Journal of the Korean Institute of Industrial Engineers
Vol. 46, No. 3, pp. 338-348, June 2020.
ISSN 1225-0988 | EISSN 2234-6457
https://doi.org/10.7232/JKIIE.2020.46.3.338

<Research Letter>

시간에 의존하는 이익이 발생하는 외판원문제에 대한 최적화 알고리즘

강문정 · 경지윤 · 이충목†

한국외국어대학교 산업경영공학과

An Algorithm for TSP with Time Dependent Profits

Munjeong Kang · Jiyoon Kyung · Chungmok Lee

Department of Industrial and Management Engineering, Hankuk University of Foreign Studies

In this paper, we introduce a TSP with Time-Dependent Profits (TSPwTDP) which differs from the existing TSPwProfits (Traveling Salesman Problem with Profits) in the sense that the profits depend on the times to visit the nodes. We will demonstrate a real-life example, called a parking lot problem that cannot be solved by the previous models. In the parking lot application, as the time to visit a node is delayed, the probability of taking the empty parking space decreases exponentially. We will present two exact algorithms for solving the problem: The first algorithm is a mixed-integer programming formulation with a nonlinear objective function. The second approach is based on the labeling algorithm. We found that the labeling algorithm can solve problems up to 20 nodes, while the nonlinear formulation failed to solve any problem with more than 4 nodes.

Keywords: TSPwTDP, Nonlinear Integer Programming, Labeling Algorithm Programming

1. 서 론

외판원 문제(Traveling Salesman Problem, TSP)는 가장 대표적인 조합 최적화(combinatorial optimization) 문제 중 하나이며 그동안 매우 많은 연구가 이루어져 왔다(Applegate *et al*., 2007). TSP는 또한 가장 대표적인 NP-hard 문제 중 하나이며(Garey and Johnson, 1979) 다양한 최적화 알고리즘의 성능을 비교하는 대상 문제(benchmark problem) 역할을 하기도 한다. TSP는 일반적으로 노드의 집합 N과 두 개의 노드 간의 거리(또는 시간)를 나타내는 D(또는 T) 행렬(matrix)로 정의되며, 주어진 노드 집합의 모든 노드를 방문하는 가장 짧은 거리(또는 시간)를 찾는 것이 목표이다.

TSP는 다양한 변형 문제들이 존재한다. 이에 대한 변형 문제 중 가장 중요한 문제는 차량경로문제(Vehicle Routing Problem, VRP)이다. VRP는 한 명의 외판원이 아닌 여러 대의 차량이 노드들을 방문하는 것으로 TSP는 VRP의 특별한 경우라 할 수 있다. VRP는 차량의 용량을 고려한 CVRP(Capacitated VRP, Toth *et al*., 2002; Uchoa *et al*., 2017), 다양한 차량의 종류를 고려한 Heterogeneous VRP(Koç *et al*., 2016; Yao *et al*., 2016), 노드를 방문하는 시간대가 정해진 VRPwTW(VRP with Time Windows, Cordeau *et al*., 2000; Miranda and Conceição, 2016), 노드를 방문하는 목적이 배달(delivery)과 수거(pick-up) 두 가지 존재하는 VRPwPD(VRP with Pick-up and Delivery, Desaulniers, 2000; Männel and Bortfeldt, 2016) 등이 있다. 또한 데이터의 불확실성을 고려한 연구로 고객의 수요(demand)의 불확실성을 고려한 VRPSD(VRP with Stochastic Demand, Bertsimas, 1992), 노드와 노드 간의 이동 시간의 변화를 고려한 TDVRP(Time-dependent VRP, Malandraki, 1992; Ritzinger *et al*., 2016), 이동

본 연구는 한국연구재단의 지원을 받아 수행된 연구임(2018R1A1A1A05077775).
본 연구는2019학년도 한국외국어대학교 교내학술 연구비의 지원에 의하여 이루어졌음.
† 연락저자 : 이충목 교수, 경기도 용인시 처인구 모현읍 외대로 81 한국외국어대학교 공학관 530, Tel : 031-330-4378, Fax : 031-330-4120
E-mail : chungmok@hufs.ac.kr
2019년 10월 30일 접수, 2019년 11월 24일 수정본 접수, 2019년 12월 6일 게재 확정.

시간과 고객의 수요에 대한 불확실성을 동시에 고려한 연구(Lee *et al*., 2012)등이 있다. 위에 언급된 다양한 VRP의 변형 문제들에 대해 차량의 대수를 1대로 고정하면 TSP의 변형 문제를 얻을 수 있다. 최근에는 드론(drone), 전기 자동차 등의 새로운 교통수단을 고려한 TSP 변형 문제들이 연구되고 있다. Murray *et al*.(2015)는 외판원과 드론이 동시에 노드를 방문하는 경우를 고려하였다. 이때 드론은 외판원이 방문하지 않는 노드들을 방문하고 이후에 외판원과 합류하여야 한다. 차량을 이용하여 드론의 임시 기지(drone station)를 운영하는 문제(TSP with Drone Station, Kim and Moon, 2018) 또한 최근에 제시되었다. Erdoğan *et al*.(2012)는 전기자동차의 제한된 이동거리를 이용한 차량 경로 문제(Electric-vehicle VRP, EVRP)를 제시하였다. 일반적인 TSP는 외판원의 이동 거리에 제한이 없지만 EVRP의 경우에는 주기적으로 특정 노드(충전 노드)에 방문하여 재충전해야 하는 차이가 있다.

TSP를 풀기 위한 최적화 알고리즘은 최적해를 보장하는지 여부에 따라 크게 두 가지로 구분할 수 있다. 최적해를 보장하는 알고리즘(exact algorithms)에는 혼합 정수 계획법 수리모형을 이용한 방법(Matai, 2010), 분지 절단법(Branch-and-Cut, Padberg *et al*., 1987), 분지 평가법(Branch-and-Price, Fukasawa *et al*., 2006), 동적계획법(dynamic programming, Balas and Simonetti, 2001) 등이 있다. 특히 TSP의 변형 문제들을 다양한 방법을 이용해 원래의 TSP 문제로 변환(transform)하는 방법들도 제시되었다(Dimitrijević and Šarić, 1997). TSP의 최적해 알고리즘 중에 가장 성능이 좋은 것으로 알려진 것은 Concorde(Applegate *et al*., 2006)이다. Concorde는 분지 절단법에 기반하여 다양한 휴리스틱(heuristics)을 적용하여 최대 85,900개 노드 문제의 최적해를 찾을 수 있었다.

최적해를 보장하지 못하는 휴리스틱 알고리즘 역시 매우 많은 연구가 이루어져 왔다. Lin and Kernighan(1973)은 K-L 휴리스틱이라 불리는 경로 개선 방법을 제시하였다. Gendreau *et al*.(1998)는 Tabu 탐색(Tabu Search)기반의 알고리즘을 사용하여 좋은 결과를 얻었다. 이 밖에도 개미 군집 최적화(Ant Colony Optimization, ACO, Branke and Guntsch, 2004; Dorigo and Stützle, 2019), 지역 탐색(Local Search, Gu and Huang, 1994; Lourenço *et al*., 2019), 입자 군집 최적화(Particle Swarm Optimization, PSO, Shen *et al*., 2006; Du and Swamy, 2016) 등이 있다.

본 논문에서 새롭게 제시하는 TSPwTDP(TSP with Time Dependent Profits)는 다음과 같이 정의된다.

TSPwTDP : 주어진 노드의 집합과 각각의 노드를 방문하는 시간에 의존하는 이익 함수(profit function) 그리고 노드와 노드 사이의 이동 시간 행렬이 주어질 때 모든 노드를 방문하면서 이익 함수의 총합과 이동 시간의 총합을 동시에 최적화하는 경로는 무엇인가?

TSPwTDP는 각각의 노드를 방문할 때마다 이익(profits)이 발생한다는 것에서 TSPwP(TSP with Profits, Feillet *et al*., 2005)와 비슷하다. 하지만 일반적인 TSPwP는 노드마다 이익이 상수(constant)로 주어지고 노드를 언제 방문 하는 것과 상관없이 변하지 않는다는 차이가 있다. 따라서 일반적인 TSPwP는 모든 노드를 방문할 필요가 없는데 이는 모든 노드를 방문할 때 이익의 총합이 일정한 상수가 되기 때문이다. 반면에 TSPwTDP는 노드를 방문할 때 발생하는 이익이 노드를 방문하는 시간에 대한 함수로 주어진다. 따라서 TSPwTDP는 TSPwP의 일반 유형(general case)이며 TSPwP와 마찬가지로 NP-hard에 속하는 문제이다. TSPwTDP는 또한 이익의 총합과 이동 시간의 총합, 두 가지 목적함수를 고려한다는 점에서 다중목적함수 최적화 문제(Multi-Objective Optimization Problem)에 해당한다고 볼 수 있다. 본 논문의 기여점은 다음과 같이 정리할 수 있다.

- 최초로 TSPwTDP를 정의하고 수리모형을 제시한다.
- TSPwTDP에 해당하는 현실적인 예를 제시한다.
- TSPwTDP를 풀 수 있는 동적계획법에 기반한 최적화 알고리즘을 제시한다.
- 비선형 수리모형은 노드가 3개보다 큰 문제를 한 시간 내에 풀 수 없음을 발견하였다.
- 동적계획법 알고리즘은 20개 정도의 노드를 가지는 문제를 한 시간 내에 풀 수 있음을 확인하였다.

본 논문은 다음과 같이 구성되어 있다. 제 2장에서는 TSPwTDP에 대해 구체적으로 정의하고 실용적인 응용 예를 제시한다. 제 3장은 본 문제에 대한 수리모형을 제시하고 동적계획법에 기반한 레이블링(labeling) 알고리즘을 제시한다. 제 4장은 제시된 알고리즘의 전산 실험 결과를 제시하고 마지막으로 결론이 제 5장에서 제시된다.

2. 문제 정의

방문할 노드의 집합을 N이라 하자. 집합 N과 별도로 출발 노드(depart node), 도착 노드(return node)를 각각 d, $\hat{d}$이라고 정의하자. N의 어떤 노드 i에 도착하는 시간을 t_i라고 할 때, 이익 함수(profit function) $f(t_i)$가 주어진다고 가정한다. 이익 함수 $f(t_i)$는 그 노드에 도착하는 시간에 의존하는 함수로써 비선형(nonlinear)일 수 있으나 다음의 성질을 만족한다고 가정한다.

(가정 A) $\quad f(t_1) \geq f(t_2), \quad \text{for any } t_1 \leq t_2$

(**가정 A**)는 이익 함수가 노드에 도착하는 시간에 따라 증가하지 않는 함수(non increasing function)임을 의미한다. 즉, 노드에서 발생하는 이익은 노드에 도착하는 시간에 따라 동일하거나 감소한다.

출발 노드에서 시작하여 N에 속하는 모든 노드를 방문하고 도착 노드에 방문하는 모든 방문 순서(permutation)의 집합 S를

정의하자. 어떤 주어진 방문 순서 $\pi \in S$에 대해 방문한 노드의 모든 이익의 총합을 $F(\pi)$라고 하고 이동 시간의 총합을 $T(\pi)$라고 정의하자. 이익의 총합과 이동 시간의 총합 사이의 상대적 중요도(trade-off)를 결정하는 매개변수 λ가 주어진다면 TSPwTDP를 다음과 같이 정의할 수 있다.

$$\text{(TSPwTDP)} \quad \underset{\pi \in S}{\operatorname{argmin}} \{-F(\pi) + \lambda T(\pi)\}$$

매개변수 λ의 값이 매우 커지면(TSPwTDP)는 일반적인 TSP와 동일한 해를 얻는다. 반면에 매개변수 λ의 값이 작아지면 이익의 총합을 최대화하는 해를 얻는다.

2.1 응용 예제

본 장에서는 본 연구에서 제시하는 TSPwTDP의 실제 적용 예를 제시한다. 혼잡한 시간대에 주차장의 빈자리가 얼마 남지 않았고 주차장에서 동시에 여러 차가 주차를 하기 위해 움직이고 있는 상황을 가정해보자. 이 상황에서 운전자의 주된 목적은 빈자리를 하나라도 찾아 주차하는 것이다. 주차장의 빈자리 위치와 두 개의 빈자리 간의 거리가 주어진다고 가정하면 주차장의 빈자리를 방문하는 것을 이익(profits)의 발생으로 정의할 수 있다. 이때 주차장의 빈자리는 항상 비어있지 않고 같은 주차장에서 빈자리를 찾는 다른 차들이 언제든지 선점할 수 있다. 잔여 공석이 얼마 남지 않은 상황에서, 최악의 상황은 운전자가 주차장 내를 배회하다가 한 자리도 주차를 하지 못하는 경우이다. 모든 빈자리의 집합을 N이라 하고 주차장의 어떤 빈자리 $i \in N$가 단위 시간 이후에도 계속 빈자리로 남아 있을 확률을 다음과 같이 정의하자.

$$P(i\text{가 시간 } t+1\text{에 빈자리로 남아 있음}) = P(i\text{가 시간 } t\text{에 빈자리로 남아 있음}) \times p$$

이때, p는 0과 1 사이의 상수로 단위 시간에 빈자리 i가 계속 비어 있을 확률을 의미한다. 위의 정의에 따라 빈자리 i를 시간 t_i에 방문할 때 빈자리 i가 계속 비어 있어 주차를 할 수 있는 확률은 p^{t_i}로 주어짐을 알 수 있다. <Figure 1>은 주차를 할 수 있는 확률 p^{t_i}를 여러 가지 매개변수 p에 대해서 그린 것이다. 매개변수 p의 값이 작아질수록 확률 p^{t_i}가 더욱 빠르게 줄어드는 것을 알 수 있다.

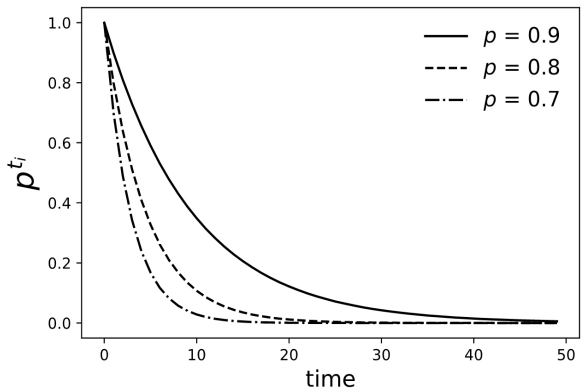

Figure 1. Probability of the Parking Spot being Available at the Time of Arrival Depending on the Parameter

처음에 사용 가능하던 자리에 도착했을 때 그 자리가 사용 가능하지 않으면 다른 비어 있는 곳으로 이동해야 할 것이다. 이때, 노드 i를 방문했을 때 그 자리가 사용 가능하지 않을 확률은 $1-p^{t_i}$로 주어지므로 모든 빈자리를 방문했을 때 빈자리를 하나라도 차지할 수 있는 확률은 다음과 같이 주어진다.

$$\theta := 1 - \prod_{i \in N} (1 - p^{t_i}) \tag{1}$$

즉, θ는 $(t_1, t_2, \cdots, t_{|N|})$에 의존하는 함수이다.

Remark 1. (1)의 함수 θ를 최대화하는 것은 다음의 함수를 최대화하는 것과 같다.

$$\bar{\theta} := -\sum_{i \in N} \log(1 - p^{t_i}) \tag{2}$$

Remark 1이 의미하는 것은 빈자리 노드 i를 시간 t_i에 방문할 때의 이익 함수를 $f(t_i) := -\log(1-p^{t_i})$라고 정의하면 $\bar{\theta}$는 N의 모든 노드를 방문할 때의 이익 함수 $f(t_i)$의 총합으로 주어진다는 것이고, 이는 결과적으로 운전자가 주차장에서 한 자리라도 주차를 할 수 있는 확률을 높이기 위해서는 $\bar{\theta}$를 최대화하는 방문 순서를 정해야 함을 의미한다. 이때 이익 함수(빈자리를 차지할 확률)의 총합은 노드를 방문하는 시간에 대한 비선형 함수로 주어진다. 따라서 단순히 노드를 모두 방문하는 시간의 총합을 최소화하는 것으로는 이익 함수의 총합을 최대화할 수 없다.

<Figure 2>는 이익 함수의 총합과 이동 시간의 총합의 관계에 대한 예제이다. 매개변수 p의 값을 0.7이라 가정하고 출발 노드는 d, 도착 노드는 $\hat{d}$으로 나타내었다. 도착 노드는 가상의 공간이므로 임의의 노드에서 도착 노드까지의 시간은 모두 0으로 가정하였다. 이익 함수의 총합을 최대화하는 방문 순서는 $d \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow \hat{d}$으로 식 (2)를 이용하여 계산한 이익 함수의 총합은 $-\{\log(1-0.7^5) + \log(1-0.7^{5+5}) + \log(1-0.7^{5+5+12})\} \cong 0.213052$이다. 이때 이동 시간의 총합은 5+5+12 = 22이다. 이동 시간의 총합을 최소화하는 방문 순서는 $d \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow \hat{d}$으로 이동 시간의 총합은 8+7+5 = 20이고 이때 이익 함수의 총합은 $-\{\log(1-0.7^8) + \log(1-0.7^{8+7}) + \log(1-0.7^{8+7+5})\} \cong 0.064933$이다. 본 예제에서 보이듯이 이익 함수의 총합을 최대화하는 방문 순서는 이동 시간의 총합을 최소화하는 방문 순서와 다를 수 있음을 알 수 있다.

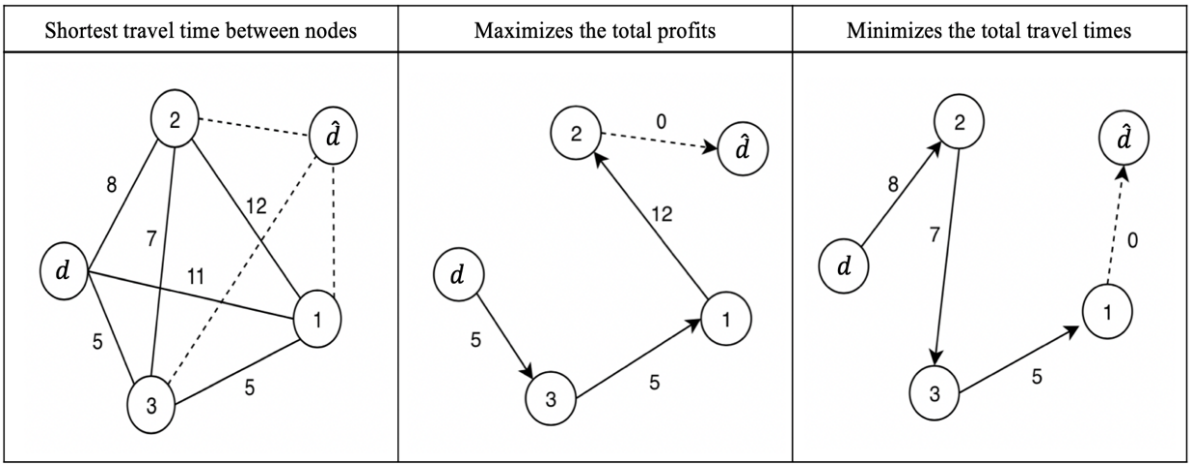

Figure 2. Example of the Relationship between Total Profits and Total Travel Times

3. 알고리즘

본 장에서는 TSPwTDP를 풀기 위한 최적해 알고리즘을 제시한다. 먼저 비선형 정수 계획법(Nonlinear Mixed Integer Programming, NLMIP) 수리모형을 제시한다. 계산 실험 결과(4장)를 보면 NLMIP 수리모형은 아주 작은 크기의 TSPwTDP 문제를 풀기도 쉽지 않다. 따라서 추가적으로 레이블(label) 기반의 최적해 알고리즘을 제시한다. 참고로 이후의 모든 내용은 편의를 위해 2장에서 제시한 응용 예제인 주차장 문제를 대상으로 기술한다. 다른 이익 함수의 경우에도 (**가정 A**)를 만족하는 경우 동일한 알고리즘을 적용할 수 있음을 쉽게 보일 수 있다.

3.1 비선형 정수계획법(Nonlinear Mixed Integer Programming) 수리모형

수리모형에 사용될 매개변수와 결정변수는 다음과 같다.

• 매개변수(Parameter)

λ : 이익 함수의 총합과 이동 시간의 총합의 비중(trade-off)을 결정하는 매개변수 값

N : depot를 제외한 모든 노드(주차장의 빈자리) 집합

d : 출발 노드(depot 또는 주차장의 입구)

$\hat{d}$: 도착 노드(depot, 또는 주차장의 출구)

p : 단위 시간이 지났을 때 빈 주차면이 계속 비어 있을 확률 $(0 < p < 1)$

T_{ij}: i노드에서 j노드까지의 최소 이동 시간

M : Big number

• 결정변수(Decision Variable)

x_{ij} : i노드에서 j노드로 이동하면 1의 값을, 그렇지 않으면 0의 값을 갖는 이진 변수

t_i : i노드에 도착하는 시간

비선형 정수계획법 수리모형(NLMIP)을 다음과 같이 제시한다.

(NLMIP)

$$\min -\left(1-\prod_{i\in N}(1-p^{t_i})\right)+\lambda \sum_{i\in N\cup\{d\},\, j\in N\cup\{\hat{d}\},\, i\neq j} T_{ij}x_{ij} \quad (3)$$

$$\text{s.t} \quad \sum_{j\in N\cup\{\hat{d}\}} x_{dj}=1 \quad (4)$$

$$\sum_{i\in N\cup\{d\},\, i\neq j} x_{ij}=1 \qquad \forall j\in N\cup\{\hat{d}\} \quad (5)$$

$$\sum_{j\in N\cup\{\hat{d}\},\, i\neq j} x_{ij}=\sum_{j\in N\cup\{\hat{d}\},\, i\neq j} x_{ji} \qquad \forall i\in N \quad (6)$$

$$t_j \geq t_i + T_{ij} - M(1-x_{ij}) \qquad \forall i\in N\cup\{d\},\, j\in N\cup\{\hat{d}\},\, i\neq j \quad (7)$$

$$t_d = 0 \quad (8)$$

$$x_{ij}\in\{0,1\} \quad \forall i\in N\cup\{d\},\, j\in N\cup\{\hat{d}\},\, i\neq j \quad (9)$$

목적함수 (3)은 이익 함수의 총합(빈 주차 자리를 하나라도 발견할 확률)과 이동 시간의 총합을 주어진 매개변수인 λ값을 통해 동시에 고려하여 최소화한다. 제약식 (4)는 출발 노드에서 경로가 시작함을 나타낸다. 제약식 (5)는 출발 노드를 제외한 모든 노드에 도착해야 함을 나타낸다. 제약식 (6)은 어느 노드에 도착했으면 다른 노드로 이동해야 함을 나타내는 소위 흐름 균형(flow balance) 제약식이다. 제약식 (7)은 i노드에서 j노드로 이동할 경우 j노드에 도착하는 시간은 i노드에 도착한 시간과 i노드에서 j노드까지의 최소 이동 시간을 더한 것과 같거나 큼을 나타낸다. 제약식 (7)은 각 노드에 도착하는 시간을 결정해 주는 동시에 subtour가 발생하지 않게 하는 역할을 한다. 제약식 (8)은 출발 노드에서의 시간은 0임을 나타낸다. 본 수리모형은 비선형 정수 계획법 Solver인 Couenne(https://www.coin-or.org) 등으로 최적해를 얻을 수 있다.

3.2 레이블링(Labeling) 알고리즘

앞에서 제시한 비선형 수리모형은 문제의 크기가 매우 작지 않으면 일반적인 Nonlinear Integer Solver로 최적해를 얻기 힘들다. 따라서 본 절에서는 레이블링(labeling) 알고리즘을 이용한 최적해 알고리즘을 제시한다. 레이블링 알고리즘은 동적계획법(dynamic programming)의 한 유형으로 주로 네트워크상에서 다양한 제약을 가지는 경로를 찾을 때 많이 사용된다. 특히 차량경로문제(VRP)를 분지평가법(Branch-and-Price)으로 풀 때 열 생성 부문제(column generation subproblem)로 자주 등장한다. 이때 차량의 용량 제약이나 방문하는 노드의 방문 시간 제약과 같이 다양한 자원제약(resource constraints)이 존재할 경우 이를 일반적으로 자원제약 최단경로문제(Shortest Path Problem with Resource Constraints, SPPRC, Irnich *et al*., 2005; Lee *et al*., 2012)라고 한다.

수리모형(NLMIP)은 다음과 같은 두 개의 서로 다른 목적함수를 주어진 λ에 대해 최소화하는 문제로 볼 수 있다.

$$(\mathrm{BP1}) \qquad \min \begin{cases} -\left(1-\prod_{i\in N}(1-p^{t_i})\right) \\ \sum_{i\in N\cup\{d\}, j\in N\cup\{\hat{d}\}, i\neq j} T_{ij}x_{ij} \end{cases} \tag{10}$$

$$\text{s.t} \quad (x,t)\in X \tag{11}$$

이때, X는 제약식 (4)~(9)를 만족하는 유효해(feasible solutions)의 집합이다. 문제(BP1)은 두 개의 목적함수를 가지는 Bi-objective 최적화 문제이며, (NLMIP)의 최적해는 위 문제의 파레토 최적해(Pareto optimal solution)들 중 하나임을 쉽게 알 수 있다. 또한, (BP1)의 첫 번째 목적함수는 **Remark 1**에 의해서 다음과 같이 바꿀 수 있다.

$$(\mathrm{BP2}) \qquad \min \begin{cases} \sum_{i\in N}\log(1-p^{t_i}) \\ \sum_{i\in N\cup\{d\}, j\in N\cup\{\hat{d}\}, i\neq j} T_{ij}x_{ij} \end{cases} \tag{12}$$

$$\text{s.t} \quad (x,t)\in X \tag{13}$$

(BP2)는 (BP1)과 같은 파레토 최적해 집합을 갖는 것을 쉽게 알 수 있다. 하지만 목적함수 (12)는 노드 i를 방문할 때마다 두 개의 목적함수에 각각 특정 값이 더해지는 형태이다. 이는 네트워크상에서 아크(arc) (i,j)를 지날 때마다 비용 벡터(cost vector) $(\log(1-p^{t_{ij}}), t_{ij})$가 더해지는 것으로 볼 수 있다. 따라서 일반적인 Bi-objective 최단경로문제(shortest path problem)의 한 형태임을 알 수 있다. 단지 일반적인 최단 경로 문제와 다르게 모든 노드를 방문해야 하는 제약을 고려하여야 한다. 이를 위해 다음과 같은 계층 네트워크(layered network)를 생각해 보자.

<Figure 3>에서 보이듯이 계층 네트워크는 노드 집합의 크기 $|N|$만큼의 계층을 가지며 각각의 계층은 또한 $|N|$개의 노드를 가진다. 계층 네트워크의 아크는 k번째 계층의 노드 i와 $k+1$번째 계층의 노드 j가 $i\neq j$일 때만 연결한다. 위와 같이 구성된 계층 네트워크에서 출발 노드와 도착 노드를 연결하는 경로는 원래 문제에서의 모든 노드를 방문하는 투어(tour)들을 모두 포함하는 것을 쉽게 알 수 있다. 즉, 계층 네트워크에서 k번째 계층에 있는 노드 i를 지나는 것은 노드 i가 k번째 방문임을 의미한다.

계층 네트워크에서 어떤 노드에 속한 하나의 레이블은 그 노드까지의 부분 경로(partial path)의 비용 벡터의 합을 표현하여야 한다. 계층 k의 어떤 노드까지의 부분 경로 $q:=(d, i_{(1)}, i_{(2)}, \cdots, i_{(k)})$를 생각해 보자. 이 부분 경로는 출발 노드 d에서 시작하며 $i_{(l)}$은 이 부분 경로가 지난 l번째 노드를 의미한다. 부분 경로 q에 대한 레이블을 다음과 같이 정의한다.

$$[\pi, \tau, v_1, v_2, \cdots, v_n]$$

그리고 레이블의 각 구성요소의 정의는 다음과 같다.

$$\pi := \sum_{i=i_{(1)}, i_{(2)}, \cdots, i_{(k)}} \log(1-p^{t_i})$$

$$\tau := T_{di_{(1)}} + \sum_{(i,j)=(i_{(1)}, i_{(2)}), (i_{(2)}, i_{(3)}), \cdots, (i_{(k-1)}, i_{(k)})} T_{ij}$$

$$v_i := 1 \text{ if } q \text{ passes node } i, \text{ 0 otherwise, } \forall i\in N$$

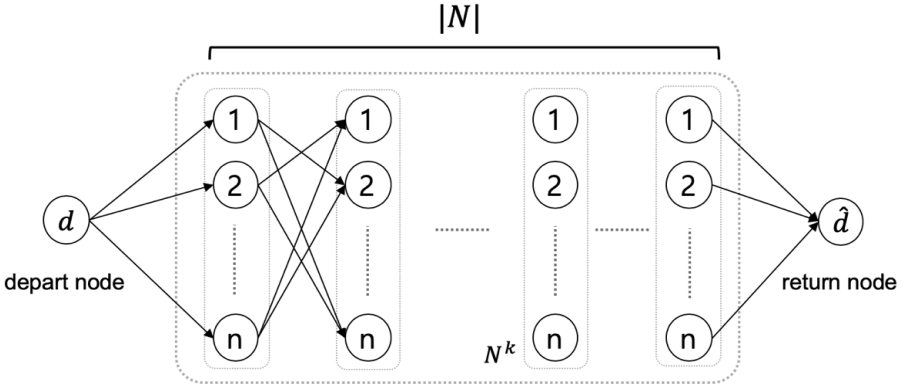

Figure 3. Example of a Layered Network for the Labeling Algorithm

즉, π와 τ는 각각 부분 경로의 비용 벡터의 합을 표현하며 v_i는 부분 경로가 노드 i를 지났는지를 나타낸다. 레이블링 알고리즘은 확장규칙(extension rule)과 지배규칙(dominance rule)이 필요하다. 본 연구에서는 TSPwTDP에 대해 다음과 같은 확장규칙과 지배규칙을 제안한다.

• 확장규칙(extension rule)

계층 네트워크의 계층 k의 어떤 노드 i의 레이블 $l:=[\pi, \tau, v_1, v_2, \cdots, v_n]$은 다음과 같은 확장규칙을 가진다.

- $k<|N|$일 경우 : $v_j=0$인 모든 j에 대해, 계층 $k+1$의 노드 j로 다음과 같이 확장된다.

$$[\pi+\log(1-p^{\tau+T_{ij}}),\ \tau+T_{ij},\ v_1, v_2, \cdots, v_j+1, \cdots, v_n]$$

- $k=|N|$일 경우 : 도착 노드 $\hat{d}$로 다음과 같이 확장된다.

$$[\pi, \tau, v_1, v_2, \cdots, v_j, \cdots, v_n]$$

본 확장규칙에 의하면 레이블은 이미 방문하지 않은 노드($v_j=0$)들로만 확장이 가능하다. 계층 네트워크는 모두 $|N|$개의 계층을 가지고 있기 때문에 계층 네트워크에서 도착 노드 $\hat{d}$로 확장된 모든 레이블은 N에 속하는 모든 노드를 방문한 경로(complete tour)를 의미한다.

• 지배규칙(dominance rule)

계층 네트워크의 어떤 노드에서 끝나는 두 개의 부분 경로 q와 r을 생각해보자. 부분 경로 q와 r로 만들어지는 모든 전체 경로(complete path)의 집합을 $E(q)$와 $E(r)$이라고 하자. 임의의 전체 경로 q^*에 대해 목적함수 (12)의 두 개의 함수의 값을 각각 $Z_1(q^*)$과 $Z_2(q^*)$라고 표시하자. 이때, 부분 경로 q와 부분 경로 r이 다음의 조건을 만족하면, q는 r을 지배(dominate)한다고 정의한다.

- $E(r)\subseteq E(q)$
- $\exists q^*\in E(q) | Z_1(q^*)\le Z_1(r^*)$ 그리고 $Z_2(q^*)\le Z_2(r^*)$, $\forall r^*\in E(r)$

즉, $E(q)$가 $E(r)$을 포함하고, $E(r)$에 속하는 임의의 전체 경로보다 (12)의 두 목적함수가 모두 같거나 작은 전체 경로가 $E(q)$에 포함되어 있을 때, 부분 경로 q는 부분 경로 r을 지배(dominate)한다고 정의한다. 부분 경로 q와 r에 해당하는 레이블을 다음과 같이 정의한다.

$$l_q := [\pi^q, \tau^q, v_1^q, v_2^q, \cdots, v_n^q]$$
$$l_r := [\pi^r, \tau^r, v_1^r, v_2^r, \cdots, v_n^r]$$

정리 1. 계층 네트워크에서의 같은 노드에 속한 두 개의 레이블 l_q와 l_r이 다음의 조건을 만족하거나 만족할 때만(if and only if) l_q는 l_r을 지배(dominate)한다.

$$\pi^q\le\pi^r,\ \tau^q\le\tau^r, \text{ 그리고 } v_i^q=v_i^r\ \forall i\in N$$

증명. 충분조건 : 위의 조건을 만족하는 두 레이블 l_q와 l_r에 대한 부분 경로 q와 r에서 확장되는 모든 레이블의 집합을 각각 $E(q)$와 $E(r)$로 정의하자. 확장 규칙에 의해서 $E(r)$의 모든 레이블에 대해서 항상 위의 지배 조건을 만족하는 또 다른 레이블이 $E(q)$에 존재함을 쉽게 알 수 있다. 필요조건 : 어떤 레이블 l_r과 l_r을 지배하는 또 다른 레이블 l_q를 생각하자. l_q는 l_r을 지배하기 때문에 $E(r)\subseteq E(q)$이 성립한다. 즉, $v_i^q\le v_i^r,\ \forall i\in N$이 성립함을 알 수 있다. 그리고 레이블 l_q와 l_r은 계층 네트워크에서 같은 노드에 속해 있기 때문에 같은 개수의 노드를 거쳐 온 부분 경로들이고 따라서 $\Sigma_{i\in N} v_i^q=\Sigma_{i\in N} v_i^r$이 성립한다. 즉, $v_i^q=v_i^r,\ \forall i\in N$이 됨을 알 수 있다. 또한, 지배의 정의에 의해서 $\pi^{\hat{q}}\le\pi^{\hat{r}},\ \tau^{\hat{q}}\le\tau^{\hat{r}},\ \forall \hat{q}\in E(q), \hat{r}\in E(r)$이 성립한다. 더욱이 (**가정 A**)에 의해서 $\pi^q\le\pi^r,\ \tau^q\le\tau^r$이 성립한다. □

레이블링 알고리즘의 성능은 네트워크 노드에 속한 레이블의 개수에 크게 의존한다. **정리 1**을 이용하여 오로지 지배하는 레이블(dominating labels)만 남기고 나머지 레이블을 삭제할 수 있다. 두 개의 레이블의 집합을 합쳐 지배하는 레이블만 가지는 하나의 집합을 만드는 과정을 일반적으로 결합(merge)이라고 한다. 계층 네트워크상의 노드 i를 마지막 노드로 가지는 부분 경로들을 표현하는 레이블들의 집합을 L_i라고 정의하자. L_i에 속한 레이블들을 다음 계층의 노드 j의 레이블들의 집합 L_j로 확장할 때 **정리 1**의 조건 중에 $v_i^q=v_i^r,\ \forall i\in N$을 만족하는 모든 레이블 $l_q\in L_i, l_r\in L_j$끼리만 결합 과정을 실행하여야 한다. 노드에 속한 레이블 중에 결합을 시도할 수 있는 레이블의 집합을 빠르게 뽑아내기 위해 본 연구는 다음과 같은 방법을 사용하였다. 각 노드에 해시멥(hashmap) 자료구조를 생성한 후 레이블 집합 L의 모든 레이블을 다음과 같이 같은 Key 값으로 추가한다.

$$\text{Key}(l_q) := \sum_{i\in N} 2^{i-1} v_i^q,\ \forall l_q\in L$$

즉, 위 Key 값은 각 레이블이 거쳐 온 부분 경로의 노드 집합으로 유일하게(uniquely) 정의되며 노드의 해시멥이 같은 Key값으로 저장된 레이블들은 서로 결합을 실행할 수 있다. 헤시멥의 Key를 찾는 작업은 생성된 Key 값의 헤시(hash)가 같지 않은 경우 $O(1)$의 시간 복잡도를 가지기 때문에 매우 빠른 속도로 결합이 가능한 두 개의 레이블 집합을 가져올 수 있다. 이처럼 레이블의 집합을 Key에 따라 나누는(partitioning) 방법은 또 하나의 장점을 가지는데 그것은 레이블의 정의에서 Key를 정의하는 부분을 제거할 수 있다는 것이다. 새로운 레이블을 다음과 같이 정의한다.

$$[\pi, \tau, u]$$

이때, u는 레이블이 확장된 바로 이전 노드를 의미하며 레이블링 알고리즘이 종료된 후 경로를 재구성(path backtracking)할 때 사용된다. 새로운 레이블의 정의는 기존 정의보다 레이블 하나당 훨씬 작은 메모리를 차지한다($3<2+|N|$). 새로운

레이블의 확장규칙과 지배규칙은 원래의 레이블 정의를 사용할 때와 근본적으로 동일하며, 하나의 차이점은 레이블이 이미 방문한 노드 정보($[v_i]_{\forall i \in N}$)가 레이블에서 레이블이 저장된 헤시멥의 Key로 옮겨갔다는 것이다.

• 결합 작업(Merging operation)

각각 M개와 N개의 레이블을 가지고 있는 두 개의 레이블링 집합에 대한 단순한 결합 작업은 모든 레이블의 조합에 대해 지배규칙을 확인해야 하므로 $O(M \times N)$의 시간 복잡도를 가진다. 본 연구에서는 $O(M+N)$의 시간 복잡도를 가지는 Brumbaugh-Smith and Shier(1989)의 개선된 결합 알고리즘을 사용하였다.

• TSPwTDP를 위한 레이블링 알고리즘

다음은 위에서 설명한 레이블링 알고리즘을 정리한 것이다. 먼저 알고리즘에서 사용할 집합과 매개변수를 다음과 같이 정의한다. 알고리즘에서는 편의를 위해 d와 $\hat{d}$을 각각 0, $n+1$로 표기한다.

L_i^k : k번째 계층, i번째 노드에서 종료되는 모든 부분 경로에 해당하는 레이블들의 집합

L^0, L^{n+1} : d와 $\hat{d}$의 레이블들의 집합

K_i^k : k번째 계층, i번째 노드에서 종료되는 모든 부분 경로에 해당하는 레이블들의 Key들의 집합

$L_i^k[key]$: L_i^k의 레이블 중에 key값에 해당하는 레이블들의 집합(hashmap)

Treat(key): key값을 이진법으로 표시했을 때 0에 해당하는 노드들의 집합이다. 즉, 확장 가능한 노드들의 집합이다.

Extend$(L_i^k[key, j])$: 확장 규칙에 따라 $L_i^k[key]$의 레이블들을 $k+1$ 계층의 j 노드로 확장한다.

B(key, j) : key값을 이진법으로 표시했을 때 j노드로 이동한 경우 j번째 자리의 수를 1로 바꾼 새로운 key값이다.

Merge$(L_j^{k+1}[newkey], \hat{L})$: 두 개의 레이블의 집합 중 다른 레이블들에 지배(dominated)되지 않은 레이블만을 반환한다.

<Figure 4>는 최종 레이블링 알고리즘의 pseudocode를 보여준다. 레이블링 알고리즘이 종료되었을 때 도착 노드 $\hat{d}$에 남아 있는 모든 레이블은 파레토 최적해에 해당한다. 즉, (BP2) 문제의 파레토 최적해의 집합임을 알 수 있다. (BP2)의 목적함수의 값은 간단한 계산을 통해(BP1)의 목적함수의 값으로 바꿀 수 있고 (BP1)의 두 개의 목적함수 값을 λ로 결합하면 TSPwTDP의 최적해를 얻을 수 있다.

Algorithm 1 Labeling algorithm for solving TSP with Time Dependent Profits

$L^0 \leftarrow \{[0,0,0]\}$
for all $k = 1, ..., n$ **do**
  for all $i = 1, ..., n$ **do**
    $K_i^k, L_i^k \leftarrow \emptyset$
  end for
end for
$L^{n+1} \leftarrow \emptyset$

for all $i = 1, ..., n$ **do**
  $\hat{L} \leftarrow$ **Extend**(L^0, i)
  $newkey \leftarrow$ **B**$(0, i)$
  $K_i^1 \leftarrow \{newkey\} \cup K_i^1$
  $L_i^1[newkey] \leftarrow$ **Merge**$(L_i^1[newkey], \hat{L})$
end for

for all $k = 1, ..., n-1$ **do**
  for all $i = 1, ..., n$ **do**
    for all $key \in K_i^k$ **do**
      $treat \leftarrow$ **Treat**(key)
      repeat
        Choose $j \in treat$
        $\hat{L} \leftarrow$ **Extend**$(L_i^k[key], j)$
        $newkey \leftarrow$ **B**(key, j)
        $K_j^{k+1} \leftarrow \{newkey\} \cup K_j^{k+1}$
        $L_j^{k+1}[newkey] \leftarrow$ **Merge**$(L_j^{k+1}[newkey], \hat{L})$
      until $treat = \emptyset$
    end for
  end for
end for

for all $i = 1, ..., n$ **do**
  for all $key \in K_i^n$ **do**
    $\hat{L} \leftarrow$ **Extend**$(L_i^n[key], n+1)$
    $L^{n+1} \leftarrow$ **Merge**$(L^{n+1}, \hat{L})$
  end for
end for
return L^{n+1}

Figure 4. Pseudocode of Labeling Algorithm for TSPwTDP.

4. 전산 실험

모든 전산 실험은 MacPro, 3.5GHz 6-Core Intel Xeon E5 프로세서, 32GB 메모리 환경에서 수행되었다. 실험에는 두 가지 알고리즘이 사용되었다. 첫 번째는 3.1절에서 제시한 NLMIP 수리모형을 이용한 것(이하 NLMIP)이며 두 번째는 3.2절에서 제시한 레이블링 알고리즘이다. NLMIP 실험의 경우 nonlinear solver인 Couenne를 사용하여 Python으로 구현되었으며, 레이블링 알고리즘 실험의 경우 numba(http://numba.pydata.org/)를 이용하여 역시 Python으로 구현되었다.

4.1 실험 대상 문제 생성

실험에 사용된 문제는 다음과 같이 생성하였다. 주어진 노드의 집합 N에 대해서 $T_{ij} = random(1,15)$로 정의하였다. 즉, i 노드에서 j노드로 이동하는 시간은 1~15의 임의의 정수 값을 가진다는 의미이다. 노드를 방문할 때 이익 함수를 결정하는 매개변수 p는 0.7, 0.8, 0.9를 사용하였다.

4.2 실험 결과

<Table 1>은 같은 문제를 대상으로 Couenne를 이용하여 NLMIP 수리모형을 푼 결과와 레이블링 알고리즘의 풀이 결과를 비교한다. 수리모형(NLMIP)의 경우 $\lambda = 0.1$을 사용하였다. Table 1의 문제의 경우, 이익함수의 총합은 확률값이므로 0-1 사이 값이고 이동 시간의 총합은 최대 대략 수십(30~45) 사이의 값을 가졌다. 따라서 λ의 값은 두 목적함수가 모두 의미 있게 반영될 수 있도록 설정하였다. 레이블링 알고리즘은 모든 파레토 최적해를 구해주기 때문에 레이블링 알고리즘이 종료될 때까지의 시간을 비교하였다. Couenne의 경우 매우 작은 문제(노드 3개)에서도 제한 시간(3600초) 동안 최적해를 찾지 못하는 경우가 많았으며 최적이라고 나온 솔루션도 레이블링 알고리즘과 비교했을 때 좋지 않은 경우도 발생했다. 이는 비선형 문제를 풀 때 생기는 계산오차(round-off error) 때문으로 보인다. Couenne의 경우 노드의 개수를 3개에서 4개로 증가시킬 경우 15번의 실험 모두 제한 시간 안에 최적해를 찾지 못하였다.

Couenne는 노드의 수가 적음에도 불구하고 문제에 따라 계산 시간이 크게 차이 나는 것도 확인할 수 있었다. 그에 비해 레이블링 알고리즘은 모든 문제에 대해서 매우 짧은 시간 안에 최적해를 찾았다. Couenne는 노드의 개수가 조금만 늘어나도 최적해를 제한 시간 안에 찾지 못하므로 레이블링 알고리즘으로 노드의 수를 늘리면서 p값을 변화시키며 실험하였다. <Table 2>에 제시된 결과는 각각의 노드 개수에 대해 30개의 랜덤 문제를 생성하고 레이블링 알고리즘으로 푼 결과값의 평균을 보여준다. # labels는 도착 노드 $\hat{d}$에서 남아있는 레이블의 개수를 의미한다. 즉, # labels는 모든 파레토 최적해의 개수라고 할 수 있다.

Table 1. Comparison of Computational Times for the Problems with $|N| = 3$

problem	p = 0.7		p = 0.8		p = 0.9	
	NLMIP	Labeling	NLMIP	Labeling	NLMIP	Labeling
1	120.39	0.0007	3600*	0.0007	179.19	0.0009
2	3600*	0.0006	3600*	0.0006	3600*	0.0005
3	3600*	0.0006	3600*	0.0006	3600*	0.0006
4	3600*	0.0009	3600*	0.0007	3600*	0.0006
5	122.21	0.0006	5.83	0.0006	1.35	0.0006
6	11.77	0.0006	3600*	0.0006	3600*	0.0005
7	3600*	0.0007	31.64	0.0006	3600*	0.0006
8	9.05	0.0006	3600*	0.0006	2.71	0.0005
9	7.23	0.0006	3600*	0.0006	3600*	0.0009
10	3600*	0.0006	3600*	0.0006	3600*	0.0006
11	5.69	0.0006	39.58	0.0009	3600*	0.0006
12	3600*	0.0006	3600*	0.0006	3600*	0.0005
13	3600*	0.0007	3600*	0.0006	3600*	0.0006
14	3600*	0.0006	10.81	0.0006	0.39	0.0006
15	3600*	0.0006	3600*	0.0006	3600*	0.0006

Table 2. Computational Results of the Labeling Algorithm for the Problems with $3 \le |N| \le 20$.

$\|N\|$	p = 0.7		p = 0.8		p = 0.9	
	time(s)	#labels	time(s)	#labels	time(s)	#labels
3	< 0.01	1.26	< 0.01	1.26	< 0.01	1.23
4	< 0.01	1.66	< 0.01	1.70	< 0.01	1.66
5	< 0.01	1.73	< 0.01	1.73	< 0.01	1.73
6	0.01	1.93	0.01	1.93	0.01	1.80
7	0.04	2.43	0.04	2.63	0.04	2.56
8	0.12	3.26	0.12	3.36	0.12	3.03
9	0.31	3.60	0.32	4.00	0.32	3.70
10	0.80	4.10	0.80	4.16	0.81	4.33
11	1.96	4.53	1.94	4.60	1.97	4.30
12	4.62	5.23	4.65	5.80	4.68	5.40
13	10.84	5.33	10.89	5.63	10.90	4.83
14	25.66	6.00	25.59	6.36	25.34	6.10
15	58.46	5.93	58.55	6.50	58.81	6.03
16	134.54	6.76	135.25	7.30	134.14	7.10
17	303.52	6.46	304.08	7.43	303.19	7.53
18	682.43	8.10	681.60	9.10	685.55	7.83
19	1535.60	8.00	1533.24	8.86	1529.22	8.60
20	3700.76	8.93	3649.85	10.36	3444.93	9.30

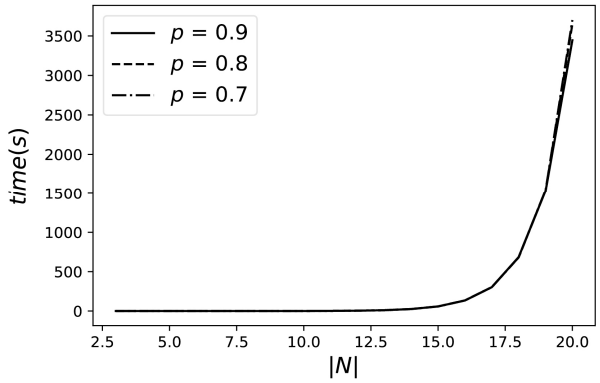

Figure 5. Solving Times by Number of Nodes Depending on the Parameter p

노드의 개수가 늘어날수록 확장되는 레이블의 수가 점점 증가하기 때문에 풀이 시간이 급격하게 증가한다. <Figure 5>는 <Table 2>의 결과를 그래프로 나타낸 것이고 이를 통해 노드의 개수가 늘어남에 따라 풀이 시간이 급격하게 증가하는 것을 확인할 수 있다. 하지만, p값에 따른 실행 시간의 차이는 확인할 수 없었다. 레이블링 알고리즘의 성능은 정리 1의 지배규칙을 통해 얼마나 많은 레이블을 제거할 수 있느냐에 크게 의존한다. 본 연구에 사용된 목적함수 (12)의 경우 어떤 두 개의 레이블에 대해서 p값이 달라지더라도 둘 간의 지배 관계는 변하지 않는다. 즉, p값이 $0<p<1$을 만족하는 경우 지배규칙으로 제거되는 레이블은 항상 동일하다. 따라서 <Figure 5>와 같이 거의 동일한 실행 시간을 얻게 된다.

NLMIP는 노드의 개수가 조금만 늘어나도 최적해를 구하기 어렵기 때문에 레이블링 알고리즘과 해의 품질을 비교하기 어렵다. 따라서 잘 알려져 있는 TSP휴리스틱 알고리즘인 Closest Neighborhood(CN)와 성능을 비교하였다. 본 연구에서는 이익함수 값과 이동 시간을 함께 고려한다. 따라서 CN에서 가장 가까운 노드를 정하는 기준은 이익함수 값과 이동 시간을 더한 값이 가장 작은 경우라고 하였다. 레이블링 알고리즘은 모든 파레토 최적해를 찾기 때문에 여러 개의 해가 존재하므로 CN에서 λ는 0.01에서 0.1까지의 값을 10개로 나누어 사용하였다. 즉, 각 λ값에 해당하는 CN의 해와 레이블링 알고리즘의 해를 비교하여 성능을 비교하였다. <Table 3>은 $p=0.7$인 같은 문제에 대하여 CN과 레이블링 알고리즘의 성능을 비교한 표이다. <Table 2>에 사용된 문제 중에 각 크기별 첫 번째 문제를 대상으로 10가지 λ값을 바꿔가면서 CN 알고리즘을 수행하였다. <Table 3>의 "avg obj."와 "avg time(s)"은 각각 10가지 λ값에 대한 목적함수 평균과 실행 시간(초)의 평균을 의미한다. 또 "avg obj. GAP (%)"은 각각의 λ에 대해서 목적함수의 GAP(|Obj. of Labeling algorithm-Obj. of CN|/(Obj. of Labeling algorithm)×100)의 평균을 표시한다. 표에서 볼 수 있듯이 CN은 매우 빠른 시간 안에 유효(feasible)한 솔루션을 찾아내지만, 해의 품질은 레이블링 알고리즘보다 굉장히 떨어지는 것을 알 수 있다.

Table 3. Comparison of Results between the Labeling Algorithm and the CN Heuristics.

$\|N\|$	CN		Labeling algorithm		avg obj. GAP (%)
	avg obj.	avg time (s)	avg obj.	avg time (s)	
3	0.14	< 0.01	0.14	< 0.01	0.00%
4	0.87	< 0.01	0.80	< 0.01	5.43%
5	1.21	< 0.01	1.21	< 0.01	0.00%
6	0.63	< 0.01	0.44	0.01	57.83%
7	1.35	< 0.01	1.14	0.04	14.77%
8	1.01	< 0.01	0.63	0.12	76.43%
9	0.69	< 0.01	0.30	0.31	105.52%
10	1.11	< 0.01	1.05	0.75	8.64%
11	0.41	< 0.01	0.15	1.93	92.69%
12	1.45	< 0.01	0.98	4.61	75.24%
13	0.64	< 0.01	0.46	10.75	32.27%
14	0.98	< 0.01	0.39	25.28	123.30%
15	1.89	< 0.01	0.95	57.58	110.41%
16	1.58	< 0.01	1.29	137.67	23.76%
17	1.26	< 0.01	0.88	299.72	53.46%
18	1.40	< 0.01	0.78	673.46	103.88%
19	1.33	< 0.01	0.63	1516.81	251.03%
20	1.74	< 0.01	0.89	3658.42	111.62%

파레토 최적해의 개수도 노드의 개수가 증가할수록 더 증가한다. 레이블링 알고리즘의 또 하나의 장점은 한꺼번에 모든 파레토 최적해를 구할 수 있다는 것이다. 반면에 수리모형(NLMIP)은 주어진 λ값에 대해 하나의 파레토 최적해를 얻게 된다. 즉, 레이블링 알고리즘으로 얻은 파레토 최적해의 집합은 모든 경우의 λ에 대한 최적해를 완전히 포함한다.

<Figure 6>은 같은 문제($|N=20|$)에 대해 다른 p값을 사용하여 얻은 모든 파레토 최적해를 보여준다. 그래프에서 오른쪽으로 갈수록 식 (10)의 첫 번째 목적함수가 더 좋은 값을 가지는 것을 의미한다. 수직 방향으로는 아래쪽으로 갈수록 총 이동 시간(또는 거리)이 작은 것을 의미한다. 각각의 파레토 최적해들은 서로 지배되지 않기 때문에 어떤 해가 더 좋다고 말할 수 없다. 만약 이익(profit)을 더 중요하게 생각한다면 오른쪽 위에 있는 점들을 선택하고 총 이동 시간을 더 중요하게 생각한다면 왼쪽 아래에 있는 점들을 선택하게 될 것이다. 이는 목적함수 (3)의 λ값을 조절함에 따라 어떤 요소를 더 중요하게 고려할 것인지에 대한 문제와 같다. λ값을 줄이면 위 그래프의 파레토 최적해 중에 오른쪽 위에 있는 해들을 선택하고 λ값을 늘리면 왼쪽 아래에 있는 해들을 선택하게 된다.

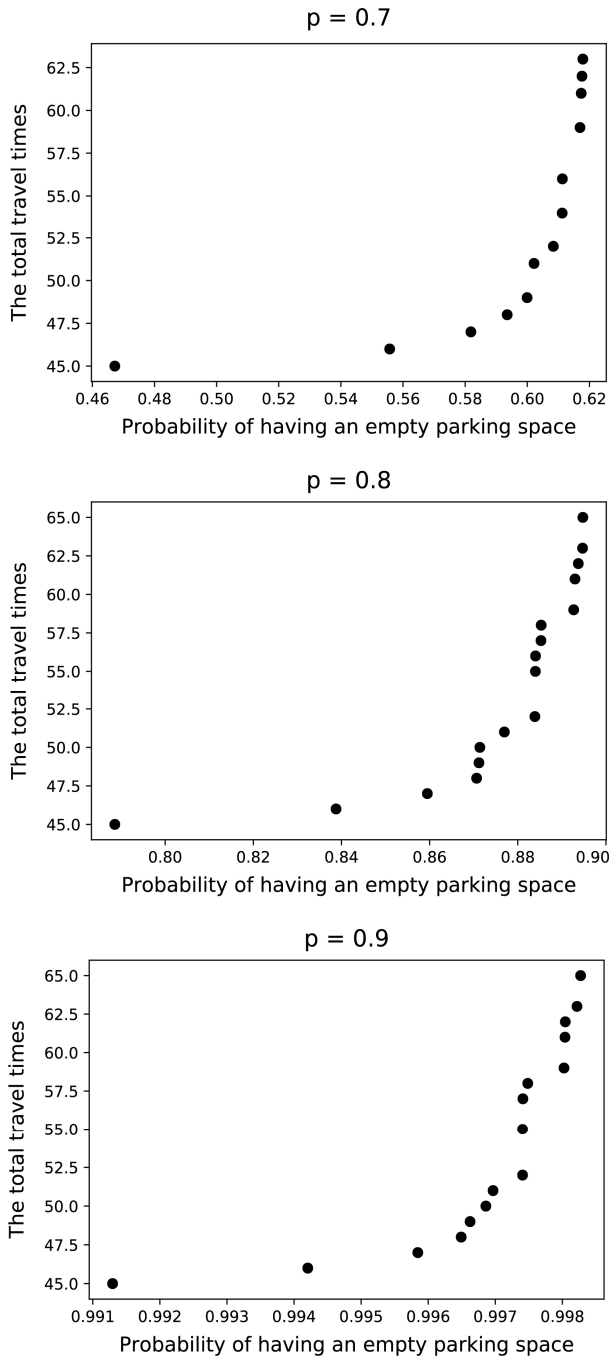

Figure 6. Plots of Pareto Optimal Solutions with Different Values of p

5. 결 론

본 연구는 도착 시간에 의존하는 이익 함수를 가지는 TSP문제를 새롭게 정의하고 이를 TSPwTDP로 정의하였다. TSPwTDP에 대한 실용적인 응용 예를 제시하고 이 응용문제에 대한 비선형 정수 계획법 수리모형을 제시하였다. 제시된 수리모형을 직접 Couenne와 같은 비선형 solver로 푸는 것은 효율적이지 못하다. 따라서 레이블링 알고리즘 기반의 최적해 알고리즘을 개발하였다. 전산 실험 결과 레이블링 알고리즘은 노드의 개수가 약 20개 가량의 문제의 최적해를 제한 시간 내에 찾을 수 있었다.

본 연구에서 제시한 레이블링 알고리즘은 두 개의 목적함수를 가지는 일반적인 TSP문제에 적용할 수 있다. 하지만 문제의 크기가 커질수록 계층 네트워크의 크기가 커지고, 레이블의 개수가 빠르게 증가하는 단점이 있다. 레이블링 알고리즘의 속도를 높이는 방법으로 계층 네트워크를 몇 가지 단계로 나누고 각각의 단계를 독립적으로 레이블링 알고리즘을 실행한 후 생성된 레이블들을 연결하는 방법을 생각해 볼 수 있다. 또한, 본 연구에서 제시한 주차장 문제의 경우 자리를 하나 이상 찾을 확률이 아니라 주차 시간의 기댓값을 최소화하는 형태로 접근할 수 있다. 이 경우에는 수리모형의 목적함수가 $\Sigma_{i\in N} t_i p^{t_i}$와 같은 항을 포함하게 될 것이다.

본 연구는 기본적으로 한 명의 외판원(또는 차량)을 가정한다. 그러나 본 연구가 제시하는 시간에 따라 달라지는 이익은 여러 대의 차량을 가정한 상황에서도 적용될 수 있다. 여러 대의 차량(또는 외판원)이 존재하는 경우는 기존 연구에서 일반적으로 Multiple TSP라 불린다(Bektas, 2006). 이 경우에는 하나의 차량이 모든 노드를 방문하는 것이 아니기 때문에 본 연구에서 제시한 계층 네트워크를 이용한 레이블링 알고리즘을 적용하기 힘들다. 또한, 차량경로문제에서 사용되는 여러 가지 추가적인 제약(차량 용량, 방문 시간대 등)을 반영하기도 쉽지 않다. 따라서, 기존에 제시된 다양한 차량경로문제에 시간에 의존하는 이익을 추가로 고려하는 것을 생각해 볼 수 있고 이것들을 향후 연구과제로 제시한다.

참고문헌

Applegate, D. L., Bixby, R. E., Chvatal, V., and Cook, W. J. (2006), *The Traveling Salesman Problem : A Computational Study*, Princeton university press.

Applegate, D. L., Bixby, R. E., Chvatal, V., and Cook, W. J. (2006), Concorde TSP solver.

Balas, E. and Simonetti, N. (2001), Linear time dynamic-programming algorithms for new classes of restricted TSPs : A computational study, *INFORMS journal on Computing*, **13**(1), 56-75.

Bektas, T. (2006), The multiple traveling salesman problem : an overview of formulations and solution procedures, *Omega*, **34**(3), 209-219.

Bertsimas, D. J. (1992), A vehicle routing problem with stochastic demand, *Operations Research*, **40**(3), 574-585.

Branke, J. and Guntsch, M. (2004), Solving the probabilistic TSP with ant colony optimization, *Journal of Mathematical Modelling and Algorithms*, **3**(4), 403-425.

Brumbaugh-Smith, J. and Shier, D. (1989), An empirical investigation of some bicriterion shortest path algorithms, *European Journal of Operational Research*, **43**(2), 216-224.

Cordeau, J. F. and Groupe d'études et de recherche en analyse des décisions (Montréal, Québec) (2000), *The VRP with time windows*, Montréal : Groupe d'études et de recherche en analyse des décisions.

Desaulniers, G. and Groupe d'études et de recherche en analyse des décisions (Montréal, Québec) (2000), *The VRP with pickup and delivery*, Groupe d'études et de recherche en analyse des décisions.

Dimitrijević, V. and Šarić, Z. (1997), An efficient transformation of the generalized traveling salesman problem into the traveling salesman problem on digraphs, *Information Sciences*, **102**(1-4), 105-110.

Dorigo, M. and Stützle, T. (2019), Ant colony optimization : overview and recent advances, In *Handbook of Metaheuristics* (pp. 311-351), Springer, Cham.

Du, K. L. and Swamy, M. N. S. (2016), Particle swarm optimization, *In Search and optimization by metaheuristics* (pp. 153-173), Birkhäuser, Cham.

Erdoğan, S. and Miller-Hooks, E. (2012), A green vehicle routing problem, *Transportation Research Part E : Logistics and Transportation Review*, **48**(1), 100-114.

Feillet, D., Dejax, P., and Gendreau, M. (2005), Traveling Salesman Problems with Profits, *Transportation Science*, **39**(2), 188-205.

Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, M. P., Reis, M., Uchoa, E., and Werneck, R. F. (2006), Robust branch-and-cut-and-price for the capacitated vehicle routing problem, *Mathematical Programming*, **106**(3), 491-511.

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

Gendreau, M., Laporte, G., and Semet, F. (1998), A tabu search heuristic for the undirected selective travelling salesman problem, *European Journal of Operational Research*, **106**(2-3), 539-545.

Gu, J. and Huang, X. (1994), Efficient local search with search space smoothing : A case study of the traveling salesman problem (TSP), *IEEE Transactions on Systems, Man, and Cybernetics*, **24**(5), 728-735.

Irnich, S. and Desaulniers, G. (2005), Shortest path problems with resource constraints, *In Column generation*, Springer, Boston, MA, 33-65.

Kim, S. and Moon, I. (2018), Traveling salesman problem with a drone station, *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, **49**(1), 42-52.

Koç, Ç., Bektaş, T., Jabali, O., and Laporte, G. (2016), Thirty years of heterogeneous vehicle routing, *European Journal of Operational Research*, **249**(1), 1-21.

Lee, C., Lee, K., and Park, S. (2012), Robust vehicle routing problem with deadlines and travel time/demand uncertainty, *Journal of the Operational Research Society*, **63**(9), 1294-1306.

Lin, S. and Kernighan, B. W. (1973), An effective heuristic algorithm for the traveling-salesman problem, *Operations Research*, **21**(2), 498-516.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2019), Iterated local search: Framework and applications, *In Handbook of Metaheuristics* (pp. 129-168), Springer, Cham.

Malandraki, C. and Daskin, M. S. (1992), Time dependent vehicle routing problems : Formulations, properties and heuristic algorithms. *Transportation Science*, **26**(3), 185-200.

Männel, D. and Bortfeldt, A. (2016), A hybrid algorithm for the vehicle routing problem with pickup and delivery and three-dimensional loading constraints, European *Journal of Operational Research*, **254**(3), 840-858.

Matai, R., Singh, S. P., and Mittal, M. L. (2010), Traveling salesman problem : an overview of applications, formulations, and solution approaches, *Traveling Salesman Problem, Theory and Applications*, 1-24.

Miranda, D. M. and Conceição, S. V. (2016), The vehicle routing problem with hard time windows and stochastic travel and service time, *Expert Systems with Applications*, **64**, 104-116.

Murray, C. C. and Chu, A. G. (2015), The flying sidekick traveling salesman problem : Optimization of drone-assisted parcel delivery, *Transportation Research Part C : Emerging Technologies*, **54**, 86-109.

Padberg, M. and Rinaldi, G. (1987), Optimization of a 532-city symmetric traveling salesman problem by branch and cut, *Operations Research Letters*, **6**(1), 1-7.

Ritzinger, U., Puchinger, J., and Hartl, R. F. (2016), A survey on dynamic and stochastic vehicle routing problems, *International Journal of Production Research*, **54**(1), 215-231.

Shen, B., Yao, M., and Yi, W. (2006), Heuristic information based improved fuzzy discrete PSO method for solving TSP, *In Pacific Rim International Conference on Artificial Intelligence*, Springer, Berlin, Heidelberg, 859-863

Toth, P. and Vigo, D. (2002), Models, relaxations and exact approaches for the capacitated vehicle routing problem, *Discrete Applied Mathematics*, **123**(1-3), 487-512.

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., and Subramanian, A. (2017), New benchmark instances for the capacitated vehicle routing problem, *European Journal of Operational Research*, **257**(3), 845-858.

Yao, B., Yu, B., Hu, P., Gao, J., and Zhang, M. (2016), An improved particle swarm optimization for carton heterogeneous vehicle routing problem with a collection depot, *Annals of Operations Research*, **242**(2), 303-320.

저자소개

강문정 : 한국외국어대학교 산업경영공학과에서 2019년 학사학위를 취득하고 한국외국어대학교 산업경영공학과 석사과정에 재학 중이다. 연구분야는 최적화이다.

경지윤 : 한국외국어대학교 산업경영공학과에 재학중이다. 관심 연구분야는 머신러닝이다.

이충목 : 고려대학교 물리학과에서 1997년 학사, 1999년 석사학위를 취득하고 KAIST에서 2009년 산업공학 박사학위를 취득하였다. ETRI 선임연구원, IBM Research-Ireland에서 Research Staff Member등을 역임하고 2015년부터 한국외국어대학교 산업경영공학과에서 교수로 재직 중이다. 관심 연구 분야는 최적화 이론, 정수계획법, 강건최적화, 데이터 마이닝 등이다.