Journal of the Korean Institute of Industrial Engineers
Vol. 32, No. 1, pp. 61-73, March 2006.

강화학습기법을 이용한 TSP의 해법

임준묵† · 배성민 · 서재준

한밭대학교 산업경영공학과

A Learning based Algorithm for Traveling Salesman Problem

JoonMook Lim · SungMin Bae · JaeJoon Suh

Department of Industrial and Management Engineering, Hanbat National University, Daejeon, 305-719

This paper deals with traveling salesman problem(TSP) with the stochastic travel time. Practically, the travel time between demand points changes according to day and time zone because of traffic interference and jam. Since the almost pervious studies focus on TSP with the deterministic travel time, it is difficult to apply those results to logistics problem directly. But many logistics problems are strongly related with stochastic situation such as stochastic travel time. We need to develop the efficient solution method for the TSP with stochastic travel time. From the previous researches, we know that Q-learning technique gives us to deal with stochastic environment and neural network also enables us to calculate the Q-value of Q-learning algorithm.

In this paper, we suggest an algorithm for TSP with the stochastic travel time integrating Q-learning and neural network. And we evaluate the validity of the algorithm through computational experiments. From the simulation results, we conclude that a new route obtained from the suggested algorithm gives relatively more reliable travel time in the logistics situation with stochastic travel time.

***Keywords*:** TSP, stochastic travel time, q-learning, neural network

1. 서 론

TSP(Traveling Salesman Problem)는 n개의 도시와 도시 간의 거리(시간)가 주어져 있을 때 모든 도시들을 정확히 한 번씩 방문하고 출발도시로 돌아올 때 이동거리(시간)를 최소화하는 도시의 방문순서를 결정하는 문제로 정의된다. 이러한 TSP는 물류문제 등에서 매우 폭넓게 활용될 수 있는 기본적이면서도 중요한 문제 중의 하나이다. 지금까지 연구되어온 일반적인 TSP에서는 각 수요지 간의 시간이 확정적으로 주어지는 경우가 대부분이었다. 그러나 도심 등에서 일어나는 물류문제에서는 이러한 가정은 현실적이지 못하다. 예를 들어, 택시로 서울의 종로1가에서 동대문을 간다고 가정해 보자. 즉, 「종로1가 → 종로3가 → 종로5가 → 동대문」의 경로를 택했다고 하자. 원활하게 소통이 될 경우는 20분 정도면 충분할 것이다. 그러나, 때로는 똑 같은 길로 간다고 해도 1시간도 넘게 걸리는 경우도 자주 발생한다. 이것은 각 지점 간의 거리는 같더라도 걸리는 시간은 커다란 편차가 있음을 의미한다. 또한, 종로1가에서 정체가 발생하면 종로3가, 종로5가도 정체가 될 가능성이 높다. 이것은 각 지점 간의 걸리는 시간이 편차(분산)를 가질 뿐 만 아니라 서로 의존적(독립이 아님)이라는 사실을 나타내고 있다. 본 연구에서는 이렇게 각 지점 간의 시간이 확률적으로 주어지며, 서로 독립적이지도 않은 환경의 TSP를 다루고자 한다. 이하에서 이러한 TSP를 STSP(Stochastic Traveling Salesman Problem)로 부르기로 한다.

수요지 간의 거리(또는 시간)가 확정적으로 주어지는 TSP에 관해서는 수학적인 모형도 제시되어 있고 그에 대한 다양한 해법의 연구가 이루어져 왔다. 하지만, STSP에 대해서는 아직까지 제시된 수학적 모형은 알려지지 않고 있다. 단지, 몇몇 기존의 연구에서 주어진 가정하에서의 발견적 기법들이 제시되고 있다.

이 논문은 2002년도 한밭대학교 교내학술연구비의 지원을 받았음.
† 연락저자 : 임준묵 교수, 305-719 대전광역시 유성구 덕명동 산 16-1 한밭대학교 산업경영공학과, Fax : (042)821-1591,
E-mail : jmlim@hanbat.ac.kr
2005년 2월 접수; 2006년 1월 수정본 접수; 2006년 2월 게재 확정.

Leipala(1978)는 도시들 간의 거리(또는 시간)가 확정적으로 주어지지 않는 경우의 TSP에 대해서 최적경로의 하한값(Lower Bound)과 상한값(Upper Bound)을 계산할 수 있는 방법론을 제시하고 있으나 최적경로를 생성하는 데까지는 미치지 못하고 있다. Kao(1978)는 STSP에 대해서 동적 계획법(Dynamic Programming)에 기반을 둔 기법과 열거법(implicit enumeration)을 사용한 2개의 발견적 기법을 제시하고 있다. 하지만, 수요지 간의 소요시간 확률변수의 합에 대한 분포가 계산될 수 있다는 전제를 두고 있어서 현실성이 결여되어 있다. 또한 Sniedovich(1981)도 STSP에 대해서 동적 계획법을 사용한 해법을 제시하고 있으나 제시된 해법이 요구하는 모노토닉 성질(Monotonicity Property)이 STSP에서는 만족될 수 없으므로 최적해를 보장하지 못하고 있다. 그 후, Carraway *et al.*(1989)은 이러한 문제점을 극복하기 위한 일반적 동적 계획법(Generalized Dynamic Programming)에 기반한 알고리듬을 제시하여 STSP에 적용 가능성을 제시하였다. 그밖에 Lambert *et al.*(1993)은 경로의 도착시간에 제약이 있는 경우를 다루고 있으며, Laporte *et al.*(1992)은 수요지에서의 서비스 시간이 확률적으로 주어지는 경우에 대해서 분지한계법(Branch and Cut)을 사용한 알고리듬을 제시하고 있으나, 두 경우 모두 경로가 결정되면 경로에 대한 기대값이 쉽게 계산될 수 있다는 가정을 내포하고 있어서 해의 정확성에는 한계가 있다.

위에서 살펴본 바와 같이 현재까지 다루어져 왔던 연구의 결과로는 도심물류에서 발생하는 수요지 간 소요시간의 확률성과 비독립성을 모두 반영하는 데 한계가 있다. 따라서 본 연구에서는 이러한 현실적인 STSP의 환경을 반영하기 위해서, 일반적인 TSP 상황에 추가로 다음을 가정한다.

(1) 각 지점 간의 시간은 평균과 분산에 의해서 주어진다.
(2) 각 지점을 방문하는 시각에 대한 제약(Time Window)이나 서비스 시간 등은 고려하지 않는다.

또한 STSP의 환경하에서 문제의 목표로는 모든 지점을 방문하고 돌아오는 데 걸리는 시간의 기대치는 물론 전체 경로의 분산의 최소화를 목표로 삼는다.

위와 같은 목표를 달성하기 위한 STSP에 대한 연구의 어려움은 STSP가 가지는 복잡성과 난해성으로 기인한다고 할 수 있다. 즉, STSP는 일반적인 TSP와 매우 다른 성격을 가져서 확률적인 부분을 해결하기 위해서 추계적 과정분석(Stochastic process analysis) 또는 시뮬레이션 등이 요구되고 그 결과를 지식으로 이용하여 해에 반영하고 수렴시켜야 하는 등의 복잡한 계산과 절차의 도입이 필요하기 때문이다.

학습이론 중의 하나인 'Q-학습'이론은 이러한 시뮬레이션적인 경험을 지식으로 축적하여 학습함으로써 장기적으로 해가 최적의 결과에 수렴함이 잘 알려져 있다(Sutton and Barto, 1998). 그러나 아직까지 Q-학습이론을 이용한 STSP에 대한 연구결과는 발표되고 있지 않다.

따라서 본 연구에서도 확률적인 부분을 시뮬레이션 개념을 도입하여 해결하고 그 결과를 'Q-학습'의 이론을 활용하여 지식으로 학습·축적하고 Q-학습이 가지는 한계를 극복하기 위해서 인공신경망(Neural Network)기법을 통합한 새로운 STSP의 알고리듬을 제시하고자 한다.

제2절에서는 Q-학습의 개념과 일반적인 Q-학습을 이용한 STSP알고리듬을 제안하고 나아가서 인공신경망을 확장시킨 STSP의 알고리듬을 제시한다. 제3절에서는 제시된 알고리듬의 검증을 위한 수치실험결과와 시뮬레이션 결과 및 분석내용을 제시한다. 마지막으로 결론을 제시한다.

2. Q-학습을 활용한 TSP 알고리듬

2.1 Q-학습의 개요

본 연구에서는 확률적 환경의 최적경로를 찾기 위해서 강화학습(Reinforcement Learning)기법의 하나인 Q-학습을 이용한다. 강화학습기법의 전체적인 의사결정과정을 도식화하면 <Figure 1>과 같다. 어떤 환경(Environment)이 주어져 있고 이 환경을 이용하는 에이전트(Agent)가 있다고 하면, 먼저 에이전트가 환경으로부터 어떤 상태(State= s_t)를 인식하게 된다. 이때 에이전트는 특정 상태에 대한 여러 가지의 행동(Action= a_t)을 취할 수가 있는데 그 행동 중 하나를 취하면, 그 행동에 대해 환경으로부터 보상(Reward= r_t) 또는 벌점(Penalty)을 받게 된다. 이러한 일련의 과정의 반복을 통해 에이전트는 각 상태에 대한 행동들 가운데 좋은 보상(또는 벌점)을 받는 쪽으로 점점 행동을 취할 가능성(확률)을 높여 가게 된다. 이러한 방식으로 무한하게 반복을 하게 되면, 결국 하나의 상태에 대해 가장 좋은 행동 하나가 선택되어질 확률이 1.0에 가깝게 되어 상태와 행동이 일대일 대응관계로 되게 된다. 결론적으로 Q-학습 기법이란 이러한 학습과정이 끝나면, 하나의 상태에 대해 취할 수 있는 여러 가지 행동 가운데 가장 성능이 좋았던 행동 하나만을 추출하여 최종적으로 하나의 상태에 대해 하나의 행동만을 취할 수 있도록 하는 기법이다. 이것은 마치 인간의 행동과도 흡사하다고 할 수 있다. 인간의 행동도 착한 일을 하면 상

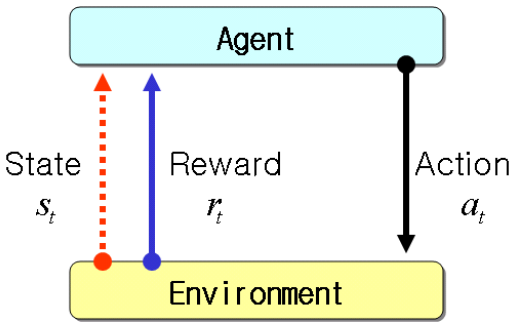

Figure 1. Reinforcement learning.

을 주고 잘못을 했을 경우에는 벌을 주어 점점 더 좋은 행동을 취할 수 있도록 하는 것과 흡사한 학습 방법이라 할 수 있다 (Lee, 1999).

2.2 Q-학습 알고리듬

일반적인 Q-학습의 알고리듬은 <Figure 2>와 같다.

Step0: Initialize $\hat{Q}(s, a)$= 0 for each (*s*,*a*)
Step1: Observe a state *s*
Step2: Repeat the following steps
Step2.1: Select an action *a* and execute it
Step2.2: Receive a reward *r*
Step2.3: Observe a new state s'
Step2.4: Update the *Q*-value as to the following equation:
$\hat{Q}(s, a) \leftarrow r + \gamma \cdot \max_{a'} \hat{Q}(s', a')$
where, γ is a discount rate
Step2.5: $s \leftarrow s'$

Figure 2. Algorithm for Q-learning.

Q-학습의 알고리듬은 앞 절에서 언급한 바와 같이 초기의 Q-값으로부터 시작하여 다음과 같은 절차를 반복적으로 수행한다(Kaelbling *et al.*, 1996).

State의 관측 → Action의 실행 → Reward의 수리 → Q-값의 갱신

그러나 Q-학습 알고리듬을 완전히 정의하기 위해서는 다음과 같은 Q-학습의 요소(elements)들을 정의할 필요가 있다.

(1) 목표(Goal)
목표(Goal)란 주어진 문제의 상황하에서 에이전트가 최종적으로 도달하고자 하는 것을 의미한다.

(2) 정책(Policy)
목표를 달성하기 위해서 행동(Action)을 선택하는 것을 의미한다. 즉, 에이전트가 목표에 도달하기 위해서는 현재부터 어떤 행동을 행하며 진행하면 되는가? 하는 행동의 집합을 나타낸다.

(3) 상태(State)
에이전트가 처한 문제환경의 상황을 의미한다.

(4) 행동(Action)
현재의 상태에서 에이전트가 취하게 되는 행동을 의미한다.

(5) 보상/벌금(Reward/Penalty)
에이전트가 현재의 상태에서 하나의 행동을 취하면 환경은 상태의 변화에 대해서 반응을 하게 되는데 그 때 에이전트가 환경으로부터 받게 되는 보상 또는 벌금을 의미한다. 예를 들어, STSP에서는 현재 상태(어느 도시)에서 다음에 방문하게 되는 도시를 선택하는 행동을 취하게 되면, 이때 환경으로부터 에이전트는 추가된 경로에 따라 소요되는 '시간'을 보상(벌점)으로 받게 된다. 여기서, 시간은 작으면 작을수록 좋은 개념이므로 보상보다는 벌점의 개념이 맞지만, 용어의 통일성을 위해서 보상(Reward)이라는 용어를 이하에서 사용할 것이며, 계산 시에는 목적함수에 최대화(max) 대신 최소화(min)를 사용한다.

(6) Q-값(Q-value)
Q-학습에서는 에이전트가 현재의 상태에서부터 시작하여 목표에 도달할 때까지 취한 행동들에 의해서 최대로 받을 수 있는 보상을 시간의 가치에 따라서 할인(discount)하여 누적합(cumulative sum)으로 나타내고 그 기대치를 계산하기 위해서 'Q-함수'를 사용하게 된다. 여기서, 'Q-함수'는 상태와 행동의 함수로 구성되는데, 그 함수가 특정상태에서 일정한 행동을 취했을 때 갖게 되는 값을 'Q-값'이라 한다.

(7) 학습규칙(Learning Rule)
Q-학습에서 학습을 수행하는 규칙은 상태와 보상이 확정적(deterministic)으로 주어지는 경우와, 확률적(stochastic)으로 주어지는 경우에 따라서 다르게 주어진다.

바로 다음의 상태(s_{t+1})가 현재의 상태(s_t)와 그 때 취한 행동(a_t)에 의해서 결정되며 그 결정되는 함수가 $s_{t+1} = \delta(s_t, a_t)$와 같이 주어지고, 보상도 마찬가지로 현재의 상태에서 취한 행동에 대해서 $r_t = r(s_t, a_t)$와 같이 주어진다고 가정하면 학습규칙은 다음과 같이 두 가지의 경우로 나누어진다.

경우 1: 확정적인 경우

$s_{t+1} = \delta(s_t, a_t)$와 $r_t = r(s_t, a_t)$가 확정적으로 주어지는 경우를 의미한다. 여기서 확정적이란 에이전트가 현 상태(s_t)에서 행동(a_t)를 취하면 특정한 하나의 상태(s_{t+1})로 전이되고 그때 환경으로부터 받는 보상값도 함수 $r(s_t, a_t)$에 의해서 특정한 하나의 값이 확정적으로 주어짐을 의미한다. 이 때의 Q-함수의 학습규칙은 다음과 같다.

$$\hat{Q}(s_t, a_t) \leftarrow r(s_t, a_t) + \gamma \max_{a'} (\hat{Q}(s_{t+1}, a'))$$

경우 2: 확률적인 경우

$s_{t+1} = \delta(s_t, a_t)$와 $r_t = r(s_t, a_t)$가 확률적으로 주어지는 경우를 의미한다. 확률적인 경우란 에이전트가 현재의 상태(s_t)에서 행동(a_t)를 취하면 전이되는 다음 상태 s_{t+1}의 값과 그에 상응하는 보상 r_t의 값이 s_t와 a_t에 따라서 정해진 하나의 값을 갖지 않고 확률적으로 주어짐을 의미한다. 예를 들어, 본

논문의 STSP에서는 다음 상태 s_{t+1}는 함수 $s_{t+1}=\delta(s_t, a_t)$에 의해서 확정적인 경우와 동일하게 주어지지만, 그에 따른 보상값 $r_t=r(s_t, a_t)$는 확률적으로 주어진다. 즉, 보상이 현재의 노드로부터 다음 노드까지 걸리는 시간이므로 두 노드 간 소요되는 시간은 평균과 분산에 의해서 주어지는 값을 가지게 된다.

결국, 이렇게 확률적으로 보상값이 주어지면 새롭게 얻어지는 보상값을 모두 새로운 Q-값, $\hat{Q}(s_t, a_t)$에 반영할 수 없게 되어 1회의 학습에서 일정한 비율(학습률(α))만큼만 반영하게 된다.

이 때의 Q-함수의 학습규칙은 다음과 같다.

$$\hat{Q}(s_t, a_t) \leftarrow (1-\alpha)\hat{Q}(s_t, a_t) + \alpha[r(s_t, a_t) + \gamma \max_{a'}(\hat{Q}(s_{t+1}, a'))]$$

여기서,

γ = 할인율(discount rate)

α = 학습률 $= \frac{1}{1+count(s_t, a_t)}$

$count(s_t, a_t)$ = 현재까지 방문한 (상태, 행동)= (s_t, a_t)의 총 수

2.3 일반적인 Q-학습을 활용한 TSP 알고리듬

(1) TSP를 위한 Q-학습의 적용

TSP를 Q-학습을 활용하여 풀기 위해서는 Q-학습의 요소들을 TSP의 특성에 맞게 정의함으로써 가능하다. <Figure 3>과 같이 출발지가 D이고 수요지가 1, 2, 3으로 주어지는 간단한 TSP를 예로 들어 설명하기로 한다.

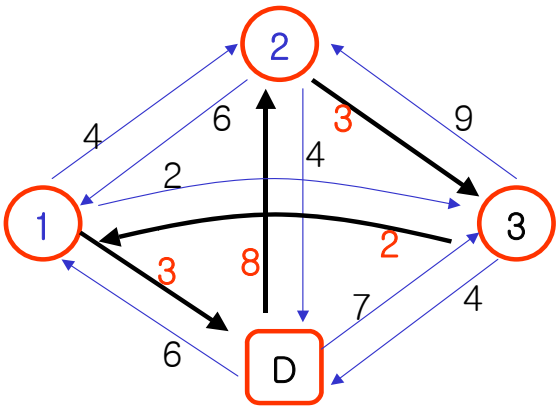

Figure 3. An example for TSP.

<Figure 3>에서 보는 바와 같이 각 수요지 간의 이동시간은 확정적인 수치에 의해서 주어진 것으로 가정하자 매우 간단한 문제이므로 최적해는D → 2 → 3 → 1 → D(굵은 실선으로 표시됨)이고 총 소요시간은 16이 됨을 쉽게 알 수 있다. 그러면 TSP를 Q-학습의 각 요소에 대응시키기로 하자.

① 목표
- D를 출발하여 가장 짧은 시간에 모든 노드를 한 번씩 방문하고 D로 돌아옴.
- 위와 같은 목표는 일반적인 TSP의 목적과 일치한다.

② 상태
- TSP에서 문제의 상황을 의미하며, TSP에서의 현재 상태는 현재까지 방문한 노드들과 가장 최근에 방문한 노드에 대한 현재의 위치로 정의될 수 있다.
- 상태(s) = 현재까지 방문한 노드의 집합 + 현재 위치

③ 행동
- 행동은 현재의 상태(s)에서 에이전트가 목표를 달성하기 위해서 취할 수 있는 행위로서, TSP에서는 현재의 위치에서 다음 방문지를 어디로 할 것인가? 하는 것으로 나타낼 수 있다.
- 행동(a) = 다음 방문지(노드)

④ 보상
- 보상은 현재의 상태(s)에서 에이전트가 취한 행동(a)에 의해 환경으로부터 받을 수 있는 대응값을 말하므로, TSP에서는 현재의 위치에서 택한 다음 행선지(노드)까지의 소요거리(시간)로 나타낼 수 있다.
- 보상($r(s, a)$) = 노드 간의 거리(시간)

⑤ Q-값
- Q-함수가 가지는 값을 의미하는데, Q-함수는 일반적으로 현재의 상태(s)와 취한 행동(a)에 의해서 정의된다.
- TSP에서 Q-값은 현재 노드에서부터 남은 노드 전부를 방문하고 목적지(D)까지 갈 때의 총 소요시간의 할인된 기대값(discounted expected value)을 의미한다. 따라서 이러한 Q-함수(값)의 추정량(estimator)으로서 $\hat{Q}(s, a)$를 다음과 같이 정의한다.
- $\hat{Q}(s, a) = r(s, a) + \gamma \cdot \min_{a'} \hat{Q}(s', a')$
- 여기서, 할인된 기대치를 사용하는 이유는 보상의 시간적 가치를 반영하기 위함이다.

⑥ 학습규칙
- 주어진 예제는 확정적인 경우로서, 다음의 상태와 보상이 확정적으로 주어짐을 의미한다. 즉, 현재의 상태에서 어느 한 행동을 취하면 상태가 확정적으로 주어지며, 그로 인한 보상값도 확정적으로 하나의 값으로 주어진다.
- 학습규칙은 다음과 같다.

$$\hat{Q}(s, a) \leftarrow r(s, a) + \gamma \cdot \min_{a'} \hat{Q}(s', a')$$

여기서, $r(s, a)$는 상태 s에서 행동 a를 취했을 때 받게 되는 보상을 의미하며, γ는 할인율(discount rate), $\hat{Q}(s, a)$는 Q-값의 추정치를 의미한다. min을 사용하는 이유는 TSP에서는 보상(Reward)이 시간치로서 작을수록 목표(Goal)에 부합하기 때문이며 Q-학습의 학습규칙에 위배되

지 않도록 단순히 max를 min으로 바꾸었다.

위에서 TSP에 정의된 Q-학습의 각 요소를 예를 들어 설명하기로 한다. <Figure 4>와 같이 방문자가 D를 출발하여 현재 2의 노드에 와 있다고 하자. 즉, D→2→?. 현재의 위치는 2이고 현재까지 방문한 노드집합은 {D,2}가 된다. 따라서, 현 상태는 {D,2} + 2로 정의된다.

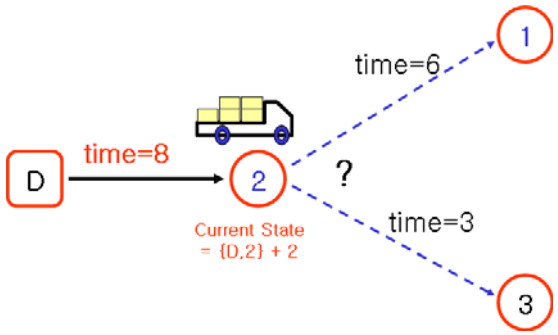

Figure 4. Explanation of state and action.

다음에 취할 수 있는 행동으로는 노드 1, 3 및 D가 예상되나 D로 가는 것은 TSP 정의에 위배되어 현재의 상태에서 선택 가능한 행동은 {1,3}이 된다. 현 상태에서 다음의 행동으로 노드 1을 선택한다면, 보상은 6(노드 2 → 노드 1의 소요시간이 6이므로)이 되고, 노드 3을 선택하면 3의 보상을 받게 된다. 현재의 상태에서 Q-값은 현 위치에서 계속해서 행동을 취하고 보상을 받으며 진행해서, 최종 목적지에 도달할 때까지 받을 수 있는 보상의 할인(割引,discount)된 기대치의 총합을 의미한다. 다시 말해, 현 위치를 출발하여 현재까지 방문한 노드를 제외한 나머지 노드를 모두 방문하고 다시 D로 돌아오는 데 소요될 것으로 예상되는 시간의 할인된 기대치를 말한다.

현재의 상태에서 다음의 행동을 선택하고 보상을 받게 되면 다음으로 Q-값을 갱신하게 되는데, Q-값은 Q-테이블에 저장되어 있는 Q-값과 현재의 행동(a)에 대한 보상(r)으로부터 최종 목적지까지 도달할 때까지 얻을 수 있는 보상의 할인된 기대치가 최적화되도록 Q-값을 갱신한다.

(2) Q-학습 과정의 예

위에서 주어진 예제를 대상으로 Q-학습이 어떠한 과정을 거쳐서 이루어지는지 알아보기로 하자.

간단한 에피소드(Episode)를 사용해서 학습시켜 나가면서, Q-테이블에서 Q-값이 어떻게 갱신되는지 과정을 살펴보기로 하자. 학습을 위한 에피소드(Episode)로는 'D → 1 → 2 → 3 → D'를 사용하는 것으로 하고 현재의 에피소드를 가지고 4회 반복학습을 한다고 가정하자. 그리고 할인율은 $\gamma = 0.9$ 로 한다.

Table 1. Change of Q-values in Q-table by learning process

State(s)			a	r	Q-value						
Denotation	Set of already visited nodes	Current position			Initial value	Iter.1	Iter.2	Iter.3	Iter.4	⋯	Final
S(t)	{D}	D	1	6	**100**	**96**(w)	**90.6**	**84.93**	**14.95**	⋯	14.95
			2	8	100	100	100	100	100	⋯	**14.51**
			3	7	100	100	100	100	100	⋯	14.96
S(t+1)	{D,1}	1	2	4	**100**(s)	**94**	**87.7**	**9.94**	**9.94**(s)	⋯	9.94
			3	2	**100**(y)	100	100	100	100	⋯	13.34
	{D,2}	2	1	6	100	100	100	100	100	⋯	11.04
			3	3	100	100	100	100	100	⋯	**7.23**
	{D,3}	3	1	2	100	100	100	100	100	⋯	8.84
			2	9	100	100	100	100	100	⋯	16.83
S(t+2)	{D,1,2}	1	3	2	100	100	100	100	100	⋯	5.6
		2	3	3	**100**	**93**	**6.6**	**6.6**	**6.6**	⋯	6.6
	{D,1,3}	1	2	4	100	100	100	100	100	⋯	7.6
		3	2	9	100	100	100	100	100	⋯	12.6
	{D,2,3}	2	1	6	100	100	100	100	100	⋯	8.7
		3	1	2	100	100	100	100	100	⋯	**4.7**
S(t+3)	{D,1,2,3}	1	D	3	100	100	100	100	100	⋯	**3**
		2	D	4	100	100	100	100	100	⋯	4
		3	D	4	**100**	**4**	**4**	**4**	**4**	⋯	4
S(t+4)	{D,1,2,3,D}	D									

초기의 모든 상태에 대한 Q-값은 100으로 하였다. 본 예제에서는 TSP가 최소화 문제이므로 0으로 하지 않고 충분히 큰 수인 100으로 하였다. 1회의 학습과정을 거치면 상태의 변화에 따른 Q값이 차례로 갱신된다. <Table 1>에서 보는 바와 같이, 예를 들어 1회의 96(<Table 1>에서 ⓦ로 표시된 값)은 96 = 6 + 0.9 × min{100,100}에 의해서 얻어진다. 여기서 6은 현재의 위치 D에서 다음 노드로 1을 택했을 경우 받게 되는 보상(소요시간)을 의미하며 0.9는 할인율을 의미하고, min{100,100}은 현재의 상태에서 행동으로 노드 1을 선택할 경우 현재까지의 학습결과로부터 판단하여 얻을 수 있게 될 것으로 기대되는 Q-값 중 최소치를 말한다. 즉, min{100,100}은, 현 상태 S(t)에서 행동으로 노드 1을 취할 경우 상태는 S(t+1) = {D,1}+1이 될 것이고 그 상황에서 다시 행동으로 노드 2 또는 노드 3을 선택할 수 있게 되는데, 현재까지의 학습결과로 볼 때 그 다음 행동으로 노드 2를 선택했을 경우 기대되는 Q-값은 100(<Table 1>에서 ⓧ로 표시된 값)이 되며, 같은 맥락으로 노드 3을 선택했을 경우 역시 100(<Table 1>에서 ⓨ로 표시된 값)이 된다. 따라서, 현재 상태 S(t)에서 노드1을 행동으로 취함으로써 그 뒤로 얻을 수 있는 보상의 최적(최소)치는 min{100,100}이 됨을 의미한다. 그리고 같은 방법으로 4회의 반복학습이 진행된다고 가정할 때, 에피소드의 4회째 반복학습에서 9.94(<Table 1>에서 ⓩ로 표시된 값)는 9.94 = 4 + 0.9 × min{6.6}에 의해서 얻어진다.

계속해서, 다양한 에피소드를 발생시켜 반복학습을 수행하면 최종적으로 Q-값은 <Table 1>의 'Final'열에 주어진 값으로 수렴하게 된다. <Table 1>에서 1, 2, 3 및 4회의 학습과정을 거치며 Q-값의 갱신되는 과정을 <Table 2>에 정리하였다.

(3) 최적경로의 생성

위와 같은 과정을 거쳐서 Q-학습이 완료되면 최종 Q-테이블의 Q-값으로부터 다음과 같은 규칙에 따라 최적경로를 생성한다.

최적경로 생성규칙:

$$D \rightarrow a^*_t \rightarrow a^*_{t+1} \rightarrow \ldots \rightarrow a^*_{t+m} \rightarrow D$$

여기서, $a^*_t = \{ a_t \mid \min_{a_t} \widehat{Q}(s^*_t, a_t) \}$,

$s^*_t = \{ s_t \mid \min_{a_{t-1}} \widehat{Q}(s_{t-1}, a_{t-1}) \}$,

m = 수요지 수

예를 들어, <Table 1>에 주어진 최종학습 Q-테이블로부터 다음과 같이 최적경로를 생성할 수 있다. D를 출발하여, 상태 S(t)

Table 2. Explanation for change process of Q-values in learning process

No. of learning	Progress of an episode	Current position	Next action (a)	Reward (r)	New state s_t		Calculation of Q-value(Update)
					Already visited nodes	Next position	
1	D→1	D	1	6	{D,1}	1	$Q_n = r(s,a) + \gamma \times \min_{a_0} Q(s_0, a_0) = 6 + 0.9 \times \min\{100, 100\} = 96.0$
	D→1→2	1	2	4	{D,1,2}	2	$Q_n = 4 + 0.9 \times \min\{100\} = 94.0$
	D→1→2→3	2	3	3	{D,1,2,3}	3	$Q_n = 3 + 0.9 \times \min\{100\} = 93.0$
	D→1→2→3→D	3	D	4	{D,1,2,3,D}	D	$Q_n = 4 + 0.9 \times \min\{0\} = 4.0$
2	D→1	D	1	6	{D,1}	1	$Q_n = 6 + 0.9 \times \min\{94.0, 100\} = 90.6$
	D→1→2	1	2	4	{D,1,2}	2	$Q_n = 4 + 0.9 \times \min\{93.0\} = 87.7$
	D→1→2→3	2	3	3	{D,1,2,3}	3	$Q_n = 3 + 0.9 \times \min\{4.0\} = 6.6$
	D→1→2→3→D	3	D	4	{D,1,2,3,D}	D	$Q_n = 4 + 0.9 \times \min\{0\} = 4.0$
3	D→1	D	1	6	{D,1}	1	$Q_n = 6 + 0.9 \times \min\{87.7, 100\} = 84.93$
	D→1→2	1	2	4	{D,1,2}	2	$Q_n = 4 + 0.9 \times \min\{6.6\} = 9.94$
	D→1→2→3	2	3	3	{D,1,2,3}	3	$Q_n = 3 + 0.9 \times \min\{4.0\} = 6.6$
	D→1→2→3→D	3	D	4	{D,1,2,3,D}	D	$Q_n = 4 + 0.9 \times \min\{0\} = 4.0$
4	D→1	D	1	6	{D,1}	1	$Q_n = 6 + 0.9 \times \min\{9.94, 100\} = 14.95$
	D→1→2	1	2	4	{D,1,2}	2	$Q_n = 4 + 0.9 \times \min\{6.6\} = 9.94$
	D→1→2→3	2	3	3	{D,1,2,3}	3	$Q_n = 3 + 0.9 \times \min\{4.0\} = 6.6$
	D→1→2→3→D	3	D	4	{D,1,2,3,D}	D	$Q_n = 4 + 0.9 \times \min\{0\} = 4.0$

에서 최소의 Q-값은 14.51이므로 2를 선택한다. 그러면 새로운 상태는 D→2가 되므로 S(t+1) = {D,2} + 2에서 최소값 7.23에 해당하는 노드 3을 선택한다. 현재까지의 경로는 D→2→3이 되고 상태는 S(t+2) = {D,2,3} + 3이 되며, S(t+2)에서 최소값인 4.7에 해당하는 노드 1을 선택하여 D→2→3→1이 된다. 마지막으로 D로 돌아옴에 의해서 최적경로 D→2→3→1→D가 완성된다. 이것은 직관적으로 구했던 최적경로와 일치함을 알 수 있다.

(4) 확률적인 시간을 가지는 경우의 학습규칙

위의 알고리듬에서 각 수요지 간의 시간이 확률적인 분포로 주어지는 경우는 Q-값의 갱신을 위한 학습규칙(Learning Rule)의 식을 다음 식으로 바꾸어 계산하기만 하면 된다.

$$\hat{Q}(s_t, a_t) \leftarrow (1-\alpha)\hat{Q}(s_t, a_t) + \alpha[r(s_t, a_t) + \gamma \max_{a'}\{\hat{Q}(s_{t+1}, a')\}]$$

여기서,

γ = 할인율

α = 학습률 $= \frac{1}{1 + count(s_t, a_t)}$

$count(s_t, a_t)$ = 현재까지 방문한 (상태, 행동) (s_t, a_t)의 총 수

2.4 Neural Network(NN)를 사용한 TSP를 위한 Q-학습 알고리듬

앞 절에서 제시된 일반 Q-학습에 의한 TSP 알고리듬은 다음과 같은 문제점을 가진다. 상태의 표현방법으로 앞의 정의를 사용하게 되면 수요지의 수가 증가하면서 상태의 수가 기하급수적으로 증가하여 수요지의 수가 10 이상의 경우는 Q-테이블의 저장에 애로가 발생한다. 따라서, 본 연구에서는 상태의 표현과 Q-값들을 Q-테이블에 직접 저장시키는 방식 대신에 NN에 학습시켜 계산하는 방법을 사용한 새로운 알고리듬을 제안한다.

(1) 상태와 Q-값의 학습을 위한 NN의 적용

본 연구에서 사용한 Q-값의 저장과 계산을 위해서 사용한 NN은 <Figure 5>에서 보여지는 형태의 일반적으로 잘 알려진 역전파 NN (Back Propagation Neural Network(BPNN))이다. NN은 입력층, 중간층(hidden layer) 및 출력층의 3개 층으로 구성된다.

본 연구에서 Q-학습 알고리듬의 단점인 Q-값의 저장과 계산의 문제점을 보완하기 위해서 BPNN을 사용한 이유는 다음과 같다.

- BPNN은 네트워크가 단순하지만 일반적으로 다량의 숫자 또는 패턴을 잘 기억하는 것으로 알려져 있어서 Q-학습에서 필요로 하는 다량의 Q-값을 용이하게 기억시킬 수 있으며 쉽게 계산할 수 있을 것으로 판단됨(Freeman and Skapura, 1992)
- BPNN은 네트워크에 사용되는 웨이트(weight)의 학습을 위한 효율적인 최적화 알고리듬(Back Propagation Algorithm)이 잘 알려져 있어 쉽게 적용이 가능함(Kim, 1992)

본 NN은 입력층에 현재의 상태와 그 때 취한 행동의 값을 입력하면 중간층을 거쳐서 출력층으로 해당되는 Q-값을 산출하게 된다.

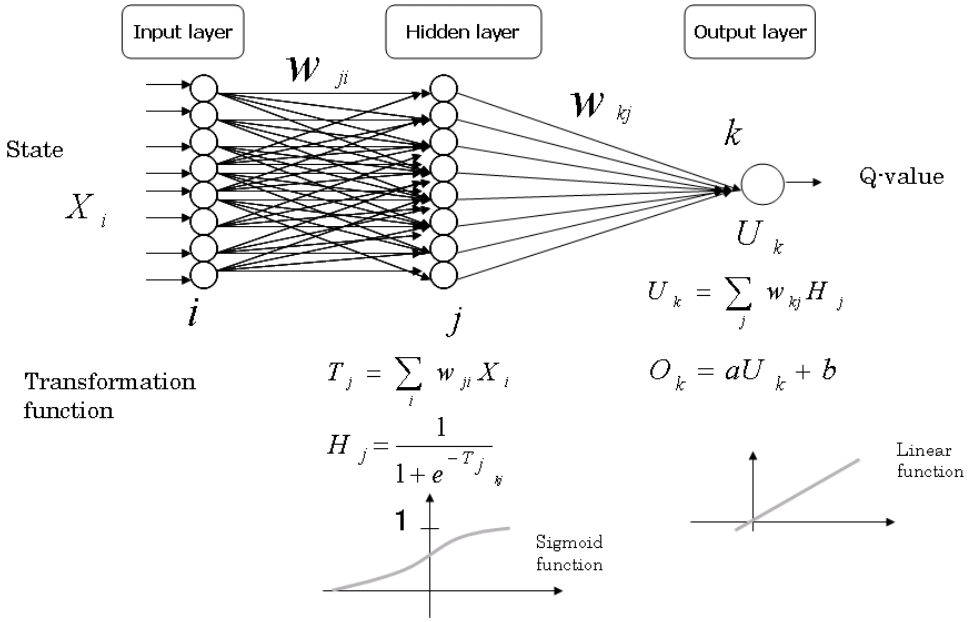

Figure 5. Neural network for calculating Q-values.

① 입력층

입력층으로 입력되는 값은 현재의 상태와 취한 행동의 값이다. 따라서, 입력층에 입력되는 값은 다음과 같이 3가지로 구성된다.

입력층의 입력값:

- 방문완료 노드의 집합
- 현재의 노드
- 현재의 상태에서 취하게 될 행동

이러한 입력층을 구성하는 방법으로는 여러 가지가 있을 수 있으나 본 연구에서는 각 노드에 0 또는 1을 입력하여 표현하는 방법을 사용하였다.

입력층 노드의 0은 출발지를 의미하며 1, 2, 3, ... , m은 방문하는 노드의 번호를 나타낸다. 앞 절의 예제에서 주어진 출발지와 3개의 방문지를 가지는 TSP의 경우의 예를 들어서 설명하기로 한다. 즉, m=3인 경우를 의미한다. <Figure 6>은 이러한 입력층의 구체적인 표현 예를 보여주고 있다. 다음에서 입력층의 노드에 대한 입력값을 표현하는 방법을 자세히 설명하기로 한다.

a. 방문완료 수요지집합의 표현

한 상태에서 현재까지 방문을 완료한 노드의 집합을 표현하기 위해서 방문완료한 수요지에 해당하는 노드는1로 아직 방문하지 아니한 수요지에 대한 노드는0의 값을 갖는 것으로 표현하였다. 예를 들어, 현재까지 방문한 노드가 출발지와 2라면 (= {0, 2}), (X1, X2, X3, X4) = (1, 0, 1, 0)으로 표현한다.

b. 현재의 노드 표현

현재의 노드는 0, 1, 2, ..., m 값 중의 하나를 갖게 되므로, 해당 값을 갖게 되는 노드만 1을 가지고 나머지는 0을 가지도록 하였다. 예를 들어, 현재의 노드가 2라면 (X5, X6, X7, X8) = (0, 0, 1, 0)으로 표현한다.

c. 행동의 표현

현재의 노드 표현과 마찬가지로, 현재 상태에서 취하게 되는 행동에 해당하는 노드의 번호값만 1을 가지고 나머지는 0을 갖는 것으로 하였다.

예를 들어, 취한 행동이 1이라면, (X9, X10, X11, X12) = (0, 1, 0, 0)

따라서, 위와 같은 입력층의 표현방법을 따른다면, NN의 입력층에 필요한 노드의 수는 [m+1개(방문완료노드집합)] + [m+1개(현재의 노드)] + [m+1개(행동)]으로 총 3(m+1)개의 노드가 필요하다. 예를 들어, 출발지와 10개의 노드로 구성된 TSP는 33개의 입력층 노드가 필요하다.

② 중간층

본 연구에서 중간층의 노드의 수는 입력층과 같은 것으로 했다. 중간층에서의 변환함수는 <Figure 7>과 같은 일반적으로 널리 사용하는 시그모이드함수,

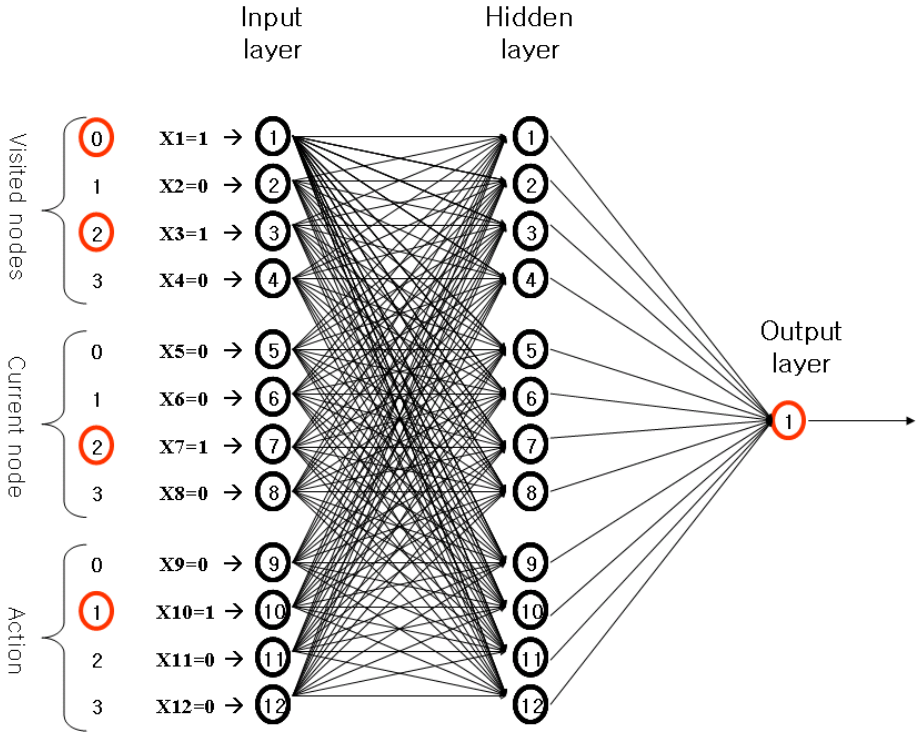

Figure 6. An example of the neural network with input, hidden and output layers for Q-learning.

$f(x) = 1/(1 + e^{-x})$

를 사용하였다.

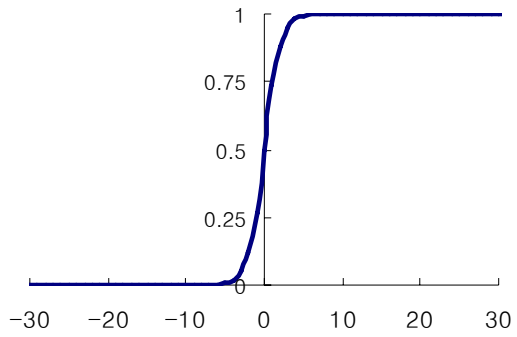

Figure 7. An example of transformation function used in the hidden layer of neural network.

원칙적으로 NN에서 중간층의 변환함수로는 <Figure 8>에 주어진 것과 같은 형태의 함수를 사용하는 것이 입력에 대한 출력값을 표현하는 데 있어서 정확성을 가지지만, NN의 학습에서 최소제곱법을 사용함에 있어서 미분 등의 수행으로 발생하는 애로점을 해결하기 위해, 많은 연구에서 채택하여 사용하고 있는 <Figure 7>과 같은 시그모이드(sigmoid)함수를 변환함수로 사용하였다.

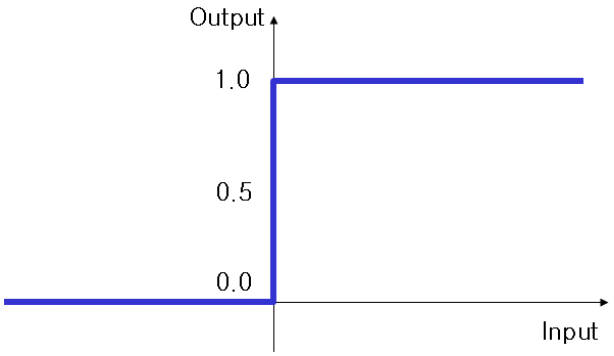

Figure 8. An example of ideal transformation function in the hidden layer.

③ 출력층

출력층에서 출력되는 값은 입력층에서의 현재상태와 취한 행동의 입력값에 대해서 가지게 되는 Q-함수의 값을 의미한다. 따라서, 1개의 노드로 표현된다. 출력층에서 출력되는 Q-값은 0 이상의 실수값을 가지게 되므로 $f(x) = ax + b$ 형태의 선형함수를 입·출력함수로 사용하였다. 또한, 본 연구에서는 <Figure 9>에서 주어진 함수와 같이 a=1, b=0인 $f(x) = x$ 형태의 함수를 사용하였다.

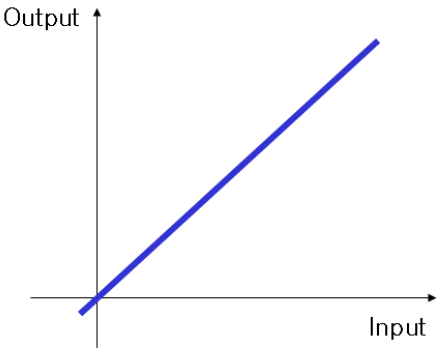

Figure 9. Transformation function in the output layer of neural network.

(2) NN의 학습과 Q-학습의 수행과정

Q-학습의 상태와 Q-값을 계산하기 위한 NN의 학습과정은 역전파 NN(Back Propagation Neural Network)에서 일반적으로 모수의 학습을 위해서 사용하는 역전파 알고리듬(Back Propagation Algorithm)을 사용하였다(Kim, 1992). 따라서, 본 연구에서는 NN의 학습에 대한 알고리듬에 대한 설명은 생략하기로 한다.

TSP를 위한 Q-학습 알고리듬을 완성하기 위해서, 일반적인 Q-학습의 프레임워크(Framework)에 상태와 Q-값을 계산해 주기 위한 'Q-net'이라는 NN이 삽입된다. <Figure 10>은 NN을 포함한 Q-학습을 사용한 TSP 알고리듬의 흐름을 보여주고 있다.

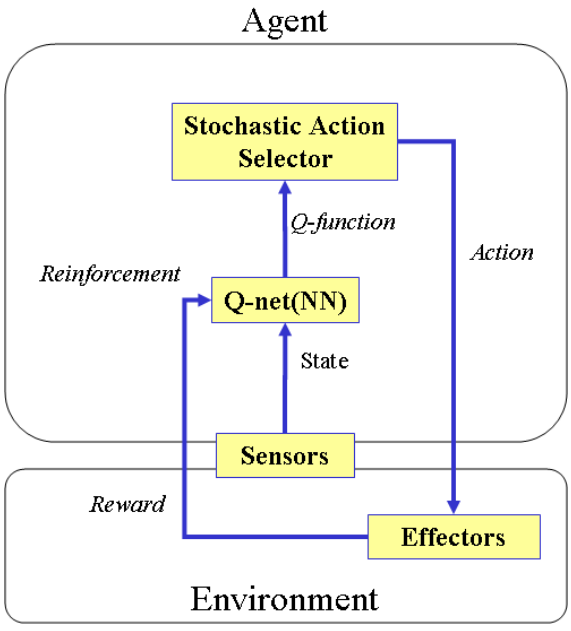

Figure 10. Flow diagram of TSP algorithm using NN and Q-learning.

<Figure 10>에서 보듯이, NN과 Q-학습을 활용한 TSP의 알고리듬의 전체적인 수행과정은 다음과 같은 절차를 반복적으로

수행하는 것으로 이루어진다.

① TSP의 Q-학습 에이전트는 환경으로부터 Sensors를 통해서 환경의 상태를 인식하고 환경의Effectors로부터 돌아오는 보상값을 이용하여Q-net의 학습을 수행한다.

② 에이전트는 Q-net으로부터 Q-값을 도출하고 Stochastic Action Selector를 활용하여 적절한 행동을 취한다.

③ 에이전트에 의해서 취해진 행동은 환경에 반영되고 환경은 Effectors로부터 그 에 상응하는 보상값을 에이전트에 돌려보내게 되며 에이전트는 해당 보상값을 활용하여 Q-net의 강화학습을 수행한다.

(3) NN을 이용한TSP의 Q-학습 알고리듬

전체적인 알고리듬은 <Figure 11>에 주어진다. 알고리듬은 크게 두 부분의 학습모듈을 가진다. 하나는 전체 흐름에 있어서의 Q-학습이고, 다른 하나는 Q-값을 저장하기 위한 Q-net (NN)의 학습이다. 따라서, 본 알고리듬은 다양한 에피소드를 발생시켜 가면서 Q-학습과 Q-net(NN)의 학습을 동시에 행하는 알고리듬이다. 여기서, 앞 절에서 언급한 바와 같이 Q-net(NN)의 학습을 위해서는 역전파 알고리듬(Back propagation algorithm)을 사용한다.

3. 수치실험

수치실험은 두 가지의 경우에 대해서 실시하였다. '실험 1'은 앞 절에서 제시한 NN과 Q학습을 사용한 TSP 알고리듬이 과연 최적해에 수렴하는가에 관한 검증실험이고, '실험 2'는 각 수요지 간의 시간자료가 확률적으로 주어질 경우와 확정적 자료만을 사용했을 경우에 대해서 본 연구에서 제시한 알고리듬의 해에 대한 성능의 비교평가에 관한 실험이다.

3.1 실험 1 - 알고리듬의 검증

제시된 알고리듬이 최적해에 수렴하는가를 알아보기 위해서 <Table 3>에 주어진 것과 같이 임의로 발생시킨 확정적 시간치 자료를 가지고 알고리듬을 수행하였다. 학습 알고리듬에 사용된 모수는 사전실험의 경험으로부터Q-학습률은 0.05, 할

Table 3. Time data between customers(deterministic case)

	0	1	2	3	4	5	6	7	8	9	10
0	0	13	7	8	17	2	17	1	18	16	2
1	10	0	9	1	21	19	3	20	9	12	17
2	3	13	0	14	14	7	7	18	1	18	13
3	21	20	6	0	2	6	1	12	4	21	15
4	13	4	4	17	0	1	7	9	8	20	21
5	10	1	7	5	5	0	14	16	8	14	6
6	5	13	1	3	11	20	0	7	7	20	14
7	14	10	3	13	1	15	20	0	11	20	10
8	15	12	12	11	9	10	7	3	0	3	1
9	1	7	9	11	5	11	7	14	12	0	2
10	6	9	4	17	11	17	13	18	2	1	0

Step1: Initialization

$x \leftarrow$ *current state*; $U_i \leftarrow Q(x, i)$ for each action i

Step2: Selection of an action

$a \leftarrow Select(U, T)$, where $Select(U, T)$ is a stochastic action selection module

Step3: Execution of the action a

$(y, r) \leftarrow$ Observation of a new state y and acquisition of a new reward r

Step4: Q-learning

(i) deterministic case: $u^0 \leftarrow r + \gamma \times \min\{Q(y, k) \mid k \in Actions\}$

(ii) stochastic case: $u^0 \leftarrow (1-\alpha)u^0 + \alpha[r + \gamma \times \min\{Q(y, k) \mid k \in Actions\}]$

where, γ is a discount rate and α is a learning rate

Step5: Learning of Q-net(Neural Network)

Adjustment of Q-net according to the error ΔU between input x and back-propagation

$$\Delta U = \begin{cases} u^0 - U_i & \text{if } i = a \\ 0 & otherwise \end{cases}$$

Step6: Go to Step1 until $\Delta U \le \varepsilon$, where ε means a threshold value

Figure 11. TSP algorithm using NN and Q-learning.

인율은 0.7, 그리고 NN의 학습률은 0.2를 사용하였다. <Table 3>의 시간치 자료에 대한 최적해는 0 → 7 → 4 → 5 → 1 → 3 → 6 → 2 → 8 → 10 → 9 → 0이고 경로에 대한 총 시간은 11이다. <Figure 12>는 알고리듬의 수행결과 학습횟수의 증가에 따른 경로시간과 Q-값의 변화를 보여주고 있다. 학습횟수가 증가함에 따라 최적해에 수렴하고 있음을 볼 수 있다.

또한, 본 연구에서는 임의로 발생된 다른 여러 가지의 문제 세트(set)에 대해서도 최적해 수렴 여부와 컴퓨터 계산시간을 살펴보았다. 방문노드의 수(m)에 대해서 각각 임의의 10개 또는 5개의 문제의 세트를 만들어 실험하였다. 각 노드 간의 시간은 U[1～20]의 일양분포(Uniform distribution)로부터 발생시켰다. 알고리듬에 사용된 (Q-학습률, 할인율, NN 학습률)의 모수값은 사전실험의 경험치를 바탕으로 (0.05, 0.7, 0.2)를 사용하였다. 알고리듬은 Visual BASIC으로 코딩하였으며 Pentium (R)4 CPU 3.0GHz, 1.0GB RAM의 PC를 사용하여 실행하였다. <Table 4>는 방문노드의 수(m)에 따른 최적해의 수렴 여부와 평균계산시간을 보여준다. <Table 4>에서 보는 바와 같이, 모든 경우에 있어서 본 연구에서 제시한 'Q-학습 알고리듬'은 최적해에 수렴하였다. 여기서, 최적해의 수렴 여부는 각각의 문제 세트에 대해서 열거법 또는 분지한계법을 활용하여 얻은 최적해와 비교를 통하여 확인하였다.

반면에 최적해에 수렴할 때까지의 컴퓨터 계산시간은 방문노드의 수(m)가 증가함에 따라서 매우 급격히 증가함은 물론 기존의 B&B 알고리듬에 비해서 m이 커질수록 차이가 더욱 뚜렷함을 알 수 있다. <Figure 13>은 방문노드의 수와 Q-학습 알고리듬의 평균계산시간과의 관계를 보여준다. 본 연구의 알고리듬이 분지한계법 등의 방법론에 비해서 오히려 많은 계산시간을 요구하지만 확률적인 시간자료가 주어질 경우에 있어서는 기존의 방법론으로 계산이 불가능한 점을 감안하면 그 효율성을 기대할 수 있다.

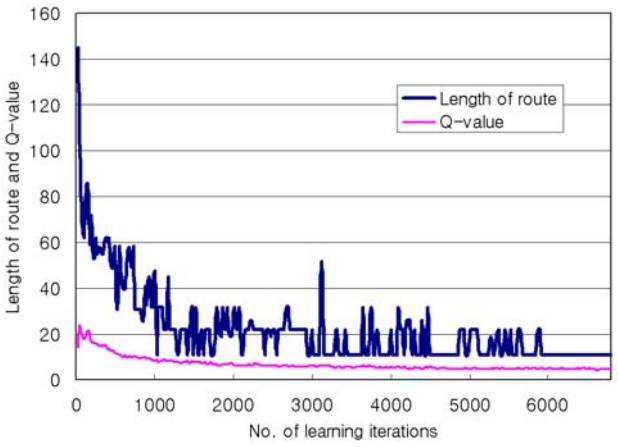

Figure 12. Results of Q-learning for TSP with deterministic time data.

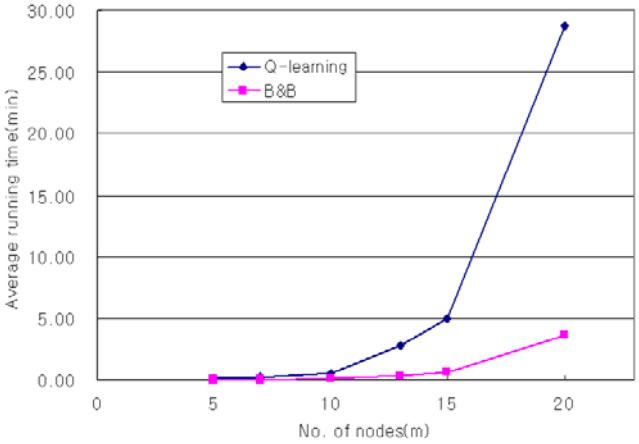

Figure 13. Average running time of Q-learning algorithm for STSP.

3.2 실험 2 - 확률적인 경우

실험 2에서는 수요지 간의 시간자료가 <Table 5>에 보는 바와 같이 평균과 분산에 의해서 주어지는 경우에 대해서 실험을 행하였다.

실험은 두 가지 방법으로 행하였는데, 하나는 <Table 5>의 자료 중 평균치만의 자료를 사용하여 알고리듬을 수행하였고, 다른 하나는 평균과 표준편차 모두를 고려하여 알고리듬을 수행하였다.

두 실험결과 얻어진 베스트 솔루션(Best solution) 및 시뮬레이션 분석결과는 <Table 6>과 같다. <Table 6>에서 'Best route'는 평균만을 고려한 경우와 평균 및 표준편차를 모두 고려한 경우의 각각의 베스트 솔루션을 의미한다. <Table 6>의 3～6행은, 얻어진 각각의 'Best route'에 대해서 10회의 시뮬레이션을 수행하여 경로시간의 최소, 최대, 평균 및 분산을 구한 결과를 나타낸다. <Table 6>의 결과를 살펴보면, 평균만을 고려한 경우에 비해서 평균과 표준편차를 모두 고려한 경우가 평균은

Table 4. Optimal convergence and average running time of Q-learning algorithm for STSP

No. of nodes		5	7	10	13	15	20
No. of experimental data set		10	10	10	10	10	5
No. of optimal convergence		10	10	10	10	10	5
Average running time(min)	B&B algorithm	0.01	0.02	0.07	0.35	0.62	3.59
	Q-learning algorithm	0.08	0.17	0.53	2.83	4.95	28.72

Table 5. Time data between customers(data in parenthesis mean (mean, standard deviation))

	0	1	2	3	4	5	6	7	8	9	10
0	(0,0)	(21,2)	(48,13)	(34,9)	(35,9)	(58,16)	(20,5)	(58,16)	(60,5)	(55,15)	(20,17)
1	(40,11)	(0,0)	(22,2)	(38,10)	(68,19)	(20,18)	(22,5)	(67,19)	(38,10)	(46,12)	(58,16)
2	(22,5)	(49,13)	(0,0)	(20,2)	(51,14)	(52,14)	(33,8)	(33,9)	(61,17)	(61,17)	(49,13)
3	(69,19)	(65,18)	(31,8)	(0,0)	(20,2)	(32,8)	(46,13)	(25,6)	(20,19)	(53,15)	(69,5)
4	(48,13)	(25,6)	(25,6)	(59,16)	(0,0)	(21,2)	(34,9)	(20,10)	(35,9)	(67,19)	(68,19)
5	(40,11)	(33,9)	(20,7)	(28,7)	(52,14)	(0,0)	(23,2)	(55,15)	(36,9)	(51,14)	(30,8)
6	(29,7)	(20,13)	(24,6)	(42,11)	(65,18)	(33,8)	(0,0)	(22,2)	(34,9)	(65,18)	(51,14)
7	(51,14)	(41,11)	(24,6)	(48,13)	(54,15)	(65,18)	(20,17)	(0,0)	(21,2)	(65,18)	(41,11)
8	(53,15)	(45,12)	(45,12)	(43,11)	(37,10)	(40,11)	(33,9)	(22,5)	(0,0)	(20,2)	(23,5)
9	(20,10)	(38,10)	(44,12)	(27,7)	(43,12)	(32,8)	(51,14)	(47,13)	(27,7)	(0,0)	(20,2)
10	(21,2)	(39,10)	(25,6)	(59,16)	(20,11)	(57,16)	(49,13)	(61,17)	(20,5)	(30,8)	(0,0)

Table 6. Analysis for the best solution and simulation results

	Case 1: considering only average(1)	Case 2: considering both average and variance(2)	Deviation(%) = $\left\vert\frac{(1)-(2)}{(1)}\right\vert\times 100$
Best route	0 → 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 0	0 → 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 0	–
Average	221.2	229.0	3.53
Minimum	166.9	221.2	32.53
Maximum	260.7	234.4	10.08
Variance	1051.8	22.05	97.90

다소 높지만 최대-최소 편차가 매우 작고 분산도 작아서 매우 안정적인 경로임을 알 수 있다. 다시 말해, 본 연구에서 제시한 알고리듬을 사용하면 시간치가 확률적으로 주어지는 경우의 TSP에서 평균뿐만 아니라 분산도 최소화되는 매우 안정적인 경로를 얻을 수 있음을 알 수 있다. 따라서, 이렇게 얻어진 경로는 실제 물류문제의 응용에 있어서 차량의 경로 스케줄 작성 시 매우 신뢰성이 높은 자료로 사용될 수 있을 것으로 기대된다.

4. 결 론

본 연구에서는, 시간자료가 확률적으로 주어지는 경우의 Q-학습과 NN을 사용한 TSP의 알고리듬을 새롭게 제시하였다. 실험을 통해서 제시된 알고리듬이 확정적 및 확률적인 시간자료를 가지는 TSP에서 모두 최적해에 수렴함을 알 수 있었다. 특히, 본 연구의 알고리듬은 시간치가 확률적으로 주어지는 경우에서 평균 및 분산을 동시에 최소화하여, 분산이 매우 작은 안정적인 경로를 발견할 수 있음을 알 수 있었다. 이러한 안정적인 경로의 생성은, 날로 도로의 사정이 복잡해지고 교통체증 등으로 인해서 각 경로에 대한 배송시간을 정확히 추정하기 힘든 상황에서 도시물류를 담당하는 택배 등의 물류회사의 신뢰성 있는 배송 스케줄을 작성할 수 있게 해주며, 이는 물류의 배송품질을 높이는 데 크게 기여할 수 있을 것으로 판단된다.

하지만, '실험 1'의 경우에서도 살펴본 바와 같이 본 연구에서 제시하고 있는 Q-학습을 활용한 알고리듬은 방문노드의 수가 증가함에 따라서 급격히 많은 계산시간을 요구하므로 실시간의 계산을 요구하는 물류문제에 직접적인 적용에는 아직까지는 무리가 있는 것으로 판단된다. 따라서, Q-학습에 포함된 NN의 학습방법과 Q-학습 자체의 방법 등에 대한 추가적인 연구가 있어야 할 것으로 판단된다. 다만 본 연구의 주된 목적이 확률적인 시간을 가지는 TSP의 알고리듬을 제안하는 데 있었고 제안한 Q-학습 알고리듬이 그러한 환경의 문제에 대한 해결의 실마리를 보여주었다고 사료된다.

또한, 본 연구에서 제시한 알고리듬이 현실적인 TSP에서 보다 더 효율성을 가지기 위해서는 NN의 학습과 Q-학습의 과정에서 학습의 효과에 커다란 영향을 끼칠 수 있는 파라미터의 최적설정에 대한 추가적인 연구와 수요지가 많을 경우 효율적으로 학습을 수행할 수 있는 방법론에 대한 연구가 추가적으로 요구된다. 더불어, Time window 등의 현실적인 제약을 반영

할 수 있는 방안에 대한 확장연구도 필요하다. 현재 계속되는 연구에서 알고리듬에 사용되는 학습률, 할인율 등의 파라미터의 설계에 대한 연구와 현실적인 제약을 반영하는 학습방법론에 대한 연구가 진행중에 있다.

참고문헌

E. P. C. (1978), A Preference Order Dynamic Program for a Stochastic Traveling Salesman Problem, *Operations Research*, **26**(6), 1033-1045.

Freeman, J. A. and Skapura, D. M.(1992), *Neural Networks (Algorithms, Applications, Programming and Techniques)*, Addison Wesley, USA.

Gambardella, L. M., and Dorigo, M. (1995), Ant-Q: A Reinforcement Learning approach to the traveling salesman problem, *Proceedings of the 11th International Conference on Machine Learning*, Morgan Kaufman, San Francisco, CA, 252-260.

Gendreau, M., Laporte, G. and Seguin, R. (1996), Stochastic vehicle routing, *European Journal of Operational Research*, **88**, 3-12.

Hagiwara, M. (1994), *Neuro・Fuzzy・Genetic Algorithm*, Sangyouzusho, Tokyo, Japan.

Kaelbling, L. P., Littman, M. L. and Moore, A. W. (1996), Reinforcement Learning: A Survey, *Journal of Artificial Intelligence Research*, **4**.

Kim, D. S.(1992), *Neural Networks(Theory and Applications)*, HightechInfo, Seoul, Korea.

Lambert, V., Laporte, G. and Louveaux, F. (1993), Designing collection routes through bank branches, *Computers and Operations Research*, **20**, 783-791.

Laporte, G., Louveaux, F. and Mercure, H. (1992), The vehicle routing problem with stochasitc travel times, *Transportation Science*, **26**(3), 161-170.

Lee, K. M. (1999), *Sequencing Delivery and Receiving Operations for Transfer Cranes*, MS Thesis, Pusan National University, Korea.

Leipala, T. (1978), On the solutions of stochastic traveling salesman problems, *European Journal of Operational Research*, **2**, 291-297.

Lin, F. and Pai, Y. H. (2000), Using Multi-Agent Simulation and Learning to Design New Business Processes, *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, **30**(3), 380-384.

Lin, L. J. (1993), *Reinforcement Learning for Robots Using Neural Networks*, Ph.D Dissertation, Carnegie Mellon University.

Sniedovich, M. (1981), Analysis of a preference order traveling salesman problem, *Operations Research*, **29**, 1234-1237.

Sutton, R. S. and Barto, A. G. (1998), *Reinforcement Learning: An Introduction*, The MIT Press.

Touzet, C. F. (1997), Neural reinforcement learning for behaviour synthesis, *Robotics and Autonomous Systems*, **22**, 251-281.