The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 15, No. 4, pp.249-257, Aug. 31, 2015. pISSN 2289-0238, eISSN 2289-0246

http://dx.doi.org/10.7236/JIIBC.2015.15.4.249
JIIBC 2015-4-32

DNN과 k-opt를 적용한 대규모 외판원 문제의 최적 해법

Optimal Solution of a Large-scale Travelling Salesman Problem applying DNN and k-opt

이상운[*]

Sang-Un Lee[*]

요 약 본 논문은 지금까지 해결하지 못한 난제 중 하나인 외판원 문제의 최적 해를 구하는 발견적 알고리즘을 제안한다. 제안된 알고리즘은 초기 경로를 결정하기 위해 기존의 DNN을 변형한 SW-DNN, DW-DNN과 DC-DNN을 제안하였다. 초기 해는 DNN, SW-DNN, DW-DNN과 DC-DNN을 적용하여 최소 경로 길이를 가진 방법을 선택한다. 초기 해에 대해 최적 해를 구하기 위해 먼저 삭제 대상 간선을 선택하는 방법을 결정하였으며, 이들 간선들에 대해 지역 탐색 방법인 k-opt 중에서 2, 2.5, 3-opt를 먼저 적용하고, 삭제 대상 간선들 중 삭제되지 않은 간선들에 대해 4-opt를 적용하였다. 제안된 알고리즘을 대규모의 TSP인 26개의 유럽 도시들을 방문하는 TSP-1과 49개의 미국 도시들을 방문하는 TSP-2에 적용한 결과 모두 최적 해를 구하는데 성공하였다. 제안된 알고리즘은 지금까지 발견적 방법으로는 TSP의 최적 해를 구하지 못한다는 미신을 타파하였고, TSP의 알고리즘으로 적용할 수 있을 것이다.

Abstract This paper introduces a heuristic algorithm to NP-hard travelling salesman problem. The proposed algorithm, in its bid to determine initial path, applies SW-DNN, DW-DNN, and DC-DNN, which are modified forms of the prevalent Double-sided Nearest Neighbor Search and searches the minimum value. As a part of its optimization process on the initial solution, it employs 2, 2.5, 3-opt of a local search k-opt on candidate delete edges and 4-opt on undeleted ones among them. When tested on TSP-1 of 26 European cities and TSP-2 of 49 U.S. cities, the proposed algorithm has successfully obtained optimal results in both, disproving the prevalent disbelief in the attainability of the optimal solution and making itself available as a general algorithm for the travelling salesman problem.

Key Words : Traveling salesman roblem, Exhaustive search method, Edge exchange method, Heuristic method, Double-sided nearest neighbor search

Ⅰ. 서 론

외판원 문제 (traveling salesman problem, TSP)는 임의의 도시에서 출발하여 경로길이 (소요 비용)의 합이 최소가 되도록 n개의 모든 도시를 한 번씩만 방문하고 출발 도시로 다시 돌아오는 문제이다.[1-4]

*정회원, 강릉원주대학교 과학기술대학 멀티미디어공학과
접수일자 : 2015년 2월 8일, 수정완료 : 2015년 6월 28일
게재확정일자 : 2015년 8월 7일

Received: 8 February, 2015 / Revised: 28 June, 2015 / Accepted: 7 August, 2015
*Corresponding Author: sulee@gwnu.ac.kr
Dept. of Multimedia Eng., Gangneung-Wonju National University, Korea

TSP의 최적 해 (optimal solution)는 n개의 도시 (노드)를 연결하는 도로 (간선)에 대해 전수 탐색 (exhaustive or complete search)을 수행하면 최적 해를 구할 수 있다. 전수 탐색은 대칭행렬 (symmetric matrix)인 경우 $(n-1)!/2$회, 비대칭 행렬 (asymmetric matrix)인 경우 $(n-1)!$회 수행해야 하므로 n이 매우 큰 경우 컴퓨터를 활용하여도 빠른 시간에 해답을 얻지 못하는 단점이 있다.[5]

TSP는 계산수학 (computational mathematics) 분야에서 가장 집중적으로 연구하고 있는 문제 중 하나로 전수 탐색 방법 이외에는 효율적인 해법이 아직 알려지지 않은 난제 (NP-hard)로 남아 있다.[4,6,7] 결국, Clay Mathematics Institute에서는 2000년에 7대 난제를 발표하였으며 첫 번째 난제가 $P vs. NP$문제이며, TSP도 $P=NP?$난제중 하나이다.[8,9]

외판원 문제의 최적 해를 도출하지는 못하지만 최적 해에 가까운 해를 빠르게 구하기 위해 일반적으로 발견적 방법 (heuristic algorithms or heuristics)을 적용하고 있다. 발견적 방법에는 탐욕 (greedy) 알고리즘인 가장 인접한 이웃 탐색 (nearest neighbor search, NN), 간선 교환 방법 (k-edge exchange, k-opt) 알고리즘, Tabu 탐색 알고리즘, 모의실험 담금질 (simulated annealing)과 유전자 알고리즘 (genetic algorithm) 등이 있다.[10]

본 논문은 발견적 방법으로도 TSP의 최적 해를 빠르게 구할 수 있는 알고리즘을 제안한다. 제안되는 알고리즘은 변종 양측 최단 인접 이웃 탐색 (variant double-sided nearest neighbor search, VDNN) 방법으로 초기 해 (initial solution)인 실현 가능 경로 (feasible path)를 설정하고, 지역 탐색 (local search) 방법인 k-opt $(k=2,2.5,3,4)$를 적용한다. 이를 위해 VDNN과 실용적인 k-opt를 제시하고, k-opt를 적용할 간선을 결정하는 방법도 함께 제시한다.

2장에서는 외판원 문제의 초기 해를 구하는 발견적 방법을 고찰한다. 3장에서는 초기 해를 구하는 VDNN과 실제 문제에 적합한 3-opt를 제안하고 간선 교환 대상을 결정하는 방법을 제안한다. 4장에서는 실제 데이터에 적용하여 제안된 알고리즘의 적합성을 검증한다.

II. 발견적 TSP 관련 연구와 문제점

TSP의 해법 (solution method)은 정확한 방법 (exact method)과 근사 방법 (approximate method)으로 분류된다. 정확한 방법은 규칙적인 열거 (systematic enumeration)로 결국 최적 해를 찾을 수 있음을 보장한다. 이 방법에는 완전 탐색 알고리즘 (complete search algorithms), 동적 계획법 (dynamic programming, DP), 제약 계획법 (constraint programming, CP)과 선형 계획법 (integer programming, IP)이 있다. 근사 방법은 최적 해를 찾음을 보장하지는 못하지만 적당히 좋은 (reasonable good) 해법을 빠르게 찾기 위해 일반적으로 사용된다. 이 방법에는 선형계획법 완화 (IP Relaxations), 무작위 귀로 (randomized backtracking)와 발견적 방법이 있다.[10]

근사방법을 적용한 최적화 문제에서는 전역 최적 점 (global optimum)과 지역 최적 점 (local optimum)이 발생한다. 전역 최적 점은 모든 다른 해법들 보다 좋은 값이며, 지역 최적 점은 특정 이웃에 있는 모든 해법들 보다 좋은 값을 의미한다.

현실세계의 많은 문제들은 정확한 방법으로 해결하기에는 너무 큰 경향이 있으며, 일반적으로 빠른 결론을 요구한다. 따라서 좋은 발견적 방법을 적용하면 양질의 해법을 찾을 수 있으며, 매우 빠른 장점을 갖고 있다.

발견적 방법은 구성 (constructive), 지역 개선 (local improvement or local search)과 확장 (extensions)으로 분류된다. 구성 알고리즘은 출발선 (scratch)으로부터 여행 경로 (tour)를 구성하는 방법이며, 지역 개선 알고리즘은 현존하는 여행경로를 개선하는 방법이다. 또한 확장은 지역적으로 최적화된 여행경로를 개선하는 방법이다.

구성 알고리즘은 일반적으로 욕심쟁이 방법으로 여행경로를 결정하며 최단 인접 이웃 탐색 (NN), 다중 파편 탐색 (multiple fragment search, MF)과 삽입 탐색 (insertion search) 방법이 가장 많이 적용된다. NN은 임의로 선택된 도시로부터 출발하여 현재 설정된 부분 경로 (partial path)의 끝단 (endpoint)에 방문하지 않은 도시들 중에서 최단 인접한 도시를 추가하면서 경로를 성장시키는 방법이다. 이 방법은 한 방향으로만 성장해 가는 단방향 (single-sided) 방식이다. 반면에 이의 변형으로 최단 인접 이웃 양측 탐색 방법 (double-sided NN, DNN)이 있다.[10-12] 이 방법은 임의의 도시 c_0를 선정하고, 현재의 부분 경로 양 끝단 a,b에 가장 가까운 이웃 도시 a',b'중 보다 가까운 도시를 선택하여 경로를 확장하는 방식으로 비교 대상은 새로 추가될 a',b'이다.[11,12] MF

는 최소신장트리 (minimum spanning tree, MST)의 Kruskal 알고리즘과 유사하게 모든 간선들을 오름차순으로 정렬시키고 최소값 간선부터 경로 파편들을 생성, 성장 또는 결합시키며, 최종적으로는 여러 개의 파편이 하나의 경로를 형성한다.[10]

지역 개선 알고리즘은 실현 가능 경로 (초기 해)에 대해 간선 교환 방법인 k-opt ($k=2,3,4,\cdots$)를 수행한다. 확장 알고리즘에는 모의실험 담금질, Tabu 탐색, 진화 알고리즘 (evolutionary algorithms), 변종 NN (variable neighborhood search, VNN), 신경망 (neural networks), 유전자 알고리즘과 개미집단 최적화 (ant colony optimization) 등 다양한 방법들이 있다.

k-opt는 Stougie[7], Bock[13], Croes[14,15]와 Lin[16], Chuin[17] 등이 제안하고 있으며, 2-opt, 3-opt, 4-opt 등 k를 무한히 확장 가능하나 실제적으로는 2-opt와 3-opt들이 주종을 이루고 있다. 이와 관련하여 Lee[18]는 k-opt를 확장시킨 방법을, Lee[19]는 양방향 탐색법을 적용하였다.

결국, 발견적 방법은 정확한 방법과 같이 최적 해를 얻음을 보장하지 못한다는 신화 (myth)가 타파되지 않고 있다. 본 논문에서는 VDNN과 k-opt를 적용하여 발견적 방법으로도 TSP의 최적 해를 얻음을 보장하여 TSP에서의 신화를 타파하고자 한다.

3장에서는 지역 최적 점에 빠지지 않고 전역 최적점에 도달 가능한 실현 가능 경로를 탐색하는 VDNN 방법을 제안하고, 2, 2.5, 3-opt와 4-opt로 최적 해를 도출하는 알고리즘을 제안한다.

Ⅲ. TSP의 휴리스틱 알고리즘

본 장에서 제안하는 TSP 알고리즘은 다음 가정에 기반하여 발견적 방법이 최적 해를 얻지 못한다는 신화를 타파하고자 한다.

[가정 1] 실현 가능 경로 (초기 해)를 적절히 형성하면 지역 최적 점에 빠지지 않고 최적 해 (전역 최적 점)에 도달할 가능성이 높다.

[가정 2] 초기 해에 대해 2-opt와 실용적인 3-opt 만을 적용하여도 최적 해를 간단히 얻을 수 있다.

1. 실현 가능 경로 탐색 방법

TSP의 최적 해를 구하기 위해서는 먼저 초기 해를 결정해야 한다. 전통적인 TSP의 초기 해는 임의의 노드를 선택하고 (보통은 첫 번째 노드 선택) 방문하지 않은 노드들 중에서 가장 인접한 노드들을 한 번에 하나씩 탐색하는 NN을 적용한다. 이 방법으로 얻은 초기 해를 개선하는 과정에서 거의 대부분은 지역 최적점에 빠져 최적 해를 얻는데 실패할 수 있다.

본 논문에서는 Johnson과 McGeoch가 제안한 DNN[11,12]의 변형 형태 3가지를 제안한다. Johnson과 McGeoch가 제안한 DNN은 먼저, 임의의 노드 (첫 번째 노드 n_0)로부터 시작한다. 이 노드의 첫 번째 가장 인접한 노드 n_r (최소 간선 값, $w\{n_0,n_r\}$을 유출 (outgoing)로 우측에 위치시키고 두 번째 인접한 노드 n_l (간선 값 $w\{n_0,n_l\}$)을 유입 (income)으로 좌측에 위치시킨다. 현재의 부분 경로 (partial path) $n_l-n_0-n_r$의 양 끝단 n_l,n_r에서 가장 가까운 도시 n_{l+1},n_{r+1}중 보다 작은 간선 (가까운 도시)인 $w\{n_l,n_{l+1}\}$또는 $w\{n_r,n_{r+1}\}$을 하나만 선택한다. 즉, 비교 대상은 새로 추가될 간선이며, 현재 부분 경로 양 끝단에 존재하는 간선이 아니다. 또한, 2개 간선을 동시에 선택하는 것이 아니라 한 번에 하나의 간선만을 선택한다.

본 논문에서는 Johnson과 McGeoch가 제안한 DNN의 변종으로 VDNN (Variant DNN) 3가지를 제안한다.

첫 번째는 현재의 부분 경로 양 끝단 간선 $w\{n_0,n_r\}$과 $w\{n_0,n_l\}$을 비교하여 작은 간선 값 노드에서 최소 간선 값을 선택하여 경로를 확장한다. 이 과정을 미 방문 노드가 1개 일 때 까지 수행한다. 미 방문 노드가 존재하지 않으면 유출 끝단 노드와 유입 끝단 노드를 연결시킨다. 즉, 제안 방법은 비교 대상이 새로 추가될 간선 $w\{n_l,n_{l+1}\}$과 $w\{n_r,n_{r+1}\}$이 아니라 현재의 부분 경로 양 끝단에 존재하는 간선 $w\{n_{l-1},n_l\}$과 $w\{n_{r-1},n_r\}$로 Johnson과 McGeoch의 DNN과 차이가 있다. 이 방법을 SW-DNN (single win-DNN)이라 하자.

두 번째는 부분 경로의 양 끝단 n_l,n_r에서 최단 간선 $w\{n_l,n_{l+1}\}$과 $w\{n_r,n_{r+1}\}$을 동시에 추가하는 방법이다. $n_{l+1}\neq n_{r+1}$인 경우는 해당 끝단에 동시에 연결시킨다. 만약 $n_{l+1}=n_{r+1}$로 동일한 도시를 탐색할 경우, 현재의 부분 경로 양 끝단 노드의 간선 값 $w\{n_{l-1},n_l\}$과 $w\{n_{r-1},n_r\}$중 최소 값에 연결하는 방법과 새로 추가될

간선 들 $w\{n_l,n_{l+1}\}$과 $w\{n_r,n_{r+1}\}$중 최소값을 연결하는 방법이 있을 수 있다. 현재의 부분 경로 양 끝단 노드의 간선 값 $w\{n_{l-1},n_l\}$과 $w\{n_{r-1},n_r\}$중 최소값에 연결하는 방법을 DW-DNN (dual win-DNN), 새로 추가될 간선 들 $w\{n_l,n_{l+1}\}$과 $w\{n_r,n_{r+1}\}$중 최소값을 연결하는 방법을 DC-DNN (dual competitive- DNN)이라 하자. 경쟁에서 패한 노드는 다음 최단 간선을 선택하여 연결시킨다.

TSP의 초기 해 (실현 가능한 경로)는 SW-DNN, DW-DNN과 DC-DNN으로 탐색한 결과 경로 길이 합이 최소값을 가진 경로를 선택한다. 결국, 제안된 알고리즘의 첫 번째 단계는 다음과 같이 수행된다.

Step 1. DNN, SW-DNN, DW-DNN과 DC-DNN으로 TSP 경로를 탐색한 결과 경로 길이 합이 최소값을 가진 경로를 초기 해로 선택한다.

TSP의 초기 해에 대해 지역탐색 방법인 k-opt를 적용하여 최적 해를 구한다. k-opt를 적용하는 방법은 다음 절에서 논한다.

2. k-opt 지역 탐색 방법

Lin[16]은 k-opt ($k=4,5,\cdots$)는 계산시간에 비해 실질적으로 비용절감 효과가 크지 않다는 결과를 얻었다. 그러나 본 절에서는 4-opt도 논하며, Stougie[7]와 Chuin[17]가 제시한 2-opt, 2.5-opt와 3-opt를 고려한다. 참고로, k-opt에서 삭제되는 간선들을 제거 간선 (out- edges, e_{out}), 새로 교체되는 간선들을 추가 간선 (in- edges, e_{in})이라 한다.[20]

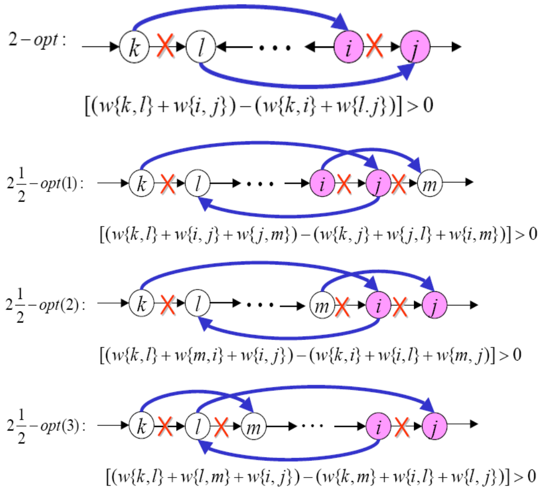
초기 해에 대해 k-opt의 간선 교환 방법을 적용하여 최적 해를 얻기 위해서는 2 가지 문제점이 제기된다.

첫 번째로, 초기 해로 얻는 n개의 간선들 중에서 e_{out}이 되는 $w\{i,j\}$를 선택하는 문제가 제기된다. 가장 일반적으로 적용되는 방법은 초기 해로 얻은 n개의 모든 간선들을 대상으로 한다. 그러나 이 방법은 불필요하게 수행 시간만 많이 소요되는 단점이 있다. 본 논문에서는 간선 제거 방식을 채택하고 있기 때문에 n개의 간선들 중에서 값이 큰 쪽이 대상이 될 수 있다. 이들 교환 대상 간선을 선택하기 위해 교환 대상 간선을 e_{out}, 최소 간선을 $e_{\min}$이라 한다.

$e_{\min}$은 다음과 같이 얻는다. 행 (유출)에서 첫 번째 노드부터 마지막 노드까지 순서대로 최소값 $w\{i,j\}$를 선택하고 $w\{j,i\}$를 삭제한다. 이는 각 노드의 최소 인접 유출 간선이 된다. 다음으로 열 (유입)에서 첫 번째 노드부터 마지막 노드까지 순서대로 최소값 $w\{i,j\}$를 선택하면서 $w\{j,i\}$를 삭제한다. 이들 선택된 간선들은 각 노드의 최소 유출과 유입 간선들로 삭제 대상인 $w\{i,j\}$가 될 가능성이 매우 낮다. 초기 해로 얻은 경로에 대해 간선 교환 대상 e_{out}은 $e_{\min}$에 속하지 않는 간선들이 된다. 또한, k-opt 수행으로 새로 연결된 간선들 e_{in}에 대해 $e_{\min}$에 속하지 않으면 e_{out}에 추가한다. 간선 $w\{i,j\}$의 교환은 e_{out}의 내림차순으로 수행한다. 간선 교환 은 k-opt들 중에서 비용을 가장 크게 단축시키는 방법을 선택하여 경로를 변경시킨다. 제안된 알고리즘의 두 번째 단계는 다음과 같이 수행한다.

Step 2. 초기 해 간선들 중에서 $e_{\min}$에 속하지 않는 간선들을 간선 삭제 대상 e_{out}으로 결정한다.

두 번째로, 어떤 k-opt ($k=2,3,4,\cdots$)를 적용할 것인가가 문제로 제기된다. 2-opt는 한 가지 방법만 존재하지만, 3-opt는 고려해야 할 경우가 많이 존재한다. 3-opt의 경우, 2개의 간선이 인접한 경우를 2.5-opt (vertex-reinsertion)라 하며, 3개 간선이 모두 인접하지 않은 경우를 일반적인 3-opt라 한다. 본 논문에서는 그림 1과 같이 2-opt, 실현 가능성이 가장 큰 2.5-opt 3개, 3-opt 3개와 4-opt 2개만을 적용하며, 3개 간선이 모두 인접하지 않는 그림 2의 일반적인 3-opt는 적용하지 않는다.



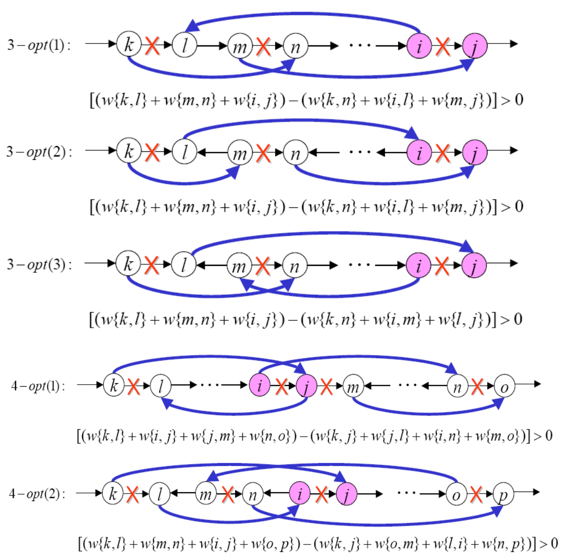


그림 1. k-opt
Fig. 1. k-opt

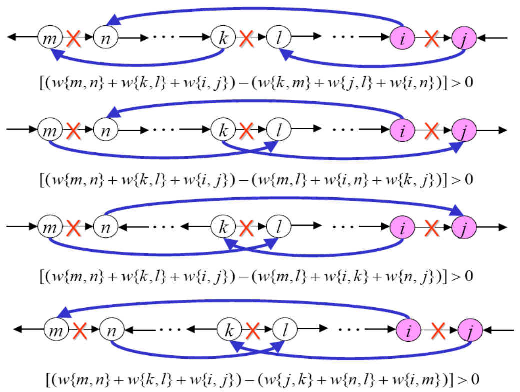


그림 2. 일반적인 3-opt
Fig. 2. General 3-opt

제안된 알고리즘의 세 번째와 네 번째 단계는 다음과 같이 수행된다.

Step 3. e_{out}에 대해 내림차순으로 2-opt, 2.5-opt, 3-opt를 수행하여 해를 개선한다. 이 과정에서 새로 추가된 간선 e_{in}들 중에서 $e_{\min}$에 속하지 않으면 e_{out}에 추가한다. 만약 삭제된 간선들 중에서 e_{out}에 존재하면 e_{out}에서 삭제한다. e_{out}에서 k-opt로 해 개선을 수행하지 못한 간선들은 e_{rem}에 추가한다.

Step 4. 만약, e_{rem}에 간선들이 존재하면 4-opt로 해 개선을 수행한다.

본 알고리즘은 k-opt를 적용하기 때문에 대칭행렬에만 적용된다. 왜냐하면 대칭행렬은 무방향 그래프(undirected graph)로 간선 교환에 따라 경로가 반대로 변경되어도 간선 $w\{i,j\} \neq w\{j,i\}$로 동일한 경로 길이를 얻을 수 있다. 반면에, 비대칭행렬은 방향 그래프(Digraph)로 간선 교환으로 경로가 반대로 변경되면 $w\{i,j\} \neq w\{j,i\}$속성을 갖고 있어 경로길이가 증가할 수 있기 때문이다. 결국, k-opt는 대칭행렬에만 한정되어 적용될 수 있다.

Ⅳ. 알고리즘 적용 및 분석

본 장에서 실험에 적용된 데이터는 Pleines[5]와 Dantzig, Fulkerson과 Johnson[21]이 제시한 데이터이다. Pleines[5]가 제시한 TSP-1 데이터는 유럽의 26개 도시를 여행하는 경우로 편의상 각 도시를 번호로 표기하였다. 참고로 1 (포르투갈 리스본), 2(핀란드 헬싱키), 3(스페인 마드리드), 4(터키 이스탄불), 5(그리스 아테네), 6(루마니아 부다페스트), 7(불가리아 소피아), 8(스웨덴 스톡홀름), 9(노르웨이 오슬로), 10(세르비아 베오그라드), 11(헝가리 부다페스트), 12(덴마크 코펜하겐), 13(이탈리아 로마), 14(폴란드 바르샤바), 15(오스트리아 빈), 16(독일 베를린), 17(네덜란드 암스테르담), 18(영국 런던), 19(벨기에 브뤼셀), 20(체코 프라하), 21(이탈리아 밀라노), 22(스위스 취리히), 23(스페인 바르셀로나), 24(스위스 제네바), 25(독일 프랑크푸르트), 26(프랑스 파리)이다.

TSP-1에 대해 전수조사로 $n(n-1)!/2=7.7556\times10^{26}$회 수행하면 최적 해를 얻을 수 있다. 각 도시간의 거리 단위는 Km이다.

Dantzig, Fulkerson과 Johnson[21]이 제시한 TSP-2 데이터는 미국의 49개 도시를 방문하는 문제로, 7개 도시는 40과 41번 도시 사이에 존재하여 실제 데이터는 42개이다. 본 데이터의 단위는 마일이며, $1/17(d_{ij}-11)$로 정규화시켜 소숫점 아래를 절삭하여 정수로 표현되었다.

TSP-1의 최적 해는 1-3-13-21-22-25-20-15-11-10-7-5-4-6-14-2-8-9-12-16-17-19-1-26-24-23-1=16,189 Km이며, TSP-2의 최적 해는 1번부터 42번까지 순서대로 방문하고 다시 1번으로 되돌아오는 경우로 12,345M=699이다.

TSP-1과 TSP-2 데이터에 대해 제안된 알고리즘의 적합성을 검증하기 위해 DNN, SW-DNN, DW-DNN과

DC-DNN을 수행하여 초기 해를 구한 결과는 그림 3과 같다. 모든 방법은 첫 번째 노드인 ① 노드에서 시작하였으며, 참고로 TSP-1에 대해 NN과 MF도 수행하였다.

NN: 1-3-23-24-22-21-13-15-11-10-7-6-4-5-14-16-20-25-19-17-26-18-12-9-8-2-1=19,762 Km
MF: 1-3-23-13-21-22-24-25-17-19-26-18-14-2-8-9-12-16-20-15-11-10-7-6-4-5-1=18,949 Km
DNN: 1-3-5-4-7-6-14-18-26-17-19-25-9-2-8-12-16-20-10-11-15-13-21-22-24-23-1=20,433 Km
SW-DNN: 1-3-13-21-22-24-26-19-17-25-20-15-11-10-7-6-4-5-14-2-8-9-12-16-18-23-1=17,475 Km
DW-DNN = DC-DNN: 1-3-13-21-22-24-20-15-11-10-7-6-4-5-2-14-8-9-12-16-18-26-17-19-25-23-1=18,523 Km

(a) TSP-1

DNN: 1-41-2-3-4-5-6-7-8-9-10-25-24-27-26-11-12-13-16-18-15-14-19-20-17-23-22-21-36-33-32-29-28-30-31-34-35-37-38-39-40-42-1=859
SW-DNN: 1-41-2-39-11-12-23-22-17-16-18-19-20-13-14-15-21-36-3-4-5-6-7-8-9-10-24-25-26-27-28-29-33-32-30-31-34-35-37-38-40-42-1=1,004
DW-DNN: 1-41-2-39-5-4-3-8-9-7-6-26-27-24-25-10-11-13-14-15-36-20-19-18-16-17-23-22-21-28-29-33-32-30-31-34-35-37-38-40-42-1=967
DC-DNN: 1-41-2-3-4-5-6-7-8-9-10-25-24-27-26-11-12-13-14-15-20-36-19-18-16-17-23-22-28-29-33-32-30-31-34-35-37-38-39-40-42-1=989

(b) TSP-2

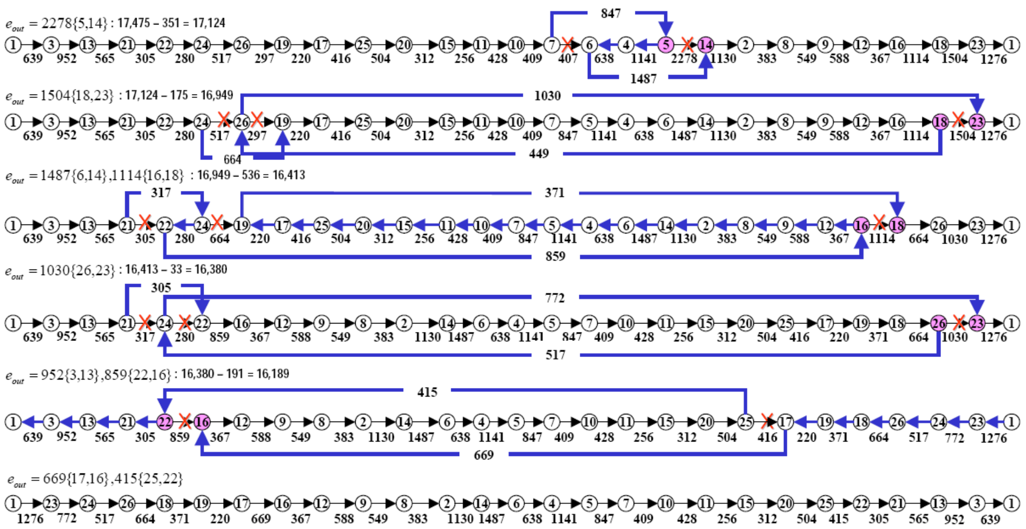
그림 3. 초기 해
Fig. 3. Initial solution

그림 3에서 최소값을 가진 경로를 초기 해로 선택하면 TSP-1은 SW-DNN, TSP-2는 DNN이 된다.

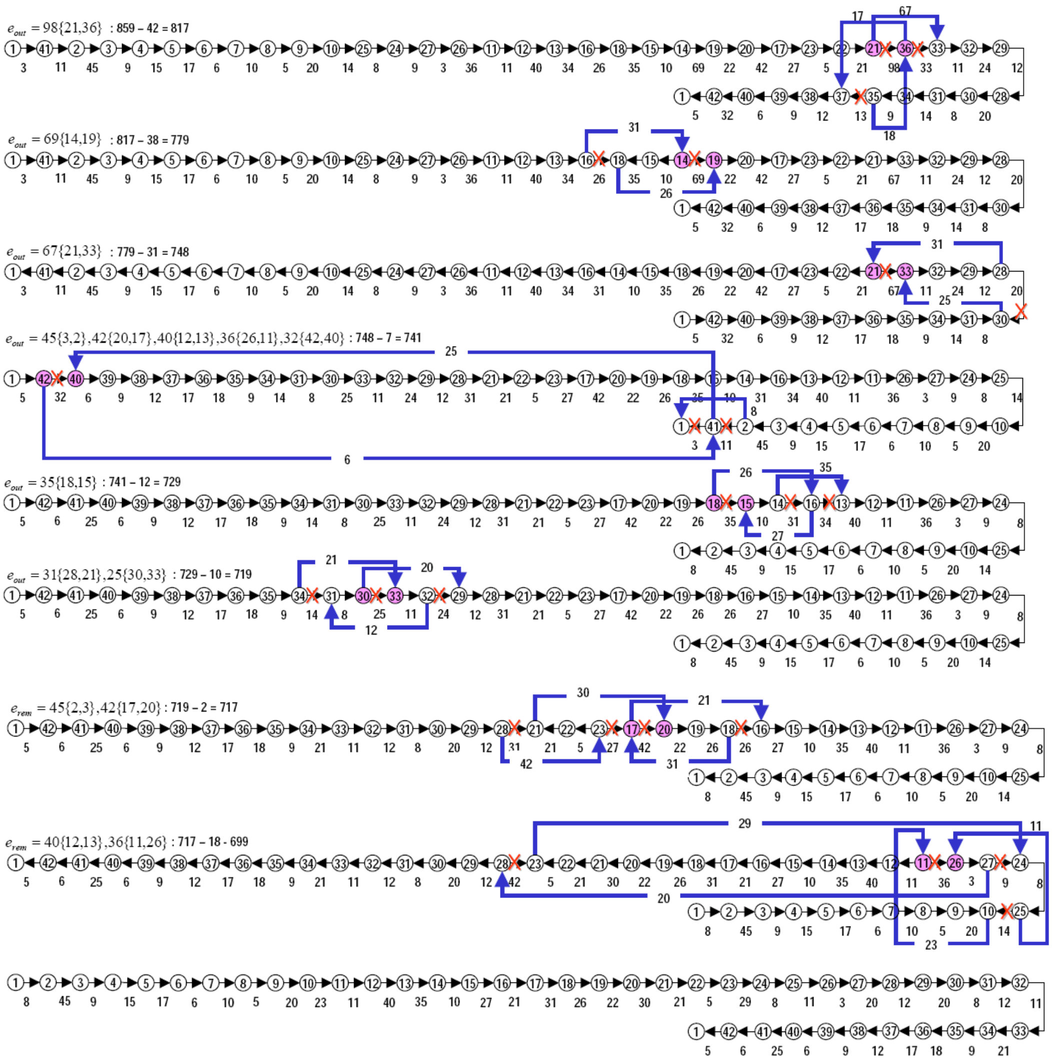
TSP-1의 $e_{\min}$을 구한 결과는 그림 4와 같다. 편의상 순서대로 값만 표기하면 $e_{\min}$={639,383,637,577,847,407,409,549,588,428,256,367,565,589,312,337,220,371,297,504,305,280,772,317,449,565,589,312,337,220,371,297,504,305,280,772,317,449,1276,907,638,1141,732,871,612,416}이다. TSP-2는 $e_{\min}$={3,8,9,15,17,6,10,5,20,14,11,26,29,10,27,21,31,27,26,22,30,21,5,27,8,29,11,3,9,12,20,8,12,11,21,9,13,17,12,9,6,1,5,11,6,5}이다.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1		4074	639	4326	4528	4073	3730	3691	3641	3354	3314	3504	2671	3438	3078	3148	2318	2260	2098	2926	2294	2299		2019	2542	1801
2	4074		3532	3128	2286	2665	2562	383		2753	1884	1035	3148	1130	1855	1459	1803	2337	1962	1800	2571	2260	3725	2422	1843	2263
3		3532		3687	2889	3475	1986	3466	3094	2715	2675	2846	952	2967	2605	2509	1781	1718	1561	2347	1728	1641	637	1380	1903	1709
4	4326	3128	3687		1141		577	3356	3306	972	1400	2689	2306	2125	1656	2304	2795	3141	2276	1968	2046	2219	3075	2363	2361	2913
5	4528	2286	3889			1254	847	3549	3499	1174	1602	2798	2508	2278	1858	2506	3027	3296	2978	2170	2248	2421	3277	2565	2563	3115
6	4073	2665	3475	638	1254		407	2825	2775		895	2102	2050	1487	1106	1749	2370	2666	2295	1408	1786	1850	2859	2116	1950	2357
7	3730	2562	1986					2779	2729	409	810	1981	1727	1508	1060	1703	2230	2564	2193	1362	1463	1769	2536	1793	1810	2335
8	3691		3466	3356	3549	2825	2779		549	2370	1969	626	2764	1665	1719	1076	1420	1954	1579	1417	2187	1877	2839	2039	1460	1880
9	3641	907	3094	3306	3499	2775	2729			2320	1919	588	2715	1615	1669	1026	1370	1904	1529	1367	2138	1827	2789	1989	1410	1830
10	3354	2753	2715	972	1174	732		2370	2320		428	1637	1334	1099	684	1332	1823	2117	1804	966	1074	1247	2103	1391	1389	1941
11	3314	1884	2675	1400	1602	895	810	1969	1919			1345	1294	698	256	904	1395	1752	1358	568	1034	1004	2093	1320	979	1527
12	3504	1035	2846	2689	2798	2102	1981	626		1637	1345		2083	1129	915	367	886	1238	1030	688	1507	1205	2234	1533	802	1252
13	2671	3148	952	2306	2508	2050	1727	2764	2715	1334	1294	2083		1872	1150	1525	1661	1898	1492	1300	565		1420	882	1275	1413
14	3438	1130	2967	2125	2278	1487	1508	1665	1615	1099	698	1129	1872		718	589	1216	1670	1299		1456	1298	2529	1575	1064	1627
15	3078	1855	2605	1656	1858	1106	1060	1719	1669	684		915	1150	718		648	1139	1582	1107	312	837	748	1928	1018	723	1271
16	3148	1459	2509	2304	2506	1749	1703	1076	1026	1332	904		1525		648		669	1114	781	337	1055	859	1847	1129	552	1073
17	2318	1803	1781	2795	3027	2370	2230	1420	1370	1823	1395	886	1661	1216	1139	669		530	220	912	1096	801	1558	964		517
18	2260	2337	1718	3141	3296	2666	2564	1954	1904	2117	1752	1238	1898	1670	1582	1114	530		371	1257	1321	942	1504	991	764	
19	2098	1962	1561	2276	2978	2295	2193	1579	1529	1804	1358	1030	1492	1299	1107	781				913	927	632	1338	664		297
20	2926	1800	2347	1968	2170	1408	1362	1417	1367	966	568	688	1300	612			912	1257	913		830	668	1480	950	504	1071
21	2294	2571	1728	2046	2248	1786	1463	2187	2138	1074	1034	1507		1456	837	1055	1096	1321	927	830		305	1043		710	842
22	2299	2260	1641	2219	2421	1850	1769	1877	1827	1247	1004	1205	871	1298	748	859	801	942	632	668			1052	280	415	572
23	1276	3725		3075	3277	2859	2536	2839	2789	2103	2093	2234	1420	2529	1928	1847	1558	1504	1338	1480	1043	1052		772	1295	1030
24	2019	2422	1380	2363	2565	2116	1793	2039	1989	1391	1320	1533	882	1575	1018	1129	964	991	664	950	317				578	517
25	2542	1843	1903	2361	2563	1950	1810	1460	1410	1389	979	802	1275	1064	723	552	416	764	409		710	415	1295	578		567
26	1801	2263	1709	2913	3115	2357	2335	1880	1830	1941	1527	1252	1413	1627	1271	1073	517	449		1071	842	572	1030	517	567	

그림 4. TSP-1의 유출과 유입 최소 간선 $(e_{\min})$
Fig. 4. Out and In minimum edges $(e_{\min})$ for TSP-1



(a) Optimal solution of SW-DNN k-opt for TSP-1



(b) Optimal solution of DNN k-opt for TSP-2

그림 5. 실험 데이터의 최적 해
Fig. 5. Optimal solution for experimental data

그림 3에서 최소값을 가진 TSP-1의 SW-DNN와 TSP-2의 DNN에 대해 $e_{\min}$에 포함되지 않는 간선들인 e_{out}의 내림차순으로 2-opt, 2.5-opt, 3-opt를 수행하고, e_{rem}에 대해 4-opt를 적용한 과정은 그림 5에 제시되어 있다.

TSP-1은 Step-3의 e_{out}에 대한 2, 2.5, 3-opt 수행으로 최적 해를 얻었으며, TSP-2는 Step-4의 e_{rem}에 대해 4-opt 수행으로 최적 해를 얻는데 성공하였다.

TSP-1은 e_{out} =2278{5,14}, 1504{18,23}, 1141{4,5}, 1130 {14,2}, 1114{16,18}, 952{3,13}, 517{24,26}이며, e_{in} 중 e_{out}에 추가된 간선은 1487{6,14}, 1030{26,23}, 859{16,22}, 371 {18,19}, 517{26,24}, 415{25,22}이다.

TSP-2는 e_{out} =98{21,36}, 69{14,19}, 45{2,3}, 42{20, 17}, 40{12,13}, 36{26,11}, 36{15,18}, 33{36,33}, 32{40, 42}, 24{32,29}, 20{28,30}이며, e_{in} 중 e_{out}에 추가된 간선은 67{21,33}, 31{28,21}, 25{30,33}, 11{2,41}, e_{rem}은 45{2,3}, 42{20,17}, 40{12,13}, 36{26,11}이다.

참고로 TSP-1에 대한 NN, MF, DNN, DW-DNN, DC-DNN의 k-opt를 수행한 결과는 그림 6에 제시되어 있다.

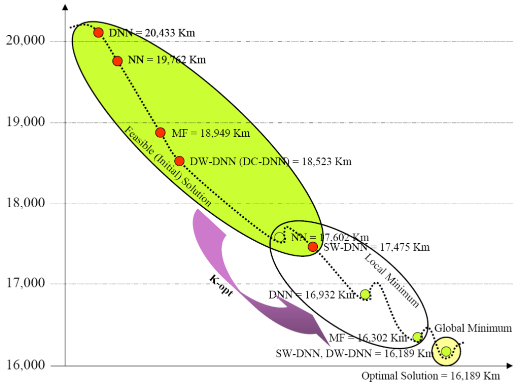


그림 6. TSP-1 초기 해의 k-opt 적용 결과
Fig. 6. Result of k-opt for initial solution of TSP-1

DNN, NN과 MF는 좋은 결과를 얻지 못하였으며, DW-DNN (DC-DNN)은 SW-DNN과 동일하게 최적 해를 얻는데 성공하였다. TSP-2에 대해서는 SW-DNN, DW-DNN과 DC-DNN 전역 최적점인 최적 해를 구하는데 실패하였으며, 단지 DNN만이 최적 해를 구하는데 성공하였다.

V. 결론

본 논문은 대규모의 외판원 문제에 대해 초기 해를 구하고 최적 해를 도출할 수 있는 발견적 알고리즘을 제안하였다. 이를 위해 DNN의 변형인 SW-DNN, DW-DNN과 DC-DNN 방법을 제시하고 실용적인 k-opt를 적절히 적용하면 빠르게 최적 해를 구할 수 있음을 보였다. 이를 위해 기존 DNN의 변형인 M-DNN과 D-DNN 방법을 제시하였으며, 2.5-opt를 포함한 특별한 형태의 3-opt 6가지를 적용하였다. 또한, k-opt를 적용하는 간선을 결정하고 적용하는 방법을 제안하였다.

제안된 M-DNN과 D-DNN은 시작 노드를 임의로 선택하여도 제안된 k-opt만 적용하면 모두 최적 해를 구할 수 있었다. 결국, 본 논문은 발견적 알고리즘으로는 TSP의 최적 해를 구하지 못한다는 미신을 타파할 수 있었으며, TSP의 최적 해를 구하는 알고리즘을 제시하였다는데 큰 의미가 있었다.

References

[1] Wikipedia, "Travelling Salesman Problem," http://en.wikipedia.org/wiki/Travelling_Salesman_Problem, Wikimedia Foundation Inc., 2014.

[2] A. Likas and V. T. Paschos, "A Note on a New Greedy-Solution Representation and a New Greedy Parallelizable Heuristic for the Traveling Salesman Problem," Chaos, Solitons and Fractals, Vol. 13, pp. 71-78, 2002, doi: 10.1016/S0960-0779(00)00227-7.

[3] A. Schrijver, "On the History of Combinatorial Optimization (till 1960)," in Handbook of Discrete Optimization'' (K. Aardal, G.L. Nemhauser, R. Weismantel, eds.), Elsevier, Amsterdam, pp. 1-68, http://homepages.cwi.nl/~lex/files/histco.pdf, 2005.

[4] J. Denzinger, D. Fuchs, M. Fuchs, and M. Kronenburg, "The Teamwork Method for Knowledge-Based Distributed Search: The travelling salesman problem," University of Kaiserslautern, 2008.

[5] J. Pleines, "ZIP-Methode: ein Kombinatorischer Ansatz zur Optimalen Lősung Allgemeiner Traveling -Salesman-Problem (TSP)," Kőnnen bekannte Lősungen nicht nur auf Gesamtgrphen sondern auf Teilgraphen angewandt werden, so bringt die ZIP-Methode den entscheidenden Quantensprung der rechentechnischen Vereinfachung, 2006.

[6] S. Vempala, "18.433 Combinatorial Optimization: NP-completeness," http://ocw.mit.edu/NR/rdonlyres/Mathematics/18-433Fall2003/778D00DB-F21C-486C-ABD8-F5E7F5C929C3/0/l20.pdf, 2003.

[7] L. Stougie, "2P350: Optimaliseringsmethoden," College Wordt ggeven op vinjdagmiddag, 2001.

[8] W. Cook, "The Traveling Salesman Problem," The School of Industrial and Systems Engineering, Gatorgia Tech, 2008.

[9] A. Battese, "Millennium Problems," Clay Mathematics Institute, 2014.

[10] E. Charniak and M. Herlihy, "CSC 751

Computational Complexity: Local Search Heuristics," Department of Computer Science, Brown University, 2008.

[11] D. S. Johnson and L. A. McGeoch, "The Traveling Salesman Problem: A Case Study in Local Optimization," Department of Mathematics and Computer Science, Amherst College, 1995.

[12] D. S. Johnson and L. A. McGeoch, "The Traveling Salesman Problem and Its Variations," Kluwer Academic Publishers, pp. 369-443, 2002.

[13] F. Bock, "An Algorithm for Solving Traveling Salesman and Related Network Optimization Problems," The 14th ORSA National Meeting, 1958.

[14] G. A. Croes, "The Traveling-Salesman Problem," Operations Research, Vol. 4, pp. 61-75, 1956.

[15] G. A. Croes, "A Method for Solving Traveling Salesman Problems," Operations Research, Vol. 6, pp. 791 - 812, 1958, doi: 10.1287/opre.6.6.791.

[16] S. Lin, "Computer Solutions of the Traveling Salesman Problem," Bell System Technical Journal, Vol. 44, pp. 2245-2269, 1965, doi: 10.1002/j.1538-7305.1965.tb04146.x.

[17] L. H. Chuin, "IS 703: Decision Support and Optimization," School of Information Systems," Department of Computer Science, Brown University, 2008.

[18] S. U. Lee, "The Extended k-opt Algorithm for Traveling Salesman Problem," Journal of KSCI, Vol. 17, No. 10, pp. 155-165, Oct. 2012, doi: 10.9708/jksci/2012.17.10.155.

[19] S. U. Lee, "A Polynomial Time Algorithm of a Traveling Salesman Problem," Journal of KSCI, Vol. 18, No. 12, pp. 75-82, Dec. 2013, doi: 10.9708/jksci.2013.18.12.075.

[20] K. Helsgaun, "An Effective Implementation of K-opt Moves for the Lin-Kernighan TSP Heuristics," Computer Science, Roskilde University, Denmark, 2007.

[21] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a Large-scale Traveling-Salesman Problem," The Rand Corporation, http://www.cse.wustl.edu/~chen/7102/TSP.pdf, 1954.

저자 소개

이 상 운(정회원)

- 1987년 : 한국항공대학교 항공전자공학과 (학사)
- 1997년 : 경상대학교 컴퓨터과학과 (석사)
- 2001년 : 경상대학교 컴퓨터과학과 (박사)
- 2003년 : 강원도립대학 컴퓨터응용과 전임강사
- 2004년 ~ 2007년 2월 : 국립 원주대학 여성교양과 조교수
- 2007년 3월 ~ 2015년 3월 : 강릉원주대학교 멀티미디어공학과 부교수
- 2015년 4월 ~ 현재 : 강릉원주대학교 멀티미디어공학과 정교수

<관심분야> : 소프트웨어 프로젝트 관리, 개발 방법론, 분석과 설계 방법론, 시험 및 품질보증, 소프트웨어 신뢰성, 그래프 알고리즘

- e-mail : sulee@gwnu.ac.kr

※ 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 방송통신정책연구센터(CPRC) 지원사업(IITP-2015-R0880-15-1007)의 연구결과로 수행되었음.