KIPS Trans. Softw. and Data Eng.
Vol.9, No.1 pp.17~24 pISSN: 2287-5905

https://doi.org/10.3745/KTSDE.2020.9.1.17

Multi-Dimensional Traveling Salesman Problem Scheme Using Top-n Skyline Query

ChangGyun Jin† · Dukshin Oh†† · Jongwan Kim†††

ABSTRACT

The traveling salesman problem is an algorithmic problem tasked with finding the shortest route that a salesman visits, visiting each city and returning to the started city. Due to the exponential time complexity of TSP, it's hard to implement on cases like amusement park or delivery. Also, TSP is hard to meet user's demand that is associated with multi-dimensional attributes like travel time, interests, waiting time because it uses only one attribute - distance between nodes. This paper proposed Top-n Skyline-Multi Dimension TSP to resolve formerly adverted problems. The proposed algorithm finds the shortest route faster than the existing method by decreasing the number of operations, selecting multi-dimensional nodes according to the dominance of skyline. In the simulation, we compared computation time of dynamic programming algorithm to the proposed a TS-MDT algorithm, and it showed that TS-MDT was faster than dynamic programming algorithm.

Keywords : Traveling Salesman Problem, Top-n Skyline, Skyline Query, Multi-Dimension Data, Dynamic Programming, Shortest Path

Top-n 스카이라인 질의를 이용한 다차원 외판원 순회문제 기법

진 창 균† · 오 덕 신†† · 김 종 완†††

요 약

외판원 순회문제(Traveling Salesman Problem)는 세일즈맨이 한 도시(node)를 출발하여 모든 도시를 한 번씩 방문한 후 다시 출발점으로 되돌아오는 최적 경로를 반환한다. 이 기법은 도시의 수가 늘어날수록 연산횟수가 기하급수적으로 늘어나는 단점으로 인해 실생활에서 여러 노드(node)를 방문해야 하는 놀이동산이나 택배에 적용하기에는 탐색 성능에 한계가 있다. 또한, 최적 경로 탐색은 각 노드 사이의 거리를 1차원 속성으로 사용하기 때문에 이동시간, 관심도, 대기시간 등의 다차원속성을 고려하는 사용자의 요구를 만족하기 어렵다. 본 논문에서는 이와 같은 단점을 해결하기 위하여 Top-n 스카이라인 질의(Skyline query)를 이용한 다차원 외판원 순회문제(TS-MDT, Top-n Skyline-Multi Dimensional TSP) 알고리즘을 제안한다. 제안기법은 스카이라인의 지배원칙에 따라 다중 속성의 노드들을 제거함으로써 연산횟수의 감소를 통한 신속한 연산과 최적 경로를 반환한다. 실험에서는 1차원 속성의 데이터를 사용한 기존의 동적 계획법과 다차원속성을 처리하는 제안기법의 연산시간을 비교한 결과, 같은 데이터 개수일 때 다차원속성을 처리하는 제안기법이 더 빠른 것으로 나타났다.

키워드 : 외판원 순회문제, TOP-n 스카이라인, 스카이라인 질의, 다차원 데이터, 동적 계획법, 최단 경로

1. 서 론

최근 사용자에게 위치정보를 제공하는 위치기반 서비스가 대두되고 있다. 위치기반 서비스(Location based service)는 휴대폰과 PDA 같은 이동통신망과 IT 기술을 종합적으로 활용한 위치정보 기반의 시스템 및 서비스이다[1]. 지하철이나 버스 노선 안내 서비스의 경우에는 노선도, 역과 역, 정류장과 정류장 사이의 이동 거리 또는 이동시간, 탑승하기 위해 기다려야 하는 시간 등 많은 속성이 존재한다. 최적 경로를 구하는 외판원 순회문제(Traveling Salesman Problem, TSP)에 다중 속성을 적용한다면 사용자의 다양한 요구에 맞추어 더 세밀한 서비스를 제공할 수 있다.

최적 경로 탐색에서 다중 속성을 고려해야 하는 이유는 다음과 같다. 입구와 출구가 같은 놀이동산에서 사용자는 각 놀이기구 사이의 거리, 대기시간 및 이용시간을 알 수 있다. 이용시간은 언제나 변함없는 불변형 데이터이지만 거리와 대기시간은 가변형 데이터로 현 위치와 대기 인원 수에 따라 변하

※ 이 성과는 2019년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2018R1D1A1B07045642, NRF-2017R1D1A1B03035884).
※ 이 논문은 2019년도 한국정보처리학회 춘계학술발표대회에서 'Top-n 스카이라인 질의를 이용한 다차원 외판원 순회문제'의 제목으로 발표된 논문을 확장한 것임.
† 비 회 원 : 삼육대학교 컴퓨터-메카트로닉스 공학부 학사과정
†† 정 회 원 : 삼육대학교 경영정보학과 교수
††† 종신회원 : 삼육대학교 스미스학부대학 조교수
Manuscript Received : August 2, 2019
Accepted : September 17, 2019
* Corresponding Author : Jongwan Kim(kimj@syu.ac.kr)

게 된다. 따라서 두 놀이기구 사이의 거리만을 평가하는 경우에, TSP는 가장 짧은 경로를 반환하지만 계속해서 변경되는 대기시간 때문에 서비스를 이용하는 사용자의 관점에서 최적의 경로로 판단하기에는 적합하지 않다.

위의 예와 같이, TSP에서 사용되는 가변형 데이터는 하나 이상일 수 있으며 모든 속성에 대한 연산이 진행되어야 더 세밀한 최단 경로를 제공할 수 있다. 즉, 사용자의 요구를 만족하는 서비스 제공을 위해서는 다중 속성에 대한 평가가 함께 반영되어야 한다.

한 가지 속성을 사용할 경우, 거리가 가장 짧은 경로나 가장 시간이 적게 걸리는 경로 중 하나만 선택할 수 있다. 하지만 다중 속성을 사용하면 거리가 짧으면서 비교적 시간도 적게 걸리는 경로를 반환할 수 있다.

예를 들어 시간과 거리비용이 각각 40, 70인 경로 A, 50, 50인 경로 B 그리고 60, 50인 경로 C가 있다고 가정하자. 만약 사용자가 짧은 거리만 선호한다면 경로 B와 C, 적게 걸리는 시간만 선호한다면 경로 A를 선택한다. 그러나 두 속성을 모두 고려하면 경로 B가 최적 경로로 서비스된다.

TSP는 탐색 대상의 수가 증가할수록 비교 횟수도 기하급수적으로 늘어나기 때문에 처리시간이 오래 걸린다. 다중 속성 데이터에서는 각 데이터의 속성별로 비교해야 하므로 비교 시간 및 횟수는 더욱 증가한다.

본 논문에서는 다중 속성을 포함한 경로 계산에서 빠른 처리속도를 보장하도록 Top-n 스카이라인을 이용한 다차원 TSP 해결기법인 TS-MDT(Top-n Skyline-Multi Dimensional TSP) 알고리즘을 제안한다.

TS-MDT는 연산시간을 줄이기 위해 탐색에 들어가기 전에 Top-n 스카이라인 알고리즘을 사용하여 다중 속성 데이터에서 TSP의 후보 경로를 연산한다. 이후, 동적 계획법 알고리즘을 사용하여 반환된 후보 경로들을 연결해서 최소비용을 가지는 하나의 경로를 최종적으로 반환한다. 논문에서는 같은 데이터 개수일 때 다차원을 사용하는 제안방법과 1차원을 사용하는 기존 기법의 연산시간을 비교하여 제안기법의 현실적인 유용성을 증명한다. 또한, 기존 기법과의 함수 호출 횟수를 비교하여 제안하는 알고리즘의 효율성을 증명한다.

본 연구의 공헌 내용은 다음과 같다.

- 기존의 1차원 속성을 다루는 외판원 순회문제를 다차원 속성을 이용한 최단 경로 탐색기법으로 확장하였다.
- TSP는 데이터 수에 따라 처리속도가 증가하여 실생활에 적용하기가 어려웠지만, 논문에서는 해결방안을 제시하였다.
- 다중 속성을 평가함으로써 사용자의 다양한 요구에 맞는 서비스를 제공할 수 있는 알고리즘을 개발하였다.
- 스카이라인 질의를 사용하여 사용자가 선호하는 다차원 데이터의 수를 감소시키고 연산시간을 줄임으로써 실용적인 최적 경로를 제공한다.

본 논문은 2019년 춘계학술대회에서 발표된 선행 연구[2]의 확장 연구로 작성된 것이며 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 관해 설명하고 3장에서는 Top-n 스카이라인 질의를 이용하여 다차원 데이터에서의 TSP 해결기법을 제안한다. 4장에서는 실험을 통해 다차원에서의 Top-n 스카이라인 질의를 이용한 TSP의 성능을 분석하고 효율성과 유용성을 증명한다. 5장에서는 결론을 기술한다.

2. 관련 연구

2.1 외판원 순회문제

최적 경로 문제에 사용되는 외판원 순회문제(TSP)는 모든 데이터가 이어져 있다는 가정하에 그 데이터들을 한 번씩만 방문한 후 다시 출발점으로 되돌아오는 최단 경로를 반환한다. TSP는 차량 경로 문제, 일정계획, 스케줄링이나 택배, 음식배달 등의 문제에 적용될 수 있다.

TSP[3]는 사용되는 데이터가 많아질수록 기하급수적으로 늘어나는 비교 횟수 때문에 최적해 보다는 근사해를 구하는 기법으로 연구됐으며 유전자 알고리즘이 근사해 기법의 대표적인 예이다[4].

기존의 연구들은 1차원 속성을 대상으로 TSP를 적용하지만 본 논문에서는 사용자들의 다양한 요구를 반영하기 위해 다차원속성을 대상으로 TSP 알고리즘을 적용하여 최단 경로를 구한다.

Fig. 1은 각 노드(Node) 사이의 거리를 나타내고 있으며 TSP에서 사용하는 대표적인 예이다. 노드 중 출발점을 a라고 가정하면 TSP의 결괏값은 출발점으로 되돌아오는 간선의 비용이 62로 가장 적은 {a, d, b, c, a}나 {a, c, b, d, a}의 두 가지 경로 중 하나를 반환한다. 이처럼 기존의 TSP는 1차원 속성을 대상으로 하므로 각 노드가 포함할 수 있는 소요시간, 이동 거리, 선호도 등 다양한 속성을 동시에 평가할 수 없다.

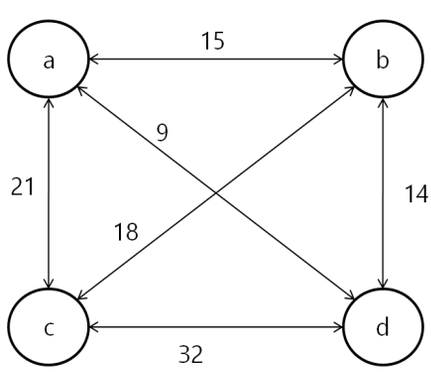

Fig. 1. Traveling Salesman Problem

2.2 Top-n 스카이라인 질의

스카이라인 질의(Skyline query)는 2차원 공간에서 서로 지배되지 않는 객체를 탐색하는 기법으로 전체 데이터에서

서로 지배되지 않는 객체들의 집합을 스카이라인이라 한다[5]. 2차원 이상의 속성에서 어떤 객체 p가 다른 객체 q보다 모든 차원 또는 적어도 한 차원 이상에서 작은 값을 가질 때 p는 q를 지배한다고 표현하며 작은 값을 갖는 p가 스카이라인 객체에 포함된다.

Fig. 2는 스카이라인 질의의 결과를 타나낸다. x축은 비용, y축은 거리를 나타내고 그래프에 있는 점들은 호텔이라고 가정하자. 호텔 d는 호텔 b보다 거리가 짧고 비용도 더 싸다. 이때, 호텔 b는 호텔 d에 지배된다고 표현한다. 따라서 하나의 객체를 기준으로 1사분면에 있는 점들은 각 속성에 대해 기준점 객체의 속성에 의해 지배되며 탐색 대상에 포함하지 않는다.

호텔 a와 d에서 비용(Cost)은 a가 저렴하지만, 거리(Distance)는 d가 가까우므로 서로를 지배하지도 지배되지도 않는 관계가 된다. 이처럼 두 속성에서 완전한 지배 관계가 아닌 경우에는 스카이라인 객체에 포함하며 의사결정 기준에 따라 사용자가 판단하도록 한다.

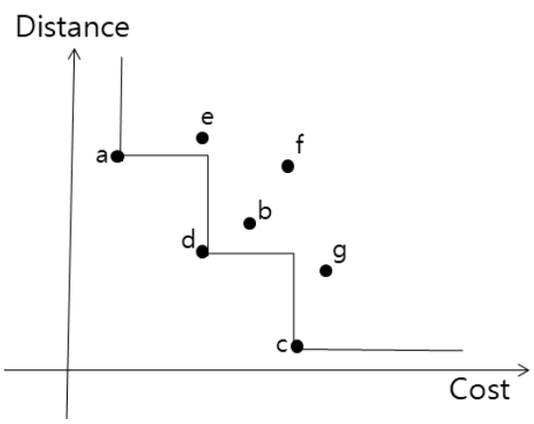

Fig. 2. SKyline Query

Top-n 스카이라인 질의는 다차원속성을 갖는 데이터에서 스카이라인을 구하고 의미 있는 n개의 데이터를 반환하는 질의이다[6]. 이때, 사용자의 선호도는 속성을 비교 및 판단하는 기준으로 사용되며 최종결과는 n개의 데이터가 된다. 여기서 n개의 데이터를 특정하는 대표적인 방법으로는 K개의 차원에서 다른 차원들에 지배되지 않는 객체들을 반환하는 K-dominant 스카이라인이 있다. 처리기법에는 One-scan, Two-scan, Sorted Retrieval 알고리즘[7] 등이 있고 그중 Two-scan 기법이 가장 우수한 처리속도를 보인다. Top-n 스카이라인 질의는 K의 값을 1부터 증가시켜가면서 n개의 데이터가 반환될 때까지 K-dominant를 호출하게 된다.

Table 1은 5개의 속성 (s1-s5)을 갖는 객체 (p1-p5)를 나타내며 Top-n 스카이라인과 K-dominant 알고리즘에 대한 예는 다음과 같다.

K=1일 때, Top-2를 구하기 위해 1차원 속성으로 비교하면 모든 객체가 서로에게 지배되지 않는다. 예를 들어 p1과 p2를 비교하면 p1은 s5에서 p2에 의해 지배되지만, s1에서는 p2를 지배하여 결론적으로는 서로 지배되지 않는다. 이를 적용하면 {p1, p2, p3, p4, p5}가 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[8]을 사용한다면 상위 2개의 후보군을 구할 수 있다. 그러나 이는 어느 시점에서 알고리즘을 사용해야 하는지 명확하지 않고 연산시간을 줄인다는 본 논문의 취지와는 맞지 않는다.

속성 K=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.

Table 1. Objects and Attributes

속성 / 객체	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

같은 방법으로 p2, p5 데이터 역시 p3에 의해 지배되기 때문에 스카이라인 후보군에서 제외되므로 {p3, p4}만 남으며 Top-2 스카이라인 질의는 {p3, p4}를 반환한다.

본 논문에서 Top-n 스카이라인 질의는 TSP를 위해 각 데이터를 방문할 때 경로에 포함되는 데이터의 수를 줄이기 위해 사용한다.

2.3 동적 계획법

동적 계획법은 Fig. 3과 같이 하나의 문제를 여러 개의 작은 문제로 나누어서 해결한다. 분할정복 알고리즘과 차이점은 문제를 계산한 후 결괏값을 재활용하는 것이다[9]. 따라서, 같은 연산을 여러 번 할 필요 없이 저장된 결과를 불러오는 것으로 문제를 해결할 수 있다.

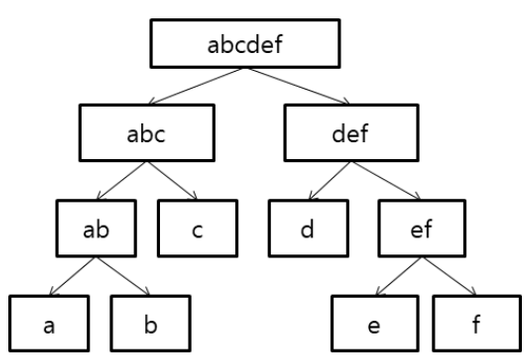

Fig. 3. Dynamic Programing

본 논문에서 동적 계획법은 비트 마스크(Bit Mask)를 이용한 경로 탐색 알고리즘으로 사용한다. 비트 마스크란 이진수를 자료구조로 사용하며 빠른 수행시간과 코드의 간결화라는 장점이 있다. Fig. 3의 동적 계획법에서는 6개 데이터 a, b, c, d, e, f에 대한 비트 마스크를 만든다. 방문 여부를 비트 마스크에 값으로 표현하고 이를 배열의 색인으로 사용하여 결괏값을 저장한다. 예를 들어 a를 방문했다고 가정한다면 $000001_{(2)}$로 표현되고 배열 색인 1번에 a의 결괏값이 저장된다.

3. 다차원 데이터의 최적 경로 탐색기법

본 절에서는 제안기법인 TS-MDT에 적용한 K-dominant와 Top-n 스카이라인 질의 및 Dynamic Programming 알고리즘의 구현 과정을 설명한다.

3.1 탐색 경로 축소

논문에서는 다차원속성을 갖는 데이터에 대해 탐색 대상을 축소한 후 최적 경로 탐색을 위해 세 개의 알고리즘을 개선하여 새로운 알고리즘을 제안한다.

논문의 알고리즘에서 사용하는 기호와 정의는 Table 2와 같다.

Table 2. Symbols and Definition

Symbol	Definition
N	# of objects in dataset
dimension	# of dimensions
T	Set of Skyline candidate
t	# of candidate in T
limit	Stands for n, the value of Top-n
D	Set of all data
s	Currently located data
visit	Checks whether each data is visited
arr	Set of distance data
R[s][]	Array indicating whether each data item is connected or not
cache	Previously calculated minimum cost array
k	# of dominant properties for K-dominant query
x	Variable to recall the stored value

최적 경로를 탐색하는 과정에서 임의의 객체 a와 객체 b가 연결될 때 이를 이어진다고 표현하며 연결된 경로를 통해 두 객체 (또는 장소) 사이를 이동하면서 최적 경로를 탐색한다. 이동이란 최단 경로를 찾기 위한 연산이 하나의 객체에서 다른 객체로 넘어가는 것을 의미한다.

Fig. 4는 객체들의 연결 상태를 표현한 2차원 배열 R이다. 배열에서 1은 해당 행과 열의 두 객체가 이어진 것을 나타내며 0은 연결되지 않은 것이다. Fig. 4(a)의 모든 객체는 자기 자신을 제외한 다른 객체와 연결된 것을 나타낸다. 객체가 서로 연결되지 않으면 동적 계획 탐색 알고리즘의 연산 과정에서 해당 객체에 대한 함수호출은 발생하지 않는다.

Top-n 스카이라인 질의는 K-dominant 알고리즘을 이용하여 객체들의 다중속성을 평가하고 객체들의 연결관계를 Fig. 4(a)에서 Fig. 4(b)로 변환한다. Fig. 4(a)에서 현재 위치가 a라고 할 때 a에서 갈 수 있는 후보 경로 4개 (b, c, d, e)를 Fig. 4(b)와 같이 스카이라인 질의를 통해 b와 e로 감소시킨 후 다시 이차원 배열 R에 저장한다.

후보 경로에 대한 축소 연산이 필요한 이유는 다음과 같다. 기존에는 최적 경로를 탐색하기 위해서 4개의 노드를 모두 방문해야 하지만 제안기법에서는 다중속성의 평가를 통해 방문 대상을 두 개로 줄임으로써 연산횟수를 감소시킨다. 총 객체의 개수만큼 반복하면 모든 행의 변환이 종료되며 새로 구성된 R 배열을 이용하여 동적 계획법 알고리즘을 실행한다. 개선된 알고리즘을 적용한 최적 경로 탐색 과정은 다음 절에서 자세히 설명한다.

	a	b	c	d	e
a	0	1	1	1	1
b	1	0	1	1	1
c	1	1	0	1	1
d	1	1	1	0	1
e	1	1	1	1	0

(a)

	a	b	c	d	e
a	0	1	0	0	1
b	1	0	0	0	1
c	0	1	0	1	0
d	1	1	0	0	0
e	1	1	1	0	0

(b)

Fig. 4. R[][] Array

3.2 스카이라인 기반의 TSP

다차원의 데이터 중 속성에 대한 평가를 통해 경로에 포함할 대상을 선택하기 위해 일차적으로 K-dominant 알고리즘을 적용한다.

Fig. 5의 K-dominant 알고리즘은 각 객체 사이의 지배관계를 평가하기 위한 속성 개수 k를 입력으로 하여 실행되며 다중 속성 객체인 p와 p'에 대해 다음의 수식을 기준으로 지배 여부를 판단한다.

$$(p,\ p') \in D,\ p \neq p' \text{ (D는 Dataset)} \qquad (1)$$

알고리즘의 1행에서는 스카이라인 후보들의 집합인 T를 전체 객체로 초기화시킨다. 이후에는 객체들의 속성을 서로

비교하여 지배되는 객체를 T에서 제거한다(2~11행). 예를 들어 p가 p'에게 지배된다면 p를 T에서 제거하고(8행) 반대의 경우에는 p'를 제거한다.

for 문에서 방문 여부를 표시하는 check[] 배열을 통해 이미 들렸던 곳을 확인하고 방문한 지점을 제외한 후에 계산한다(3~7행). 마지막으로 Top-n의 질의를 위해 서로 지배되지 않는 객체들의 집합인 T를 반환한다. 예를 들어, 지배 관계를 판단하기 위한 속성을 하나만 지정한다면 k=1이 되며, 하나의 속성에 대해 각 객체의 지배 관계를 판단하기는 어렵다. 객체 A와 B의 두 속성이 p1={3, 4}, p2={5, 2}의 값을 갖는다면 A는 p1에서 B를 지배하지만 B는 p2에서 2의 값으로 A를 지배하게 되므로 두 객체는 모두 T로 반환된다.

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
```

Fig. 5. K-dominant Algorithm

Top-n 알고리즘은 Fig. 6과 같이 각 조건문에서 수행되는 내용이 같으므로 하나로 코드 블록으로 통합이 가능하지만 가독성을 위해 네 개의 조건을 나누어 표현하였다. 또한, limit은 전체 객체 개수인 N과의 혼동을 피하고자 Top-n의 n을 재지정한 것이다. limit은 전체 객체 수의 절반이다.

Top-n 스카이라인 알고리즘은 k를 1부터 늘려가면서 K-dominant 알고리즘을 호출하여 기준치인 limit에 근접하거나 일치하는 개수가 반환될 때까지 반복한다. 알고리즘을 통해 반환된 객체 집합 T는 현재 위치인 s에서 갈 수 있는 후보객체에 해당한다. 반환되지 않은 나머지 객체들은 특정 속성에서 다른 객체에 의해 지배되어 반환되지 않아서 경로 탐색 과정에서 제외된다. 연결 상태는 2차원 배열 R의 R[s] 행에 저장함으로써 현재 위치인 s에서 연결된 객체를 나타낸다. 알고리즘에서는 R[s]=T로 의미를 나타내었다.

Top-n 알고리즘은 지배 속성의 개수를 k=1에서 하나씩 증가시켜가면서 결과가 Top-n에서 찾는 limit에 가까운 수를 반환할 때까지 함수 Kdo(k) 호출한다. 따라서 처음에는 지배 관계를 판단하기 위한 속성을 하나만 지정하여 k를 1로 초기화한 후 지배 속성을 증가시키면서 속성 개수만큼 루프를 실행한다(3~19행). 4행에서 K-dominant 알고리즘을 호출하여 객체 집합 T를 반환받고 T에 속해 있는 객체의 개수를 t에 저장한다(5행).

변수 t는 T에 포함된 후보객체의 개수를 나타내며 k가 1일 때는 다중 속성에서 서로 지배할 수 있는 객체가 극히 적어서 객체 대부분이 반환되어 가장 큰 수를 가진다. 하지만 k가 증가할수록 지배하는 속성의 수가 많아지므로 T에 포함되는 객체의 수는 적어진다.

예를 들어, K=2이면 비교할 속성수가 2개로 늘어났기 때문에 K=1일 때 보다 지배 가능성이 커지기 때문에 지배되지 않는 객체의 수는 감소한다. 결과적으로 k=1일 때 t가 limit보다 작으면 k가 증가하더라도 t는 limit보다 작으므로 함수를 끝낸다(6~8행).

반대로 9행은 마지막 루프일 때 t가 limit보다 크면 모든 k에서도 t는 limit보다 크기 때문에 함수를 끝낸다. 12행과 15행은 위의 두 경우를 제외한 것으로서 t가 limit과 같으면 조건을 만족했으므로 함수를 끝낸다. 만일 루프 중간에 t가 limit보다 작아진다면 k-1의 값으로 K-dominant 알고리즘을 재수행 후 T를 반환받고 알고리즘을 종료한다. 여기에서 재연산을 하는 이유는 루프 중간에 limit보다 작은 t의 개수가 나올 때 t=0이 될 수 있기 때문이다. t=0이라면 현재 위치에서 이동할 수 있는 객체가 없어지므로 재연산을 통해 0이 아닌 t를 찾기 위한 것이다. 알고리즘의 최종결과는 방문해야 할 객체들의 연결 관계를 표시한 이차원 배열 R로 반환된다.

TOP-n Algorithm

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

Fig. 6. Top-n Algorithm

동적 계획법 알고리즘 Fig. 7은 Top-n 스카이라인 질의로 축소한 경로들을 모두 확인해 보는 알고리즘으로 연산된 모든 경로 중 가장 최적의 경로를 반환한다.

동적 계획법 함수인 DP는 현재 위치인 s와 지금까지 방문

한 노드들을 표시한 값인 visit을 입력값으로 받아서 실행된다. 여기서 visit은 2진 비트로 표현된 비트 마스크이다. 모든 지점을 방문하면 재귀가 종료된다(1~2행). 현재 위치와 방문한 곳의 최소값이 저장된 cache 배열로 부터 해당 객체의 최소값을 변수 x에 초기화한다(3행). cache 배열에는 같은 연산을 여러 번 하지 않도록 한번 연산 된 결과가 저장되어 있다. x는 cache에 저장된 값을 불러와 값이 있다면 이미 연산된 경로로 확인하고 0이라면 연산 되지 않은 것이므로 다음 경로 연산을 진행하기 위한 변수이다. 즉, x가 0이 아니라면 이미 한번 연산 된 경로이므로 저장된 값을 바로 반환한다(4~5행). x가 0이라면 최댓값(MAX)을 저장한다(6행). 총 객체의 개수만큼 반복문을 실행하고 배열 R을 통해서 현재의 경로에서 이전에 방문했던 객체이거나 현재 위치와 이어져 있지 않은 객체는 해당 루프를 중단하고 다음 루프를 실행한다(7~9행). x에 현재 객체의 비용에 다음 호출 시 반환된 값을 더한 값과 현재의 x 값을 비교하여 더 낮은 값을 저장한다(10행). cache[s][visit]에 연산된 x 값을 저장하고 반환한다(11~12행).

Dynamic Programing Algorithm

```
DP(s,visit) function
1:   if visit == all
2:      return arr[s][0]
3:   initialize x =cache[s][visit]
4:   if x≠0
5:      return x
6:   x=max
7:   for every J∈N
8:      if visit≠1 or R[s][J]==0
9:         start next roop
10:     x = min(x, DP(I,visit+(1<<J)) + arr[s][J])
11:  cache[s][visit]=x
12:  return x
```

Fig. 7. Dynamic Programming Algorithm

4. 실 험

제안기법의 실험은 비트 마스크를 사용하여 연산하는 동적 탐색 알고리즘의 특성상 24개의 데이터에서 2차원 배열의 크기가 약 1GB를 넘게 되므로 메모리에 제약이 있다. 실험에서는 데이터를 무작위로 생성하였으며 배열의 메모리 용량 문제로 인하여 23개의 데이터를 사용하였다. 각 데이터는 5개의 속성을 가지며, 실험횟수는 총 1만 회를 수행하였다.

제안기법의 실험환경은 Table 3과 같다.

비교 대상인 동적 계획법은 3절의 Fig. 7에서 제안한 알고리즘을 사용하였다. 각 데이터는 1개의 속성을 갖고 실험횟수는 동일하게 1만 회 진행하였으며 실험은 같은 시스템 사양에서 이루어졌다. 실험에서는 같은 데이터 개수일 때 기존의 1차원과 제안한 다차원 데이터의 연산시간, 평균호출횟수의 차이를 비교하였다.

Table 3. Experiment Environments

Division	Contents
System	Intel i5 CPU 2.3 GHz, RAM 4GB
# of data	19~23 objects
Dimensions	5
# of experiments	10,000

Fig. 8은 데이터 수 변화에 따른 질의 처리시간을 비교한 것으로 단위는 초(Second)이다. 실험 결과 23개의 데이터에서 Dynamic은 54.7초 제안기법은 3.24초로 약 18배의 차이를 나타낸다.

제안기법은 동적 계획법을 수행하면서 Top-n 스카이라인 질의에서 후보 데이터를 줄이는 연산시간을 포함하지만, 기존의 TSP 동적 계획법보다 더 빠른 처리시간을 나타냈다. 처리시간에 대한 증가율 측면에서도 제안기법은 상대적으로 낮은 변화를 나타내고 있다.

연산속도의 차이를 자세히 살펴보면 다음과 같다. Fig. 8의 동적 계획법에서 데이터가 19에서 21까지, 2개 증가할 때 연산속도는 약 10배의 차이를 보인다. 이는 연산횟수가 기하급수적으로 늘어나는 TSP의 단점을 보여준다. 하지만 제안기법의 경우는 같은 위치에서 처리속도의 증가가 크지 않으며 데이터 개수가 증가할수록 제안기법과 Dynamic의 효율 차이는 더 크게 나타난다.

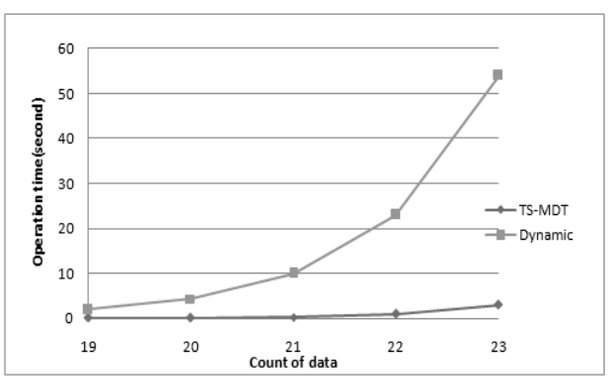

Fig. 8. Average Operation Time

Fig. 9는 제안한 기법의 효율에 대한 실험으로 동적 계획법과 제안기법의 평균함수호출 횟수를 비교한 것이다. 결과적으로 제안기법인 TS-MDT 기법의 함수 호출횟수가 Dynamic 기법보다 낮고 데이터의 증가에 따른 크기 변화 또한 낮게 유지하고 있으므로 알고리즘 처리에 있어서 제안기법이 더 효율적이다.

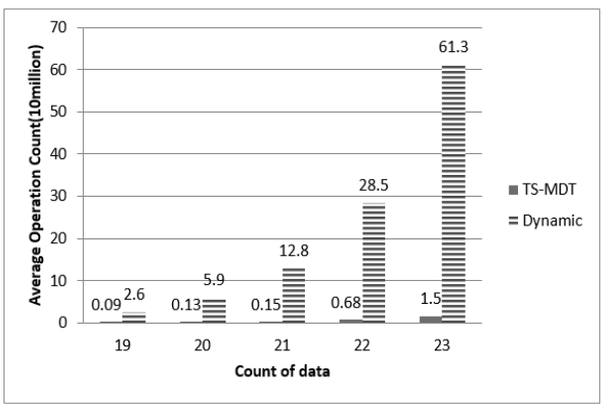

Fig. 9. Average Operation Count

Fig. 8과 Fig. 9에서 연산시간과 호출횟수에서 큰 차이가 나타나는 이유는 Top-n 스카이라인 질의에 있다. 스카이라인 질의에서 데이터를 지배원칙에 따라 제거함으로써 방문할 경로의 개수를 최소 $(n-1)!/2^n$만큼 감소시킴으로써 성능에 차이가 발생하는 것이다. 예를 들어 기존의 기법은 19에서 20으로 데이터 개수가 늘어날 때 경로의 개수는 18!×19가 되지만 제안기법은 약 $18!/2^{19}\times10$이 된다.

5. 결 론

본 논문은 위치기반 서비스에서 생성된 데이터를 활용하는 외판원 순회문제 알고리즘에서 처리시간이 너무 오래 소요되는 단점과 사용자의 다양한 요구에 맞추지 못하는 1차원 데이터의 문제에 대한 해결책으로 Top-n 스카이라인 질의를 이용한 다차원 데이터에서의 외판원 순회문제(TS-MDT)를 제안하였다.

제안기법은 다차원속성을 TSP에서 처리하기 위해 대상 데이터의 수를 스카이라인으로 축소함으로써 처리시간과 사용자의 다양한 선호도에 부합하는 TSP를 구현하였으며 실용적 결과를 나타내었다. 즉, 기존 기법과의 비교 실험에서 23개 데이터를 대상으로 할 때 약 18배 정도 이른 연산시간을 보였다.

본 연구에서는 TSP에서 데이터가 증가할수록 처리시간이 기하급수적으로 증가하는 문제로 인해 모바일 장치에서 최대 23개의 데이터를 대상으로 하였다. 그러나 지금까지의 데이터 크기에 따른 차이를 보면, 더 많은 데이터에 대한 실험에서도 실용성에 대한 긍정적인 결과를 기대할 수 있을 것으로 보인다.

References

[1] S. M. Nam, M. S. Park, K. S. Kim, and S. J. Kim, "A Study on the Regulations and Market of Location Based Service (LBS)," *Journal of Internet Computing and Services*, Vol. 15, No.4, pp.141-152, 2014.

[2] C. G. Jin, J. Kim, and D. Oh, "Multi-dimensional Traveling salesman problem using Top-n Skyline query," *Korea Information Processing Society Spring Conference*, Vol.26, No.1, pp.371-374, 2019.

[3] S. H. Kwon, S. M. Kim, and M. K. Kang, "Cost Relaxation Method to Escape from a Local Optimum of the Traveling Salesman Problem," *Journal of Korean institute of industrial engineers*, Vol.30, No.2, pp.120-129, 2004.

[4] K. B. Kim and D. H. Song, "Path Search Method using Genetic Algorithm," *The journal of the Korea Institute of Maritime Information & Communication Sciences*, Vol.15, No.6, pp.1251-1255, 2011.

[5] Z. H. Li and Y. B. Park, "Efficient Processing using Static Validity Circle for Continuous Skyline Queries," *Journal of KIISE:Databases*, Vol.33, Issue 6, pp.631-643, 2006.

[6] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with Presorting," *Proceedings 19th International Conference on Data Engineering*, pp.717-719, 2003.

[7] 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.

[8] H. S. Park and J. W. Lee, "Correct Linear Skyline Algorithm in High-Dimensional Space," *Journal of KIISE*, Vol.45, No.10, pp.1089-1095, 2018.

[9] R. Neapolitan and K. Naimipour, "Foundation of Algorithms Using C++ Pseudocode 3rd edition," Jones & Bartlett, 2004.

진 창 균

https://orcid.org/0000-0002-0153-4552

e-mail : jcg6074@naver.com

2015년 ~ 현 재 삼육대학교 컴퓨터-메카트로닉스 공학부 학사과정

관심분야 : Traveling Salesman Problem, Skyline Query

오 덕 신

https://orcid.org/0000-0001-8004-3614

e-mail : ohds@syu.ac.kr

1997년 ~ 현 재 삼육대학교 경영정보학과 교수

관심분야 : MIS, e-Learning, e-Business & Artificial Intelligence

김 종 완

https://orcid.org/0000-0003-4716-8380
e-mail : kimj@syu.ac.kr
2016년 ~ 현 재 삼육대학교
스미스학부대학 조교수
관심분야 : Big Data & Smart Data, Skyline Query, Data Mining, Machine & Deep Learning