Top-n 스카이라인 질의를 이용한 다차원 외판원 순회문제†

진창균*, 양세빈*, 강은진*, 김지윤**, 김종완+, 오덕신++
삼육대학교 {*컴퓨터·메카트로닉스공학부, **식품영양학과, +스미스교양대학, ++경영정보학과}
jcg6074@naver.com, 1213hope@naver.com, ejk0729@naver.com,
wldbs3592@naver.com, kimj@syu.ac.kr, ohds@syu.ac.kr

Multi-dimensional Traveling salesman problem using Top-n Skyline query

ChangGyun Jin*, Sevin Yang*, Eunjin Kang*, JiYun Kim**, Jongwan Kim+, Dukshin Oh++
*Division of Computer · Mechatronics technology, Sahmyook University
**Department of Food&Nutrition, Sahmyook University
+Smith Liberal Arts College, Sahmyook University
++Department of Management Information Systems, Sahmyook University

요 약

PDA나 휴대폰 단말로 여러 속성의 데이터를 이용하여 사용자에게 필요한 정보를 제공하는 위치기반 서비스는 물류/운송 정보 서비스, 버스/지하철 노선 안내 서비스 등에 사용된다. 여기에서 제공하는 데이터들을 최적 경로를 구하는 외판원 순회문제 (Traveling Salesman Problem)에 사용한다면 더 정확한 경로 서비스 제공이 가능하다. 하지만 데이터의 수가 많아질수록 비교 횟수가 기하급수적으로 늘어나는 외판원 순회 알고리즘의 특성상 일반 단말기에서 활용하기에는 배터리의 제약이 따른다. 본 논문에서는 이와 같은 단점을 해결하기 위해서 최적 경로의 후보군을 줄일 수 있는 스카이라인 질의를 이용하여 n차원 속성에 대한 최적 경로 알고리즘을 제안한다. 실험에서 정확도와 오차율을 통해 제안한 방식의 유용성을 보였으며 기존방식과 연산시간 차이를 비교하여 다차원방식의 효율성을 나타내었다.

1. 서론

최근 사용자에게 위치에 따른 데이터들을 제공하는 위치기반 서비스가 대두되고 있다. 지하철/버스 노선 안내 서비스의 경우 노선도, 역과 역 정류장과 정류장 사이의 이동 거리 또는 이동시간, 타기 위해 기다려야 하는 시간 등 많은 속성의 데이터들을 얻을 수 있다. 그리고 이 데이터들을 최적의 경로를 구하는 문제인 외판원 순회문제 (Traveling Salesman Problem, TSP)에 적용한다면 더 세밀한 서비스를 제공할 수 있다.

입구와 출구가 같은 놀이동산에서 사용자는 각 놀이기구 사이의 거리와 대기시간 및 이용시간 데이터를 얻을 수 있다. 여기에서 이용시간은 언제나 변함없는 불변형 데이터이지만 거리와 대기시간은 가변형 데이터로 현 위치와 시각에 따라 변하게 된다. 따라서 거리만을 이용하여 계산된 TSP는 가장 짧은 경로지만 바뀌는 시간 때문에 가장 빠른 경로라고는 할 수 없다. 따라서 시간까지 포함한 TSP가 더 정확한 경로를 제공하게 된다.

TSP는 데이터 수가 증가할수록 비교 횟수도 기하급수적으로 늘어나기 때문에 일반 단말기에서 처리하기에는 오랜 처리시간이 소요된다. 또한, 다중속성 데이터 (도시)에서 TSP는 다중속성을 비교해야 하므로 비교 횟수는 더욱 증가한다.

본 논문에서는 근접성, 관광 선호도 등 다양한 속성을 포함하는 도시에서 최단 경로를 탐색하기 위한 TSP 알고리즘을 적용한다. 즉, 다중속성을 포함한 도시의 경로 계산에서 빠른 처리속도를 보장하도록 Top-n 스카이라인을 이용한 다차원 TSP 해결방식인 TS-MDT (Top-n Skyline-Multi Dimension TSP) 알고리즘을 제안한다.

TS-MDT는 연산시간을 줄이기 Top-n 스카이라인 알고리즘을 사용하여 다중속성데이터 (도시)에서 TSP 경로의 후보 도시를 줄이는 방식이다. 실험에서는 TSP 해결의 대표적인 방법인 완전 탐색에서 반환된 경로와 비교하며 이때, 완전 탐색은 다차원데이터를 사용할 수 없으므로 각 속성의 데이터들을 더하여 1차원으로 변환 후 적용한다 (모든 데이터가 낮은 값이 더 좋다는 가정).

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 관해 설명하고 3장에서는 Top-K 스카이라인 질의방식을 이용하여 다차원데이터에서 TSP를 해결하는 방식을 제안한다. 4장에서는 실험을 통해 다차원에서의 Top-K 스카이라인 질의를 이용한 TSP의 성능을 분석하고 효율성과

++교신 저자
† 이 논문은 2019년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (NRF-2018R1D1A1B07045642, NRF-2017R1D1A1B03035884).

유용성을 증명한다. 5장에서는 결론을 기술한다.

2. 관련 연구

2.1 외판원 순회문제

최적의 경로를 구하는 문제에서 사용되는 외판원 순회문제 (TSP)는 모든 도시가 이어져 있다는 가정하에 그 도시들을 한 번씩만 들려서 다시 출발점으로 되돌아오는 최단 경로를 반환한다. TSP는 차량 경로 문제, 일정 계획, 스케줄링이나 택배, 음식 등의 배달 문제에서 실용적으로 적용될 수 있다.

TSP는 사용되는 데이터가 많아질수록 기하급수적으로 늘어나는 비교 횟수 때문에 최적해 구하는 방식보다는 근사한 해를 구하는 해법이 연구됐다 [1]. 근사해 해결방법의 대표적인 예로는 유전 알고리즘 [3]이 있다.

기존의 연구들은 모두 1차원 속성에서의 TSP 해결에 중심을 두고 있지만 본 논문에서는 다차원속성에서의 TSP 해결에 그 무게를 두고 있다.

2.2 Top-n 스카이라인 질의

스카이라인이란 전체 데이터에서 서로 지배되지 않는 객체들의 집합이다. 작은 값을 탐색하는 과정에서 어떤 객체 p가 다른 객체 q보다 모든 차원 또는 적어도 한 차원 이상에서 작은 값을 가질 때 p는 q를 지배한다고 표현한다.

Top-n 스카이라인 질의는 다차원속성을 갖는 데이터에서 사용자의 선호도에 따라 스카이라인을 구하고 의미 있는 n 개의 데이터를 반환하는 질의이다 [4]. 이때, 사용자의 선호도는 속성을 비교 및 판단하는 기준으로 사용되며 최종 결과는 n개의 데이터가 된다. 대표적인 방법으로는 K개의 차원에서 다른 차원들에 지배되지 않는 객체들을 반환하는 K-dominant 스카이라인이 있다. 처리기법으로 One-scan, Two-scan, Sorted Retrieval 알고리즘 [2] 등이 있고 그중 Two-scan 기법이 가장 우수한 처리속도를 자랑한다.

<표 1>은 5개의 속성 (s1~s5)을 갖는 5개의 객체 (p1~p5)를 나타내고 있으며 Top-2 스카이라인을 구한다고 가정하면 K를 1부터 늘려가며 전체 속성에 대해 계산하게 된다.

K가 1일 때, Top-2를 구하면 즉, 속성이 1차 원이라면 모든 데이터가 서로에게 지배되지 않는다. 예를 들어 p1과 p2를 비교하면 p1은 s5에서 p2에 의해 지배되지만, s1에서는 p2를 지배하여 결론적으로는 서로 지배되지 않는다. 이를 적용하여보면 {p1, p2, p3, p4, p5}가 모두 스카이라인 후보가 되므로 Top-2의 결과로 2개 객체만 한정할 수 없다. 5개의 객체를 구한 후 또 한 번 사용자의 선호도에 따라 스카이라인 질의를 수행하는 선형 알고리즘 [5]을 사용한다면 상위 2개의 후보군을 구할 수 있다. 그러나 이는 연산시간을 줄인다는 본 논문의 취지와는 맞지 않는다.

<표 1> 객체와 속성

Point	s1	s2	s3	s4	s5
p1	4	2	3	5	1
p2	2	2	2	5	8
p3	3	5	6	7	4
p4	5	4	2	6	8
p5	2	2	6	6	1

속성인 K가 2일 경우에는 p3이 속성 {s2, s3, s4, s5}에서 p1을 지배하지만, p3는 p1의 s1 차원에서만 지배되므로 많은 속성이 지배되는 p1은 p3에 의해 지배된다고 표현할 수 있다. 같은 방식으로 p2, p5 데이터 역시 p3에 의해 지배되기 때문에 스카이라인 후보군에서 제외되므로 남아있는 데이터는 {p3, p4}가 된다. 두 개의 데이터가 후보군으로 남아있으므로 Top-2 스카이라인 질의는 {p3, p4}를 반환하게 된다.

본 논문에서 Top-n 스카이라인 질의는 TSP를 위해 각 도시를 방문할 때 경로에 포함되는 도시의 수를 줄이기 위해 사용된다.

3. Top-n 스카이라인 질의를 이용한 다차원데이터의 TSP 해결방식

제안하는 기법은 Top-n 스카이라인 질의를 이용하여 위치마다 존재하는 경로들의 후보군을 줄이고 완전 탐색을 이용하여 남아있는 경로 중 가장 비용이 적은 경로를 구하는 방법이다. <표 2>는 본 논문에서 사용된 기호와 그 정의들이다.

<표 2> 기호 및 정의

기호	의미
N	모든 데이터 수
dimension	모든 속성 수
T	스카이라인 후보들의 집합
t	T에 속한 후보들의 수
limit	Top-n에서 n의 값
D	모든 데이터의 집합
s	현재 위치한 도시의 번호
R[s][]	각 도시의 경로를 표시한 배열
check[]	도시의 방문 여부 확인 배열
cnt	현재까지 방문한 도시들의 개수
kdo(k)	K-dominant 알고리즘 호출함수
Top(cnt)	Top-n 알고리즘 호출함수
K	K-dominant 질의를 위한 K값

제안기법에서 사용되는 알고리즘은 k-dominant, Top-n, 깊이 우선 탐색을 이용한 완전 탐색으로 총 3가지가 있다.

(그림 1)의 K-dominant 알고리즘은 스카이라인 후보들의 집합인 T를 데이터 모두가 적합하다고 초기화시킨다 (1행). 그 후 모든 데이터를 서로 비교하여 지배되는 데이터는 T에서 제거한다 (2~11행). 이 루프에서 check[] 배열

을 통해 이미 들렸던 곳을 확인하고 방문한 지점이라면 제외하고 계산한다 (3~4, 6~7행). 마지막으로 Top-n의 질의를 위하여 지배되지 않는 데이터들의 집합인 T와 T에 속해있는 데이터 수인 t를 반환함으로써 마무리된다.

K-dominant Algorithm

```
Kdo(k) function
1: initialize set of K-dominant data T = ALL
2: for every p ∈ D
3:    if check[p] == true
4:      start next-loop
5:    for every p' ∈ D, p' ≠ p
6:      if check[p'] == true
7:        start next-loop
8:      if(p' k - dominant p)
9:        delete p from T
10:    if(p k - dominant p')
11:      delete p' from T
12: return T
```

(그림 1) k-dominant 알고리즘

T는 스카이라인 후보군이다. 마지막에 반환되는 T는 현재 위치에서 갈 수 있는 경로를 나타내는 것이다. 본래는 모든 지점에 대한 경로가 존재하지만, 스카이라인으로 그 경로를 줄이는 것이 제안기법의 핵심이다.

TOP-n Algorithm

```
TOP(cnt) function
1:    intialize the number of limit = (N-cnt+1) / 2
2:    intialize k = 1
3:    while(k <= dimension)
4:       kdo(k)
5:       if k == 1 and t <= limit
6:         R[s] = T
7:         return R
8:       if k == dimention and t > limit
9:         R[s] = T
10:        return R
11:     if k ≠ 1 and k ≠ dimention and t == limit
12:       R[s] = T
13:       return R
14:     if k ≠ 1 and k ≠ dimention and t < limit
15:       kdo(k - 1)
15:       R[s] = T
16:       return R
17:   k = k + 1
```

(그림 2) Top-n 알고리즘

(그림 2)의 Top-n 알고리즘은 기존의 방식과 다르게 스카이라인의 개수가 n과 반드시 같다고 조건을 정해 놓을 수는 없다. 그 이유는 k가 1부터 속성의 개수만큼 루프를 도는데 모든 k에서 n만큼의 t가 나오지 않을 수도 있기 때문이다. 그렇게 되면 오류가 발생하기 때문에 본 논문에서는 다른 방식으로 진행하였다. 먼저 limit은 Top-n의 n으로, 구해야 할 후보군 개수의 기준에 해당한다 (1행). 변수 이름을 n이 아닌 limit으로 정한 이유는 데이터 개수인 N과의 혼동을 방지하기 위함이다. (그림 2)에서는 전체 데이터 개수인 N에서 현재 방문한 점의 개수 cnt만큼을 뺀 후 1을 더하고 그 수를 다시 2로 나눈다. 이렇게 되면 limit은 현재 남은 점들의 절반이 된다.

k를 1로 초기화한 후 속성개수만큼 루프를 돌린다 (3~16행). 4행에서 k-dominant 알고리즘을 호출하여 T와 t를 반환받고 k가 1일 때, t가 limit보다 작으면 k가 늘어나더라도 t는 limit보다 작아서 함수를 끝낸다 [2]. 8행은 마지막 루프일 때 t가 limit보다 크면 모든 k에서도 t는 limit보다 크기 때문에 함수를 끝낸다.

11행과 14행은 위의 두 경우를 제외한 것으로서 t가 limit과 같으면 바로 함수를 끝내고 t기 limit보다 작으면 k를 하나 줄인 후 k-dominant 알고리즘을 한 번 더 호출한 후 함수를 끝낸다. k를 하나 줄이는 이유는 k가 늘어나면 t가 줄어들기 때문에 정확도 역시 감소하기 때문이다.

모든 조건문에서 보이는 R[s]은 모든 점의 경로를 표시하는 배열로 k-dominant 알고리즘에서 반환된 T를 복사한다. 즉, k-dominant 알고리즘은 경로의 후보군을 줄이는 역할이고 Top-n 알고리즘은 줄여진 후보군의 수를 적절하게 만드는 역할이다.

본 논문에서 사용된 TSP는 (그림 3)과 같이 완전 탐색이며 재귀함수로 구성되어 있다.

Exploration Algorithm

```
EXP() function
1: if cnt == n
2:   return
3: Top(cnt)
4: for every I ∈ N
5:   if R[s][i] == 0
6:     start next loop
7:   if check[i] == false and R[s][i] ≠ 0
8:      check[i] = true
9:      EXP(cnt + 1, i)
10:    check[i] = false
```

(그림 3) 완전 탐색 알고리즘

알고리즘은 모든 지점을 방문하면 재귀가 종료된다 (1~2행). 처음으로 Top-n 함수를 호출하여 경로인 현재 위치에서 이어진 경로의 집합인 R을 반환받는다 (3행). 모든 데이터의 개수만큼 루프를 시키고 현재 지점과 이어지지 않은 점은 루프를 진행하지 않고 다음으로 변경한다 (5~6행). 어느 한 점이 방문 되지 않았고 경로가 이어져 있다면 현재 지점을 방문 표시를 True로 변경하고 (8행) 이어진 지점으로 가서 다시 재귀를 시작한다 (9행). 재귀가 끝나고 돌아오면 현재 지점에서 이어진 점이 하나가 아니므로 다음 경로를 위해서 방문 표시를 false로 변경한다 (10행).

4. 실험

제안한 기법은 완전 탐색기법과 비교하였으며 성능 평가를 위한 데이터는 무작위로 생성하여 진행하였다. 제안기법의 실험환경은 <표 3>과 같다.

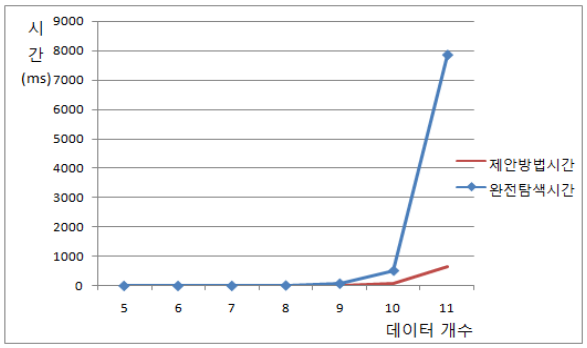
데이터의 개수는 보유 중인 기기의 사양 문제로 인하여 11개의 데이터를 대상으로 하였으며 각 데이터는 3개의 속성을 갖는다. 실험횟수는 총 1만 회를 수행하였으며 이때, Top-n에서 n의 수는 사용자가 도시를 방문할 때마다 변하는 가변형 변수이다. 왜냐하면 본 논문에서 Top-n

은 대상 도시의 수를 줄이기 위해 사용되기 때문이며 각 단계에서 남은 도시 수의 절반으로 지정한다. n의 값이 크면 연산시간은 느리고 정확도는 증가한다. 반대로 n의 값이 낮을수록 연산시간은 빨라지고 정확도는 감소한다.

<표 3> 실험환경

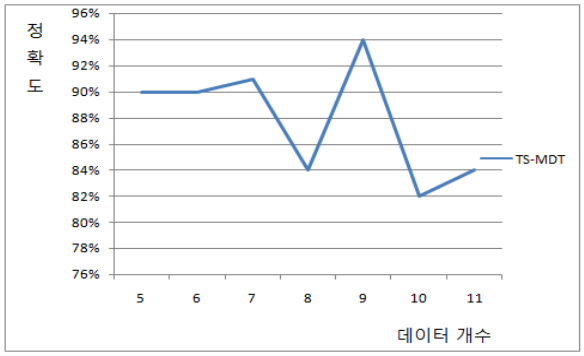
구분	내용
시스템	Intel i5 CPU 2.3 GHz, RAM 4GB
Top-n의 n 개수	(N-cnt+1)/2개
데이터 수	5~11개
속성 수	3개
실험횟수	10000회

(그림 4)는 데이터 수 변화에 따른 질의 처리시간을 비교한 것이다. 데이터 수가 증가할수록 질의처리시간은 월등히 줄어들었다. 제안하는 기법은 완전 탐색을 수행하면서 Top-n 스카이라인 질의에서 후보 도시를 줄이는 연산시간을 포함하지만 기존의 TSP 완전 탐색에 비해 더 빠른 처리시간을 나타냈다.



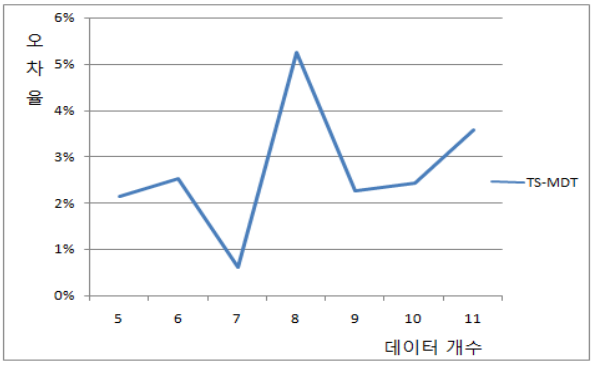
(그림 4) 데이터 수에 따른 연산시간 차이

(그림 5)는 데이터 수 증가에 따른 정확도 비교로 모든 속성의 데이터를 합친 후 계산한 완전 탐색의 최적 해와 제안한 기법에서 나온 경로 비용을 비교하여 나온 결과이다. 데이터가 9개일 때 가장 높은 94%를 기록하였고 10개일 때 가장 낮은 82%를 기록하였다. 그 외의 데이터들도 80%에서 95% 사이를 오가는 형태를 보였다.



(그림 5) 데이터 수에 따른 정확도

(그림 6)은 데이터 수 증가에 따른 오차율 비교로 정확도 비교에서 일치되지 않는 데이터의 최적 해와 비용 차이를 타낸 것이다. 8개의 데이터에서 가장 높은 오차율인 5.24%를 기록하였고 7개의 데이터에서 가장 낮은 0.62%를 기록하였다. 그 외의 데이터들은 0%에서 4% 사이를 오가는 형태를 나타냈다.



(그림 6) 데이터 수에 따른 오차율

5. 결론

본 논문은 위치기반 서비스를 이용하여 얻은 데이터로 활용하는 외판원 순회문제에서 시간이 너무 오래 소요되는 단점을 해결하기 위해 후보군을 줄이는 Top-n 스카이라인 질의를 이용하는 기법을 제안하였다. 제안된 기법은 실험을 통해 11개 데이터에서 10배 정도 빠른 연산시간을 보였다. TSP 알고리즘의 특성상 많은 데이터처리를 위해서는 높은 사양의 컴퓨터가 필요하지만 본 연구에서는 컴퓨터의 사양 제약으로 더 많은 데이터를 실험해 보지 못하였다. 그러나 적은 데이터를 사용한 다중속성에 대한 처리에서 유의미한 결과를 얻을 수 있었으며 향후에는 다량의 데이터에 대한 실험을 수행하여 결과를 공유할 것이다.

참고문헌

[1] 권상호, 김성민, 강맹규, “외판원문제에서 국지해를 탈출하기 위한 비용완화법,” 대한산업공학회지, 30(2), pp. 120-129, 2004.

[2] C. Y. Chan, H. V. Jagadish, K. L. Tan, A. K. H. Tung, and Z. Zhang, “Finding k-Dominant Skyline in High Dimensional Space,” Proc. of the 2006 ACM SIGMOD international conference on Management of data, pp. 503-514, 2006.

[3] 김광백, 송두헌, “유전자 알고리즘을 이용한 경로 탐색,” 한국해양정보통신학회, 제15권, 제6호, pp. 1251-1255. 2011.

[4] J. Chomicki, P. Godfrey, J. Gryz, D. Liang, “Skyline with Presorting,” Proceedings 19th International Conference on Data Engineering, pp. 717-719, 2003.

[5] 박희수, 이종욱, “다차원 공간에서 정확한 선형 스카이라인 알고리즘,” 정보과학회논문지 제45권 제10호, pp. 1089-1095, 2018.