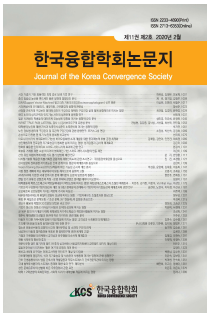

한국융합학회논문지
제11권 제2호

ISSN : 2233-4890(Print)

시간 가중치 기반 효율적인 최적 경로 탐색 기법 연구

허유성, 김태우, 안용학

To cite this article : 허유성, 김태우, 안용학 (2020) 시간 가중치 기반 효율적인 최적 경로 탐색 기법 연구, 한국융합학회논문지, 11:2, 1-8

Journal of the Korea Convergence Society
Vol. 11. No. 2, pp. 1-8, 2020

ISSN 2233-4890 / eISSN 2713-6353
https://doi.org/10.15207/JKCS.2020.11.2.001

시간 가중치 기반 효율적인 최적 경로 탐색 기법 연구

허유성[1], 김태우[2], 안용학[3*]
[1]세종대학교 컴퓨터공학과 학사과정, [2]세종대학교 컴퓨터공학과 학사, [3]세종대학교 컴퓨터공학과 교수

Search Algorithm for Efficient Optimal Path based on Time-weighted

Yu-sung Her[1], Tae-woo Kim[2], Yonghak Ahn[3*]
[1]Student, Department of Computer Science, Sejong University,
[2]Bachelor, Department of Computer Science, Sejong University,
[3]Professor, Department of Computer Science, Sejong University

요 약 본 논문에서는 시간 가중치를 적용하여 각 노드들 간의 중간지점까지의 최적 경로 탐색 기법을 제안한다. 중간지점을 이용하는 서비스들은 주로 사용자들의 위치를 기반으로 하여 제공한다. 위치 기반 탐색 방법은 단순히 위치에 대해서만 고려하기 때문에 시간의 측면에서 효율적이지 못하다는 문제점이 있다. 제안된 방법은 기존의 위치 기반 탐색 방법의 문제점을 해결하기 위해 사용자들의 위치와 중간지점까지의 소요시간을 반영함으로써 각 노드와 중간지점까지의 가중치를 설정하고, 이를 활용하여 최적의 경로를 탐색할 수 있다. 또한, 탐색의 효율성을 증대하기 위해 주어지는 정보들에 적응적으로 가중치를 설정함으로써 높은 정확성을 갖도록 한다. 실험 결과, 제안된 방법은 기존 방법에 비해 중간지점까지의 최적 경로를 효과적으로 찾을 수 있음을 확인하였다.

주제어 : 융합, 최적 경로 탐색, 적응형 가중치, 실시간 정보, 중간지점 탐색, 다각형 경계

Abstract In this paper, we propose an optimal path search algorithm between each node and midpoint that applies the time weighting. Services for using a location of mid point usually provide a mid point location-based on the location of users. There is a problem that is not efficient in terms of time because a location-based search method is only considered for location. To solve the problem of the existing location-based search method, the proposed algorithm sets the weights between each node and midpoint by reflecting user's location information and required time. Then, by utilizing that, it is possible to search for an optimum path. In addition, to increase the efficiency of the search, it ensures high accuracy by setting weights adaptively to the information given. Experimental results show that the proposed algorithm is able to find the optimal path to the midpoint compared with the existing method.

Key Words : Convergence, Optimal Path Search, Adaptive Weight, Real Time Data, Mid-Point Search, Polygon Boundary

This research was supported by the Korean MSIT(Ministry of Science and ICT), under the National Program for Excellence in SW (2015-0-00938), supervised by the IITP (Institute of Information & communications Technology Planning&evaluation).
*Corresponding Author : Yonghak Ahn(yohans@sejong.ac.kr)
Received October 28, 2019 Revised January 28, 2020
Accepted February 20, 2020 Published February 28, 2020

1. 서론

사람들과 약속을 잡는 것이 일상화된 현재, 많은 사람들이 약속 장소를 정하는 것에 많은 노력과 시간을 들이고 있다. 약속 장소를 정하는 것에 있어, 사용자 간의 중간 지점을 구하는 것은 필수적이다. 시중에 나와 있는 약속 장소를 정해주는 서비스들은 주로 사용자들의 위치를 기반으로 중간지점을 제공하는 위치 기반 중간지점 탐색 기능을 이용하고 있다[1,2]. 이러한 위치 기반 중간지점 탐색 기능은 단순히 위치에 대해서만 고려하기 때문에 시간의 측면에서 효율적이지 못하다. 효율적으로 중간지점을 정하기위해서는 위치뿐만 아니라 소요되는 시간에 대해서도 고려해야 한다. 본 논문에서는 소요되는 시간에 대해서도 효율적인 중간지점을 도출하기위한 시간 가중치 기반 효율적인 최적 경로 탐색 방법을 제안한다.

최적 경로 탐색 알고리즘은 사용자에게 최적의 경로를 제공하는 서비스이다. 최적 경로는 소요시간, 이동거리, 이동속도, 안전성, 정시성 등을 감안하여 산정한 최적 통행 비용을 기준으로 이용자에게 가장 적절한 경로를 제공한다. 이 중 이동거리를 제외한 요소들은 대부분은 실시간 상황에 따라 변하기 때문에 가장 합리적인 최적경로의 산정은 주기적이며 실시간으로 수집된 정보에 의해 산출되어 반영되어야 한다[3-5]. 특히나 이용자가 다수일 경우 이용자마다 실시간으로 산출되는 정보를 반영하여 최적 통행 비용을 산출해야 하므로 많은 어려움이 따른다[1,6].

최근 들어 최적 경로 탐색을 위한 다양한 연구들이 진행되고 있는데, 이 중 다수의 이용자를 대상으로 각 이용자들의 최적 경로를 탐색하는 연구는 출발 위치를 기반으로 하여 중간지점을 찾는 방법이나 위치 기반 삼각형의 중심 좌표를 구하는 방법[4,7], 위치 좌표의 평균을 이용하여 근거리의 지점을 찾아주는 방법[7] 등이 있다.

하지만 출발 위치 기반의 중간지점을 찾는 방법은 단순히 이동거리만을 반영하므로 실시간 정보에 대한 반영이 미흡하며, 위치 기반의 삼각형 무게 중심을 구하는 방법은 거리 기반의 중간지점을 찾기 때문에 최적 통행 비용을 산출하기가 어렵다. 위치 좌표의 평균을 이용하는 방법은 평균값으로 거리에 대한 근사치를 얻을 수 있지만 각 출발지로부터 중간지점까지의 이동시간에 대한 정보를 반영하지 못한다.

따라서 본 논문에서는 이러한 문제점들을 해결하기 위해 각 사용자들의 출발지에 해당하는 노드로부터 거리 기반으로 산출된 중간지점에 시간 가중치를 적응적으로 적용하여 각 시작 노드로부터 중간지점 노드까지의 최적 경로 탐색이 가능한 방법을 제안한다.

2. 시간 가중치 기반 최적 경로 탐색

본 논문에서는 각 시작 노드부터 중간지점 노드까지의 최적 경로를 탐색하기 위한 정보로 거리뿐만 아니라 실시간 이동 정보를 반영하므로 실제 이동에 대한 시간을 고려함으로써 최적의 경로를 탐색할 수 있는 방법을 제안한다.

제안된 방법은 먼저, 효율적인 최적 경로 탐색을 위한 경계 범위를 설정하고, 적응형 시간 가중치를 기반으로 최적 통행 비용을 가지는 중간지점을 선정한다.

2.1 중간지점 탐색을 위한 경계범위 설정

효율적인 중간지점 탐색을 위하여 경계 범위 설정이 필요하다. 중간지점 탐색은 바깥 이용자들의 위치를 이어 형성된 다각형 범위 안에서만 수행된다. 따라서 본 논문에서는 다양한 Convex Hull 알고리즘[8] 중 그라함 스캔(Graham's Scan) 알고리즘[9,10]을 사용하여 탐색 범위를 제한한다. 해당 알고리즘은 2차원 평면상의 다수의 점들이 존재할 때, 다른 점을 가둘 수 있는 볼록(Convex) 다각형을 이루는 외곽 점들을 찾는 알고리즘이다. 동작 과정은 먼저, 무작위로 구성된 점이 주어지면 제일 왼쪽에 있는 점을 기준점으로 선택하여 다른 점과의 각도를 구한 후 정렬한다. 그 다음 정렬된 점을 순서대로 이은 두 선분씩 비교하는데, CCW(Counterclockwise) 방식으로 두 선분의 방향성을 외적을 이용하여 비교해나가면서 경계를 이루는 좌표들을 선별한다. CCW는 다음 〈수식 1〉과 같다.

$$
\begin{aligned}
&A=(x_1,y_1,0)\ B=(x_2,y_2,0)\ C=(x_3,y_3,0)\\
&\vec{u}:\overrightarrow{AB}\quad \vec{v}:\overrightarrow{AC}\\
&u\times v=(0,0,(x_2-x_1)(y_3-y_1)\\
&\qquad\qquad -(x_3-x_1)(y_2-y_1))\\
&d=(x_2-x_1)(y_3-y_1)-(x_3-x_1)(y_2-y_1)\\
&d>0:\text{반시계방향}\\
&d=0:\text{평행}\\
&d<0:\text{시계방향}
\end{aligned}
\tag{1}
$$

여기서 A,B,C는 z=0인 평면 위의 좌표, u는 점 A와 B의 벡터, v는 A와 C의 벡터, ×는 외적의 연산 기호, d는 xy 평면에 수직인 벡터 성분이다.

본 논문에서는 사용자들의 위치가 점으로 주어졌을 때 반시계 방향으로 외각 점들을 찾는 과정을 수행한다. 전체 동작 과정은 Fig. 1과 Fig. 2와 같다.

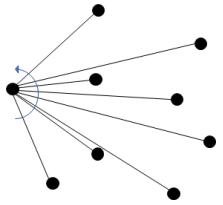

Fig. 1. Angle Sequential Alignment

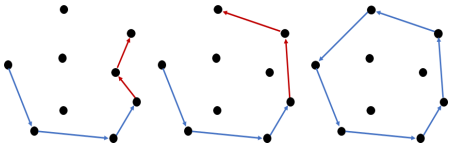

Fig. 2. Application Process of Graham's Scan

Fig. 1은 제일 왼쪽 점을 기준으로 반시계 방향으로 좌표들을 정렬하는 것을 나타내고, Fig. 2는 왼쪽에서 부터 세 점에 대해서 CCW 방식을 적용하며 반시계 방향으로 외각 점을 찾는 과정을 보여준다.

2.2 시간 가중치 기반 중간지점 선정

본 논문은 중간지점 선정 과정에서 각 노드로부터 균일한 통행 비용을 갖는 최적의 중간지점을 선정하기 위해 시간 가중치 기반의 알고리즘을 제안한다. 중간지점 탐색은 각 노드의 위치에서 중간지점까지 통행 비용을 사용한 시간 가중치와 두 노드에 대한 단위 벡터 값을 사용하여 수행된다. 중간지점 선정 과정에 사용되는 통행 비용은 중간지점까지 이동하는데 소요되는 최단 시간이다. 이는 추후에 각 노드에 대한 다양한 가중치로의 확장이 가능하다.

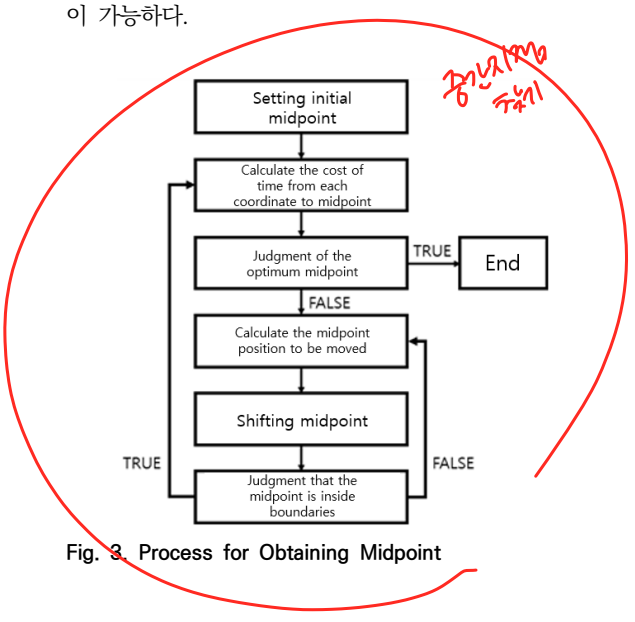

Fig. 3. Process for Obtaining Midpoint

중간지점 선정을 위한 탐색 과정은 다음 Fig. 3과 같다. 탐색과정은 노드들의 좌표 평균인 곳에서 시작한다. 먼저, 현재 위치한 중간지점이 최적 통행 비용을 가지는지 확인하여, 최적 통행비용을 가지지 않는 경우 중간지점이 다음 위치할 곳을 구한다. 중간지점을 시점으로 보고 한 노드를 종점으로 보았을 때 생기는 벡터를 단위 벡터로 만들고, 시간 가중치와 곱한 값을 모두 더한다. 그 후, 노드의 개수와 상수로 나누어 다음 중간지점이 위치할 곳에 대한 정보를 가진 벡터를 생성하고, 현재 위치한 중간지점에 더하여 중간지점을 이동한다. 상수는 벡터의 크기를 조절하는 역할로 중간지점 이동 범위를 조절한다. 이는 〈수식 2〉와 같다.

$$\overrightarrow{center_{i+1}} = \overrightarrow{center_i} + \frac{\sum_{j=1}^{n}(time\,W_j \times \overrightarrow{user_j})}{n \times \alpha} \quad (2)$$

여기서 center는 중간지점, timeW는 시간 가중치를 시간 가중치들의 평균으로 나눈 값, user는 노드와 중간지점의 단위벡터, n은 노드들의 총 개수, α는 벡터의 크기를 결정할 상수이다.

그 후, 이동된 중간지점이 경계를 벗어났는지 판단하고 최적의 중간지점인지 확인한다. 경계를 벗어났는지에 대한 판단 방법으로 각의 합, 직선 긋기 등을 이용한 방법들[11]이 있지만 본 논문에서는 직선 긋기를 사용했다. 직선 긋기의 원리는 Fig. 4와 같다[12-15]. 중간지점에서 오른쪽으로 직선을 그어 경계를 이루는 선분들과 만나는 개수를 보고 판단한다. 경계를 벗어났을 경우, 경계 범위안의 임의의 좌표로 이동시켜 탐색을 다시 수행한다. 최적의 중간지점이란 모든 노드에서 중간지점까지 통행 비용을 비교했을 때, 최대, 최소 통행 비용의 차이가 설정한 오차 내에 속한 경우를 말하며 이때의 각 경로들을 최적 경로라고 하며 통행 비용을 최적 통행 비용이라고 한다. 중간지점 선정 방식의 전체 동작 과정은 Fig. 5와 같다.

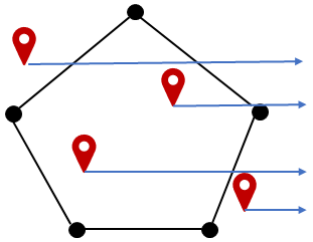

Fig. 4. Judgment of Points Inside Boundaries

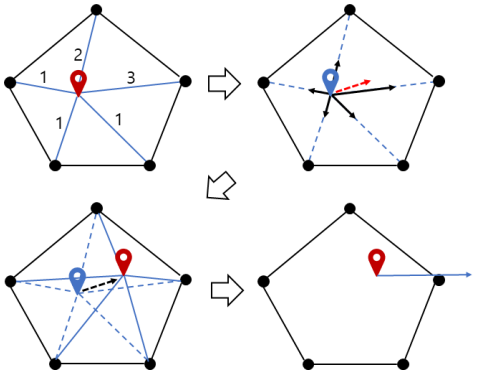

Fig. 5. Time-weighted Midpoint Search

Fig. 5는 Fig. 3을 도식화 하였다. 위쪽 2개의 과정은 현재 위치가 최적의 중간지점인지 판단 후, 각 노드에서 중간지점까지의 통행 비용과 단위 벡터를 계산하여 이동시킬 위치에 대한 벡터를 구한다. 아래 2개의 과정은 중간지점에 벡터를 더해 이동한 다음 해당 지점이 경계 내부에 존재하는지 확인한다. Fig. 6은 Fig. 3에 대한 알고리즘이다.

```
// Check the time from point to midPoint for judging a new midPoint
for (Position point : users) {
        int time = getPathTime(point, midPoint);
        movingTimes.push_back(time);

        Position unitVector = getUnitVector(point, midPoint);
        latVector += unitVector.getLatitude() * time;
        lonVector += unitVector.getLongitude() * time;
        avgTime += time;
}

avgTime /= users.size();
latVector /= (avgTime * users.size());
lonVector /= (avgTime * users.size());

// Judgment : a new midPoint is optimized
if (isOptimizedResult(movingTimes, consideredUserCnt))
        return midPoint;

// If a new midPoint is not optimized, change the position for a new position
midPoint.setPosition(midPoint.getLatitude()  + latVector / users.size(),
        midPoint.getLongitude() + lonVector / users.size());

// Check that a new position is in the polygon
if (PIPAlgorithm.isInside(midPoint, boudaryPoints) == NOT_INSIDE) {
        midPoint = getCenterOfGravity();
        consideredUserCnt--;
}
```

Fig. 6. Algorithm for Obtaining Midpoint

2.3 적응형 시간 가중치 설정

본 논문은 시간 가중치 기반 중간지점 선정 알고리즘에서 보다 효율적인 중간지점 탐색을 수행하기 위하여 적응형 시간 가중치를 설정한다. 이는 추후에 다양한 가중치에 대한 확장을 고려함으로써 최적 경로 탐색에 대한 효율성을 증대한다.

중간지점 탐색 과정에서 중간지점의 이동에 따라 통행 비용과 벡터의 방향이 변경되므로 시간 가중치가 통행 비용으로만 고정된다면, 중간지점이 매우 민감하게 반응하며 움직이므로 최적의 중간지점을 찾기가 어렵다.

따라서 본 논문에서는 변화하는 정보들에 대해 적응적으로 시간 가중치를 설정하는 방법을 제안한다. 시간 가중치 초기 값은 한 노드와 중간지점간의 통행 비용을 전체 노드와 중간지점간의 통행 비용의 평균으로 나눈 값으로 설정한다. 적응형 시간 가중치는 다음 〈수식 3〉과 같이 얻을 수 있다.

$$time\,W_j = \frac{time_j}{time_{avg}} \tag{3}$$

여기서 timej은 통행 비용, timeavg는 통행 비용들의 평균값이다.

적응형 가중치는 중간지점까지의 통행 비용을 평균 통행 비용으로 나눈 값으로 설정한다. 시간 가중치 값과 적응형 가중치 값의 예시는 Table 1과 같다.

Table 1. Example of the general time weight and adaptive weight according to the cost of time

The cost of time (Min)	Time weight	Adaptive weright
5	5	0.42
12	12	1
19	19	1.58

중간지점 이동으로 인해 변화하는 정보들에 적응하여 평균값과의 차이에 비례한 가중치가 설정되고, 벡터와의 연산을 통해 다음으로 이동할 중간지점을 선정하는데 효율적이다. 탐색 과정에서 적응형 시간 가중치를 사용하여 최적중간지점에 점차 도달해 가는 경우 시간 가중치가 1에 근접해 가면서 적합한 최적중간지점을 찾는다.

3. 실험 및 결과

3.1 실험 환경 및 실험 데이터

본 논문에서 시간 가중치 기반 중간지점 선정 실험에 사용된 환경은 Table 2와 같다.

중간지점 선정 실험에 사용된 각 노드와 결과 값은 임의의 좌표를 노드로 선정하여 네이버 지도와 다음 지도의 길찾기 서비스를 통해 직접 중간지점 선정 및 통행 비용을 계산 했고, 입력 노드 개수와 항목별 개수는 Table

3과 같고, 총 75개로 구성되어 있다. Table 4는 실험에서 사용된 항목의 예시이다.

Table 2. Experiment environment

Classification	Experiment Environment
Operating System	Windows 10 Education
CPU	Intel Core i5-8600K
RAM	16GB
Storage	SSD 256GB
GPU	NVIDIA GTX 1070Ti

Table 3. The number of nodes

The number of input nodes	The number of results
3	25
4	25
5	25
Total	75

Table 4. Example of experimental data with 4 users

User \ Coordinate	Latitude	Longitude
User 1	37.487636	126.993374
User 2	37.517097	127.041324
User 3	37.648654	127.034658
User 4	37.565674	126.977102
MidPoint	37.553410	127.004546

3.2 실험 결과

실험 방법은 실험 데이터의 입력 노드에 대해서 알고리즘을 실행하여 나온 중간지점 좌표와 실험 데이터의 결과 값을 비교하였을 때 평균 통행 비용과 위치를 비교하였고, 통행 비용은 ODsay 길 찾기 API를 통해 얻은 경로 중 최단 시간 경로의 시간을 사용하였다.

중간지점의 적정성의 판단 기준은 최적 통행 비용을 가지는 경우와, 그렇지 않을 때 실험데이터의 중간지점의 위치에서 반경 500m안에 알고리즘의 중간지점이 존재하는 경우로 나뉜다.

적응형 시간 가중치 기반 중간지점 선정방식을 적용함에 따라 최적 통행 비율, 선정된 장소의 적정성과 수행시간을 비교하기 위해 위치 기반 중간지점 선정 방식을 사용하여 실험을 진행하였다. 이를 위해 최적 통행 비용을 얻기 위한 비율은 〈수식 4〉를 사용하였고, 적정성 비율은 〈수식 5〉를 사용하였고, 수행 시간은 〈수식 6〉을 사용하였다.

$$optP_{ratio} = \frac{optPN}{totalN} * 100 \quad (4)$$

여기서 optPratio는 최적 통행 비율, optPN는 최적 통행 비용을 만족하는 데이터의 개수, totalN는 전체 실험 데이터의 개수이다.

$$pathA_{ratio} = \frac{pathAN}{totalN} * 100 \quad (5)$$

여기서 pathAratio는 적정성 비율, pathAN는 적정성 판단 기준을 만족하는 데이터의 개수, totalN는 전체 실험 데이터의 개수이다.

$$timeP = \frac{(timeC_{end} - timeC_{begin})}{timeF} \quad (6)$$

여기서 timeP는 수행 시간, timeCend는 함수가 끝난 시점의 CPU 클럭수, timeCbegin는 함수가 호출된 시점의 CPU 클럭수, timeF는 고해상도 타이머의 주파수이다. 컴퓨터의 메인보드에는 하나의 고해상도 타이머가 존재하는데, 이 타이머는 정확하고 일정한 주파수를 갖기 때문에 측정하고자 하는 코드들의 시작과 끝에서 CPU 클럭 수를 얻어 그 차이로 수행시간을 얻을 수 있어 정밀한 시간 측정이 가능하다.

Table 5. Percentage of optimal cost of time for location-based and proposed method

Item \ Method	Location-based	Proposed
Percentage of optimal cost of time(%)	32	68

실험 결과에서 최적 통행 비율의 비교는 Table 5, 선정된 장소의 적정성 비교는 Table 6, 수행시간 비교는 Table 7과 같다.

위치 기반 방법은 빠른 수행시간을 가지지만 32%로 비율로 대부분 최적 통행 비용을 갖지 않는 것을 보여주며, 적정성은 34.6%로 매우 낮은 수치를 보여준다. 실제 중간지점 선정을 이용하여 소비자에게 최적 경로 탐색 서비스를 제공하기 위한 시스템에 적용하기 힘든 수치를 보여준다.

Table 6. Results for adequacy of location-based and proposed method

The number of input nodes \ method	Location-based	Proposed
	Ratio of adequacy(%)	Ratio of adequacy(%)
3	40	72
4	28	84
5	36	80
Average	34.6	78.6

Table 7. Results of average performance time measurements for location-based and proposed method

The number of input nodes \ method	Location-based	Proposed
	Average performance time(s)	Average performance time(s)
3	3.8346	20.5393
4	4.2329	18.0825
5	4.8014	33.8241
Average	4.2896	24.1486

반면, 시간 가중치 기반 중간지점 선정 방식에서의 실험 결과를 살펴보면 최적 통행 비용을 갖는 경우는 68%이고, 적정성은 78.6%였다. 위치 기반의 중간지점 선정과는 달리 높은 비율로 최적 통행 비용의 중간지점을 산출했으며, 모든 경우에서 높은 적정성을 가지는 것을 볼 수 있다.

앞선 두 결과에 따라 위치 기반 중간지점 선정 방식과 시간 가중치 기반 중간지점 선정방식의 실험 결과들을 비교해 보았을 때 시간 가중치 기반 중간지점 선정방식의 수행시간은 API 호출로 인해 다소 오래 걸리지만 최적 통행 비율과 적정성에서 매우 큰 차이를 보인다.

Fig. 7에서 보는 것처럼 위치 기반과 시간 가중치 기반 중간지점 선정 방식에서 노드의 개수와 무관하게 차이가 벌어진 모습을 보여준다. 중간지점 선정에서 노드가 3개인 경우 위치 기반에서는 40%이고, 시간 가중치 기반에서는 72%로 약 32%로 높은 적정성의 모습을 보여준다. 또한, 노드가 4개인 경우 각각 28%, 84%로 약 56%의 차이가 나며, 5개인 경우 각각 36%, 80%로 약 44%의 차이로 노드가 4개인경우 50%가 넘는 매우 큰 차이를 볼 수 있다. 종합적으로 위치 기반의 방식의 적정성은 34.6%, 시간 가중치 기반의 방식의 적정성은 78.6%로 시간 가중치 기반의 방식이 높은 적정성을 가짐을 알 수 있다. 이는 시간 가중치 기반 중간지점 선정 방식을 이용하는 것이 더 높은 신뢰도로 최적의 중간지점을 선정하여 최적 경로를 제공할 수 있다는 것을 의미한다. 따라서 사용자들을 각 노드로 하는 네트워크에서 가중치 기반의 최적 경로 탐색[16,17], 최적 경로 탐색에 다양한 가중치를 적용하는 방식[18], LBSNS 기반 장소 추천 시스템[19] 등에 적용하였을 때 사용자에게 더욱 높은 신뢰도로 최적 경로를 제공할 수 있기 때문에 효율적인 서비스 제공이 가능해진다.

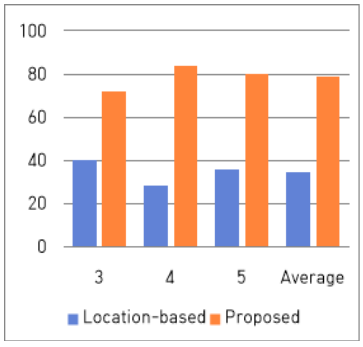

Fig. 7. Adequacy of location-based and proposed method

4. 결론

기존의 위치 기반 중간지점 탐색 방식은 소요시간을 고려하지 않아 최적 경로를 제공하기에 어려운 문제점이 있다. 이를 해결하기 위하여 본 논문에서는 최적 경로 탐색을 위한 방법으로 적응형 시간 가중치 기반 중간지점 선정 방식을 사용하였다. 효율적인 탐색을 수행하기 위하여 사용자와 중간지점의 위치 따라 변화하는 소요시간에 대해 적응하는 가중치를 설정하고, 탐색할 방향을 벡터 연산을 통해 최적 통행 비용을 가지는 효율적인 중간지점 탐색 방법을 제안하였다. 실험 결과 위치 기반 중간지점 선정 방식보다 44%가 더 높은 78.6%의 적정성을 가지는 최적 중간지점을 선정하는 것을 확인 할 수 있었다.

본 논문에서 제안한 중간지점까지의 최적 경로 탐색 방법은 사용자들을 각 노드로 하는 네트워크에서 가중치 기반의 최적 경로를 찾는데 활용 가능할 것으로 여겨지며, 추후 최적 경로 탐색에 다양한 가중치를 적용하는 방식에 관한 연구를 통해 더 높은 적정성과 최적 통행 비용의 비율을 얻을 수 있을 것으로 기대된다.

REFERENCES

[1] H. J. Bae, J. N. Song, Y. J. Lee & J. W. Lee. (2015). The Implementation of a User Location and Preference-based Appointed Place Recommendation Mobile Application. *KIISE Transaction on Computing Practices, 21(6)*, 403-411.

[2] C. Y. Wang, P. S. Jang & H. H. Kim. (2019). A Study on the Characteristics of the Seasonal Travel Path of Individual Chinese Travellers in Korea. *Journal of the Korea Convergence Society, 10(7),* 23-31.

[3] S. H. Bae. (2006). Development of a Method for Partial Searching Technique for Optimal Path Finding in the Long Journey Condition. *Journal of The Korea Society of Civil Engineers D, 26(3D),* 361-366.

[4] G. S. Nam, B. H. Jeong, J. H. No, J. W. Choi & Y. S. Kim. (1995, Mar). Development of An Optimal Path Finding System(X-Path) : An Interim Report. *Symposium of The Korea Society of Management information Systems,* 486-505.

[5] S. T. Park & Y. K. Kim. (2019). A Study on Deriving an Optimal Route for Tourists through the Analysis of Big Data. *Journal of Convergence for Information Technology, 9(10),* 56-63.

[6] S. H. Hong. (2018). Research on Security Model and Requirements for Fog Computing : Survey. *Journal of the Korea Convergence Society, 9(5),* 27-32.
DOI : 10.15207/JKCS.2018.9.5.027

[7] E. Y. Jeong. (2009). Master dissertation. *Analytic proofs of theorems in the plane geometry using homogeneous barycentric coordinates on triangles.* Konkuk University, Seoul.

[8] P. Mitura, I. Simecek & I. Kotenkov. (2017). Effective Construction of Convex Hull Algorithms. *2017 19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2017 19th International Symposium on,* 105-112.
DOI : 10.1109/SYNASC.2017.00028

[9] G. Ronald. (1972). An efficient algorithm for determining the convex hull of a finite planar set. Info. Pro. Lett. 1. 132-133.

[10] V. Tereshchenko, Y. Tereshchenko & D. Kotsur. (2015). Point triangulation using Graham's scan. *Fifth International Conference on Innovative Computing Technology (INTECH 2015).*
DOI : 10.1109/INTECH.2015.7173370

[11] F. M. Chowdhury et al. (2005). The Unidirectional Edge Method: A New Approach for Solving Point Enclosure problem for Arbitary Polygon. *Asian Journal of Information Technology, 4(5),* 537-540.

[12] D. Kularathne & L. Jayarathne. (2018, Oct). Point in Polygon Determination Algorithm for 2-D Vector Graphics Applications. *2018 National Information Technology Conference(NITC).* Colombo : IEEE

[13] O. Rakhmanov. A new approach (extra vertex) and generalization of Shoelace Algorithm usage in convex polyon (Point-in-Polygon). (2019). *2018 14th International Conference on Eclectronics Computer and Computation(ICECCO).*
DOI : 10.1109/ICECCO.2018.8634725

[14] C. W. Huang et al. (1997). On the complexity of point-in-polygon algorithms. *Computer& Geosciences, 23(1),* 109-118.

[15] P. Bourke. (1987, Nov). *Determining if a point lies interior of Polygon.* Polygons and meshes [Online]. http://paulbourke.net/geometry/polygonmesh

[16] H. J. Kang & H. C. Kim. (2013). Dynamic Resource Assignment in the Multi-layer Networks. *Journal of The Korea Society of Information Assurance, 13(6),* 77-82

[17] Y. T. Kim & Y. S. Jeong. (2015). Optimization Routing Protocol based on the Location, and Distance information of Sensor Nodes. *Journal of Digital Convergence, 13(2),* 127-133
DOI :10.14400/JDC.2015.13.2.127

[18] K. A. Yu. (2010). Implementation of Tactical Path-finding Integrated with Weight Learning. *Journal of The Korea Society for Simulation, 19(2),* 91-98

[19] K. I. Jung, B. I. Ahn, J. J. Kim & K. J. Han. (2014). Location Recommendation System based on LBSNS. *Journal of Digital Convergence, 12(6),* 277-287
DOI : 10.14400/JDC.2014.12.6.277

허 유 성(Yu-Sung Her) [학생회원]

· 2020년 2월 : 세종대학교 컴퓨터공학과(공학사 졸업예정)
· 관심분야 : 통계, AI
· E-Mail : hus5522@naver.com

김 태 우(Tae-Woo Kim) [학생회원]

· 2019년 8월 : 세종대학교 컴퓨터공학과(공학사)
·관심분야 : 통계, 미분, ML, Data Analytics
· E-Mail : viodle238@naver.com

안 용 학(Yong-Hak Ahn) [정회원]

· 1997년 8월 : 경희대학교 컴퓨터공학과(공학석사)
· 2005년 2월 : 경희대학교 컴퓨터공학과(공학박사)
· 1999년 12월 : 한국통신정보기술 GIS 공학연구소 전임연구원
· 현재 : 세종대학교 컴퓨터공학과 교수

· 관심분야 : Computer Vision, Web Service, AI, DIP
· E-Mail : yohans@sejong.ac.kr