

외판원문제에 대한 유전알고리즘 성능평가

김동훈*, 김종율*, 조정복*

*동서대학교 컴퓨터정보공학부

Performance Evaluation of Genetic Algorithm for Traveling Salesman Problem

Dong Hun Kim* · Jong Ryul Kim* · Jung Bok Jo*

*Dongseo University Computer Information Engineering Division

E-mail : genetic.kim@gmail.com, xmaskjr@gdsu.dongseo.ac.kr, jobok@gdsu.dongseo.ac.kr

요 약

외판원문제(Traveling Salesman Problem: TSP)는 전형적인 조합최적화 문제로 위치하는 n 개의 모든 지점을 오직 한 번씩만 방문하는 순회경로를 결정하는 과정에서 순회비용 또는 순회거리를 최소화한다. 따라서 본 논문에서는 종래의 NP-hard문제로 널리 알려진 TSP를 해결하기 위해서 메타 휴리스틱기법 중에서 가장 널리 이용되고 있는 유전 알고리즘(Genetic Algorithm: GA)을 이용한다. 마지막으로, 유전 알고리즘을 이용해 외판원문제에 적합한 성능을 보이는 유전 연산자를 찾아내기 위해 수치 실험을 통해 그 성능에 대한 평가를 한다.

키워드

외판원문제(Traveling Salesman Problem), 유전 알고리즘(Genetic Algorithm), 유전 연산자 (Genetic Operator)

I. 서 론

외판원문제(Traveling Salesman Problem: TSP)는 전형적인 조합최적화 문제로 폭 넓은 응용 분야를 가진 문제로서 공학, 생물학, 물리학 분야 등의 문제들에도 많이 이용되고 있는 중요한 문제이다. 그 예로는 차량경로문제(Vehicle Routing Problem: VRP), X-ray 결정학실험문제, 직접회로 삽입(IC Insertion)문제, 항공기 노선의 유도날개 문제 등과 같이 다양한 분야에서 적용되고 있다. 1859년 Hamilton[1]이 처음으로 이 문제를 제시한 이후 수리계획법의 주된 관심 대상으로서 많은 연구가 되어져 왔다. 외판원문제는, n 개의 모든 지점을 오직 한 번씩만 방문하는 순회경로를 결정하는 과정에서 순회비용 또는 순회거리를 최소화하는 문제이다. 따라서, 종래의 NP-hard문제에 속하며, 흔히 계산 복잡도 이론에서 해를 구하기 어려운 문제의 대표적인 예로 많이 다루고 있다. 지난 수년간 많은 연구자들에 의해서 새로운 모델과 기법들이 제안되고 있다.

최근에 보편적이면서도 널리 실세계에 적용되는

최적화 문제들을 푸는데 많은 메타휴리스틱 방법들이 제안되고 있다. 이러한 메타휴리스틱 방법들 중에는 진화 연산법, 유전적 프로그래밍, 진화 전략 혹은 타부 검색(Tabu Search), 시뮬레이티드 어닐링(Simulated Annealing)등과 같은 방법들이 있다. 이들 가운데 유전 알고리즘은 매우 주목받는 최적화 방법들 중에 하나이다. 유전 알고리즘은 생물의 유전현상을 모방한 해 탐색 방법을 이용하고 있다. 유전 알고리즘은 비선형의 문제를 비교적 단시간에 용이하게 풀 수 있다는 점에서 주목받고 있고 여러 가지 분야에 적용되고 있다. 유전 알고리즘을 이용한 외판원문제제의 적용도 그 하나이다[2].

본 논문에서는 메타휴리스틱기법 중에서 가장 널리 이용되고 있는 유전 알고리즘을 이용한 유전 연산자의 성능을 평가 하고자 한다. 유전 알고리즘을 이용함에 있어서 외판원문제에 적합한 성능을 보이는 유전 연산자의 조합을 찾아 내기 위해서 수치 실험을 통해 그 성능에 대한 평가를 한다.

II. 문제기술

외판원문제는, n 개의 도시 집합과 도시 i 와 j 의 사이의 거리가 주어졌을 때, 모든 도시를 정확한 한 번만 방문해서 출발점으로 돌아올 때 가능한 페로(해밀톤 페로)중에서 거리가 최소인 것을 탐색하는 문제이다.

이 문제는 도시를 어느 차례로 순회할까를 요구하는 문제이지만, 출발점은 어느 도시에서 만나도 최종적으로 해는 페로가 되기 때문에 변화는 없다. 따라서, 어느 도시를 지나는 것이 문제가 된다. 그림 1은 6개 도시의 위치를 좌표로 나타내고 순회경로를 나타낸 간단한 예이다.

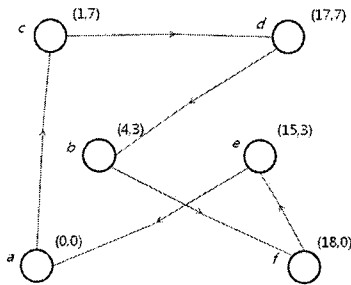


그림 1. TSP의 예

TSP를 설명하기 위해 필요한 기호들을 정의하면 다음과 같다.

첨자(Indices)

n : 도시의 집합 $V=\{1,2,\dots,n\}$

매개변수(Parameters)

d_{ij} : 도시 i 와 j 간의 거리($i,j \in V$)

의사결정변수(Decision Variables)

$x_{ij}=1$: 도시 i, j 간을 지나갈 경우

$x_{ij}=0$: 도시 i, j 간을 지나지 않을 경우

이 문제를 정식화하면 아래와 같이 나타낼 수 있다[3].

$$\min \sum_{i=1}^n \sum_{j=1, j \neq i}^n d_{ij} x_{ij} \quad (1)$$

$$\text{s.t.} \sum_{j, j \neq i} x_{ij} = 1, \quad i = 1, 2, \dots, n \quad (2)$$

$$\sum_{i, i \neq j} x_{ij} = 1, \quad j = 1, 2, \dots, n \quad (3)$$

$$x_{ij} \in \{0, 1\}, \quad i, j = 1, 2, \dots, n \quad i \neq j \quad (4)$$

목적 함수 (1)은 모든 도시를 정확히 한 번만 방문해서 출발점으로 돌아오는 총 순회거리를 최소화하는 식이다. 제약식 (2)은 도시 j 는 정확히 한번만 도시를 방문하는 제약조건이고 제약식 (3)은 외판원이 동시에 2개의 도시를 방문할 수 없는

제약조건이다. 제약식(4)은 도시를 지나는 경우는 1, 지나지 않을 경우는 0으로 나타내는 0-1 변수로 i 와 j 는 같을 수 없다.

III. TSP를 위한 유전 알고리즘 설계

3.1 유전 알고리즘

유전 알고리즘(GA)은 1960 년대에 John Holland 에 의해 창안되었다. Holland는 1975 년도 저서인 「자연과 인공 시스템에서의 적응 (Adaptation in Nature and Artificial Systems)」을 통해 GA를 생물학적 진화의 추상적 개념으로 진술함과 동시에 생물유전의 원리를 바탕으로 하여 최적의 방법을 제시하였다. GA는 접근하는 문제에 대해서 가능한 해들을 정해진 형태의 자료구조로 표현한 다음, 이들을 점차적으로 변형함으로써 점점 더 좋은 해들을 생성하게 된다. 이는 세대가 진행될수록 환경에 대한 적합도가 높고 상대적으로 우월한 해가 남는다는 이론에 근거한 것이다.

3.2 유전 연산자

3.2.1 교차 연산자

교차(Crossover)는 아주 중요한 유전 알고리즘의 연산자이다. 이는 서로 다른 부모 유전자를 교차에 의해 새로운 자손을 생성하는 과정에서 이용되는 연산자이다. 이는 유전 알고리즘에 있어서 새로운 해 영역을 탐색할 때, 해 영역의 지역탐색이라 한다. 순열 교차(Permutation Crossover) 연산자, 순서기반 교차(Order-based Crossover) 연산자, 위치기반 교차(Position-based Crossover) 연산자를 교차 연산자로 사용하였다.

3.2.1.1 순열교차(Permutation Crossover)

순열교차는 두 개의 인자위치를 임의로 선택하여 선택한 위치 사이에 있는 한 부모(P1)와 다른 부모(P2)사의 값을 순서대로 위아래를 바꾼다. 그리고 인자값과 똑같은 값을 임의로 선택한 이외의 위치에서도 똑같이 바뀌어서 자손(O1, O2)을 얻어낸다. 아래와 같은 방법으로 자손을 얻을 수 있다.

$$P1 = (2 \ 5 \ 0 \ 3 \ 6 \ 1 \ 4 \ 7)$$

$$P2 = (3 \ 4 \ 0 \ 7 \ 2 \ 5 \ 1 \ 6)$$

$$O1 = (6 \ 1 \ 0 \ 7 \ 2 \ 5 \ 4 \ 3)$$

$$O2 = (7 \ 4 \ 0 \ 3 \ 6 \ 1 \ 5 \ 2)$$

3.2.1.2 순서기반교차(Order-based Crossover)

순서기반교차는 Syswerda[4]에 의해 제안되었다. 이 교차는 여러 인자위치를 임의로 선택하여 그 위치에 있는 다른 부모(P2)의 인자값과 같은 값을 갖는 인자를 한 부모(P1)에서 삭제하고 자손(O1)에 상록하고, 미확정된 위치에 부모(P2)에서 나타난 순서대로 선택된 인자를 복사한다. 다른 자손(O2)는 부모 P1과 P2의 역할을 바꾸어 생산

한다.

3.2.1.3 위치기반교차(Position-based Crossover)

위치기반교차와 순서기반교차는 Syswerda[4]에 의해 제안 되었다. 이 교차는 임의로 몇 개의 위치를 선택하여 한 부모에서 선택된 인자의 위치와 인자값을 자손에 그대로 상속하고, 다른 부모에서 선택된 인자값을 갖는 인자들을 지우고 나타난 순서대로 미확정된 자손의 인자위치에 유전하는 교차이다.

3.2.2 돌연변이 연산자

개체의 각 유전자에 대하여 일정한 돌연변이 확률을 적용하여 대립 유전자의 값으로 바꾸는 것이다. 개체에 근접한 새로운 개체를 생성하는 국소적인 랜덤 탐색의 일종이다. 또한 집단에서 잃어버린 유전형질을 복구하여 다양성을 유지하기 위한 수단으로도 사용된다. 돌연변이를 너무 큰 변이확률로 설정하면 적합도가 떨어지는 경우도 생기게 되지만 오히려 적합도가 높아지는 경우도 생긴다. 국부적 최적해(local minimum)에 빠질 위험을 배제할 수 있다. 돌연변이 연산자로는 교환(Exchange) 연산자, 전환(Displacement) 연산자, 삽입(Insertion) 연산자를 사용하였다.

3.2.2.1 교환 돌연변이(Exchange Mutation)

교환 연산자는 부모에서 서로 같지 않은 임의의 두 인자를 교환하여 자손을 생산하는 방법이다. 예를 들어 아래와 같이 부모(P1)에서 교환될 인자로 네 번째와 여덟 번째 인자 4와 8이 임의로 선택되었다면 선택된 두 인자를 서로 교환하여 자손 (O1)을 생산한다[5].

P1 = (1 2 3 4 5 6 7 8 9)
 O1 = (1 2 3 8 5 6 7 4 9)

3.2.2.2 전환 돌연변이(Displacement Mutation)

전환 연산자는 염색체 내에서 두 개의 절단점을 임의로 선택하고 절단점 사이의 인자들을 임의의 위치로 전위하여 자손을 생산하는 방법이다[5].

3.2.2.3 삽입 돌연변이(Insertion Mutation)

삽입 연산자는 랜덤으로 한 유전자를 선택하여 다시 랜덤으로 위치를 선택하여 삽입하는 방법이다[6].

IV. 수치실험 및 결과

본 논문에서는 제안한 유전 연산자의 성능을 평가하기 위해 수치 데이터를 수집하고 TSP를 위한 각각의 교차와 돌연변이를 이용한 9가지 방법을 이용하여 비교하였다. 실험을 위해 Intel Pentium(TM)2 Duo CPU 2.40Ghz를 이용하여 유전 알고리즘을 작성하여 수치실험 하였다. 유전 알고리즘에 사용된 유전 연산자들은 표 1과 같다.

표 1. 외판원문제에 사용된 연산자

Operator	Parameter
초기모집단	200
교배확률	0.2
돌연변이확률	0.7
교배연산자	Permutation, Position-Based, Order-Based
돌연변이연산자	Inversion, displacement Insertion
종료조건	최적값을 구하면 자동종료

표2. 유전 연산자에 대한 실험결과 비교

Crossover + Mutation	Min Generation	Max Generation	Average Generation	STDV	Average Time(s)
Permutation + Insertion	12648	68680	41552	18997	1409
Permutation + Displacement	34669	135934	77852	31643	3201
Permutation + Exchange	-	-	-	-	-
Order-based + Insertion	9651	21146	15274	4690	492
Order-based + Displacement	29510	49413	37317	7144	1247
Order-based + Exchange	-	-	-	-	-
Position-based + Insertion	7460	37569	19931	8227	5193
Position-based + Displacement	3379	11376	52280	32302	6148
Position-based + Exchange	15880	115678	72145	31172	12713

- : 실험시간이 2시간 이상

각각의 유전 연산자에 대한 비교를 위해 노드의 수를 일정하게 100개로 하고 노드의 위치를 원형으로 정해 놓았다. 이 문제에서는 원형으로 이루어졌을 때가 최적일 때이다. 각각의 유전 연산자에 따라 초기모집단수(Population)는 200으로 하여 실험을 수행하였다. 모든 실험 데이터상에서 각 알고리즘을 10번씩 수행하였으며, 실험결과는 수행 후 출력된 결과값에 대해 최소 세대수(Min Generation), 최대 세대수(Max Generation), 평균 세대수(Average Generation), 표준편차(STDV), 그리고 이에 대한 평균시간(Average Time)을 중심으로 나타내었고, 그 결과 데이터는 표 2와 같다. 순열 교차와 조합으로는 삽입 돌연변이가 가장 좋은 최소 세대수, 최대 세대수, 평균 세대수, 표준편차, 평균시간을 얻어 낼 수 있었다. 순서기반 교차와의 조합으로는 삽입 돌연변이가 가장 좋은 최소 세대수, 최대 세대수, 평균 세대수, 표준편차, 평균시간을 얻어 낼 수 있었다. 마지막으로 위치기반 교차와 전환 돌연변이의 조합으로는 최소 세대수와 최소 세대수가 가장 좋게 나왔고, 삽입 돌연변이와의 조합으로는 평균 세대수, 표준편차, 평균시간이 가장 좋은 값을 얻을 수 있었다.

V. 결 론

본 논문에서는 TSP에 대해서 유전 연산자인 순열 교차, 순서기반 교차, 위치기반 교차와 삽입 돌연변이, 전환 돌연변이, 교환 돌연변이를 이용하여 각각에 대한 성능을 비교하였다.

순열 교차는 삽입 돌연변이와 조합 했을 경우, 가장 좋은 최소 세대수, 최대 세대수, 평균 세대수, 표준편차, 평균시간을 산출해 냈다.

순서기반 교차는 삽입 돌연변이와 조합 했을 경우, 가장 좋은 최소 세대수, 최대 세대수, 평균 세대수, 표준편차, 평균시간을 산출해 냈다.

위치기반 교차는 전환 돌연변이와 조합 했을 경우, 최소 세대수와 최소 세대수가 가장 좋게 나왔고, 삽입 돌연변이와 조합 했을 경우, 평균 세대수, 표준편차, 평균시간이 가장 좋은 값을 얻을 수 있었다.

본 실험을 통해 9가지의 유전 연산자를 비교해본 결과 순서기반 교차와 삽입 돌연변이가 다른 연산자들에 비해 가장 안정되고 최적의 비용 값을 산출함이 증명되었다. 따라서 TSP에 있어서 순서기반 교차와 삽입 돌연변이가 가장 적합하다는 것을 확인 할 수 있었다.

참고문헌

- [1] S. Kabadi, A. P. Punnen (2003). "Weighted graphs with all Hamiltonian cycles of the same length", *Discrete Mathematics*, Vol. 271, Issues 1-3, pp. 129-139.
- [2] M. Gen and R. W. Cheng (2000). *Genetic Algorithm and Engineering Optimization*, John Wiley and Sons, New York.
- [3] I. Kara, T. Bektas (2006). "Integer linear programming formulations of multiple salesman problems and its variations", *European Journal of Operational Research*, Vol. 174, Issue 3, pp.1449-1458.
- [4] G. Syswerda (1991). *Schedule Optimization Using Genetic Algorithms*, Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 332-349.
- [5] M. Gen, and R. Cheng (1997). *Genetic Algorithms and Engineering Design*, John Wiley and Sons, New York.
- [6] M. Gen and R. Cheng and L. Lin (2008). *Network Models and Optimization: Multiobjective Genetic Algorithm Approach*, Springer Verlag.