The Path Compression Problem on Graphs

Hsin-Hung Chou

Chang-Biau Yang

Correspondence address:

Prof. Chang-Biau Yang

Department of Applied Mathematics

National Sun Yat-sen University

Kaohsiung, Taiwan 804

Republic of China

TEL: 886-7-5316171 ext. 3666
FAX: 886-7-5319479
E-MAIL: cbyang@math.nsysu.edu.tw

Keywords. Algorithm, graph, path compression, NP-complete.

Abstract

Given a graph, how do we represent the paths in the graph with the least information? This induces the path compress problem on graphs in which we are asked to represent a path by using fewer edges or vertices so that any two different paths are distinctive. There are two versions of compression problem: edge compression and vertex compression. For the edge compress problem, we show that it can be solved in linear time on general graphs. For the vertex compression, we prove that it is NP-hard. Besides, we propose a heuristic algorithm with polynomial time to solve it. We also do some experiments and obtain some experiment results which illustrate the efficiency of our heuristic algorithm.

1 Introduction

Given a graph, how do we represent the paths in the graph with the least information? For instance, how do we represent the city bus tours in a city? How do we record the walks which can get the highest scores in a computer video game, PCMAN? Of course, we hope that the least information is used to represent the tours or walks. For simplifying the problem, we concentrate on the restricted case that a path does not meet the same vertex twice. In other words, we are asked to represent a path by using fewer edges or vertices so that any two different paths are distinctive. And, we call this problem the **path compression problem**.

In this problem, a subset of vertices or edges are selected to represent the paths such that different paths still have distinct representations. There are two versions of this problem : **vertex compression** and **edge compression**. For the **vertex compression problem**, paths are represented by vertices and we are asked to find a subset of vertices with minimum size to represent all paths in the given graph. Similarly, paths are represented by edges for the **edge compression problem** and we want to select a subset of edges with minimum size.

Let $G = (V, E)$ be a connected and undirected simple graph. A **reserved vertex set** is a subset V' of V that every path in G has a unique representation by using only the vertices in V'. And the vertex compression problem is to find a reserved vertex set with minimum size, called a **minimum reserved vertex set**. Similarly, a **reserved edge set** is a subset E' of E that every path in G has a unique representation by using only the edges in E'. And the edge compression problem is to find a **minimum reserved edge set**.

For example, for the edge compression problem, consider the graph shown in Figure 1. We select a minimum reserved edge set $E' = \{e_1, e_2\}$. As we can see in Table 1, each

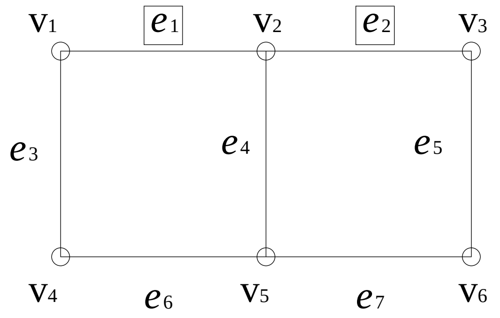

Figure 1: An example for the edge compression problem.

Source vertex	Destination vertex	Represented by E	Represented by E’
v_1	v_2	e_1	e_1
		$e_3e_6e_4$	null
		$e_3e_6e_7e_5e_2$	e_2
v_1	v_6	$e_1e_2e_5$	e_1e_2
		$e_1e_4e_7$	e_1
		$e_3e_6e_7$	null
		$e_3e_6e_4e_2e_5$	e_2

Table 1: Paths in the graph shown in Figure 1.

path represented by the edges in E, now also has a unique representation with the edges in E'.

Figure 2 shows an example of the vertex compression problem. We select a minimum reserved vertex set $V' = \{v_1, v_2, v_4, v_5\}$. Table 2 shows that each path represented by the vertices in V, now also has a unique representation with the vertices in V'.

The rest of this paper is organized as follows. In Section 2 and Section 3, we shall discuss the edge compression problem and the vertex compression problem respectively. For the edge compression problem, it can be solved in linear time on general graphs. Nevertheless, for the vertex compression problem, we prove that it is NP-hard and NP-complete on general graphs and planar graphs respectively. In Section 4, we shall propose a heuristic algorithm to solve the vertex compression problem on planar graphs. In Section 5, we shall analyze the time complexity of the algorithm proposed in Section 4, which is $O(|V|^4 \log |V|)$. And in Section 6, some experiment results will be illustrated for this problem. Finally, some concluding remarks will be given Section 7.

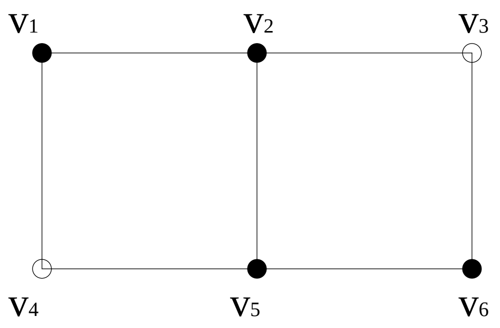

Figure 2: An example for the vertex compression problem.

Source vertex	Destination vertex	Represented by V	Represented by V'
V_1	V_2	null	null
		$V_4 V_5$	V_5
		$V_4 V_5 V_6 V_3$	$V_5 V_6$
V_1	V_6	$V_2 V_3$	V_2
		$V_2 V_5$	$V_2 V_5$
		$V_4 V_5$	V_5
		$V_4 V_5 V_2 V_3$	$V_5 V_2$

Table 2: Paths in the graph shown in Figure 2.

2 The Edge Compression Problem

There are two disjoint paths between any two vertices on a cycle. They will have the same representations, *null*, if there is no reserved edge on the cycle. And if at least one edge is reserved on the cycle, the two paths will have distinct representations. Thus, we have the following observation:

Observation 1 *For a connected and undirected simple graph, each cycle must have at least one edge in a reserved edge set.*

Thus, the edge compression problem is equivalent to the arc deletion problem [5] described as follows:
Given a graph $G = (V, E)$, find a subset E', with minimum size, of E such that the deletion of all edges in E' breaks all cycles.

For solving the edge compression problem, we have the following theorem:

Theorem 1 *Let $G = (V, E)$ be a connected and undirected simple graph, and $T = (V, E_T)$ be one of spanning trees in G. For the edge compression problem, $E - E_T$ is a minimum reserved edge set.*

Proof: Let E' be a minimum reserved edge set in G. From Observation 1, in each cycle of G, at least one edge belongs to E'. Thus, the subgraph composed of $E - E'$ is acyclic. Let $E_T = E - E'$. Since E' is minimum, then E_T is maximum. And it is obvious that an acyclic subgraph with maximum number of edges is a spanning tree. Therefore, $T = (V, E_T)$ is a spanning tree in G. This completes our proof.□

The algorithm for the edge compression problem:

Step 1: Find a spanning tree $T = (V, E_T)$ by **depth-first search**.

Step 2: Let $E' = E - E_T$, which is a minimum reserved edge set.

The correctness of the algorithm is obvious according to Theorem 1. And the time complexity of the algorithm is $O(|V| + |E|)$, which is required for finding a spanning tree.

3 The Vertex Compression Problem

Let $G = (V, E)$ be a connected and undirected simple planar graph and F be the set of the internal faces on G. A **cycle** $\{v_1, v_2, \cdots, v_k\}$ on G is a subset, with distinct vertices, of V such that v_i is adjacent to v_{i+1}, $1 \leq i \leq k-1$, and v_k is adjacent to v_1. A **region** is an area composed by a set of **internal faces** in F. The cycle of a region R, denoted as $C(R)$, is the boundary of R. A **fitted cycle** is a cycle containing three or more reserved vertices. An **unfitted cycle** is a cycle with no more than two reserved vertices. An unfitted cycle with zero, one or two reserved vertices is called a **0-unfitted cycle**, **1-unfitted cycle** or **2-unfitted cycle** respectively. A **basic region** is a region without proper subregions having unfitted cycles, and the corresponding cycle is called a **basic cycle**. A basic cycle is a **basic unfitted cycle** if itself is unfitted.

Suppose that $f_1, f_2, \cdots, f_{|F|}$ are the internal faces in F, it is obvious that the cycle set $\{C(f_1), C(f_2), \cdots, C(f_{|F|})\}$ is a cycle basis, where $C(f_i)$ is the cycle corresponding to the boundary of f_i,$1 \leq i \leq |F|$. A **cycle basis** is a collection of cycles such that each cycle on G can be composed by only using the cycles in the cycle basis in the following way. Let $P = \{u_1, u_2, \cdots, u_p, v_1, v_2, \cdots, v_q\}$ and $Q = \{v_1, v_2, \cdots, v_q, w_1, w_2, \cdots, w_r\}$ be two cycles. Suppose the common edges between P and Q are $\{v_1, v_2\}, \{v_2, v_3\}, \cdots, \{v_{q-1}, v_q\}$. The combination of P and Q , denoted as $P \oplus Q$, is defined as edges in either P or Q, but not both. In other words, $P \oplus Q = \{u_1, u_2, \cdots, u_p, v_1, w_r, w_{r-1}, \cdots, w_1, v_q\}$. And it can be easily verified that the operation $\oplus$ is associative.

For the vertex compression problem, it is clear that if there are no reserved vertices on a cycle, there will be two ambiguous paths, *null*, between any pair of vertices on the cycle. Similarly, if there is only one reserved vertex on a cycle, there will be two ambiguous paths, *null*, between the reserved vertex and any other vertex on the cycle. And if there are only two reserved vertices on a cycle, there will be two ambiguous paths, *null*, between these two reserved vertices. And if there are at least three reserved vertices on a cycle, there will be at least one reserved vertex on one of the two paths between any two vertices. It is obvious that the representations of these two paths are unique. Thus, we have the following observation.

Observation 2 *For a connected and undirected simple graph, each cycle must have at least three reserved vertices in a reserved vertex set.*

For the vertex compression problem, it is NP-hard on general graphs, and it is NP-complete on planar graphs. Furthermore, it is NP-complete on planar graphs without vertex degree exceeding 8. In order to prove the NP-hardness, we redefine the vertex compression problem to a decision problem as follows:

Given a graph $G = (V, E)$ and a positive integer $K \leq |V|$. Is there a reserved vertex set $V' \subseteq V$ such that $|V'| \leq K$ and, for each cycle, at least three vertices on the cycle belong to V'?

Before presenting our theorems, we present another NP-complete problem, the vertex covering problem [3], as follows:

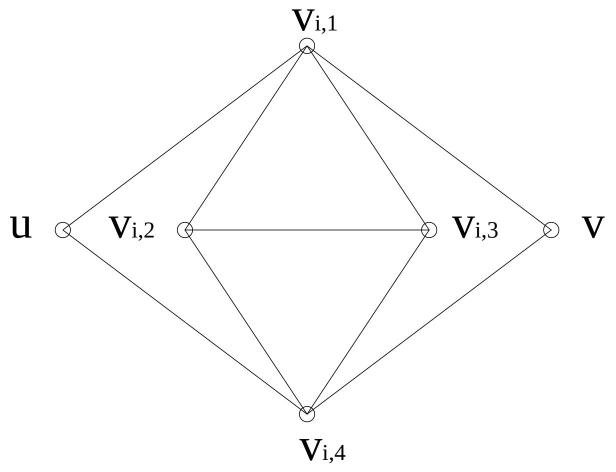

Figure 3: Local replacement for $e_i = \{u, v\} \in E$ for transforming the vertex covering problem to the vertex compression problem.

Given a graph $G = (V, E)$ and a positive integer $K \leq |V|$. Is there a subset $V' \subseteq V$ such that $|V'| \leq K$ and, for each edge $\{u, v\} \in E$, at least one of u and v belongs to V'?

Theorem 2 *The vertex compression problem on general graphs is NP-hard.*

Proof: We shall prove NP-hardness of the vertex compression problem by reducing the vertex covering problem to it. Let $G = (V, E)$ with a positive integer $K \leq |V|$ be an arbitrary instance of the vertex covering problem. We shall construct a graph $G' = (V', E')$ such that the vertex compression problem has a reserved vertex set of size $K + 4|E|$ or less in G' if and only if G has a vertex cover of size K or less.

For each edge $e_i = \{u, v\} \in E$, four vertices $v_{i,1}$, $v_{i,2}$, $v_{i,3}$ and $v_{i,4}$ are added, and the corresponding edge set $E'_i = \{\{u, v_{i,1}\},\{u, v_{i,4}\},\{v_{i,2}, v_{i,1}\},\{v_{i,2},v_{i,4}\}, \{v_{i,2},v_{i,3}\},\{v_{i,1},v_{i,3}\},\{v_{i,4}, v_{i,3}\},\{v_{i,1}, v\},\{$
which is shown in Figure 3. Thus $G' = (V', E')$ is defined as

$V' = V \cup \bigcup_{i=1}^{|E|} \{v_{i,j} \mid 1 \leq j \leq 4\}$ and

$E' = \bigcup_{i=1}^{|E|} E'_i$.

Notice that $|V'| = |V| + 4|E|$ and $|E'| = 9|E|$. It is not hard to see that this instance of the vertex compression problem can be constructed in polynomial time from the vertex covering instance.

We claim that G' has a reserved vertex set of size $K + 4|E|$ or less if and only if G has a vertex cover of size K or less. It is obvious that any reserved vertex set in G' must contains all $v_{i,j}$s, $1 \leq i \leq |E|$ and $1 \leq j \leq 4$, because of the triangles $\{v_{i,1}, v_{i,2}, v_{i,3}\}$ and

$\{v_{i,2}, v_{i,3}, v_{i,4}\}$. Hence each reserved vertex set contains at least $4|E|$ vertices. With these $4|E|$ vertices, it is clear that the only unfitted cycles are $\{u, v_{i,1}, v, v_{i,4}\}$ in E_i', $1 \leq i \leq |E|$. Since there are two reserved vertices on such a cycle, at least one of u and v for each E_i' have to be selected as a reserved vertex. If there is a reserved vertex set of size $K+4|E|$, it means that we select K vertices from V to satisfy each E_i'. It is clear that the K vertices of V compose a vertex covering in G.

Conversely, if G has a vertex covering of size K, it is clear that these K vertices and all vertices $v_{i,j}$s, $1 \leq i \leq |E|$ and $1 \leq j \leq 4$, compose a reserved vertex set in G'. □

It is known that the vertex covering problem is NP-complete even for planar graphs with no vertex degree exceeding 4 [3]. Thus, we have the following theorems:

Theorem 3 *The vertex compression problem on planar graphs with no vertex degree exceeding 8 is NP-complete.*

Proof: For a given vertex set, we can decide whether it is a reserved vertex set or not on planar graphs by the algorithm illustrated in the next section in polynomial time. Thus, the vertex compression problem on planar graphs belongs to NP. And the NP-hardness can be proved in the same way as Theorem 2, transforming the vertex covering problem on planar graphs with no vertex degree exceeding 4 to this problem. And it is easy to see that the graph we construct has no vertex degree exceeding 8. □

With Theorem 3, we have the following Corollary:

Corollary 1 *The vertex compression problem on planar graphs is NP-complete.*

4 Heuristic algorithm for the vertex compression problem

To solve the vertex compression problem on planar graphs, with Observation 2, we propose the following heuristic algorithm to select the reserved vertices such that there are at least three reserved vertices in each cycle. The major stages in our main algorithm are selecting the reserved vertices and searching the unfitted cycles. The main algorithm for solving the vertex compression problem is as follows:

Step 1: Set the input cycles for Subalgorithm I to be the cycles of the internal faces initially.

Step 2: Call Subalgorithm I for selecting the reserved vertices to satisfy all the input cycles.

Step 3: Call Subalgorithm II for finding 0-unfitted cycles.

Step 4: Call Subalgorithm III for finding 1-unfitted cycles.

Step 5: Call Subalgorithm IV for finding 2-unfitted cycles.

Step 6: Delete the duplications in the set of the unfitted cycles that were found in Steps 3 through 5.

Step 7: Set the input cycles for Subalgorithm I to be the unfitted cycles that were found in Steps 3 through 5.

Step 8: Repeat Steps 2 through 7 until no unfitted cycles can be found.

For selecting the reserved vertices, at first, we select the vertices contained in the cycles of length 3. Furthermore, we select one vertex with the maximum contribution step by step until all input cycles are fitted.

Subalgorithm I:

C *: the set of the unfitted cycles.*

NR *: the set of the vertices which are not reserved.*

VPC_i *: number of reserved vertices on cycle* C_i, $1 \leq i \leq |C|$.

CAV_j *: number of unfitted cycles containing vertex* v_j, $1 \leq j \leq |V|$.

Step 1:

```
    For each cycle $C_i \in C$ of length 3
        $C = C - \{C_i\}$
        For each $v_j \in C_i$ and $v_j \in NR$, select $v_j$ as a reserved vertex
            $NR = NR - \{v_j\}$
            For each $C_k \in C$ containing $v_j$
                $VPC_k = VPC_k + 1$
    For each cycle $C_i \in C$ of length greater than 3 and $VPC_i \geq 3$
        $C = C - \{C_i\}$
        For each $v_j \in C_i$ and $v_j \in NR$
            $CAV_j = CAV_j - 1$
```

```
Step 2:
    Select $v_k$ as a reserved vertex such that $CAV_k = max\{CAV_j \mid 1 \leq j \leq |V|$ and
      $v_j \in NR\}$.
    Set $NR = NR - \{v_k\}$.
Step 3:
    For each $C_i \in C$ containing $v_k$
        $VPC_i = VPC_i + 1$
        if $VPC_i = 3$ then
            $C = C - \{C_i\}$
            For each $v_j \in C_i$ and $v_j \in NR$
                $CAV_j = CAV_j - 1$
Step 4: Repeat Steps 2 and 3 until $C = \emptyset$.
```

Some notations will be used in Subalgorithms II through IV, which are defined as follows.

F : the set of the internal faces on the given planar graph.
I : the set of the internal faces being included for searching unfitted cycles.
B : the set of the vertices on the boundary of the union of all faces in I.
R : the set of the reserved vertices belonging to B.
$FC(v)$: the set of the faces containing vertex v.

In Subalgorithm II, for finding 0-unfitted cycles, we start searching from each internal face in a searching iteration. We repeatedly include other internal faces to remove the reserved vertices on the boundary to get a new boundary that may be an unfitted cycle. The algorithm is described as follows:

Subalgorithm II:

```
for each $f_j \in F, 1 \leq j \leq |F|$
    $I = \{f_j\}$
    find $B$ and $R$
    repeat
        for each vertex $v \in R$
            $I = I \bigcup FC(v)$
        end for
        find $B$ and $R$
    until ($|R| \leq 2$) or (there exists one reserved vertex in $R$ on the external boundary)
    if $|R| \leq 2$ then store $B$ as an unfitted cycle
end for
```

In Subalgorithm III, for finding 1-unfitted cycles, we want to find an unfitted cycle with one reserved vertex. Therefore, in a searching iteration, we start searching from each internal face and keep one reserved vertex v on that face throughout the iteration. In the subsequent works, we repeatedly include other internal faces to remove the reserved vertices other than v on the boundary. The algorithm is described as follows:

Subalgorithm III:

```
for each $f_j \in F$,$1 \leq j \leq |F|$
    for each reserved vertex $r$ on the boundary of $f_j$
        $I = \{f_j\}$
        find $B$ and $R$
        repeat
            for each vertex $v \in R$ and $v \neq r$
                $I = I \bigcup FC(v)$
            end for
            find $B$ and $R$
        until ($|R| \leq 2$) or ($r \notin R$) or (there exists one reserved vertex in $R$ except $r$
            on the external boundary)
        if $|R| \leq 2$ then store $B$ as an unfitted cycle
    end for
end for
```

In Subalgorithm IV, for finding 2-unfitted cycles, we keep two reserved vertices and start searching from each internal face containing one of these two reserved vertices. The algorithm is described as follows:

Subalgorithm IV:

```
for every two distinct reserved vertices $r_1$ and $r_2$
    for each $f_j$ in $FC(r_1)$
        $I = \{f_j\}$
        find $B$ and $R$
        repeat
            for each vertex $v \in R$ and $v \neq r_1$ and $v \neq r_2$
                $I = I \bigcup FC(v)$
            end for
            find $B$ and $R$
        until ($|R| \leq 2$) or ($r_1 \notin R$) or ($FC(r_2) \bigcap I \neq \emptyset$ and $r_2 \notin R$) or (there exists
          one reserved vertex in $R$ except $r_1$ and $r_2$ on the external boundary)
        if $|R| \leq 2$ then store $B$ as an unfitted cycle
    end for
end for
```

5 Analysis of the heuristic algorithm

In this section, we shall analyze the time complexity of the algorithm proposed in the previous section.

Lemma 1 *Suppose there are two basic regions R_1 and R_2 on a planar graph G and R_3 is the intersection region of R_1 and R_2. If $C(R_1)$ and $C(R_2)$ are both unfitted cycles, then R_3 must be empty.*

Proof: Assume that R_3 is not empty and let n_1 and n_2 denote the numbers of reserved vertices on the two paths of the intersections of $C(R_1)$ and $C(R_2)$ respectively, as shown in Figure 4. And let the numbers of reserved vertices in $C(R_1) - C(R_3)$, $C(R_3) - C(R_2)$, $C(R_3) - C(R_1)$ and $C(R_2) - C(R_3)$ be denoted as r_1, r_2, r_3 and r_4 respectively, where $C(R_i) - C(R_j) = \{v_k | v_k \in C(R_i) \text{ and } v_k \notin C(R_j)\}$.

If R_3 is not empty, then $C(R_3)$ is a fitted cycle because there is no unfitted cycle contained in basic regions R_1 and R_2. Then we have the following inequalities:

(1) $r_1 + r_2 + n_1 + n_2 \leq 2$ from $C(R_1)$

(2) $r_3 + r_4 + n_1 + n_2 \leq 2$ from $C(R_2)$

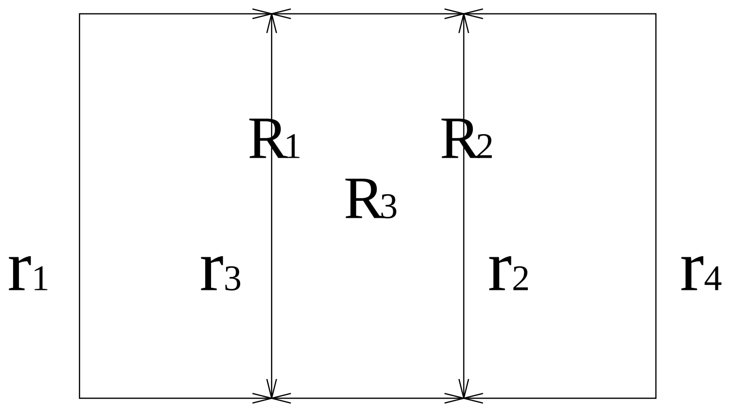

Figure 4: Two basic regions R_1 and R_2.

(3) $r_1 + r_3 \geq 3$ from $C(R_1 - R_3)$

(4) $r_2 + r_3 + n_1 + n_2 \geq 3$ from $C(R_3)$

(5) $r_2 + r_4 \geq 3$ from $C(R_2 - R_3)$

where $n_1 = 0$, 1 or 2.

$n_2 = 0$, 1 or 2.

$r_i = 0$, 1 or 2 , $1 \leq i \leq 4$.

Consider the three possible values of r_1:

Case 1: $r_1 = 2$.

From (1), we get

$r_2 + n_1 + n_2 = 0.$

Then, from (4), we obtain

$r_3 \geq 3.$

This is contrary to (2).

Case 2: $r_1 = 1$.

From (1), we get

$r_2 + n_1 + n_2 \leq 1.$

Then, from (4), we obtain

$r_3 \geq 2.$

Then from (2), we have

$r_4 + n_1 + n_2 = 0.$

Implies $r_4 = 0$.

Then, from (5), we get

$r_2 \geq 3.$

This is contrary to (1).

Case 3: $r_1 = 0$.

From (3), we get

$r_3 \geq 3.$

This is contrary to (2).

Thus, if $C(R_1)$ and $C(R_2)$ are both unfitted cycles,then R_3 must be empty.□

Lemma 2 *Subalgorithms II through IV will find all the basic unfitted cycles.*

Proof: There are three types of unfitted cycles as shown in Figure 5.

In Subalgorithm II, if there exists a basic 0-unfitted cycle C, as shown in Figure 5 (a), we can search from some f_i, as shown in Figure 6 (a). Then the unique stop condition is that we find an unfitted boundary; otherwise there must be at least one reserved vertex on the boundary of C, it is contrary to that C is a 0-unfitted cycle. And if the unfitted boundary contains more than one cycle, as shown in Figure 7, it is obvious that C is not a basic unfitted cycle. Thus, all basic 0-unfitted cycles will be found by Subalgorithm II, which starts searching from each internal face.

For Subalgorithms III and IV, the searching directions are shown in Figure 6 (b) and Figure 6 (c) respectively. And with the same reasons as Subalgorithm II, all basic 1-unfitted and 2-unfitted cycles will be found by Subalgorithms III and IV respectively. □

Lemma 3 *The time complexity of Subalgorithm I is $O(|C| \times |V|)$.*

Proof: For the statement $C = C - \{C_i\}$, it will be executed only when C_i contains 3 reserved vertices. It is clear that it will be executed $|C|$ times. For the statement $NR = NR - \{v_i\}$, it will be executed only when v_i is selected as a reserved vertex. Thus, it will be executed $|V|$ times. And for the statement $VPC_i = VPC_i + 1$, for each cycle C_i, this statement will be executed no more than 3 times. Therefore, it will be executed

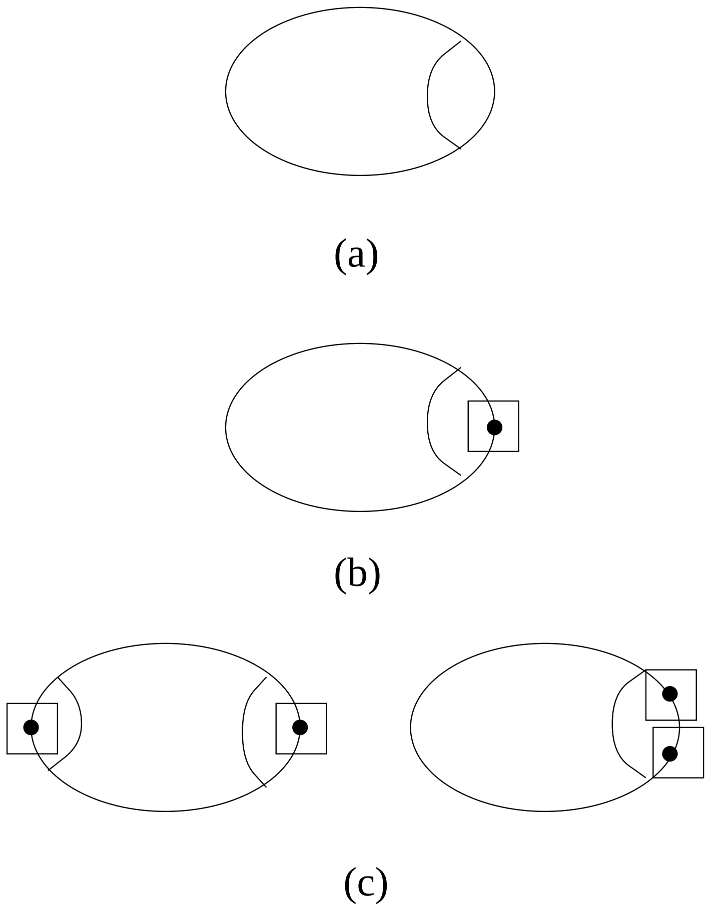

Figure 5: Three types of unfitted cycles. (a) A 0-unfitted cycle. (b) A 1-unfitted cycle. (c) 2-unfitted cycles.

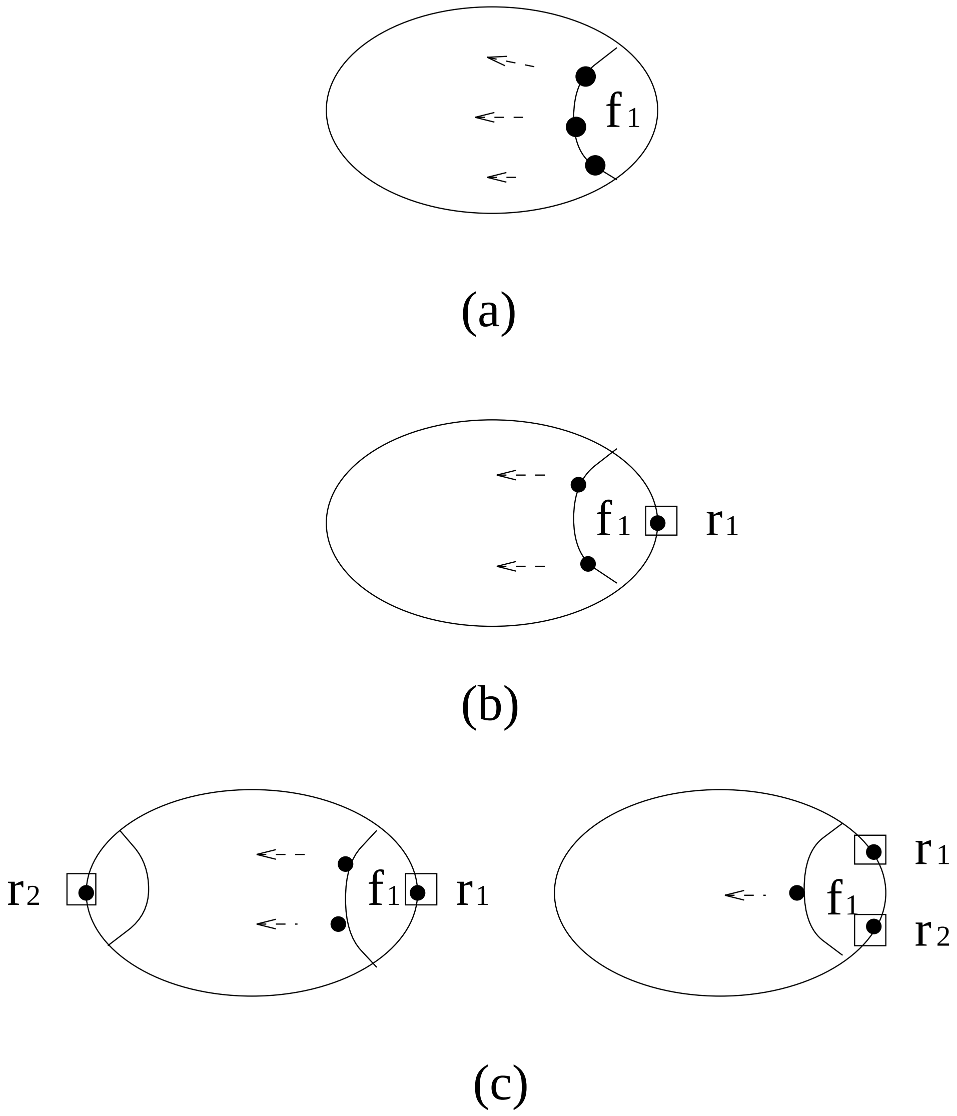

Figure 6: Searching directions for the three types of unfitted cycles.

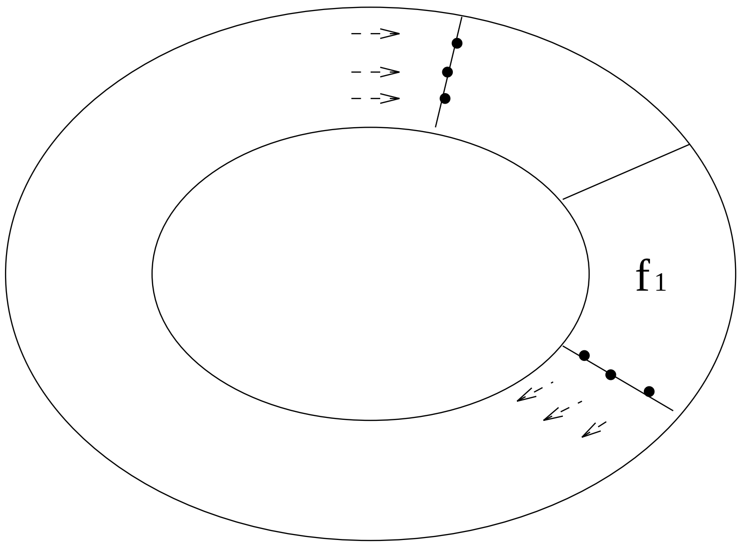

Figure 7: An unfitted boundary with two disjoint cycles.

in $O(|C|)$ times. And for the statement $CAV_i = CAV_i - 1$, it will be executed only when the cycle containing v_i contains 3 reserved vertices. In the worst case, it will be executed $|C| \times |V|$ times. In Step 2, we need $O(|V|)$ time to select the maximum and it will be executed $O(|C|)$ times. So its time complexity is $O(|C| \times |V|)$. By summing up the above individual time complexities, we have the total time complexity of the Subalgorithm I is $O(|C| \times |V|)$. □

Lemma 4 *The time complexity of Subalgorithm II is* $O(|F| \times (|V| + |E| + |F|))$.

Proof: At first, let us consider the time complexity in a searching iteration from an internal face f_j. For finding the new boundary, for each vertex v_i, we record the number, n_i, of the faces containing v_i in I. We know that $\sum_{i=1}^{|V|} d_i = 2 \times |E|$, where d_i is the degree of v_i. And the number of the internal faces containing a vertex is less than or equal to the degree of the vertex. Thus, the time complexity for the statement *find B* is $O(|E|)$. If n_i is not zero and it is less than the degree of v_i, v_i will be on the boundary. Besides, if a vertex is on the boundary and it is a reserved vertex, we put it in R. Therefore, the time complexity for the statement *find B and R* in a searching iteration is $O(|E|)$.

And for checking the stop condition, we can do it in constant time after a repetition since we can use variables to denote the states of the conditions and verify the variables in finding R.

For the statement $I = I \bigcup FC(v)$, the largest possible number of internal faces that will be included is $|F|$. Thus, this statement will be executed at most $|F|$ times in a searching iteration. For saving an unfitted cycle, it is obvious that we need only $O(|V|)$ time to store a cycle.

Thus, the total time complexity of Subalgorithm II is $O(|F| \times (|V| + |E| + |F|))$. $\Box$

Lemma 5 *The time complexity of Subalgorithm III is* $O(|E| \times (|V| + |E| + |F|))$.

Proof: The total number of iterations for finding unfitted cycles is $O(|E|)$. In an iteration, the time complexity is the same as that in Lemma 4. Thus, the time complexity of Subalgorithm III is $O(|E| \times (|V| + |E| + |F|))$. $\Box$

Lemma 6 *The time complexity of Subalgorithm IV is* $O(|V| \times |E| \times (|V| + |E| + |F|))$.

Proof: The total number of iterations for finding unfitted cycles is $O(|E| \times |V|)$. In an iteration, the time complexity is the same as that in Lemma 4. Thus, the time complexity of Subalgorithm IV is $O(|V| \times |E| \times (|V| + |E| + |F|))$. $\Box$

Theorem 4 *The time complexity of the main algorithm is* $O(|V|^4 \log |V|)$.

Proof: Lemma 2 states that all basic unfitted cycles in iteration i can be found. Thus, each basic unfitted cycle found in iteration $i+1$ must be an unfitted cycle that is not basic in iteration i. In the other words, each basic unfitted cycle found in iteration $i+1$ will contain some basic unfitted cycles found in iteration i. And from Lemma 1, we know that the intersection of the regions of any two basic unfitted cycles must be empty. Therefore, the number of basic unfitted cycles must be less than or equal to $|F|$. Because the number of basic unfitted cycles is limited in an iteration and the region of a basic unfitted cycle enlarges gradually, it is obvious that our main algorithm will stop at most $O(|F|)$ iterations.

From Lemma 4 through Lemma 6, the time required for searching unfitted cycles is $O(|V| \times |E| \times (|V| + |E| + |F|))$. And for the Step 6 in the main algorithm, a sorting algorithm can be used to sort the unfitted cycles by comparing the indices of the vertices in the cycles. The number of comparisons for sorting is $O(|C| \log |C|)$, and it takes $O(|V|)$ time in a comparison. Thus, the time complexity of deleting the duplications in the set of the unfitted cycles is $O(|C| \log |C| \times |V|)$.

On a planar graph, it is clear that $|E| = O(|V|)$ and $|F| = O(|V|)$. Therefore, the time complexity of the main algorithm is $O(|C| \log |C| \times |V|^2 + |V|^4)$. And $|C| \leq |V|^2$ since the number of searching iterations in Subalgorithm II through IV is $O(|V|^2)$. Thus, the time complexity of the main algorithm is $O(|V|^4 \log |V|)$. $\Box$

6 Experiments for the vertex compression problem

In our experiment, we propose another algorithm, Subalgorithm I', for selecting the reserved vertices. In this algorithm, we use a variable x_i to represent the state of vertex v_i. We set $x_i = 1$ if v_i is selected as a reserved vertex and $x_i = 0$ if otherwise. We transfer the input cycles to the constraints of the linear programming. Then we solve it by a linear programming package, LINGO. The algorithm is described as follows:
Subalgorithm I':

Step 1: Generate the linear programming constraints corresponding to the input cycles in the following form.
LP:

$$\textit{Minimize} \sum_{i=1}^{|V|} x_i$$

$$\textit{Subject to} \sum_{v_i \in C_j} x_i \geq 3, \textit{ where } C_j \textit{ is the input cycles, } 1 \leq j \leq |C|, \textit{ and}$$

$$x_i = 1, \textit{ if } v_i \textit{ has been selected as a reserved vertex before, and}$$

$$0 \leq x_i \leq 1, \textit{ otherwise.}$$

Step 2: Solve LP. Set $x_i = 1$ for all $x_i > 0$. If $x_i = 1$, fix the vertex v_i to be a reserved vertex.

Moreover, we use the 0/1 integer programming to obtain the optimal solution. Now, in the end of each iteration, we keep the unfitted cycles, do not keep reserved vertices. And, in each iteration, some unfitted cycles are added and reserved vertices are found according to the new set of unfitted cycles. Because the problem space of the package, LINGO, which we use to solve linear programming, is limited, the number of the constraints in LINGO cannot be more than 400, we are restricted to solve the problem with no more than 200 vertices.

In the following experiments, the number of data instances in each testing case is 50. Table 3 illustrates some experiments solved by our heuristic algorithm with Subalgorithm I. And Figure 8 illustrates the relation between the number of vertices and the execution time in Table 3. In Table 4, we provide some experiments to compare the heuristic algorithm with Subalgorithm I and the one with Subalgorithm I'. In these experiments, we also provide the optimal solutions obtained from the algorithm with the 0/1 integer programming.

In Table 4, one can also see that the solutions obtained from the heuristic algorithm with Subalgorithm I are superior to those obtained from Subalgorithm I'. Moreover, the

	# of faces	# of vertices	# of reserved vertices	# of iterations	total time (seconds)
1	10	18	10	1	1.15
2	50	73	53	2	36.36
3	100	151	93	2	468.35
4	150	252	147	2	1573.94
5	200	309	190	2	3674.90
6	280	408	263	2	10228.77

Table 3: Some experiments which select the reserved vertices by Subalgorithm I.

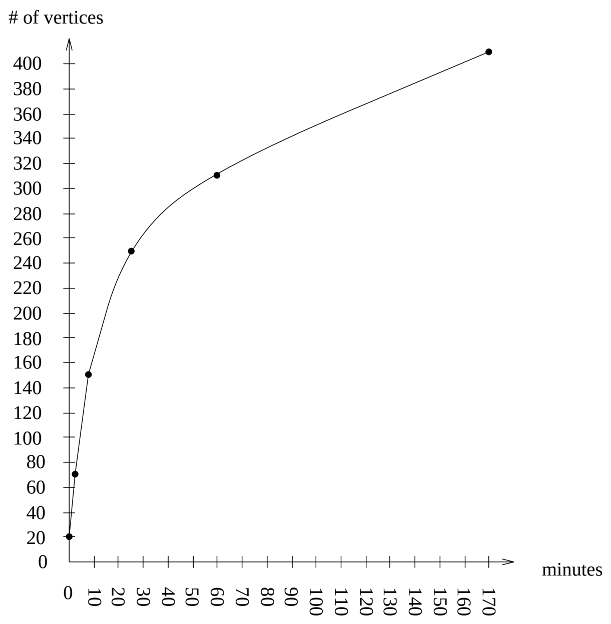

Figure 8: Relation between number of vertices and execution time.

maximum # of vertives per face	# of vertices (average)	# of reserved vertices of optimal solution (average)	# of reserved vertices (average)	maximum difference to optimal solution	# of iterations (average)	# of experiments achieving optimal solution
5	26.60	20.12	20.44	2	1.76	37
7	36.84	22.08	22.58	3	1.82	32
11	58.90	24.48	25.30	3	1.86	22
15	83.78	25.60	26.60	3	1.78	18
20	114.20	26.58	27.98	5	1.88	12

(a)

maximum # of vertives per face	# of vertices (average)	# of reserved vertices of optimal solution (average)	# of reserved vertices (average)	maximum difference to optimal solution	# of iterations (average)	# of experiments achieving optimal solution
5	26.60	20.12	20.74	3	1.82	26
7	36.84	22.08	22.88	3	1.92	27
11	58.90	24.48	25.46	3	1.96	18
15	83.78	25.60	26.98	6	1.90	14
20	114.20	26.58	28.40	5	1.98	10

(b)

Table 4: # of internal faces $= 20$ and # of data instances in each testing case $= 50$. Select reserved vertices by (a) Subalgorithm I (b) Subalgorithm I'.

number of iterations is less than 2 in average in each testing case. And it is one of our open problems: how many iterations are needed in average?

7 Concluding Remarks

The idea of path compression on graphs came from a video game, PCMAN, as we described in Section 1. In this paper, we concentrate on *simple path* instead of *walk*, and consider the general case that source and destination vertices are arbitrary. Moreover, we study this problem on two points of view, edge version and vertex version. We call them edge compression and vertex compression respectively.

For the edge compression problem, it is easier to be solved. It is equivalent to the arc deletion problem and can be solved in linear time. Nevertheless, for the vertex compression problem, we proved that it is NP-hard and NP-complete on general graphs and planar graphs respectively. We proposed a heuristic algorithm with time complexity $O(|V|^4 \log |V|)$ to solve the vertex compression problem. And we also proposed another heuristic algorithm by applying linear programming [2]. In order to get the optimal solution, we restrict the linear programming to the 0/1 integer programming [1]. We also show some experiment results.

In the future, we may develop a new algorithm with the branch and bound strategy [4], the simulated annealing strategy [6] or the genetic strategy [7] or other strategies for solving NP-complete problems.

In this paper, we still have some open problems.

Problem I:

Is the vertex compression problem NP-complete on general graphs?

Problem II:

What is the tight bound of the number of searching iterations in our main algorithm?

Problem III:

In the edge compression problem, each cycle must have at least one reserved edge, how about at least two reserved edges?

References

[1] E. Balas and C. H. Martin, "Pivot and complement - a heuristic for 0-1 programming," *Management*, Vol. 26, No. 1, pp. 86–96, January 1980.

[2] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows.* John Wiley & Sons, 1990.

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability : A Guide to the Theory of NP-Completeness.* San Francisco, USA: Freeman, 1979.

[4] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms.* Computer Science Press, 1978.

[5] R. M. Karp, “Reducibility among combinatorial problems,” *Complexity of Computer Computations* (R. E. Miller and J. W. Thatcher, eds.), pp. 85–103, Plenum Press, 1972.

[6] P. J. M. V. Laarhoven and E. H. L. Aarts, *Simulated Annealing : Theory and Applications.* D. Reidel Publishing Company, 1987.

[7] Z. Michalewicz, *Genetic Algorithm + Data Structures = Evolution Programs.* Springer-Verlag, 1992.