대규모 TSP의 효율적 해결을 위한 분할 및 병합 알고리즘

설춘룡°, 신태지, 양명국
울산대학교 전기전자정보시스템공학과
{crseol°, shintaezi, mkyang}@mail.ulsan.ac.kr

Korea Information Science Society

Choonryong Seol°, Taezi Shin, Myungkook Yang
School of Electrical Engineering, University of Ulsan.

요 약

TSP(Traveling Salesman Problem)는 주어진 N개의 City들을 단 한번씩만 거쳐 출발지로 되돌아오는 경로들 중 가장 작은 비용이 소요되는 경로를 찾는 문제이며, 고전적인 최적화 문제로 널리 알려져 있다. City의 수가 증가하면 최적 Tour를 찾기 위한 연산 시간이 길어지는 단점이 있다.

본 논문에서는 대규모 TSP의 효율적 해결을 위해 새로운 알고리즘을 제안한다. 본 논문에서는 대규모의 City들의 집합을 두개의 소집합으로 분할하고, 병합을 위해 하나의 Junction City를 지정한다. 분할된 두개의 소집합 각각의 최적 Tour를 구한 후 분할된 두 최적 Tour를 병합하여 하나의 근사 Tour를 구한다. 지정된 Junction City는 병합 시 최적 병합조건을 구하는 연산의 간편화를 기대할 수 있다.

1. 서 론

TSP(Traveling Salesman Problem)는 주어지 N개의 City들을 단 한번씩만 거쳐 출발지로 되돌아오는 경로(Tour)들 중 가장 작은 비용이 소요되는 경로를 찾는 문제로 고전적인 최적화 문제로 널리 알려져 있다. TSP의 해를 구하는 방법은 네트워크 최적화, 공장에서의 공정처리 순서 결정 문제, 전력 전송 경로 설정 등 광범위한 분야에 적용될 수 있다. 이와 같이 다양한 응용분야에 널리 활용되는 TSP는 City 개수 증가에 따라 해를 구하는데 요구되는 계산량이 기하급수적으로 증가하여 NP-Hard 문제로 분류된다.

TSP의 해결을 위한 여러 이론들로 발견적 방법, 확률적 방법, 결정적 방법등이 발표되었다.

결정적 방법 중 열거법과 B&B 알고리즘(Branch and Bound Algorithm)이 널리 알려져 있다. 열거법은 가능한 Tour를 모두 열거하고 그 결과를 서로 비교하여, Tour 비용이 가장 작은 Tour가 최적 Tour가 된다. 하지만 City의 개수가 조금만 증가해도 가능한 Tour가 너무 많아져, 이들을 열거하고 비교하는 것이 불가능하게 되는 단점을 가진다. Land와 Doig[1]에 의해 처음 제안된 B&B 알고리즘은 열거법의 단점을 개선한 방법이다. B&B 알고리즘은 이진트리를 응용하여 가능한 Tour들의 집합을 두개이상의 부분으로 분할하여 최적해를 찾는다. 이 또한 최악의 경우 (N-1)!개의(N: City의 수) 연산을 필요로 한다. 하지만, 평균 연산 수는 상당히 신뢰할 수 있고, 개념이 단순하여 이해가 쉬우면, 응용이 용이하다. Little, Murty, Sweeney, Karel[2]이 제안한 알고리즘은 B&B 알고리즘의 대표적은 예이다. 그러나 이러한 B&B 알고리즘은 또한 대규모 TSP에서 평균 연산량이 급격하게 증가하기 때문에, 연산 시간이 수용할 수 없을 정도로 길어지는 문제가 있다.

본 논문에서는 대규모 TSP를 해결하는데 소요되는 계산량을 대폭 줄이고 최적해에 근사한 결과치를 얻을 수 있는 TSP 분할 및 병합 알고리즘을 제안하고, 알고리즘의 실효성을 검토하였다. 제안된 알고리즘은 주어진 대규모 TSP를 복수개의 중·소규모 TSP로 분할하여 처리한다. 분할 과정에서 경계선상에 위치한 임의 Junction City를 지정하여, 이들 Junction City가 포함된 중·소규모 TSP의 최적해를 구한다. 그리고 분할된 중·소규모 TSP의 최적해를 병합하여 하나의 근사해를 구한다. 2.1절에서는 TSP에 관한 수학적 모델을 제시하였다. 그리고, 2.3절 분지한계법에 대해 간략히 소개하였다. 분할과 병합의 방법은 2.4장에서 자세히 소개하였다. 또한 2.5절에서는 제안된 알고리즘을 컴퓨터 시뮬레이션을 통해 그 타당성을 검증하였다. 3장에서는 결과의 고찰과 향후과제에 대하여 고찰하였다.

2. 본 론

2.1. TSP의 수학적 모형

유클로이드 공간에서 N개 City들의 집합 V와 City들간 Edge의 Cost들의 집합(비용행렬) A에 대한 Network $G=(V,A)$로 표현할 수 있다. Network G의 변수를

$$x_{ij}=\begin{cases}1, & <i,j>\text{가 } Tour\text{에 포함될 경우}\\ 0, & <i,j>\text{가 } Tour\text{에 포함되지 않을 경우}\end{cases} \quad (2.1)$$

으로 정의 한다.

TSP(Traveling Salesman Problem)는 다음과 같이 정수 계획법으로 모형화가 된다.

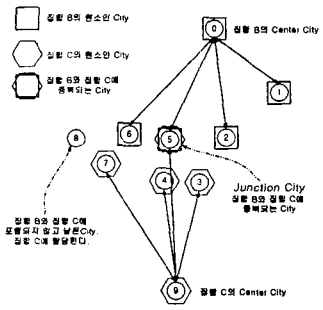
최소화 : $$z=\sum_{(i,j)} C_{ij}\,x_{ij} \quad (2.2)$$

(단, C_{ij}는 City i에서 City j 사이의 Cost(i, j))

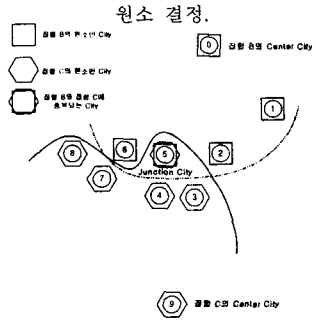
제약조건 : $$\sum_{i=1}^{N} x_{ij}=1 \text{ (모든 j에 대해)} \quad (2.3)$$

$$\sum_{j=1}^{N} x_{ij}=1 \text{ (모든 i에 대해)} \quad (2.4)$$

단 하나의 Tour만을 형성함. (2.6)



(a) 알고리즘 1의 1.2 소집합의 원소 결정.



(b) 알고리즘 1의 2의 최종 소집합

그림 1. 분할 알고리즘의 개념도

식(2, 3)과 식(2, 4)는 Tour를 구성하는 각 City는 들어오는 Edge와 나가는 Edge가 필요함을 말한다. 그리고 식(2, 5)에서는 완전한 Tour가 완성되기 전에 폐루프를 허락하지 않는다.

2.2. 분지한계법

본 논문에서는 B&B 알고리즘에 의해 구해진 근사 Tour와의 비교를 위해 분지한계법을 사용하여 최적 Tour를 구하였다. Little 등[2]이 제안한 방법은 분지한계법에서 가장 널리 이용되고 있는 알고리즘이다. 자세한 사항은 참고문헌[2]를 참조.

2.3. 분할과 병합

본 논문에서 대규모의 City를 보유한 TSP를 위해 새로운 알고리즘을 제안한다. 본 논문의 알고리즘은 기본적으로 대규모 City들의 집합을 Junction City를 공통으로 포함하는 두개의 소집합으로 분할하며, 각 소집합의 최적 Tour를 구한다. 두 소집합의 최적 Tour들을 Junction City를 사용해 간단하게 병합하여 하나의 근사 Tour를 구한다. 분할과 병합을 통해 최악의 경우의 연산량이 $(N-1)!$ 에서 $2\times(\lfloor \frac{N}{2} \rfloor -1)!$으로 감소한다.

2.3.1. 가정

분할 알고리즘은 기본적으로 분할의 방법이 간단해야 한다. 아래와 같은 4 가지 가정을 바탕으로 알고리즘 1.을 제안한다.

가정1. 유클리드 공간에서 Cost는 거리의 개념으로 나타낸다.
가정2. Cost가 큰 Edge들은 최적 Tour에 포함될 가능성이 낮다.
가정3. 소집합 B와 C의 각 원소들 중 일부는 서로 근접할 것이다.
가정4. 두 소집합의 원소들이 균등하게 분할될수록 최적 Tour에 가깝다.

1. 다수 City들의 집합 A를 두개의 소 집합 B, C로 분할 한다.

1.1. Center City 선정 :

집합 A의 원소인 City i와 City j의 Edge Cost (i, j)들 중 가장 큰 값의 Edge를 선정하여 City i를 집합 B에 포함 시키고, City j를 집합 C에 포함시킨다. 이때의 City i를 집합 B의 Center City라고 하고, City j를 집합 C의 Center City라고 한다.

1.2. 각 소집합의 원소 결정 :

집합 A의 원소들 중 각 Center City와 인접한 City(Cost가 작은 Edge에 연결된 City)을 선정하여 각 소 집합 B와 C에 포함시킨다. 소집합 원소의 개수가 개 일 때까지 1.2.를 반복한다.

1.3. 집합 B와 집합 C에서 중복되는 원소가

1.3.1. 있을 경우,

소집합 B의 Center City와 거리와 집합 C의 Center City와 거리의 차가 가장 작은 원소 City만을 남기고, 나머지는 각 Center City와 먼 쪽의 소집합에서 제외시킨다. (이때 남겨진 중복된 City가 Junction City가 된다.)

1.3.2. 없을 경우,

소집합 B의 원소 City들과 집합 C의 City들과의 거리가 가장 작은 City를 선정하여 집합 C에 포함시킨다.(이때 선정된 City가 Junction City가 된다.)

2. 최종 소집합 완성 :

소집합 B, C에 포함되지 않는 집합 A의 나머지 원소들은 집합 B의 원소 City들과 집합 C의 원소 City들 중 가까운 곳에 있는 소집합에 포함시킨다. 분할 과정을 종료한다.

알고리즘 1. 분할 알고리즘

두개의 소집합에서 구해진 분할 최적 Tour를 병합한다.

1. Junction City로 연결된 Edge들 중 잘라낼 Edge선택 :

unction City를 중심으로 두 소집합을 연결 가능한 4개의 Edge 중 하나를 선택하고, 선택한 Edge에 인접한 City들과 Junction y 사이의 Edge를 제거한다. 이때의 근사 Tour의 Cost를 구한

2. 반복 :

.1에서 선택한 Edge를 제외하고 다른 Edge를 선택하여 1.1을 수한다.

3. 근사 Tour의 선택 :

개의 근사 Tour의 Cost를 비교하여 그중 가장 작은 값을 가진 Tour를 선택한다.

완전한 근사 Tour 완성 :

선택된 근사 Tour를 최종 근사 Tour로 설정하고, 과정을 마친다.

알고리즘 2. 병합 알고리즘

2.3.2. 분할

유클리드 공간 S에 포함되는 City들의 집합을 집합 A라 할 때, 두 City들의 소집합은 각각 집합 B와 집합 C $(B,C\subset A)$ 이다. 병합 과정의 간략화와 오차가 발생을 최소로 하기 위해 두 가지 특별한 City를 정의한다.

○Junction City : 두 소집합 사이에 공통적으로 포함되는 City. 두 소집합간의 간단하고 쉬운 병합을 위해 지정한다.
○Center City : 두 소 집합에서 각 소집합에 처음으로 포함 되는 City.

각 City들 간의 Edge Cost (i, j)가 가장 큰 값을 가지는 Edge <i, j>의 City i, j 각 소집합의 원소들은 Center City와 가까운 원소를 중심으로 하나의 집합을 완성한다. 원소의 개수가 $\lfloor \frac{N}{2} \rfloor$ (N이 홀수 일 때) 또는 $\frac{N}{2}$ (N이 짝수 일 때) 될 때 까지 선정하게 된다.

2.4.3. 병합

분할된 각 소집합들의 최적 Tour를 하나의 근사 Tour로 병합시키는 방법은 가능한 간단한 방법이 필요하다. Junction City는 두 소집합들의 병합을 간편화 시킨다.

Junction City가 있을 경우, 그림 2(a)에서 보는 것과 같이 두개의 소집합을 병합시킬 수 있는 4개의 Edge들을(ⓐ, ⓑ, ⓒ, ⓓ) 명확하게 알 수 있다. 이 4개의 Edge들 중 하나를 선택하여 간단하게 근사 Tour를 만들게 된다.

최종적으로 근사 Tour를 구하기 위해, 비교적 간단한 발견적 방법을 이용한다. 4개의 병합 가능한 Edge 중 하나를 선택한 근사 Tour의 Cost들을 서로 비교하여 가장 작은 Cost를 가진 근사 Tour를 선택한다. 이를 통해 발생할 수 있는 오차의 가능성을 줄이게 된다. 병합 알고리즘의 결과로 구하게 되는 근사 Tour는 그림 3과 같다.

2.5. 시뮬레이션

시뮬레이션에서 최적 Tour를 구하기 위해 분지한계법을 사용하였으며, 최적 Tour와 분할 및 병합 과정을 거친 근사 Tour를 비교하여 프로그램 수행 횟수와 오차를 구하였다.

표 1에서 최적 Tour의 연산횟수가 두 근사 Tour의 연산횟수를 합한 것에 비해 매우 큰 것을 볼 수 있다. 그리고 근사한 오차범위를 보여준다.

3. 결 론

본 논문에서 제안한 분할 및 병합 알고리즘은 대규모 TSP (Traveling Salesman Problem)의 효율적 해결을 위해 제안하였다. 또한 시뮬레이션을 통해 전체 계산량이 급격하게 감소됨을 보였다. 분할 및 병합 알고리즘은 분지한계법 뿐만이 아니라, 다양한 알고리즘에서도 응용 및 적용될 수 있다.

대규모의 City를 가진 TSP(Traveling Salesman Problem)에서 연산량의 문제는 발견적 방법이나 결정적 방법, 확률적 방법에서 공통적으로 발생한다. 이러한 많은 연산량을 해결하기 위한 방법으로 City들을 두개의 소집합으로 분할하는 것이 좋은 해결책으로 제시될 수 있다.

4. 참고문헌

[1] A. H. Land and A. G. Doig, "An automatic method for solving discrete programming problems", Econometrica, 28, pp. 497--520, (1960).

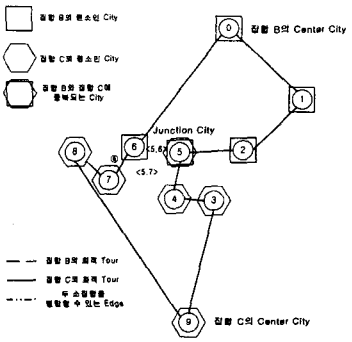
[2] Little, J.D.C., K.G. Murty, D.W. Sweeney and K. Caroline, "An algorithm for the Traveling Salesman Problem," Operations Research, vol. 14 (1963): 972-989.

[3] Anja Hamacher, Christoph Moll. "The Euclidian Traveling Salesman Selection Problem" OR Proceedings of SOR '95

[4] Kobayashi K. "Introducing a clustering technique into recurrent neural networks for solving large-scale traveling salesman problems." In Proceedings of the 8th International Conference on Artificial Neural Networks, Vol.2, pp. 935--940, 1998.

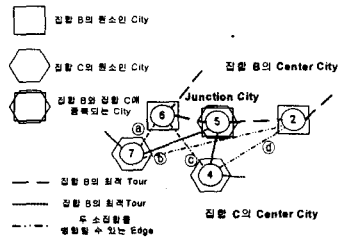
[5] R. Lüling, B. Monien "Load Balancing for Distributed Branch & Bound Algorithms." 6th international parallel processing symposium, p. 543--548, 1992.

[6] Shen Lin, "Computer Solutions of the Traveling Salesman Problem", The Bell System Technical Journal(Dezember 1965), pp2245-2269



(b) 알고리즘 2의 1.1에서 근사 Tour

그림 2. 병합 알고리즘의 개념도



(a) 병합 가능한 Edge들

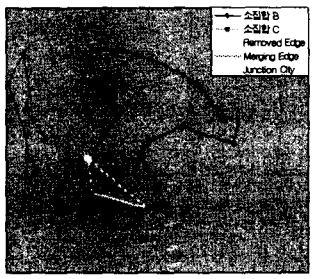


그림 3. 병합된 근사 Tour

최적 Tour 연산 횟수			184409	
B의 근사 Tour 연산횟수			1850	
C의 근사 Tour 연산횟수			475	
City No.	수행 횟수	최적 Tour의 총Cost	근사 Tour의 총Cost	오차(%)
30	1	505	521	3.168
20	1	326	355	8.896

표 1. 그림 4.의 연산 결과

City No.	수행 횟수	최대 오차(%)	최소 오차(%)	평균 오차(%)
30	10	16.016	1.761	8.223

표 2. 오차의 평균