

On shortest weak Hamiltonian path problem of the cactus graph

TOPIC : Graph Theory, Combinatorial Optimization

강원과학고등학교 김준혁

quickn@kangwon-sh.gwe.hs.kr

TABLE OF CONTENTS

1. Shortest weak Hamiltonian path problem

2. Algorithm and correctness

3. Conclusion and its applications

Shortest weak Hamiltonian path problem

Shortest weak Hamiltonian path problem

1

Shortest weak Hamiltonian path problem

1

An undirected connected simple graph $G = (V, E_G, w_G)$ with non-negative weight function $w_G: V \times V \rightarrow \mathbb{R}$

2

Output: A walk $p = v_1 \dots v_n$ through all vertices **at least once**

3

Optimization: $\min(\sum_{i=1}^{n-1} w_G(v_i, v_{i+1}))$

Shortest weak Hamiltonian path problem

2

Frequency Misunderstanding(s)

Q

It is equivalent to TSP?

A

Currently, we haven't any **known proof**. I think it's **not true**. Because reduction to the TSP or reduction from the TSP is **not easy**. This problem guarantees that existence of solution for all undirected simple non-negative weighted graph. However Hamiltonian path or Hamiltonian is not.

Shortest weak Hamiltonian path problem

2

Frequency Misunderstanding(s)

Q

We have an easy reduction from the Hamiltonian path problem.

A

There exists a counterexample.

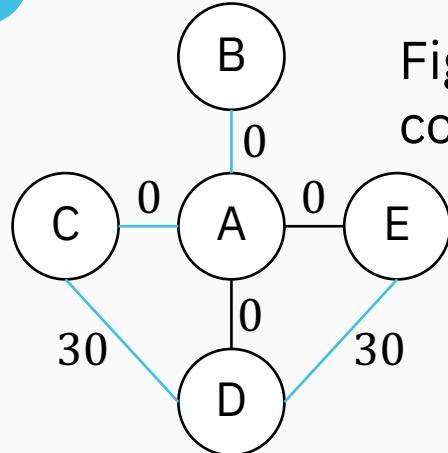


Fig. 1. Optimal cost: 60

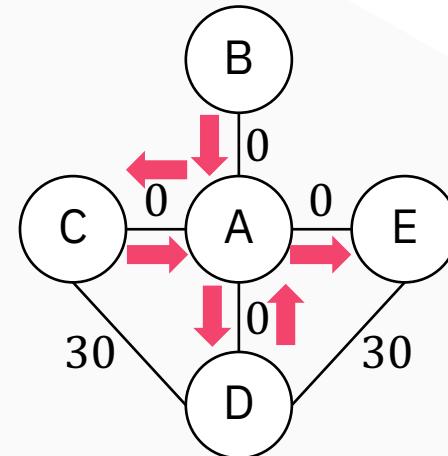


Fig. 2. Optimal cost: 0

Shortest weak Hamiltonian path problem

3

Hypothesis

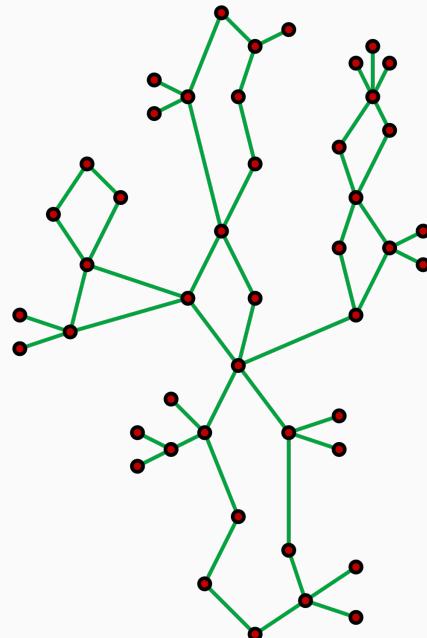
Hypothesis - NP-hard?

- Unknown deterministic-polynomial time algorithm
- It can be considered possible reduction from the Hamiltonian path. (We can use Rahman–Kaykobad's theorem [1] and may select another shortest path of vertex by vertex from the solution of the shortest weak Hamiltonian path)

Shortest weak Hamiltonian path problem

4

Case – Cactus graph



- **Cactus graph** is a finite, connected, undirected, simple graph in which any two simple cycles have at most one vertex in common. [2]
- This study gives a deterministic-polynomial time solution of the Shortest weak Hamiltonian path problem on the cactus graph.

Fig. 3. An example of the cactus graph [3]

Algorithm and correctness

Algorithm and correctness

1

Cycle component transform of the cactus graph

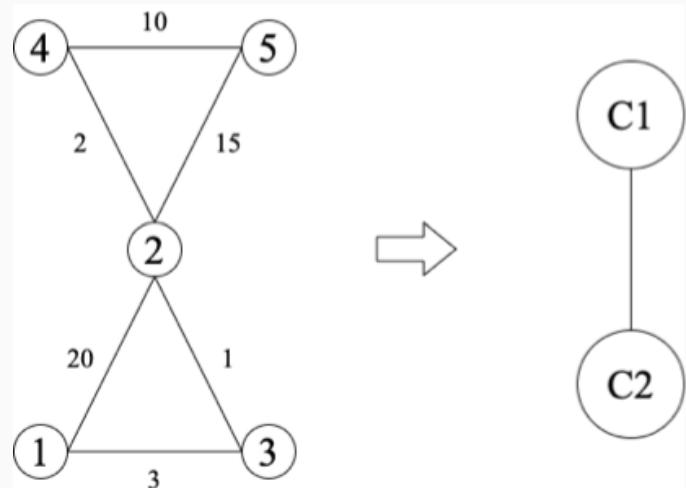


Fig. 5. Graphic Representation of the cycle component transform

- **Cycle component transform of the cactus graph** is a representation of the cycles of the cactus graph.
- Let C be a finite set of the cycles of the cactus graph G .
- INPUT: A cactus graph G
- OUTPUT: A tree $T := (C, E_C)$ where $E_C := \{(C_i, C_j) | C_i, C_j \in C \wedge \exists u \in C_i \cap C_j\}$

Algorithm and correctness

1

Cycle component transform of the cactus graph

- Cycle component transform of the cactus graph can be performed in the $O(|V| + |E_G|)$ time.
- Run DFS(Depth-first-search) in the graph and push each vertex to a stack by pre-order and pop vertices belonging to a cycle if it detects back edge.

Algorithm and correctness

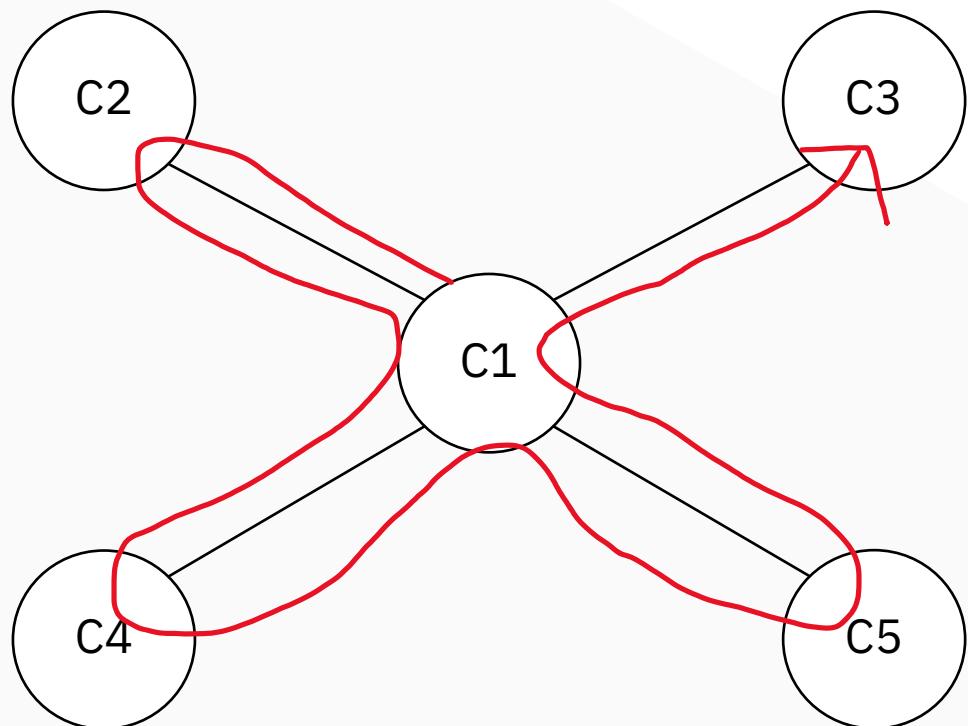


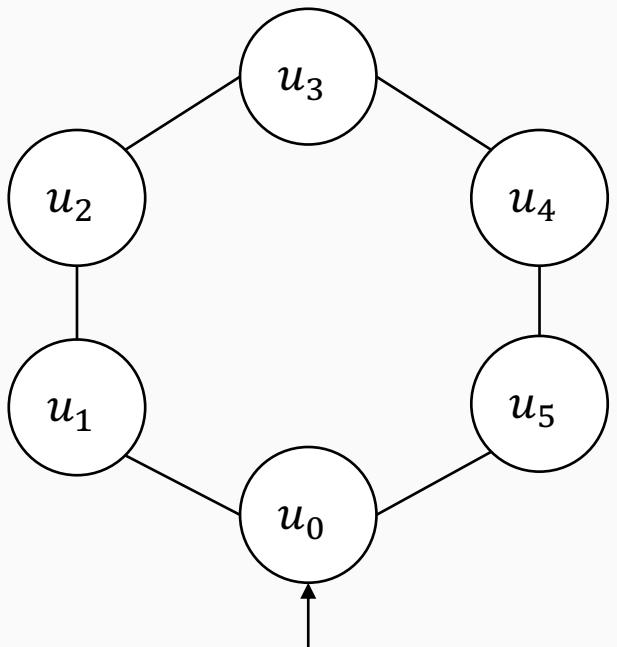
Fig. 6. An example of the optimal path

- We have rooted tree $T = (C, E_C)$ with an arbitrary root $C_r \in C$.
- Firstly, We can find a path through all vertices and visit all edges twice.
- We can remove a walk with maximum cost it leads to leaf from root.

Algorithm and correctness

2

STEP 1 – Find optimal in the cycle



- For a cycle $C_i \in C$, Let $u_0 \in C_i$ be a vertex lead from the parent of the C_i . (If C_i is a root, u_0 is an arbitrary vertex)
- We have an order left to right $u_0 < \dots < u_m$. (clockwise order)

Fig. 7. A cycle with special order.

Algorithm and correctness

2

STEP 1 – Find optimal in the cycle

- We define

$A_{0k}[i] := (\text{Minimum cost with visit } i - \text{th vertex by clockwise order})$

$A_{1k}[i] := (\text{Minimum cost with visit } i - \text{th vertex by counterclockwise order})$

$B_k[i] := (\text{Minimum cost of a path through all vertices and ends at } u_i)$

$\pi_u^1[v] := (\text{Previous vertex of the vertex } v \text{ in the optimal path})$

- The time complexity is $O(|V| + |E_G|)$

Algorithm and correctness

2

STEP 1 – Find optimal in the cycle

Algorithm 1 위 부분 문제를 해결하는 알고리즘

```
1 INPUT: Cycle  $C_k \in C$ 
2 OUTPUT: Arrays  $A_{0k}, A_{1k}, B_k$ 
3 Let  $A_{0k}, A_{1k}$  be 0-indexed arrays with size  $m + 1$ 
4  $A_{0k}[0] \leftarrow 0, A_{1k}[0] \leftarrow 0$ 
5 Let  $\pi_k^1$  be arrays with size  $m + 1$ 
6 for  $i \leftarrow 1$  to  $m$ , do
7    $A_{0k}[i] \leftarrow A_{0k}[i - 1] + w(u_{i-1}, u_i)$ 
8    $A_{1k}[i] \leftarrow A_{1k}[i - 1] + w(u_{(m-i+2) \bmod (m+1)}, u_{m-i+1})$ 
9 Let  $B_k$  be an array with size  $m + 1$ 
10  $\forall i \in \{0, \dots, m\}: B_k[i] \leftarrow \infty$ 
11 for  $i \leftarrow 1$  to  $m - 1$ , do
12   if  $B_k[i + 1] > 2A_{0k}[i] + A_{1k}[m - i]$ , then
13      $B_k[i + 1] \leftarrow 2A_{0k}[i] + A_{1k}[m - i]$ 
14      $\pi_k^1[i + 1] \leftarrow (i, 1)$ 
15   if  $B_k[m - i] > 2A_{1k}[i] + A_{0k}[m - i]$ , then
```

Algorithm and correctness

2

STEP 1 – Find optimal in the cycle

```
10    $B_k[m-i] \leftarrow 2A_{1k}[i] + A_{0k}[m-i]$ 
11    $\pi_k^1[m-i] \leftarrow (m-i+1, 0)$ 
12   if  $B_k[1] > A_{1k}[m]$ , then
13      $B_k[1] \leftarrow A_{1k}[m]$ 
14      $\pi_k^1[1] \leftarrow (0, 0)$ 
15   if  $B_k[m] > A_{0k}[m]$ , then
16      $\pi_k^1[m] \leftarrow (0, 1)$ 
17      $B_k[m] \leftarrow A_{0k}[m]$ 
18   for  $i \leftarrow 1$  to  $m$ , do
19     if  $B_k[0] > B_k[i] + \min(A_{0k}[i], A_{1k}[m-i+1])$ , then
20       if  $A_{0k}[i] \leq A_{1k}[m-i+1]$ , then
21          $B_k[0] \leftarrow B_k[i] + A_{0k}[i]$ 
22          $\pi_k^1[0] \leftarrow (i, 1)$ 
23       else
24          $B_k[0] \leftarrow B_k[i] + A_{1k}[m-i+1]$ 
25      $\pi_k^1[0] \leftarrow (i, 0)$ 
```

Algorithm and correctness

2

Correctness – Find optimal in the cycle

Lemma 1. 위와 같은 사이클의 최단 보행에서
한 방향(clockwise or counterclockwise)으로
진행하다가 다른 방향으로 전환되는 것을 ‘틀어짐’이라고 할때,
이런 ‘틀어짐’은 최대 한번만 발생한다.

Algorithm and correctness

2

Correctness – Find optimal in the cycle

Proof. Let define $S_i := \sum_{e \in E_i} w_G(e)$ and assume that it occurs at u_i . Then, we have $w_G(u_i, u_{(i+1)mod(m+1)}) > S_i/2$. And assume that it occurs at another u_j . And it implies that $w_G(u_j, u_{(m+1+j-1)mod(m+1)}) > S_i/2$. Then, which satisfies $w_G(u_i, u_{(i+1)mod(m+1)}) + w_G(u_j, u_{(m+1+j-1)mod(m+1)}) > S_i$. Which is a contradiction. Thus, It occurs at most once. ■

Algorithm and correctness

2

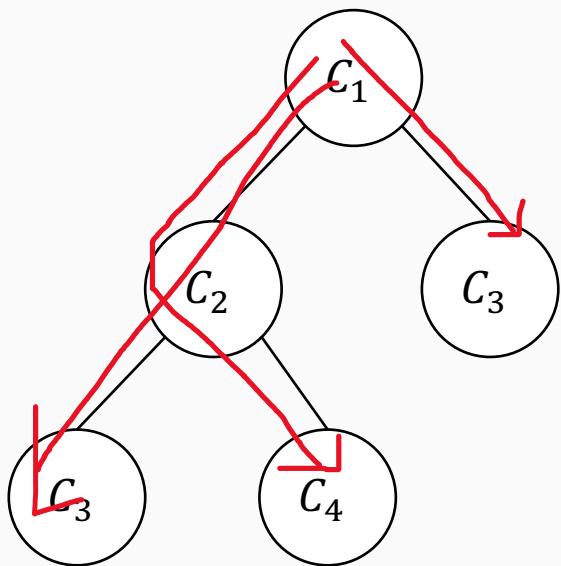
Correctness – Calculation in the cycle

Theorem 1. 위 알고리즘을 사이클 C_k 에 대하여 호출한 뒤, $B_k[0]$ 은 사이클 C_k 의 모든 정점을 순회하는 회로의 최소 비용이다.

Algorithm and correctness

2

STEP 2 – Lead to leaf



- We define a dynamic programming array $D[i] := \left(\max_{(c_j, c_i) \in E_C, \pi[i] \neq j} D[j] \right) + B_i[0] - \min(A_{0i}[m], A_{1i}[m]).$
- By optimizing this, We have an optimal longest path root to leaf.
- The time complexity is $O(|C| + |E_C|)$.

Fig. 8. An example of the path root to leaf.

Algorithm and correctness

2

STEP 2 – Lead to leaf

Algorithm 2 배열 D, π^2 을 채우는 알고리즘

- 1 **INPUT:** Tree $T = (C, E')$ and arrays A_{0k}, A_{1k}, B_k for all $C_k \in C$
- 2 **OUTPUT:** An array π^2
- 3 Let E, π^2 be an array with size $|C|$
- 4 $\forall C_i \in C: D[i] = 0$
- 5 **procedure** $DFS(u)$
- 6 **for** $(C_j, C_u) \in E' \wedge \pi[u] \neq j$, **do**
- 7 **call** $DFS(j)$
- 8 **if** $D[u] < D[j]$, **then**
- 9 $D[u] \leftarrow D[j]$
- 10 $\pi^2[u] \leftarrow j$
- 11
- 12 $D[u] \leftarrow D[u] + B_i[0] - \min(A_{0u}[m], A_{1u}[m])$

Algorithm and correctness

2

STEP 3 – Generate a walk

- Primary Idea: Construct a shortest walk through all vertices in C . And subtract by longest path to leaf.
- Let $Component(u)$ be a function k which satisfies $u \in C_k$
- Let $IsBelongOptimal(u)$ is a boolean function that decides belong to the optimal path.

Algorithm and correctness

2

STEP 3 – Generate a walk

Algorithm 3 최적의 보행 p 를 생성하는 알고리즘

```
1 INPUT: Tree  $T = (C, E')$  and Cactus graph  $G = (V, E)$ , an
   array  $\pi^2$  and for all  $C_k \in C$ ,  $A_{0k}, A_{1k}, B_k, \pi_k^1$ 
2 OUTPUT: Optimal walk  $p$ 
3 Let  $p$  be a dynamic array
4 procedure  $Move(S, m, u, v, b)$ 
5   if  $b = 1$ , then
6      $i \leftarrow u$ 
7     while  $i \neq v$ , do
8       call  $Push(S, i)$ 
9        $i \leftarrow (i + 1) \text{ mod } (m + 1)$ 
10      call  $Push(S, i)$ 
11    else
12       $i \leftarrow u$ 
13      while  $i \neq v$ , do
14        call  $Push(S, i)$ 
15         $i \leftarrow (m + i) \text{ mod } (m + 1)$ 
16        call  $Push(S, i)$ 
17  procedure  $DFS2(u)$ 
18     $\{u_0, \dots, u_m\} \leftarrow AllVertices(u)$ 
19    Let  $V$  be an array with size  $m + 1$ 
20    Let  $S$  be an stack
21
```

```
20  Let  $S$  be an stack
21   $\forall i \in \{0, \dots, m\}: V[i] = \text{false}$ 
22   $t \leftarrow \pi_u^1[0]$ 
23   $s \leftarrow t.\text{first}$ 
24  if  $IsBelongOptimal(u) \neq \text{true}$ , then
25    call  $Move(S, m, 0, t.\text{first}, t.\text{second})$ 
26     $t \leftarrow \pi_u^1[s]$ 
27    call  $Move(S, m, s, t.\text{first}, t.\text{second})$ 
28     $s \leftarrow t.\text{first}$ 
29    if  $A_{0u}[s] < A_{1u}[s]$ , then
30      call  $Move(S, m, s, 0, 0)$ 
31    else
32      call  $Move(S, m, s, 0, 1)$ 
33    else
34       $u_{min} \leftarrow 0, val_{min} \leftarrow \infty$ 
35
```

Algorithm and correctness

2

STEP 3 – Generate a walk

```
36  for  $i \leftarrow 1$  to  $m$ , do
37    if  $val_{min} > B_u[i]$ , then
38       $val_{min} \leftarrow B_u[i]$ 
39       $u_{min} \leftarrow i$ 
40       $t \leftarrow \pi_u^1[u_{min}]$ 
41      call  $Move(S, m, u_{min}, t.first, t.second)$ 
42       $s \leftarrow t.first$ 
43      if  $A_{0u}[s] < A_{1u}[s]$ , then
44        call  $Move(S, m, s, 0, 0)$ 
45      else
46        call  $Move(S, m, s, 0, 1)$ 
47    while  $IsEmpty(S) = false$ , do
48       $v \leftarrow Pop(S)$ 
49      call  $Pushback(p, u_v)$ 
50      if  $V[v] = false$ , then
51         $V[v] \leftarrow true$ 
52      for  $\forall (q, u_v) \in E$ , do
53        if  $Component(v_v) \neq Component(q)$ , then
54          call  $DFS2(q)$ 
```

Conclusion and its applications

Conclusion and its applications

1

Conclusion

- 1 NP-hard 문제로 예상되는 “Shortest weak Hamiltonian path problem”에 대해서 소개하고, 다행 시간에 결정론적으로 해결되는 인스턴스인 Cactus 그래프의 경우를 소개하고, 해법을 제시함.
- 2 Cactus 그래프를 부분 그래프로 가지는 그래프에 대한 효율적인 휴리스틱의 제시 가능성.

References

1. M.Sohel Rahman and M. Kaykobad. 2005. On Hamiltonian cycles and Hamiltonian paths. *Information Processing Letters* 94, 1 (2005), 37–41. DOI:<http://dx.doi.org/10.1016/j.ipl.2004.12.002>
2. K. Das, “Cactus Graphs and Some Algorithms”, arXiv:1408.4005, 2014.
3. Anon. 2020. Cactus graph. (November 2020). Retrieved December 8, 2020 from https://en.wikipedia.org/wiki/Cactus_graph (Image)

본 발표 자료 (*.pptx), Minor reversion
변경 사항은 다음 링크에서 확인하실 수
있습니다.

<https://github.com/ANEP-Research/KSC2020>