

외판원 문제의 확장된 k-opt 알고리즘

이 상 운*

The Extended k-opt Algorithm for Traveling Salesman Problem

Sang-Un, Lee *

요 약

본 논문은 지금까지 해결하지 못한 NP-Hard 문제들 중의 하나인 외판원 문제를 해결할 수 있는 알고리즘을 제안한다. 제안된 알고리즘은 간선교환 방법을 적용한 발견적 알고리즘이다. 초기해를 구하는 전형적인 방법은 첫 번째 노드부터 가장 인접한 노드를 방문하여 외판원의 경로를 결정하는 방법이다. 본 논문에서는 각 노드의 최소 간선을 선택하여 선택된 간선들 중 최소값을 가진 노드부터 출발하는 Min-Min 방법과 최대값을 가진 노드부터 출발하는 Min-Max 방법을 적용하고 두 방법 중 최소 경로길이를 가진 방법을 초기해로 결정하였다. 초기해로부터 최적해를 구하는 과정은 기존의 2-간선 교환 방법 (2-opt)을 기본적으로 적용하고, 추가로 확장된 3-opt와 4-opt를 제안하였다. 이와 같은 방법을 7개의 실제 데이터들에 적용한 결과 지금까지 알려진 최적해를 빠르고 정확히 구하는데 성공하였다.

▶ Keywords : 외판원 문제, 전수조사 방법, 간선 교환법, 발견적 방법, 최소-최소 탐색과 최소-최대 탐색

Abstract

This paper suggests traveling salesman problem algorithm that have been unsolved problem with NP-Hard. The proposed algorithm is a heuristic with edge-swap method. The classical method finds the initial solution starts with first node and visits to mostly adjacent nodes then decides the traveling path. This paper selects minimum weight edge for each nodes, then perform Min-Min method that start from minimum weight edge among the selected edges and Min-Max method that starts from maximum weight edges among it. Then we decide tie initial solution to minimum path length between Min-Min and Min-Max method. To get the final optimal solution, we apply

• 제1저자 : 이상운*

• 투고일 : 2012. 06. 19. 심사일 : 2012. 07. 22. 게재확정일 : 2012. 09. 10.

* 강릉원주대학교 멀티미디어공학과 (Dept. of Multimedia Eng., Gangneung-Wonju National University)

previous two-opt to initial solution. Also, we suggest extended 3-opt and 4-opt additionally. For the 7 actual experimental data, this algorithm can be get the optimal solutions of state-of-the-art with fast and correct.

▶ Keywords : Traveling Salesman Problem, Exhaustive Search Method, Edge Exchange Method, Heuristic Method, Min-Min and Min-Max Search

I. 서 론

외판원 문제 (Traveling Salesman Problem, TSP)는 n 개의 도시가 주어졌을 때, 임의의 도시에서 출발하여 경로를 이 (소요 비용)의 합이 최소가 되도록 모든 도시를 한 번씩만 방문하고 출발 도시로 다시 돌아오는 문제이다[1-4].

외판원 문제를 해결하는 가장 쉬운 방법은 n 개의 도시를 연결하는 도로 (간선)에 대해 전수조사 (exhaustive search)를 대칭행렬 (symmetric matrix)인 경우 $\frac{(n-1)!}{2}$ 회, 비대칭 행렬 (asymmetric matrix)인 경우 $(n-1)!$ 회 수행하여 최적해를 구한다. 결국 이 방법은 컴퓨터를 활용하여도 원하는 시간에 해답을 얻지 못하는 단점이 있다[5]. 예를 들면, $n=10$ 인 비대칭 행렬인 경우 $9! = 362,880$, $n=26$ 인 경우 $25! = 1.551 \times 10^{25}$ 으로 기하급수적으로 증가하여 아직까지 해결하지 못한 NP-Hard 문제로 알려져 있다[6]. 외판원 문제의 최적해를 도출하지는 못하지만 최적해에 가까운 해를 빠르게 구하기 위해 발견적 방법인 욕심쟁이 (greedy) 알고리즘, 간선 교환 방법 (2-opt, 3-opt, k-opt) 알고리즘, Tabu 탐색 알고리즘, 모조의 담금질 (simulated annealing)과 유전자 알고리즘 (genetic algorithm)들이 적용되고 있다[7].

외판원 문제는 계산수학 (computational mathematics) 분야에서 가장 집중적으로 연구하고 있는 문제 중의 하나로 일반적인 경우에 대한 효율적인 해법이 아직 알려지지 않고 있다[4,8]. 결국, Clay Mathematics Institute에서는 2000년에 이 문제 해결에 100만불의 상금을 제시하고 있다[9,10].

본 논문은 발견적 방법 중 하나인 k-opt인 간선 교환 방법 (edge exchange method)을 적용한 알고리즘을 제안하여 외판원 문제의 최적해를 빠르게 구할 수 있음을 증명한다. 이를 위해 기본적으로는 기존의 2-OPT를 적용하고, 보다 실용적인 3-OPT와 4-OPT를 제안한다.

2장에서는 외판원 문제의 초기해를 구하는 방법과 간선 교환 방법의 문제점을 고찰해 본다. 3장에서는 실제 문제에 적합한 3-OPT와 4-OPT를 추가적으로 제안하고 간선 교환 방

법을 적용하여 외판원 문제의 최적해를 도출할 수 있는 발견적 알고리즘을 제안한다. 4장에서는 실제 데이터들에 적용하여 제안된 알고리즘의 적합성을 검증한다.

II. 관련연구와 연구 배경

그림 1은 Kratica와 Radojevi[11]에서 인용된 TSP이다. Kratica와 Radojevi[11]는 전형적인 임의선택 방법 (classical arbitrary chosen method)으로 첫 번째 노드 (노드 1)에서 출발하여 일명 욕심쟁이 알고리즘인 “가장 인접한 이웃 탐색 (Nearest Neighbor Search)”을 수행하여 $1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 4 \rightarrow 6 \rightarrow 1 = 58$ 을 얻었다. 이를 “1st Node Method”라 하자.

또한, 최적해는 $1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 4 \rightarrow 5 \rightarrow 1 = 56$ 이라고 제시하고 있다. 그러나 1st Node Method로부터 최적해를 구하는 방법은 제시하지 못하였다.

	1	2	3	4	5	6
1		3	10	11	7	25
2	3		6	12	8	26
3	10	6		9	4	20
4	11	12	9		5	15
5	7	8	4	5		18
6	25	26	20	15	18	

그림 1. TSP-1
Fig. 1. Traveling Salesman Problem-1

참고로 모든 노드에서 시작하여 탐색한 결과 최적해 $z_{opt} = 56$ 을 얻지 못한다. 그러나 이 방법은 특정 문제에 대해서는 최적해를 얻을 수도 있다.

노드 1 : $1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 4 \rightarrow 6 \rightarrow 1 = 58$

노드 2 : $1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 4 \rightarrow 6 \rightarrow 1 = 64$

노드 3 : $1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 4 \rightarrow 6 \rightarrow 1 = 69$

$$\text{노드 4 : } 1 \xrightarrow{3} 2 \xrightarrow{6} 3 \xrightarrow{4} 5 \xrightarrow{5} 4 \xrightarrow{15} 6 \xrightarrow{25} 1 = 58$$

$$\text{노드 5 : } 1 \xrightarrow{3} 2 \xrightarrow{6} 3 \xrightarrow{4} 5 \xrightarrow{5} 4 \xrightarrow{15} 6 \xrightarrow{25} 1 = 57$$

$$\text{노드 6 : } 1 \xrightarrow{3} 2 \xrightarrow{6} 3 \xrightarrow{4} 5 \xrightarrow{5} 4 \xrightarrow{15} 6 \xrightarrow{25} 1 = 58$$

초기 경로가 설정된 값에 대해 최적해에 근접하기 위해 간선 교환 방법 (k-opt)이 제안되었다[1,2,7,8,12]. 그림 2는 Stougie[8]가 제시한 2-간선 교환 (2-opt) 방법과 3-간선 교환 (3-opt) 방법의 예이다. 그러나 전형적인 첫 번째 노드부터 가장 이웃한 노드들을 탐색하는 초기해에 대해 그림 2의 간선 교환 방법을 적용하여도 최적해를 얻지 못한다. 이는 발견적 방법들이 공통적으로 직면하고 있는 문제로 언덕 오르기 방법 (hill climbing method)을 적용하여 초기해가 최적해로 도달하는 과정에서 지역 최적점 (local minimum)에 빠져 전역 최적점 (global minimum)인 최적해 z_{opt} 에 도달하지 못하기 때문이다[7,13]. 또한, 대부분의 외판원 문제는 2-opt만 적용되며, Stougie[8]가 제시한 3-opt는 거의 적용되지 않는 문제점이 있다. 이에 대한 근거는 4장에서 적용된 실험 데이터들에서 참고할 수 있다.

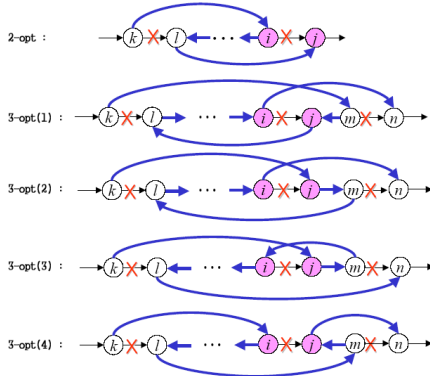


그림 2. 간선 교환 방법 (2-opt와 3-opt)
Fig. 2. Edge Swap Method (2-opt and 3-opt)

m 개의 노드가 존재하는 경우의 일반적인 TSP는 $m = n$ 인 $m \times m$ 정방행렬에 대해서만 고려하며, 비대칭 행렬에 대해서는 고려하지 않는다. 왜냐하면, TSP는 m 개의 노드를 목걸이 (또는 염주)와 같이 사이클을 만드는 문제로, 반드시 $m = n$ 이 되어야 하며, $m \neq n$ 이 될 수 없기 때문이다. 또한, 행렬 값 c_{ij} 는 i 노드에서 j 노드로의 이론상의 직선거리가 아닌 실제 경로 (직접 경로 또는 다른 노드를 거치는 간접 경로)의 최단거리이다. 따라서, $|c_{ij}| = m^2 - m$, ($c_{ij} = \infty$, $i = j$)

의 최단경로길이를 생성한다. 이와 같은 이유로, 본 논문에서는 일반적인 TSP에 한정하여 대칭행렬에 대해서만 고려하며, 비대칭 행렬은 고려하지 않는다.

TSP에 대한 k-opt 간선교환 기본 개념은 Barun et al.[14]과 Helsgaun[15]에서 제시하고 있다. 그러나 TSP의 해밀턴 사이클에 대한 3-opt와 4-opt의 가능한 모든 경우수에 대해서는 제시되지 않고 있으며, 어떤 간선을 대상으로 k-opt를 수행하는지에 대한 기준을 제시하지 않고 있다. 만약, 모든 간선을 대상으로 k-opt를 수행하면 전수탐색법과 유사한 수행 횟수가 필요하다. 3장에서는 2-opt 간선 교환 방법 개념에 기반하여 TSP의 실제 문제에 적합한 확장된 3-opt와 4-opt를 추가적으로 제시하고, k-opt를 수행하는 대상을 확정하는 기준을 제시하여 TSP의 최적해 z_{opt} 를 얻을 수 있는 알고리즘을 제안한다.

III. 외판원 문제의 k-opt 알고리즘

TSP의 최적해를 구하기 위해서는 초기해를 먼저 결정해야 한다. 전통적인 TSP의 초기해는 첫 번째 노드부터 시작하여 가장 인접한 노드들을 방문하는 방법을 적용하고 있다. 본 논문에서는 2가지 다른 방법을 적용한다. 먼저, 각 노드의 가장 인접한 노드 (행의 최소 간선)를 선택한다. 이때 $\{i, j\}$, ($i = j$)는 고려하지 않는다. 선택된 값들 중에서 최소값을 가진 노드부터 시작하여 가장 인접한 노드들을 방문하는 Min-Min Method와 선택한 값들 중 최대값을 가진 노드부터 시작하여 가장 인접한 노드들을 방문하는 Min-Max Method를 적용한다. Min-Min과 Min-Max Method로 얻은 경로 길이의 합이 최소가 되는 z_{min} 을 초기해로 결정한다.

간선 교환 방법을 적용하여 초기해 z_{min} 을 개선하여 최적해 z_{opt} 를 얻기 위해서는 어떤 간선을 교환할 것인가가 문제로 제기된다. 본 논문에서는 초기해의 간선들의 집합 e_{tsp} 을 교환 가능성이 높은 간선 집합 e_{ex} 와 교환 가능성이 희박한 간선들의 집합 e_{min} 으로 분류한다. 이를 위해 e_{tsp} 의 간선들에 대해 대칭행렬이므로 $w\{i, j\}$ 와 동일한 값인 $w\{j, i\}$ 를 삭제한다. 다음으로 행과 열 모두 최소값 ($w\{i, j\} = (\min c_i \cap \min c_j)$)이면 e_{min} 에, 어느 하나가 최소가 아닌 간선 ($w\{i, j\} \neq (\min c_i \cap \min c_j)$)이면 e_{ex} 에 저장한다.

초기해에 대해 1차 개선은 e_{ex} 간선들에 대해 최대값 간선부터 내림차순으로 그림 3의 간선 교환 방법 (2-opt와 3-opt)을 적용한다. 이때 새로 추가된 간선들은 e_{min} 에 저장한다.

본 논문에서는 k-opt를 그림 3과 같이 2-opt, 3-opt (1,2,3)와 4-opt로 제시하였다. 여기서 2-opt는 일반적으로 적용되는 방법이며, 3-opt (1,2,3)와 4-opt는 기존의 방법과 다른 것이다. 3-opt 적용 방법은 $(w\{i,j\} \neq (\min c_i \cap \min c_j))$ 의 배치 형태에 따라 결정될 수 있다.

본 논문에서 적용된 실험 데이터들은 모두 그림 3의 2-opt, 3-opt (1,2,3)과 4-opt로 문제가 해결되었다. 그러나 보다 일반적인 문제 해결을 위해 그림 4의 3-opt도 추가로 제안한다. 그림 4 이외에도 다양한 종류의 3-opt들이 추가로 고려될 수 있다.

기존의 k-opt를 적용하는 방법은 n 개 도시를 연결하는 $n-1$ 개 간선 중에서 $w\{i,j\}$ 를 기준으로 $w\{i,j\}$ 에 인접된 2개의 간선을 제외한 $n-3$ 개의 간선을 모두 비교하는 방법을 적용하였다. 반면에, 비교 횟수를 단축시킬 수 있는 방법을 2-opt에 대해 정의하면 $w\{i,j\}$ 를 기준으로 교환될 간선 $w\{k,l\}$ 을 다음과 같이 결정한다.

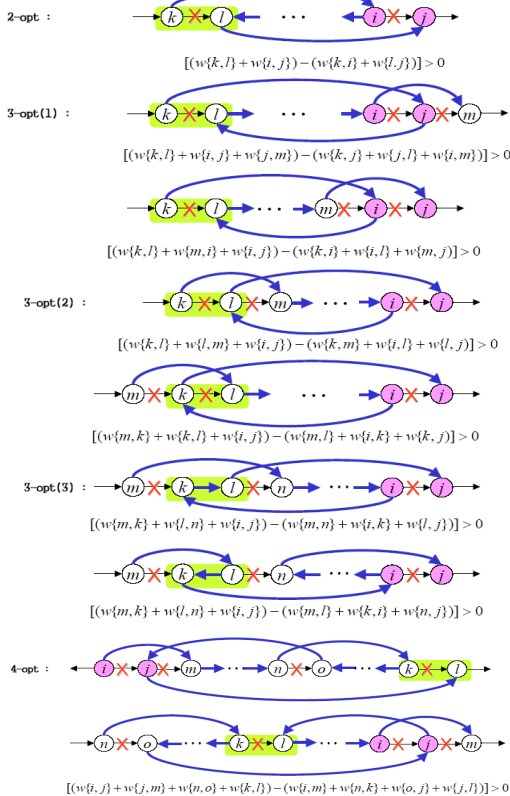


그림 3. 기본적인 k-opt
Fig. 3. Basic k-opt

- (1) k 노드 (유출)에서 i 노드 (유입)로의 간선 $w\{k,i\}$ 를 찾는 것으로 i 열 (유입)에서 $w\{i,j\}$ 보다 작은 비용을 갖는 행 (유출)이 대상이 된다.
- (2) l 노드 (유출)에서 j 노드 (유입)로의 간선 $w\{l,j\}$ 를 찾는 것으로 j 열 (유입)에서 $w\{i,j\}$ 보다 작은 비용을 갖는 행 (유출)이 대상이 된다.
- (3) 초기해로 연결된 경로에서 k 노드와 l 노드들의 간선이 $w\{k,l\}$ 로 결정된다.

본 논문에서 적용된 실험 데이터들은 모두 그림 3의 2-opt, 3-opt (1,2,3)과 4-opt로 문제가 해결되었다. 그러나 보다 일반적인 문제 해결을 위해 그림 4의 3-opt도 추가로 제안한다. 그림 4 이외에도 다양한 종류의 3-opt들이 추가로 고려될 수 있다.

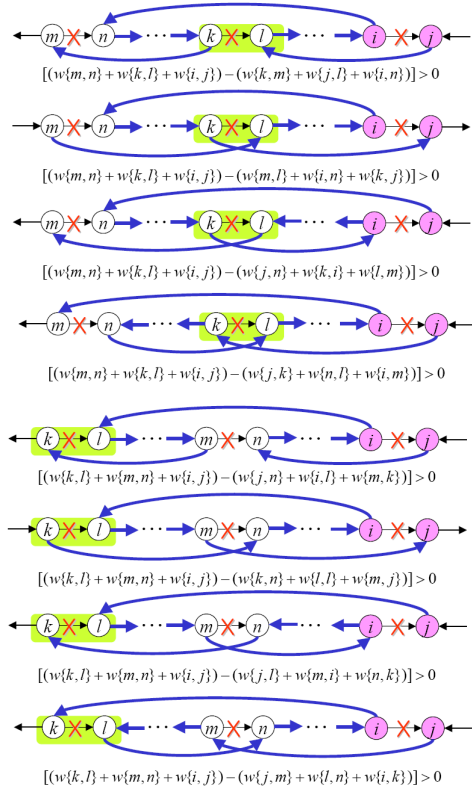


그림 4. 추가적인 3-opt
Fig. 4. Additional 3-opt

그림 3의 2-opt와 3-opt (1,2,3)를 적용하여 비용 절감 효과가 가장 큰 간선들을 교환하면 새로운 경로가 설정된다. 초기해에 대해 1차 개선 결과 얻은 경로는 초기해와 다른 경

로가 되어 $e_{\min}$ 간선들에 대해 교환 가능한 경우가 발생할 수 있다. 따라서 2차 개선은 $e_{\min}$ 간선들에 대해 비용 내림차순으로 2-opt, 3-opt (1,2,3)를 적용하여 새로운 경로를 설정한다. 이때 교환되지 않은 간선들을 e_{rem} 에 저장한다. 경로가 변경되면 2차 개선과정에서 교환되지 않은 간선들에 대해 3-opt로 간선 교환이 이루어질 가능성이 있다. 따라서 2차 개선 후 변경된 경로에 대해 e_{rem} 간선들을 대상으로 3-opt와 4-opt를 적용하여 최종적으로 z_{opt} 를 얻는다. 본 알고리즘의 상세한 내용은 그림 5에 제시되어 있다.

Step 1. 초기 경로 설정

1-1. 각 노드 (행)의 최소값 ($\min C_i$) 선택

1-2. 이중 탐색

1-2-1. 선택된 값들 중 최소값 (Min) 행 노드 i 에서 출발, "Nearest Neighbor Search" 수행. (Min-Min Method)

1-2-2. 선택된 값들 중 최대값 (Max) 행 노드 i 에서 출발, "Nearest Neighbor Search" 수행. (Min-Max Method)

1-3. 최소 경로길이 $z_{\min}$ 방법 선택, 간선들을 e_{up} 에 저장

Step 2. $z_{\min}$ 에 대해 해 개선

2-1. $w\{i,j\}$ 에 대해 $w\{j,i\}$ 삭제, $e_{up} = e_{ex} + e_{\min}$ 로 분리
 e_{ex} = 각 노드의 행 (유출)과 열 (유입)이 최소값이 아닌 간선들
 $e_{\min}$ = 각 노드의 행 (유출)과 열 (유입)이 최소값인 간선들
 if $e_{ex} = \{\emptyset\}$ then $z_{opt} \leftarrow z_{\min}$, e_{up} 를 최적해 경로로 설정, 알고리즘 종료.

2-2. 1차 개선
 e_{ex} 에 대해 비용 내림차순으로 2-opt와 해당 3-opt (1,2,3) 수행, 가장 큰 비용 절감 값으로 경로 수정, 새로 추가된 간선을 $e_{\min}$ 에 저장.

2-3. 2차 개선
 $e_{\min}$ 에 대해 비용 내림차순으로 2-opt와 해당 3-opt (1,2,3) 수행, 가장 큰 비용 절감 값으로 경로 수정, 교환된 간선은 $e_{\min}$ 에서 삭제, 새로 추가된 간선은 $e_{\min}$ 에 저장, 교환되지 않은 간선은 e_{rem} 에 저장.

2-4. 3차 개선
 e_{rem} 에 대해 비용 내림차순으로 3-opt와 4-opt 수행, 가장 큰 비용절감 값으로 경로 수정.

(Nearest Neighbor Search)

- 초기값 : 방문 노드 집합 $T = \{i\}$ 저장, 미방문 노드 집합 $V = V \setminus i$ 저장
- 다음 방문 노드 결정 : $j \in V$ 인 최소값 간선 선택
 (미방문 노드들 중 가장 가까운 노드 방문)
- if $|V| = 1$ then 해당 노드의 간선 선택
- if $V = \emptyset$ then 시작 노드의 간선 선택

그림 5. TSP의 확장된 k-opt 알고리즘
Fig. 5. The Extended k-opt TSP Algorithm

간선 교환 방법인 k-opt는 대칭행렬에만 적용된다. 왜냐하면 대칭행렬은 무방향 그래프 (Undirected Graph)로 간선 교환에 따라 경로가 반대로 변경되어도 간선 $w\{i,j\} = w\{j,i\}$ 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 경로길이 증가할 수 있기 때문이다. 결국, k-opt는 대칭행렬에만 한정되어 적용될 수 있다.

TSP-1에 대해 Min-Min Method와 Min-Max Method로 초기 경로를 탐색한 결과는 그림 6과 같다. 초기해에 대해

Min-Min Method와 Min-Max Method가 동일한 경로길이를 얻어 Min-Min Method를 대상으로 해 개선을 수행하였으며, 본 문제는 $e_{ex} = 25\{6,1\}$ 에 대해 2-opt만을 수행하여 최적해 $z_{opt} = 56$ 을 얻는데 성공하였으며, $e_{\min}$ 과 e_{rem} 은 간선 교환 대상이 존재하지 않아 해 개선이 수행되지 않았다.

	1	2	3	4	5	6
1		3	10	11	7	25
2	3		6	12	8	26
3	10	6		9	4	20
4	11	12	9		5	15
5	7	8	4	5		18
6	25	26	20	15	18	

최소값 = $3\{1,3\}$, 최대값 = $15\{6,4\}$

Min-Min Method : $1 \xrightarrow{3} 2 \xrightarrow{6} 3 \xrightarrow{4} 5 \xrightarrow{4} 6 \xrightarrow{15} 1 \xrightarrow{25} 58$

방문 노드	$\min C_i$	T	V	선택	다음 방문노드
1	$\{1,2\} = 3$	$\{1\}$	$\{2,3,4,5,6\}$	$\{1,2\} = 3$	2
2	$\{2,1\} = 3$ $\{2,3\} = 6$	$\{1,2\}$ $\{1,2\}$	$\{3,4,5,6\}$ $\{3,4,5,6\}$	- $\{2,3\} = 6$	3
3	$\{3,5\} = 4$	$\{1,2,3\}$	$\{4,5,6\}$	$\{3,5\} = 4$	5
5	$\{5,3\} = 4$ $\{5,4\} = 5$	$\{1,2,3,5\}$ $\{1,2,3,5\}$	$\{4,6\}$ $\{4,6\}$	- $\{5,4\} = 5$	4
4	$\{4,6\} = 15$	$\{1,2,3,4,5\}$	$\{6\}$	$\{4,6\} = 15$	6
6	$\{6,1\} = 25$	$\{1,2,3,4,5,6\}$	$\emptyset$	$\{6,1\} = 25$	1
경로길이				58	

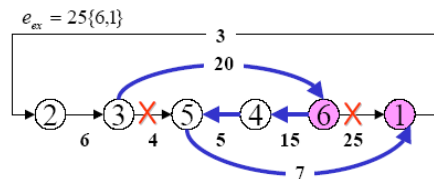
Min-Max Method : $6 \xrightarrow{15} 4 \xrightarrow{5} 5 \xrightarrow{4} 3 \xrightarrow{2} 1 \xrightarrow{25} 6 = 58$

초기해 : $z_{ini} = 1 \xrightarrow{3} 2 \xrightarrow{6} 3 \xrightarrow{4} 5 \xrightarrow{4} 6 \xrightarrow{15} 1 \xrightarrow{25} 58$

	1	2	3	4	5	6	$\min C_i$
1		3	10	11	7		○
2			6	12	8	26	○
3	10			9	4	20	○
4	11	12	9		5	15	×
5	7	8	4	5		18	○
6	25	26	20		18		×
$\min C_j$	×	○	×	○	○	○	

$e_{ex} = 25\{6,1\}$,

$e_{\min} = 3\{1,2\}, 6\{2,3\}, 4\{3,5\}, 15\{4,6\}, 5\{5,4\}$



최적해 : $z_{opt} = 1 \xrightarrow{3} 2 \xrightarrow{6} 3 \xrightarrow{20} 6 \xrightarrow{15} 4 \xrightarrow{5} 5 \xrightarrow{7} 1 = 56$

그림 6. TSP-1의 2-opt 최적해
Fig. 6. Optimal solution of 2-opt for TSP-1

IV. 알고리즘 적용 및 분석

실험에 적용되는 데이터는 6개로 그림 7에 제시되어 있다. TSP-2는 Pleines[5]에서, TSP-3는 Tsilingiris[16]에서, TSP-4는 Pleines[5]에서, TSP-5와 TSP-6는 Kratica와 Radojevi[11]에서, TSP-7은 Pleines[5]에서 인용되었다. TSP-2은 TSP-1과 동일하게 6개 도시를 방문하는 문제이며, TSP-3~TSP-6은 10개 도시를 방문하는 경우이다. TSP-7은 26개 도시를 방문하는 경우로 본 실험에서는 가장 어려운 문제이다.

	1	2	3	4	5	6
1		12	25	30	28	22
2	12		16	20	22	10
3	25	16		23	26	21
4	30	20	23		31	18
5	28	22	26	31		14
6	22	10	21	18	14	

(a) TSP-2

	1	2	3	4	5	6	7	8	9	10
1		8	5	9	12	14	12	16	17	22
2	8		9	15	17	8	11	18	14	22
3	5	9		7	9	11	7	12	12	17
4	9	15	7		3	17	10	7	15	18
5	12	17	9	3		8	10	6	15	15
6	14	8	11	17	8		9	14	8	16
7	12	11	7	10	10	9		8	6	11
8	16	18	12	7	6	14	8		11	11
9	17	14	12	15	15	8	6	11		10
10	22	22	17	18	15	16	11	11	10	

(b) TSP-3

	1	2	3	4	5	6	7	8	9	10
1		16	44	93	1	30	30	5	78	42
2	16		68	61	42	77	41	79	22	32
3	44	68		39	48	21	36	28	40	80
4	93	61	39		43	8	66	46	30	35
5	1	42	48	43		67	69	11	84	91
6	30	77	21	8	67		97	43	63	67
7	30	41	36	66	69	97		85	89	18
8	5	79	28	46	11	43	85		2	85
9	78	22	40	30	84	63	89	2		5
10	42	32	80	35	91	67	18	85	5	

(c) TSP-4

	1	2	3	4	5	6	7	8	9	10
1		96	105	50	41	86	46	29	56	70
2	96		78	49	94	21	64	63	41	37
3	105	78		60	94	61	54	86	76	51
4	50	49	60		45	38	20	26	17	18
5	41	94	84	45		80	36	55	59	64
6	86	21	61	35	80		46	50	28	8
7	46	64	54	20	36	46		45	37	30
8	29	63	86	26	55	90	45		21	45
9	56	41	76	17	59	28	37	21		25
10	70	37	51	18	64	8	30	45	25	

(d) TSP-5

	1	2	3	4	5	6	7	8	9	10
1		72	43	217	92	183	116	143	161	141
2	72		51	222	52	182	99	85	186	72
3	43	51		229	97	199	121	115	203	101
4	217	222	229		219	40	156	285	63	289
5	92	52	97	219		179	57	123	159	100
6	183	182	199	40	179		130	265	65	281
7	116	99	121	156	97	130		158	91	184
8	143	85	115	285	123	265	158		269	14
9	161	186	203	63	159	65	91	269		255
10	141	72	101	299	100	251	154	14	255	

(e) TSP-6

	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	4538	4538	4073	3730	3031	3641	3384	3314	3904	4071	5430	3079	3140	2318	2260	2008	2288	2204	2288	1876	2019	2542	1901
2	4074		3532	5126	2206	2605	2662	3857	907	2751	1884	1058	5140	1380	1469	1805	2357	1961	1800	2871	2205	3738	2632	1845	2263	2603
3	639	3532		3687	2889	5478	1086	3466	3004	2715	2875	2846	952	2987	2605	2509	1731	1718	1561	2347	1728	1841	637	1380	1903	1709
4	4538	5126	3687		1141	638	977	3396	3006	972	1400	2609	2506	2128	1696	2304	2795	1141	2276	1968	2048	2218	3079	2853	2861	2913
5	4538	2206	2889	1141		1284	447	3849	3409	1174	1001	2738	2800	2770	1808	2606	2037	3206	2079	2170	2448	2421	5277	2845	2653	5119
6	4073	2605	5478	638	1284		407	2828	2775	752	898	2102	2050	1487	1106	1749	2070	2668	2688	1406	1795	1858	2659	2116	1890	2387
7	3730	2662	1086	977	647	407		2779	2729	409	810	1381	1727	1509	1060	1703	2220	2864	2183	1362	1463	1768	2536	1793	1830	2338
8	3031	3857	907	2751	1884	1058	5140		2870	1269	626	2764	1658	1712	1076	1440	1054	1079	1417	2187	1877	2070	3039	1440	1380	1590
9	3641	907	3004	3006	3409	2775	2729	409		2330	1919	908	2715	1615	1669	1036	1370	1904	1529	1367	2135	1527	2799	1869	1430	1850
10	3384	2751	2715	972	1174	752	409	2370	2330		426	1637	1354	1099	684	1332	1823	2117	1804	966	1074	1827	2101	1391	1891	1941
11	3904	4071	5430	2609	1400	1001	810	1381	1727	1509		1060	1703	2220	2864	2183	1362	1463	1768	2536	1793	1830	2338	1590	1380	1590
12	3079	3140	2318	2260	2008	2808	2204	2288	1876	2019	2542		1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901
13	3079	3140	2318	2260	2008	2808	2204	2288	1876	2019	2542	1901		1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901
14	3079	3140	2318	2260	2008	2808	2204	2288	1876	2019	2542	1901	1901		1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901
15	3079	3140	2318	2260	2008	2808	2204	2288	1876	2019	2542	1901	1901	1901		1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901
16	3079	3140	2318	2260	2008	2808	2204	2288	1876	2019	2542	1901	1901	1901	1901		1901	1901	1901	1901	1901	1901	1901	1901	1901	1901
17	3079	3140	2318	2260	2008	2808	2204	2288	1876	2019	2542	1901	1901	1901	1901	1901		1901	1901	1901	1901	1901	1901	1901	1901	1901
18	3079	3140	2318	2260	2008	2808	2204	2288	1876	2019	2542	1901	1901	1901	1901	1901	1901		1901	1901	1901	1901	1901	1901	1901	1901
19	3079	3140	2318	2260	2008	2808	2204	2288	1876	2019	2542	1901	1901	1901	1901	1901	1901	1901		1901	1901	1901	1901	1901	1901	1901
20	3079	3140	2318	2260	2008	2808	2204	2288	1876	2019	2542	1901	1901	1901	1901	1901	1901	1901	1901		1901	1901	1901	1901	1901	1901
21	3079	3140	2318	2260	2008	2808	2204	2288	1876	2019	2542	1901	1901	1901	1901	1901	1901	1901	1901	1901		1901	1901	1901	1901	1901
22	3079	3140	2318	2260	2008	2808	2204	2288	1876	2019	2542	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901		1901	1901	1901	1901
23	3079	3140	2318	2260	2008	2808	2204	2288	1876	2019	2542	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901		1901	1901	1901
24	3079	3140	2318	2260	2008	2808	2204	2288	1876	2019	2542	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901		1901	1901
25	3079	3140	2318	2260	2008	2808	2204	2288	1876	2019	2542	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901		1901
26	3079	3140	2318	2260	2008	2808	2204	2288	1876	2019	2542	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	

(f) TSP-7

그림 7. 실험 데이터
Fig. 7. Experimental Data

Pleines[5]는 TSP-2에 대한 최적해로 $z_{opt} = 1 \rightarrow 2 \rightarrow 3 \rightarrow 12 \rightarrow 16$

$\rightarrow 4 \rightarrow 6 \rightarrow 5 \rightarrow 1 = 111$ 을 제시하고 있다. TSP-2 문제에 제안된 간선 교환 알고리즘을 적용한 결과는 그림 8에 제시되어 있다. 본 문제에 대해 초기해로 Min-Min 방법을 선택하고 e_{ex} 에 대해 해 개선작업을 수행한 결과 Pleines[5]가 제시한 최적해 z_{opt} 와 동일한 값을 얻었다.

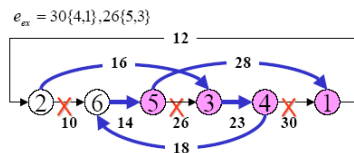
Min-Min Method : $2 \rightarrow 6 \rightarrow 5 \rightarrow 3 \rightarrow 4 \rightarrow 1 \rightarrow 2 = 115$

Min-Max Method : $4 \rightarrow 6 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 5 \rightarrow 4 = 122$

초기해 : $z_{ini} = 2 \rightarrow 6 \rightarrow 5 \rightarrow 3 \rightarrow 4 \rightarrow 1 \rightarrow 2 = 115$

	1	2	3	4	5	6	$m_{in}C_i$
1		12	25		28	22	○
2			16	20	22	10	○
3	25	16		23		21	×
4	30	20			31	18	×
5	28	22	26	31			×
6	22		21	18	14		○
$m_{in}C_i$	×	○	×	×	○	○	

$e_{ex} = 30 \{4, 1\}, 26 \{5, 3\}, 23 \{3, 4\}$



최적해 : $z_{opt} = 2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 5 \rightarrow 1 = 111$

그림 8. TSP-2의 3-opt 알고리즘 최적해
Fig. 8. Optimal solution of 3-opt for TSP-2

TSP-3에 대해 Tsilingiris[16]는 Nearest-Neighbor Heuristic Method (1st Node Method)로 $1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 8 \rightarrow 7 \rightarrow 9 \rightarrow 6 \rightarrow 2 \rightarrow 10 \rightarrow 1 = 95$, Closest Insertion Heuristic Method는 79, Geometric Heuristic Method는 73을, TSPdyn via Dynamic Programming Method는 $2 \rightarrow 6 \rightarrow 5 \rightarrow 4 \rightarrow 8 \rightarrow 10 \rightarrow 9 \rightarrow 7 \rightarrow 3 \rightarrow 1 \rightarrow 2 = 73$ 을 얻었으며, 최적해로 $z_{opt} = 73$ 을 제시하고 있다. 제안된 간선 교환 알고리즘을 적용한 결과는 그림 9에 제시되어 있다. 본 문제에 대해 초기해로 Min-Max 방법을 선택하고 $e_{ex} = \{\emptyset\}$ 가 되어 초기해가 최적해로 결정되었으며, Tsilingiris[14]와 동일한 결과를 얻었다.

TSP-4에 대해 Pleines[5]는 Erster Schritt는 176, Weitere Iterationschritte는 176, Ergebnis는 175를 얻었음을 제시하였다. 제안된 간선 교환 알고리즘을 적용한 결과는 (그림 10)에 제시되어 있다. 본 문제에 대해 초기해로 Min-Min 방법을 선택하고 e_{ex} 의 61 {4,2}에 대해서만 해 개선작업이 수행되어 Pleines[5]가 제시한 최적해 z_{opt} 와 동일한 값을 얻는데 성공하였다.

	1	2	3	4	5	6	7	8	9	10
1		8	5	9	12	14	12	16	17	22
2	8		9	15	17	8	11	18	14	22
3	5	9		7	9	11	7	12	12	17
4	9	15	7		3	17	10	7	15	18
5	12	17	9	3		8	10	6	15	15
6	14	8	11	17	8		9	14	8	16
7	12	11	7	10	10	9		8	6	11
8	16	18	12	7	6	14	8		11	11
9	17	14	12	15	15	8	6	11		10
10	22	22	17	18	15	16	11	11	10	

Min-Min Method : $4 \rightarrow 5 \rightarrow 8 \rightarrow 7 \rightarrow 9 \rightarrow 6 \rightarrow 8 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 10 \rightarrow 4 = 87$

Min-Max Method : $10 \rightarrow 9 \rightarrow 7 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow 8 \rightarrow 6 \rightarrow 5 \rightarrow 4 \rightarrow 8 \rightarrow 10 = 73$

초기해 : $z_{ini} = 10 \rightarrow 9 \rightarrow 7 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow 8 \rightarrow 6 \rightarrow 5 \rightarrow 4 \rightarrow 8 \rightarrow 10 = 73$

	1	2	3	4	5	6	7	8	9	10	$min C_i$
1		8		9	12	14	12	16	17	22	○
2			9	15	17	8	11	18	14	22	○
3	5	9		7	9	11		12	12	17	○
4	9	15	7			17	10	7	15	18	○
5	12	17	9	3			10	6	15	15	○
6	14		11	17	8		9	14	8	16	○
7	12	11	7	10	10	9		8		11	○
8	16	18	12		6	14	8		11	11	×
9	17	14	12	15	15		6	11			○
10	22	22	17	18	15	16	11		10		○
$min C_j$	○	○	○	○	×	○	×	×	×	○	

$$e_{ex} = \{\emptyset\} \Rightarrow z_{ini} = z_{opt}$$

최적해 : $z_{opt} = 10 \rightarrow 9 \rightarrow 7 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow 8 \rightarrow 6 \rightarrow 5 \rightarrow 4 \rightarrow 8 \rightarrow 10 = 73$

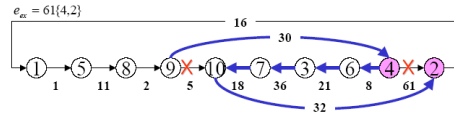
그림 9. TSP-3의 0-opt 알고리즘 최적해
Fig. 9. Optimal solution of 0-opt for TSP-3

Min-Min $z_{21} \rightarrow 6 \rightarrow 4 \rightarrow 30 \rightarrow 9 \rightarrow 8 \rightarrow 5 \rightarrow 1 \rightarrow 5 \rightarrow 42 \rightarrow 32 \rightarrow 10 \rightarrow 18 \rightarrow 7 \rightarrow 36 = 195$

초기해 : $z_{ini} = 1 \rightarrow 5 \rightarrow 11 \rightarrow 2 \rightarrow 9 \rightarrow 5 \rightarrow 10 \rightarrow 18 \rightarrow 36 \rightarrow 21 \rightarrow 8 \rightarrow 61 \rightarrow 16 = 179$

	1	2	3	4	5	6	7	8	9	10	$min C_i$
1			44	93	1	30	30	5	78	42	○
2	16		68		42	77	41	79	22	32	○
3	44	68		39	48	21		28	40	80	○
4	93	61	39		43		66	46	30	35	×
5		42	48	43		67	69	11	84	91	○
6	30	77		8	67		97	43	63	67	○
7	30	41	36	66	69		97	85	89		×
8	5	79	28	46		43	85		2	85	○
9	78	22	40	30	84	63	89		5		○
10	42	32	80	35	91	67	18	85			○
$min C_j$	×	×	×	○	○	○	○	×	○	○	

$$e_{ex} = 61 \{4, 2\}, 36 \{7, 3\}$$



최적해 : $z_{opt} = 1 \rightarrow 5 \rightarrow 11 \rightarrow 2 \rightarrow 9 \rightarrow 5 \rightarrow 10 \rightarrow 18 \rightarrow 36 \rightarrow 21 \rightarrow 8 \rightarrow 61 \rightarrow 16 = 175$

그림 10. TSP-4의 2-opt 알고리즘 최적해
Fig. 10. Optimal solution of 2-opt for TSP-4

TSP-5에 대해 Kratica와 Radojevi[11]는 1st Node Method는 $1 \rightarrow 8 \rightarrow 9 \rightarrow 4 \rightarrow 10 \rightarrow 6 \rightarrow 2 \rightarrow 7 \rightarrow 5 \rightarrow 3 \rightarrow 1 = 403$, Min-Min Method는 $6 \rightarrow 10 \rightarrow 4 \rightarrow 9 \rightarrow 8 \rightarrow 1 \rightarrow 5 \rightarrow 7 \rightarrow 3 \rightarrow 2 \rightarrow 6 = 323$, 최적해는 $6 \rightarrow 10 \rightarrow 4 \rightarrow 9 \rightarrow 8 \rightarrow 1 \rightarrow 5 \rightarrow 7 \rightarrow 3 \rightarrow 2 \rightarrow 6 = 323$ 을 제시하고 있다. 제안된 간선 교환 알고리즘을 적용한 결과는 그림 11에 제시되어 있다. 본 문제에 대해 초기해로 Min-Min 방법을 선택하고 e_{ex} , e_{min} 에 대해 해 개선 작업을 수행한 결과 개선 대상이 없어 초기해를 최적해로 결정하였으며, 이는 Kratica와 Radojevi[11]가 제시한 최적해 z_{opt} 와 동일한 값을 알 수 있다.

Min-Min Method : $6 \rightarrow 10 \rightarrow 4 \rightarrow 9 \rightarrow 8 \rightarrow 1 \rightarrow 5 \rightarrow 7 \rightarrow 3 \rightarrow 2 \rightarrow 6 = 323$

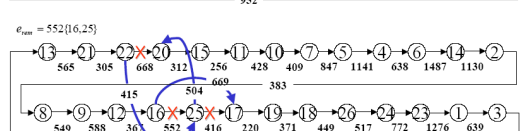
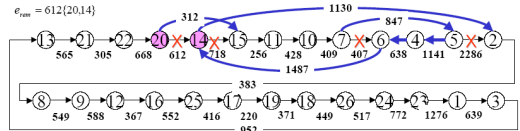
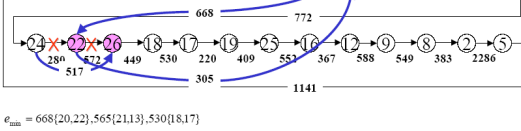
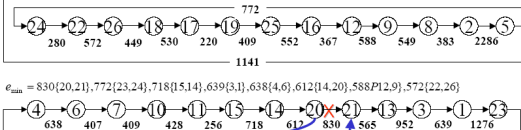
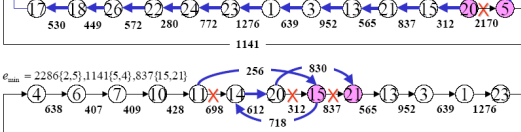
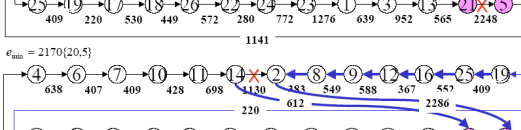
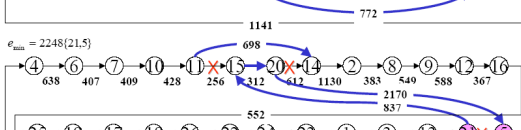
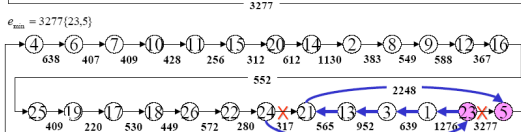
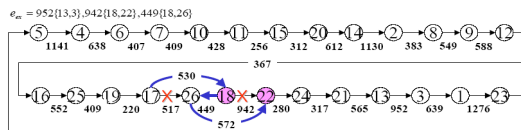
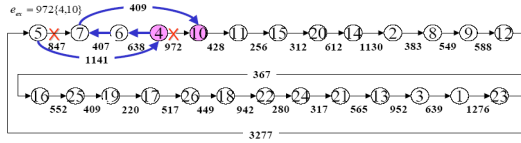
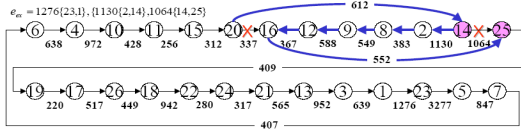
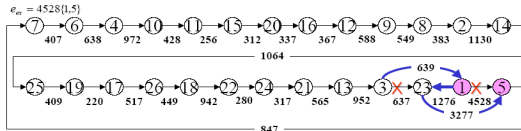
Min-Max Method : $5 \rightarrow 7 \rightarrow 4 \rightarrow 9 \rightarrow 8 \rightarrow 1 \rightarrow 10 \rightarrow 6 \rightarrow 2 \rightarrow 3 \rightarrow 5 = 384$

초기해 : $z_{ini} = 6 \rightarrow 10 \rightarrow 4 \rightarrow 9 \rightarrow 8 \rightarrow 1 \rightarrow 5 \rightarrow 7 \rightarrow 3 \rightarrow 2 \rightarrow 6 = 323$

그림 12. TSP-6의 2-opt 알고리즘 최적해
Fig. 12. Optimal solution of 2-opt for TSP-6

$$e_{ex}=265 \{6.8\}, 92 \{1.5\}$$

$e_{c,2} = 4528 \{1,5\}, 1276 \{23,1\}, 1130 \{2,14\}, 1064 \{14,25\}, 972 \{4,10\}, 952 \{13,3\}, 942 \{18,22\}, 517 \{17,26\}, 449 \{26,18\}$



최적해 : $z_{opt} = 1 \xrightarrow{639} 2 \xrightarrow{952} 3 \xrightarrow{565} 13 \xrightarrow{21} 21 \xrightarrow{305} 22 \xrightarrow{415} 25 \xrightarrow{504} 20 \xrightarrow{312} 15$
 $\xrightarrow{256} 11 \xrightarrow{428} 10 \xrightarrow{409} 7 \xrightarrow{847} 5 \xrightarrow{1141} 4 \xrightarrow{638} 6 \xrightarrow{1487} 14 \xrightarrow{1130} 2 \xrightarrow{383} 8 \xrightarrow{549}$
 $\xrightarrow{588} 12 \xrightarrow{367} 16 \xrightarrow{669} 17 \xrightarrow{220} 19 \xrightarrow{371} 18 \xrightarrow{449} 26 \xrightarrow{517} 24 \xrightarrow{772} 23 \xrightarrow{1276} 1 =$
16,189

그림 13. TSP-7의 k-opt ($k = 2, 3, 4$) 알고리즘 최적해
Fig. 13. Optimal solution of k-opt ($k = 2, 3, 4$) for TSP-7

제안된 알고리즘은 6개의 도시 방문 문제 (TSP-1, TSP-2), 10개 도시 방문 문제 (TSP-3, TSP-4, TSP-5, TSP-6)와 26개 도시 방문 문제 (TSP-7)에 대해 기존에 알려진 최적해를 구하는데 성공하였다.

제안된 min(min-min, min-max) 초기치에 대해 k-opt를 적용한 결과는 표 1과 같다. TSP-3와 TSP-5는 초기치에 대해 k-opt 적용 없이 최적해를 구하였음을 알 수 있다. TSP-1, TSP-4와 TSP-6는 2-opt로 최적해를 구하였으며, TSP-2는 3-opt로 최적해를 구하였다. 특히하게도 TSP-7과 같이 큰 데이터의 경우 2,3과 4-opt가 혼합 적용되어 최적해를 구하였다. 결론적으로, 제안된 알고리즘은 기존의 k-opt 기본개념을 확장시킨 확장된 k-opt를 제안하였으며, 초기해를 min(min-min, min-max)로 구하면 쉽게 k-opt를 적용할 수 있음을 보였다.

표 1. 제안 알고리즘 결과
Table 1. The Result of Proposed Algorithm

문제	알려진 최적해	제안 알고리즘	
		k-opt	최적해
TSP-1	56	2-opt	56
TSP-2	111	3-opt	111
TSP-3	73	0-opt	73
TSP-4	175	2-opt	175
TSP-5	323	0-opt	323
TSP-6	725	2-opt	725
TSP-7	16,189	2-, 3-, and 4-opt	16,189

V. 결론

본 논문은 지금까지 해결하지 못한 외판원 문제에 대해 k-opt 간선 교환 방법으로 지금까지 알려진 최적해를 구할 수 있음을 보였다. 이를 위해 초기해를 구하는 방법을 새로 제시하였으며, 기존의 간선 교환 방법인 2-opt를 확장하여 3-opt와 4-opt를 추가로 제안하였다. 또한, 초기해로부터 최적해를 도출하는 과정을 체계적으로 제안하였다.

제안된 알고리즘은 7개의 다양한 실제 문제들에 적용한 결과 모든 데이터에 대해 지금까지 알려진 최적해를 빠르게 구하는데 성공하였다. 제안된 알고리즘 검증에는 관련 데이터 부족으로 단지 7개 데이터만 적용하여 일반화된 알고리즘으로의 한계점이 존재한다. 따라서 추후 보다 많은 실제 데이터들에 적용하여 3-opt (1,2,3) 만으로 해결되는지 아니면 추가적으로 제안한 3-opt (4-11)가 활용되는지를 검증할 예정이다.

참고문헌

- [1] Wikipedia, "Travelling Salesman Problem," http://en.wikipedia.org/wiki/Travelling_Salesman_Problem, Wikimedia Foundation Inc., 2012.
- [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.
- [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, 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 Lösung Allgemeiner Traveling-Salesman-Problem (TSP)," Können bekannte Lösungen nicht nur auf Gesamtgraphen 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/O/120.pdf>, 2003.
- [7] E. Charniak and M. Herlihy, "CSC 751 Computational Complexity: Local Search Heuristics," Department of Computer Science, Brown University, 2008.
- [8] L. Stougie, "2P350: Optimisieringsmethoden," <http://www.win.tue.nl/~leen/OW/2P350/Week8/week8.pdf>, College Wordt gegeven op vrijdagmiddag, 2001.
- [9] W. Cook, "The Traveling Salesman Problem," The School of Industrial and Systems Engineering, Georgia Tech, 2008.
- [10] A. Battese, "Millennium Problems," Clay Mathematics Institute, <http://www.claymath.org/millennium/>, 2008.
- [11] J. Kratica and S. Radojevi, "One Improvement to Nearest Neighbor Method for Solving Traveling Salesman Problem," LIRA 95 Proceedings, pp. 77-82, Novi Sad, 1996.
- [12] 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.
- [13] H. D. Beale, "Neural Network Design," PWS Publishing Company, 1996.
- [14] B. Chandra, H. Karloff, and C. Tovey, "New Results on the Old k-Opt Algorithm for the TSP," *SIAM Journal on Computing*, Vol. 28, No. 6, pp. 1998-2029, 1999.
- [15] K. Helsagaun, "General k-Opt Submoves for the Lin-Kernighan TSP Heuristic," *Math. Prog. Comp.*, Vol. 1, pp. 119-163, 2009.
- [16] P. S. Tsilingiris, "A Multi-stage Decision-Support Methodology for The Optimization-based Linear-network Design," Degree of Diploma in Naval Architecture and Marine Engineering at the School of Naval Architecture and Marine Engineering of the National Technical University of Athens, 2005.

저 자 소 개



이 상 운(Sang-Un, Lee)

1983 ~ 1987년 :

한국항공대학교 항공전자공학과 (학사)

1995 ~ 1997년 :

경상대학교 컴퓨터과학과 (석사)

1998 ~ 2001년 :

경상대학교 컴퓨터과학과 (박사)

2003. 3 ~ 현재 :

강릉원주대학교 멀티미디어공학과 부교수

관심분야 : 소프트웨어 프로젝트 관리,

소프트웨어 개발 방법론,

소프트웨어 신뢰성,

신경망, 뉴로-퍼지,

그래프 알고리즘

e-mail : sulee@gwmu.ac.kr

