Journal of the Korean Institute of Industrial Engineers
Vol. 27, No. 1, pp. 18-24, March 2001.

지연비용을 고려한 서비스 시간대가 존재하는 외판원 문제에 대한 발견적 해법

서병규 · 김종수†

한양대학교 산업공학과

A Heuristc Algorithm for the Traveling Salesman Problem with Time Windows and Lateness Costs

Byung Kyu Suh · Jong Soo Kim

Department of Industrial Engineering, Hanyang University, Ansan

This paper presents a model and a heuristic algorithm for the Traveling Salesman Problem with Time Windows(TSPTW). The main difference of our model compared with the previous ones lies in that the time windows we are concerned are more flexible and realistic than the previous ones. In the typical TSPTW, the service at a node must begin within the time grid called the time window that is defined by the earliest and the latest time to start the service at each node. But, in real business practice, a lateness cost is usually penalized rather than the service is prohibited at all when a vehicle arrives after the latest time. Considering this situation, we develop a model with a new time window that allows an arrival after the latest time and penalizes the late arrival by charging a lateness cost. A two-phased heuristic algorithm is proposed for the model and is extensively tested to verify the accuracy and efficiency of the algorithm.

Keywords : TSP, Time windows

1. 서 론

1.1 연구의 배경과 목적

서비스 시간대가 존재하는 외판원 문제(Traveling Salesman Problem with Time Windows; TSPTW)는 임의의 노드(출발점)를 출발하여 각 노드를 해당 노드가 요구하는 서비스 시간대(time window)에 반드시 한 번만 방문하고 출발점으로 되돌아오는 최소비용의 순환로를 찾는 문제이다. 서비스 시간대는 서비스 시작이 허용되는 가장 빠른 시각과 더 이상 늦게 방문해서는 안 되는 시각 사이의 구간이다. 이들은 각각 서비스 시간대의 가장 이른 시각(earliest time)과 가장 늦은 시각(latest time)으로 불린다. 만약 임의의 노드에 정해진 가장 이른 시각보다 일찍 방문하게 되면 가장 이른 시각까지 기다려야 하며, 반면에 가장 늦은 시각이 지나서 방문하게 되면 서비스 자체를 개시할 수 없게 되어 방문이 성립되지 않는다. 이와 같은 서비스 시간대 제약조건은 고정된 시간계획에 따라 운영되는 환경에서 필연적으로 발생하는 문제로, 택배 및 우편물 배달문제, 산업폐기물 수거문제, 통학버스 경로 문제, A/S 방문문제 등 다양한 유형의 문제에서 존재하고 있다.

기존의 서비스 시간대가 있는 외판원문제에서는 가장 이른 시각과 가장 늦은 시각으로 이루어진 서비스 시간대 사이에서만 서비스가 허용되고 가장 늦은 시각 이후에 도착하는 경우는 방문이 허용되지 않는 것으로 가정하였다. 그러나 현실에서 이와 같이 엄격하게 가장 늦은 시각을 적용하는 경우는 거의 없으므로 이에 적합한 모형과 해법이 제시되어야 한다. 따라서 본 연구에서는 가장 늦은 시각 이후에 도착하는 경우를

이 연구는 2001년도 두뇌한국21사업에 의해 지원되었음.
† Corresponding author : Professor Jong Soo Kim, Department of Industrial Engineering, Hanyang University, Ansan, 425-791, Korea,
Fax : +82-31-409-2423, e-mail : jskim@mecors.hanyang.ac.kr
1999년 5월 접수, 2회 수정 후, 2000년 12월 게재 확정.

허용하되 지연비용(lateness cost)을 부가 하는, 좀더 현실적인 모형을 제안하고 제시된 모형을 풀 수 있는 효율적인 해법을 개발한다.

1.2 기존 연구의 고찰

외판원 문제는 1759년 Euler와 Vandermonde가 체스판에서 기사(knight)가 움직이는 경로에 대한 논의로부터 시작해 1856년 Hamilton이 Hamiltonian Cycle을 제시하면서 학문적으로 구체화되기 시작하였다. 이 후 다양한 분야에 폭넓게 응용될 수 있는 점과 매우 간단해 보이면서도 최적해를 구하기가 어렵다는 특징으로 인하여 지속적으로 연구되어 왔다.

최근에 와서는 순수 외판원 문제에 좀더 현실성을 부여하기 위하여 몇 가지 제약을 부여한 문제들과 이를 응용한 차량경로 문제에 대한 연구가 많이 이루어지고 있다. 제약이 추가된 외판원문제를 크게 두 가지로 분류하면 TSPTW와 우선순위 제약을 갖는 외판원 문제(Precedence-Constrained Traveling Salesman Problem; PCTSP)로 나눌 수 있다(Bodin and Golden, 1981). 이를 차량경로 문제(Vehicle Routing Problem; VRP)로 확장하게 되면, 시간 제약이 있는 경우는 스케쥴링 문제로, 시간 제약이 없고 우선순위가 있는 경우는 순수 경로 문제(routing problem)로 표현되어지기도 한다. 또한 서비스 시간대 제약과 우선순위 제약을 동시에 갖는 경우를 combined routing and scheduling 문제라 하는데, 현실에 가장 근접하게 모형화한 것이다(Desrochers, Desrosiers, and Solomon, 1992 ; Gendreau, Hertz, Laporte, and Stan, 1998 ; Mingozzi, Bianco, and Ricciadelli, 1997 ; Solomon, 1987).

TSPTW에 관한 대표적 연구들을 열거하면, 우선 Savelsbergh(1985)는 TSPTW의 가능해를 구하는 것조차 NP-complete 문제임을 증명하고 실행 가능해를 산출하기 위한 해법으로 interchange heuristic을 제시하였다. 이후 많은 학자들이 최소비용(거리, 시간)을 산출하기 위한 최적해법 도출에 많은 연구를 진행시켜 왔는데, 예를 들어 Baker (1983)는 이완된 모델의 쌍대문제를 통해 하한을 얻어내는 분지한계법을 제시하여 서비스 시간대가 중복되는 50개의 노드를 갖는 문제를 효과적으로 풀어내었다. Dumas *et al.* (1995)은 상태와 단계 및 상태변화(state transition)를 획기적으로 줄일 수 있는 해법을 동적계획법을 이용하여 제시하였다. Desrosiers *et al.* (1986)은 dial-a-ride 문제의 최적해를 구하는 해법을 제시하고 CYBER 173으로 6초 안에 80개의 노드를 갖는 문제를 풀어내었다.

기존의 모든 TSPTW 관련 연구에서는 서비스 시간대를 엄격하게 적용하여 가장 늦은 시각 이후에 도착하는 경우는 허용하지 않는 것을 가정하였다. 그러나 이러한 원칙은 현실에서는 거의 적용되지 않고 있으며 차량이 서비스 시간대 가장 늦은 시각 이후에 도착하는 경우에는 서비스 지연과 이로 인한 고객의 대기시간 발생으로 인하여 지연비용이 발생하는 것으로 간주하는 것이 훨씬 현실적이라 할 수 있다.

본 연구에서는 이러한 점을 고려하여 서비스지연에 대한 고객의 대기시간을 인정하여 가장 늦은 시각 이후에 도착하는 경우를 허용하되 이러한 경우에는 지연비용을 부과하는 모형을 제시하고, 이 모형에 대한 효율적 알고리듬을 개발하였다.

2. 모형 및 해법

2.1 기호의 정의

본 연구에서 사용하는 기호는 다음과 같다.

N : 전체 노드의 수 (문제의 크기)

$[a_j, b_j]$: 노드 j에서의 서비스 시간대

a_j : j번째 노드에서의 서비스 시간대에서 가장 이른 시각(the earliest time)

b_j : j번째 노드에서의 서비스 시간대에서 가장 늦은 시각(the latest time)

t_j : j번째 노드에서의 서비스 시작시간

lc_j : j번째 노드의 지연비용

a_j : 지연비용 계수(즉, j번째 노드에서 단위 시간당 발생하는 지연비용)

U : 전체 경로에 대한 지연 한계시간

s_j : 노드 j의 서비스 소요시간

c_{jk} : 노드 j에서 노드 k까지의 이동비용

τ_{jk} : 노드 j에서 노드 k까지의 이동시간

u_j : 순환경로에서 j번째 노드의 방문 순서

K : 충분히 큰 수

m_j : 노드 j에서의 지연시간

p_j : 노드 j에서의 지연시간을 결정하기 위해 필요한 변수

2.2 서비스지연을 허용하는 서비스 시간대와 모형의 기본가정

기존연구에서는 각 노드에 대해 $[a_j, b_j]$와 같이 가장 이른 시각과 가장 늦은 시각으로 구성되는 서비스 시간대와 서비스 소요시간(s_j)을, 그리고 각 호에는 이동시간인 τ_{jk}와 이동비용 c_{jk}를 가정하였다. 이때 서비스 시간대 $[a_j, b_j]$의 의미는 노드 j에 가장 이른 시각(a_j) 이전에 도착하였을 경우 가장 이른 시각이 될 때까지 기다려야 한다는 것을, 가장 늦은 시각(b_j) 이후에 도착하는 경우는 방문이 성립되지 않는 것을 나타낸다.

이에 반하여 본 연구에서 제안하는 서비스 시간대는 <그림 1>에 나타난 것과 같이 노드 j에 서비스 시간대 안에 도착한 경우($a_j \le t_j \le b_j$)는 지연비용이 발생하지 않는다고 간주하며

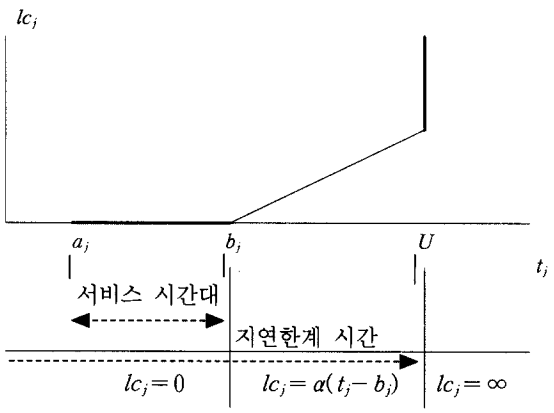

그림 1. Proposed Time Window and Its Lateness Cost.

($lc_j=0$), 가장 늦은 시각 이후에 도착하게 되는 경우 ($b_j<t_j\leq U$)는 $\alpha_j(t_j-b_j)$ 만큼의 지연비용이 부과된다고 가정한다. 또한 각 노드에 동일하게 적용되는 지연 한계시간(U)을 두어 이 시각 이후에 도착하는 경우에 대해서는 방문을 허용하지 않는다고 가정한다. 즉 지연되는 시간에 합리적인 제한을 두어 지연시간의 정도가 비현실적으로 크게 되는 경우를 방지한다.

제안모형의 기타 가정들은 다음과 같다.

(1) 지연비용은 노드를 방문하여 서비스가 끝나는 시각을 기준으로 발생한다. 즉, 각 노드에서 방문시각이 서비스 시간대 내에 있다고 하더라도 서비스가 완료되는 시각이 서비스 시간대를 벗어나는 경우 그 시간만큼의 지연비용이 부과된다. 이 가정은 서비스 시간대에 방문시각만을 적용하는 것보다 현실적이다.

(2) 모든 노드를 방문한 뒤 출발점으로 귀환하는 시간도 U의 제약에 포함된다.

2.3 수리적 모형

본 연구에서 제시된 새로운 서비스 시간대 모형을 고려한 외판원문제는 다음과 같이 수리적으로 모형화할 수 있다.

$$\text{Min} \sum_{j}^{N}(\alpha_j m_j + \sum_{k}^{N}(c_{jk}x_{jk})) \quad (1)$$

$$\text{s.t.} \sum_{j=1}^{N} x_{jk}=1, \quad \text{for } k=1,2,\ldots,N \quad (2)$$

$$\sum_{k=1}^{N} x_{jk}=1, \quad \text{for } j=1,2,\ldots,N \quad (3)$$

$$u_j-u_k+Nx_{jk} \leq N-1, \quad \text{for } j\neq k;\ j=2,3,\ldots, N;\ k=2,3,\ldots,N \quad (4)$$

$$t_j+s_j+\tau_{jk}-K(1-x_{jk})\leq t_k, \text{ for all } j \text{ and } k=2,3,4,\ldots,N \quad (5)$$

$$a_j \leq t_j, \quad \text{for all } j \quad (6)$$

$$t_j-b_j-m_j+p_j\geq 0, \quad \text{for all } j \quad (7)$$

$$m_j,\ p_j \geq 0, \quad \text{for all } j \quad (8)$$

$$t_j \leq U, \quad \text{for all } j \quad (9)$$

$$x_{jk}\in\{0,1\} \text{and } u_k \geq 0, \quad \text{for all } j \text{ and } k \quad (10)$$

식 (4)는 전체경로상에 구성될 수 있는 부분경로를 제거하며, 식 (5)는 노드 j에서 노드 k로 경로가 연결될 경우 노드 j에서의 '서비스 시간'과 '노드 k로의 이동시간'을 합한 것이 노드 k에서 서비스 시작시각보다는 작아야 한다는 것을 의미한다. 식 (6)은 노드 j에서의 서비스 시작시각은 노드 j의 서비스 시간대에서 가장 이른 시각보다는 커야 한다는 것을 나타내며, 식 (7)에서 각 노드에서의 지연시간이 결정된다.

2.4 발견적해법

위에서 언급한 바와 같이 TSPTW가 NP-hard 문제이므로 다양한 발견적해법에 대한 연구가 있어 왔다. 그러나 본 연구에서 제시하는 서비스 시간대 모형은 기존연구와 다르므로 다항식의 계산량을 갖는 경로 개선기법인 3-optimal 알고리듬(Lin, Kernighan, 1973)을 이용한 새로운 발견적 해법을 제시한다.

3-optimal 알고리듬의 기본개념은 임의의 초기 경로로부터 3개의 호를 선택하여 경로에 포함되지 않은 호를 대상으로 교환하여 값이 개선되면 순환로를 재구성하여 위의 절차를 반복하는 방법이다. 3-optimal 알고리듬을 본 연구에서 제시하는 모형에 적용하는 가장 간단한 방법은 순환로의 개선기준에 2.3절의 목적식 (1)과 같이 지연비용을 함께 고려하는 것이다. 이 방법의 단점은 교환대상의 모든 경로에 대한 지연비용을 계산함으로써 수행시간이 크게 증가하는 점이다.

본 연구에서는 이러한 단점을 보완하여 3-optimal 알고리듬 적용시 단순히 이동비용만을 고려하여 가능해 영역을 구성한 후 지연비용을 부과하는 두 단계 알고리듬을 제안한다. Phase Ⅰ과 Phase Ⅱ로 구성된 제안해법은 해의 정확도를 높이기 위해 Phase Ⅰ 과정을 반복하여 가능해 영역을 확장하는 절차를 거친다.

2.4.1 Phase Ⅰ(3-opt 알고리듬 절차)

임의의 초기해를 생성한 후 3-optimal 알고리듬을 적용하여 이동비용을 구한다. 3-optimal 알고리듬의 경우 초기해가 좋다고 해서 최종적으로 산출되는 해가 좋다는 보장은 없으나, 최종해가 초기해에 영향을 받으므로 여러 개의 초기해로 반복적으로 수행하여 개선할 수 있다.

2.4.2 Phase Ⅱ(지연비용 부과)

본 해법의 Phase Ⅱ에서는 Phase Ⅰ에서 구한 최소 이동비용에 각 노드별로 지연비용을 더해서 총비용을 산출하게 된다. 우선 Phase Ⅰ을 통해 얻어낸 경로들을 이동비용이 낮은 순으로 나열한 후 첫번째 경로(즉, 가장 낮은 이동비용을 갖는 경로)의 지연비용을 계산하여 이 경로의 총비용 TC_1을 얻어낸

다. 만약 이때 임의의 노드에서의 방문시간이 U를 초과한 시간으로 판명되면 해당경로는 실행불가능 경로이므로 이를 제거하고 다음 순위의 경로를 고려한다. 처음으로 계산되어진 실행가능 경로의 총비용을 임시 최소 비용(temporary minimum cost)에 저장하고 이것을 Phase Ⅰ에서 얻은 다른 나머지 경로들의 이동비용과 비교해서 이보다 더 큰 이동비용을 갖는 경로들을 미리 제거한다. 남은 경로들은 임시 최소 비용보다 낮은 총비용을 가질 가능성이 있으므로 이들에 대해서는 지연비용을 고려하는 추가작업을 진행한다. 이때에도 실행불가능 경로는 제외시킨다.

여기서 주의할 점은 주어진 경로들이 순방향으로만 표시되어 있다는 것이다. 즉, 고려대상인 비용행렬이 대칭행렬이므로 순방향 경로인 $v_1-v_2-v_3-\cdots-v_n-v_1$(v_i : i번째 노드)과 이것의 역방향 경로인 $v_1-v_n-v_{n-1}-\cdots-v_2-v_1$의 경우 이동비용만을 고려한 비용값은 같지만 지연비용까지를 고려하는 Phase Ⅱ에서는 이들이 서로 다른 총비용을 가지게 된다. 따라서 항상 순방향 경로를 다른 경로와 비교한 후 다음으로 낮은 경로를 계산하기 전에 역방향 경로의 총비용을 먼저 구해야 한다. 이러한 절차를 반복하여 현재의 임시 최소 비용보다 낮은 비용을 갖는 경로가 나타나면 총비용을 이 값으로 수정하고 남은 경로들에 대하여 같은 과정을 반복하여 해법의 최종 결과를 도출한다.

2.5 해법의 절차

Phase Ⅰ : [3-optimal 절차]

단계 0. Phase Ⅰ의 반복횟수(β)를 결정한다. β가 크면 클수록 정확도는 향상되나 수행시간이 증가하므로 적절한 값으로 설정한다.

단계 1. 임의의 초기해를 구성하고 알고리듬을 수행한다. 해(경로와 그에 따른 이동비용)를 개선해 나가는 과정에서 발생하는 임시경로를 경로집합에 포함시킨다.

단계 2. 반복횟수가 β를 초과하면 Phase Ⅱ로 진행하고 그렇지 않으면 단계 1로 간다.

Phase Ⅱ :

단계 1. [경로의 순차적 나열]
Phase Ⅰ에서 얻은 경로를 비용이 작은 순으로 나열하고 이를 탐색할 경로집합으로 지정한다. 이러한 경로집합의 경로와 해당 비용을 Phase Ⅰ 순방향 경로 및 순방향 비용이라 한다. 경로집합 중 순방향 비용이 최소인 경로를 현재의 경로로 한다. 임시값 ← ∞로 한다.

단계 2. [경로 탐색]
(2.1) 경로집합에 더 이상 탐색할 경로가 없으면 단계 5로 간다. 그렇지 않으면 탐색되지 않은 순방향 경로 중 비용이 가장 작은 것을 현재의 경로로 한다.
(2.2) 현재의 경로에 지연비용을 추가하여 총비용을 구한다.
현재 경로가 순방향일 때는 단계 3으로 그렇지 않으면 단계 4로 간다.

단계 3.
(3.1) 임의의 노드를 방문하는 시간이 U를 초과하거나, (총비용 ≥ 임시 최소 비용)이면 (3.3)으로 간다.
(3.2) (총비용 < 임시 최소 비용)이면, 임시 최소 비용 ← 총비용으로 하고 이보다 큰 이동비용을 갖는 경로를 제거한다.
(3.3) 역방향 경로를 현재 경로로 하여 (2.2)로 간다.

단계 4.
(4.1) 임의의 노드를 방문하는 시간이 U를 초과하거나, (총비용 ≥ 임시 최소 비용)이면 (4.3)으로 간다.
(4.2) (총비용 < 임시 최소 비용)이면, 임시 최소 비용 ← 총비용으로 하고 이보다 큰 이동비용을 갖는 경로를 제거한다.
(4.3) (2.1)로 간다.

단계 5. [완료]
만약 임시 최소 비용 = ∞이면, 주어진 문제가 실행가능 경로가 존재하지 않는 경우이다.
그렇지 않으면, 현재 임시 최소 비용과 해당 경로를 해로 제시한다.

3. 수치예제

6개의 노드로 구성된 네트워크의 비용행렬(cost matrix) 및 이동시간행렬(traveling time matrix)이 각각 <표 1>과 <표 2>에 주어져 있다.

본 예제에서는 지연비용 함수($lc_j=\alpha_j(t_j-b_j)$)의 지연비용 계수를 $\alpha_j=2$로, 지연한계 시간을 $U=22$시로 하며, 출발점은 노드 1로 한다. Phase Ⅰ의 반복횟수는 $\beta=3$이다. <표 3>은 예제에 해당하는 각 노드의 서비스 시간대와 서비스 시간을

표 1. Cost Matrix

	1	2	3	4	5	6
1	–	13	15	8	2	20
2		–	18	24	9	12
3			–	12	21	5
4				–	16	11
5					–	7
6						–

표 2. Traveling Time Matrix

	1	2	3	4	5	6
1	–	0.5	2.5	1	1	1.5
2		–	1.5	6	1.5	1.5
3			–	1	0.5	1.5
4				–	4	0.5
5					–	0.5
6						–

표 3. Time Windows and Service Time

Node	Time Window $[a_j, b_j]$	Service Time s_j (minutes)
1	08 ~ 22	0
2	09 ~ 11	20
3	10 ~ 15	10
4	12 ~ 16	30
5	11 ~ 17	5
6	15 ~ 18	45

나타내고 있다.

<표 1>과 <표 2>에 주어진 문제를 Phase Ⅰ을 적용하여 얻은 경로집합은 <표 4>와 같다. Phase Ⅱ에서 지연비용 계산의 대상이 되는 경로집합은 초기해의 경로와 개선된 경로이다. 표에서 괄호안의 숫자는 전체 경로집합에서 이동비용의 오름차순으로 정렬한 순서이며 (-)는 중복된 경로를 의미하고 경로집합에 포함되지 않는다.

다음으로 해법의 Phase Ⅱ의 결과를 <표 5>에 요약하였다. 수행과정 1에서는 <표 4>의 경로비용 중 가장 작은 값을 갖는 경로인 (1)번 경로의 순방향 지연비용을 포함한 총비용을 계산해보면 66.2를 얻으므로 이 값이 현재의 임시 최소 비용이 되고 이 값보다 큰 경로비용을 갖는 (9)~(11) 경로를 제거하고 (1)의 역방향 총비용을 계산한다. <표 5>의 수행과정 2에서 역방향 총비용이 62.5가 되므로 임시 최소 비용을 이 값으로 대체하고 (6)~(8) 경로를 제거한 후, 경로 (2)의 순방향으로 진행

표 4. Result after Phase Ⅰ

반복	초기해		개선된 경로	이동비용
	경 로	이동비용		
1	1-2-3-6-4-5-1	65 (7)	1-4-2-3-6-5-1	64 (6)
			1-2-4-3-6-5-1	63 (5)
			1-4-3-2-6-5-1	59 (4)
			1-4-6-3-2-5-1	53 (2)
			1-4-3-6-2-5-1	48 (1)
2	1-3-6-4-5-2-1	69 (9)	1-2-3-6-4-5-1	65 (-)
			1-4-3-6-5-2-1	54 (3)
			1-4-3-6-2-5-1	48 (-)
3	1-6-4-5-3-2-1	99 (11)	1-4-6-5-3-2-1	78 (10)
			1-5-4-6-3-2-1	65 (8)
			1-4-6-3-2-5-1	53 (-)
			1-4-3-6-5-2-1	48 (-)

표 5. Result of the Algorithm

수행과정	경 로	총비용	임시최소비용
1	(1)의 순방향	66.2	66.2
2	(1)의 역방향	62.5	62.5
3	(2)의 순방향	82.0	62.5
4	(2)의 역방향	60.5	60.5
5	(3)의 순방향	68.7	60.5
6	(3)의 역방향	64.7	60.5
7	(4)의 순방향	70.7	60.5
8	(4)의 역방향	90.2	60.5

한다. 수행과정 3에서는 총비용이 82.0으로 현재의 임시 최소비용인 62.5보다 크므로 역방향으로 진행한다. 수행과정 4에서 (2)에 대한 역방향 총비용이 60.5이므로 이 값을 임시 최소비용으로 대체하고 경로 (5)를 제거할 수 있다. 수행과정 5~8에서 구한 총비용이 현재 임시 최소 비용보다 크므로 채택되지 않으며 더 이상의 경로가 존재하지 않으므로 해법을 끝낸다. 최종해는 경로 1-5-2-3-6-4-1이며 총비용은 60.5로 산출된다. 이상에서 알 수 있듯이 Phase Ⅰ 단계는 최선해를 찾을 수 있도록 가능해 영역을 확장시키는 역할을 하며, Phase Ⅱ 단계에서는 가능해 영역에 포함되는 경로들 중 일부만을 탐색함으로써 계산량을 줄여준다.

4. 해법의 평가

본 해법과 비교할 수 있는 기존 해법들이 없으므로 평가를 수행하기 위해 최적해를 complete enumeration을 통해 산출하여 비교하였다. Object Pascal 언어로 코드화하여 펜티엄 Ⅱ 333 Mhz IBM PC에서 수행하였고, 같은 수의 노드를 갖는 문제에 대한 실험횟수는 20회로 하였다. 실험에 사용된 거리, 이동시간 및 서비스 시간대 자료는 불규칙하게 발생시켰다. 이때 한 도시를 기준으로 하여 거리비용의 경우 1~50 사이의 값을, 이동시간의 경우는 2시간이내의 값을 갖도록 제약을 주었다. 거리 행렬과 이동시간 행렬의 경우는 삼각부등식(triangle inequality)을 만족하며 대칭을 이루도록 하였다. 서비스 시간대는 일일 12시간 운영을 가정하여 9시부터 21시 사이에 발생하도록 하였다. 지연비용 계수는 $0 \le \alpha_j \le u \cdot M$으로 하여 해법의 수행도를 평가하였다. 여기서 $u \sim$ Uniform[0.1~10]의 값이고 M=두 지점간의 거리비용의 평균 × 노드 수 ÷ 12 로 지정하였다. M은 경로상 발생하는 총거리 비용의 근사값을 운영시간(12시간)으로 나눈 값이므로 시간당 발생하는 거리비용의 값을 표시하며, 따라서 시간당 지연비용은 시간당 거리 비용의 0.1~10배의 범위에서 발생시켰다고 할 수 있다. 본 해법에서 Phase Ⅰ의 반복횟수는 β=3으로 하였다.

표 6. Average Percent Deviation of Computational Results
(unit : %)

노드의 수 \ 오차	평균오차	최대오차	최소오차
5	0.82	0.88	0
6	1.05	1.10	0
7	0.90	2.78	0
8	1.08	5.72	0
9	1.28	2.33	0
10	1.19	1.24	0

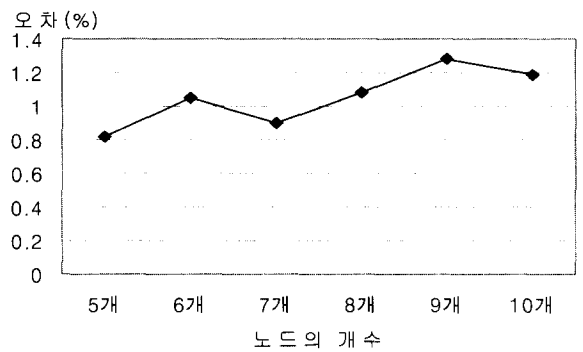

그림 2. Average Percent Deviation of the Algorithm for Various Sizes of Nodes.

우선, 해법의 정확성을 실험하기 위해 제시된 해법의 결과와 complete enumeration에 의한 최적해를 비교하였다. 또한 노드 수 100개까지의 문제에 대하여 수행시간을 측정하였다. 마지막으로 본 해법의 특성상 서비스 시간대의 크기는 최적해와의 오차의 정도에 영향을 줄 가능성이 크므로 서비스 시간대의 크기에 따른 오차의 변화를 실험을 통해 알아보았다.

먼저, 정확성에 관한 실험은 <표 6>에서 볼 수 있듯이 최적해와 본 해법의 해의 오차가 평균 1.05 %(= (0.82 + 1.05 + 0.90 + 1.08 + 1.28 + 1.19) / 6)인 것으로 나타났다. <그림 2>를 통해서 알 수 있듯이 문제의 크기가 커질수록 평균오차가 증가는 하지만 완만한 선형의 형태로 증가하는 바람직한 특징을 보여주고 있다.

<표 7>은 본 해법과 complete enumeration의 수행시간을 표로 나타낸 것이다. 노드의 수가 증가함에 따라 완만한 형태로 증가함을 알 수 있다.

마지막으로 서비스 시간대의 변화에 따른 오차의 정도를 알아보기 위한 실험을 수행하였다. 이를 위해 노드 8개를 갖는 문제에 대해 서비스 시간대의 평균간격을 두 시간에서부터 다섯 시간까지 한 시간씩 증가시켜가면서 오차의 변화를 측정하였다. 실험을 통해 산출된 오차의 값들을 <표 8>에 나타내었고 <그림 3>에 그래프로 나타내었다.

<표 8>의 결과를 보면 서비스 시간대의 간격이 늘어날수록 오차의 평균은 개선됨을 알 수 있다. 즉, 서비스 시간대의 간격이 커지게 되면 그 만큼 지연하게 될 가능성이 감소하게 되어

표 7. Computation Times for Various Sizes of Nodes
(unit : second)

노드의 수 \ 알고리듬	Complete Enumeration	본 연구의 알고리듬 (평균)
5개	1.1	0.27
6개	1.5	0.32
7개	15.3	0.56
8개	111.8	1.18
9개	1,218.8	1.53
10개	22,390.2	1.67
20개	–	8.36
30개	–	21.63
40개	–	45.23
50개	–	60.23
60개	–	89.71
70개	–	154.23
80개	–	311.58
90개	–	578.21
100개	–	1056.82

표 8. Mean Error of the Algorithm for Various Sizes of the Time Windows (Unit : %)

서비스 시간대 평균 시간 간격	오차의 평균	개선율(%)*
2시간	2.13^{E_1}	–
3시간	1.62^{E_2}	23.94
4시간	0.91^{E_3}	57.28
5시간	0.49^{E_4}	76.99

* 개선율 $= [(E_1 - E_i) / E_1] \times 100$, where $i = 2, 3, 4$.

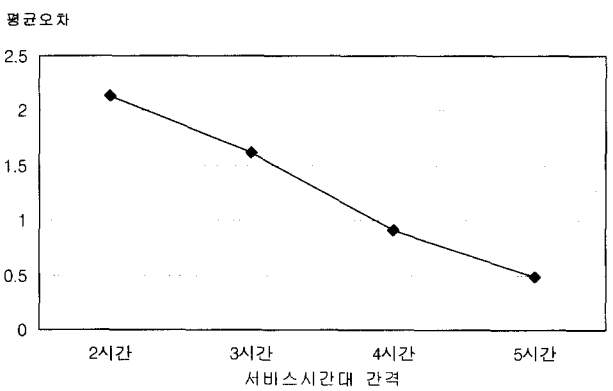

그림 3. Computational Behavior of the Algorithm for Various Intervals of the Time Windows.

지연비용의 부과 또한 감소하게 된다. 따라서 본 해법은 서비스 시간대의 영향이 감소되는 경우 최적해에 매우 가까운 해를 산출할 수 있는 것으로 판단된다. 또한 개선율, 즉 최적해와의 오차가 개선되는 비율을 살펴보면 4시간 이후로는 현저히 완만해짐을 알 수 있다.

지금까지의 세 가지 항목에 대한 실험 결과를 요약하면 다음과 같다. 5개의 노드를 갖는 문제에서 10개의 노드를 갖는 문제에 대한 오차는 평균 1.05 % 이내로 제시하고 있고 더 많은 노드를 갖는 문제에서도 β 값에 변화를 주면 낮은 수준의 오차를 유지할 것으로 예측된다. 본 해법의 수행시간은 다항식 알고리듬에 기초하고 있을 뿐만 아니라 수행시간에 상당한 영향을 미치는 지연비용계산 횟수를 현저히 줄였기 때문에 빠른 수행도를 보인다.

또한 본 해법은 서비스 시간대의 간격이 늘어나면 근사해에 효과적으로 접근할 수 있다. 기존의 TSPTW 해법들은 서비스 시간대의 간격이 늘어나면 오차가 증가하는 데 반하여, 본 해법은 이와 상반된 특징을 보인다. 서비스 시간대가 넓은 문제가 보다 일반적이고 현실적인 문제이므로(Lin and Kernighan, 1973)이러한 점은 본 해법의 큰 장점으로 판단된다.

본 해법의 또 다른 장점으로는 해법의 유연성을 들 수 있다. 즉, 본 해법은 가장 이른 시각 이전에 도착하였을 경우 조기도착에 대하여 비용을 부과하는 경우나, 각 노드에서 서비스하는 시간을 고려하지 않고 서비스 시간대에 도착하는 시각을 기준으로 지연비용을 부과하는 문제에도 쉽게 적용될 수 있다. 예를 들어, 전자의 경우는 서비스 시간대의 가장 이른 시각과의 차이를 비용계산시 부과하기만 하면 되고, 후자의 경우는 비용 계산시 간단한 비용계산수식의 변화를 주어 해결할 수 있다. 본 연구에서는 지연 한계 시각이 전체 노드에 대해 U로 동일하게 설정이 되어 있으나, 각 노드별로 지연한계 시각(U_j)이 설정되는 경우에도 비용 함수에 해당 U_j에 관련된 지연비용 계수를 추가하여 적용할 수 있으며 지연비용의 형태가 비선형인 경우에도 응용이 가능하다.

5. 결 론

본 연구에서는, 서비스 시간대가 있는 외판원 문제에 대해서 서비스 지연이 허용되는 상황에 적합하도록 기존의 서비스 시간대의 개념을 수정 보완한 새로운 형태의 모형을 개발하고 해법을 제시하였다. 실험을 통해서 제시된 오차의 값이나 수행시간은 본 해법이 합리적인 시간 내에 정확한 값을 효율적으로 산출한다고 입증하고 있다. 따라서 본 연구는 기존의 TSPTW 모델을 보다 현실적으로 수정 · 보완하였다는 점과, 수정된 모형의 해를 합리적인 시간 내에 효율적으로 산출할 수 있는 해법을 제시하고 있다는 점에 큰 의의를 부여할 수 있다.

추후 연구과제로는 서비스 시간대가 존재하는 차량경로 문제로의 응용연구 등을 들 수 있다.

참고문헌

Bodin, L. D. and Golden, B. L. (1981), Classification in Vehicle Routing and Scheduling, *Networks*, **11**, 97-108.

Baker, E. K. (1983), An Exact Algorithm for the Time-constrained Traveling Salesman Problem, *Opns. Res.*, **31**, 938-945.

Desrosiers, J., Dumas, Y. and Soumis, F. (1986), A Dynamic Programming Solution of the Large-Scale Single-vehicle Dial-a-ride Problem with Time Windows, *American J. Math. and Mgmt. Sci.*, **6**, 301- 325.

Desrochers, M., Desrosiers, J. and Solomon, M. (1992), A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows, *Opns. Res.*, **40**, 342-354.

Dreyfus, S. and Law, A. (1977), *The Art and Theory of Dynamic Programming*, Academic Press, Inc.

Dumas, Y., J. Desrosiers, Gelinas, E. and Solomon, M. (1995), An Optimal Algorithm for the Traveling Salesman Problem with Time Windows, *Opns. Res.*, **43**, 367- 371.

Gendreau, M., Hertz, A., Laporte, G. and Stan, M. (1998), A Generalized Insertion Heuristic for the Traveling Salesman with Time Windows, *Opns. Res.*, **43**, 330- 335.

Lin, S. and Kernighan, B. W. (1973), An Effective Heuristic Algorithm for the Traveling Salesman Problem, *Opns Res.*, **21**, 498-516.

Mingozzi, A., Bianco, L. and Ricciadelli, S. (1997), Dynamic Programming Strategies for the Traveling Salesman Problem with Time Window and Precedence Constraints, *Opns. Res.*, **45**, 365-377.

Savelsbergh, M. (1985), Local Search in Routing Problems with Time Windows, *Anns. Opns. Res.*, **4**, 285-305.

Solomon, M. M. (1987), Algorithms for the Vehicle and Scheduling Problems with Time Window Constraints, *Opns. Res.*, **5**, 254-265.