JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 25 No. 12, pp. 55-61, December 2020
https://doi.org/10.9708/jksci.2020.25.12.055

An Efficiency Analysis on Mutation Operation with TSP solved in Genetic Algorithm

Hoijin Yoon*

*Associate Professor, Dept. of Computer Engineering, Hyupsung University, Gyeonggi-do, Korea

[Abstract]

Genetic Algorithm(GA) is applied to a problem that could not figure out its solution in a straightway. It is called as NP-hard problem. GA requires a high-performance system to be run on since the high-cost operations are needed such as crossover, selection, and mutation. Moreover, the scale of the problem domain is normally huge. That is why the straightway cannot be applied. To reduce the drawback of high-cost requirements, we try to answer if all the operations including mutation are necessary for all cases. In the experiment, we set up two cases of with/without mutation operations and gather the number of generations and the fitness of a solution. The subject in the experiment is Travelling Salesman Problem(TSP), which is one of the popular problems solved by GA. As a result, the cases with mutation operation are not faster and the solution is fitter than the case with mutation operation. From the result, the conclusion is that mutation operation does not always need for a better solution in a faster way.

▸ **Key words**: Mutation Operation, Genetic Algorithm, Travelling Salesman Problem, Efficiency, Fitness

[요 약]

유전자 알고리즘은 명료한 방식으로 답을 찾기 어려운 문제, 즉 NP 문제의 경우 효과적인 솔루션을 찾을 수 있다. 단 유전자 알고리즘의 실행 비용은 기존 프로그래밍 방식에 비하여 높은 비용을 요구하게 되므로, 높은 성능의 실행환경을 전제로 한다. 이러한 문제를 조금이나마 줄여보기 위하여 본 연구는 유전자 알고리즘의 돌연변이 연산자를 초점을 맞추고, 돌연변이 연산의 복잡한 실행을 위한 비용을 고려하여, 과연 해당 연산자가 모든 문제 영역에서 반드시 요구될까를 분석하기 위한 실험을 진행한다. 우리 실험 주체는 유전자 알고리즘을 적용하는 대표적인 문제 중의 하나인 TSP(Travelling Salesman Problem)으로 하였다. 돌연변이 연산을 적용하는 경우와 적용하지 않는 경우에 대한 결과값들을 세대수와 적합도 값을 수집하여 분석한다. 그 결과 돌연변이 연산자를 적용하는 경우가 세대수 감소와 적합도 향상의 효과적인 결과를 반드시 보이지는 않았다.

▸ **주제어**: 돌연변이 연산자, 유전자 알고리즘, 순회 세일즈맨 문제, 효율성, 적합성

• First Author: Hoijin Yoon, Corresponding Author: Hoijin Yoon
*Hoijin Yoon (hjyoon@uhs.ac.kr), Dept. of Computer Engineering, Hyupsung University
• Received: 2020. 09. 23, Revised: 2020. 11. 30, Accepted: 2020. 12. 09.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

유전자 알고리즘(Genetic Algorithm : GA)은 다윈의 자연 진화 이론에 영감을 얻어서, Holland가 1975년에 개발하여 발표한 알고리즘이다[1]. GA의 첫 번째 버전은 최적화 문제를 풀기 위한 것이었다. 이는 경험적 탐색 접근법으로서, 진화 개념을 적용하여 문제를 해결한다. 즉, 개체들은 상황에 적합한지를 평가받고 적합한 개체는 살아남고 나머지는 사라지며, 살아남은 개체들은 교배를 통하여 다른 세대를 생산해낸다. 이렇게 세대를 반복하여 지나는 중에, 최적으로 적합한 개체 그룹이 만들어지며, 이 그룹의 유전자가 해당 문제의 솔루션으로 선택되어진다. GA가 다루는 연산들은 선택(Selection), 교배(Crossover), 돌연변이(Mutation), 우생학(Elitism) 등이다. 선택, 교배, 우생학은 더 나은 개체들을 생산하기 위한 것들이다[2]. 이에 반하여 돌연변이는 답을 찾아서 점점 진화하는 개체의 방향성을 흐트러뜨리는 역할을 하게 된다[3,4]. 즉, 산을 올라가는 방식(Hill Climbing)과 같이, 바로 앞의 땅이 높은 쪽으로 계속 진행하면 어느 순간 산꼭대기에 오를 수 있으나, 그 산 꼭대기보다 더 높은 꼭대기로는 갈 수 없는 상황이 되고 만다. 이 문제를 최적화 문제에서는 local optimum 문제라고 한다. GA는 이 문제에 대한 해법으로 돌연변이 연산을 활용한다. 돌연변이는 우수한 유전자를 갖는 개체들끼리 교배하여 얻어진 다음 세대 개체들의 유전자를 변형시켜 local optimum으로 가는 일을 방지하는 역할을 한다.

GA를 적용하는 문제들은 솔루션 후보의 규모가 매우 커서 명시적인 접근이 어려운 경우들이다. 이를 NP-hard 문제라고 한다. GA는 문제 도메인의 규모가 크며, 연산이 복잡하여, 실행을 위한 시스템의 높은 성능을 요구한다. 즉 실행 비용이 크다고 할 수 있다. 우리는 GA의 비용을 줄이기 위한 방법으로 돌연변이 연산의 제거를 고려해보았다. 돌연변이 연산의 의미는 local optimum을 해결하기 위한 것이라고 한다면, 돌연변이 연산을 적용한 경우의 솔루션이, 돌연변이 연산을 적용하지 않는 솔루션보다 적합도가 크다는 기대가 가능하다. 그러나 만일 반드시 그렇지 않다면, GA에서 돌연변이 연산을 적용하지 않아도 그 솔루션이 의미가 있다고 볼 수 있으며, 항상 돌연변이 연산을 포함할 필요는 없다.

본 연구는 “돌연변이 연산을 적용하면 솔루션을 얻는 세대수는 짧아지고 적합도는 높아지는가? 즉 비용이익면에서 효과적일까?”를 RQ(Research Question)으로 하여, (1) 돌연변이 연산을 제거하여 local optimum을 피하기 위하여 투입하는 노력을 절감하는 효과와 (2) 솔루션의 fitness값을 비교하여, (1)과 (2)사이의 비용-이익관계(trade-off)를 분석한다.

2장은 돌연변이 연산을 포함한 GA의 연산자들을 소개하고, 본 연구의 예제로 활용하는 TSP(Travelling Salesman Problem)을 소개한다. 3장은 실험을 위하여 TSP를 GA로 구현하고 이를 설계된 조건에 따라 실행하여 결과를 측정한다. 4장은 실험 결과를 기반으로 비용-이익 관계를 분석하며, 5장은 결론을 언급한다.

II. Preliminaries

1. Operations in GA

GA에서 주로 사용되는 연산은 크게 3가지- 복제(Reproduction), 돌연변이(Mutation), 교배(Crossover)-이다. ‘복제’는 선택된 개체를 그대로 다음 세대(generation)로 넘기는 연산을 의미하며, ‘돌연변이’는 랜덤 선정된 일부 영역을 거꾸로 뒤집어서 원래의 유전자 모양을 변형시키는 연산이다. 이렇게 변형된 유전자를 갖는 개체는 세대에 다양성을 제공하여 local optimum 문제 예방을 위하여 활용된다. ‘교배’는 랜덤하게 선정된 교배 포인트를 중심으로 두 개의 개체를 두 파트로 나누고 짝으로 선정된 개체와 나누어진 부분을 서로 어긋나게 연결시킴으로써 새로운 후손(offspring)을 만들어낸다.

복제와 돌연변이는 하나의 개체에 적용하는 연산자이며, 교배는 두 개의 개체에 적용한다. 즉, 이들 연산자의 피연산자가 되는 개체를 우선 선택해야 하는데, 주로 적합도(Fitness)값을 활용한다. 적합도는 적합도 함수가 결정하며, 적합도란 생성된 솔루션이 얼마나 문제 해결에 적합한지를 나타내는 척도이다. 적합도 값이 높을수록 더 좋은 솔루션으로 볼 수 있다.

GA는 후손을 만들어가면서 일정 적합도 이상을 갖는 개체를 솔루션으로 찾는다. 후손을 만들어 구성하는 세대의 수가 적을수록, 그리고 솔루션의 적합도가 높을수록 효과적이라고 평가할 수 있다

2. Optimization Problem

GA 연구들은 유전자 알고리즘을 적용하여 (1) 기존의 어려운 문제를 푸는 연구들과 (2) GA을 동작시키는 주요 연산자, 즉 교배(Crossover), 선택(Selection), 돌연변이(Mutation) 연산자를 개발하는 연구들과 (3) GA의 단점인 고비용에 대한 해결책으로서 GA의 각 단계를 분석하여 효

과적인 방법을 개발하는 연구들이 있다[5]. 본 연구는 이 가운데 (3)에 해당된다.

이렇게 GA의 효과성을 위한 연구들은 본 연구의 기본 설정과 유사하게 답을 찾기 위해 거치는 세대(generation) 또는 반복(iteration)의 수를 비용(cost)으로 보고, 제안한 솔루션의 적합성을 효과(benefit)로 본다. 따라서 세대 수를 최소화하는 동시에 솔루션의 적합성을 최대로 만드는 방안들을 제안하고 있다. Kim[6]은 비용을 연산실행 횟수로 정의하고, GA의 최소비용을 위하여, 세대 구성원 전체를 모두 교배하는 대신 세대 구성원(population) 전체를 분류(clustering)하여 각 묶음(cluster)의 대푯값에만 교배 연산자를 적용하는 방법을 제안하였다. 그리고 Wang[7]은 본 연구와 같이 세대 수를 비용으로 정의하고, 보다 짧은 세대 수를 거치는 최소비용 경로를 찾을 수 있도록 TSP에 적용한 GA을 변형하였다. 즉, 각 세대에서 생성된 솔루션의 경로에서 꼬임이 발생하는 경우에 대하여, 솔루션의 꼬임을 푸는 *untwist*연산을 개발하여 적용함으로써, TSP 최적 솔루션에 보다 빨리 도달하도록 하였다.

이렇게 GA를 효과적으로 진행하기 위한 연구들과 같이, 본 연구는 돌연변이 연산으로 인한 GA의 효과성을 평가한다. GA에서 돌연변이 연산자를 적용하는 목적은 최종결과물이 지역적 최적 솔루션 (local optimum)머무는 것을 방지하여 최종 솔루션의 적합도를 높이기 위함, 즉 혜택을 높이고자 함이다. 지역적 최적 솔루션을 피하기 위한 최적화 문제는 매우 큰 볼륨의 가능한 솔루션들 가운데 하나를 찾아 최적 솔루션으로 명시한다. 이때 솔루션을 평가하는 척도(Criterion)를 설정하여, 척도에 의하여 높은 적합도를 갖는 값을 찾는 노력을 지속적으로 하여 일정수준에 도달하는 경우, 해당 값을 최적값으로 간주하게 된다. 최적값을 지속적으로 찾아가는 과정에서 만나는 지점들이 있는데, 이들을 지면의 높낮이로 표현하여, peak, valley, plateau로 구분한다. 최적화 문제는 plateau를 찾고자 하나, 가끔은 peak에 머물고 plateau로 진행하지 못하는 경우들도 있다. 이때 돌연변이 연산을 적용하면, 확인 가능한 시야를 넓힐 수 있으므로, peak로 표현되는 지역적 최적 솔루션에 머무르는 일을 줄일 수 있다. 이것이 돌연변이 연산을 적용하는 의미이다.

본 연구는 GA의 효과성을 지원하기 위하여 GA의 비용 즉 세대 수와, 혜택 즉 최종결과물의 적합도의 상호관계가 돌연변이 연산자의 적용 여부에 영향을 받는지를 평가하였다. 이때 비용도 함께 평가하여, 돌연변이 연산의 효과성을 분석한다.

3. TSP(Travelling Salesman Problem)

TSP는 유전자 알고리즘을 이용하여 문제를 해결하는 고전적 문제 가운데 하나이다. 어느 한 도시에서 시작하여 연결된 다른 도시를 방문하고 그 자리로 돌아오는 길을 찾아주는 문제로서, 최소거리를 여행하도록 디자인한 최소비용 경로를 솔루션으로 찾아야 한다. TSP는 NP-complete 문제로서, 기존의 정공법으로는 해결하기 어렵고, 또한 그 복잡도가 매우 높은 문제이다. 본 연구에서는 TSP를 유전자 알고리즘으로 해결하기 위한 코드를 파이썬으로 작성하여 실험에 활용하였다[8,9].

유전자 알고리즘을 적용하여 TSP를 해결하는 연구로서, 앞서 언급한 Wang[7]의 연구와 같이 그 성능 향상을 위한 특별한 연산자를 제시하거나, 또는 Lu[10]의 연구와 같이 보다 특별한 선택(selection) 연산을 적용하여 답을 찾기까지의 세대 수를 줄이기도 한다. Lu의 선택 연산 대신 교배와 돌연변이 연산에 휴리스틱 정보를 적용하여 비용절감 경로를 빨리 찾도록 하는 연구[11]도 있다. 본 연구는 특히 돌연변이 연산이 GA에서 갖는 의미를 분석하여, 돌연변이 연산을 통하여 얻는 적합도 향상과 세대수 감소를 어느정도 보장받을 수 있을지를 보이고자 한다.

TSP는 도시 간의 거리를 최소로 하는 경로를 다음의 과정으로 구하게 된다. 가능한 경로 100개를 먼저 구하고, 각 경로에 대한 거리를 계산한다. GA에 의하면 100개의 후보 경로들은 population으로 관리되며, 100개 크기를 갖는 population으로 한 세대가 구성된다. 이때 구해진 거리가 짧을수록 좋은 적합도, 즉 솔루션으로써 적합하게 되므로, 경로 거리의 합과 적합도는 서로 반비례 관계를 갖는다. 따라서 총 거리의 합을 분모로 하여 적합도를 계산한다.

한 세대를 이루는 100개 경로들의 각 적합도 값을 기준으로 상위 50%를 선택, 즉 우생학에 따른 선택을 한다. 우수한 이들을 서로 교배하여 다음 세대를 구성하게 된다. 이때 돌연변이 연산이 함께 적용되며, 이는 설정된 돌연변이 정도, 즉 mutation_rate에 따라 돌연변이를 할 대상의 크기가 달라진다. 일반적으로 mutation_rate를 0.2, 즉 20%에 대하여 돌연변이 연산을 진행한다.

아래 그림은 10개의 도시일 경우, 우리 프로그램을 실행하여 TSP 문제를 해결한 결과를 보여주며, 그림의 내용은 71번째 세대에 나타난 하나의 유전자를 10개 도시 방문 경로로 표현하였다. 도시 10개의 위치를 x, y 좌표로 설정하고, 그에 대한 거리를 피타고라스 정리를 적용하여 구한다. 그 거리의 합을 구하고, 그 값의 역수로 적합도를 구하여, 설정한 적합도 이상의 경로를 찾아가게 된다. 찾

아진 경로를 솔루션으로 제시하면, TSP 프로그램 실행이 종료된다. 아래 그림에서 돌연변이는 0.2를 적용하였다.

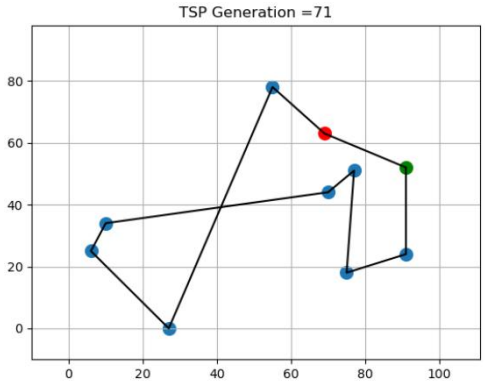

Fig. 1. A Sample Solution from TSP programmed in GA

III. Experiment

1. TSP Implementation

우리 실험을 위하여 TSP를 파이썬으로 구현하였으며, 도시 50개를 적용하였다. 초기 설정값들은 다음과 같으며, 이때 mutation_rate를 변경하면서 실험을 진행한다.

```
POPULATION = 100 #경로의 개수
SELECTION = 0.5 #선택 비율
CitySize = 50 #경로 안에 들어있는 city 개수
```

한 세대를 구성하는 개체의 규모, 즉 population은 100으로 하여, 한 세대에 100개의 후보경로들이 존재하도록 하였다. 100개 개체 가운데 높은 적합도를 갖는 50개를 선정하도록 Selection의 값은 0.5로 하였다. 이때 mutation_rate는 0.2를 기본으로 하며, 우리 실험에서 mutation_rate가 독립변수이므로 이 값을 0으로 설정하기도 한다.

그림 2는 GA로 구현한 TSP 실행 결과의 한 예이다. 그림의 예는 113 세대가 지났을 때, 조건에 맞는 솔루션 경로를 찾았으며, 그 적합도는 0.512인 경우이다. 이와 같은 과정을 mutation_rate가 0인 경우와 0.2인 경우로 나누어 실행하고, 확률적 실행에 의한 무작위성을 고려하여 20회 반복 실행하여 분포를 확인하는 방법으로 프로그램을 활용한다.

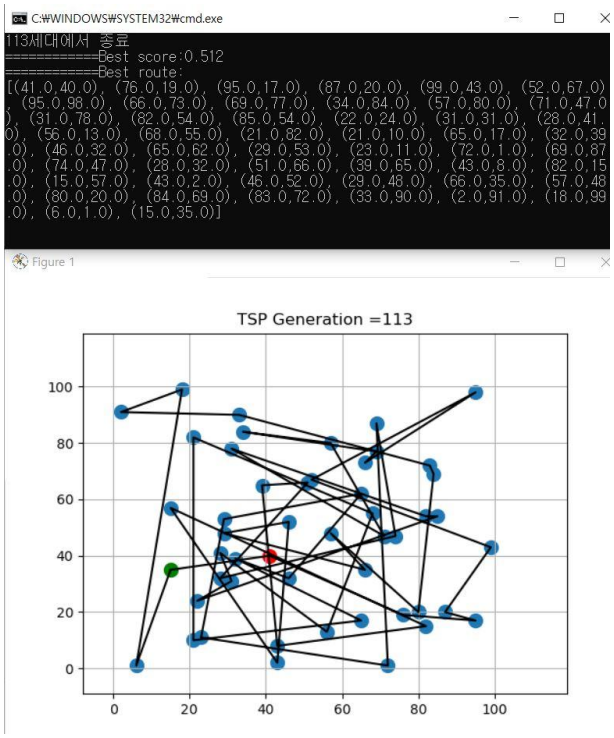

Fig. 2. A Result of TSP program in the experiment

2. Experiment Settings

우리는 실험을 통하여 돌연변이 연산의 적용 여부가 실행 결과에 어떤 영향을 주는지를 분석하고자 한다. 이 분석을 우리는 비용에 대한 이익 분석이라고 하였다. 비용은 “솔루션을 찾을 때까지의 세대수”로 볼 수 있고, 이익은 “솔루션이 갖는 적합도”로 볼 수 있다. 결국 짧은 세대수를 거쳐서 높은 적합도의 솔루션을 얻는 것이 비용이익면에서 효율적이다[5].

구현한 TSP 프로그램은 동일한 돌연변이 연산 적용에 대하여 실행할 때마다 같은 결과를 보장하지 않는다. 즉 매번 다른 세대수와 솔루션 적합도를 나타낸다. 그 이유는 GA의 특성상 난수 발생을 활용하는 지점들이 존재하기 때문이다. 교배방식과 돌연변이 방식에 임의선택이 포함된다. 따라서 우리는 여러 번 반복하여 그 트렌드를 파악할 필요가 있다.

실험의 독립변수는 mutation_rate, 즉 돌연변이 연산을 어느 정도로 적용할지를 결정하는 변수이다. 그에 대한 의존변수는 일정수준 이상의 적합도를 갖는 솔루션을 생성하게 되는 세대 수와 그의 적합도 값이다.

TSP를 GA로 구현한 프로그램을 실행할 때, 동일한 설정에 대하여 첫 번째 시도와 두 번째 시도가 다른 결과를 나타낸다. 그 이유는 GA에서 선택과 교배 그리고 돌연변이 연산이 확률적 임의성(randomness)을 보인다. 앞서 언급한대로 독립변수 mutation_rate를 변경하면서 세대수와 해당 솔루션의 적합도를 구할 때, 임의선택 절차로

인하여 그 결과값이 매번 다르게 생성된다. 따라서 이와 같은 비일관성을 고려하여 본 실험은 동일한 mutation_rate에 대하여 TSP 프로그램 실행을 반복적으로 실시하여 솔루션을 생성하는 세대수와 해당 솔루션의 적합도 값들이 어떤 흐름을 갖는지를 분석한다.

IV. Discussion

1. Research Question

본 연구는 유전자 알고리즘이 갖는 돌연변이 연산의 역할에 대한 고찰로부터 시작하였다. 돌연변이 연산은 솔루션이 답으로 가지 못하고 정체하는 현상 즉 지역적 최적 솔루션에 머물게 되는 것을 막기 위하여 적용되는 연산이다. 이때 세대 수를 '비용'으로 보고, 지역적 솔루션을 지나 더 좋은 솔루션을 내는 것을 '이익'으로 볼 때, 돌연변이 연산을 적용하면 짧은 세대를 거쳐 높은 적합도를 갖는 솔루션을 갖게 되는 결과를 예상할 수 있다. 우리는 이를 다음의 RQ로 정의하고, 그에 대한 답을 찾기 위한 실험을 설계하였다.

RQ : 돌연변이 연산을 적용하면 세대수는 짧아지고 적합도는 높아지는가? 즉 비용이익면에서 효과적일까?

RQ에 대한 답을 얻기 위하여 우리는 3장에서 TSP를 유전자 알고리즘으로 구현하여 실험을 설계하였다. 돌연변이 연산을 하는 경우와 하지 않는 경우로 나누고, mutation_rate를 '독립변수'로 하고, 각 경우에 일정수준 이상의 솔루션을 내는 세대 수와 솔루션의 적합도 값을 '의존변수'로 설정하여 실험을 진행하였다.

2. Relation between Fitness and Generations

실험을 통하여 산출되는 결과값을 분석하기 위하여, 세대수를 x축으로 하고 적합도를 y축으로 하는 좌표평면으로 만들었다. 이때 각 분면을 크게 4개로 나누어볼 수 있으며, 그림 3과 같다.

Fig. 3. Relation of Fitness and the number of generations

A분면에 실험 결과가 놓이는 경우가 가장 비용이익면에서 효과적이며, D분면의 경우는 가장 비효과적 결과가 된다. 돌연변이 연산을 적용하는 경우와 적용하지 않는 경우의 실행 결과를 아래 좌표평면에 분포시키고, 그 분포가 A분면에 있을수록 비용이익면에서 효과적인 것이므로, 우리가 독립변수로 설정한 돌연변이 연산의 적용 비율, 즉 mutation_rate를 조정하여 실험을 진행한다.

mutation_rate를 0으로 하는 실행과 mutation_rate를 0.2로 설정한 실행 결과는 그림 4와 같은 분포를 보인다. 첫 번째 분포는 mutation_rate를 0으로 설정하여 TSP의 솔루션을 찾았다. 이를 반복적으로 실행하여 얻은 세대수와 적합도 값으로 그린 분포도이다. 앞서 분석한 4개의 분면 가운데 A분면에 주로 놓여 있다.

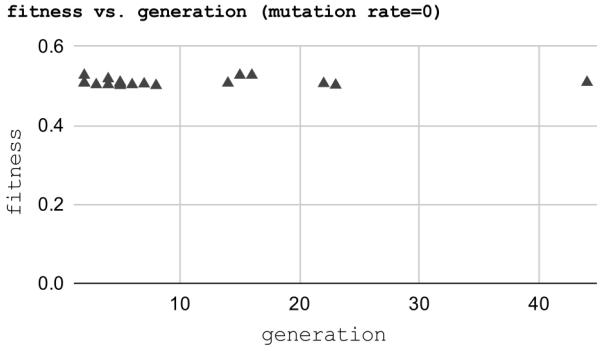

Fig. 4. Solutions in the case where mutation_rate is 0

mutation_rate를 0.2로 설정하여 같은 TSP 솔루션을 찾는 프로그램을 실행시켰다. 돌연변이 연산을 적용하여 지역적 최적 솔루션에 머무는 것을 방지하기 위한 연산을 적용한 것이다. 반복 실행하여 얻은 실행 결과값을 그림 5의 분포도로 표현하였다. A분면에 위치해 있으나, mutation_rate가 0인 경우에 비하여 좀 더 많은 경우가 B분면으로 펼쳐져 있다.

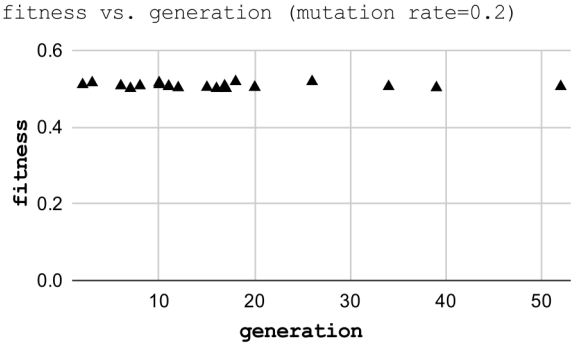

Fig. 5. Solutions in the case where mutation_rate is 0.2

돌연변이 연산을 적용한 경우가 적용하지 않는 경우에 비하여 더 효과적인 결과를 보일 것으로 기대하였으나, 세대수 면에서는 더 효과적인 결과로 보기 어려운 분포를 보였다. 그렇다면 적합도가 더 높은 솔루션을 얻은 것일까? 돌연변이 연산을 적용하지 않은 mutation_rate를 0으로 설정한 경우의 반복 실행에 대한 적합도 평균은 0.509였으며, 돌연변이 연산을 적용한 mutation_rate를 0.2로 한 반복 실행의 평균은 0.507로서 더 높은 적합도를 보이지는 않았다. 참고로 세대수의 평균을 비교하면, 돌연변이 연산이 없는 경우 평균 세대수는 9.85이고, 돌연변이 연산을 적용한 경우의 평균 세대수는 16.85로서, 돌연변이 연산이 없는 경우 솔루션을 찾는 세대수가 오히려 더 적게 나타났다.

돌연변이 연산으로 기대했던 가설은 더 좋은 적합도를 보이고, 더 짧은 세대수를 나타낼 것이라는 기대였다. 그러나 실험의 결과로는 꼭 그렇지는 않다고 볼 수 있다. 돌연변이 연산은 더 짧은 세대를 보장하지 않으며, 더 높은 적합도를 보장하지 않는다. 실험에서 돌연변이 연산을 위한 실행 시간 비용은 고려하지 않았으므로, 실행 시간 비용까지 고려한다면, 돌연변이 연산의 긍정적 효과를 오히려 반할 것으로 예상할 수 있다.

3. Threats to Validity

GA를 적용하는 NP 문제로서 TSP, Knapsack, Shubert 등의 어플리케이션을 들 수 있으며, 이들 모두를 프로그래밍하여 돌연변이에 대한 내용을 적용할 수 있다[12]. 그러나 본 실험은 TSP 문제를 해결하는 어플리케이션을 대상으로 하여 돌연변이 연산자를 적용하여, 본 연구의 RQ에 대한 결론을 정리하였다. 본 연구의 RQ는 돌연변이 연산과 솔루션의 비용 이익면에서의 상호관계에 대한 것이다. 즉 그림 4와 그림 5에서 보듯이, 세대수를 비용으로 하고, 솔루션에 대한 적합도를 이익으로 하는 비용 효과적 평가를 하며, 이는 이 두 가지 값의 상대적 관계를 분석한다. 또한 돌연변이는 국지적 솔루션에 급속도로 가까워지는 것을 흩어서 시야를 넓게 만드는 역할을 한다. 돌연변이 연산이 없다면, 이후 후손들은 첫 번째 조상 세대의 개체들의 범위에서 벗어날 수 없다. 즉, 처음 세대의 범위에서 벗어나기 위한 변형을 돌연변이 연산자가 맡아서 한다. 이 역할은 어떤 어플리케이션에서도 동일하게 적용되므로, TSP에 대한 평가 결과로 본 실험의 RQ에 대한 답을 유도하였다.

V. Conclusions

본 연구는 유전자 알고리즘이 갖는 돌연변이 연산에 대한 효과성 분석을 하였다. 지역적 최적 솔루션에 머무는 문제를 해결하기 위한 돌연변이 연산자는 프로그램 실행 면에서 또 다른 실행 비용을 발생하게 된다. 본 연구에서는 돌연변이 연산을 적용하는 경우와 적용하지 않는 경우의 솔루션을 찾는 세대수와 솔루션의 적합도를 분석하여, "돌연변이 연산을 적용하면 세대수는 짧아지고 적합도는 높아지는가? 즉 비용이익면에서 효과적일까?"에 대한 답을 얻고자 하였다. 유전자 알고리즘을 적용하는 대표적인 문제인 TSP를 파이썬으로 구현하였으며, 돌연변이 연산 비율, 즉 mutation_rate를 독립변수로 하는 실험을 수행하였다. 그 결과, 돌연변이 연산을 적용하지 않는 경우에 비하여 돌연변이 연산을 적용하는 경우가 더 높은 적합도 또는 짧은 세대수를 보장하지는 않음을 알 수 있었다. 즉, 돌연변이 연산으로 인한 추가 실행 시간 비용까지 고려한다면, 무조건적인 돌연변이 연산은 반드시 필요하지 않으며, 만일 요구되는 적합도의 수준이 매우 높으며, 시스템의 성능이 높은 경우로서 세대수가 많아지거나 돌연변이 연산 적용의 비용에 영향을 받지 않는다면 솔루션의 질적 향상, 즉 적합도를 높이는 방향으로 돌연변이 연산을 고려할 수 있다. 본 실험은 돌연변이 연산에 대한 기대 가설이 반드시 참이 아님을 보임으로써, 돌연변이 연산을 반드시 적용할 필요가 없음을 보였다.

ACKNOWLEDGEMENT

This work was supported by the Hyupsung University Research Grant of 2019.

REFERENCES

[1] Holland, J.H.. "*Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*," MIT Press: Cambridge, MA, USA, pp.89-120, 1992.

[2] M. Mitchell, "*An Introduction to Genetic Algorithms,*" MIT Press, pp.128-130, 1999

[3] H. Yoon, "Fitness-Orientated Mutation Operators in Genetic Algorithm," IJITEE, Vol. 9, No. 4, pp.1769-1772, Feb. 2020.

doi:10.35940/jijtee.D1692.029420

[4] Alan T Piszcz, Terence Soule, “A Survey of Mutation Techniques in Genetic Programming,” Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp. 951–952, Seattle, Washington, USA, July 2006. DOI: 10.1145/1143997.1144165

[5] E. Osaba, R. Carballedo, F. Diaz, E. Onieva, I. de la Iglesia, A. Perallos, "Crossover versus Mutation: A Comparative Analysis of the Evolutionary Strategy of Genetic Algorithms Applied to Combinatorial Optimization Problems", The Scientific World Journal, vol. 2014, Article ID 154676, 22 pages, 2014. DOI: 10.1155/2014/154676

[6] Hee-Su Kim and Sung-Bae Cho, "An efficient genetic algorithm with less fitness evaluation by clustering," Proceedings of the 2001 IEEE Congress on Evolutionary Computation, pp.887-894, Seoul, South Korea, May 2001. DOI: 10.1109/CEC.2001.934284.

[7] L. Wang, J. Zhang and H. Li, "An Improved Genetic Algorithm for TSP," Proceedings of 2007 International Conference on Machine Learning and Cybernetics, pp.925-928, Hong Kong, Aug. 2007. DOI: 10.1109/ICMLC.2007.4370274.

[8] Clinton Sheppard, “*Genetic Algorithms with Python*,” CreateSpace Independent Publishing Platform, pp.169-186, 2016

[9] Giancarlo Zaccone, “*Natural Computing with Python*,” bpb publications, pp.85-118, 2019

[10] J. Lu, N. Fang, D. Shao and C. Liu, "An Improved Immune-Genetic Algorithm for the Traveling Salesman Problem," Proceedings of 3rd International Conference on Natural Computation (ICNC 2007), pp. 297-301, Haikou, Aug. 2007. DOI: 10.1109/ICNC.2007.217.

[11] Y. Liu and J. Huang, "A Novel Genetic Algorithm and Its Application in TSP," Proceedings of 2008 IFIP International Conference on Network and Parallel Computing, pp. 263-266, Shanghai, Oct. 2008. DOI: 10.1109/NPC.2008.27

[12] B. H. Hasan and M. S. Mustafa, "Comparative Study of Mutation Operators on the Behavior of Genetic Algorithms Applied to Non-deterministic Polynomial (NP) Problems," Proceedings of 2011 Second International Conference on Intelligent Systems, Modelling and Simulation, pp. 7-12, Kuala Lumpur, Jan. 2011. DOI: 10.1109/ISMS.2011.11.

Authors

Hoijin Yoon received the B.S., M.S., and Ph.D. degrees in Computer Science from Ewha Womans University, Korea, in 1993, 1998, and 2004, respectively. She has been working at Hyupsung University since 2007, after a couple of years’ carrier in Ewha Womans University as a full-time lecturer. Currently, Hoijin Yoon is an associate professor of the Department of Computer Engineering at Hyupsung University. Her research work focuses specifically on Software Testing and Mutation Analysis.