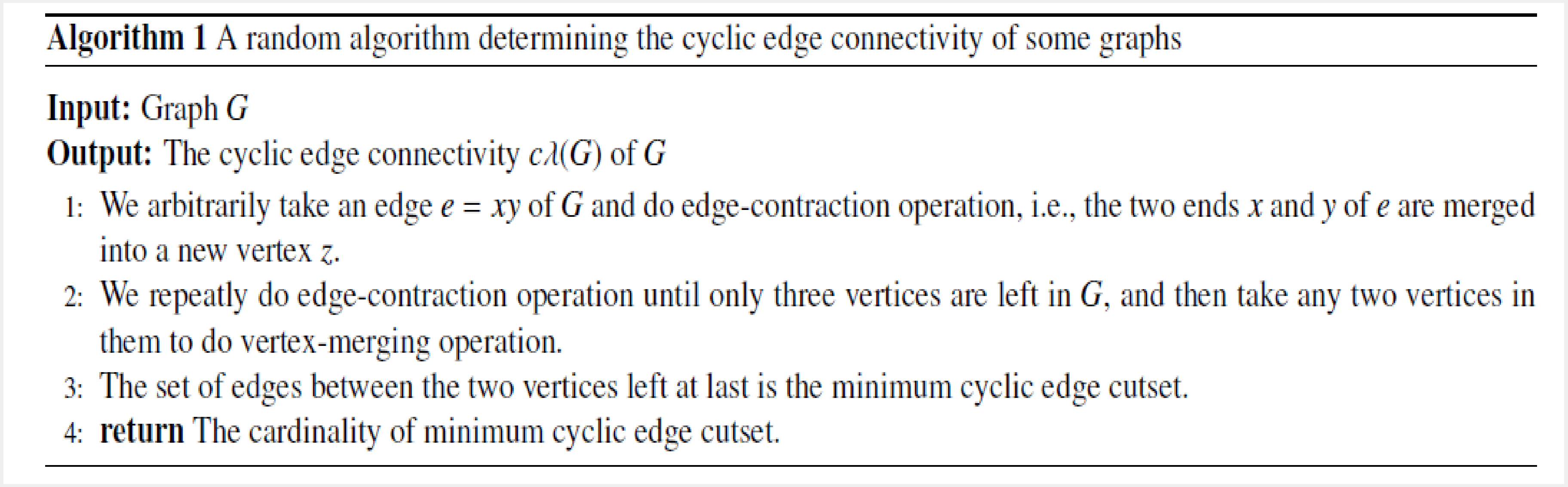


Figure 12: A similar structure.

Random Algorithms for cyclic edge connectivity of some graphs



Algorithm 1 A random algorithm determining the cyclic edge connectivity of some graphs

Input: Graph G
Output: The cyclic edge connectivity $c\lambda(G)$ of G

```
1: We arbitrarily take an edge $e = xy$ of $G$ and do edge-contraction operation, i.e., the two ends $x$ and $y$ of $e$ are merged
   into a new vertex $z$.
2: We repeatly do edge-contraction operation until only three vertices are left in $G$, and then take any two vertices in
   them to do vertex-merging operation.
3: The set of edges between the two vertices left at last is the minimum cyclic edge cutset.
4: return The cardinality of minimum cyclic edge cutset.
```

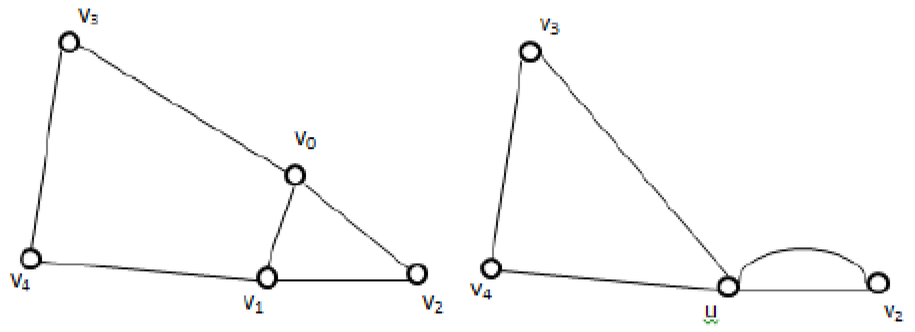
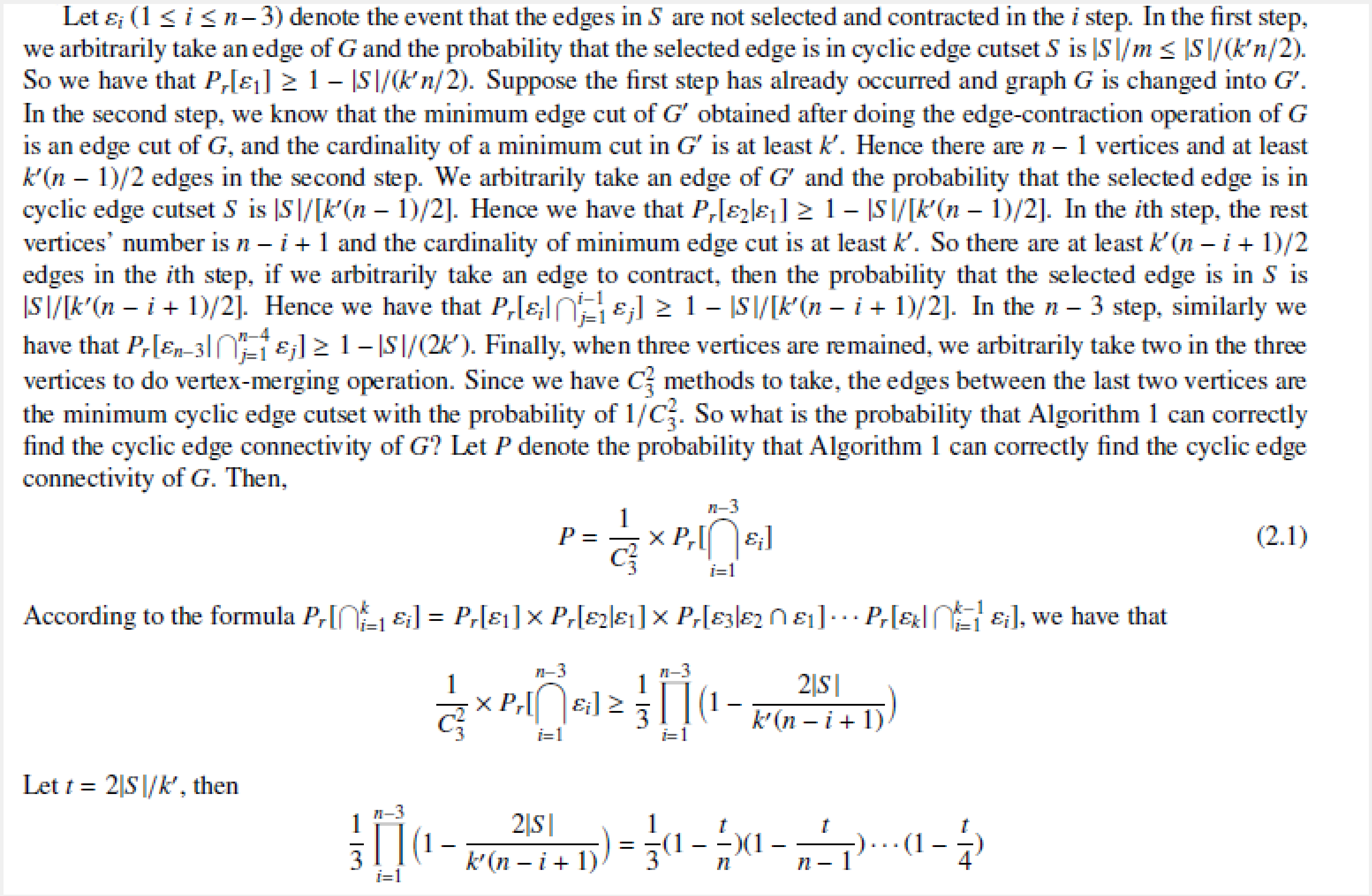


Figure 1: An example for the vertex-merging operation



Let ε_i $(1 \le i \le n-3)$ denote the event that the edges in S are not selected and contracted in the i step. In the first step, we arbitrarily take an edge of G and the probability that the selected edge is in cyclic edge cutset S is $|S|/m \le |S|/(k'n/2)$. So we have that $P_r[\varepsilon_1] \ge 1 - |S|/(k'n/2)$. Suppose the first step has already occurred and graph G is changed into G'. In the second step, we know that the minimum edge cut of G' obtained after doing the edge-contraction operation of G is an edge cut of G, and the cardinality of a minimum cut in G' is at least k'. Hence there are $n-1$ vertices and at least $k'(n-1)/2$ edges in the second step. We arbitrarily take an edge of G' and the probability that the selected edge is in cyclic edge cutset S is $|S|/[k'(n-1)/2]$. Hence we have that $P_r[\varepsilon_2|\varepsilon_1] \ge 1 - |S|/[k'(n-1)/2]$. In the ith step, the rest vertices' number is $n-i+1$ and the cardinality of minimum edge cut is at least k'. So there are at least $k'(n-i+1)/2$ edges in the ith step, if we arbitrarily take an edge to contract, then the probability that the selected edge is in S is $|S|/[k'(n-i+1)/2]$. Hence we have that $P_r[\varepsilon_i|\bigcap_{j=1}^{i-1}\varepsilon_j] \ge 1 - |S|/[k'(n-i+1)/2]$. In the $n-3$ step, similarly we have that $P_r[\varepsilon_{n-3}|\bigcap_{j=1}^{n-4}\varepsilon_j] \ge 1 - |S|/(2k')$. Finally, when three vertices are remained, we arbitrarily take two in the three vertices to do vertex-merging operation. Since we have C_3^2 methods to take, the edges between the last two vertices are the minimum cyclic edge cutset with the probability of $1/C_3^2$. So what is the probability that Algorithm 1 can correctly find the cyclic edge connectivity of G? Let P denote the probability that Algorithm 1 can correctly find the cyclic edge connectivity of G. Then,

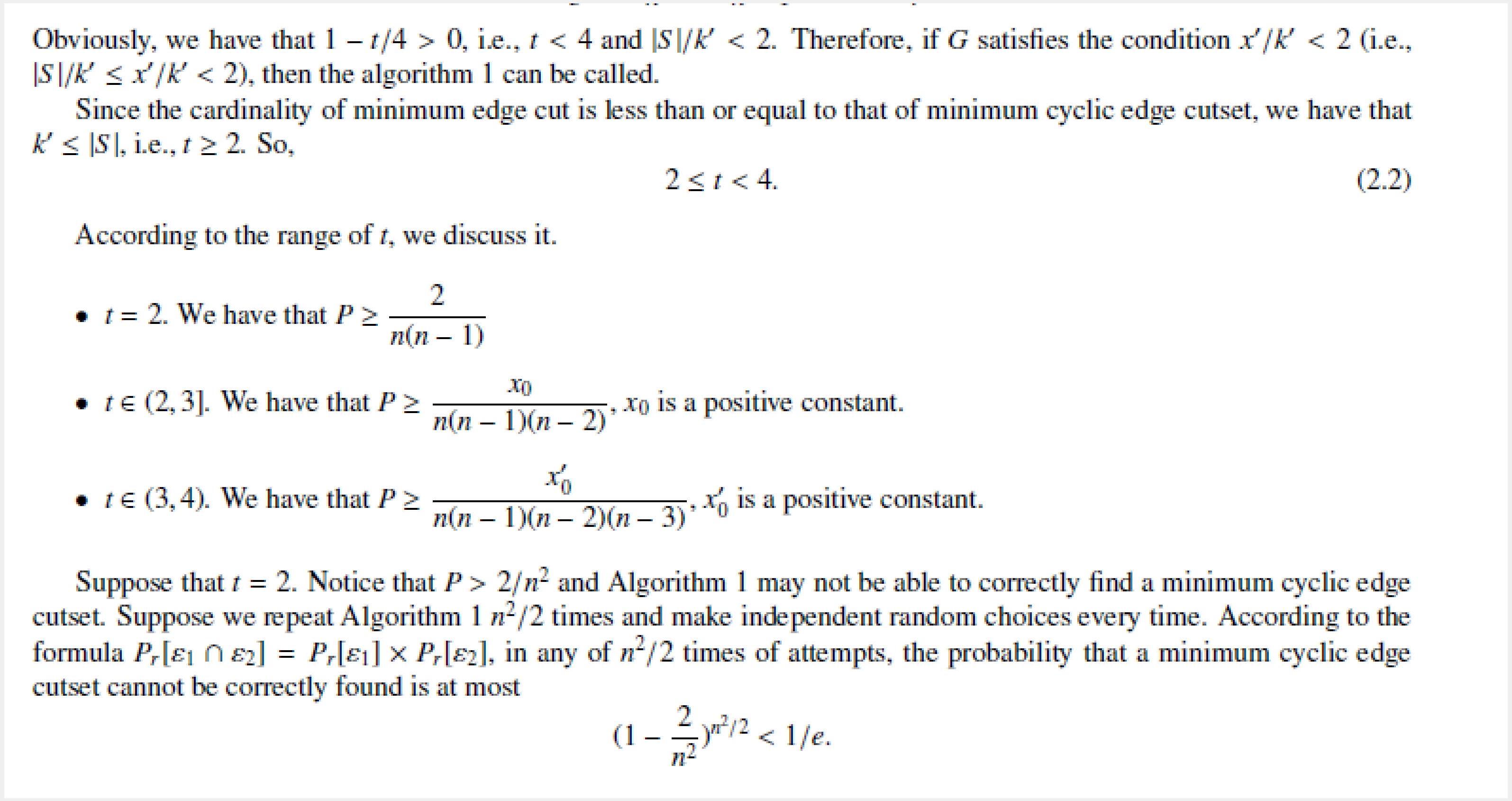
$$P = \frac{1}{C_3^2} \times P_r[\bigcap_{i=1}^{n-3}\varepsilon_i] \qquad (2.1)$$

According to the formula $P_r[\bigcap_{i=1}^{k}\varepsilon_i] = P_r[\varepsilon_1] \times P_r[\varepsilon_2|\varepsilon_1] \times P_r[\varepsilon_3|\varepsilon_2 \cap \varepsilon_1] \cdots P_r[\varepsilon_k|\bigcap_{i=1}^{k-1}\varepsilon_i]$, we have that

$$\frac{1}{C_3^2} \times P_r[\bigcap_{i=1}^{n-3}\varepsilon_i] \ge \frac{1}{3}\prod_{i=1}^{n-3}\left(1 - \frac{2|S|}{k'(n-i+1)}\right)$$

Let $t = 2|S|/k'$, then

$$\frac{1}{3}\prod_{i=1}^{n-3}\left(1 - \frac{2|S|}{k'(n-i+1)}\right) = \frac{1}{3}(1-\frac{t}{n})(1-\frac{t}{n-1})\cdots(1-\frac{t}{4})$$



Obviously, we have that $1 - t/4 > 0$, i.e., $t < 4$ and $|S|/k' < 2$. Therefore, if G satisfies the condition $x'/k' < 2$ (i.e., $|S|/k' \leq x'/k' < 2$), then the algorithm 1 can be called.

Since the cardinality of minimum edge cut is less than or equal to that of minimum cyclic edge cutset, we have that $k' \leq |S|$, i.e., $t \geq 2$. So,

$$2 \leq t < 4. \tag{2.2}$$

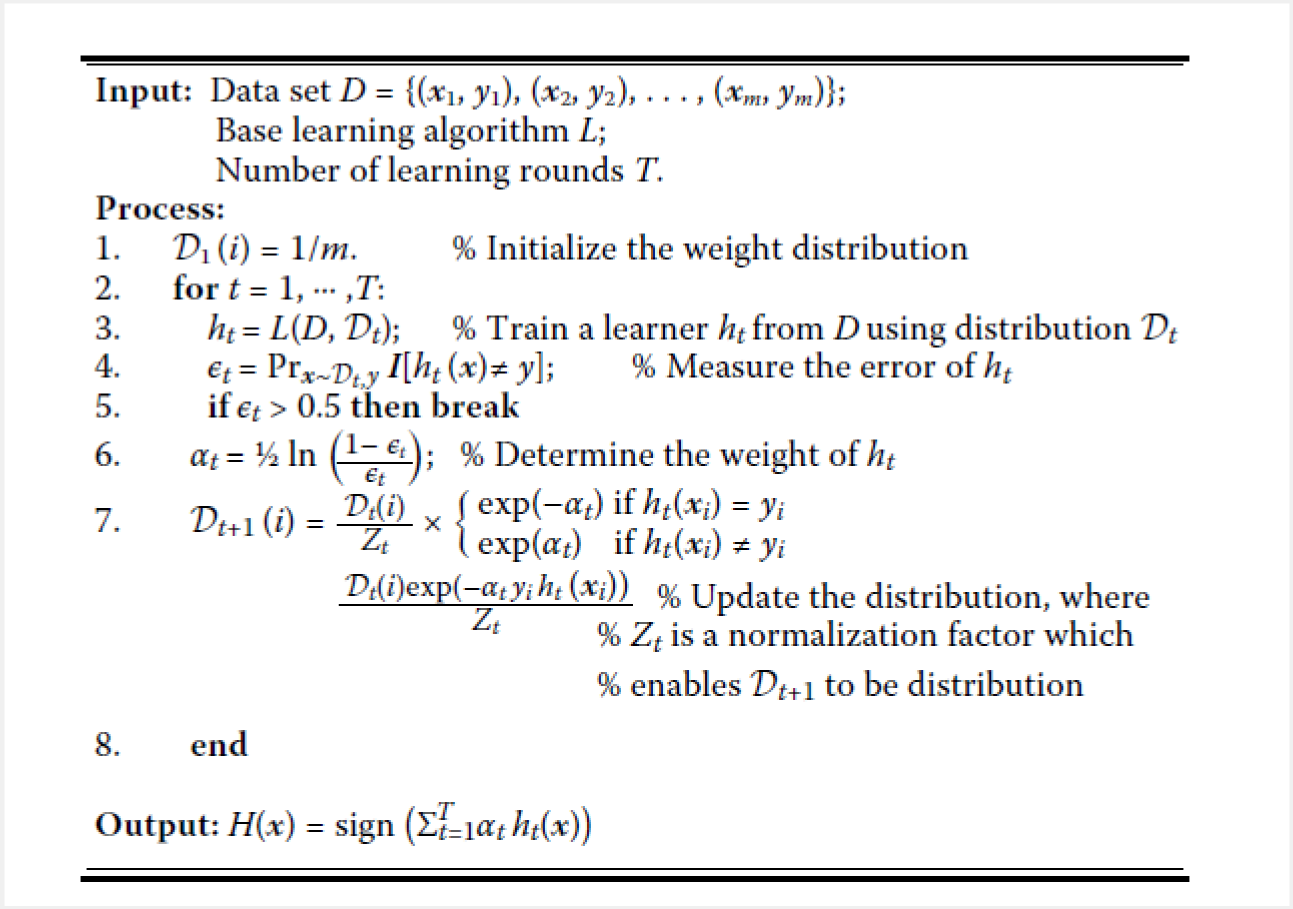
According to the range of t, we discuss it.

- $t = 2$. We have that $P \geq \dfrac{2}{n(n-1)}$
- $t \in (2, 3]$. We have that $P \geq \dfrac{x_0}{n(n-1)(n-2)}$, x_0 is a positive constant.
- $t \in (3, 4)$. We have that $P \geq \dfrac{x'_0}{n(n-1)(n-2)(n-3)}$, x'_0 is a positive constant.

Suppose that $t = 2$. Notice that $P > 2/n^2$ and Algorithm 1 may not be able to correctly find a minimum cyclic edge cutset. Suppose we repeat Algorithm 1 $n^2/2$ times and make independent random choices every time. According to the formula $P_r[\varepsilon_1 \cap \varepsilon_2] = P_r[\varepsilon_1] \times P_r[\varepsilon_2]$, in any of $n^2/2$ times of attempts, the probability that a minimum cyclic edge cutset cannot be correctly found is at most

$$(1 - \frac{2}{n^2})^{n^2/2} < 1/e.$$

Machine learning algorithm——Adaboost:



```
Input:  Data set $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$;
        Base learning algorithm $L$;
        Number of learning rounds $T$.
Process:
1.  $\mathcal{D}_1(i) = 1/m$.        % Initialize the weight distribution
2.  for $t = 1, \cdots, T$:
3.      $h_t = L(D, \mathcal{D}_t)$;      % Train a learner $h_t$ from $D$ using distribution $\mathcal{D}_t$
4.      $\epsilon_t = \Pr_{x \sim \mathcal{D}_t, y} I[h_t(x) \neq y]$;      % Measure the error of $h_t$
5.      if $\epsilon_t > 0.5$ then break
6.      $\alpha_t = ½ \ln \left(\frac{1-\epsilon_t}{\epsilon_t}\right)$;  % Determine the weight of $h_t$
7.      $\mathcal{D}_{t+1}(i) = \frac{\mathcal{D}_t(i)}{Z_t} \times \begin{cases} \exp(-\alpha_t) & \text{if } h_t(x_i) = y_i \\ \exp(\alpha_t) & \text{if } h_t(x_i) \neq y_i \end{cases}$
        $\frac{\mathcal{D}_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$  % Update the distribution, where
                                   % $Z_t$ is a normalization factor which
                                   % enables $\mathcal{D}_{t+1}$ to be distribution
8.  end
Output: $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$
```

Reference: 机器学习十大算法.pdf

The thought of the algorithm Adaboost

1. First, the weight distribution D_1 of training data is initialized. each training sample is given the same weight : w_i $(1 \leq i \leq N) = 1 / N$.
2. The weak classifiers are iteratively trained. If a training data is misclassified by a weak classifier, then its corresponding weight should be increased in constructing the next training set used by the next weak classifier. on the contrary, if a training data is classified accurately , then its weight should be reduced.
3. Finally, the weak classifiers obtained by each training are combined into a strong classifier. A weak classifier with a lower error rate takes a larger weight.

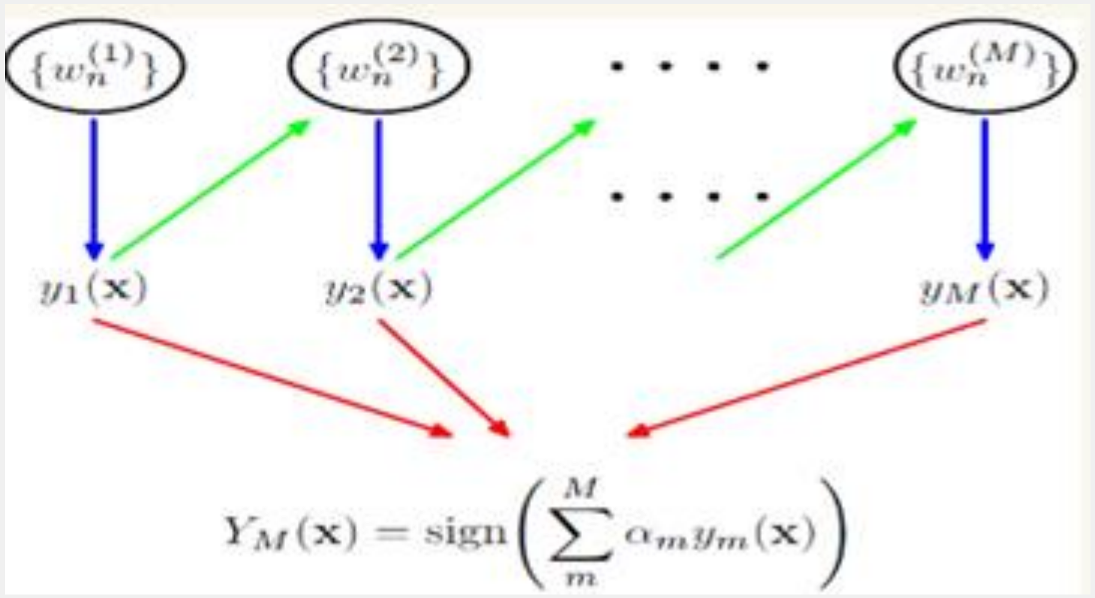


Figure 1 (From PRML)

Error rate:

$$e_t = P(H_t(x_i) \neq y_i) = \sum_{i=1}^{N} w_{ti} I(H_t(x_i) \neq y_i)$$

The weight of each weak classifier :

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-e_t}{e_t}\right)$$

Update the weight distribution of training samples:

$$D_{t+1} = \frac{D_t(i)\exp(-\alpha_t y_i H_t(x_i))}{Z_t}$$

- Combining these weak classifiers:

$$\mathrm{f(x)} = \sum_{t=1}^{T} a_t H_t(x)$$

- Symbol function *Sign:*

$$\begin{aligned}\mathrm{H(x)} &= \mathrm{sign(f(x))} \\ &= \mathrm{sign}(\sum_{t=1}^{T} a_t H_t(x))\end{aligned}$$

Further Problems

- whether the determination of the cyclic vertex connectivity of graphs (k-regular graphs) is an NP problem.

- How to use the distributed algorithm and the random algorithm to determine the cyclic edge connectivity and the cyclic vertex connectivity of graphs.

Thank you!