IE Interfaces
Vol. 24, No. 2, pp. 112-118 , June 2011.

시간제약이 있는 상금 획득 외판원 문제에 대한 동적 계획 접근 방법

태현철[1] · 김병인[1]

[1]포항공과대학교 산업경영공학과

Dynamic Programming Approach for Prize Colleting Travelling Salesman Problem with Time Windows

Hyun chul Tae[1] · Byung In Kim[1]

[1]Department of Industrial and Management Engineering,
Pohang University of Science and Technology, Pohang, Gyeongsangbuk-do, 790-784, Korea

This paper introduces one type of prize collecting travelling salesman problem with time windows (PCTSPTW), proposes a mixed integer programming model for the problem, and shows that the problem can be reduced to the elementary shortest path problem with time windows and capacity constraints (ESPPTC). Then, a new dynamic programming algorithm is proposed to solve ESPPTC quickly. Computational results show the effectiveness of the proposed algorithm.

Keyword: Prize collecting, Travelling salesman problem with time windows, Dynamic programming

1. 연구 개요

외판원 문제(Travelling Salesman Problem : 이하 TSP)는 매우 유명한 문제로, 외판원이 주어진 모든 고객들을 최소비용으로 방문할 수 있는 고객 방문 순서를 구하는 문제이다. TSP에서 외판원은 주어진 고객 중 어느 곳에서도 출발할 수 있으며 반드시 출발지점으로 돌아와야 한다. TSP는 NP-Hard로 알려져 있다(Garey and Johnson, 1979).

상금 획득 외판원 문제(Prize Collecting Travelling Salesman Problem : 이하 PCTSP)는 TSP의 한 변형으로 각 고객마다 상금(Prize)이 있다고 가정한 문제다. PCTSP 또한 NP-Hard이다(Feillet 등, 2005). PCTSP는 TSP와 달리 외판원은 정해진 출발지점에서만 출발 할 수 있고 고객들을 방문한 후 반드시 출발지점으로 돌아와야 한다. PCTSP에서 외판원은 모든 고객을 방문할 필요는 없다. Feillet 등(2005)은 PCTSP를 다음과 같이 3가지 유형으로 나누었다.

(1) 최대 허용 고객방문비용이 주어진 상태에서 획득 상금을 최대화 하는 유형. 즉, 고객방문비용을 제한조건으로 보고 획득 상금 최대화를 목적 식으로 갖는 유형.

(2) 최소 요구 상금 액 이상의 상금 액을 얻으며 고객방문비용을 최소화하는 유형. 즉, 획득 상금을 제한조건으로 보고 고객방문비용 최소화를 목적 식으로 갖는 유형.

(3) 획득한 상금의 합에서 고객방문비용을 제외한 금액, 즉, 이익(Profit)을 최대화 하는 유형. 획득 상금과 고객방문비용 모두를 목적 식에서 고려하는 유형.

시간제약이 있는 상금 획득 외판원 문제(Prize Collecting Travelling Salesman Problem with Time Windows : 이하 PCTSPTW)는 PCTSP의 일반화된 문제로, PCTSP에 시간제약(Time Windows) 제한 조건이 추가된 문제이다. PCTSPTW도 PCTSP와 같이 목적 식과 제한 조건을 어떻게 정하느냐에 따라 문제 유형이 달라진다.

이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2010-0015611).
† 연락저자 : 김병인 교수, 790-784, 경북 포항시 남구 효자동 산31번지 포항공과대학교 산업경영공학과,
Fax : 054-279-2870, E-mail : bkim@postech.ac.kr
투고일(2010년 12월 04일), 심사일(2011년 02월 21일), 게재확정일(2011년 04월 04일)

Reuven 등(2005)과 Xiaohu 등(2008)은 각 고객 별로 상금이 있다고 보고 상금을 최대로 얻을 수 있는 경로를 찾는 PCTSPTW 문제에 대한 접근방법을 제시하였다. Zang과 Tang(2007)은 고객을 방문하지 못할 경우 고객 별로 벌금(Penalty)이 있다고 보고 고객방문 비용과 방문 못한 고객들로부터의 벌금들의 합을 최소화 시키는 PCTSPTW에 대한 알고리듬을 제시하였다.

본 논문에서 다루려는 PCTSPTW는 유형 (3)과 같이 이익을 최대화 시키는 유형의 문제이다. 하지만 유형 (3)과 달리 각 고객은 수요량(Demand)을 갖고 있고 외판원은 용량(Capacity)을 갖고 있어서, 외판원이 방문하는 고객들의 수요량의 합은 반드시 외판원의 용량보다 작아야 한다. 또한 각 고객을 방문할 때마다 고객에게 서비스를 제공하는 서비스 시간(Service time)이 소요된다.

이러한 유형의 PCTSPTW는 Tae 등(2010)이 지적한 것과 같이 시간제약이 있는 차량 경로 문제(Vehicle Routing Problem with Time Windows : 이하 VRPTW)의 비용 할당 문제에서 사용된다. Tae 등(2010)은 50명의 고객 크기 문제에서 VRPTW 비용 할당에 PCTSPTW가 약 200번 이상 계산됨을 보였으나 효율적인 PCTSPTW 알고리듬을 제시하지는 못하였다. 따라서 본 연구에서는 VRPTW 비용 할당 문제에 적합한 효율적인 PCTSPTW 알고리듬을 제안하고자 한다.

2. 문제 설명

노드(Node)들의 집합 N은 외판원의 출발지점 $\{S_0\}$과 n개의 고객 지점 $\{S_1, S_2, \cdots, S_n\}$을 포함한다, $N=\{S_0, S_1, S_2, \cdots, S_n\}$. 모든 노드들은 고유의 2차원 좌표 (x, y) 값을 갖고 있다. 모든 노드는 자신을 제외한 다른 모든 노드들에 대해 아크(Arc)를 갖고 있으며 아크들의 집합을 A로 나타낸다. 두 노드 간 아크 길이는 유클리디안 거리(Euclidean distance)로 가정하며, 한 예로 노드 1과 노드 2의 길이는 다음과 같다, $d_{12}=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$. 모든 고객은 고유의 수요량 r_i와 시간제약$[e_i, l_i]$, 그리고 상금 p_i를 갖고 있다, $\forall i \in N \setminus S_0$. 고객 i를 가장 이르게 방문할 수 있는 시간을 e_i로 나타내며, l_i는 고객 i를 가장 늦게 방문할 수 있는 시간이다. 외판원은 고객 i를 e_i보다 일찍 도착해서 기다릴 수 있다고 가정한다. 외판원의 출발지점에서의 출발 시간은 0이라 가정한다.

각 고객은 또한 고유의 서비스 시간을 갖을 수 있는데 본 연구에서는 모든 고객의 서비스 시간은 ST로 동일하다고 가정한다. 배달원의 용량은 Q로 나타내며 방문이 불가능한 경우를 제외하기 위해 $r_i \le Q$라 가정한다, $\forall i \in N \setminus S_0$. 노드 i에서 노드 j로 이동하는데 소요되는 비용을 c_{ij}로 나타내며 $c_{ij}=d_{ij}$라 가정한다. 노드 i에서 노드 j로 이동하는데 소요되는 시간을 t_{ij}로 나타내며 서비스 시간을 더 이상 고려하지 않기 위해서 t_{ij}를 아래와 같이 정의한다.

$$t_{ij}=\begin{cases} d_{oj} & i=0,\ \forall j \epsilon N \setminus S_0 \\ d_{ij}+ST & \forall i,\ j \epsilon N,\ i \neq j,\ i \neq 0 \end{cases}$$

본 논문에서 다루는 PCTSPTW에 대한 수리 식(Mathematical Formulation)과 그에 쓰이는 변수에 대한 설명은 다음과 같다.

$$x_{ij}=\begin{cases}1 & \text{if vertex } i \text{ immediately proceeds } j \\ 0 & \text{otherwise}\end{cases}$$

$$y_i=\begin{cases}1 & \text{if vertex } i \text{ is visited} \\ 0 & \text{otherwise}\end{cases}$$

L_i = cummulative loads after visiting vertex i

T_i = arrival time at vertex i

$$\max \sum_{i \in N \setminus S_0} p_i y_i - \sum_{i \in N} \sum_{j \in N, j \neq i} c_{ij} x_{ij} \quad (1)$$

subject to

$$\sum_{j \in N, j \neq i} x_{ij} = y_i \qquad i \in N \quad (2)$$

$$\sum_{j \in N, j \neq i} x_{ij} = y_j \qquad j \in N \quad (3)$$

$$y_0 = 1 \quad (4)$$

$$L_i + r_j \le L_j + M(1-x_{ij}) \qquad (i,j) \in A,\ i, j \in N \setminus s_0 \quad (5)$$

$$r_i \le L_i \le Q \qquad i \in N \setminus s_0 \quad (6)$$

$$T_i + t_{ij} \le T_j + M(1-x_{ij}) \qquad (i,j) \in A,\ i, j \in N \setminus s_0 \quad (7)$$

$$e_i \le T_i \le l_i \qquad i \in N \setminus s_0 \quad (8)$$

$$y_i = \{0,1\} \qquad i \in N \quad (9)$$

$$x_{ij} = \{0,1\} \qquad (i,j) \in A \quad (10)$$

식 (1)은 목적 식으로 이익을 최대화하는 것을 목적으로 한다. 식 (2)와 식 (3)은 외판원이 한 고객을 방문했을 경우 반드시 그 고객을 빠져나가게 하는 제약식이다. 식 (4)는 Depot에 대한 방문 조건이다. 식 (5)~식 (6)은 수요량과 관련된 제약식이다. 식 (5)는 arc (i, j)가 경로에 포함된 경우 L_j는 L_i와 d_j의 합보다 커야 한다는 것을 나타낸다. 식 (6)은 외판원의 용량에 관한 제약이다. 식 (7)은 두 연속하는 vertex들의 도착 시간에 관한 제약식이고 식 (8)은 시간 제약(time windows)이다. 식 (9)~식 (10)은 이진변수 제약식이다.

3. 시간제약과 용량제한이 있는 유일 방문 최단 경로 문제로의 변환

시간제약과 용량제한이 있는 최단 경로 문제(Shortest Path Problem with Time Windows and Capacity Constraints : 이하 SPPTC)는 출발지점에서 목적지점까지 시간제약과 용량제한을 만족시키며 최소비용으로 도달하는 경로를 찾는 문제이다. SPPTC에서는 동일한 노드에 두 번 이상의 방문이 허용되며 그러한 경우 Cycle이 존재한다고 한다. 동일한 노드가 k번 만에 재 방문

되는 경우 K-Cycle이 존재한다고 한다. 예로 2-3-4-2의 경우는 3-Cycle이 존재한다고 한다. Cycle이 허용되지 않는 경우 SPPTC는 시간제약과 용량제한이 있는 유일 방문 최단 경로 문제(Elementary Shortest Path Problem with Time Windows and Capacity Constraints : 이하 ESPPTC)가 된다. Dror(1994)는 ESPPTC가 NP-hard임을 보였다.

노드 i에서 노드 j로 이동하는 비용 C_{ij}를 아래와 같이 $\overline{C_{ij}}$로 변환한다면 PCTSPTW의 목적함수 식 (1)을 식 (1')로 바꿀 수 있다. 목적식이 (1')로 바뀐 PCTSPTW 문제는 출발지점과 목적지점이 동일한 ESPPTC와 동일하다.

$$\overline{c_{ij}} = \begin{cases} c_{i0} & i=0,\ \forall j \in N \setminus s_0 \\ c_{ij} - p_i & \forall i,\ j \in N,\ i \neq j,\ j \neq 0 \end{cases}$$

$$\min \sum_{i \in V} \sum_{j \in V, j \neq i} \overline{c_{ij}} x_{ij} \qquad (1')$$

4. ESPPTC 동적 계획 접근 방법에 관한 이전 연구들

Desrochers 등(1992)은 SPPTC에 대해 동적 계획법(Dynamic Programming)을 이용한 근사 다항 시간(pseudo-polynomial time) 알고리듬을 제시하였다. Desrochers 등(1992)이 제안한 알고리듬은 K-Cycle(K≥3)을 허용하는 해를 찾아주기에 PCTSPTW에 바로 적용하기는 불가능하다.

Feillet 등(2005)은 Desrochers 등(1992)이 제안한 알고리듬을 발전시켜서 ESPPTC의 최적 해를 찾아주는 알고리듬을 제안하였다. 하지만 Feillet 등(2005)이 제시한 알고리듬은 문제 크기가 커졌을 때 해를 찾지 못하는 경우가 많았다. 그들은 고객 수가 100개인 문제 40개 중에서 300초안에 오직 17개의 문제에서만 해를 찾을 수 있었다.

Righini and Salani(2006)은 Feillet 등(2005)이 제시한 동적 계획법이 양방향 탐색(Bi-directional Search)를 통해 어떻게 개선 될 수 있는지를 실험결과를 통하여 제시하였다. 그들은 양방향 탐색을 통해 Feillet 등(2005)이 구하지 못한 문제들에 대해서도 해를 구할 수 있었으나 문제 크기가 큰 문제에서 1000초 이상의 시간이 걸리는 경우가 있었고 특정 문제에선 3600초 안에 해를 찾는데 실패하는 경우도 있었다.

Righin and Salani(2008)은 Decremental State Space Relaxation(이하 DSSR)을 이용한 ESPPTC에 대한 해결 방법을 제안하였다. 그들은 모든 문제에서 성공적으로 해를 구할 수 있었으나 문제 크기가 큰 문제들의 해를 구할 때 여전히 계산 시간이 100초 이상 걸리는 경우가 적지 않았으며 가장 나쁜 경우 900초 이상 걸리는 경우도 있었다.

Feillet 등(2005)과 Righin and Salani(2006) Righin and Salani(2008)이 제안한 알고리듬들은 ESPPTC의 해를 성공적으로 찾아주지만 문제 크기가 커지면 해를 찾지 못하거나 계산 시간이 급격히 커짐이 확인되었다. 하지만 VRPTW 비용 할당 문제(Tae *et al.*, 2010)에서는 50명의 고객에게 비용을 할당하는 경우에도 ESPPTC를 200번 이상 계산해야 했다. 따라서 차량 경로 비용 할당 문제에서는 최적 해가 아니더라도 최적 해와 근사한 해를 빠른 시간에 찾을 수 있는 ESPPTC 알고리듬이 필요하다.

Desrochers 등(1992)이 제안한 SPPTC 알고리듬은 위에서 살펴본 ESPPTC 알고리듬에 비해 상당히 간단하다. 본 연구는 Desrochers 등(1992)이 제안한 SPPTC 알고리듬이 간단하다는 점에 착안해, 이 알고리듬을 보다 효율적으로 변형시킨 새로운 동적 계획법을 제시한다.

5. 새로운 동적 계획법 알고리듬

5.1 Desrochers 등(1992)의 동적 계획법

Desrochers 등(1992)은 외판원이 용량 $q(r_j \le q \le Q)$를 가지고 출발지점 s_0에서 출발해서 고객 j에 시간 t 이전에 도착할 수 있는 경로의 최소비용을 $S_j(q,t)$로 나타내었다. 그들이 제시한 동적 계획법은 다음과 같다.

$$S_{S0}(0,0) = 0$$
$$S_j(q,t) = \min_{(i,j)\in A}[S_i(q',t') + \overline{c_{ij}} | t' + t_{ij} \le t,\ e_i \le t' \le l_i, q' + r_j \le q]$$
$$\forall j, q, t,\ j \in N \setminus s_0, r_j \le q \le Q, e_j \le t \le l_j$$

Desrochers 등 (1992)은 State를 용량 q, 시간 t, 고객 j로 정의하였다. 쉬운 이해를 위해 이를 3차원 큐빅으로 <그림 1>에 나타내었고 이를 앞으로 State Cube라 명하겠다. 이 State Cube를 채우는 요소를 State Cell이라 명하겠다. <그림 1>과 같이 시간을 이산적으로 보는 State Cube를 앞으로 Discrete State Cube라 명하겠다. <그림 1>은 외판원 용량이 5이고 시간은 0, 1, 2, …, 10, 11까지 12개이며 4개의 고객에 대한 State Cube이다. 이 State Cube에서 채워져 있는 부분은 고객의 시간제약을 나타낸다.

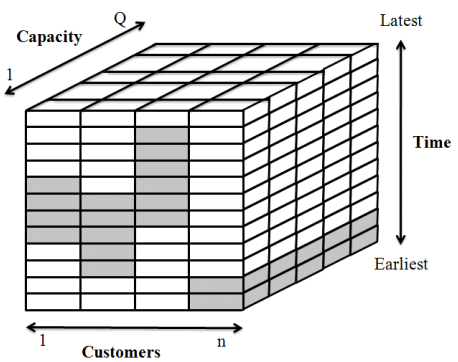

그림 1. Discrete State Cube의 예

Desrochers 등(1992)이 제기한 동적 계획법 알고리듬의 핵심을 요약하면 이 State Cube를 가장 작은 용량 1부터 가장 큰 용량 Q까지 고객들의 시간제약에 포함된 State Cell들을 점진적으로 채워 나가는 것이다. 우선 q = 1인 경우의 State Cell들을 모

두 채우고 그 다음은 $q = 2$인 경우의 State Cell들을 모두 채우고 이 과정을 $q = Q$일 때까지 반복한다. 모든 고객의 수요량(r_j)이 1이라 가정했을 때 <그림 1>의 State Cube에서 채워야 할 Cell의 개수는 다음과 같다.

$$q \times \sum_{i \in N \setminus s_0} (l_i + 1 - e_i) = 5 \times (4+5+6+2) = 85$$

5.2 정리 1 : if $q' \geq q$ then $S_j(q', t) \leq S_j(q, t)$

동일한 고객 j에 동일한 시간 t에 도착한다면 보다 큰 용량에서 출발한 외판원이 항상 더 적거나 같은 비용으로 도착할 수 있다. 따라서 Desrochers 등 (1992)이 제기한 동적 계획법의 탐색 범위를 정리 1을 통해 줄일 수 있다. 개선된 동적 계획법은 아래와 같다.

$$S_j(q, t) = \min_{(i,j) \in A} [S_i(q', t') + \overline{c_{ij}} | t' + t_{ij} \leq t, \\ e_i \leq t' \leq l_i,\ q' = q - r_j]$$

5.3 정리 2 : if $t' \geq t$ then $S_j(q, t') \leq S_j(q, t)$

동일한 용량 q에서 시작하여 동일한 고객 j에 도착하는 경우 보다 늦게 도착하는 외판원이 항상 더 적거나 같은 비용으로 도착할 수 있다. 따라서 정리 1에서 개선된 동적 계획법의 탐색 범위를 정리 2를 통해서 다시 줄일 수 있다. 또 다시 개선된 동적 계획법은 아래와 같다.

$$S_j(q, t) = \min_{(i,j) \in A} [S_i(q', t') + \overline{c_{ij}} | t' = t - t_{ij}, \\ e_i \leq t' \leq l_i,\ q' = q - r_j]$$

5.4 State Stick

Desrochers 등(1992)이 제안한 State는 연속적인 시간을 이산적인 정수 형으로 보기 때문에 시간을 정확하게 보면 볼수록 채워야 할 State Cell의 개수가 많아진다. 그들은 실수인 d_{ij}를 정수로 변환하기 위해서 소수점 2번째 자리부터 내림을 한 후 10을 곱하여 실수인 d_{ij}를 정수로 변환하였다. 그 후 다른 모든 단위에 10을 곱하여 동적 계획법을 진행하였다. 이 경우 [10, 20]의 시간제약에서 채워야 할 State Cell의 개수는 101개가 된다. 보다 정확한 시간을 구하기 위해서 소수점 3번째 자리에 내림을 한 후 100을 곱했다면은, [10, 20]의 시간제약에서 1001개의 State Cell들을 계산해야 했을 것이다.

채워야 할 State Cell의 개수가 많다는 것은 그만큼 계산해야 할 양이 많다는 것이다. Desrochers 등(1992)이 제안한 State 정의는 시간제약 범위가 넓으면 넓을수록 그리고 시간을 정확하게 보면 볼수록 계산해야 하는 양이 급격히 많아진다. 이를 해결하고자 새로운 관점으로 State를 정의하는 State Stick을 제안한다. 기존의 Discrete State는 비용 정보와 이전 방문 고객 Index정보를 갖고 있지만 State Stick은 비용 정보와 이전 고객 Index정보 뿐만 아니라 그 Stick의 시작 시간(Start Time)정보도 포함하고 있다(<그림 2>참조). 참고로 시간이 이를수록 비용이 올라가는 것(정리 2)도 확인 할 수 있다.

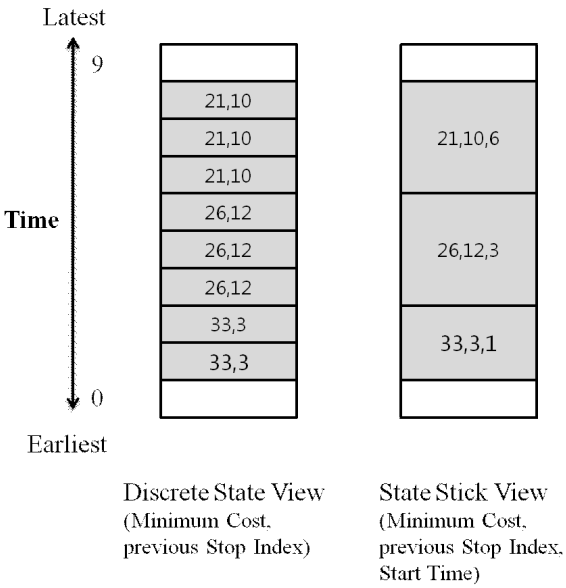

그림 2. Discrete State와 State Stick 관점 비교

기존 Discrete State 관점에서의 문제점은 동일한 Cell들의 계산을 여러 번 반복해야 한다는 점과 시간을 정수형으로 다루어야만 하고 시간을 정확하게 다루기 위해서는 추가적인 계산이 급격하게 늘어나는 점이다. State Stick 관점에서는 시작 시간을 원하는 대로 정할 수가 있어서 동일한 Cell에 대한 반복적인 계산이 사라지고 시간을 실수 형으로 정확하게 다루는 데에 어떠한 추가 계산이 들지 않는다. State Stick의 State Cube에 대한 예는 <그림 3>과 같다.

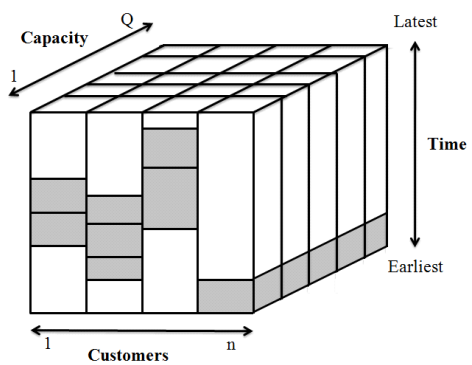

그림 3. Continuous State Cube의 예

5.5 State Stick 관점의 동적 계획법 알고리듬

<그림 4>와 같이 State Stick들이 쌓여있는 것을 State Stick Array라 명하겠다. Continuous State Cube는 각 용량과 각 고객 별로 고유

의 State Stick Array를 갖고 있으며 앞으로 이를 $SA(q, j)$로 나타내겠다($r_j \le q \le Q$, $j \in N \setminus s_0$). $SA(q, j)_k$는 $SA(q, j)$의 시간이 늦은 순서로 k번째 State Stick을 나타낸다. <그림 4>를 예로 들면 $SA(q, j)_1 \cdot \min cost$는 36이고 $SA(q, j)_2 \cdot startTime$은 2.224이고 $SA(q, j)_3 \cdot previousStopIndex$는 3이다. getSA($(q, j), t$)는 $SA(q, j)$에서 시간 t에 해당하는 State Stick을 의미한다. <그림 4>를 예로 들면 get($SA(q, j)$, 7)는 $SA(q, j)_1$와 같다. 시간 t가 State Stick의 시작 시간과 같은 경우는 다음 State Stick이 선택된다, 예로 get($SA(q, j)$, 2.224)는 $SA(q, j)_3$와 같다. 시간 t가 e_j보다 작은 경우, get($SA(q, j)$, 0.5)와 같은 경우는 해당하는 State Stick이 존재하지 않는다. 시간 t가 l_j 보다 큰 경우, get($SA(q, j)$, 8.9)와 같은 경우는 $SA(q, j)_1$와 같이 가장 느린 State Stick이 선택된다.

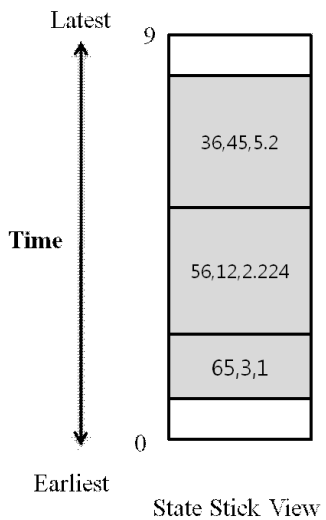

그림 4. State Stick Array의 예

getMin(q, j, t)함수는 용량 q를 갖고 출발해서 고객 j에 t시간 이전까지 최소비용으로 도착할 수 있는 이전 State Stick을 의미한다. 자세한 Procedure는 다음과 같다.

procedure *getMin(q, j, t)*
{
$i_0 = \arg min_{i \in N \setminus s_0} [get(SA(q-r_j, i), t-t_{ij}) \cdot \min Cost + \overline{c_{ij}}]$;
minStick. minCost
$= get(SA(q-r_j, i_0), t-t_{i_0 j}) \cdot \min Cost + \overline{c_{i_0 j}}$;
minStick. startTime
$= get(SA(q-r_j, i_0), t-t_{i_0 j})$ · startTime + $t_{i_0 j}$;
minStick. previousStopIndex
$= i_0$;
Return *minStick;*
}

Desrochers 등(1992)이 제안한 알고리듬을 개선하여 State Stick 관점으로 변환한 알고리듬은 다음과 같다.

procedure *State Stick Based D.P. Algorithm*
{
for(q = 1; $q \le Q$; q++) {
for(j = 1; $j \le$ j++) {
if(r_j - $q \ge 0$) {
$t = l_j$;
Iterate{
BestStick = getMin(q, j, t);
If(*BestStick.startTime* > e_j)
Add BestStick to SA(q, j);
t = BestStick.startTime;
Else
BestStick.startTime = e_j;
Add BestStick to SA(q, j);
Break;
}
}
}
}
}

6. 실험 결과

6.1 2-Cycle 제거와 K-Cycle 제거

본 연구에서 제안한 알고리듬은 동일 방문이 허용되는 해를 찾아주는 SPPTC 알고리듬이다. Desrochers 등(1992)은 해에 존재하는 Cycle을 줄이기 위해서 Houck 등(1980)이 제안한 2-Cycle 제거 방법을 도입하였다. 3이상의 Cycle을 제거하는 방법은 Irnich 와 Villeneuve(2003)에 의해 제안되었지만 K값이 커질수록 계산 시간이 급격히 증가하므로 계산 시간과의 Trade-Off 관계를 고려해서 Cycle 제거 범위를 결정해야 한다. 본 연구에서는 앞에서 밝혔듯이 계산 시간이 빠른 알고리듬을 목적으로 하기에 Houck 등(1980)이 제시한 2-Cycle 제거 방법만을 도입하였다. K-Cycle (K $\ge$ 3)은 단순하게 getMin(q, j, t)에서 고객 j를 포함하는 경로를 지닌 State Stick이 선택되지 않게 함으로써 제거하였다. 물론 이처럼 단순하게 중복되지 않게만 선택하는 것은 Feillet 등(2005)이 밝힌 대로 최적 해를 보장해주지 않는다. 하지만 실험 결과 이런 방법이 최적해와 근접한 값을 빠른 시간 안에 찾아 주는 것이 확인되었다.

6.2 최적해

Feillet 등(2005)과 Righin and Salani(2006), Righin and Salani(2008)은 모두 최적해를 구하였다. 제안된 알고리듬의 성능비교를 위하여 Righin and Salani(2008)가 제공한 최적해를 사용하였다.

6.3 Bench Mark Data

Feillet 등(2005)과 Righini and Salani(2006), Righini and Salani(2008)은 Solomon(1987)이 제시한 VRPTW 데이터를 변형하여 사용하였다. Feillet 등(2005)은 노드 사이의 아크 거리에 0~20사이의 정수를 무작위로 빼서 음의 값을 갖는 아크가 존재하도록 하였다. Righini and Salani(2008)는 각 노드에 0~20사이의 Prize를 주어서 음의 값을 갖는 아크가 존재하도록 하였다. 본 연구에서는 Righini and Salani(2008)가 사용한 Bench Mark Data를 사용하였다.

6.4 실험 결과

실험은 Intel Dual core E7300 2.66GHz 2.67GHz/4GB RAM PC 환경에서 C++언어를 통해 행해졌다. PCTSPTW 문제이기에 ESPPTC 해의 비용의 부호를 바꾸어 이익으로 표현하였다.

표 1. 계산 결과 및 최적해와 비교

Problem	Profit	Time (Sec.)	Exact	Gap	Gap (%)
c101	84	0.1	84	0	0.0
c102	83	1.3	84	1	1.2
c103	83	6.5	84	1	1.2
c105	84	0.2	84	0	0.0
c106	84	0.3	84	0	0.0
c107	84	0.2	84	0	0.0
c108	84	0.5	84	0	0.0
c109	84	0.8	84	0	0.0
r101	36	0.1	36	0	0.0
r102	77	1.5	84	7	8.3
r103	98	3.9	99	1	1.0
r104	100	8.7	100	0	0.0
r105	64	0.3	65	1	1.5
r106	91	1.7	92	1	1.1
r107	104	6.0	105	1	1.0
r108	111	12.1	111	0	0.0
r109	97	0.8	97	0	0.0
r110	97	2.9	102	5	4.9
r111	94	4.5	102	8	7.8
r112	102	7.9	107	5	4.7

Gap = Exact - Profit

Gap(%) = (Exact - Profit) / Exact × 100

<표 1>은 제안된 알고리듬의 해의 값을 최적해와 비교한 것이다. Gap은 제안된 알고리듬이 찾은 해의 값과 최적해와의 차이를 의미한다. Gap(%)는 최적해 대비 GAP의 값을 백분율로 나타낸 것이다. Gap이 0이란 것은 제안된 알고리듬이 최적해를 찾아 주었음을 의미한다. 알고리듬은 대체로 최적해에 근접한 값을 찾을 수 있었다. 단, 특정문제에서 약 8%정도 차이가 났음을 알 수 있다.

표 2. 속도 비교(Sec.)

Problem	Feillet 등 (2005)	Righini and Salani(2006)	Righini and Salani(2008)	제안된 방법
c101	0.1	0.1	0.0	0.1
c102	58.0	4.0	0.6	1.3
c103	-	148.5	40.2	6.5
c105	0.2	0.2	0.1	0.2
c106	0.3	0.3	0.1	0.3
c107	0.3	0.3	0.1	0.2
c108	0.6	0.6	0.2	0.5
c109	2.4	1.9	9.2	0.8
r101	0.1	0.1	0.0	0.1
r102	236.8	4.5	21.7	1.5
r103	-	105.8	159.7	3.9
r104	-	1278.6	78.3	8.7
r105	0.1	0.2	0.0	0.3
r106	167.9	11.0	71.6	1.7
r107	-	141.2	136.4	6.0
r108	-	1094.8	146.6	12.1
r109	2.1	0.9	2.9	0.8
r110	66.5	12.7	19.3	2.9
r111	-	39.4	53.2	4.5
r112	-	1019.1	316.9	7.9

<표 2>는 Feillet 등(2005)과 Righini and Salani(2006) Righini and Salani(2008)이 제시한 실험들과의 계산 시간 비교이다. Feillet 등(2005)이 밝힌 대로 음의 값을 갖는 아크가 많을수록 문제의 복잡성은 커진다. Feillet 등(2005)이 제시한 데이터에서 음의 값을 갖는 아크 비율은 7~11%정도였고 본 논문에서 사용한 데이터의 음의 아크 비율은 약 8~9%이다. Feillet 등(2005)이 제시한 문제와 거의 비슷한 음의 아크 비율을 갖고 있지만 본 논문에서 제안된 알고리듬이 이전에 제기된 알고리듬에 비해서 상당히 빠른 시간 안에 근사 해를 찾아 줌을 알 수 있다.

7. 결론 및 추후 연구

본 연구는 PCTSPTW의 한 유형을 수리모델과 함께 정의한 후 그 유형이 ESPPTC와 동일한 문제임을 보였다. ESPPTC에 관한 이전 연구들을 살펴본 후, 짧은 시간 안에 최적에 근접한 해를 찾아주는 ESPPTC 알고리듬의 필요성을 확인했다. 그 후 새로운 방식의 개선된 동적 계획법을 제안했다. 실험 결과, 제안된 방법론이 빠른 시간 안에 만족할만한 해를 찾아주는 것으로 확인되었다.

추후에는 보다 지능적인 Cycle 제거 방법의 개발이 필요하다. 현재 제안된 K-Cycle 제거 방법은 경로를 만드는 과정에서 Cycle이 생기는 것을 막는 방법인데, 이와 달리 Cycle이 있는 해를 얻은 후 Post Optimization과 같이 해를 발전(Improvement)시키는 형식으로 Cycle을 제거하는 방법도 고려해 볼 만하다.

참고문헌

Desrochers, M., Desrosiers, J., and Solomon, M. (1992), A new optimization algorithm for the vehicle routing problem with time windows, *Operations Research*, **40**, 342-354.

Dror, M. (1994), Note on the complexity of the shortest path models for column generation in VRPTW, *Operations Research*, **42**, 977-978.

Feillet, D., Dejax, P., and Gendreau, M. (2005), Traveling salesman problems with profits, *Transportation Science*, **39**, 188-205.

Garey, M. R. and Johnson, D. S. (1979), Computers and Intractability : A Guide to the Theory of NP-Completeness, W. H. Freeman, San Francisco.

Houck, D., Picard, J., Queyranne, M., and Vemuganti, R. (1980), The travelling salesman problem as a constrained shortest path problem : Theory and computational experience, *Opsearch*, **17**, 93-109.

Irnich, S. and Villeneuve, D. (2003), The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$, Cahiers du GERAD G-2003-55, HEC Montréal.

Righini, G. and Salani, M. (2006), Symmetry helps : Bounded bidirectional dynamic-programming for the elementary shortest path problem with resource constraints, *Discrete Optimization*, **3**, 255-273.

Righini, G. and Salani, M. (2008), New dynamic programming algorithms for the resource constrained elementary shortest path problem, *Networks*, **51**, 155-170.

Reuven, B.-Y., Guy, E., and Shimon, S. (2005), On approximating a geometric prize-collecting traveling salesman problem with time windows. *Journal of Algorithms*, **55**, 76-92.

Solomon, M. M. (1987), Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints, *Operations Research*, **35**, 254-265.

Tae, H. C., Jun, Y., and Kim, B.-I. (2010), A vehicle routing cost allocation problem with time windows, Working Paper.

Xiaohu, S., Liupu, W., You, Z., and Yanchun, L. (2008), An Ant Colony Optimization Method for Prize-collecting Traveling Salesman Problem with Time Windows Natural Computation, *Fourth International Conference on Natural Computation*, 480-484.

Zhang, Y. and Tang, L. (2007), Solving Prize-Collecting Traveling Salesman Problem with Time Windows by Chaotic Neural Network, ISNN '07 Proceedings of the 4th international symposium on Neural Networks : *Part II-Advances in Neural Networks*, 63-71.

태 현 철

포항공과대학교 산업경영공학과 석박사 통합과정

연세대학교 정보산업공학사

관심분야 : column generation, vehicle routing problem

김 병 인

포항공과대학교 산업경영공학과 부교수

포항공과대학교 산업공학 학사/석사

미 런슬레어 공대 산업공학 박사

LG전자 생산기술원 주임연구원

미 멤피스대 조교수

미 Institute of Information Technology 사 R&D Director 등 역임

관심분야 : vehicle routing problem, generic simulation, 최적화