韓國經營科學會誌
第32권 第3號
2007年 9月

선행순서결정문제를 위한 Out-of-Kilter 해법의 적용과 부분순환로의 제거

권 상 호*

Elimination of Subtours Obtained by the Out-of-Kilter Algorithm for the Sequential Ordering Problem

Kwon, Sang-Ho*

■ Abstract ■

This paper presents two elimination methods of subtours, which is obtained by applying the Out-of-Kilter algorithm to the sequential ordering problem (SOP) to produce a feasible solution for the SOP. Since the SOP is a kind of asymmetric traveling salesman problem (ATSP) with precedence constraints, we can apply the Out-of-Kilter algorithm to the SOP by relaxing the precedence constraints. Instead of patching subtours, both of two elimination methods construct a feasible solution of the SOP by using arcs constructing the subtours, and they improve solution by running 3-opt and 4-opt at each iteration. We also use a perturbation method, cost relaxation to explore a global solution. Six cases from two elimination methods are presented and their experimental results are compared to each other. The proposed algorithm found 32 best known solutions out of the 34 instances from the TSPLIB in a reasonable time

Keyword : Elimination of Subtours, Out-of-Kilter, Sequential Ordering, Cost-Relaxation

1. 서 론

선행순서결정문제(Sequential Ordering Problem ; SOP)는 네트워크 $G(V, A)$ 교점집합, $V=\{1, 2, \cdots, n\}$ 호 집합 $A=\{(i, j): i \neq j, \forall i, j \in V\}$, 그리고 호 (i, j)의 비용 c_{ij}가 주어진 상태에서 선행제약조건(pre-

논문접수일 : 2007년 01월 02일 논문게재확정일 : 2007년 08월 16일
* 삼성전자 LCD 총괄 소속

cedence constraints)을 만족시키는 최소비용의 해밀턴경로(Hamiltonian path)를 찾는 문제이다. 이 문제는 비용행렬로 표현할 수 있는데, 비용행렬 $C=(c_{ij})$에서 호 (i,j)의 비용이 -1이면 교점 j는 교점 i에 선행되어야 한다는 조건이다. 또한 이 문제에서 해밀턴경로는 시작교점은 1이고 마지막교점은 n이라는 것을 가정한다. 다시 말해, 교점 1은 다른 모든 교점에 선행되어야 하며, 교점 n은 다른 모든 교점을 선행해서는 안 된다.

SOP는 선행관계가 있는 작업들을 그래프로 표현했을 때 최소거리의 위상순서(topological order)를 결정하는 위상정렬(topological sorting)문제라고도 할 수 있다. 비록 SOP는 생산, 라우팅, 스케줄링 등 실제적인 현실문제들에 다양하게 나타나고 있지만 이에 대한 최적해법은 과도한 계산량(complexity)으로 인해 비현실적이다. 따라서 빠른 시간내에 좋은 해를 구하는 효율적인 발견적 해법을 개발하는 것은 중요한 과제이다. SOP의 수리적 모형에 기초한 해법으로는 cutting plane[1], Lagrangian relax and cut[2], 그리고 branch and cut[3] 등이 있다. 발견적 해법(heuristic)은 크게 두 가지로 나누어 볼 수 있다. 그 중 하나는 유전해법(genetic algorithm)으로 MPO/AI[4]가 있으며, 이것은 최대부분순서/임의삽입(maximum partial order/arbitrary insertion)이라 불리는 교배(cross over)방법을 사용한다. 또한 Hybrid 해법[5]은 voronoi-quantized 교배방법을 이용하여 일부 TSPLIB[6]의 문제들에 대해 가장 좋은 해를 구하였다. 발견적 해법에서의 다른 하나는 가장 잘 알려진 Ant Colony System으로 HAS-SOP(hybrid ant colony system for SOP)[7]는 이웃해를 찾는 과정에서 선행제약조건을 만족시키며 3개의 호를 교환하는 방법을 이용하여 TSPLIB의 문제들에 대해 몇 개의 문제를 제외하고 지금까지 알려진 가장 좋은 해의 하한(lower bound)과 상한(upper bound)을 제시하였다. 그리고 최근 소개된 cooperative parallel rollout 해법[8]도 좋은 결과를 보여주었다.

본 연구의 목적은 SOP에 대해 지금까지 시도되지 않은 Out-of-Kilter 해법(OKA)[9, 10]을 적용하여 지금까지 알려진 가장 좋은 해를 효율적으로 구하는 것이다. OKA는 primal-dual이론을 기반으로 최소비용유량문제(capacitated minimum-cost flow problem)에 대해 최적해를 찾는 해법이다. SOP는 선행제약조건이 있는 비대칭 외판원문제(ATSP ; asymmetric traveling salesman problem)의 일종이므로, 만약 SOP의 선행제약조건을 완화하면 이 문제는 해밀턴경로조건을 갖는 최소비용유량문제로 모형화할 수 있고, OKA를 적용할 수 있다. 최근에 개발된 ATSP를 위한 OKA[11]는 수리계획법의 primal-dual, 최적화이론을 기반으로 ATSP에 대해 효율적으로 최적해 또는 최적해에 근사한 해를 구한다. 선행제약조건이 완화된 SOP에 OKA를 적용하면 해법의 매 단계(iteration)에서 복수 개의 부분순환로(subtour)가 발생하는데, 본 논문은 이렇게 발생된 복수 개의 부분순환로를 이용하여 SOP의 가능해(feasible solution)를 만드는 두 가지 방법을 소개한다. 이 두 방법은 모두 부분순환로를 구성하는 호들을 최대한 이용하여 선행제약조건을 만족하는 해밀턴경로를 구성한다. 여기서 부분순환로를 구성하는 호들을 최대한 이용하는 이유는 이 호들이 좋은 호들이기 때문이다. 본 논문에서 제시하는 방법 중 가장 좋은 해를 구하는 방법은 부분순환로를 이용하여 해밀턴경로를 구성하는 과정에서 호의 상태를 개선한다. 또한 본 논문에서 제시하는 해법은 두 방법에 의해 구성된 해를 개선하기 위해 해법의 매 단계에서 k-opt[12]를 수행하며, 전역탐색(global search)을 위해 비용환화법(cost relaxation)[13]을 수행한다.

실험에서는 두 가지 방법을 통한 여섯 가지의 경우들에 대해 실험결과를 비교분석하고, 다른 발견적 해법들과 해를 비교한다. 실험결과, 본 논문에서 제시한 방법 중 호의 상태를 개선하며 가능해를 구성해 가고, 3-opt와 4-opt 모두를 이용하는 방법은 34개의 TSPLIB문제 중 32개에서 지금까지 알려진 가장 좋은 해를 구하였다. 그리고 지금까지 알려진 4개의 가장 좋은 해법들과 해를 비교한 결과,

제안하는 해법이 가장 좋은 해를 가장 많이 구하였다. 본 논문의 구성은 다음과 같다. 제 2장에서는 선행제약조건을 완화한 SOP를 해밀턴경로조건을 갖는 최소비용유량문제로 모형화하고 OKA를 적용하는 방법을 설명한다. 제 3장에서는 복수 개의 부분순환로를 이용하여 선행제약조건을 만족시키는 해밀턴경로를 구하는 두 가지 방법을 설명하고, 제 4장에서 이 두 방법에 대해 간단한 예를 든다. 제 5장에서는 k-opt를 적용하는 방법과 비용완화법을 적용하는 방법을 설명한다. 그리고 제 6장에서는 실험을 통해 결과를 비교분석하며, 제 7장에서 결론을 맺는다.

2. Out-of-Kilter를 이용한 선행순서결정문제(SOP)

OKA는 primal-dual 이론인 최적화조건을 이용하여 최소비용유량문제에 대해 최적해를 찾는 해법이다[9, 10]. SOP의 선행제약조건을 완화하면 해밀턴경로조건을 갖는 최소비용유량문제로 모형화 할 수 있다. Kwon[11]은 최근에 OKA를 ATSP에 성공적으로 적용하였기 때문에 우리는 쉽게 OKA를 선행제약조건이 완화된 SOP에 적용할 수 있다. Problem A는 선행제약조건이 완화된 SOP문제를 수리적으로 최소비용유량문제로 모형화 한 것이다.

Problem A :

$$\text{Minimize} \quad \sum_{(i,j)\in A} c_{ij} f_{ij} \tag{1}$$

$$\text{Subject to} \quad \sum_{j\in V} f_{ji} = 1 \quad \text{for all } i \in V \tag{2}$$

$$\sum_{j\in V} f_{ij} = 1 \quad \text{for all } i \in V \tag{3}$$

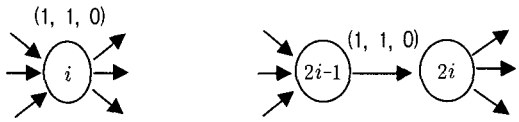
$$f_{ij} \in \{0,1\} \quad \text{for all } (i,j) \in A \tag{4}$$

Hamiltonian path starting from 1

terminating to n (5)

이 모형에서 f_{ij}는 호 (i, j)에 흐르는 유량이다. 제약조건 (2)와 제약조건 (3)은 유량 1이 각 교점에 반드시 들어오고 나가야 한다는 조건이다. 제약조건 (5)는 교점 1에서 시작하고 교점 n에서 끝나는 해밀턴경로조건이다.

OKA를 Problem A에 적용하기 위해 해밀턴경로조건을 완화하고 네트워크를 변경한 후 최적조건을 찾는다. 네트워크는 $G(V, A)$에서 $G'(V', A')$로 변경한다. 즉 [그림 1]과 같이 한 교점을 두 개의 교점과 이들을 연결하는 호로 변경한다.

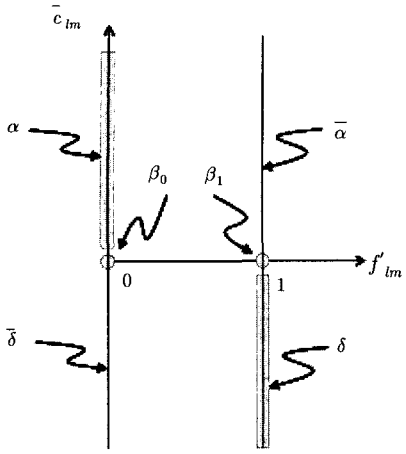


[그림 1] 한 교점을 두 개의 교점과 이를 연결하는 호로 변경

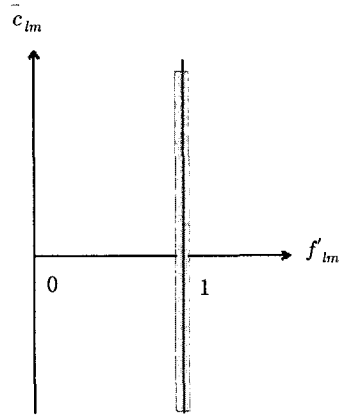
그러면 교점은 짝수교점과 홀수교점으로 구분할 수 있고, 호는 내부(inner) 호, 즉 교점 변경 시 발생하는 호와 외부(outer) 호로 구분할 수 있다. $G'(V', A')$에서 호의 비용은 c'_{lm}, 유량을 f'_{lm}이라고 정의하자. 그리고 쌍대변수를 $\Pi = (\pi_1, \pi_2, \cdots, \pi_{2n})$라고 정의하고 $\bar{c}_{lm} = \pi_l + c'_{lm} - \pi_m$이면 최적해를 구하기 위한 필요충분조건인 상보잉여조건(complementary slackness condition)[10, 12, 13]은 다음과 같다.

1. If $\bar{c}_{lm} > 0$, then $f'_{lm} = 0$
2. If $\bar{c}_{lm} < 0$, then $f'_{lm} = 1$
3. If $\bar{c}_{lm} = 0$, then $f'_{lm} = 0$ or 1

만약 호 (l, m)이 조건 1, 2, 3을 만족시키면 이 호는 *in-kilter*호라고 하며, 그렇지 않으면 *out-of-kilter*호라고 한다. 모든 호들이 *in-kilter*호이면 그 때의 해가 최적해이다. 호의 kilter상태를 그림으로 표현한 것을 kilter도라고 한다. [그림 2]는 네트워크 $G'(V', A')$에서 호에 대한 kilter도이다[10].



(a) 외부 호에 대한 Kilter 도



(b) 내부 호에 대한 Kilter 도

[그림 2] Kilter 도

[그림 2]에서 호가 회색의 두꺼운 실선 또는 점에 위치하면 *in-kilter*상태이고 그렇지 않으면 *out-of-kilter*상태이다. K_{lm}이 호(l, m)에 대한 kilter상태를 나타낸다면, kilter상태를 다음과 같이 여섯 가지로 구분할 수 있다[11].

$$K_{lm} = \begin{cases} \alpha & \text{if} \quad \overline{c_{lm}} > 0 \quad \text{and} \quad f_{lm} = 0 \\ \bar{\alpha} & \text{if} \quad \overline{c_{lm}} > 0 \quad \text{and} \quad f_{lm} = 1 \\ \delta & \text{if} \quad \overline{c_{lm}} < 0 \quad \text{and} \quad f_{lm} = 1 \\ \bar{\delta} & \text{if} \quad \overline{c_{lm}} < 0 \quad \text{and} \quad f_{lm} = 0 \\ \beta_0 & \text{if} \quad \overline{c_{lm}} = 0 \quad \text{and} \quad f_{lm} = 0 \\ \beta_1 & \text{if} \quad \overline{c_{lm}} = 0 \quad \text{and} \quad f_{lm} = 1 \end{cases}$$

만약 호의 kilter상태가 α, δ, β_0 또는 β_1이면 이 호는 *in-kilter*호이고, $\bar{\alpha}$ 또는 $\bar{\delta}$이면 이 호는 *out-of-kilter*호이다. 내부 호들은 용량하한과 용량상한이 모두 1이므로 항상 *in-kilter*상태이다.

OKA는 모든 *out-of-kilter*호들을 하나씩 *in-kilter*호로 변화시켜가며 최적해를 찾는다. 하나의 *out-of-kilter*호를 *in-kilter*호로 변화시키기 위해 증량가능경로(flow-augmenting path)를 찾는데, 이 과정에서 유량을 변경하거나 쌍대변수 값인 교점가(node potential)를 변경한다. 또한 증량가능경로를 찾는 과정에서 다른 호들의 kilter상태를 악화시키지 않는다. 이렇게 하여 OKA를 $G'(V', A')$상에서 Problem A에 적용하면 매 단계(하나의 *out-of-kilter*호에 대해 증량가능경로를 찾고 유량을 변경하는 단계)에서 복수 개의 부분순환로를 구한다. 본 연구에서는 이러한 복수 개의 부분순환로를 구성하는 호들을 최대한 이용하여 선행제약조건을 만족하며 해밀턴경로를 구성하는, 즉 SOP의 가능해를 구하는 두 가지 방법을 제시한다.

3. 선행순서결정문제(SOP)의 가능해 구성방법

본 장에서는 OKA를 Problem A에 적용한 결과 구해지는 복수 개의 부분순환로를 이용하여 SOP의 가능해인 선행제약조건을 만족시키는 해밀턴경로를 구하는 두 가지 방법에 대해 설명한다. 일반적으로 부분순환로를 구성하는 호들은 좋은 호들이기 때문에 본 논문에서 제시하는 방법들은 이 호들을 최대한 이용하여 선행제약조건을 만족시키는 해밀턴경로를 구한다.

3.1 최소비용을 고려하는 방법

부분순환로를 구성하는 호들의 집합을 A_{sub}라고 정의한다. 그리고 t는 해밀턴경로의 순서이고, i_t는 교점 i가 해밀턴경로의 t번째 위치에 있다는 것을 나타낸다. 그러면 복수 개의 부분순환로를 선행제

약조건을 만족시키는 해밀턴경로로 만드는 과정에서 최소비용을 고려하는 절차는 다음과 같다.

1. Put node 1 on the position 1
2. For $t \leftarrow 1$ to $n-2$ do
3. If an arc, $(i_t, j) \in A_{sub}$ and j_{t+1}(put j on the position of $t+1$) satisfies the precedence constraints
4. Put j on the position of $t+1$
5. Go to 2
6. Otherwise, find an arc such that is $\min\{c_{i_t k} : (i_t, k), k \in V-j\}$ and k_{t+1}(put k on the position of $t+1$) satisfies the precedence constraints
7. Put k on the position of $t+1$
8. Go to 2
9. Put node n on the position n

line 3, 7에서 $t+1$위치에 교점 j 또는 k가 위치하는 것이 선행제약조건을 만족하는지를 알아보기 위한 계산량은 $O(n)$이다. 이를 효율적으로 수행하기 위해 선행교점(predecessor)과 후행교점(successor)에 대한 데이터를 알고리즘의 수행 전에 미리 저장한다.

최소비용을 고려하여 복수 개의 부분순환로를 선행제약조건을 만족시키는 해밀턴경로로 만드는 방법의 단점은 해밀턴경로를 구성해가는 과정에서 후반부로 갈수록 좋은 호를 해밀턴경로에 위치시킬 확률이 적다는 것이다. 최근 거리인접점(nearest neighbor) 해법이 후반부로 갈수록 선택할 수 있는 교점의 수가 한정되기 때문에 좋은 해를 구하지 못하듯이 이 방법 또한 A_{sub}에 속한 호를 선택하지 못할 경우 현재의 위치에서 최소비용의 호를 선택하기 때문에 후반부로 갈수록 선택할 수 있는 호들이 적어 나쁜 호가 해밀턴경로에 포함될 수 있다. 그러나 A_{sub}에 속한 많은 호들이 해밀턴경로를 구성하기 때문에 greedy 방법이라고 할 수 없으며 비교적 좋은 해를 구한다. 제 6장 실험에서 해밀턴경로를 구성하는 호들 중 A_{sub}에 속한 호들의 비율을 알아본다.

3.2 kilter상태를 고려하는 방법

이 방법은 호의 kilter상태를 고려하여 전체적으로 kilter상태를 향상시키며 선행제약조건을 만족시키는 해밀턴경로를 구성하는 방법이다. 이 방법의 절차는 다음과 같다.

1. Put node 1 on the position 1
2. For $t \leftarrow 1$ to $n-2$ do
3. If an arc, $(i_t, j) \in A_{sub}$ and j_{t+1} satisfies the precedence constraints
4. Put j on the position $t+1$
5. Go to 2
6. Otherwise, find an arc $\min\{\bar{c}_{i_t k} : (i_t, k), k \in V-j\}$ and k_{t+1} satisfies the precedence constraints
7. Put k on the position $t+1$
8. Go to 2
9. Put node n on the position n

3.1절의 절차와 다른 부분은 line 6이다. Line 6에서 3.1절과 같이 최소비용을 고려하는 것이 아니라 $\bar{c}_{lm}$이 최소가 되는 호를 선택한다. 일반적으로 $\bar{c}_{lm}$의 최소값은 음수가 되어서 f'_{lm}이 0인 호들 중 $\bar{c}_{lm}$이 최소인 호는 kilter상태가 $\bar{\delta}$이다. 이 호를 해밀턴경로에 포함시키면 유량이 0에서 1로 변경되므로 kilter상태가 $\bar{\delta}$에서 δ로 변경된다. 따라서 $\bar{c}_{lm}$이 최소인 호를 찾아 이 호를 해밀턴경로에 포함시킴으로써 *out-of-kilter*인 호를 *in-kilter*호로 변하게 하여 *out-of-kilter*호의 수를 줄일 수 있다. 이 방법 또한 제 6장 실험에서 해밀턴경로를 구성하는 호들 중 A_{sub}에 속한 호들의 비율을 알아본다.

4. 수치예제

이 장에서는 간단한 예를 통해 제 3장에서 설명

한 두 방법을 설명한다. [그림 3]은 TSPLIB[6]에서 문제의 크기가 가장 작은 esc07을 나타낸 그림이다. [그림 3(a), (b)]는 각각 이 문제의 비용행렬과 선행제약조건이며, 선행제약조건은 해법의 효율성을 높이기 위해 문제에서 선행교점과 후행교점을 취하여 테이블로 정리해 놓았다.

본 논문에서 제시하는 해법은 초기해를 구하기 위해 헝가리언(hungarian)해법을 사용한다. 헝가리언 해법은 배정문제(assignment problem)에서 최적해를 찾는 해법으로 [그림 3(a)]의 비용행렬에서 선행제약조건과 해밀턴경로조건을 완화한 후 헝가리언 해법을 적용하면 [그림 4]와 같이 복수 개의 최소비용을 갖는 부분순환로를 구성한다. 본 논문에서 제시하는 해법은 매 단계에서 [그림 4]와 같은 복수 개의 부분순환로를 구성하기 때문에 본 예제에서는 제 3장의 두 방법을 이용하여 SOP의 초기해를 구하는 방법만을 예로 설명한다.

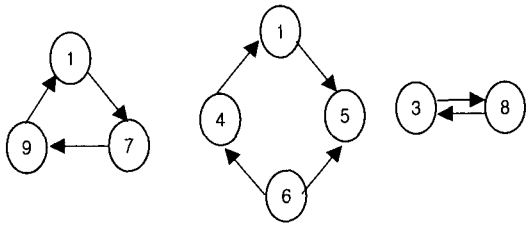
node #	1	2	3	4	5	6	7	8	9
1	0	0	0	0	0	0	0	0	10000
2	-1	0	100	200	75	0	300	100	0
3	-1	400	0	500	325	400	600	0	0
4	-1	700	800	0	550	700	900	800	0
5	-1	-1	250	225	0	275	525	250	0
6	-1	-1	100	200	-1	0	-1	-1	0
7	-1	-1	1100	1200	1075	1000	0	1100	0
8	-1	-1	0	500	325	400	600	0	0
9	-1	-1	-1	-1	-1	-1	-1	-1	0

(a) 비용행렬esc07

predecessor	successor
2	5, 6, 7, 8
5	6
7	6
8	6

(b) 선행제약조건

[그림 3] 예제(esc07)



[그림 4] 헝가리언 해법으로 구한 복수 개의 부분순환로

첫 번째로 최소비용을 고려하는 방법을 사용한다. 우선 교점 1을 $t=1$위치에 놓는다. A_{sub}에 속한 호인(1, 7)이 선행제약조건을 만족하는 가를 본다. 교점 2가 교점 7을 선행해야 하기 때문에 교점 1 다음에 교점 7이 오는 것은 선행제약조건을 만족시키지 못한다. 따라서 교점 1에서 출발하고 선행제약조건을 만족시키는 호들 중 최소비용의 호를 찾는다. 교점 1에서 출발하고 선행제약을 만족하는 호들은 (1, 2), (1, 3), (1, 4)이다. 이 세 호들의 비용은 모두 0이므로 임의로 호 (1, 2)를 선택하고, 교점 2를 $t=2$위치에 놓는다. 이제 호 (2, 5)가 A_{sub}에 속하기 때문에 교점 2 다음에 교점 5가 올 수 있는가를 본다. 교점 5가 교점 2 다음에 오는 것은 선행제약조건을 만족시키므로 교점 5를 $t=3$위치에 놓는다. 다음으로 호 (5, 6)이 A_{sub}에 속하기 때문에 교점 5 다음에 교점 6이 올 수 있는가를 본다. 선행제약조건을 만족시키지 못하므로 교점 5에서 출발하고 선행제약조건을 만족시키는 호들 중 최소비용의 호를 찾는다. 최소비용의 호는 (5, 4)이므로 $t=4$위치에 교점 4를 놓는다. 이렇게 계속 진행하여 $t=8$위치까지 교점을 채우고 $t=9$위치에는 교점 9를 놓는다. 최종적으로 구한 해밀턴경로는 1-2-5-4-3-8-7-6-9이며, 비용은 2700이다. 사실 이 문제의 최적해는 2125이므로 2700은 좋은 해라고 할 수 없으며, 해밀턴경로에 포함된 8개의 호들 중 A_{sub}에 속한 호는 2개이다.

두 번째로 *kilter*상태를 고려하는 방법을 사용한다. 초기 교점가는 해당 교점으로 들어오는 호의

비용, 즉 Π= (10000, 700, 0, 200, 75, 276, 0, 0, 0)이다. 우선 교점 1을 $t=1$위치에 놓는다. 교점 1 다음에 교점 7을 놓는 것은 선행제약조건을 만족시키지 못하므로 교점 1에서 출발하고 선행제약조건을 만족시키는 호들 중 $\bar{c}_{lm}$이 최소인 호를 찾는다. 호(1, 2), (1, 3), (1, 4) 중 호(1, 2)의 $\bar{c}_{12}=-200$이므로 최소값을 갖는다. 따라서 교점 1 다음에 교점 2를 위치시킨다. 그 다음 호(2, 5)는 선행제약조건을 만족시키므로 교점 2 다음에 교점 5를 위치시킨다. 교점 5 다음에 교점 3을 놓은 것은 선행제약조건을 만족시키지 못하므로 교점 5에서 출발하고 선행제약조건을 만족시키는 호들 중 $\bar{c}_{lm}$이 최소인 호를 찾는다. 호(5, 3), (5, 4), (5, 7), (5, 8) 중 호(5, 3)과 (5, 4)의 $\bar{c}_{53}=\bar{c}_{54}=25$이므로 최소값을 갖는다. 이 중 임의로 (5, 3)을 선택하여 교점 5 다음에 교점 3을 위치시킨다. 교점 3 다음에 교점 8을 놓는 것은 선행제약조건을 만족시키므로 교점 3 다음에 교점 8을 위치시킨다. 이렇게 하여 최종적으로 구한 해밀턴경로는 1-2-5-3-8-4-7-6-9이며, 비용은 2500이다. 해밀턴경로에 포함된 8개의 호들 중 A_{sub}에 속한 호는 2개이다.

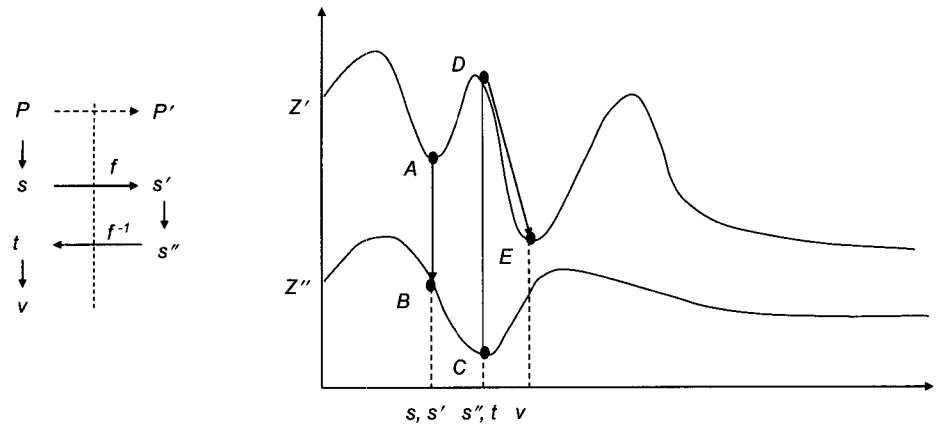
이 문제의 최적 해밀턴경로는 1-2-5-3-8-7-6-4-9이며, 비용은 2125이다. 또한 해밀턴경로에 포함된 8개의 호들 중 A_{sub}에 속한 호는 3개이다. 두 방법 모두 최적 해밀턴경로를 구성하지 못하였지만, 3-opt을 통해 3개의 호를 교환함으로써 간단히 최적해를 구할 수 있다. 또한 OKA를 해밀턴경로인 초기해에 계속 적용하면 매 단계에서 복수 개의 부분순환로를 구할 수 있는데, 그 때마다 위의 방법을 적용하여 해밀턴경로를 구성한다.

5. 해의 개선과 전역해 탐색

Problem A에 OKA를 적용하면 매 단계에서 [그림 4]와 같은 복수 개의 부분순환로가 발생한다. 이러한 복수 개의 부분순환로에 제 3장에서 기술한 방법을 적용하면 해법의 매 단계에서 SOP의 가능해인 해밀턴경로를 구성할 수 있다. 본 장에서는 해밀턴경로를 개선하는 방법과 국지해(local solution)에 빠졌을 때 국지해를 탈출하여 전역해(global solution)를 탐색하는 방법을 설명한다.

해밀턴경로를 개선하는 방법으로 k-opt을 사용한다. k-opt은 이웃해(neighborhood)를 탐색하기 위해 k개의 호를 교환하는 방법이다. 이 해법은 잘 알려진 국지탐색해법으로 계산량은 $O(n^k)$이며, k= 2 또는 3인 경우 효율적으로 좋은 해를 찾는다[12]. 비용행렬이 비대칭인 경우 2-opt를 적용할 수 없기 때문에 본 연구에서는 해법의 매 단계에서 복수 개의 부분순환로를 이용하여 구성한 해밀턴경로에 3-opt과 4-opt을 적용한다. 3-opt를 적용하여 해를 개선시키는 이웃해만을 선행제약조건을 만족하는가를 확인한다. 만약 선행제약조건을 만족하면 현재까지의 가장 좋은 해를 그 이웃해로 대체한다. 이 절차를 반복하여 더 이상 해를 개선시키는 이웃해가 나타나지 않거나, 해를 개선시키는 이웃해가 나타나더라도 선행제약조건을 만족시키는 이웃해가 없을 경우 3-opt를 종료한다. 3-opt가 종료되면 4-opt를 수행하는데 4-opt도 3-opt과 같은 절차로 수행한다. 이렇게 하여 구한 해는 해법의 매 단계에서 나타나는 해이다. 그러므로 해법이 종료될 때까지 매 단계에서 구해진 해가 현재까지의 가장 좋은 해보다 좋을 경우 현재까지의 가장 좋은 해를 더 좋은 해로 대체한다. 제 6장 실험에서는 k-opt의 효과를 알아보기 위해 k-opt를 사용하지 않는 경우, 3-opt만을 사용하는 경우, 3-opt와 4-opt 모두를 사용하는 경우로 나누어 실험한 후 그 결과를 비교분석한다.

본 연구에서는 국지해를 탈출하여 전역해를 탐색하기 위해 변형법(perturbation)[14] 중 최근 소개된 비용완화법(cost-relaxation)[13]을 이용한다. 비용완화법은 최근에 소개된 방법으로 외판원문제에 매우 효과적으로 적용되었다. 원 문제를 P, 변형된 문제를 P'라고 하자. P'는 P의 일부 호들의 비용을 완화(일부 호들의 비용을 0으로 감소시킴)한 문제이다. 따라서 같은 해밀턴경로라도 경로의



(a) (b)

[그림 5] 비용완화법의 개념도

전체 비용은 P'에서의 해가 P에서의 해보다 작거나 같다.

[그림 5]는 비용완화법을 사용하여 국지해를 탈출하고 전역해를 탐색하는 개념도이다. [그림 5(a)]는 비용완화법의 전체 개념도를 보여주며 [그림 5(b)]는 비용완화법에서 목적 값의 변화를 보여준다. [그림 5]를 통해 비용완화법의 단계를 간단히 설명한다. 해법을 수행하면 국지해, s([그림 5]에서 점 A)를 구한다. 일부 호들의 비용을 완화하여 P'를 만들면 P'에서의 s는 s'([그림 5(b)]에서 점 B)가 된다. 다시 말해, 해밀턴 경로 s와 s'는 같지만 목적 값이 다르게 된다. 변형된 문제 P'에서의 s'는 국지해가 아니므로 s'를 초기해로 P'에서의 국지해를 구하면 s''([그림 5(b)]에서 점 C)가 된다. 이제 원 문제 P에서 s''는 t([그림 5(b)]에서 점 D)가 된다. 다시 말해, 해밀턴 경로 s''와 t는 같지만 목적 값은 다르다. P에서의 t는 국지해가 아니므로 t를 초기해로 P에서 국지해를 구하면 v([그림 5(b)]에서 점 E)가 된다.

이러한 비용완화법을 여러 번 반복하면 더 좋은 해를 찾을 수 있지만 많은 수행시간으로 인하여 일정 수만큼만 반복한다. 비용완화법을 한 번 수행할 때마다 국지해법을 두 번 수행해야한다. 비용완화법을 수행하는 과정인 [그림 5(a)]에서 v가 s보다 좋으면 해를 대체하여 항상 가장 좋은 해를 유지한다.

6. 실험결과

실험을 위해 500M 메모리, 2.2GHz CPU가 장착된 PC를 이용하였으며, 구현을 위한 언어와 툴은 각각 C++, Microsoft Visual Studio 6.0을 사용하였다. 실험계획은 제 3장에서 제시한 두 가지 방법과, k-opt를 사용하지 않는 경우, 3-opt만을 사용하는 경우, 3-opt와 4-opt 모두를 사용하는 경우로 나누어 각각의 조합인 여섯 가지의 경우에 대해 실험을 수행하였다. 실험에서 비용완화법은 한 문제에 대해 네 번을 수행하여 다섯 개의 국지해를 구한 후 이 중 가장 좋은 해와 가장 좋은 해가 나타난 회수, 그리고 해의 평균을 비교대상으로 하였다. 실험문제로는 TSPLIB[6]의 34개 문제를 대상으로 실험을 수행하였다. 여섯 가지의 경우에 대한 실험을 위해 우선 문제들 중 문제의 크기가 70개 이하인 16개의 문제에 대해 실험을 수행하여 가장 좋은 경우의 조합을 찾고, 가장 좋은 경우의 조합으로 전체 문제에 대해 실험을 수행하였다. 그리고 마지막에 지금까지 알려진 가장 좋은 해법들과 해를 비교하였다.

<표 1>~<표 3>은 최소비용을 고려하는 방법으로 실험을 수행한 결과이다. <표 1>은 k-opt을 사용하지 않고 실험한 결과로써 가장 좋은 해(BNS : best known solution)를 찾은 경우는 16개의 문제들 중 3개에 불과하다. 그런 반면 k-opt를 수행하지 않기 때문에 평균 수행시간은 모두 1초 이내로 빠르게 나타났다.

<표 2>는 3-opt만을 사용한 경우이다. <표 2>에서는 한 개의 문제를 제외한 모든 문제에서 가장 좋은 해를 찾았으며, 다섯 번의 국지해 탐색 중 가장 좋은 해를 찾은 회수의 평균값은 3.13으로 나타났다. 수행시간은 문제의 크기가 커짐에 따라, 선행제약조건이 많음에 따라 더 오래 걸리는 것으로 나타났다. 해의 측면에서 보면 k-opt를 전혀 사용하지 않는 방법에 비해 3-opt을 사용함으로써 해가 많이 개선되었는데, 수치로 보면 다섯 번의 수행에서 가장 좋은 해를 찾은 평균회수는 0.69에서 3.13으로 좋아졌으며, 평균해가 가장 좋은 해에서 벗어난 정도는 평균적으로 6.88에서 0.63으로 좋아졌다. 반면, ft53.2, ft53.3, ry48p.3, 그리고 ry48p.4의 경우 수행시간은 3-opt을 사용한 실험에서 월등히 오래 걸린 것으로 나타났는데, 이는 제안하는 해법의 매 단계에서 3-opt를 수행하기 때문이며, 수행시간에서 k-opt의 비중이 큼을 알 수 있다. 또한 문제의 크기가 같은데도 불구하고 수행시간이 차이가 나는 것은 선행제약조건이 많은 경우 부분순환로를 제거하여 SOP의 가능해를 만드는데 수행시간이 많이 걸리기 때문이다.

<표 3>은 3-opt와 4-opt를 모두 사용한 경우 실험결과이다. <표 3>에서 해는 3-opt만을 사용한 경우에 비해 평균적으로 조금 좋아졌지만 수행시간은 조금 나빠진 것을 알 수 있다. 다섯 번의 수행에서 가장 좋은 해를 찾은 평균회수는 3.25로 <표 2>의 3.13에 비해 조금 좋아졌지만, 가장 좋은 해를 찾지 못한 문제가 3개나 된다. 그리고 <표 1>~<표 3>에서 해밀턴경로에 포함된 호들 중 A_{sub}에 속한 호들의 평균 비율은 각각 66.21%, 60.41%, 60.95%로 나타나 A_{sub}에 속한 많은 호들이 해밀턴

〈표 1〉 최소비용을 고려하는 방법을 사용한 결과(k-opt를 사용하지 않는 경우)

Instance	Size	Best known solution (BNS)	# of finding BNS	Best solution (% above BNS)	Ave. solution (% above BNS)	Ave. running time	% of A_{sub} arcs of feasible Hamiltonian path
esc07	9	2125	5	2125(0.00)	2125(0.00)	0.00	44.4
esc12	14	1675	0	1685(0.60)	1735(3.58)	0.00	59.9
esc25	27	1681	0	1757(4.52)	1812(7.79)	0.00	68.4
esc47	49	1288	0	1362(5.75)	1678(30.28)	0.04	65.8
esc63	65	62	5	62(0.00)	62(0.00)	0.02	77.1
br17.10	18	55	0	58(5.45)	58(5.45)	0.00	59.3
ft53.1	54	7531	0	7777(3.27)	7904(4.95)	0.15	76.8
ft53.2	54	8026	0	8410(4.78)	8624(7.45)	0.32	68.5
ft53.3	54	10262	0	10812(5.36)	12037(17.30)	0.70	60.5
ft53.4	54	14425	0	14969(3.77)	15186(5.28)	0.50	70.0
ry48p.1	49	15805	0	16656(5.38)	16986(7.47)	0.15	75.1
ry48p.2	49	16666	0	17108(2.65)	17632(5.80)	0.14	74.2
ry48p.3	49	19894	0	20777(4.44)	21467(7.91)	0.34	64.2
ry48p.4	49	31446	0	32038(1.88)	32513(3.39)	0.53	58.0
rbg048a	50	351	1	351(0.00)	358(1.99)	0.04	67.4
rbg050a	52	467	0	468(0.21)	474(1.50)	0.09	69.7
Average			0.69	(3.00)	(6.88)		66.21

〈표 2〉 최소비용을 고려하는 방법을 사용한 결과(3-opt만 사용하는 경우)

Instance	Size	Best known solution(BNS)	# of finding BNS	Best solution (% above BNS)	Ave. solution (% above BNS)	Ave. running time	% of A_{sub} arcs of feasible Hamiltonian path
esc07	9	2125	5	2125(0.00)	2125(0.00)	0.00	46.5
esc12	14	1675	5	1675(0.00)	1675(0.00)	0.00	60.3
esc25	27	1681	1	1681(0.00)	1684(0.18)	0.00	65.2
esc47	49	1288	0	1311(1.79)	1388(7.76)	0.43	63.9
esc63	65	62	5	62(0.00)	62(0.00)	0.02	76.8
br17.10	18	55	5	55(0.00)	55(0.00)	0.00	59.2
ft53.1	54	7531	2	7531(0.00)	7553(0.29)	0.78	73.5
ft53.2	54	8026	1	8026(0.00)	8059(0.41)	6.45	62.7
ft53.3	54	10262	5	10262(0.00)	10262(0.00)	28.25	43.1
ft53.4	54	14425	5	14425(0.00)	14425(0.00)	0.70	55.1
ry48p.1	49	15805	4	15805(0.00)	15811(0.04)	0.61	71.7
ry48p.2	49	16666	2	16666(0.00)	16744(0.47)	1.82	61.7
ry48p.3	49	19894	3	19894(0.00)	20017(0.62)	21.45	50.8
ry48p.4	49	31446	1	31446(0.00)	31489(0.14)	54.82	49.4
rbg048a	50	351	5	351(0.00)	351(0.00)	0.06	65.8
rbg050a	52	467	1	467(0.00)	468(0.21)	3.26	60.9
Average			3.13	(0.11)	(0.63)		60.41

〈표 3〉 최소비용을 고려하는 방법을 사용한 결과(3-opt와 4-opt 모두를 사용하는 경우)

Instance	Size	Best known solution(BNS)	# of finding BNS	Best solution (% above BNS)	Ave. solution (% above BNS)	Ave. running time	% of A_{sub} arcs of feasible Hamiltonian path
esc07	9	2125	5	2125(0.00)	2125(0.00)	0.00	46.5
esc12	14	1675	5	1675(0.00)	1675(0.00)	0.00	60.7
esc25	27	1681	0	1684(0.18)	1684(0.18)	0.00	66.6
esc47	49	1288	0	1332(3.42)	1362(5.75)	1.15	64.8
esc63	65	62	5	62(0.00)	62(0.00)	0.04	77.7
br17.10	18	55	5	55(0.00)	55(0.00)	0.03	59.0
ft53.1	54	7531	3	7531(0.00)	7545(0.19)	3.70	71.3
ft53.2	54	8026	2	8026(0.00)	8053(0.34)	5.26	61.5
ft53.3	54	10262	5	10262(0.00)	10262(0.00)	32.56	43.4
ft53.4	54	14425	5	14425(0.00)	14425(0.00)	0.15	53.3
ry48p.1	49	15805	2	15805(0.00)	15826(0.13)	1.25	72.3
ry48p.2	49	16666	4	16666(0.00)	16695(0.17)	3.27	63.3
ry48p.3	49	19894	0	20050(0.78)	20117(1.12)	34.06	51.9
ry48p.4	49	31446	4	31446(0.00)	31446(0.00)	159.51	50.3
rbg048a	50	351	5	351(0.00)	351(0.00)	0.67	67.0
rbg050a	52	467	2	467(0.00)	468(0.21)	0.72	65.6
Average			3.25	(0.27)	(0.50)		60.95

〈표 4〉 kilter상태를 고려하는 방법을 사용한 결과(k-opt를 사용하지 않는 경우)

Instance	Size	Best known solution(BNS)	# of finding BNS	Best solution (% above BNS)	Ave. solution (% above BNS)	Ave. running time	% of A_{sub} arcs of feasible Hamiltonian path
esc07	9	2125	5	2125(0.00)	2125(0.00)	0.00	46.7
esc12	14	1675	0	1685(0.60)	1737(3.70)	0.00	62.8
esc25	27	1681	0	1684(0.18)	1807(7.50)	0.00	64.1
esc47	49	1288	0	1433(11.26)	1537(19.33)	0.06	67.1
esc63	65	62	5	62(0.00)	62(0.00)	0.14	74.9
br17.10	18	55	5	55(0.00)	55(0.00)	0.01	59.1
ft53.1	54	7531	0	7846(4.18)	8312(10.37)	0.18	76.1
ft53.2	54	8026	0	8812(9.79)	9103(13.42)	0.46	64.6
ft53.3	54	10262	0	11185(8.99)	11914(16.10)	0.86	51.3
ft53.4	54	14425	0	14814(2.70)	15302(6.08)	0.57	58.9
ry48p.1	49	15805	0	16790(6.23)	17614(11.45)	0.29	72.7
ry48p.2	49	16666	0	17341(4.05)	18182(9.10)	0.25	68.0
ry48p.3	49	19894	0	21840(9.78)	22618(13.69)	0.56	55.1
ry48p.4	49	31446	0	32408(3.06)	32968(4.84)	0.62	54.6
rbg048a	50	351	1	351(0.00)	354(0.85)	0.06	73.4
rbg050a	52	467	0	475(1.71)	483(3.43)	0.18	66.3
Average			1.00	(3.91)	(7.49)		63.48

〈표 5〉 kilter상태를 고려하는 방법을 사용한 결과(3-opt만 사용하는 경우)

Instance	Size	Best known solution(BNS)	# of finding BNS	Best solution (% above BNS)	Ave. solution (% above BNS)	Ave. running time	% of A_{sub} arcs of feasible Hamiltonian path
esc07	9	2125	5	2125(0.00)	2125(0.00)	0.00	47.8
esc12	14	1675	5	1675(0.00)	1675(0.00)	0.00	60.8
esc25	27	1681	5	1681(0.00)	1681(0.00)	0.00	64.8
esc47	49	1288	0	1306(1.40)	1394(8.23)	0.20	67.1
esc63	65	62	5	62(0.00)	62(0.00)	0.00	74.3
br17.10	18	55	5	55(0.00)	55(0.00)	0.00	54.5
ft53.1	54	7531	2	7531(0.00)	7559(0.37)	1.25	73.1
ft53.2	54	8026	2	8026(0.00)	8046(0.25)	5.53	61.7
ft53.3	54	10262	5	10262(0.00)	10262(0.00)	32.62	40.7
ft53.4	54	14425	5	14425(0.00)	14425(0.00)	5.12	55.9
ry48p.1	49	15805	3	15805(0.00)	15908(0.65)	1.78	70.1
ry48p.2	49	16666	5	16666(0.00)	16666(0.00)	6.56	59.7
ry48p.3	49	19894	3	19894(0.00)	19919(0.13)	33.85	47.8
ry48p.4	49	31446	5	31446(0.00)	31446(0.00)	37.35	49.4
rbg048a	50	351	5	351(0.00)	351(0.00)	0.26	68.3
rbg050a	52	467	1	467(0.00)	468(0.21)	2.34	60.2
Average			3.81	(0.08)	(0.62)		59.76

〈표 6〉 kilter상태를 고려하는 방법을 사용한 결과(3-opt와 4-opt 모두를 사용하는 경우)

Instance	Size	Best known solution(BNS)	# of finding BNS	Best solution (% above BNS)	Ave. solution (% above BNS)	Ave. running time	% of A_{sub} arcs of feasible Hamiltonian path
esc07	9	2125	5	2125(0.00)	2125(0.00)	0.00	47.8
esc12	14	1675	5	1675(0.00)	1675(0.00)	0.00	61.8
esc25	27	1681	5	1681(0.00)	1681(0.00)	0.00	61.8
esc47	49	1288	2	1288(0.00)	1324(2.80)	0.71	62.6
esc63	65	62	5	62(0.00)	62(0.00)	0.03	75.4
br17.10	18	55	5	55(0.00)	55(0.00)	0.00	54.5
ft53.1	54	7531	5	7531(0.00)	7531(0.00)	3.57	72.6
ft53.2	54	8026	2	8026(0.00)	8047(0.26)	8.22	62.4
ft53.3	54	10262	5	10262(0.00)	10262(0.00)	36.06	41.4
ft53.4	54	14425	5	14425(0.00)	14425(0.00)	12.37	53.7
ry48p.1	49	15805	3	15805(0.00)	15867(0.39)	1.48	69.8
ry48p.2	49	16666	5	16666(0.00)	16666(0.00)	4.26	60.7
ry48p.3	49	19894	4	19894(0.00)	19894(0.00)	36.76	45.8
ry48p.4	49	31446	5	31446(0.00)	31446(0.00)	75.93	44.9
rbg048a	50	351	5	351(0.00)	351(0.00)	0.67	68.4
rbg050a	52	467	4	467(0.00)	467(0.00)	1.02	62.2
Average			4.37	(0.00)	(0.21)		59.11

경로를 구성하는 것을 알 수 있다.

<표 4>~<표 6>은 kilter상태를 고려하는 방법으로 실험을 수행한 결과이다. <표 4>는 k-opt를 사용하지 않고 실험한 결과로 <표 1>의 결과와 비슷하게 나타났다. <표 5>는 3-opt을 사용한 경우로 k-opt를 사용하지 않은 경우에 비해 최소비용을 고려하는 방법에서와 같이 해는 많이 개선된 것을 알 수 있다. <표 5>의 실험결과를 <표 2>의 실험결과와 비교하면 다섯 번의 수행에서 가장 좋은 해를 구한 평균회수는 각각 3.81과 3.13으로 kilter상태를 고려하는 방법이 좀 더 우수하게 나타났다. 그리고 평균해가 가장 좋은 해에서 벗어난 정도는 평균적으로 각각 0.62와 0.63으로 비슷하게 나타났다. 수행시간에서는 큰 차이를 보이지 않았다. <표 6>은 3-opt과 4-opt 모두를 사용한 경우의 실험결과로써 모든 문제에서 가장 좋은 해를 찾았으며, 가장 좋은 해를 구한 평균회수는 4.37로 나타나 이 경우의 조합이 가장 좋은 해를 가장 많이 구하는 방법으로 나타났다. 또한 평균해가 가장 좋은 해에서 벗어난 정도는 평균적으로 0.21로 나타나, <표 3>의 최소비용을 고려하는 방법의 결과인 0.5에 비해 좋은 결과를 나타냈다. 수행시간은 <표 3>의 최소비용을 고려하는 방법과 큰 차이를 보이지 않았다. 그리고 <표 4>~<표 6>에서 해밀턴경로에 포함된 호들 중 A_{sub}에 속한 호들의 평균 비율은 각각 63.48%, 59.76%, 59.11%로 나타나, 최소비용을 고려하는 방법과 비슷하였다.

<표 7>은 가장 좋은 결과를 나타낸 경우의 조합, 즉 kilter상태를 고려하는 방법을 수행하고, 3-opt와 4-opt 모두를 사용하는 방법으로 TSPLIB의 34개 문제에 대해 실험한 결과이다. 실험에서 두 개의 문제를 제외한 모든 문제에서 가장 좋은 해를 구했으며, 수행시간은 문제의 크기가 커짐에 따라 커지고, 문제의 크기가 같더라도 선행제약조건이 많은 문제에서 많은 시간이 소요되었다.

<표 8>은 지금까지 가장 좋은 해법들과 본 연구에서 제시한 해법의 실험결과를 비교한 것이다. <표 8>에서 진하게 표시한 숫자는 가장 좋은 해가

〈표 7〉 kilter상태를 고려하는 방법을 사용한 결과(3-opt와 4-opt 모두를 사용하는 경우)

Instance	Size	TSPLIB bounds	Best known solution	# of finding best known solution	Best solution	Ave. solution	Ave. running time
esc07	9	2125	2125	5	2125	2125	0.00
esc12	14	1675	1675	5	1675	1675	0.00
esc25	27	1681	1681	5	1681	1681	0.00
esc47	49	1288	1288	2	1288	1324	0.71
esc63	65	62	62	5	62	62	0.03
esc78	80	18230	18230	5	18230	18230	62.25
br17.10	18	55	55	5	55	55	0.00
br17.12	18	55	55	5	55	55	0.00
ft53.1	54	[7438, 7570]	7531	5	7531	7531	3.57
ft53.2	54	[7630, 8335]	8026	2	8026	8047	8.22
ft53.3	54	[9473, 10935]	10262	5	10262	10262	36.06
ft53.4	54	14425	14425	5	14425	14425	12.37
ft70.1	71	39313	39313	5	39313	39313	4.18
ft70.2	71	[39739, 40422]	40419	2	40419	40470	22.18
ft70.3	71	[41305, 42535]	42535	4	42535	42541	62.20
ft70.4	71	[52269, 53562]	53530	1	53530	53572	661.33
kro124p.1	101	[37722, 40186]	39420	1	39420	39482	87.92
kro124p.2	101	[38534, 41677]	41336	2	41336	41370	117.10
kro124p.3	101	[40967, 50876]	49499	2	49499	49992	183.87
kro124p.4	101	[64858, 76103]	76103	4	76103	76108	1665.28
ry48p.1	49	[15220, 15805]	15805	3	15805	15867	1.48
ry48p.2	49	[15524, 16666]	16666	5	16666	16666	4.26
ry48p.3	49	[18156, 19894]	19894	3	19894	19894	36.76
ry48p.4	49	[29967, 31446]	31446	4	31446	31446	75.93
rbg048a	50	351	351	5	351	351	0.67
rbg050a	52	467	467	4	467	467	1.02
rbg109a	111	1038	1038	5	1038	1038	28.13
rbg150a	152	[1748, 1750]	1750	3	1750	1751	85.69
rbg174a	176	2033	2033	3	2033	2034	165.62
rbg253a	255	[2928, 2987]	2950	5	2950	2950	763.30
rbg323a	325	[3136, 3157]	3140	1	3140	3151	1815.98
rbg341a	343	[2543, 2597]	2568	1	2568	2585	10021.23
rbg358a	360	[2518, 2599]	2545	0	2556	2563	14534.97
rbg378a	380	[2761, 2833]	2816	0	2825	2833	22705.65

아님을 나타낸다. HAS-SOP[7]와 MPO/AI[4]는 모두 제안하는 해법과 같이 다섯 번 시도한 실험결과이다. 22개의 문제들 중 HAS-SOP와 MPO/AI는 모두 네 개의 문제에서 가장 좋은 해를 찾지 못했고, 제안하는 해법은 두 개의 문제에서 가장 좋은 해를 찾지 못했다. Hybrid 해법[5]은 17개의 문제들에서 모두 가장 좋은 해를 찾았는데, 이는 최대 1000번을 시도한 실험결과이다. 그리고 Cooperative parallel rollout 해법[8]은 12개의 문제들 중 세 개의 문제에서 가장 좋은 해를 찾지 못하였다.

〈표 8〉 지금까지 알려진 가장 좋은 네 개의 해법들과 제안하는 해법의 해 비교

Instance	Best Known Solution	HAS-SOP	MPO/AI	Hybrid	Cooperative Parallel Rollout	Proposed
esc63	62	62	62	-	-	62
esc78	18230	18230	18230	18230	-	18230
ft53.1	7531	7531	7531	-	7531	7531
ft53.2	8026	8026	8026	-	8026	8026
ft53.3	10262	10262	10262	-	10310	10262
ft53.4	14425	14425	14425	-	14425	14425
ft70.1	39313	39313	39313	39313	39313	39313
ft70.2	40419	40419	40419	40419	40422	40419
ft70.3	42535	42535	42535	42535	42566	42535
ft70.4	53530	53530	53530	53530	53530	53530
kro124p.1	39420	39420	39420	39420	39420	39420
kro124p.2	41336	41336	41336	41336	41336	41336
kro124p.3	49499	49499	49519	49499	49499	49499
kro124p.4	76103	76103	76103	76103	76103	76103
rbg109a	1038	1038	1038	1038	-	1038
rbg150a	1750	1750	1750	1750	-	1750
rbg174a	2033	2033	2033	2033	-	2033
rbg253a	2950	2950	2950	2950	-	2950
rbg323a	3140	3141	3141	3140	-	3140
rbg341a	2568	2576	2572	2568	-	2568
rbg358a	2545	2549	2555	2545	-	2549
rbg378a	2816	2817	2816	2816	-	2817

7. 결 론

본 논문에서는 선행순서결정문제(SOP)에 대해 선행제약조건을 완화하고 해밀턴경로조건을 갖는 최소비용유량문제로 모형화하여 Out-of-Kilter 해법을 적용하였다. 이 과정에서 매 단계에서 복수 개의 부분순환로가 발생하는데 이들을 이용하여 선행제약조건과 해밀턴경로조건을 만족시키며 SOP의 가능해를 구하는 두 가지 방법을 제시하였다. 그리고 해법의 매 단계에서 구한 해밀턴경로에 k-opt를 적용하여 해를 구하였으며, 국지해에 빠질 경우 비용완화법을 이용하여 효과적으로 국지해를 탈출하여 전역해를 탐색하였다. 실험결과, kilter상태를 고려하고 3-opt와 4-opt 모두를 사용하는 방법이 가장 우수한 실험결과를 나타냈다. 이 실험에서 해밀턴경로에 포함된 호들 중 복수 개의 부분순환로에 속한 호들의 평균 비율은 60% 이상으로 나타나 좋은 호들이 해밀턴경로에 많이 포함되는 것을 알 수 있었다. 또한 문제의 크기가 같더라도 선행제약조건이 많아짐에 따라 해밀턴경로에 포함된 호들 중 복수 개의 부분순환로에 속한 호들의 평균 비율은 줄어들고, 수행시간은 더 오래 걸리는 것으로 나타났다. 가장 우수한 방법을 사용하여 34개의 TSPLIB 문제에 적용한 결과 두 개의 문제를 제외하고 모두 가장 좋은 해를 구하였고, 제안하는 해법이 지금까지 알려진 가장 좋은 해법들 중 하나임을 검증하였다.

참 고 문 헌

[1] Ascheuer, N., L.F. Escudero, M. Grotschel, and M. Stoer, "A cutting plane approach to the sequential ordering problem (with applications to job scheduling in manufacturing)," *SIAM Journal on Optimization*, Vol.3(1993), pp.25-42.

[2] Escudero, L.F., M. Guignard, and K. Malik, "A Lagrangian relax-and-cut approach for the sequential ordering problem with precedence relationships," *Annals of Operations Research*, Vol.50(1994), pp.219-239.

[3] Ascheuer, N., M. Junger, and G. Reinelt, "A branch & cut algorithm for the asymmetric traveling salesman problem with precedence constraints," *Computational Optimization and Applications*, Vol.17(2000), pp.61-84.

[4] Chen, S. and S.Smith, "Commonality and genetic algorithms," Technical Report CMU-RI-TR-96-17(1996), *The Robotic Institute*, Carnegie Mellon University, PA.

[5] Seo, D.I. and B.R. Moon, "A hybrid genetic algorithm based on complete graph representation for the sequential ordering problem," *GECCO 2003*, LNCS 2723(2003), pp.669-680.

[6] Reinelt, G., "TSPLIB : A Traveling Salesman Problem Library," *ORSA Journal on Computing*, Vol.3(1991), pp.376-384.

[7] Gambardella, L.M. and M. Dorigo, "An ant colony system hybridized with a new local search for the sequential ordering problem," *INFORMS Journal on Computing*, Vol.12(2000), pp.237-255.

[8] Guerriero, F. and M. Mancini, "A cooperative parallel rollout algorithm for the sequential ordering problem," *Parallel Computing*, Vol.29(2003), pp.663-677.

[9] Ahuja, R.K., T.L. Magnanti, and J.B. Orlin, *Network Flows*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1993.

[10] Ball, M.O., T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, *Network Models*, Elsevier Science B.V., Amsterdam, 1995.

[11] Kwon, S.H., Y.G.G, M.K. Kang, "Application of the Out-of-Kilter Algorithm to the Asymmetric Traveling Salesman Problem," *Journal of the Operational Research Society*, Vol.54(2003), pp.1085-1092.

[12] Lin, S., "Computer Solution of the Traveling Salesman Problem", *Bell System Technology Journal*, Vol.44(1965), pp.2245-2268.

[13] Kwon, S.H., "Perturbation using out-of-kilter arc of the asymmetric traveling salesman problem", *Journal of Korean Operations Research and Management Science Society*, Vol.30, No.2(2005), pp.157-167.

[14] Brun, C., M. Giovanni, M. Luciano, and R. Giovanni, "Perturbation : An Efficient Technique for the Solution of Very Large Instances of the Euclidean TSP," *INFORMS Journal on Computing*, Vol.8, No.2 (1996), pp.125-133.