JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 26, No. 1: 128~133, Jan. 2022

거리공간속 경로 그래프에 간선추가를 통한 지름의 최소화

김재훈*

Minimizing the Diameter by Augmenting an Edge to a Path in a Metric Space

Jae-Hoon Kim*

*Professor, Department of Computer Engineering, Busan University of Foreign Studies, Busan, 46234 Korea

요 약

본 논문은 거리 공간(metric space) 속에 포함된 그래프에서 각 간선의 가중치가 거리 공간 상의 두 끝 정점간의 거리로 주어지는 그래프를 다룬다. 특별히 우리는 이러한 그래프 중 n개 정점을 가진 경로 P에 관해서 연구한다. 우리는 경로 P에 하나의 간선을 추가해서 새로운 그래프 $\overline{P}$ 얻을 수 있다. 그러면 그래프 $\overline{P}$의 두 정점 사이의 최단 경로의 길이를 생각하고 이 길이들 중 최댓값에 주목한다. 이 최댓값을 그래프 $\overline{P}$의 지름(diameter)라고 부른다. 우리는 그래프 $\overline{P}$의 지름이 최소가 되도록 추가하는 간선을 찾고 싶다. 특별히 임의의 실수 $\lambda > 0$에 대해서, $\overline{P}$의 지름이 λ 이하가 되는 추가 간선이 존재하는지 여부를 결정하는 문제에 대해 $O(n)$ 시간 알고리즘을 제안한다. 이것은 이전 알려진 시간복잡도 $O(n\log n)$을 개선한다. 이 결정 알고리즘을 이용해서 주어진 경로 P의 길이 D에 대해서, $\overline{P}$의 지름의 최솟값을 찾는 $O(n\log D)$ 시간 알고리즘을 제안한다.

ABSTRACT

This paper deals with the graph in which the weights of edges are given the distances between two end vertices on a metric space. In particular, we will study about a path P with n vertices for these graphs. We obtain a new graph $\overline{P}$ by augmenting an edge to P. Then the length of the shortest path between two vertices on $\overline{P}$ is considered and we focus on the maximum of these lengths. This maximum is called the diameter of the graph $\overline{P}$. We wish to find the augmented edge to minimize the diameter of $\overline{P}$. Especially, for an arbitrary real number $\lambda > 0$, we should determine whether the diameter of $\overline{P}$ is less than or equal to λ and we propose an $O(n)$-time algorithm for this problem, which improves on the time complexity $O(n\log n)$ previously known. Using this decision algorithm, for the length D of P, we provide an $O(n\log D)$-time algorithm to find the minimum of the diameter of $\overline{P}$.

키워드 : 거리 공간, 그래프, 경로, 지름, 알고리즘

Keyword : Metric space, Graph, Path, Diameter, Algorithm

Received 14 October 2021, **Revised** 22 October 2021, **Accepted** 8 December 2021

*** Corresponding Author** Jae-Hoon Kim(E-mail: jhoon@bufs.ac.kr, Tel:+82-51-509-6226)
Professor, Department of Computer Engineering, Busan University of Foreign Studies, Busan, 46234 Korea

Open Access http://doi.org/10.6109/jkiice.2022.26.1.128 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

간선들이 가중치를 가진 그래프 $G=(V,E)$ 상의 임의의 두 정점 v와 w 사이의 거리 $d(v,w)$는 v와 w 사이의 최단 경로의 간선들의 가중치 합으로 정의한다. 그러면 그래프 G의 지름(diameter)는 모든 정점 쌍 (v,w)에 대한 거리 $d(v,w)$의 최댓값으로 정의한다.

본 논문에서는 거리 공간 (metric space) 속의 n개 정점들을 가진 경로 그래프 P를 생각한다. 거리 공간에서 두 점 v와 w사이의 거리를 $\|vw\|$로 나타내면, 임의의 세 점 x, y, z에 대해서 삼각 부등식 $\|xz\| \leq \|xy\| + \|yz\|$이 성립한다. 경로 P의 n개의 정점 $v_1, v_2, ..., v_n$에 대해서, 정점 v_i과 v_{i+1}사이에 간선이 존재하고, 간선의 가중치는 $\|v_iv_{i+1}\|$로 주어진다. 우리는 경로 P에서 인접하지 않은 두 정점 v_i와 v_j사이에 간선을 추가할 것이다. 이 간선의 가중치는 $\|v_iv_j\|$가 된다. 이 간선이 추가된 그래프를 $\overline{P}$라고 할 때, $\overline{P}$의 지름이 최소가 되는 간선 $\overline{v_iv_j}$를 찾을 것이다.

우리가 추가하는 간선 $\overline{v_iv_j}$은 두 정점 v_i와 v_j를 잇는 지름길(shortcut)이라고 생각할 수 있고, 본 논문에서 다룰 문제는 경로 P에 지름길을 추가해서 가장 먼 두 정점 사이의 거리를 최소화하는 문제라 할 수 있다. 특별히 임의의 실수 $\lambda > 0$에 대해서, 지름길 $\overline{v_iv_j}$을 추가했을 때, $\overline{P}$의 지름이 λ보다 작거나 같은지 여부를 알아내는 문제를 생각할 것이다. 이 문제를 결정 문제라고 부른다.

Ⅱ. 관련 연구

[1]에서 저자들은 거리 공간 속에 경로와 트리 그래프에서 하나의 간선을 추가해서 그래프의 지름이 최소가 되게 하는 문제를 연구하였다. 그들은 그래프가 경로인 경우 지름이 최소가 되는 지름길을 구하는 문제에 대한 $O(n\log^3 n)$ 시간 알고리즘을 제안하고, 그래프가 트리인 경우에 $O(n^2\log n)$ 시간 알고리즘을 제안한다. 또한 거리 공간이 R^d인 경우에 경로에 대한 문제를 푸는 $O(n+1/\epsilon^2)$ 시간 $(1+\epsilon)$-근사 알고리즘을 제안한다. 그들은 경로에 대해서, $\lambda > 0$가 주어질 때, 결정 문제를 $O(n\log n)$ 시간에 해결한다. 그리고 Megiddo의 매개변수 탐색(parametric search) [2]을 이용해서 지름의 최소화를 찾는 최적화 문제를 $O(n\log^3 n)$ 시간에 해결한다.

[3]에서는 공간 속의 트리에 대한 문제의 [1]에서의 결과를 개선한다. 저자들은 최적화문제에 대한 $O(n\log n)$ 시간 알고리즘을 제안하고, $O(n+1/\epsilon \log 1/\epsilon)$ 시간 $(1+\epsilon)$-근사 알고리즘을 제안하다. 또한 저자들은 일반적인 가중치 간선을 가진 트리에 대한 문제를 연구한다. 이 문제에 대해서는 [4]에서 $O(n^2\log^3 n)$ 시간 알고리즘이 제안되었다. [3]의 저자들은 이 결과를 개선하는 $O(n^2)$ 시간 알고리즘을 제안한다.

보다 일반적이고 많은 변형 문제들이 연구되어 왔다 [5, 6, 7, 8, 9]. 특별히 음이 아닌 가중치의 간선을 가진 일반 그래프 G에 k개의 새로운 간선을 추가해서 그래프의 지름을 최소화는 문제를 생각할 수 있다. 하지만 그 문제는 NP-hard 임이 증명되었다 [10]. 또한 일반 그래프 문제에 대한 근사 알고리즘들이 제안되었다 [6, 7].

그래프 G의 지름을 최소화하기 위해 간선을 추가하는 문제에서 그래프 G가 평면 위에 주어지고 추가되는 간선의 두 끝점이 그래프 G 위에 어느 곳에도 위치할 수 있는 경우를 생각할 수 있다. 다시 말해서, 간선의 두 끝점이 기존 그래프 G의 간선 위에 임의의 위치에 놓일 수 있는 경우이다. 이 문제를 연속(continuous) 버전이라 한다. 연속 버전에 대해서, [11]의 저자는 지름을 최소화하기 위해 경로에 간선을 추가하는 문제를 연구하고, 근사 알고리즘을 제안한다. [12]에서 저자들은 연속 버전에서 평면 위의 사이클에 대한 문제를 연구한다. 그들은 사이클의 지름을 최소화하기 위해 추가하는 두 간선을 찾는 문제를 해결한다. 또한 그들은 사이클이 볼록한 경우에 최적의 두 간선을 찾는 $O(n)$ 시간 알고리즘을 제안한다.

간선을 추가해서 그래프의 반지름을 최소화하는 문제도 생각해볼 수 있다. 그래프의 어떤 정점에서 가장 먼 정점까지의 거리가 최소가 되는 정점을 중심(center)이라고 하고, 중심에서 가장 먼 정점까지의 거리를 그래프의 반지름이라고 한다. [13]에서 저자들은 중심이 간선의

중간에 놓일 수 있는 연속 버전을 생각한다. 그들은 거리 공간 속 경로에 대한 문제를 푸는 $O(n)$ 시간 알고리즘을 제안한다. 중심이 정점에 놓여야 하는 경우에 대해서는 [14]에서 역시 $O(n)$ 시간 알고리즘을 제안한다.

Ⅲ. 정의

거리 공간 속 n개 정점들을 가진 경로 $P=(p_1,...,p_n)$가 주어지고, 정점들은 배열 $P[1,...,n]$에 저장된다고 가정한다. 다시 말해서, 모든 $i(1\le i\le n)$에 대해서, $p_i=P[i]$. 특별히, 경로의 시작 정점 p_1과 끝 정점 p_n을 각각 s와 e로 나타낸다. 여기서 우리는 논문 [1]의 정의와 표기법을 따를 것이다.

$1\le k<h\le n$에 대해서, P의 부분경로 $(p_k,...,p_h)$를 $P[k,h]$로 나타낸다. 또한 경로 $P[k,h]$의 양 끝점을 간선 $\overline{p_kp_h}$으로 연결해서 얻는 사이클을 $C[k,h]$로 나타낸다. 결과적으로 경로 P에 간선 $\overline{p_kp_h}$을 추가해 얻어진 그래프를 $\overline{P}[k,h]$로 나타낸다. δ_G를 그래프 G의 두 정점사이의 거리를 나타낸다고 하면, $\delta_{\overline{P}[k,h]}$를 $\overline{p}_{k,h}$로 $\delta_{C[k,h]}$를 $c_{k,h}$로 나타낸다. 그러면 $M(k,h)$를 다음과 같이 정의한다:

$$M(k,h)=\max_{1\le x<y\le n}\overline{p}_{k,h}(x,y). \quad (1)$$

$M(k,h)$는 간선이 추가된 그래프 $\overline{P}[k,h]$의 지름이다. 따라서 우리는 이 지름이 최소가 되게 추가되는 간선 $\overline{p_kp_h}$를 찾고 싶다. 이때의 최소 지름을 $m(\overline{P})$로 나타낸다. 우선 우리는 임의의 실수 $\lambda>0$에 대해서, $m(\overline{P})\le\lambda$ 인지 여부를 결정하는 문제를 다룰 것이다.

우리는 네 개의 함수 $S(k,h)$, $E(k,h)$, $U(k,h)$, $O(k,h)$을 활용할 것이다. 각 함수들은 다음과 같이 정의된다:

$$S(k,h)=\max_{k\le x\le h}\overline{p}_{k,h}(s,x), \quad (2)$$

$$E(k,h)=\max_{k\le x\le h}\overline{p}_{k,h}(x,e), \quad (3)$$

$$U(k,h)=\overline{p}_{k,h}(s,e), \quad (4)$$

$$O(k,h)=\max_{k\le x<y\le h}c_{k,h}(x,y). \quad (5)$$

그러면 $M(k,h)$은 네 개의 함수 값 $S(k,h)$, $E(k,h)$, $U(k,h)$, $O(k,h)$의 최댓값임을 알 수 있다.

Ⅳ. 알고리즘

본 장에서 우리는 임의의 실수 $\lambda>0$에 대해서, $m(\overline{P})\le\lambda$ 를 결정하는 알고리즘을 생각할 것이다. 그러면 어떤 k,h가 존재해서, $M(k,h)\le\lambda$를 만족해야 한다. 따라서 알고리즘은 각 $1\le k<n$에 대해서, $M(k,h)\le\lambda$가 되는 어떤 $k<h\le n$가 존재하는지 여부를 결정할 것이다.

우선 경로 P의 가중치들로 정의되는 전위 합(prefix sum)을 생각한다. $PSum[i]$를 정점 p_1과 p_i사이의 간선들의 가중치 합이라고 하자. 모든 전위 합은 $O(n)$ 시간에 계산할 수 있다. 정점 p_i와 p_j 사이의 부분합 $BSum[i,j]$은 두 정점사이의 간선들의 가중치 합으로 정의한다. 그러면 $BSum[i,j]=PSum[j]-PSum[i]$와 같이 구할 수 있다.

이전 장의 네 개의 함수 중 우선 $U(k,h)$를 생각한다. $U(k,h)$는 간선 $\overline{p_kp_h}$를 추가했을 때, s와 e 사이의 거리이다. 위의 부분 합을 활용하면 다음과 같이 구할 수 있다:

$$U(k,h)=BSum[s,k]+\parallel p_kp_h\parallel+BSum[h,e]. \quad (6)$$

또한 $U(k,h)\ge U(k,h+1)$이고, $U(k,h)\le U(k+1,h)$임을 알 수 있다. $k=1$에 대해서, h는 n부터 작아지는 순서로 $U(1,h)$를 위 식 (6)에 의해서 계산한다. 그래서 $U(1,h)\le\lambda$를 만족하는 가장 작은 정수 h를 찾아서 $u(1)$로 나타낸다. $k>1$에 대해 $u(k)$를 구하고, $k+1$일 때를 생각해보자. h는 $u(k)$이하만을 고려해도 충분하다. 왜냐하면, $\lambda<U(k,u(k)+1)\le U(k+1,u(k)+1)$ 이기 때문이다. 따라서 모든 $1\le k<n$에 대해서, $U(k,u(k))\le\lambda$를 만족하는 $u(k)$를 찾는데 총 $O(n)$ 시간이면 충분하다. 아래 그림 1을 참조한다.

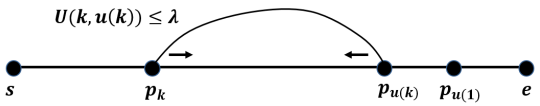

Fig. 1 compute $u(k)$

다음은 $S(k,h)$를 생각한다. $S(k,h)$는 간선 $\overline{p_k p_h}$를 추가했을 때, 정점 p_k, ..., p_h 중에서 s로부터 가장 먼 거리를 나타낸다. 우선 $PSum[i] \le \lambda$인 가장 큰 정수 i를 찾고 이를 α로 나타낸다. 이 정수를 찾는데 $O(n)$시간이면 충분하다. 또한 $S(k,h) \le S(k,h+1)$이고, $S(k,h) \le S(k+1,h)$임을 알 수 있다. $k=1$에 대해서, h는 $\alpha+1$부터 증가하는 순서로 고려해서, $S(1,h) \le \lambda$인 가장 큰 정수 h를 찾아서 $s(1)$로 나타낸다. $k>1$에 대해서, $s(k)$를 구한 후, $k+1$의 경우를 생각한다. h는 $s(k)$에서부터 작은 순서로 고려한다. 왜냐하면, $\lambda < S(k,s(k)+1) \le S(k+1,s(k)+1)$이기 때문이다. 결과적으로 h는 $s(1)$부터 작은 순서로 α까지 고려된다. 모든 $1 \le k < \alpha$에 대해서, $S(k,s(k)) \le \lambda$를 만족하는 $s(k)$를 찾는데 총 $O(n)$시간이면 충분하다. 아래 그림 2를 참조한다.

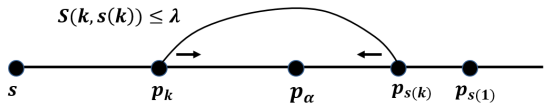

Fig. 2 compute $s(k)$

$E(k,h)$를 생각해보면, $E(k,h)$는 간선 $\overline{p_k p_h}$를 추가했을 때, 정점 p_k, ..., p_h 중에서 e로부터 가장 먼 거리를 나타낸다. 이것은 위의 $S(k,h)$와 대칭적으로 생각할 수 있다. $BSum[j,n] \le \lambda$를 만족하는 가장 작은 정수 j를 찾고 이를 β라고 한다. $h=n$인 경우, $E(k,n) \le \lambda$인 가장 작은 정수 k를 찾고 이를 $g(n)$로 나타낸다. $E(k,h) \ge E(k+1,h)$이기 때문에 $l=1,...,g(n)-1$에 대해서, $E(l,n) > \lambda$를 만족한다. h를 n부터 작은 순서로 고려하고, h에 대한 $g(h)$를 찾은 후, $h-1$의 경우, $\lambda < E(g(h)-1,h) \le E(g(h)-1,h-1)$이기 때문에 $g(h-1)$은 $g(h)$이상에서 찾을 수 있다. 따라서 k는 $g(n)$부터 큰 순서로 β까지 고려된다. 결과적으로 모든 $\beta < h \le n$에 대해서, $E(g(h),h) \le \lambda$를 만족하는 $g(h)$를 찾는데 총 $O(n)$시간이면 충분하다. 아래 그림 3을 참조한다.

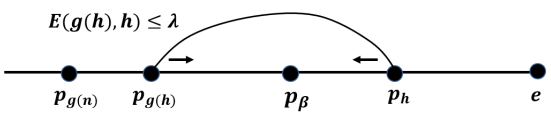

Fig. 3 compute $g(k)$

마지막으로 함수 $O(k,h)$를 생각한다. 이 함수는 간선 $\overline{p_k p_h}$를 추가했을 때, 사이클 $C[k,h]$의 임의의 두 정점간의 거리의 최댓값이다. 다시 말해서, $C[k,h]$의 지름이다. 각 $1 \le k < n$에 대해서, $O(k,h) \le \lambda$인 가장 큰 정수 h를 찾아서 $c(k)$로 나타낼 것이다.

우선 우리는 경로 P상의 각 정점 p에 대해서, p 이후의 정점 중 p로부터의 거리가 λ를 최초로 초과하는 정점을 생각하고, 이 정점을 $Next(p)$로 나타낸다. 또한 사이클 $C[k,h]$의 임의의 두 정점 u, v에서, 둘 사이의 경로 중, 간선 $\overline{p_k p_h}$를 지나는 경로의 길이를 두 정점의 역거리(reverse distance)라고 부른다.

k를 고정하고 h를 $k+1$부터 증가하는 순서대로 하나씩 고려할 것이다. 이 때, $k \le p$이고 $Next(p) \le h$인 각 정점 p에 대해서, p와 $Next(p)$ 사이의 거리를 d할 때, 덱 DQ의 tail에 저장된 거리가 d보다 큰 것들은 delete 하고 tail에 (p,d)를 저장한다.

사이클 $C[k,h]$의 지름이 λ이하임을 검사하기 위해서, 덱 DQ의 head에 저장된 $(\bar{p},\bar{d})$에 대해서, $\bar{p}$와 $Next(\bar{p})$ 사이의 역 거리가 λ이하임을 검사한다. 왜냐하면, 덱 DQ에 저장된 거리들이 head부터 tail 방향으로 증가하는 순서로 저장되어 있으므로, 역 거리는 반대로 감소하는 순서가 된다. 따라서 DQ의 head에 저장된 정점의 역 거리가 가장 크다. 그러므로 위의 역 거리가 λ이하이면, 사이클 $C[k,h]$의 지름이 λ이하임을 알 수 있다 (그림 4). 따라서 이 검사에서 $\bar{p}$와 $Next(\bar{p})$ 사이의 역 거리가 λ를 초과하면, 이때의 h에 대해서 $c(k)=h-1$이 된다. 또한 k가 덱 DQ의 head에 저장된 $\bar{p}$이상이 되면 DQ의 head를 delete 한다. 이 과정은 덱 DQ를 활용해서 전체적으로 $O(n)$시간에 수행할 수 있다.

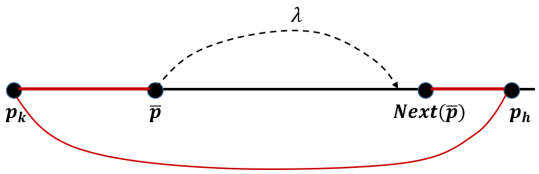

Fig. 4 reverse distance

Table. 1 Pseudo code of the decision algorithm

```
01: procedure ShortCut(λ)
02: Compute an array u[], where u[k] is the smallest vertex
    index s.t. U(k, u[k]) ≤ λ, for k = 1, ..., n
03: Compute an array s[], where s[k] is the largest
    vertex index s.t. S(k, s[k]) ≤ λ, for k = 1, ..., n
04: Compute an array c[], where c[k] is the largest
    vertex index s.t. O(k, c[k]) ≤ λ, for k = 1, ..., n
05: Compute an array g[], where g[k] is the smallest
    vertex index s.t. E(g[k], k) ≤ λ, for k = n, ..., 1
06: for each k from 1 to n do
07:     if k < g[n] then
08:         return false
09:     else
10:         if min(s[k], c[k]) < u[k] then
11:             return false
12:         else
13:             h′← u[k]
14:             k′← g[h′]
15:             if k < k′ then
16:                 return false
17:             else
18:                 return true
19:             end if
20:         end if
21:     end if
22: end for
```

따라서 종합적으로, 각 $1 \le k < n$에 대해서, $k < g(n)$이면, $E(k,h) \ge E(k,n) > \lambda$이므로 모든 $k < h \le n$에 대해서, $M(k,h) > \lambda$. 따라서 $k \ge g(n)$라고 가정하자. 우선 $\min(s(k), c(k)) < u(k)$이면, $M(k,h) \le \lambda$를 만족하는 h는 존재하지 않는다. 따라서 $\min(s(k), c(k)) \ge u(k)$라면, $h' = u(k)$, $k' = g(h')$라 하고, $k < k'$이면, $M(k,h) \le \lambda$를 만족하는 h는 존재하지 않는다. $k \ge k'$이면, 모든 $k < h \le h'$에 대해서, $M(k,h) \le \lambda$. 구체적으로, 위의 표 1과 같은 알고리즘을 얻을 수 있다.

정리 4. 1 거리 공간 안의 n개 정점들을 가진 경로 P가 주어질 때, 임의의 실수 $\lambda > 0$에 대해서, 간선을 추가했을 때 결과 그래프의 지름이 λ이하인지 결정하는 $O(n)$ 시간 알고리즘이 존재한다.

주어진 경로 P에 대해서, 시작 정점 s와 끝 정점 e사이의 거리 $d(s,e)$를 D라고 하자. 간선 추가 시 결과 그래프의 지름의 최솟값을 λ^*라고 하면, $\lambda^* \le D$이고, 따라서 구간 $(0, D]$ 안에서 이진 탐색(binary search)을 이용해서 λ^*를 찾을 수 있다. 위의 결정 알고리즘을 활용해서 이진 탐색을 $O(n \log D)$ 시간 안에 수행할 수 있다.

보조 정리 4. 2 거리 공간 안의 n개 정점들을 가진 경로 P가 주어지고 시작 정점 s와 끝 정점 e사이의 거리를 D라고 할 때, 간선을 추가했을 때 결과 그래프의 지름의 최솟값을 찾는 $O(n \log D)$ 시간 알고리즘이 존재한다.

Ⅴ. 결 론

본 논문에서는 거리 공간 속의 n개 정점들을 가진 경로 P가 주어지고, 각 간선의 가중치는 두 정점간의 거리 공간의 거리로 주어진다. 경로 P에 하나의 간선을 추가해서 새로운 그래프 $\overline{P}$를 얻을 때, 그래프 $\overline{P}$의 지름이 최소가 되는 추가 간선을 찾는 문제를 다룬다. 임의의 실수 $\lambda > 0$에 대해서, 간선을 추가했을 때 그래프 $\overline{P}$의 지름이 λ이하가 되는지 여부를 결정하는 $O(n)$ 시간 알고리즘을 제안한다. 이 알고리즘은 이전 연구 [1]의 $O(n \log n)$ 시간 복잡도를 개선한다. 그리고 주어진 경로 P의 길이 D에 대해서, $\overline{P}$의 지름의 최솟값을 찾는 $O(n \log D)$ 시간 알고리즘을 제안한다. 향후에 거리 공간 속에 있지 않고 일반적인 간선 가중치를 가진 경로에서 대해서 지름을 최소화하는 간선 추가의 문제를 연구할 수 있을 것이다.

ACKNOWLEDGEMENT

This work was supported by the research grant of the Busan University of Foreign Studies in 2021

REFERENCES

[1] U. Grobe, C. Knauer, F. Stehn, J. Gudmundsson, and M. Smid, “Fast algorithms for diameter-optimally augmenting paths and trees,” *International Journal of Foundations of*

Computer Science, vol. 30, pp. 293-313, Jan. 2019.

[2] N. Megiddo, “Applying parallel computation algorithms in the design of serial algorithms,” *Journal of the ACM*, vol. 30, pp. 852-865, Jan. 1983.

[3] D. Bilo, “Almost optimal algorithms for diameter-optimally augmenting trees,” in *Proceedings of the 29th International Symposium on Algorithms and Computation*, pp. 40:1-40:13, 2018.

[4] E. Oh and H. K. Ahn, “A near-optimal algorithm for finding an optimal shortcut of a tree,” in *Proceedings of the 27th International Symposium on Algorithms and Computation*, pp. 59:1-59:12, 2016.

[5] N. Alon, A. Gyarfas, and M. Ruszinko, “Decreasing the diameter of bounded degree graphs,” *Journal of Graph Theory*, vol. 35, pp. 161-172, Sep. 2000.

[6] D. Bilo, L. Guala, and G. Proietti, “Improved approximability and non-approximability results for graph diameter decreasing problem,” *Theoretical Computer Science*, vol. 417, pp. 12-22, Feb. 2012.

[7] F. Frati, S. Gaspers, J. Gudmundsson, and L. Mathieson, “Augmenting graphs to minimize the diameter,” *Algorithmica*, vol. 72, pp. 995-1010, May. 2015.

[8] D. Garijo, A. Marquez, N. Rodriguez, and R. I. Silveira, “Computing optimal shortcuts for networks,” *European Journal of Operational Research*, vol. 279, pp. 26-37, Nov. 2019.

[9] H. Wang and Y. Zhao, “Algorithms for diameters of unicycle graphs and diameter-optimally augmenting trees,” in *Proceedings of the 15th International Conference and Workshops on Algorithms and Computation*, pp. 27-39, 2021.

[10] A. A. Schoone, H. L. Bodlaender, and J. V. Leeuwen, “Diameter increase caused by edge deletion,” *Journal of Graph Theory*, vol. 11, pp. 409-427, Mar. 1987.

[11] B. Yang, “Euclidean chains and their shortcuts,” *Theoretical Computer Science*, vol. 497, pp. 55-67, July. 2013.

[12] J. L. De Carufel, C. Grimm, A. Maheshwari, and M. Smid, “Minimizing the continuous diameter when augmenting paths and cycles with shortcuts,” *Computational Geometry*, vol. 89, pp. 409-427, Aug. 2020.

[13] C. Johnson and H. Wang, “A linear-time algorithm for radius-optimally augmenting paths in a metric space,” in *Proceedings of the 16th Algorithms and Data Structures Symposium*, pp. 466-480, 2019.

[14] H. Wang and Y. Zhao, “A linear-time algorithm for discrete radius optimally augmenting paths in a metric space,” *International Journal of Computational Geometry & Applications*, vol. 30, pp. 167-182, Sep. 2020.

김재훈(Jae-Hoon Kim)

1994 서강대학교 수학과 이학사
1996 KAIST 수학과 이학석사
2003 KAIST 전산과 공학박사
2003~ 부산외국어대학교 컴퓨터공학과 교수
※관심분야 : 알고리즘, 최적화, 스케줄링