Collection @ chosun

creative commons

COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

[UCI]I804:24011-200000238483

Collection @ chosun

2009년 8월
박사학위논문

최적의 TSP 해결을 위한 유전자 알고리즘의 새로운 집단 초기화 및 순차변환 기법 제안과 구현

조선대학교 대학원

전산통계학과

강 래 구

Collection @ chosun

최적의 TSP 해결을 위한 유전자 알고리즘의 새로운 집단 초기화 및 순차변환 기법 제안과 구현

The Proposal and Implementation of New Population Initialization and Sequential Transformation Method for Genetic Algorithm to Find Optimal TSP

2009년 8월 25일

조선대학교 대학원

전산통계학과

강 래 구

Collection @ chosun

최적의 TSP 해결을 위한 유전자 알고리즘의 새로운 집단 초기화 및 순차변환 기법 제안과 구현

지도교수 정 채 영

이 논문을 이학박사학위 신청논문으로 제출함

2009년 4월

조선대학교 대학원

전산통계학과

강 래 구

Collection @ chosun

강래구의 박사학위논문을 인준함

위 원 장 조선대학교 교수 이 웅 기 (인)

위 원 초당대학교 교수 정 남 채 (인)

위 원 조선대학교 교수 배 상 현 (인)

위 원 조선대학교 교수 장 인 홍 (인)

위 원 조선대학교 교수 정 채 영 (인)

2009년 6월

조선대학교 대학원

Collection @ chosun

목 차

목 차 ····· i
표 목 차 ····· iv
그림목차 ····· v
Abstract ····· viii

Ⅰ. 서 론 ····· 1
A. 연구 배경 ····· 1
B. 연구 방법 및 목적 ····· 3

Ⅱ. 관련 분야 연구 ····· 6
A. 순회판매원문제(Traveling Salesman Problem : TSP) ····· 6
B. 유전자 알고리즘(Genetic Algorithm : GA) ····· 9
1. 집단 초기화 ····· 13
2. 적합도 함수 ····· 16
3. 스키마 ····· 17
4. 엘리트 전략 ····· 20
C. 유전자 알고리즘 대표 연산자들 ····· 21
1. 선택(Selection) 연산자 ····· 22
2. 교배(Crossover) 연산자 ····· 28
3. 돌연변이(Mutation) 연산자 ····· 40

Collection @ chosun

Ⅲ. 새로운 알고리즘 제안 ····· 44

A. 혼합 초기화법 ····· 44

B. 순차변환 기법 ····· 46

C. 제안한 알고리즘 구현 ····· 47

Ⅳ. 구현 및 평가 ····· 51

A. 50개 도시 순회 실험 ····· 51

1. 토너먼트(Tournament)-PMX-Inversion ····· 51

2. 토너먼트(Tournament)-PMX-Swap ····· 54

3. 토너먼트(Tournament)-CX-Inversion ····· 56

4. 토너먼트(Tournament)-CX-Swap ····· 58

5. 토너먼트(Tournament)-OX-Inversion ····· 60

6. 토너먼트(Tournament)-OX-Swap ····· 62

B. 100개 도시 순회 실험 ····· 64

1. 룰렛 휠(Roulette-Wheel)-PMX-Inversion ····· 64

2. 룰렛 휠(Roulette-Wheel)-PMX-Swap ····· 66

3. 룰렛 휠(Roulette-Wheel)-CX-Inversion ····· 68

4. 룰렛 휠(Roulette-Wheel)-CX-Swap ····· 70

5. 룰렛 휠(Roulette-Wheel)-OX-Inversion ····· 72

6. 룰렛 휠(Roulette-Wheel)-OX-Swap ····· 74

C. 실험에 대한 평가 ····· 76

Ⅴ. 결론 및 향후 과제 ····· 82

Collection @ chosun

참고문헌 ····· 84

부록 1 50개 도시 거리 정보 ····· 87

부록 2 100개 도시 거리 정보 ····· 90

dCollection @ chosun

표 목 차

【표 3-1】 실험에 사용한 연산자 ····· 49
【표 4-1】 토너먼트 연산자 실험 결과 비교 1 ····· 52
【표 4-2】 토너먼트 연산자 실험 결과 비교 2 ····· 55
【표 4-3】 토너먼트 연산자 실험 결과 비교 3 ····· 57
【표 4-4】 토너먼트 연산자 실험 결과 비교 4 ····· 59
【표 4-5】 토너먼트 연산자 실험 결과 비교 5 ····· 61
【표 4-6】 토너먼트 연산자 실험 결과 비교 6 ····· 63
【표 4-7】 룰렛 휠 연산자 실험 결과 비교 1 ····· 65
【표 4-8】 룰렛 휠 연산자 실험 결과 비교 2 ····· 67
【표 4-9】 룰렛 휠 연산자 실험 결과 비교 3 ····· 69
【표 4-10】 룰렛 휠 연산자 실험 결과 비교 4 ····· 71
【표 4-11】 룰렛 휠 연산자 실험 결과 비교 5 ····· 73
【표 4-12】 룰렛 휠 연산자 실험 결과 비교 6 ····· 75
【표 4-13】 50개 도시 실험 결과 ····· 78
【표 4-14】 100개 도시 실험 결과 ····· 79

Collection @ chosun

그 림 목 차

【그림 1-1】 진화 프로그램의 모형 ····· 1
【그림 1-2】 유전자 알고리즘 흐름도 ····· 4
【그림 2-1】 4개 도시의 순회경로 ····· 7
【그림 2-2】 도시 수에 따른 순회경로 ····· 7
【그림 2-3】 최적화 문제 표현식 ····· 8
【그림 2-4】 유전자 알고리즘 구조 ····· 11
【그림 2-5】 유전자 알고리즘 연산 구조 ····· 11
【그림 2-6】 목적함수 ····· 13
【그림 2-7】 무작위 초기화법에 의한 초기 집단 생성 과정 ····· 14
【그림 2-8】 균등 분할하여 얻어진 초기화 점 ····· 16
【그림 2-9】 스키마 정의길이 ····· 18
【그림 2-10】 룰렛 휠 공간 배정 ····· 23
【그림 2-11】 룰렛 휠 구현 ····· 23
【그림 2-12】 순위 기반 선택 ····· 25
【그림 2-13】 확률균등 선택 ····· 26
【그림 2-14】 공유 함수 예 ····· 27
【그림 2-15】 일점 교배를 통한 자식해 생성 ····· 29
【그림 2-16】 생물학적 측면의 일점 교배 ····· 29
【그림 2-17】 다점 교배를 통한 자식해 생성 ····· 30
【그림 2-18】 균등 교배의 예 ····· 31
【그림 2-19】 싸이클 교배의 예 ····· 32
【그림 2-20】 싸이클 교배의 진행 과정 ····· 32

Collection @ chosun

【그림 2-21】 싸이클 교배의 진행 과정 ····· 33
【그림 2-22】 순서 교배의 예 ····· 33
【그림 2-23】 PMX 예 ····· 34
【그림 2-24】 산술적 교배의 예 ····· 35
【그림 2-25】 Edge Recombination(ER) 교배의 예 ····· 36
【그림 2-26】 Cohoon과 Paris의 다차원 교배 예 ····· 37
【그림 2-27】 불록 균등 교배 예 ····· 38
【그림 2-28】 직선의 제약을 없앤 2차원 지오그래픽 교배 예 ····· 38
【그림 2-29】 전형적 돌연변이 예 ····· 41
【그림 2-30】 역치 돌연변이 예 ····· 42
【그림 2-31】 교체 돌연변이 예 ····· 43
【그림 2-32】 삽입 돌연변이 예 ····· 43
【그림 3-1】 혼합 초기화법 ····· 45
【그림 3-2】 순차변환 진행 과정 ····· 46
【그림 3-3】 실험에 사용한 UI ····· 48
【그림 3-4】 어플리케이션 소스 코드(예) ····· 50
【그림 4-1】 토너먼트 연산자 실험 결과 1 ····· 52
【그림 4-2】 제안한 알고리즘을 적용한 UI 실행 결과 1 (50개 도시) ····· 53
【그림 4-3】 토너먼트 연산자 실험 결과 2 ····· 54
【그림 4-4】 제안한 알고리즘을 적용한 UI 실행 결과 2 (50개 도시) ····· 55
【그림 4-5】 토너먼트 연산자 실험 결과 3 ····· 56
【그림 4-6】 제안한 알고리즘을 적용한 UI 실행 결과 3 (50개 도시) ····· 57
【그림 4-7】 토너먼트 연산자 실험 결과 4 ····· 58
【그림 4-8】 제안한 알고리즘을 적용한 UI 실행 결과 4 (50개 도시) ····· 59

Collection @ chosun

【그림 4-9】 토너먼트 연산자 실험 결과 5 ····· 60
【그림 4-10】 제안한 알고리즘을 적용한 UI 실행 결과 5 (50개 도시) ····· 61
【그림 4-11】 토너먼트 연산자 실험 결과 6 ····· 62
【그림 4-12】 제안한 알고리즘을 적용한 UI 실행 결과 6 (50개 도시) ····· 63
【그림 4-13】 룰렛 휠 연산자 실험 결과 1 ····· 64
【그림 4-14】 제안한 알고리즘을 적용한 UI 실행 결과 1 (100개 도시) ····· 65
【그림 4-15】 룰렛 휠 연산자 실험 결과 2 ····· 66
【그림 4-16】 제안한 알고리즘을 적용한 UI 실행 결과 2 (100개 도시) ····· 67
【그림 4-17】 룰렛 휠 연산자 실험 결과 3 ····· 68
【그림 4-18】 제안한 알고리즘을 적용한 UI 실행 결과 3 (100개 도시) ····· 69
【그림 4-19】 룰렛 휠 연산자 실험 결과 4 ····· 70
【그림 4-20】 제안한 알고리즘을 적용한 UI 실행 결과 4 (100개 도시) ····· 71
【그림 4-21】 룰렛 휠 연산자 실험 결과 5 ····· 72
【그림 4-22】 제안한 알고리즘을 적용한 UI 실행 결과 5 (100개 도시) ····· 73
【그림 4-23】 룰렛 휠 연산자 실험 결과 6 ····· 74
【그림 4-24】 제안한 알고리즘을 적용한 UI 실행 결과 6 (100개 도시) ····· 75
【그림 4-25】 룰렛 휠-PMX-Inversion 연산자 조합 (50개 도시) ····· 80
【그림 4-26】 토너먼트-PMX-Inversion 연산자 조합 (100개 도시) ····· 80

Collection @ chosun

ABSTRACT

The Proposal and Implementation of New Population Initialization and Sequential Transformation Method for Genetic Algorithm to Find Optimal TSP

Rae-Goo Kang
Advisor : Prof. Jung Chai-yeoung Ph.D.
Department of Computer Science & Statistics,
Graduate School of Chosun University

As information and telecommunication (IT) becomes ever-developed, the shortest distance setting in GIS, mass transit network design, network optimization and logistic system has begun to have firsthand effects on cost of distance and time in more cases of our daily life than before.

Most of problems with shortest distance setting are based on TSP (Traveling Salesman Problem).

Given certain number of cities and certain distance between cities, TSP refers to a problem of finding shortest distance across all turnaround routes from a city, starting point, via all other cities visited once only to the original city. However, as the number of cities gets increasing, the load of calculation to find optimal solution tends to increase exponentially, so such calculation has been typically classified as an NP-hard problem.

In order to resolve this problem, the Genetic Algorithm proposed by

Collection @ chosun

John Holland has been used as the most representative algorithm to calculate the optimal solution of TSP.

The Genetic Algorithm, which is modeled conceptually after the survival of the fittest dominating in the natural world, has an advantage of investigating in parallel the domain of combinable solutions for a given problem.

This study proposed two major ways to more effectively find out an optimal TSP solution using Genetic Algorithm: One refers to a new initialization method that uses a combination of random initialization method and induced initialization method, which belong to population initialization method, one of Genetic Algorithm procedures.

The other refers to algorithmic implementation and experiment using sequential transformation method for applying an alternative operator, so as to enhance probability of selecting a combination of dominant factors and thereby find out an optimal TSP solution.

In order to verify potential performance of algorithm proposed herein, this study conducted experiments with 50 cities and 100 cities set as variables. As a result, 50 cities showed 3.86% higher performance on the average, and 100 cities showed 6.97% higher performance on the average.

That is, it was found that group initialization method using information on known distance among cities and a method for higher probability of selecting a combination of dominant factors by rearranging subjects could be effective in improving performance to find out an optimal TSP solution.

And it was found that the larger number of cities was directly associated with the higher effectiveness in improving performance.

As demonstrated by findings hereof, it is expected that the algorithm

Collection @ chosun

proposed in a field based on TSP will be useful in various ways and will be helpful to find out an optimal solution in quicker and more correct ways.

dCollection @ chosun

Ⅰ. 서 론

A. 연구 배경

정보통신의 발달과 함께 지리정보시스템(GIS), 대중교통망 설계, 네트워크 최적화, 택배시스템 등 최단거리 설정이 거리비용 및 시간비용에 직접적인 영향을 미치는 경우가 일상생활에서 자주 나타나기 시작했다. 이러한 대부분의 최단거리 설정문제는 TSP(Traveling Salesman Problem : 순회판매원문제)에 근간을 두고 있는 경우가 대부분이다.

지난 수십 년 간 TSP와 같이 진화를 기초로 하는 문제 해결 시스템에 대한 관심이 증가하면서 다양한 연구가 진행되어왔고 이러한 진화기반의 시스템에 대해 진화 프로그램이라는 공통용어를 사용하기 시작하였다. 그림 1-1은 진화 프로그램의 모형을 나타낸다.[1]

```
Procedure evolution program
Begin
   a ← Ø
   initialize T(a)
   evaluate T(a)
   while (not termination-condition) do
   begin
      a ← a + 1
      select T(a) from T(a-1)
      alter T(a)
      evaluate T(a)
   end
end
```

【그림 1-1】 진화 프로그램의 모형

Collection @ chosun

TSP란 N개의 도시와 각 도시 사이의 거리가 주어질 때, 임의의 한 도시에서 출발하여 모든 도시를 단 한 번만 방문하고 원래의 출발지로 되돌아오는 경로 중 최단거리를 찾는 문제이다.

경로 탐색 범위는 모든 여행의 집합 $\{T_1, T_2, ..., T_{n!}\}$이 되고 총 경로는 $N!$이며 이 중 순회 거리가 가장 짧은 것이 해가 된다. 만약 4개의 도시를 방문할 경우에는 4!, 즉 24가지 경로에 대해 거리 검사를 해야 하며, 도시 수가 30개일 경우에는 무려 30!에 달하는 경로를 검사하여야만 정확한 최단거리를 찾을 수 있다.

일반적으로 N개의 도시로 이루어진 TSP에서 모든 가능한 경로의 수는 $N!$로 수학적 모델을 세우기는 용이하지만 N이 증가함에 따라 해를 찾기 위한 계산량이 기하급수적으로 증가하여 짧은 시간 내에 이 문제를 해결하기는 실질적으로 불가능하게 된다. 때문에 TSP는 NP(Nondeterministic Polynomial)-hard 문제로도 분류되고 있다.[2][3]

이러한 문제를 해결하기 위해 다양한 알고리즘을 적용한 연구가 진행되었고, 시뮬레이티드 어닐링 (Simulated Annealing), 신경망(Neural Network), 그리고 유전자 알고리즘(Genetic Algorithm) 등이 그 대표적인 방법으로 알려져 있다. 시뮬레이티드 어닐링은 S. Kirkpatrick 등에 의해 처음 제안된 방법으로 조합 최적화 문제와 관련하여 소개되었으며 매개변수가 되는 '온도'를 조금씩 낮추어 가면서 주어진 문제의 최적해를 찾아가는 방법이다. 신경망은 Rosenblatt에 의해 제안된 방법으로 생물학적 뇌신경의 정보처리 메커니즘을 모방하여 최적해를 찾아가는 방법이다.[4]

유전자 알고리즘(Genetic Algorithm : GA)은 John Holland에 의해 제안된 알고리즘으로서 TSP 최적의 해를 계산하기 위한 가장 대표적인 알고리즘으로 이용되고 있다. 유전자 알고리즘은 주어진 집단이 다음 세대의 집단으로 진화하는 과정에서 다양한 알고리즘을 적용하여 자연계의 적자생존의 개념을 모델링한 것

Collection @ chosun

으로 주어진 문제에 대한 조합 가능한 해의 영역을 병렬적으로 조사한다는 장점을 갖고 있다.[5][6]

B. 연구 방법 및 목적

TSP는 구현하기가 용이하여 다양한 분야에서 많이 응용되고 있는데, 특히 택배 차량의 최단거리 계산이나 최소비용 계산, GPS, 공장자동화 등과 같은 최적화 문제에 많이 이용되고 있다.

이러한 TSP의 최적해를 빠르고 정확하게 계산하기 위해 다양한 알고리즘이 연구되어왔고 John Holland에 의해 유전자 알고리즘이 처음 제안되었다. 이후 유전자 알고리즘은 TSP 최적해를 구하기 위한 가장 대표적인 알고리즘으로 이용되고 있다.

유전자 알고리즘은 자연의 생물들이 다음 세대로 진화해 가는 과정에서 주변 환경에 대한 적응력이 높은 개체가 살아남아 또 다른 우수한 유전 형질 개체와 교배를 통해 더욱 우수한 개체를 생산하고, 돌연변이를 통해 급작스런 환경 변화에 대처하는 점을 응용한 일종의 탐색 알고리즘이다.[7]

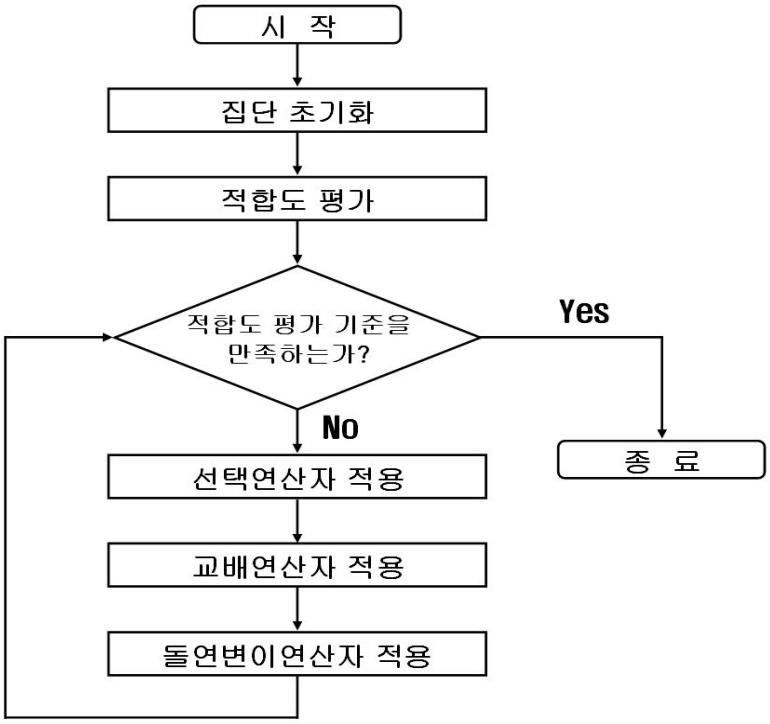
유전자 알고리즘은 특히 다른 인공지능 기법들과 통합되어 많이 적용되고 있는데, 이는 유전자 알고리즘이 인공지능 모형 구축에 있어서 최적화 필요를 만족시키는데 효과적이기 때문이다. 유전자 알고리즘은 초기화(Initialization), 선택(Selection), 교배(Crossover), 그리고 돌연변이(Mutation)와 같은 절차를 통해 탐색을 수행하게 된다.[8][9][10]

그림 1-2는 유전자 알고리즘의 실행 흐름을 나타내고 있다. 집단 초기화 단계에서 염색체(Chromosome)라고 불리는 개체를 미리 결정해 놓은 개체 수 만큼 임의로 생성하여 적합도 함수(Fitness Function)를 사용해서 그 적합도를 평가한다. 일정한 평가 기준을 만족하지 못할 경우 선택, 교배, 돌연변이 연산자를 각각

dCollection @ chosun

적용하여 평가 기준을 만족할 때까지 계속해서 수행하게 된다.

본 논문에서는 유전자 알고리즘 수행 과정의 하나인 집단 초기화 방법 중 무작위 초기화법과 유도된 초기화법을 동시에 이용하는 새로운 혼합 초기화법을 유추해 냈고, 선택연산자를 적용할 때 순차변환 기법을 통해 우성 인자들끼리 선택될 수 있는 확률을 높임으로써 최적의 TSP 해를 구할 수 있도록 구현하고 실험하였다.



【그림 1-2】 유전자 알고리즘 흐름도

본 논문의 제 Ⅰ장에서는 논문의 목적 및 배경에 대해 기술하였고, 제 Ⅱ장에서는 TSP와 유전자 알고리즘을 전반적으로 설명을 하였다. 제 Ⅲ장에서는 본 논문에서 제안한 알고리즘을 설명하였고, 제 Ⅳ장에서는 제안한 알고리즘을 직접

Collection @ chosun

구현하고 실험한 결과에 대해 평가하였다. 마지막으로 제 V장에서는 본 논문의 종합적인 결론과 향후 연구과제에 대하여 기술하였다.

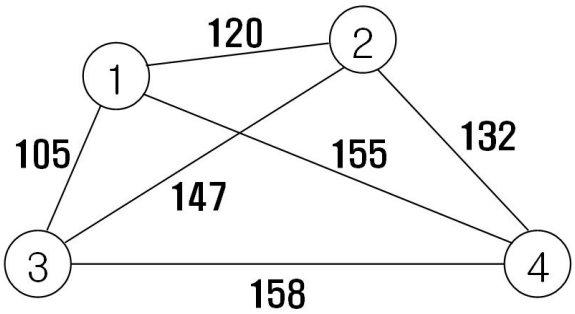
Collection @ chosun

Ⅱ. 관련 분야 연구

A. 순회판매원문제(Traveling Salesman Problem : TSP)

TSP문제는 비교적 오래된 문제이다. 이 문제는 “기사들의 여행문제(kinghts' tour problem)”에 관심이 있던 Euler에 의해 1759년에 처음으로 알려졌다. 이 문제에 대한 정확한 답은 기사가 체스판 64개 스퀘어를 단 한 번씩 모두 지나가야 하는 것이었다.

“순회판매원”이라는 용어는 1932년 한 베테랑 순회판매원에 의해 독일어로 쓰여진 “순회판매원 : 그가 수수료를 받고 사업에서 성공하기 위해 무엇을 어떻게 해야 하는가”라는 책에서 처음으로 사용되었고, 그 후 수년 동안 TSP는 NP-hard문제로 분류되어 지금까지 다양한 방법으로 연구되어오고 있다.[11][12]



T_1 = 1-2-3-4-1 Dis_1 = 580

T_2 = 1-2-4-3-1 Dis_2 = 515

T_3 = 1-3-2-4-1 Dis_3 = 539

⋮

dCollection @ chosun

T_{22} = 4-2-3-1-4 Dis_{22} = 539

T_{23} = 4-3-1-2-4 Dis_{23} = 515

T_{24} = 4-3-2-1-4 Dis_{24} = 580

【그림 2-1】 4개 도시의 순회경로

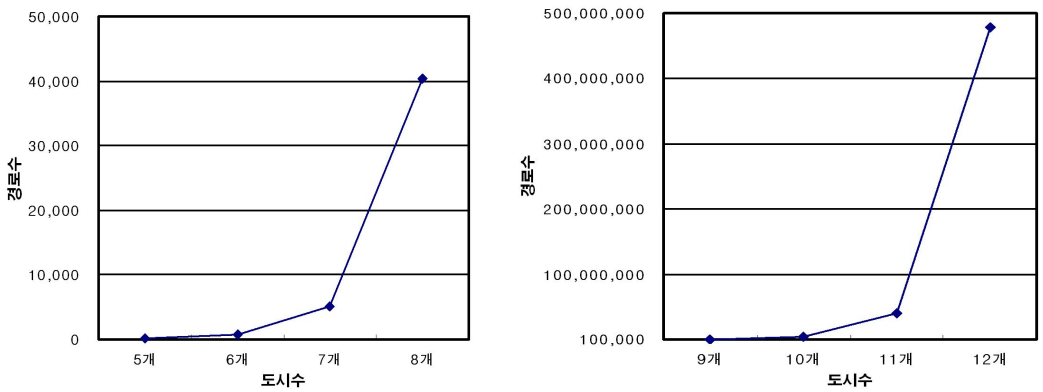
그림 2-1은 4개의 도시가 주어졌을 때 구할 수 있는 총 순회경로를 나타내는 거리 계산이다. ($4! = T_1, \ldots, T_{24}$)

4개의 도시가 주어졌을 때는 총 24가지 경로만을 계산하면 최소거리를 찾을 수 있지만 도시의 수가 증가할수록 이동 경로의 경우의 수는 기하급수적으로 증가하게 된다.

$5! = T_1, \ldots, T_{120}$ $6! = T_1, \ldots, T_{720}$ $7! = T_1, \ldots, T_{5040}$

$8! = T_1, \ldots, T_{40320}$ $9! = T_1, \ldots, T_{362880}$ $10! = T_1, \ldots, T_{3628800}$

$11! = T_1, \ldots, T_{39916800}$ $12! = T_1, \ldots, T_{479001600}$



【그림 2-2】 도시 수에 따른 순회경로

그림 2-2에 나타나 있듯 도시 수가 하나씩 증가할 때마다 순회경로 수가 기하

dCollection @ chosun

급수적으로 증가한다는 걸 알 수 있다. 4개의 도시가 주어졌을 경우엔 24가지 방법에 불과하던 경우의 수가 12개의 도시로 증가했을 경우 무려 4억8천여 가지의 경우의 수로 급증하는 문제가 발생한다. 따라서 TSP는 그림 2-3과 같은 식으로 표현된 최적화 문제로 나타낼 수 있다.

$$Minimize \quad Z = \sum_{j}^{n}\sum_{j}^{n} T_{ij}\, d_{ij} \qquad \text{식 2-1}$$

$$Subject \quad to \quad \sum_{i}^{n} T_{ij} = 1 \quad , \quad \sum_{j}^{n} T_{ij} = 1 \qquad \text{식 2-2}$$

$$T_{ij} = \begin{cases} 1 : If\ Salesman\ Travels\ From\ City\ i\ to\ City\ j \\ 0 : Otherwise \end{cases} \qquad \text{식 2-3}$$

$$T = \{(i_1, i_2), (i_2, i_3), (i_3, i_4), \ \ldots\ldots\ , (i_{N-1}, i_N), (i_N, i_1)\} \qquad \text{식 2-4}$$

【그림 2-3】 최적화 문제 표현식

경로 T는 각 도시에 하나씩 짝을 이루어 전체가 하나의 원을 형성하게 된다. 그리고 경로는 식 2-4처럼 일련의 도시 쌍으로 나타낼 수 있고 각 요소 (i, j)는 경로의 node를 타나낸다.

TSP의 중요성은 단지 세일즈맨의 여행 거리를 최소화하는 것뿐만 아니라, TSP 요소를 내재하고 있는 응용문제들이 많이 있다는 사실이다. 예를 들어, 17,000개의 도시를 갖는 회로 기판 드릴링 문제나 14,000개의 도시를 갖는 X선 결정학 문제, 그리고 120만개의 도시를 갖는 초고밀도집적회로(VLSI) 제작의 경우 등이 그것이다. 이런 경우 최적의 답을 구하기 위해 수백만 달러나 하는 컴퓨터에서 몇 시간씩 작업을 진행하는 것은 전혀 효율적이지 못하기 때문에 개인용

Collection @ chosun

컴퓨터를 이용하여 수초 내에 몇 퍼센트의 오차 내로 답을 구하기 위한 다양한 연구가 꾸준히 진행되어오고 있다.[12][13]

B. 유전자 알고리즘(Genetic Algorithm : GA)

다윈(Charles Darwin)은 1859년 발행된『The origin of species』에서 종이 어떻게 변할 수 있는지를 설명하는 구조로 자연도태(Natural Selection)를 제시하였다.

다윈은 자연에 잘 적응하는 개체는 생존하고 그렇지 못하면 도태하는 적자생존의 원리로 개체군(Population) 중에서 환경에 대한 적합도가 높은 개체가 살아남을 확률이 높아서 재생할 수 있게 되며, 이때 교배와 돌연변이로 다음 세대의 개체군을 형성하게 된다고 했다.[14]

자연계의 생물들은 다음 세대로 진화해 가는 과정에서 주변 자연 환경에 대한 적응력이 높은 개체가 높은 확률로 살아남아 자신의 유전 형질을 복제·선택하고 교배를 통해 더욱 우수한 유전자를 보유한 개체를 생산해 내며 돌연변이를 일으킴으로써 급작스런 환경 변화에 유연하게 대처한다.

John Holland에 의해 처음 제안된 유전자 알고리즘은 진화 알고리즘의 일종으로 이러한 진화과정에서 나타나는 자연도태와 유전학에 개념을 두고 최적해를 찾기 위한 일종의 탐색 알고리즘이다.[7]

유전자 알고리즘이란 자연계에 있어서 생물의 유전과 진화의 메카니즘을 공학적으로 모델화한 것으로, 생물이 갖는 환경에서의 적응능력을 취급한 것이다. 1975년에 John Holland가 저서『Adaptation on Natural and Artificial Systems』에 처음 소개한 자연도태의 원리를 기초로 한 최적화(Optimization) 방법이다. 유전자 알고리즘은 탐색(Search), 최적화 및 기계학습(Machine Learning)을 위한 도구로도 많이 사용한다.

생물은 세포로 구성되고 세포에는 핵이 있으며 그 핵 중에는 염색체 (Chromo

dCollection @ chosun

some) 라는 것이 있다. 사람의 체세포에는 46개의 염색체가 있고, 염색체는 주로 DNA로 구성되어 있다. 이 DNA는 4종류의 염기화학물질로 구성되어 있는데, 이 염기배열에 따라 DNA 정보가 구성된다. DNA는 2중 나선구조로 되어 있으며, 이들이 복잡하게 겹쳐져서 염색체를 구성하고 있다. 유전자(Gene)란 유전정보를 담당하는 DNA를 말하고 특정의 유전자는 염색체의 특정 위치에 존재한다. 결국 유전정보는 염색체상에서의 위치(유전자 위치)와 염기의 배열에 의해 표현되는 것이다.

부모의 유전자로 부터 생물로서의 정보 전달이 행해지면 다음 세대에는 각 개체 중에서도 보다 우수한 즉, 환경에 적응도가 높은 개체의 유전 정보가 우선적으로 전해진다. 적응도가 낮은 개체는 수명이 짧고, 증식할 수 없기 때문이다. 따라서 적응도가 낮은 종족은 자연 도태되어 간다. 이러한 원리에 기초하여 세대를 거듭해 가면 차례로 환경에 적응도가 높은 개체가 많아진다. 이것이 유전(Genetics)과 진화(Evolution)의 기본적인 원리이다.

유전자 알고리즘은 풀고자 하는 문제에 대한 가능한 해들을 정해진 형태의 자료구조로 표현한 다음, 이들을 점차적으로 변형함으로써 점점 더 좋은 해들로 생성하게 된다. 각각의 가능한 해를 하나의 유기체(Organism) 또는 개체(Individual)로 보며 이들의 집합을 개체군(Population)이라 한다. 하나의 개체는 보통 한 개 또는 여러 개의 염색체로 구성되며 염색체를 변형하는 연산자들을 유전연산자(Genetic Operator)라 한다. 기본적인 연산자로는 선택(Selection), 교배(Crossover), 돌연변이(Mutation)가 있다.

선택연산자는 집단 중에서 적응도의 분포에 따라서 다음의 단계로 교배를 행할 개체의 생존 분포를 결정한다. 적응도의 분포에 기초하고 있기 때문에 적응도가 높은 개체일수록 보다 많은 자손을 남기기 쉽게 된다. 교배 연산자는 2개의 염색체 사이에서 유전자를 바꾸어 넣어 새로운 개체를 발생시킨다. 돌연변이 연산자는 유전자의 어떤 부분의 값을 강제적으로 변화시킨다.

Collection @ chosun

n 개의 초기 염색체 생성 ;

$repeat$ {

$for\ i = 1\ to\ k$ {

두 염색체 p_1, p_2 선택 ;

$offspring_i = crossover(p_1, p_2);$

$offspring_i = mutation(offspring_i);$

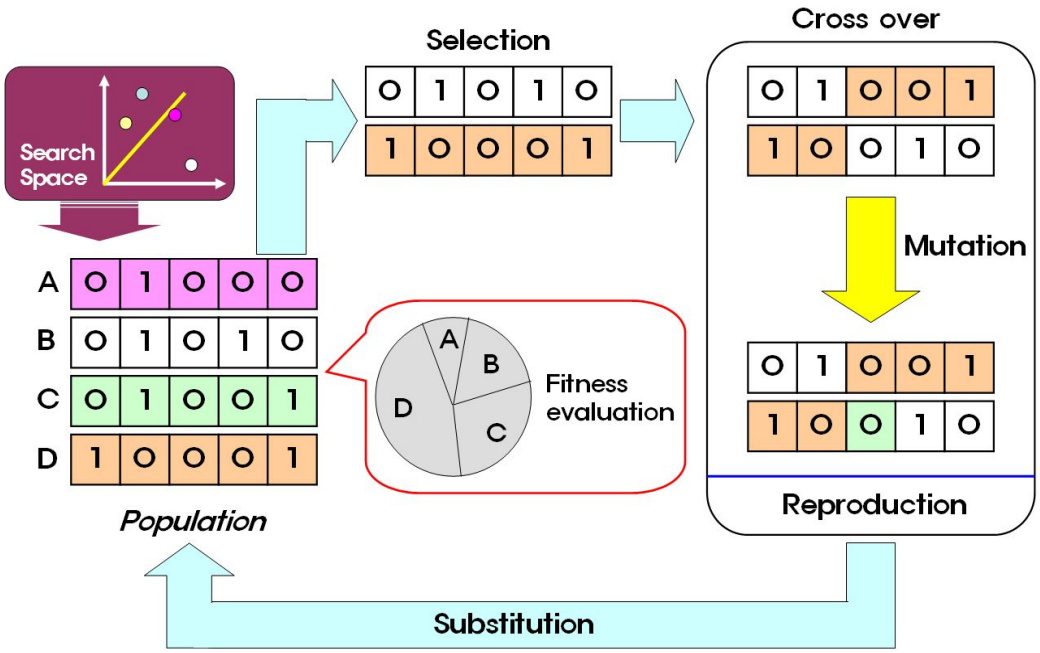
}

$offspring_1$, ... , $offspring_k$ 를 k 개의 염색체와 대치 ;

}$until$ (정지조건 만족);

남은 해 중 최상의 염색체를 $return$;

【그림 2-4】 유전자 알고리즘 구조



【그림 2-5】 유전자 알고리즘 연산 구조

Collection @ chosun

그림 2-4와 그림 2-5는 유전자 알고리즘의 구조를 나타내고 있다. 유전자 알고리즘은 자연진화를 모방하고 있다는 사실 외에도 기존의 탐색 알고리즘과 구분되는 몇 가지 특징들을 가지는데 이는 다음과 같다.[8][15]

○ 문제의 파라미터를 직접 이용하기 보다는 이를 적절히 기호로 표현하여 이용한다.
○ 하나의 해를 다루기보다는 집단을 취급한다.
○ 이득함수(Payoff Function) 또는 목적함수(Objective Function) 값만을 필요로 하고, 연속성(Continuity), 미분가능성(Differentiability), 단봉성(Unimodality) 등과 같은 다른 보조 정보를 요구하지 않는다.
○ 유전자 연산자들을 사용한다.
○ 얻어진 결과는 결정적이기 보다는 확률적이다.

이 외에도 유전자 알고리즘은 매우 복잡하고 규모가 큰 탐색공간에서도 원만히 동작하며, 목적함수를 선택하는 데 있어서도 매우 유연하다. 다시 말하면, 그림 2-5와 같이 다봉성의 특징을 가지는 탐색환경에서도 전역해를 얻을 수 있고, 구배에 기초한 기존의 방법들이 도함수를 쉽게 얻을 수 있는 “제곱오차의 적분(ISE)”을 목적함수로 취하지만, 유전자 알고리즘은 “절대오차의 적분(IAE)”, “시간곱 절대오차의 적분(ITAE)”등과 같이 어떤 형태의 함수도 목적함수로 쉽게 이용할 수 있다.

이러한 특징 때문에 유전자 알고리즘은 TSP를 비롯하여 함수의 최적화, 자원분배, 스케쥴링, 운송문제, 시스템제어 및 식별, 신경망 등 다양한 분야에서 폭넓게 응용되고 있다.[12][16]

Collection @ chosun

$$ISE = \int_0^{\infty} e^2(t)dt \qquad \text{식 2-5}$$

$$IAE = \int_0^{\infty} |e(t)|dt \qquad \text{식 2-6}$$

$$ITAE = \int_0^{\infty} t|e(t)|dt \qquad \text{식 2-7}$$

【그림 2-6】 목적함수

1. 집단 초기화

탐색공간의 점들을 표현하는 방법이 결정되면 다음은 연산자를 통해 모의진화가 수행될 초기 개체집단을 생성하는 절차가 요구된다. 개체들이 진화해 가는 도중에 집단은 총체적 기억장소로서의 역할을 담당하게 되고, 많은 세대를 거치는 동안 좋은 해(우성인자)의 유전자를 보유하게 된다.

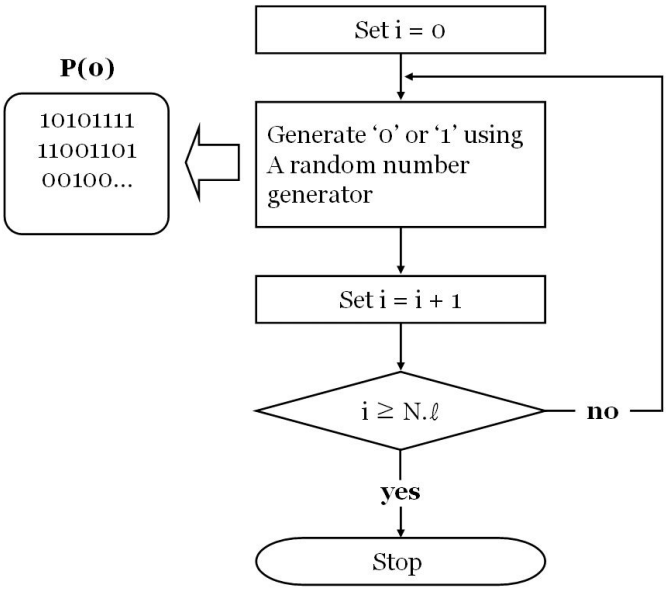
$$P(k) = \{s_1(k), s_2(k), \ldots\ldots, s_i(k), \ldots\ldots, s_n(k)\} \qquad \text{식 2-8}$$

식 2-8에서처럼 k세대 (k번째 반복횟수)에서의 집단 $P(k)$는 염색체로 특정 지워지는 N개 개체들의 집합으로 정의된다. 여기서 $s_i(k)$는 i번째의 염색체로서 탐색공간 상의 한 점을 나타내며, $N(>1)$은 집단크기(Population Size)이다.

계절의 변화와 세대에 따라 집단의 크기를 달리하는 자연계의 일부 곤충들과 같이 가변집단을 채택하고 있는 유전알고리즘도 있지만 대부분의 경우 집단크기는 세대 변화와는 무관하게 항상 고정된 것으로 간주된다. 또한 일반성을 유지하여 집단 내의 염색체들의 길이는 서로 같으며, 세대 변화와는 상관없이 일정한 것으로 간주된다.

Collection @ chosun

초기 집단 $P(0)$는 연구용 유전자 알고리즘이냐 아니면 응용을 위한 유전자 알고리즘이냐에 따라 그 생성 방법이 판이하게 다르다. 자주 이용되는 방법으로는 두 가지가 있다. 하나는 무작위 초기화(Random Initialization)법이고 다른 하나는 사전 지식 또는 경험을 바탕으로 하는 유도된 초기화(Directed Initialization)법이다. 무작위 초기화법은 단순히 난수발생기에서 생성되는 $N\ell$(= 집단크기 × 염색체 길이)개의 2진 정수로 염색체를 초기화하게 된다. 그림 2-7은 무작위 초기화법을 이용한 초기 집단 생성과정을 보여주고 있다. 이 방법은 해에 대한 사전 지식이 전혀 없거나 임의로 발생된 집단으로부터 전역해를 탐색해가는 유전자 알고리즘의 성능을 평가할 때 주로 이용된다.



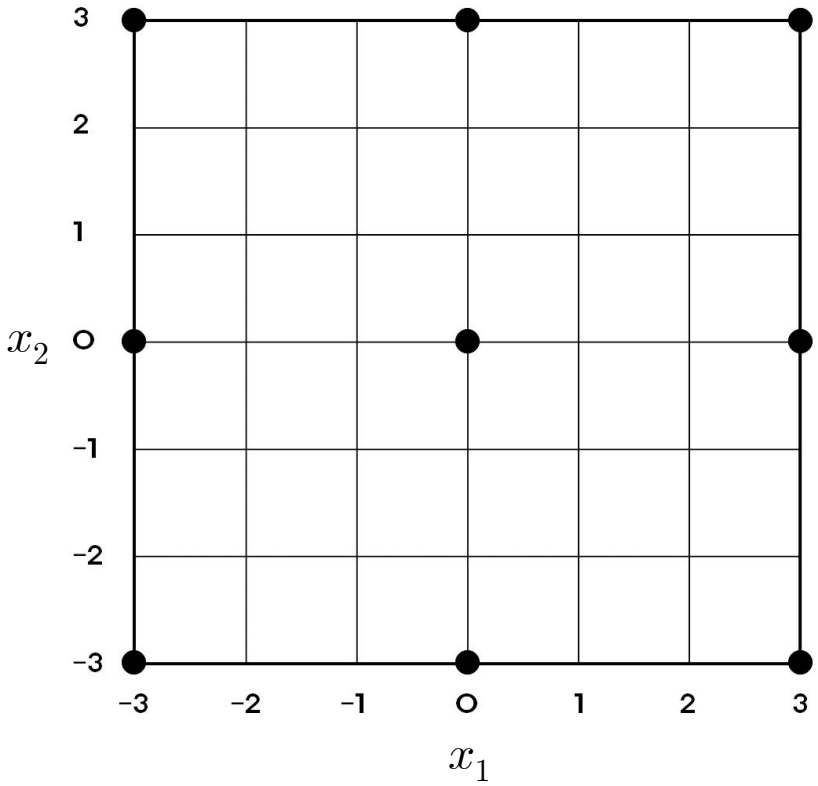
【그림 2-7】 무작위 초기화법에 의한 초기 집단 생성 과정

유도된 초기화법은 다른 간단한 알고리즘을 돌려 얻게 되는 근사 결과치를 이용하거나 아니면 직관과 경험을 기초로 한다. 해에 대한 사전지식이 있을 경우는

dCollection @ chosun

이 방법을 이용하는 것이 더 바람직하다.

예를 들어 어떤 알고리즘을 사용한 결과, 2차원 최적화 문제의 해가 구간 $-3 \le x_1, x_2 \le 3$ 사이에 존재함을 알았다고 하자. 이 정보로부터 크기가 9인 초기 집단을 생성한다면, 먼저 각 구간을 균등하게 분할하여 그림 2-8과 같이 9개의 초기화 점을 선택한다. 주어진 문제에서 각 변수가 6비트 길이의 2진 스트링으로 표현된다면 다음과 같은 초기 집단을 생성할 수 있다.



Collection @ chosun

$$x_1(0) = [-3 \quad -3]^T, \qquad s_1(0) = (000000000000)$$
$$x_2(0) = [\ \ 0 \quad -3]^T, \qquad s_2(0) = (011111000000)$$
$$x_3(0) = [\ \ 3 \quad -3]^T, \qquad s_3(0) = (111111000000)$$
$$x_4(0) = [-3 \quad\ \ 0]^T, \qquad s_4(0) = (000000011111)$$
$$x_5(0) = [\ \ 0 \quad\ \ 0]^T, \qquad s_5(0) = (011111011111)$$
$$x_6(0) = [\ \ 3 \quad\ \ 0]^T, \qquad s_6(0) = (111111011111)$$
$$x_7(0) = [-3 \quad\ \ 3]^T, \qquad s_7(0) = (000000111111)$$
$$x_8(0) = [\ \ 0 \quad\ \ 3]^T, \qquad s_8(0) = (011111111111)$$
$$x_9(0) = [\ \ 3 \quad\ \ 3]^T, \qquad s_9(0) = (111111111111)$$

【그림 2-8】 균등 분할하여 얻어진 초기화 점

유전자 알고리즘을 실세계 문제에 응용할 때는 해에 대한 사전지식이 있다면 무작위 초기화법 보단 유도된 초기화법을 택하는 것이 유리하다. 그 이유는 유도된 초기화법은 방법이 전역해로의 수렴 속도가 빨라 계산 시간을 상당히 줄일 수가 있기 때문이다. 그러나 이 때 초기 집단이 유전적 다양성을 유지하도록 세심하게 배려할 필요가 있다. 그렇지 않으면 탐색은 어느 한쪽으로 치우칠 수 있어 전역해를 찾기가 어렵게 된다.[2]

2. 적합도 함수

자연계에서 생물들은 생존경쟁을 통해 끊임없이 투쟁하며 환경에 적응해 가는데, 한 생물의 생존능력은 그 환경에 대한 적합도(Fitness)를 반영한다. 환경에 대한 생물의 적응 능력은 유전자 알고리즘에서는 개체의 적합도를 평가하는 과정을 통해 반영된다.

Collection @ chosun

선택, 교배, 돌연변이 연산을 거쳐 새로운 집단이 완성될 때마다 개체들의 적합도가 평가되는데, 이는 대개 목적함수로부터 계산된다. 환경에 대한 생물의 적합도가 크면 클수록 더 많은 보상을 받듯이, 유전자 알고리즘에서도 적합도가 큰 개체가 더 많은 보상을 받을 수 있도록 배려되어야 한다. 그러기 위해서는 적합도 함수는 최대화 문제 형태로 기술되어야 하고 음의 값을 가져서는 안된다. 이 때문에 적합도 함수를 얻기 위해서는 목적함수의 적절한 사상(mapping)이 요구되는 경우가 발생한다.

이윤 또는 생산율의 극대화와 같이 최대화 문제로 기술되는 목적함수가 정의 영역 안에서 항상 양의 값을 가질 때는 이를 적합도 함수로 취할 수 있지만, 어떤 구간에서 음의 값을 가지게 될 때는 다음과 같은 사상을 통해 적합도 함수를 얻을 수 있다.

$$f(s) = F(x) - \gamma \qquad \text{식 2-9}$$

식 2-9에서 F는 $\Omega \rightarrow R$의 목적함수, f는 $S \rightarrow R^{+}$의 적합도 함수, γ는 항상 $f(s) \geq 0$의 관계를 만족시키는 상수다. γ는 성능에 영향을 끼치는 요소로서 부적절하게 선정되면 탐색 효율을 오히려 저하시킨다.[2][17]

3. 스키마

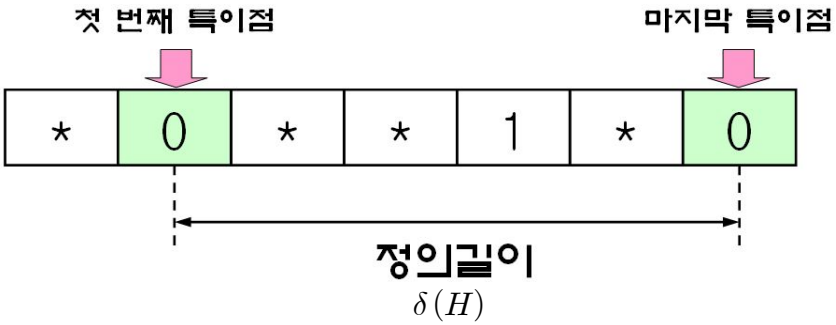
유전자 알고리즘이 한 세대에서 다음 세대로 넘어가면서 어떻게 진화되어 가는지를 설명해 줄 수 있는 메카니즘이 있는데, 이것이 스키마이다. 즉 유전자 알고리즘의 이론적인 기초는 해의 2진 스트링 표현과 스키마 개념이다.

스키마란 스트링 위치의 유사성을 효과적으로 표시할 수 있는 템플릿으로 정의 된다. 2진 스트링의 경우에는 스키마는 2진 알파벳에 추가로 심볼 '*'을 도입

dCollection @ chosun

하여 만들어 진다. 따라서 스키마는 3진 알파벳 {0, 1, *}로 구성되는 또 하나의 스트링으로 나타낼 수 있다. '*'는 "don't care" 문자로서 스트링과 비교할 때 '*'가 있는 위치는 상관하지 않는다는 것을 뜻한다. 스키마 템플릿에 '*'가 많을수록 특이점 0과 1의 수가 적게 되므로 더 많은 스트링을 표현할 수 있는 반면에 '*'가 적으면 적을수록 특이점의 수가 많아 표현할 수 있는 스트링의 수가 적어진다. 스키마는 어떤 위치에서 유전자 간의 유사성을 기술하는 것 외에 또 다른 중요한 성질들을 가지고 있는데 정의길이와 차수 및 빈도가 그것이다. 이들은 스키마 정리를 유도하는 데 중요하다.

○ 정의길이 : 스키마 H의 정의길이는 첫 번째 특이점의 위치와 마지막 특이점의 위치 상이의 거리로 정의된다. 그림 2-9에서 정의길이는 첫 번째 특이점이 2이고 마지막 특이점의 위치가 7이므로 정의길이는 5가 된다.



【그림 2-9】 스키마 정의길이

○ 차수 : 스키마 H의 차수는 스키마에 포함된 특이점의 수로 정의된다. 즉 스키마 전체 길이에서 '*'의 수를 뺀 것이다. 그림 2-9를 예로 들면 차수는 3이다.

Collection @ chosun

○ 빈도 : K세대에서 어떤 스키마 H의 빈도는 한 집단 내에서 이와 일치하는 스트링의 수로 정의된다. 즉 스키마 H=10**0*1 이라 가정하고 집단 $P(k)$=(1000011, 0110110, 0011001, 1001011, 1100011, 1011010}이 있다면 빈도는 2가 된다.

이러한 정의를 바탕으로 선택, 교배, 돌연변이 등의 유전자 연산자에 대하여 영향을 살펴보면 어떤 스키마 H와 일치하는 스트링의 수는 증가 또는 감소할 수가 있다. 스키마의 생성과 파괴는 집단의 구성과 연산자들의 확률적 특성에 따라 매우 복잡한 양상을 띠면서 일어나기 때문에 빈도를 정확하게 예측하는 것은 매우 어렵다. 그러나 근사적인 한계치를 유추해내는 것은 가능하다.[7][8][12]

유전자 알고리즘의 세대 $k+1$에서의 임의의 스키마 H를 포함하는 염색체의 개수의 기대치는 다음과 같다.

$$m(H,k+1) \geqq (H,k)\frac{\overline{f}(H,k)}{\overline{f}(k)}\ [1-P_c\frac{\delta(H)}{L-1}-p_m o(H)] \qquad \text{식 2-10}$$

$m(H,k)$: k세대에서 스키마 H의 빈도

$\overline{f}(H,k)$: 스키마 H와 일치하는 모든 스트링의 평균 적합도

$\overline{f}(k)$: k세대의 평균 적합도

$\delta(H)$: 정의길이

$o(H)$: 차수

L : 염색체 길이

P_c : 교배확률

P_m : 돌연변이 확률

dCollection @ chosun

여기서 $\frac{\delta(H)}{L-1}$은 염색체에서 자름선이 놓일 수 있는 모든 경우에 대한 스키마 길이의 비율 즉, 스키마가 파손될 확률을 나타낸다. 그러나 자름선이 스키마 내부에 놓이는 경우라 하더라도 상대 부모해가 똑같은 특정 기호를 갖고 있다면 그 스키마는 파손되지 않는다. 즉, 스키마 정리는 스키마의 생존 가능성에 대한 하한선을 제시한다.

모든 개체는 각 세대를 통해 선택적 과정을 경험하게 되며 그럼으로서 보다 유리한 변이를 보유한 개체는 덜 유리한 변이를 보유한 개체보다 훨씬 적응된 자손을 남기게 된다. 결국 그 집단은 점차적으로 그 환경을 이겨내는 데 보다 적합하도록 진화하게 되며, 반면 적응도가 결핍된 개체는 선택적으로 불리하게 되어, 이러한 형질을 보유한 개체는 점차 그 수가 줄어들게 되어 연속적인 세대를 거치면서 완전히 사라지게 된다.[15][18]

4. 엘리트 전략

교배 및 돌연변이의 결과로 특별히 좋은 적합도를 갖는 개체가 소실되는 것을 방지하기 위하여 현세대의 가장 좋은 해를 보존하여 다음 세대에 남기는 것이 선택방법이다. 이 방식을 제안하였던 Kenneth De Jung은 엘리트 보존 선택을 다음과 같이 정의하였다.

> "$a^*(t)$를 세대 t까지의 가장 좋은 개체라고 하자. 일반적인 유전자 알고리즘의 메커니즘을 통해 생성된 $t+1$세대의 개체군 $A(t+1)$에 $a^*(t)$가 포함되어 있지 않다면 $a^*(t)$를 개체군 $A(t+1)$의 $(N+1)$번째 개체로 추가한다."

De Jung은 엘리트 보존 선택이 단봉성 문제 즉, 전역적인 해가 하나인 문제의

Collection @ chosun

경우에는 온라인과 오프라인 성능이 향상됨을 보였으나 다봉성 문제에서는 단봉성에서 보인 만큼의 성능 향상을 얻지 못했다. 이는 엘리트로 선정된 개체의 유전자가 세대를 거듭할수록 개체군 전체로 급속히 확산될 가능성이 높아 국소적인 해로 수렴할 확률이 높아지기 때문이다.[19]

C. 유전자 알고리즘 대표 연산자들

유전자 알고리즘에서는 대표적으로 선택(Selection), 교배(Crossover), 그리고 돌연변이(Mutation) 연산자가 이용되는데, 초기에는 주어진 문제에 대한 해가 될 가능성이 있는 개체들의 모집단이 형성된다. 이 때에는 해의 공간 내에서 무작위로 선택하여 모집단을 형성한다. 다음 단계에서는 개체들의 목적함수로 구성된 적합도 항목으로 평가된다. 이후 상대적인 적합도에 따라 개체들이 선택되고 재생산된다. 보통은 적합도가 높은 개체들이 많이 선택될 수 있도록 한다.

다음 단계는 선택된 개체들을 교배를 통해 재결합되는데 서로 간에 유전정보를 교환함으로써 집단 내에 새로운 개체를 생성하게 되며 집단 내에 존재하는 정보만을 이용하여 변화를 시도하게 된다. 만약 이전 세대의 선택 과정에서 특정 정보를 잃게 되면 교배는 이러한 정보를 보유하는 새로운 개체들을 생산할 수 없게 된다.

마지막으로 돌연변이는 선택된 개체의 하나 이상의 유전정보를 임의로 변경하여 집단에 새로운 정보를 도입하는 수단을 제공한다. 그러나 이러한 변화는 매우 낮은 확률로 이행된다.

이렇게 해서 새롭게 형성된 집단은 다시 평가되고 최적해를 발견할 수 있을 때까지 앞서 설명한 연산과정을 반복하게 된다.[18]

dCollection @ chosun

1. 선택(Selection) 연산자

선택 연산자는 교배에 쓰이는 두 개의 부모해를 고르기 위한 연산자이다. 다양한 선택 연산자들이 있으나 공통된 원칙은 우수한 해가 선택될 확률을 높여야 한다는 것이다. 우수한 해들과 열등한 해들 사이의 적합도 차이를 조절함으로써 선택 확률을 조절할 수 있는데, 이 차이의 정도를 선택압(Selection Pressure)이라 한다. 선택압이 높을수록 수렴은 빠르나 설익은 수렴(Premature Convergence)의 가능성이 높아진다. 반면 선택압이 너무 낮으면 해집단의 평균 품질이 좋아지지 않을 가능성이 높다. 선택압은 조절이 가능한 파라미터이다. 다음은 대표적인 선택 연산자에 대한 정의이다.[15]

a. 룰렛 휠 선택(Roulette-Wheel Selection)

가장 대표적인 선택 연산자이다. 각 해의 품질을 평가한 다음 가장 좋은 해의 적합도가 가장 나쁜 해의 적합도보다 k배가 되도록 조절한다. 임의의 비용 최소화 문제의 경우 해집단 내의 해 i의 적합도는 다음과 같이 계산된다.

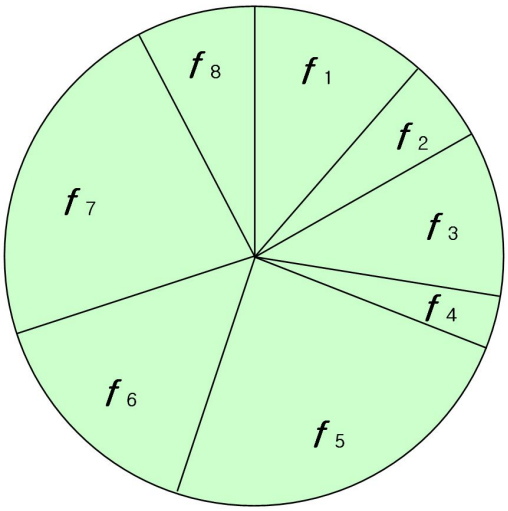
$$f_i = (C_w - C_i) + (C_w - C_b)/(k-1), k > 1 \qquad \text{식 2-11}$$

C_w : 해집단 내에서 가장 나쁜 해의 비용
C_b : 해집단 내에서 가장 좋은 해의 비용
C_i : 해 i의 비용

여기서 k값을 높이면 선택압이 높아진다. 일반적으로 가장 흔히 쓰는 k값은 3~4 이다. 이 적합도 값을 기준으로 룰렛 휠 선택을 하게 된다.

dCollection @ chosun

위와 같이 계산한 각 염색체의 적합도를 모두 합한 값 만큼의 크기를 가진 룰렛 휠을 가정한다. 각 염색체는 이 룰렛 휠 상에 자신의 적합도 만큼의 공간을 배정받는다. 그림 2-10은 8개의 염색체가 적합도에 비례해서 룰렛 휠 상의 공간을 배정받은 모양을 나타내고 있다. 여기에 활을 쏘면 각 염색체의 선택 확률은 배정된 공간의 크기에 비례하게 된다.



【그림 2-10】 룰렛 휠 공간 배정

룰렛 휠 선택은 다음과 같이 간단히 구현될 수 있다.

```
point = random(0, Sum Of Fitness)
sum = 0
for i= 0 to n-1 {
    sum = sum + f i ;
    if (point < sum) return i ;
}
```

【그림 2-11】 룰렛 휠 구현

Collection @ chosun

여기서 f_i는 염색체의 i의 적합도이고, Sum Of Fitness 는 모든 염색체들의 적합도 값을 더한 수치이다. random(0, Sum Of Fitness)은 (0, Sum Of Fitness) 구간에서의 임의의 값을 의미한다.

초보자들이 유전자 알고리즘을 처음 구현하면서 흔히 범하는 실수는 적합도를 조정하지 않고 주어진 문제에서의 해의 품질을 그대로 적합도로 사용하는 것이다. 이렇게 하면 대부분의 경우 해집단에서 가장 좋은 해와 가장 나쁜 해의 품질이 너무 차이가 나서, 적합도에 비례해서 선택 기회를 갖게 되면 나쁜 해들은 거의 선택될 기회를 잃게 된다. 결과적으로 엄청나게 큰 선택압을 주어 해의 다양성을 급속히 떨어뜨리는 결과를 부르게 된다.

b. 토너먼트 선택(Tournament Selection)

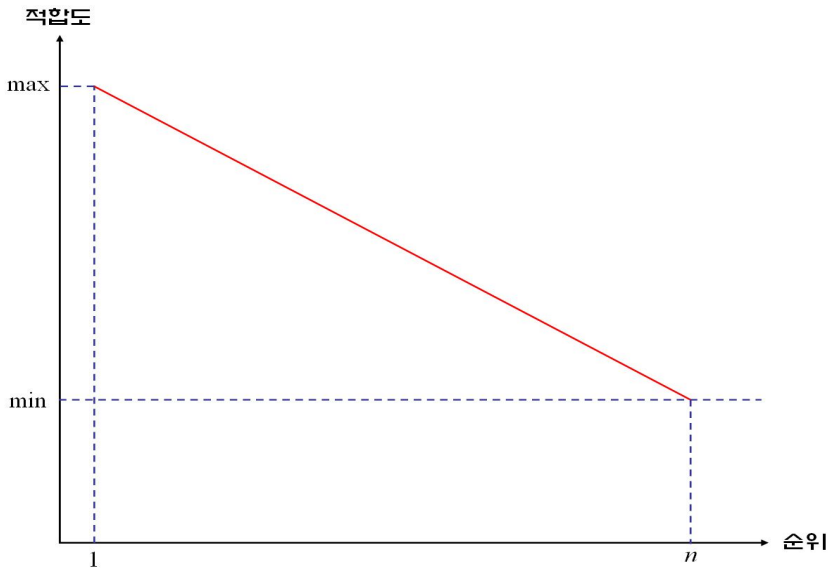
선택 연산자 중 가장 간단한 토너먼트 선택은 두 개의 염색체를 임의로 선택하여 (0, 1) 범위의 난수를 발생시킨 다음 이것이 t보다 적으면 두 염색체 중 품질이 좋은 것을 선택하고 그렇지 않으면 품질이 나쁜 것을 선택하는 것이다. 여기서 t는 파라미터화 할 수 있는 것으로 t값이 높을수록 선택압이 높아진다. 두 개의 염색체를 선택하는 대신 2^r개의 염색체를 선택한 다음 이들을 난수 발생에 의해 토너먼트 형식으로 비교하여 마지막에 남는 것을 선택하는 방법도 있다. 이 경우에도 r값이 클수록, 즉 토너먼트에 참가하는 염색체가 많을수록 선택압이 높아진다. 토너먼트 선택은 실행에 필요한 시간이 매우 짧다는 장점이 있다.

c. 순위 기반 선택(rank-based selection)

룰렛 휠 선택에서는 k값을 조정함으로써 좋은 품질의 해와 나쁜 품질의 해가 지나치게 적합도의 차이를 갖게 되는 것은 막을 수 있다. 그러나 해들의 적합도의 분포는 조절할 수 없다. 반면 순위 기반 선택은 해집단 내의 해들을 품질 순

Collection @ chosun

으로 순위를 정한 다음 가장 좋은 해부터 일차적으로 적합도를 배정하는 방법이다. 그림 2-12는 순위 기반 선택의 적합도 배정 함수를 보여 준다.



【그림 2-12】 순위 기반 선택

n개의 염색체 중 i번째 순위를 가진 염색체의 적합도는 식 2-12와 같이 계산할 수 있다.

$$f_i = max + (i-1) \times (min - max)/(n-1) \qquad \text{식 2-12}$$

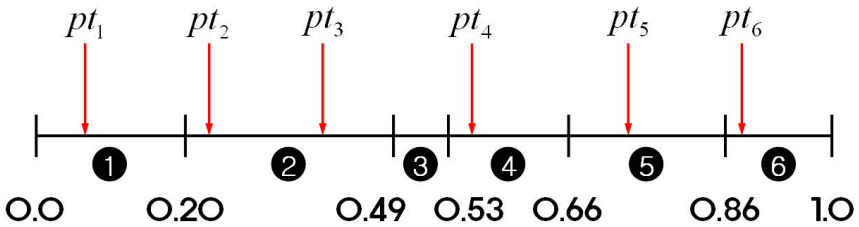
이 식에서 max 와 min 값의 차이를 바꿈으로써 선택압을 조절할 수 있다. 해들의 적합도는 max 와 min 사이에서 균일한 간격으로 분포하게 된다.

d. 확률균등 선택(Stochastic Universal Sampling Selection)

확률균등 선택은 룰렛 휠 선택과 유사하지만 두 방법을 구별하는 차이점이 존

dCollection @ chosun

재한다. 먼저 누적 선택확률을 계산하고 개체의 위치를 정한다. 다음 선택되는 개체 수만큼 그 위에 점들을 위치시킨다. 만약 집단의 크기가 N이라면 점들 간의 간격은 $1/N$이 된다. 첫 번째 점의 위치는 $[0, 1/N]$ 사이에서 임의로 정하고 나머지 점들의 위치는 $1/N$을 누적 합산하면 자동적으로 결정된다. 그림 2-13은 확률균등 선택의 진행과정을 나타낸다. 첫 번째 점은 $[0, 1/6]$ 사이에서 균등 확률로 발생되어 $pt_1 = 0.06$ 이 되었고 나머지 점들은 $pt_2 = pt_1 + 1/6 = 0.23$, $pt_3 = pt_2 + 1/6 = 0.39$, $pt_4 = pt_3 + 1/6 = 0.56$, $pt_5 = pt_4 + 1/6 = 0.73$, $pt_5 = pt_6 + 1/6 = 0.89$ 가 되어, 결과적으로 1, 2, 2, 4, 5, 6번 순으로 선택이 되었다.[2]



【그림 2-13】 확률균등 선택

e. 공유(Sharing Selection)

공유는 해집단의 다양성을 보다 오래 유지하고자 하는 목적에서 고안되었다. 공유는 먼저 품질 비례 선택이나 순위 기반 선택에서 쓰는 적합도 값을 구한다. 여기에 해당 염색체가 해집단 내의 나머지 염색체들과 유사한 (특징을 '공유' 하는) 정도가 높을수록 나누어 적합도를 조정한다. 공유를 감안한 적합도는 아래와 같이 계산된다.

Collection @ chosun

$$F_i = \frac{f_i}{\sum_{j=1}^{n} s(d_{i,j})} \qquad \text{식 2-13}$$

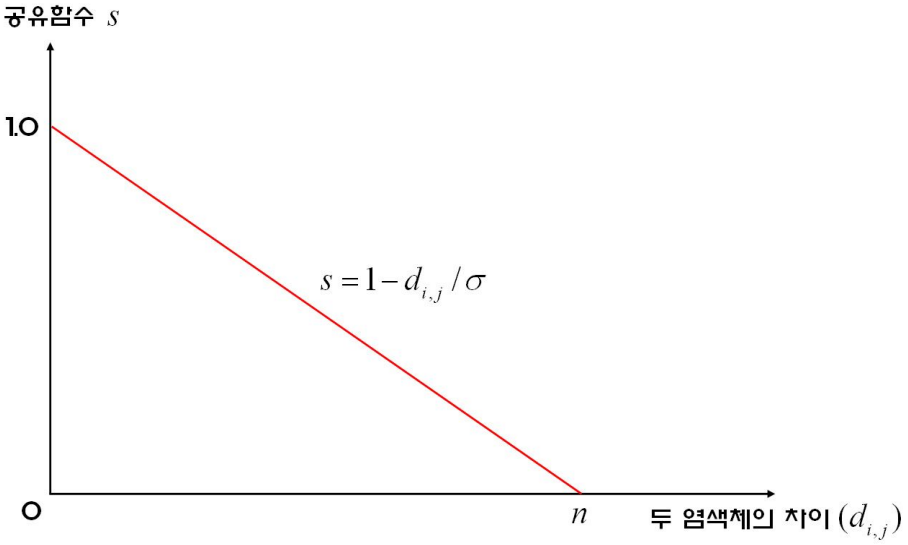
F_i : 공유를 고려한 염색체 i의 적합도

f_i : 일반 적합도

$s(\,)$: 공유 함수

$d_{i,j}$: 염색체 i와 염색체 j의 차이

위의 공유 함수 $s(\,)$는 두 염색체의 차이가 적을수록 큰 값을 가져야 한다.



【그림 2-14】 공유 함수 예

전형적인 공유 함수의 모양은 그림 2-14와 같이 일차 함수로 나타낼 수 있으며, σ는 해집단에서 가장 차이가 많이 나는 두 해의 차이로 잡는 것이 합리적이

dCollection @ chosun

다. 공유 함수는 그림과 같은 일차 함수를 쓰지 않고 다양한 변형이 가능하다. 예를 들어 $s(d) = 1 - (d/\sigma)^2$ 과 같이 잡으면 염색체간의 유사성에 불이익을 더욱 많이 주게 된다.

공유는 해집단 내의 해들을 문제 공간상에서 지리적으로 분산되게 도와주는 효과가 있다. 이는 몇 개의 부분 해집단을 각각 독립적으로 운영하며 가끔 교류시키는 섬식 방법 (island method) 과 유사한 동기를 갖고 있다.

공유는 두 해의 차이를 계산하는 기준을 잡는 방법에 따라 유전자형 공유와 표현형 공유로 나눌 수 있다. 유전자형 공유는 $d_{i,j}$를 위해 염색체의 모양 그 자체가 다른 정도를 사용하는 것이고, 표현형 공유는 표현형의 차이를 계산 기준으로 삼는 것이다. 예를 들어, 각 해가 100개의 십진수를 파라미터로 갖는 문제의 경우, 2진 표현에서 염색체는 400비트의 2진 스트링으로 표현할 수 있다. 유전자형 공유의 경우 두 염색체 간의 거리, 즉 400비트 중 서로 다른 비트의 총 수를 사용하며, 표현형 공유의 경우 400비트로부터 100개의 십진수를 추출하여 두 염색체에서 각 십진수간의 크기의 차이를 사용한다. 공유는 Goldberg와 Richardson에 의해 제안된 이후 많은 연구 결과들이 발표되고 있다.[21][22]

2. 교배(Crossover) 연산자

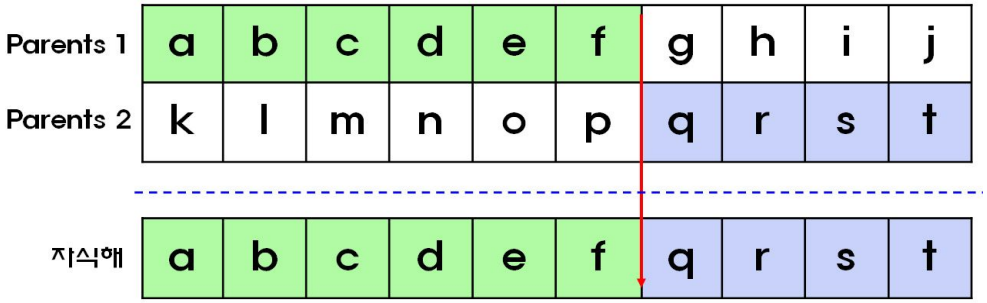
교배 연산은 유전자 알고리즘의 대표적인 연산자이며 연산자들 중 가장 많은 대안을 갖고 있다. 각 종 선택 연산자에 의해 선택된 개체들은 두 개씩 쌍을 이루어 부모 개체를 형성한다. 이들 부모 개체의 염색체를 교환하여 새로운 자식 개체를 형성함으로써 양쪽 개체의 부분 형질을 적당히 조합하여 다음 세대의 개체에 전승시키는 것이 교배이다. 이 때 부모의 우수한 부분 형질이 자식에게 전해지고 세대가 진행될수록 개체군에는 우수한 형질을 지닌 개체들의 수가 증가하여 더 좋은 개체를 생산해 낼 수 있는 기반이 된다.[15]

Collection @ chosun

a. 일점 교배(One-point Crossover)

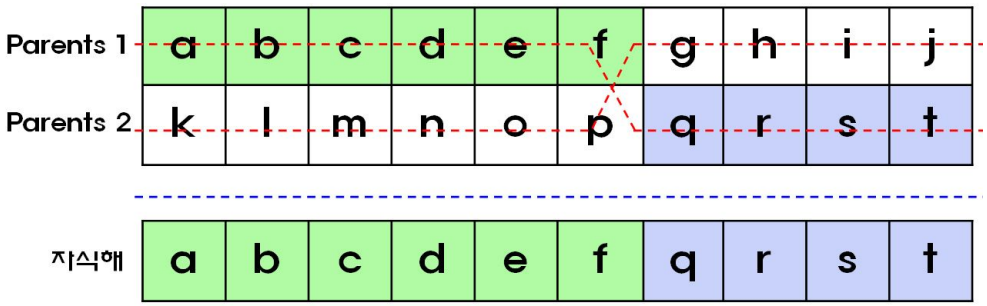
일점 교배는 대표적인 교배 연산자이다. 초기의 유전자 알고리즘은 대부분 일점 교배를 사용하였고, 요즘도 가장 많이 사용되는 연산자이다. 길이가 n인 1차원 문자열로 된 염색체 상에서 일점 교배로 자르는 방법의 총 수는 $n-1$ 가지이다.

두 부모해의 동일한 위치에 자름선을 정하면, 이 자름선은 각 부모해를 두 부분으로 나눈다. 이때 각 부모해로부터 한 부분씩만 가져와서 합치면 새로운 해. 즉, 자식해 하나가 만들어진다.



【그림 2-15】 일점 교배를 통한 자식해 생성

그림 2-15보다 생물학에서의 교배와 유사하게 설명하면 다음과 같다.

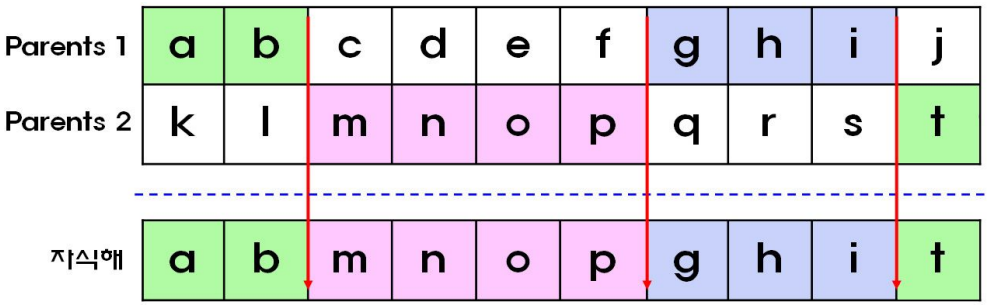


【그림 2-16】 생물학적 측면의 일점 교배

Collection @ chosun

b. 다점 교배(Multi-point Crossover)

다점 교배는 일반적으로 일점 교배보다 자르는 방법의 수가 훨씬 많다. 염색체의 길이가 n일 때 k점 교차로 자르는 방법의 총 수는 ${}_{n-1}C_k$가지이다. 만일 맨 오른쪽 유전자의 오른쪽에도 자름선이 올 수 있도록 하면 방법의 총 수는 ${}_{n}C_k$가 된다. 전통적인 유전자 알고리즘은 위의 ${}_{n-1}C_k$방식을 쓰고 있으나 사실은 ${}_{n}C_k$방식이 더 합리적이다. 다점 교배는 일점 교배보다 교란(Perturbation)의 정도가 크다. 따라서 보다 넓은 탐색 공간을 탐색할 수 있다. 반면에 교란이 강하면 수렴성이 떨어져 주어진 시간내에 정확히 수렴하지 않을 가능성이 크다. 교배 연산에서 스키마의 파손 확률을 자름선의 개수에 영향을 많이 받는다. 따라서 교란이 강하면 강할 수록 스키마가 파손될 확률이 높아진다. 대신 새로운 스키마를 더 다양하게 생성할 수 있다. 일점 교차는 스키마의 길이가 결정적인 영향을 미치고, 다점 교차는 이의 영향을 덜 받는다.



【그림 2-17】 다점 교배를 통한 자식해 생성

그림 2-17은 세 개의 자름선을 사용하는 삼선 교배의 예이다. 두 부모해의 동일한 위치에 세 개의 자름선을 설정하면 각 부모해는 네 부분으로 나뉜다. 두 부모해로부터 이 네부분을 번갈아가며 복사하면 하나의 새로운 자식해가 만들어진다.

일반적으로 다점 교배는 순수 유전자 알고리즘보다 혼합형 유전자 알고리즘에

dCollection @ chosun

더 어울린다. 왜냐하면 혼합형 유전자 알고리즘에는 지역 최적화 기능이 있으므로 순수 유전자 알고리즘보다 교란에 대한 회복력이 강하기 때문이다. 혼합형 유전자 알고리즘의 경우, 교란의 정도가 너무 미약하면 후반에 부모해의 염색체와 비슷한 자식해가 만들어질 확률이 높아져 탐색 시간의 낭비가 커진다.

c. 균등 교배(Uniform Crossover)

일점 교배와 다점 교배가 자름선을 이용하여 구성되는 데 반해 균등 교배는 자름선을 이용하지 않는다. 균등 교배는 먼저 임계 확률 P_0를 설정한다. 가장 일반적인 임계 확률은 0.5이다. 두 부모해를 s_1, s_2라 하자. 각각의 유전자 위치에 대하여 난수를 발생한 다음 이 값이 P_0이상이면 s_1의 같은 위치로부터 유전자를 복사해오고, 그렇지 않으면 s_2의 같은 위치로부터 복사를 한다. 그림 2-18은 P_0 = 0.6인 균등 교차의 한 예를 보여준다.

S_1	a	b	c	d	e	f	g	h	i	j	
S_2	k	l	m	n	o	p	q	r	s	t	
난수	.83	.27	.33	.89	.91	.66	.44	.72	.42	.19	P_0=0.6
자식해	a	l	m	d	e	f	q	h	s	t	

【그림 2-18】 균등 교배의 예

Syswerda에 의해 제안된 균등 교배는 자름선을 이용하지 않으므로 일점 교배나 다점 교배에 비해 스키마의 결합 형태가 다양하다는 장점이 있지만 반면 일반적으로 교란의 정도가 크므로 탐색 시간이 오래 걸린다는 단점이 있다.[23]

dCollection @ chosun

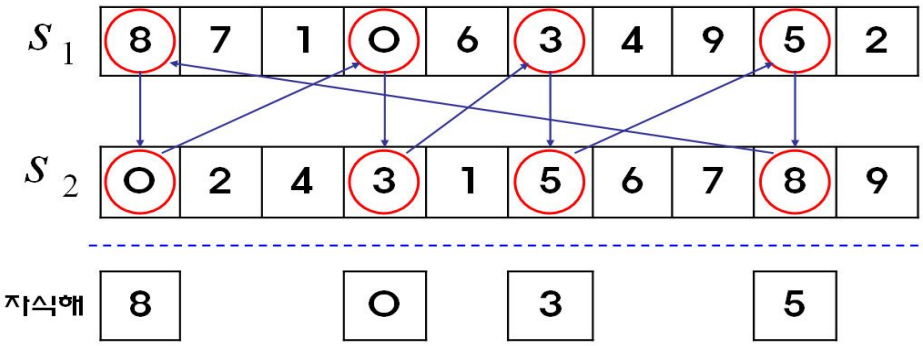
d. 싸이클 교배(Cycle Crossover, CX)

싸이클 교배는 TSP처럼 염색체가 순열로 표현되는 경우에 적용 가능한 교배 연산이다. 그림 2-19와 같이 두 부모해 s_1, s_2로부터 싸이클 교배를 한다고 가정해 보자.

S_1	8	7	1	0	6	3	4	9	5	2
S_2	0	2	4	3	1	5	6	7	8	9

【그림 2-19】 싸이클 교배의 예

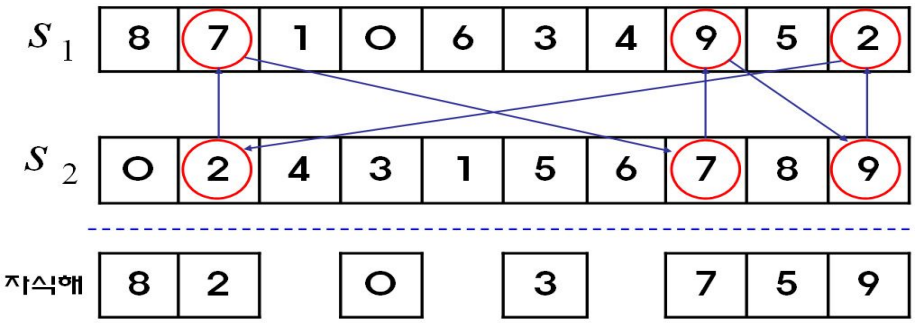
s_1의 첫 번째 유전자 8로부터 복사를 시작한다. 방금 복사한 위치에 대응되는 유전자는 0이므로 s_1상에서 0이 있는 위치를 찾아 자식해의 같은 위치에 0을 복사한다. 같은 원리로 s_1상에서 3이 있는 위치를 찾아 자식해의 같은 위치에 복사하게 되면 5가 복사된다. 이제 5와 같은 위치에는 8이 있는데 8은 이미 한번 복사가 되었으므로 더 이상 복사를 진행할 수가 없다.



【그림 2-20】 싸이클 교배의 진행 과정

dCollection @ chosun

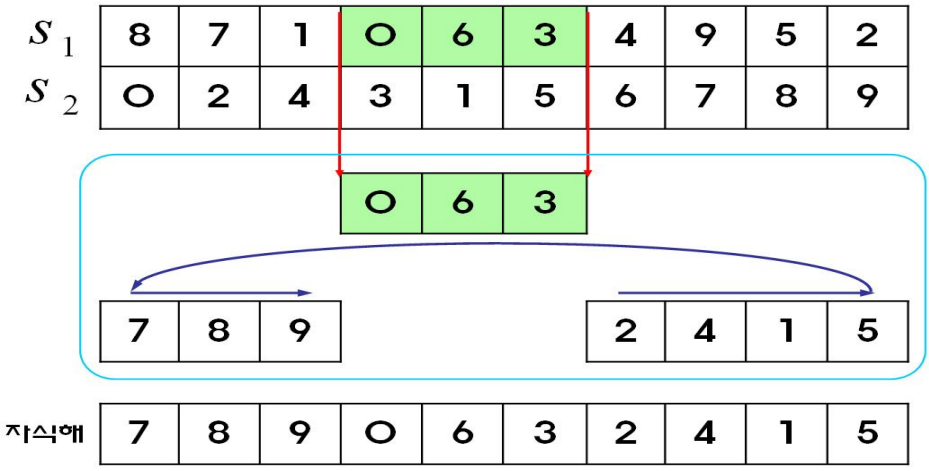
그림 2-20과 같이 화살표를 따라 표시해 보면 하나의 싸이클이 만들어진다는 데에서 싸이클 교배란 이름이 붙게 됐다. 자식해의 유전자 값이 아직 결정되지 않은 위치들 중 가장 왼쪽의 위치로부터 시작하는데 이번에는 s_2로부터 시작한다. 2, 7, 9순으로 복사하면서 그림 2-21처럼 다시 싸이클이 만들어진다.[24]



【그림 2-21】 싸이클 교배의 진행 과정

e. 순서 교배(Order Crossover, OX)

순서 교배는 염색체가 순열로 표시되는 경우를 위해 고안된 교배 연산자이다.



【그림 2-22】 순서 교배의 예

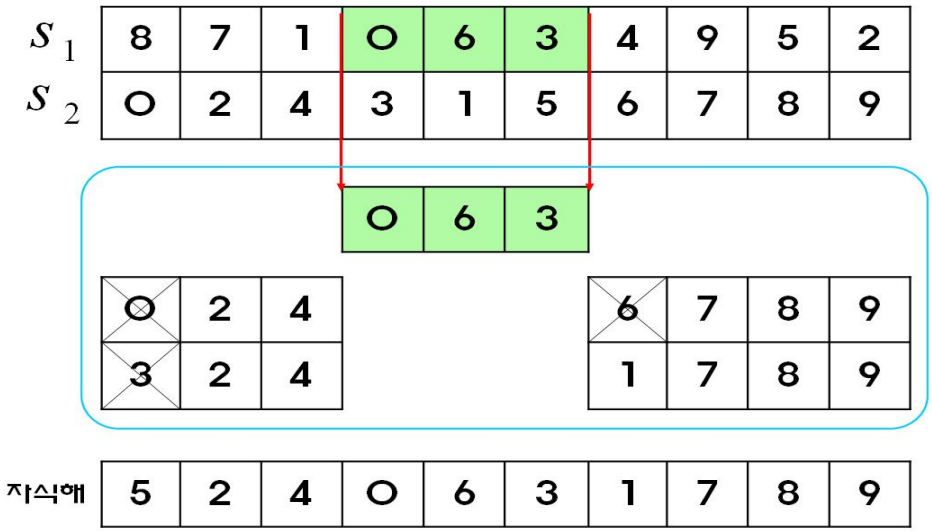
dCollection @ chosun

두 부모해 s_1, s_2가 주어질 경우 먼저 임의로 두 개의 자름선을 정한 다음 두 자름선 사이에 있는 부분을 s_1으로부터 복사한다. 나머지 위치는 s_2로부터 복사하는데, 두 번째 자름선 바로 다음 위치부터 시작하여 사용되지 않은 기호들 중 s_2에서 나타난 순서대로 복사한다.[25]

f. PMX(Partially Matched Crossover)

PMX 역시 염색체가 순열로 표시되는 경우를 위하여 고안된 교차 연산자이다. 두 부모해 s_1, s_2에 임의로 두 개의 자름선을 정한 다음 두 자름선 사이에 있는 부분을 s_1으로부터 복사한다. 나머지 위치는 s_2로부터 복사하되 만일 해당 값이 이미 사용되었으면 같은 값을 가진 s_1의 위치를 찾아 같은 위치의 s_2로부터 복사한다. 복사된 값은 아직 사용되지 않았을 수도 있고, 이미 사용되었을 수도 있다.[26]

그림 2-23은 PMX 연산자로 자식해를 산출해 내는 과정을 나타내는 예이다.



【그림 2-23】 PMX 예

dCollection @ chosun

g. 산술적 교배(Arithmatical Crossover)

산술적 교배는 실수 염색체를 사용하는 경우에 사용할 수 있는 교배 연산자이다. 염색체 각 위치에 대해 두 부모 염색체의 두 유전자 값의 평균을 내어 자식해의 해당 위치의 값으로 배정한다. 그림 2-24는 간단한 산술적 교배의 한 예를 나타내고 있다.[27]

S_1	1.98	3.31	20.43	12.01	-2.34	8.34	98.86
S_2	11.28	2.21	12.39	1.44	2.45	3.55	87.44
자식해	6.63	2.76	16.41	6.73	0.06	5.95	93.15

【그림 2-24】 산술적 교배의 예

산술적 교배는 수의 '크기'라는 개념을 교배 행위에 사용한다는 것이 큰 장점이고 수들의 산술평균을 지향해서 탐색 속도가 매우 빠르다는 특징이 있다.[27]

h. Edge Recombination(ER)

Edge Recombination(ER)은 TSP를 위해 개발된 연산자이다. TSP의 두 부모해를 분석, 각 도시에 연결된 인접 도시 목록을 만든다. 한 도시는 최대 4개까지의 인접 도시를 가질 수 있고, 최소 2개는 가진다. 시작 도시는 부모해 중 하나가 될 수도 있고, 인접 도시 수가 가장 적은 도시가 될 수도 있다. 현재 도시의 인접 도시 리스트에 있는 도시 중 인접 도시 수가 가장 적은 도시를 고른다. 후보 도시가 둘 이상이면 그 중 하나를 임의로 선택한다. 이렇게 부모해의 특징을 분석하여 휴리스틱하게 구성하는 방식이다.[28]

dCollection @ chosun

S_1	0	1	2	3	4	5	6	7	8	9
S_2	2	5	0	9	7	3	8	6	1	4

City	Links	City	Links
0	9 1 5	5	4 6 2 0
1	0 2 6 4	6	5 8 1
2	1 3 4 5	7	8 9 3
3	2 4 7 8	8	6 7 3
4	3 5 1 2	9	7 0

	1	0	9	7	5	0	1	5	2
자식해	0	9	7	8	6	1	2	4	3	5
	5		3	3	8	6	3	1	7
						4	5	2	4

【그림 2-25】 Edge Recombination(ER) 교배의 예

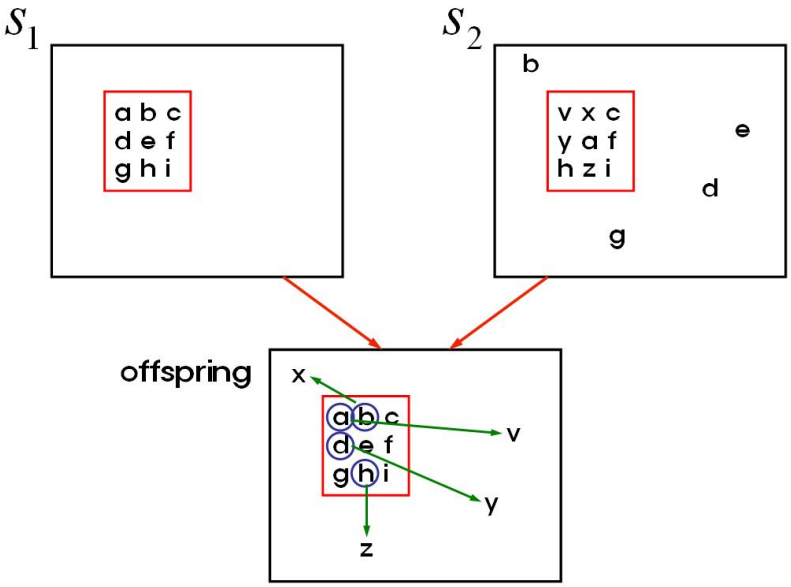
그림 2-25는 ER 연산자의 예를 나타낸 것이다. 먼저 무작위로 하나의 도시를 선택한 다음(그림 2-25에서는 0번 도시를 선택하였다.) 선택한 도시 중 연결 가능한 에지수가 제일 적은 도시를 고른다. 만약 에지수가 같은 경우에는 무작위로 하나를 선택한다. 이 과정을 마지막까지 수행하여 새로운 자식해를 구한다.

i. 다차원 교배

앞에서 소개한 교배 연산자들은 1차원 염색체를 대상으로 한 것이다. 그 밖에

Collection @ chosun

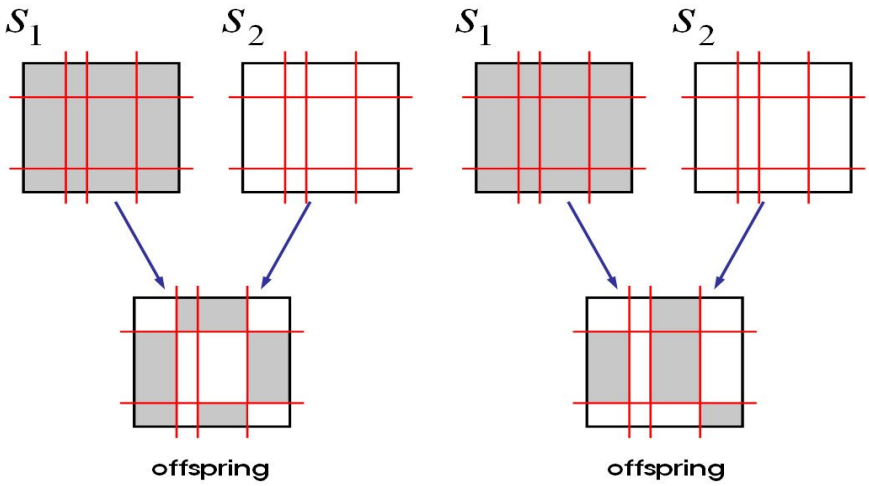
2차원 이상의 표현 방법을 쓴 염색체들에 대한 연산자도 있다. Cohoon과 Paris는 VLSI 회로 배치 문제를 위해 2차원의 염색체를 사용하였고 이를 위한 교배 연산을 제안하였다. 이들은 한 부모해에서 정사각형을 잘라낸 다음 다른 부모해의 해당 부분으로 보충해 넣는 방식으로 교배를 하였다. 그림 2-26은 다차원 교배의 예를 나타낸다. 그림의 한 부모해 s_1에서 3×3의 사각형을 선택하여 이를 s_2의 같은 자리에 놓는다. 그러면 s_2에서 심볼 B, D, E, G는 중복이 되므로 이들을 제거하고 s_2의 사각형에 들어 있던 V, X, Y, Z를 대치해 넣는다.[29]



【그림 2-26】 Cohoon과 Paris의 다차원 교배 예

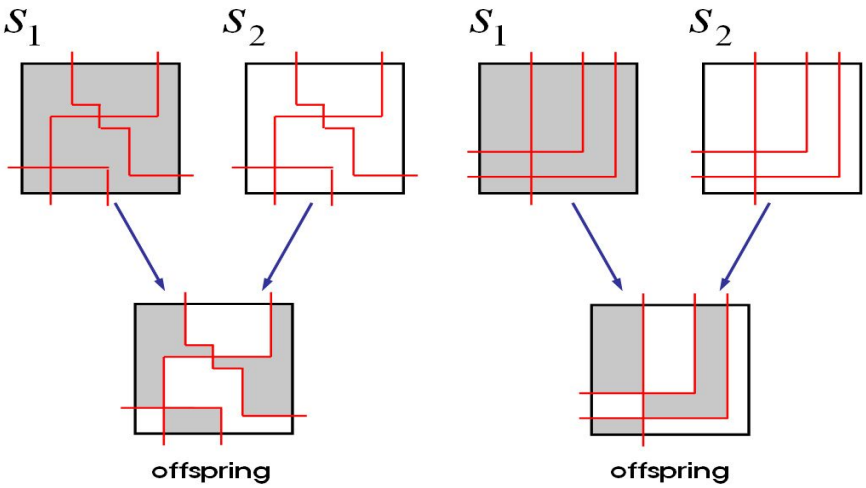
Anderson은 염색체를 $i \times j$블록으로 분할한 다음 각 블록에 대해 난수를 발생시켜 0 또는 1의 값을 얻은 다음, 0 이면 s_1으로부터, 1 이면 s_2로부터 복사하는 블록 균등 교배(block uniform crossover)를 제안하였다. 그림 2-27은 이의 예를 보인다.[30]

Collection @ chosun



【그림 2-27】 불록 균등 교배 예

다차원의 교배 연산에서 직선만을 사용하여 염색체를 분할하는 방법은 1차원 상의 교배 연산들에 비해 다양성이 떨어져 새 스키마의 생성 능력이 떨어지므로 직선의 제약을 없앤 다차원 지오그래픽(Geographic) 교배 연산자가 제안되었다 [Kahng & Moon, 1995]. 그림 2-28은 이의 예를 나타내고 있다.[31]



【그림 2-28】 직선의 제약을 없앤 2차원 지오그래픽 교배 예

Collection @ chosun

j. 내추럴 교배(Natural Crossover)

2차원 염색체를 사용하더라도 행렬과 같은 모양으로 각 유전자를 위치시키는 방법은 1차원 표현보다는 유전자들 간의 지리적 인접성을 잘 반영한다는 장점이 있지만 문제 자체에 깃든 유전자들 간의 지리적 정보를 한동안 잃어버리는 단점은 피할 수 없다. 앞에서 소개한 다차원 교배들은 모두 행렬 형태의 정형적인 염색체를 사용하였다. 이를 착안하여 Jung 과 Moon은 TSP 를 위한 2차원 교차인 내추럴 교배를 제안하였다. 내추럴 교배는 2차원 지도상의 도시들의 분포와 투어의 모양 그 자체를 염색체로 삼는다. 이렇게 하면 염색체 표현 과정에서 도시들의 인접성 정보에는 손실이 발생하지 않는다. 내추럴 교배는 다음과 같은 과정으로 이루어진다.[32][33]

① 2차원 상에서 도시들의 위치가 표시된 그래픽 이미지 상에서 자연스런 손의 궤적을 따라 k개의 자름선을 긋는다.

② 이로 인해 모든 도시들은 두 개의 그룹으로 나뉜다. 이를 각각 클래스 1, 클래스 2라 하자.

③ 부모해 1에 속한 간선들 중 두 끝점이 모두 클래스 1에 속한 간선들을 자식해에 포함시킨다. 마찬가지로 부모해 2에 속한 간선들 중 두 끝점이 모두 클래스 2에 속한 간선들을 자식해에 포함시킨다.

④ 위 ③의 결과로 나온 부분적 투어들에 간선들을 보충하여 유효한 투어를 만든다.

앞에서 설명한 다른 교배 연산자들에 비해 내추럴 연산자는 구현하기 다소 까

Collection @ chosun

다로운 교배 연산자 중 하나이다. 내추럴 교배는 지오그래픽 교배에서 더욱 발전한 형태로, 다차원 표현의 이점은 극대화하되 이의 약점이 될 수 있는 새 스키마의 생성 능력을 효율적으로 보완한 연산자이다.[32][33]

3. 돌연변이(Mutation) 연산자

교배는 두 부모의 속성을 부분 복사함으로써 새로운 해를 만든다. 따라서 교배로 인해 만들어지는 해의 유전자들은 모두 부모해로부터 물려받은 것이다. 그러면 부모해의 유전자들을 좋은 방향으로 결합하는 것은 가능하지만 부모해에 없는 유전자를 도입하기는 힘들다. 이 역할을 하는 것이 돌연변이 연산자이고 새로운 유전 형질을 부여할 수 있다는 점에서 돌연변이 연산자는 필수적 요소이다. 때로는 교배 연산자 없이 돌연변이 연산자만으로 탐색을 시도하는 진화 연산 알고리즘도 있다.

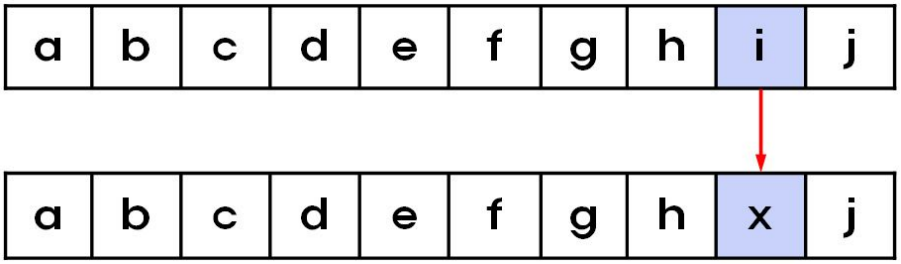
유전자 알고리즘에서는 보통 하나의 비트를 주어진 확률에 따라서 다른 값으로 바꾸는 것으로 돌연변이 연산자를 정의한다. 돌연변이 확률을 높이면 보다 다양한 해를 생성해낼 수 있어서 유전자 알고리즘의 역동성은 늘어나지만 해집단의 수렴성이 떨어져 탐색 시간이 길어지고 개선의 속도가 느려지는 경향이 있다.

a. 전형적 돌연변이

돌연변이 연산은 부모해에 없는 속성을 도입하여 탐색 공간을 넓히려는 목적을 가진 연산자이다. 전형적 돌연변이 연산자는 각각의 유전자에 대하여 (0, 1) 범위의 난수를 생성하여 미리 정한 임의의 임계값 미만의 수가 나오면 해당 유전자를 임의로 변형시키고, 그 이상의 수가 나오면 그냥 다음 단계를 진행한다. 전형적 임계값으로는 0.015 등을 들 수 있는데 이것은 해결하고자 하는 문제와 유전자 알고리즘의 타입에 따라 상당한 폭으로 변할 수 있다. 변이의 결과로 잘

Collection @ chosun

못 만들어진 유전자는 시간이 흐르면서 사라지고 가끔 발생하는 성공적 변이는 집단의 품질을 향상시키는 데 기여를 하게 된다.[15]



【그림 2-29】 전형적 돌연변이 예

그림 2-29는 전형적 돌연변이 연산자의 예를 나타내는 것으로써 열 개의 유전자 중 단 하나만 변이가 일어난 경우이다.

b. 비균등 돌연변이

초기에는 유전자 알고리즘 해의 품질이 좋지 않은 것이 보통이나 시간이 지남에 따라 점점 품질이 좋아진다. 따라서 초반에는 돌연변이의 정도가 다소 강해도 품질 향상이 일어날 가능성이 있지만, 해가 상당한 수준에 이른 후반에는 돌연변이가 강하면 품질 향상이 일어나기 힘들다. 이런 과정에 착안하여 유전자 알고리즘이 진행됨에 따라 점점 돌연변이의 강도를 줄여가는 비균등 돌연변이 연산자가 제안되었다. 임의의 실수 유전자 v에 대한 비균등 변이는 2진 난수 r을 발생시킨 다음 아래 식과 같이 이루어진다.

$$v = \begin{pmatrix} v + \Delta(t,\ UB - v),\ \text{if}\ \ r = 0 \\ v - \Delta(t,\ v - LB),\ \text{if}\ \ r = 1 \end{pmatrix} \qquad \text{식 2-14}$$

Collection @ chosun

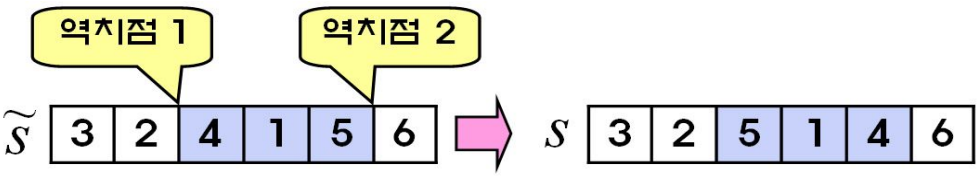
여기서 UB와 LB는 유전자 v가 가질 수 있는 상한값과 하한값이다. $\Delta(t, y)$는 0과 y사이의 값을 갖는데 시간 t가 증가함에 따라 점점 0으로 근접하는 특성을 갖는다.[27]

c. 기호코딩

순서화된 파라미터 집합의 순열을 최적화하는 문제를 다루는 유전자 알고리즘에 기존의 돌연변이를 사용하면 TSP에서 도착지가 두 번 나타난다든가 또는 VLSI 설계 문제에서 한 회로의 요소가 빠진다거나 하는 적합하지 못한 해를 생성할 수 있다. 이러한 문제를 다루는 돌연변이 연산자들은 부적합한 해의 생산을 예방하기 위하여 부모가 가지는 모든 요소들이 자손 안에 나타나도록 해준다.[2]

(1) 역치(Inversion)

역치는 부모 염색체안에서 두 역치점을 임의로 선택하고, 선택된 부스트링을 역순으로 재배치시키게 된다. 그림 2-30은 역치의 한 예를 보여준다. 이를 확장하면 다점교배와 유사하게 다점역치 연산이 가능하다.

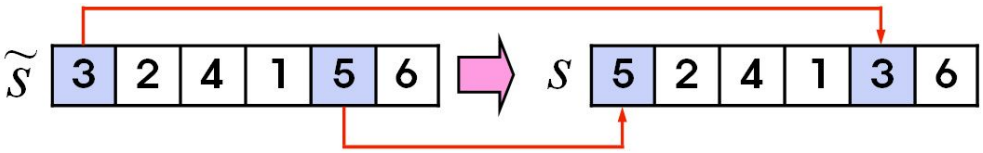


【그림 2-30】 역치 돌연변이 예

(2) 교체 돌연변이(Swap Mutation)

상호 돌연변이라고도 불리우는 이 돌연변이는 염색체 내에서 임의로 두 요소를 취하고 이들을 서로 교환함으로써 진행된다. 그림 2-31은 교체 돌연변이의 예를 보여주고 있다.

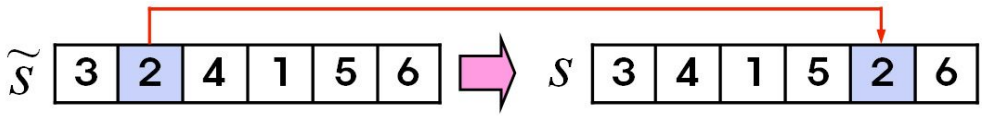
Collection @ chosun



【그림 2-31】 교체 돌연변이 예

(3) 삽입 돌연변이(Insertion Mutation)

삽입 돌연변이는 염색체 내에서 한 요소를 임의로 선정하고 이를 다시 임의의 위치에 삽입함으로써 실현된다. 아래 그림 2-32는 요소가 선택되어 5번 위치에 삽입된 결과를 보여주고 있다.



【그림 2-32】 삽입 돌연변이 예

dCollection @ chosun

Ⅲ. 새로운 알고리즘 제안

본 논문에서는 최적의 TSP결과 값을 얻기 위해 혼합 초기화법과 순차변환 기법을 제안한다.

TSP의 최적의 결과를 얻기 위해서는 최초 집단을 생성하기 위한 염색체 초기화 과정이 무엇보다도 중요하다. 각 종 연산자를 적용하기 위한 염색체 집단이 어떠한 구조로 생성이 되느냐가 최적의 해를 구하기 위한 중요한 첫 번째 과정이기 때문이다.

최적해를 구하기 위한 집단 초기화방법 중에는 무작위 초기화법과 유도된 초기화법이 대표적으로 이용되는데 본 논문에서는 무작위 초기화법과 유도된 초기화법을 동시에 이용하는 새로운 방법의 혼합 초기화법을 유추해 냈다.

또한 각 세대에서 선택 연산자를 적용할 때 우성 인자들끼리 선택될 수 있는 확률을 높이기 위해 인위적으로 개체의 순서를 변화시키는 새로운 방법을 제안하여 최적의 TSP 해를 구하기 위한 실험을 하였다.

A. 혼합 초기화법

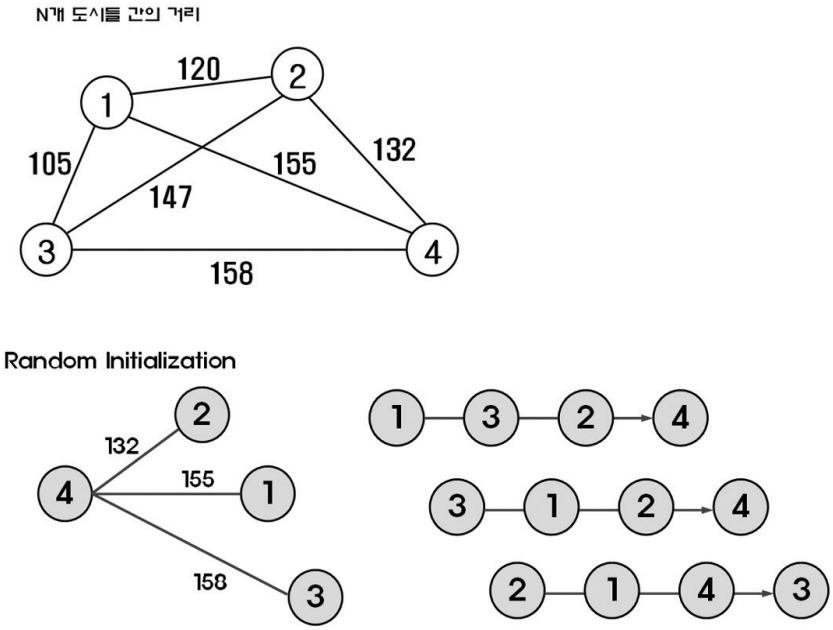
각종 연산자를 통해 국소 최적해에 가장 근접한 결과 값을 얻기 위해서는 무엇보다 최초 집단을 초기화시키는 방법이 중요하다.

집단 초기화 방법에는 크게 두 가지 방법이 있다. 하나는 집단 생성에 있어서 특별한 규칙 없이 무작위로 추출해서 집단을 생성하는 무작위 초기화법(Random Initialization)이고, 다른 하나는 주어진 값을 통해 어떠한 사전 지식이나 정보를 바탕으로 일관성을 유지하여 집단을 생성하는 유도된 초기화법(Directed Initialization)이다.

Collection @ chosun

지금까지 TSP 해결을 위해 사용된 집단 초기화 방법으로는 무작위 초기화법이 많이 사용되어 왔다.

본 논문에서는 난수발생기를 이용한 무작위 초기화법과 사전에 주어진 도시 정보나 경험을 바탕으로 한 유도된 초기화법을 동시에 사용하는 혼합 초기화법을 제안한다.



T_1 = 4-2-1-3 T_2 = 1-3-2-4

T_3 = 3-1-2-4 T_4 = 2-1-4-3

【그림 3-1】 혼합 초기화법

그림 3-1은 혼합초기화법의 진행 과정을 보여주고 있다.

난수발생기를 이용하여 시작 도시를 무작위로 선택한 다음 사전에 알고 있는

dCollection @ chosun

각 도시들 간의 거리 정보를 바탕으로 선택된 도시를 기준으로 상호간의 거리가 가장 짧은 도시부터 순서대로 나열한다. 이러한 순서로 N개의 모든 도시를 나열하면 총 $N \times N$ 개의 행렬이 나오게 된다. 이렇게 생성된 행렬을 최초 집단으로 사용하였다.

이 경우 무작위 초기화법을 사용했을 때와는 달리 사전에 알고 있는 거리 정보를 최대한 활용함으로써 보다 나은 집단을 생성할 수 있도록 집단의 배열을 인위적으로 유도하는 역할을 하게 된다. 따라서 무작위 초기화법과 같이 무원칙에 의한 초기화 방법보단 주어진 정보를 최대한 활용하는 혼합 초기화법이 집단을 초기화함에 있어서 보다 효과적임을 유추하고 설계하였다.

B. 순차변환 기법

세대가 거듭될수록 우수한 인자를 얼마만큼 많이 유지를 하느냐가 좋은 결과를 얻을 수 있는 결정적인 요소이다.

본 논문에서는 각 세대별로 우수한 인자를 많이 유지하기 위해 선택 연산이 진행되기 전에 전 세대에서 생성된 집단의 해를 순차적으로 변환시킨 후 선택 연산을 적용시키는 순차변환 기법을 제안한다.

1세대 진행 결과			순차변환 적용	
T_1 = 4-2-1-3	Dis_1 = 515		T_1 = 4-2-1-3	Dis_1 = 515
T_2 = 1-3-2-4	Dis_2 = 539	$\Rightarrow$	T_3 = 3-1-2-4	Dis_3 = 515
T_3 = 3-1-2-4	Dis_3 = 515		T_2 = 1-3-2-4	Dis_2 = 539
T_4 = 2-1-4-3	Dis_4 = 580		T_4 = 2-1-4-3	Dis_4 = 580

【그림 3-2】 순차변환 진행 과정

Collection @ chosun

그림 3-2는 순차변환 과정을 설명한 것으로 선택 연산이 진행될 때 전 세대의 해를 가지고 가장 우수한 해를 만들어낸 개체부터 순차적으로 재배열한다. 우수한 인자 순으로 재배열 한 후 다음 세대의 연산을 진행할 경우 우성인자들끼리 선택될 확률이 높아진다.

이러한 재배열 과정 중 우성인자를 다음 세대까지 계속해서 유지시키기 위해 엘리트 전략을 사용하였다. 제Ⅱ장에서 설명하였듯 엘리트 전략은 집단 내에서 가장 강한 개체가 다음 세대까지 소멸되지 않고 전달되는 것을 보장하기 위해 고안된 것이다. 이전 세대의 최적 개체를 저장하고 있다가 일련의 모의진화가 수행된 후 현 세대에서 소멸된 것이 확인되면, 저장된 최적 개체를 현 세대에서 가장 약한 개체 또는 임의로 선택된 개체와 교환하게 된다. 이처럼 순차변환 과정을 진행할 때 엘리트 전략을 사용하여 우수한 개체를 유지하면서 다음 세대에 활용할 수 있도록 하였다.

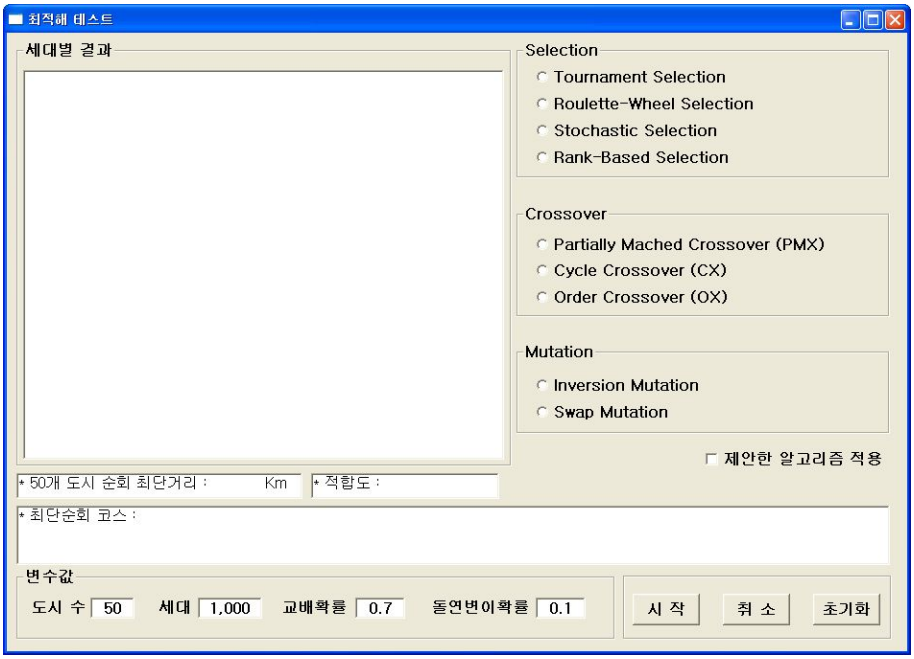
C. 제안한 알고리즘 구현

본 논문에서 제안한 혼합 초기화법과 순차변환 기법의 결과 분석을 위해 4개의 선택 연산자와 3개의 교배 연산자, 2개의 돌연변이 연산자를 사용하여 실험을 진행하였다. 우성인자를 보존하기 위해서는 모든 실험에 엘리트 전략을 사용하여 모든 세대에서 2개의 우성인자를 보존하였다.

도시 수는 50개일 경우와 100개일 경우를 각각 실험하였고, 집단의 크기는 1,000으로 한정하였으며, 교배 확률(P_c)과 돌연변이 확률(P_m)은 평균값인 0.7과 0.1을 사용하였다.

위 변수를 조건으로 주어진 도시의 최단거리를 계산하는 실험을 진행하였고, 본 논문에서 제안한 알고리즘을 적용하였을 경우와 적용하지 않았을 경우의 결과 값을 서로 비교하여 제안한 알고리즘의 성능을 검증하였다.

Collection @ chosun



【그림 3-3】 실험에 사용한 UI

그림 3-3은 본 논문에서 제안한 알고리즘을 적용하여 그 성능을 검증하기 위해 직접 개발한 실험용 어플리케이션의 UI(User Interface)를 보여주고 있다.

어플리케이션 구현 환경은 아래와 같고, 3-Tire 방식에 CS 환경으로 구성하였다.

- ◉ OS - Solaris 5.8
- ◉ Middleware - Tmax 3.8
- ◉ Compiler - Pro C
- ◉ User Interface - PowerBuilder 9.0
- ◉ DBMS - Oracle 10g

모든 실험은 동일한 클라이언트에서 각각 두 차례씩 진행하였고, 우수한 결과

dCollection @ chosun

를 얻은 데이터를 해당 실험 결과 값으로 사용하였다. 다양한 연산자에서 반복 실험한 이유는 제안한 알고리즘을 폭넓게 적용하였을 때 어느 한 가지 연산자에만 국한되지 않고 얼마만큼 효과적인 결과를 얻어낼 수 있는지를 확인하기 위함이다.

선 택	교 배	돌연변이	선 택	교 배	돌연변이
Tournament	PMX	Inversion	Stochastic	PMX	Inversion
		Swap			Swap
	CX	Inversion		CX	Inversion
		Swap			Swap
	OX	Inversion		OX	Inversion
		Swap			Swap
Roulette-Wheel	PMX	Inversion	Rank-Based	PMX	Inversion
		Swap			Swap
	CX	Inversion		CX	Inversion
		Swap			Swap
	OX	Inversion		OX	Inversion
		Swap			Swap

【표 3-1】 실험에 사용한 연산자

표 3-1은 실험에 사용한 연산자 목록을 나타내고 있다. 모든 연산자 소스는 직접 코딩하여 서버에서 실행되도록 컴파일하였다.

그림 3-4는 실험을 위해 PowerBuilder로 개발한 어플리케이션의 소스 코드 일부를 보여주고 있다.

dCollection @ chosun

```
/*****************************************************************************/
/*                                                                           */
/*   윈도우명 : TSP 실행                                                     */
/*   기    능 : 제안한 알고리즘과의 성능 비교                                */
/*   개 발 자 : 강 래 구                                                     */
/*                                                                           */
/*****************************************************************************/

int     i_key
string  s_genmax, s_popsize, s_citysize, s_crossp, s_mutp, s_rqst
string  s_selection, s_s_selection
string  s_crossover, s_s_crossover
string  s_mutation,  s_s_mutation

s_genmax    = l_genmax.text
s_popsize   = l_popsize.text
s_citysize  = l_citysize.text
s_crossp    = l_crossp.text
s_mut       = l_crossp.text
s_rqst      = s_genmax + TAB + s_popsize + TAB + s_citysize + TAB + s_crossp + TAB + s_mut


If i_key = 0 Then  //제안 알고리즘 적용
   CHOOSE CASE s_selection  //선택 연산자
      CASE rb_tou
         s_s_selection = fg_selection(i_key, "tournament",    s_rqst)
      CASE rb_rou
         s_s_selection = fg_selection(i_key, "roulettewheel", s_rqst)
      CASE rb_sto
         s_s_selection = fg_selection(i_key, "stochastic",    s_rqst)
      CASE rb_ran
         s_s_selection = fg_selection(i_key, "rankbased",     s_rqst)
   END CHOOSE

   CHOOSE CASE s_crossover  //교배 연산자
      CASE rb_PMX
         s_s_crossover = fg_crossover(i_key, "PMX",           s_rqst)
      CASE rb_CX
         s_s_crossover = fg_crossover(i_key, "CX",            s_rqst)
      CASE rb_OX
         s_s_crossover = fg_crossover(i_key, "OX",            s_rqst)
   END CHOOSE

   CHOOSE CASE s_mutation  //돌연변이 연산자
      CASE rb_inv
         s_s_mutation  = fg_mutation(i_key, "inversion",      s_rqst)
      CASE rb_swap
         s_s_mutation  = fg_mutation(i_key, "swap",           s_rqst)
   END CHOOSE
Else  //무작위 초기화법 적용
   CHOOSE CASE s_selection  //선택 연산자
      CASE rb_tou
         s_s_selection = fg_selection(i_key, "tournament",    s_rqst)
      CASE rb_rou
         s_s_selection = fg_selection(i_key, "roulettewheel", s_rqst)
      CASE rb_sto
         s_s_selection = fg_selection(i_key, "stochastic",    s_rqst)
      CASE rb_ran
         s_s_selection = fg_selection(i_key, "rankbased",     s_rqst)
   END CHOOSE

   CHOOSE CASE s_crossover  //교배 연산자
      CASE rb_PMX
         s_s_crossover = fg_crossover(i_key, "PMX",           s_rqst)
      CASE rb_CX
         s_s_crossover = fg_crossover(i_key, "CX",            s_rqst)
      CASE rb_OX
         s_s_crossover = fg_crossover(i_key, "OX",            s_rqst)
   END CHOOSE
                                   연산자
                                   rsion",      s_rqst)
                                   rqst)
```

【그림 3-4】 어플리케이션 소스 코드(예)

Collection @ chosun

Ⅳ. 구현 및 평가

A. 50개 도시 순회 실험

먼저 50개의 도시가 주어졌을 때 표 3-1의 연산자를 각각 적용하여 본 논문에서 제안한 혼합 초기화법과 순차변환 기법을 적용했을 때 최단거리를 찾아내는 실험을 통해 본 논문에서 제안한 알고리즘의 성능을 확인하였다.

총 세대 수는 1,000세대 동안 반복하면서 최단거리를 검색하도록 설계하였다.

1. 토너먼트(Tournament) - PMX - Inversion

선택 연산자로 토너먼트, 교배 연산자는 PMX, 돌연변이 연산자로는 Inversion 연산자를 사용하였고, 기타 변수로는 기본값을 사용하였다.

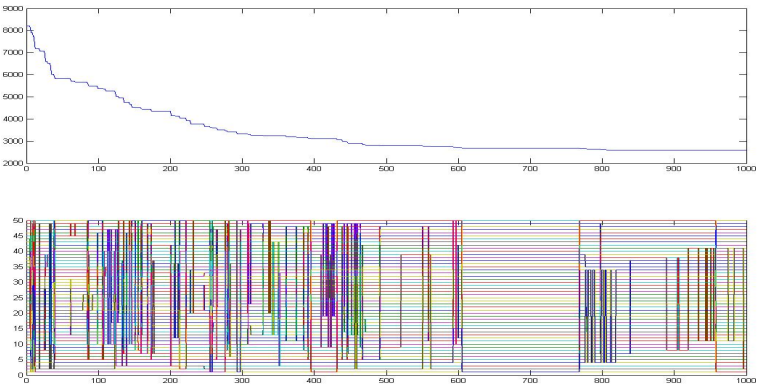
그림 4-1은 본 논문에서 제안한 혼합 초기화법 및 순차변환 기법을 적용한 알고리즘 결과와 적용하지 않은 결과를 각각 보여주고 있다.

각각의 결과에서 위쪽에 나타나 있는 그래프는 세대가 진행됨에 따라 해당 세대별로 최단거리를 찾아 변화하는 모습을 그래프로 나타내고 있는데, 세대가 진행될 수록 최단거리가 점점 줄어드는 걸 확인할 수 있다. (x축 : 세대, y축 : 최단거리)

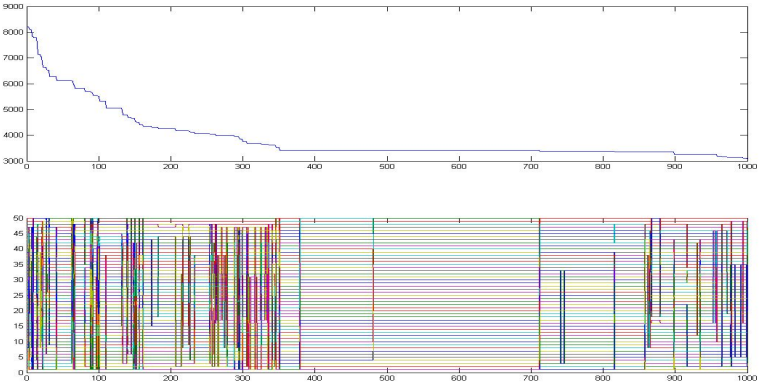
아래 쪽에 나타나 있는 그래프는 세대별로 계산된 최단거리의 도시 방문 경로를 보여주고 있다. 각 세대에서 최단거리로 계산된 도시의 방문 경로가 세대가 거듭될수록 어떠한 순서로 바뀌어 가는지를 확인할 수 있다. 처음 몇 세대 동안에는 최단거리가 매 세대에서 수시로 변하기 때문에 도시 방문 경로에도 잦은 변화가 생기지만, 세대가 거듭될수록 최단거리를 찾게 되어 이후 방문 경로에는

Collection @ chosun

큰 변화가 생기지 않는다.



▷ 제안한 알고리즘 적용 ◁



▷ 제안한 알고리즘 미적용 ◁

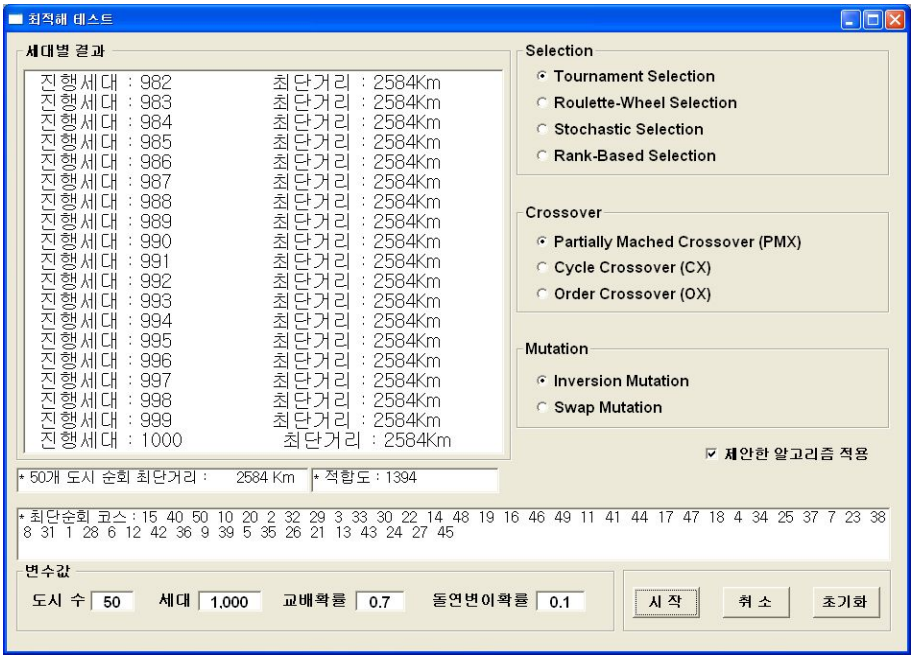
【그림 4-1】 토너먼트 연산자 실험 결과 1

적용 연산자			제안 알고리즘 적용			제안 알고리즘 미적용		
선택	교배	돌연변이	최단거리	세대	적합도	최단거리	세대	적합도
토너먼트	PMX	Inversion	2584	807	1394	3109	998	983

【표 4-1】 토너먼트 연산자 실험 결과 비교 1

Collection @ chosun

표 4-1은 실험 결과를 비교한 표이다. 본 논문에서 제안한 알고리즘을 적용하여 실험했을 경우가 그렇지 않은 경우보다 좀 더 짧은 최단거리를 찾아낸 걸 확인 할 수 있다.



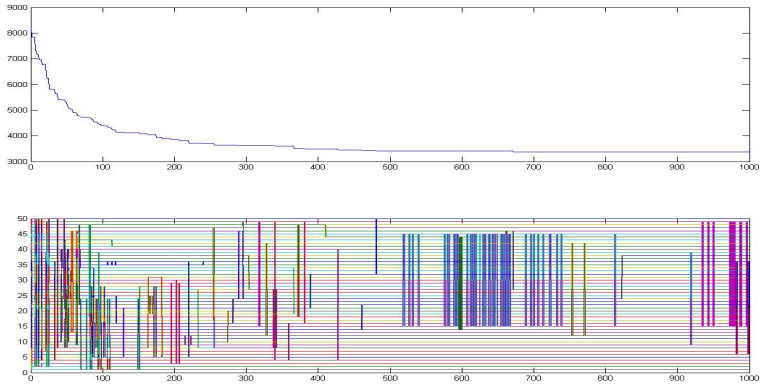
【그림 4-2】 제안한 알고리즘을 적용한 UI 실행 결과 1 (50개 도시)

그림 4-2는 50개의 도시를 대상으로 본 논문에서 제안한 알고리즘을 적용했을 때의 결과를 보여주는 UI화면으로서, 최단거리를 찾았을 때에 해당하는 도시의 순회 순서를 나열하고 있다.

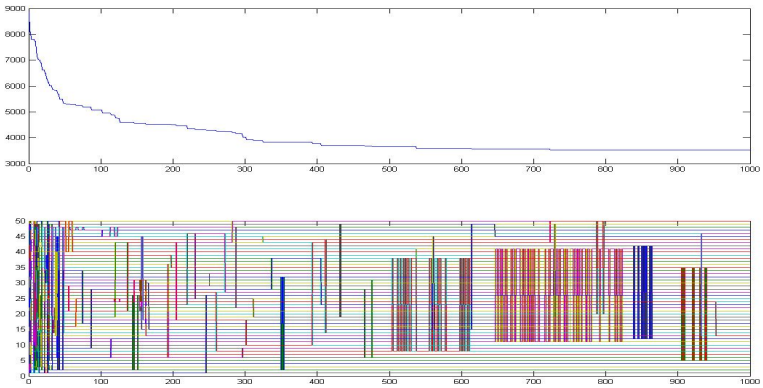
Collection @ chosun

2. 토너먼트(Tournament) - PMX - Swap

다음은 돌연변이 연산자로 Swap연산자를 적용하여 실험한 결과이다. 나머지 변수는 다른 실험들과 동일하다.



▷ 제안한 알고리즘 적용 ◁



▷ 제안한 알고리즘 미적용 ◁

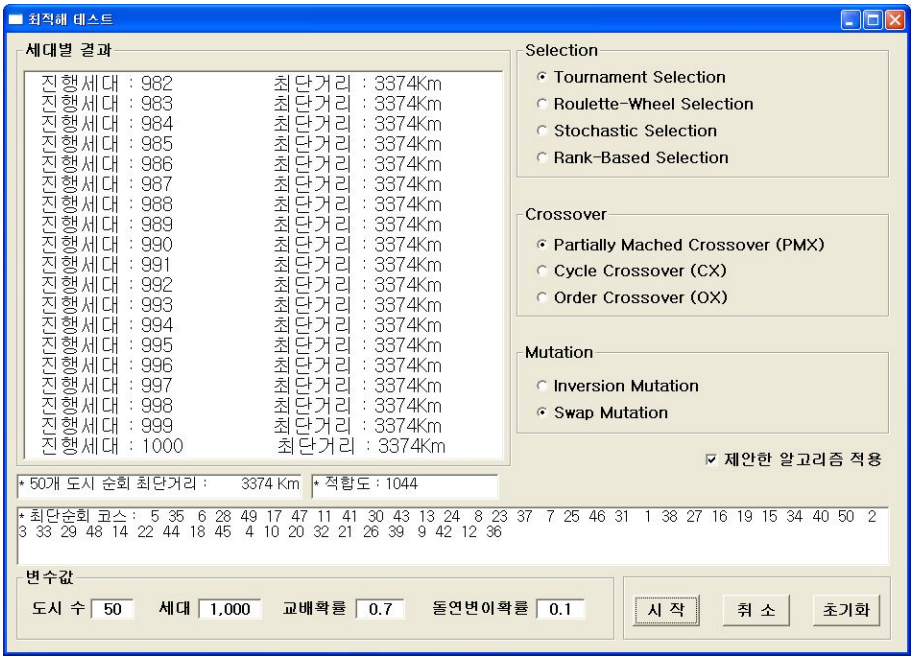
【그림 4-3】 토너먼트 연산자 실험 결과 2

Collection @ chosun

적용 연산자			제안 알고리즘 적용			제안 알고리즘 미적용		
선택	교배	돌연변이	최단거리	세대	적합도	최단거리	세대	적합도
토너먼트	PMX	Swap	3374	824	1044	3525	921	699

【표 4-2】 토너먼트 연산자 실험 결과 비교 2

그림 4-3과 표 4-2는 제안한 알고리즘을 적용한 결과와 적용하지 않은 결과를 각각 보여주고 있다. 이번 실험 결과 역시 본 논문에서 제안한 두 가지 알고리즘을 적용했을 때 보다 우수한 결과를 얻었다.



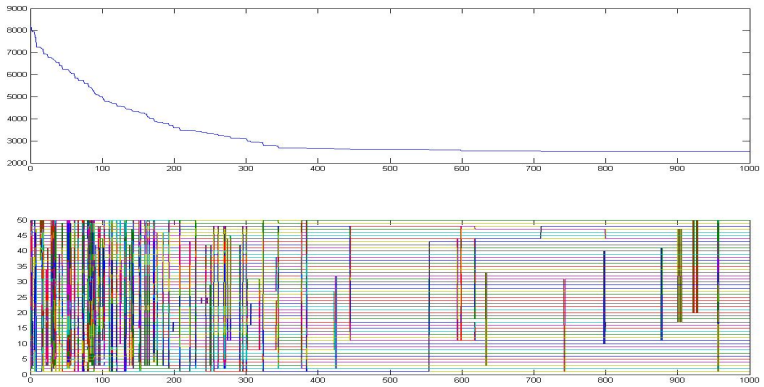
【그림 4-4】 제안한 알고리즘을 적용한 UI 실행 결과 2 (50개 도시)

그림 4-4는 이번 실험에서 제안한 알고리즘을 적용했을 때의 결과를 보여주고 있는 UI이다.

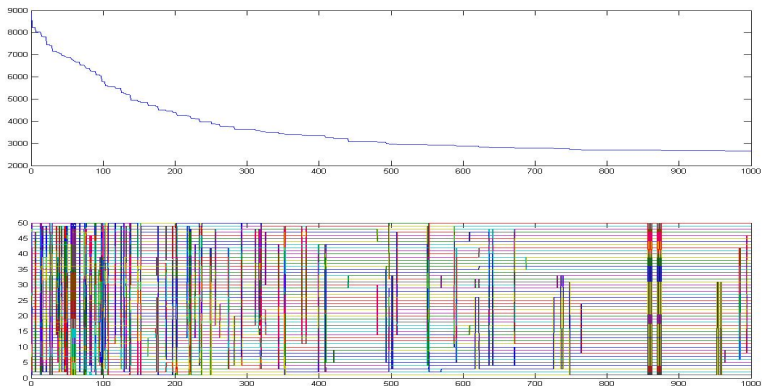
Collection @ chosun

3. 토너먼트(Tournament) - CX - Inversion

다음은 교배 연산자로 싸이클 교배(Cycle Crossover, CX)를 적용하여 실험하였다. 실험 방법은 PMX연산자를 적용하였을 때와 동일한 방법으로 진행하였다.



▷ 제안한 알고리즘 적용 ◁



▷ 제안한 알고리즘 미적용 ◁

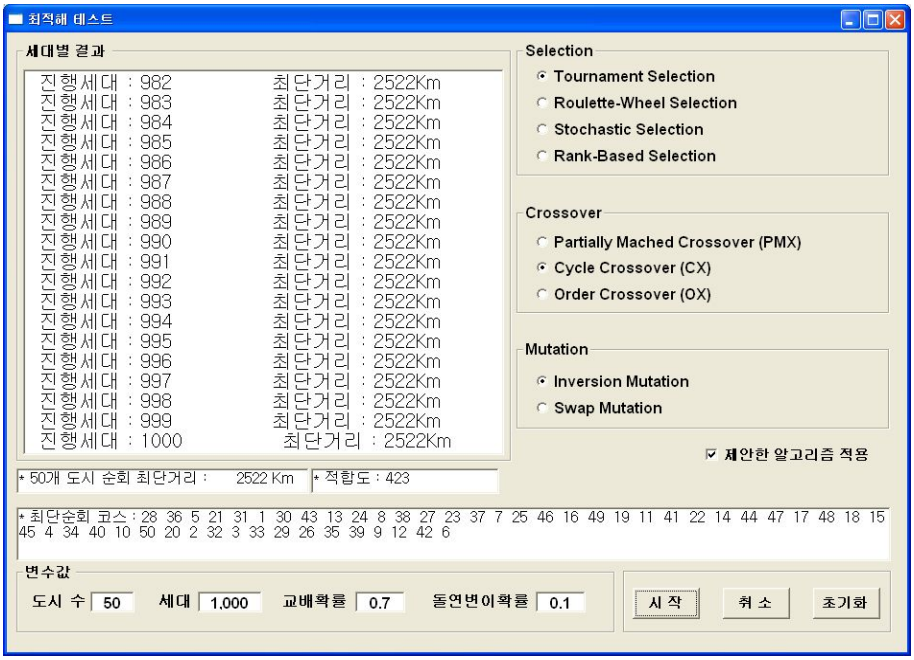
【그림 4-5】 토너먼트 연산자 실험 결과 3

Collection @ chosun

적용 연산자			제안 알고리즘 적용			제안 알고리즘 미적용		
선택	교배	돌연변이	최단거리	세대	적합도	최단거리	세대	적합도
토너먼트	CX	Inversion	2522	801	423	2658	975	586

【표 4-3】 토너먼트 연산자 실험 결과 비교 3

그림 4-5와 표 4-3은 해당 연산자에 대한 실험 결과를 나타내고 있는 것으로, 이번 실험 역시 제안한 알고리즘에서 보다 우수한 결과를 얻을 수 있음을 보여준다.



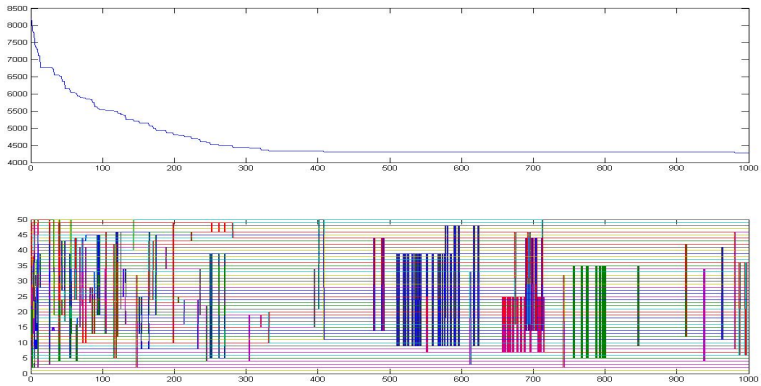
【그림 4-6】 제안한 알고리즘을 적용한 UI 실행 결과 3 (50개 도시)

그림 4-6은 이번 실험에서 제안한 알고리즘을 적용했을 때의 결과를 보여주고 있는 UI이다.

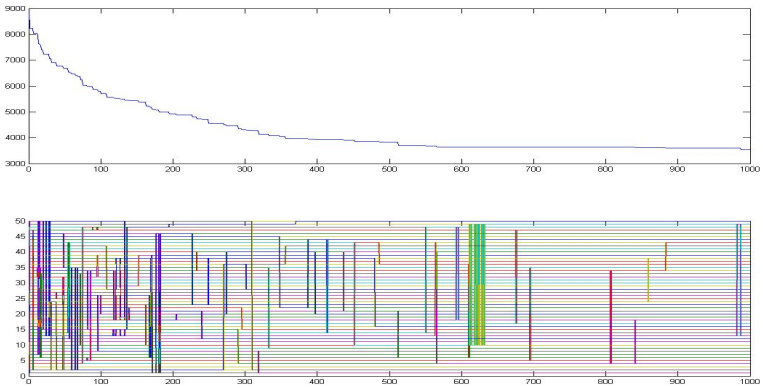
Collection @ chosun

4. 토너먼트(Tournament) - CX - Swap

다음은 돌연변이 연산자로 Swap연산자를 적용하여 실험을 하였다. 이번 실험은 앞의 실험과는 조금 다른 결과가 나왔다. 그림 4-7과 표 4-4의 결과를 보면 지금까지의 실험과는 달리 본 논문에서 제안한 알고리즘을 적용하지 않았을 때보다 좋은 결과가 나왔다는 걸 확인할 수 있다.



▷ 제안한 알고리즘 적용 ◁



▷ 제안한 알고리즘 미적용 ◁

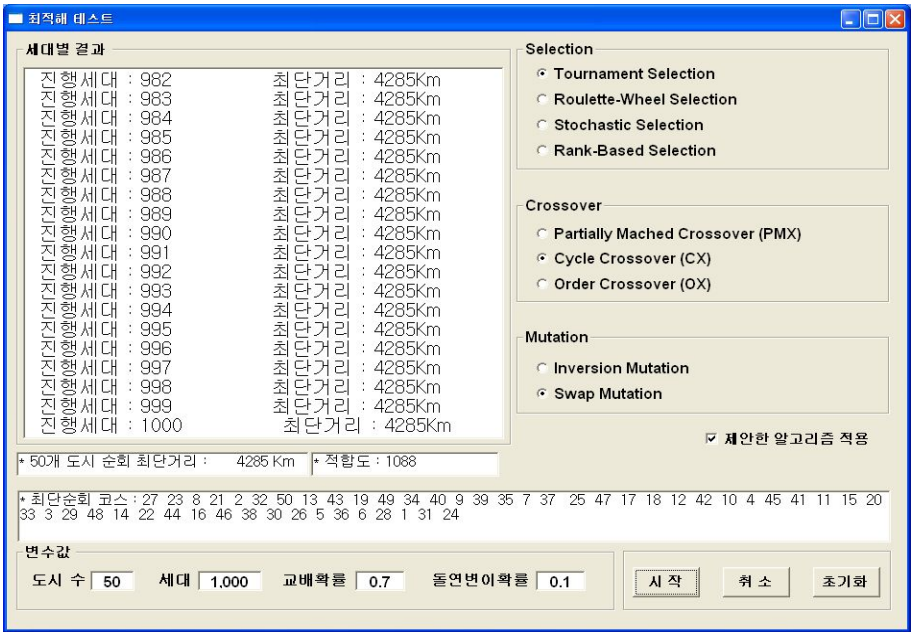
【그림 4-7】 토너먼트 연산자 실험 결과 4

Collection @ chosun

적용 연산자			제안 알고리즘 적용			제안 알고리즘 미적용		
선택	교배	돌연변이	최단거리	세대	적합도	최단거리	세대	적합도
토너먼트	CX	Swap	4285	982	1088	3543	991	993

【표 4-4】 토너먼트 연산자 실험 결과 비교 4

그림 4-7의 제안한 알고리즘의 도시 경로 변화 그래프를 보면 마지막 세대까지 계속해서 도시 경로가 변화되고 있는 걸 알 수 있다. 즉, 최적해를 찾기 위해 최단거리의 변화가 계속해서 발생하고 있는 과정 중에 변수로 정해놓은 1,000세대에 도달하는 바람에 더 이상 진화과정이 이루어지지 않고 종료가 됐다고 볼 수 있다. 이런 경우 한계 세대수를 좀 더 늘려 실험을 할 경우 지금보다 우수한 결과를 얻을 수 있을 것으로 예상해 볼 수 있다.



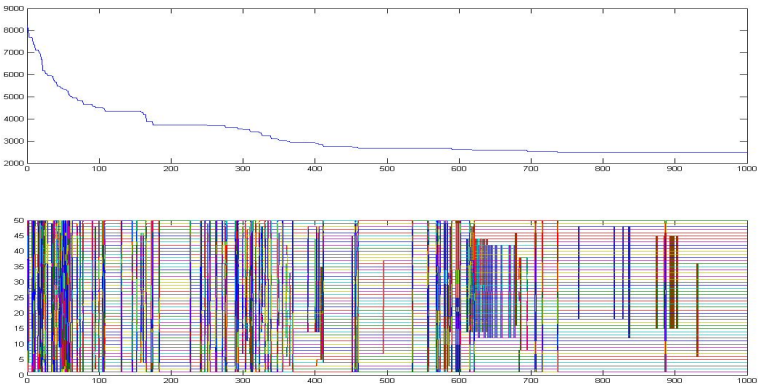
【그림 4-8】 제안한 알고리즘을 적용한 UI 실행 결과 4 (50개 도시)

그림 4-8은 제안한 알고리즘을 적용했을 때의 결과를 보여주는 UI화면이다.

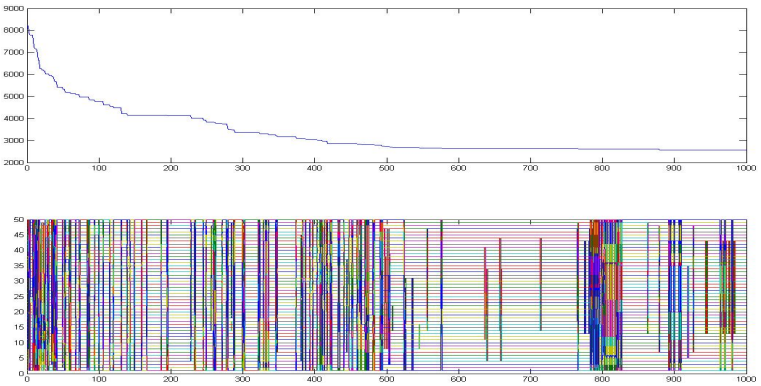
Collection @ chosun

5. 토너먼트(Tournament) - OX - Inversion

다음은 교배 연산자로 순서 교배(Order Crossover, OX)를 적용하여 실험하였다. 그림 4-9와 표 4-5의 결과와 같이 순서 교배 연산자를 사용하였을 경우도 제안한 알고리즘에서 보다 우수한 결과 값을 얻은 걸 확인할 수 있다.



▷ 제안한 알고리즘 적용 ◁



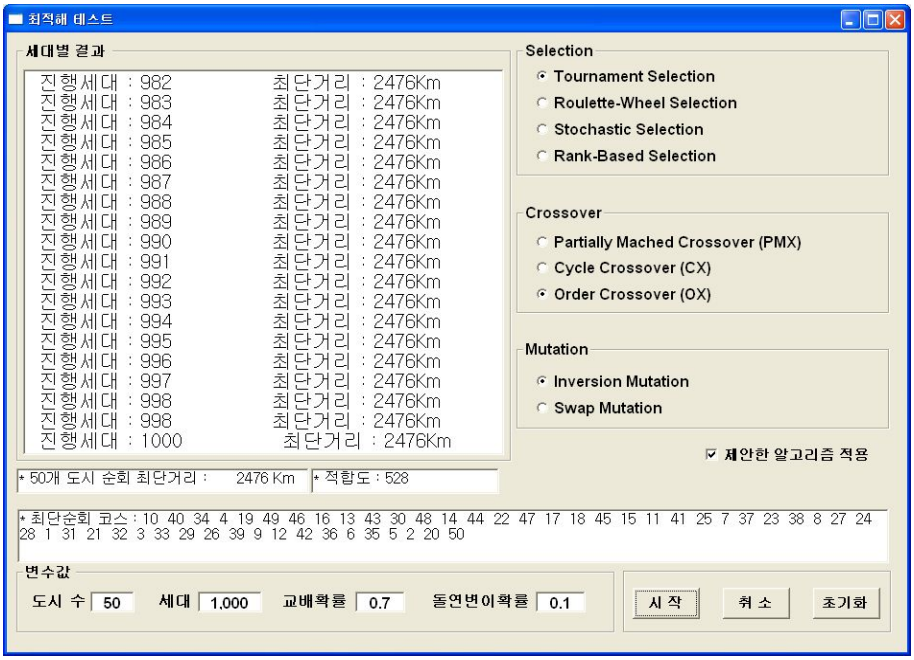
▷ 제안한 알고리즘 미적용 ◁

【그림 4-9】 토너먼트 연산자 실험 결과 5

Collection @ chosun

적용 연산자			제안 알고리즘 적용			제안 알고리즘 미적용		
선택	교배	돌연변이	최단거리	세대	적합도	최단거리	세대	적합도
토너먼트	OX	Inversion	2476	928	528	2505	884	1025

【표 4-5】 토너먼트 연산자 실험 결과 비교 5



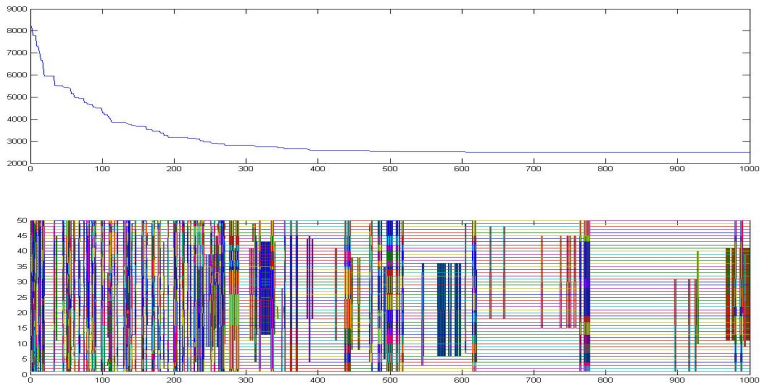
【그림 4-10】 제안한 알고리즘을 적용한 UI 실행 결과 5 (50개 도시)

그림 4-10은 이번 실험에서 제안한 알고리즘을 적용했을 때의 결과를 보여주고 있는 UI이다.

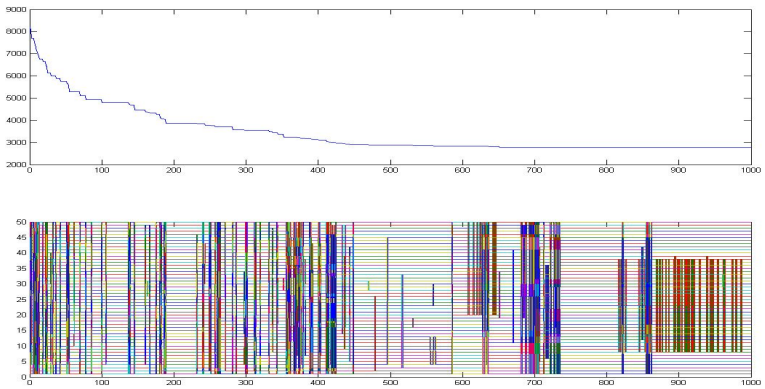
Collection @ chosun

6. 토너먼트(Tournament) - OX - Swap

다음은 돌연변이 연산자로 Swap 연산자를 적용하여 실험한 결과이다.



▷ 제안한 알고리즘 적용 ◁



▷ 제안한 알고리즘 미적용 ◁

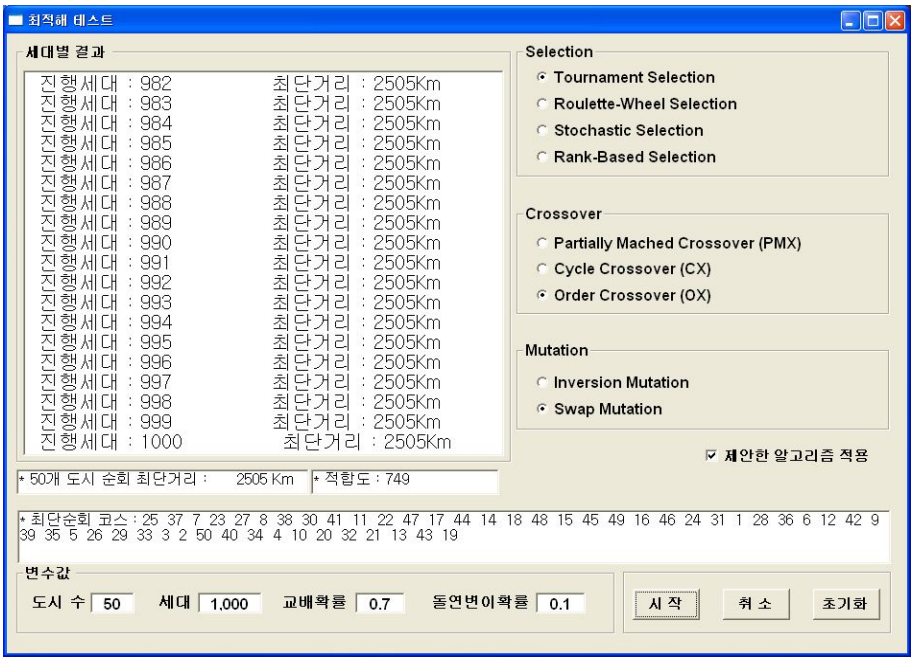
【그림 4-11】 토너먼트 연산자 실험 결과 6

Collection @ chosun

적용 연산자			제안 알고리즘 적용			제안 알고리즘 미적용		
선택	교배	돌연변이	최단거리	세대	적합도	최단거리	세대	적합도
토너먼트	OX	Swap	2505	606	749	2760	655	588

【표 4-6】 토너먼트 연산자 실험 결과 비교 6

그림 4-11과 표 4-6의 결과와 같이 이번 실험 역시 제안한 알고리즘에서 보다 우수한 결과를 얻었다.



【그림 4-12】 제안한 알고리즘을 적용한 UI 실행 결과 6 (50개 도시)

그림 4-12는 이번 실험에서 제안한 알고리즘을 적용했을 때의 결과를 보여주고 있는 UI이다.

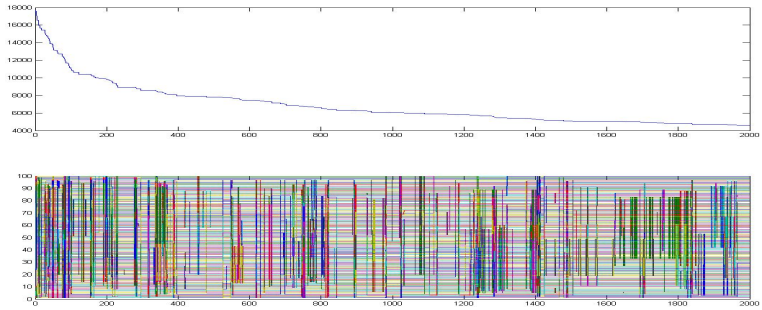
Collection @ chosun

B. 100개 도시 순회 실험

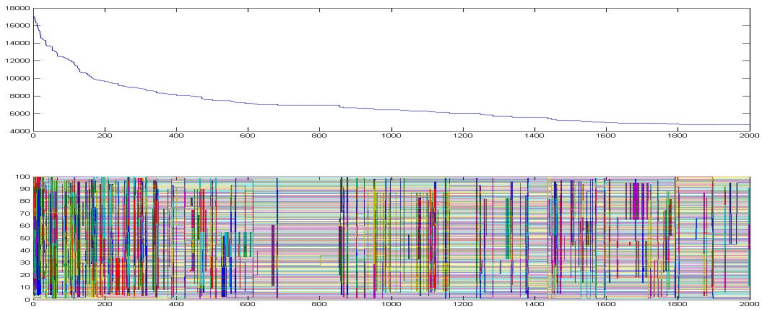
이번 장에서는 100개의 도시를 변수로 하여 실험하였고 그 결과를 비교하였다. 이번 실험에 사용한 연산자 역시 표 3-1과 동일한 연산자를 적용하였고, 총 세대 수는 2,000세대 동안 반복하면서 최단거리를 검색하도록 설계하였다.

1. 룰렛 휠(Roulette-Wheel) - PMX - Inversion

100개 도시를 대상으로 룰렛 휠 선택 연산자와 PMX 교배 연산자 그리고 Inversion 돌연변이 연산자를 적용하여 실험하였다.



▷ 제안한 알고리즘 적용 ◁



▷ 제안한 알고리즘 미적용 ◁

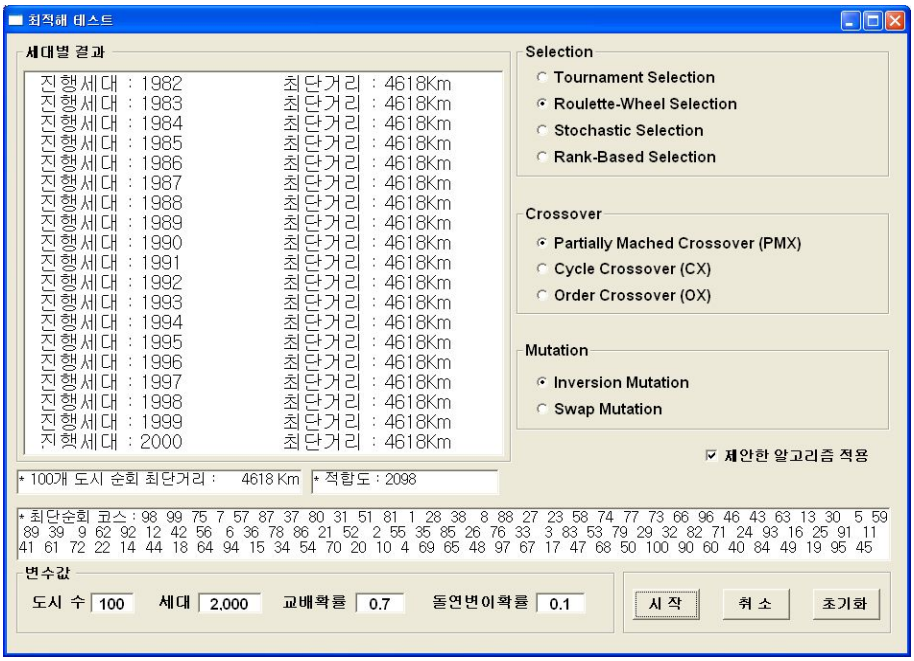
【그림 4-13】 룰렛 휠 연산자 실험 결과 1

Collection @ chosun

적용 연산자			제안한 알고리즘 적용			제안한 알고리즘 미적용		
선택	교배	돌연변이	최단거리	세대	적합도	최단거리	세대	적합도
룰렛 휠	PMX	Inversion	4618	1966	2098	4714	1899	2233

【표 4-7】 룰렛 휠 연산자 실험 결과 비교 1

그림 4-13과 표 4-7의 결과처럼 100개 도시의 경우에도 제안한 알고리즘을 적용했을 때 우수한 결과를 얻은걸 확인할 수 있다.



【그림 4-14】 제안한 알고리즘을 적용한 UI 실행 결과 1 (100개 도시)

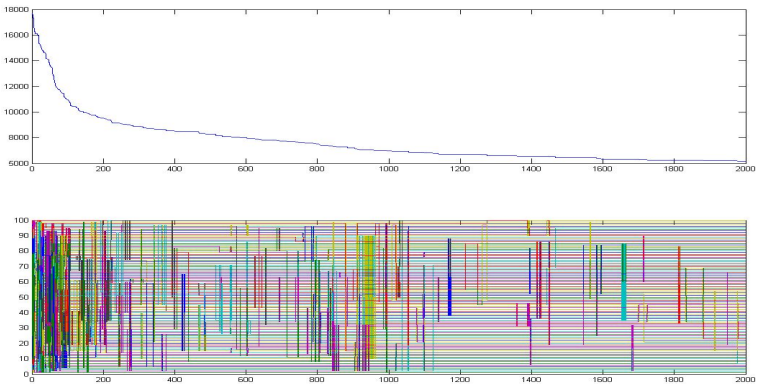
그림 4-14는 100개의 도시를 대상으로 제안한 알고리즘을 적용했을 때의 룰렛 휠-PMX-Inversion 연산자의 최적 결과를 보여주는 UI화면으로서, 최단 거리에 대한 100개의 도시 경로를 보여주고 있다.

Collection @ chosun

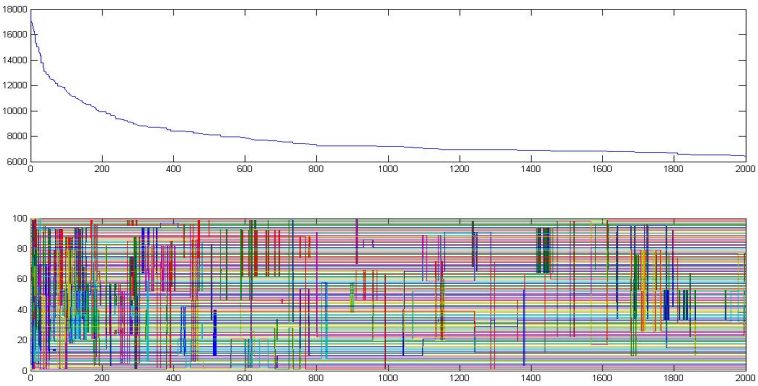
2. 룰렛 휠(Roulette-Wheel) - PMX - Swap

다음은 돌연변이 연산자로 Swap 연산자를 적용했을 때의 실험 결과이다.

이번 결과 역시 본 논문에서 제안한 알고리즘을 적용한 경우에 우수한 결과를 얻었음을 그림 4-15와 표 4-8에서 확인할 수 있다.



▷ 제안한 알고리즘 적용 ◁



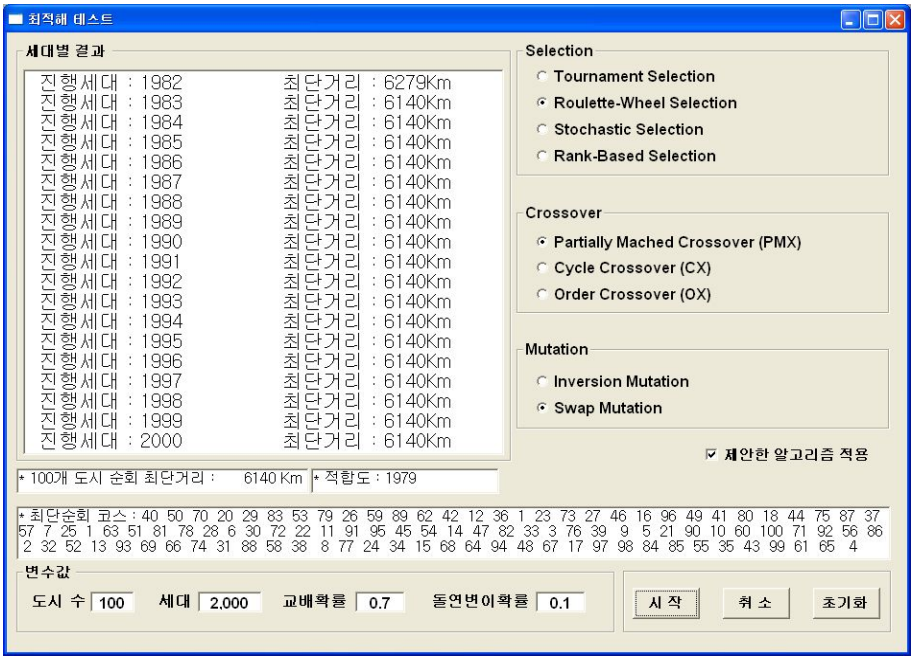
▷ 제안한 알고리즘 미적용 ◁

【그림 4-15】 룰렛 휠 연산자 실험 결과 2

Collection @ chosun

적용 연산자			제안한 알고리즘 적용			제안한 알고리즘 미적용		
선택	교배	돌연변이	최단거리	세대	적합도	최단거리	세대	적합도
룰렛 휠	PMX	Swap	6140	1983	1235	6499	1997	1985

【표 4-8】 룰렛 휠 연산자 실험 결과 비교 2



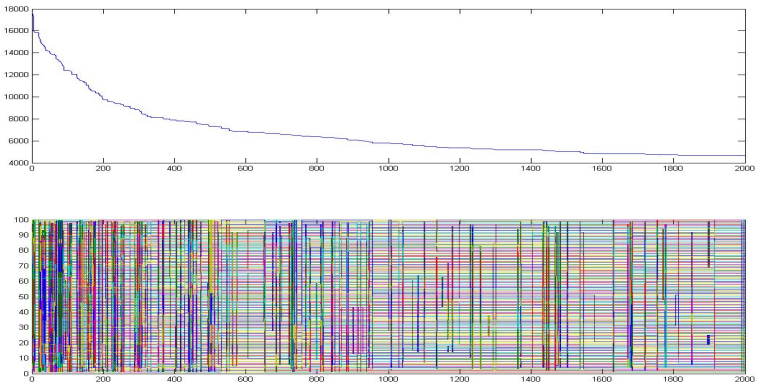
【그림 4-16】 제안한 알고리즘을 적용한 UI 실행 결과 2 (100개 도시)

그림 4-16은 실험 결과에 대한 UI화면을 보여주고 있다.

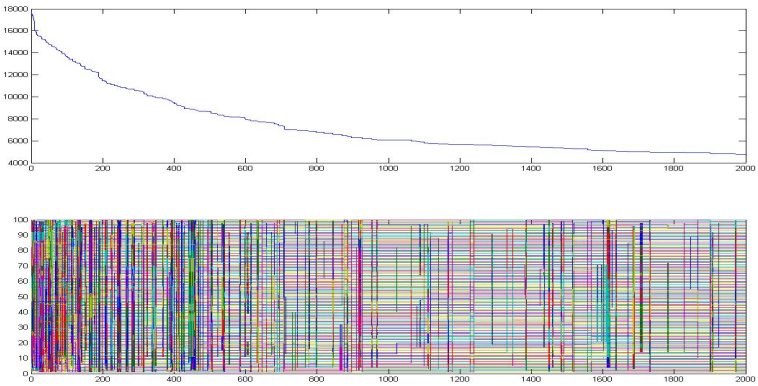
Collection @ chosun

3. 룰렛 휠(Roulette-Wheel) - CX - Inversion

다음은 교배 연산자로 싸이클 교배(Cycle Crossover, CX)를 적용하여 실험하였다. 실험 방법은 PMX연산자를 적용하였을 때와 동일하고 도시 수는 100개, 세대는 2,000세대로 제한하였다.



▷ 제안한 알고리즘 적용 ◁



▷ 제안한 알고리즘 미적용 ◁

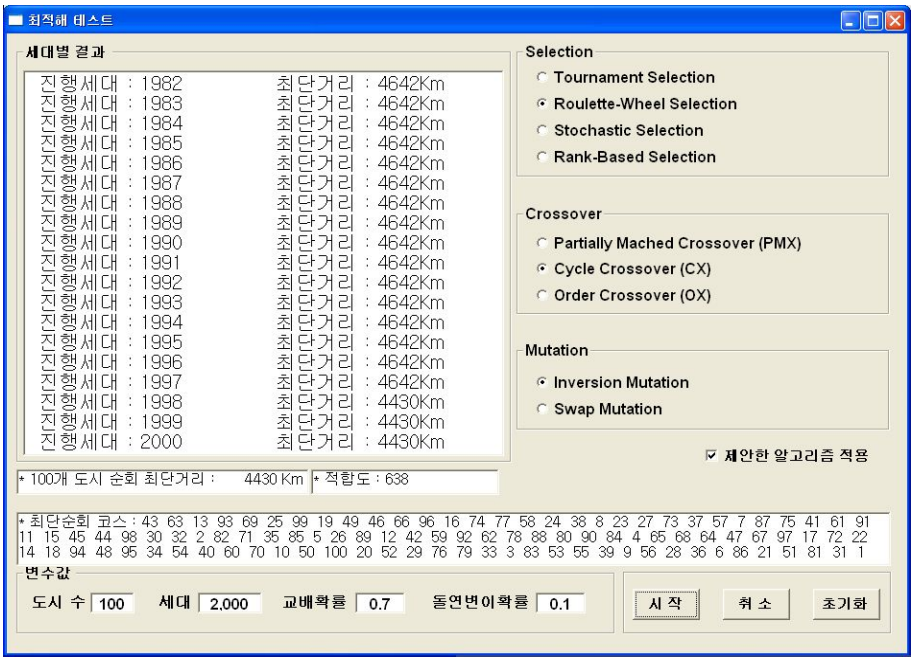
【그림 4-17】 룰렛 휠 연산자 실험 결과 3

Collection @ chosun

적용 연산자			제안한 알고리즘 적용			제안한 알고리즘 미적용		
선택	교배	돌연변이	최단거리	세대	적합도	최단거리	세대	적합도
룰렛 휠	CX	Inversion	4430	1998	638	4772	1973	1021

【표 4-9】 룰렛 휠 연산자 실험 결과 비교 3

교배 연산자로 싸이클 연산자를 적용한 결과 역시 제안한 알고리즘에서 우수한 결과를 얻었다.



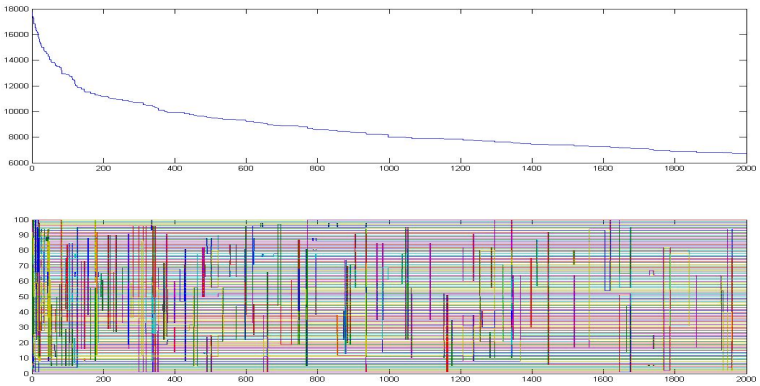
【그림 4-18】 제안한 알고리즘을 적용한 UI 실행 결과 3 (100개 도시)

그림 4-18은 제안한 알고리즘을 적용했을 때의 결과를 보여주고 있는 UI이다.

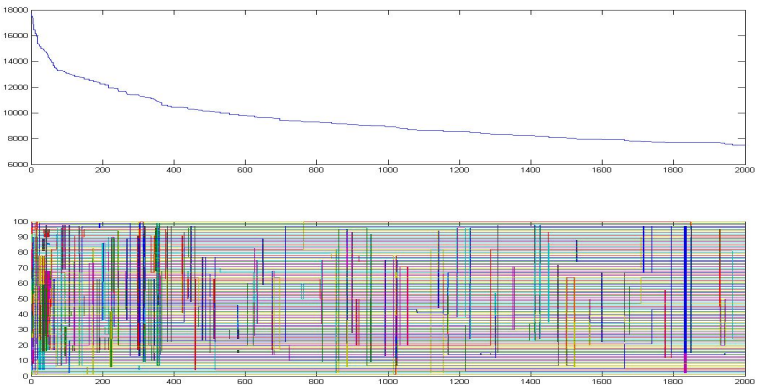
Collection @ chosun

4. 룰렛 휠(Roulette-Wheel) - CX - Swap

교배 연산자로 싸이클 교배, 돌연변이 연산자로 Swap을 적용하여 실험하였다.



▷ 제안한 알고리즘 적용 ◁



▷ 제안한 알고리즘 미적용 ◁

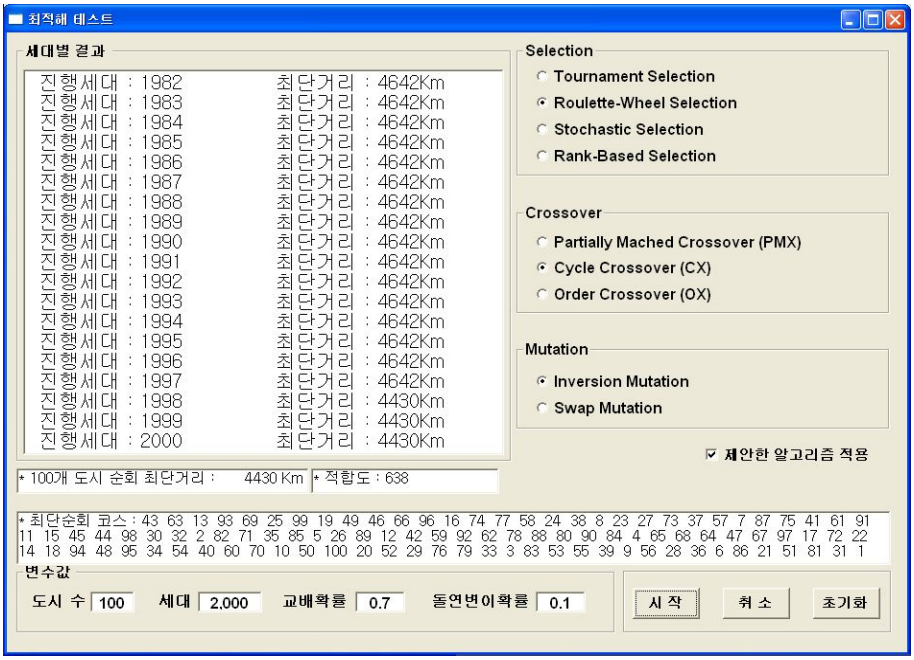
【그림 4-19】 룰렛 휠 연산자 실험 결과 4

dCollection @ chosun

적용 연산자			제안한 알고리즘 적용			제안한 알고리즘 미적용		
선택	교배	돌연변이	최단거리	세대	적합도	최단거리	세대	적합도
룰렛 휠	CX	Swap	6751	1961	1479	7514	1966	2210

【표 4-10】 룰렛 휠 연산자 실험 결과 비교 4

그림 4-19와 표 4-10과 같이 이번 실험 역시 제안한 알고리즘에서 보다 더 우수한 결과를 얻었다.



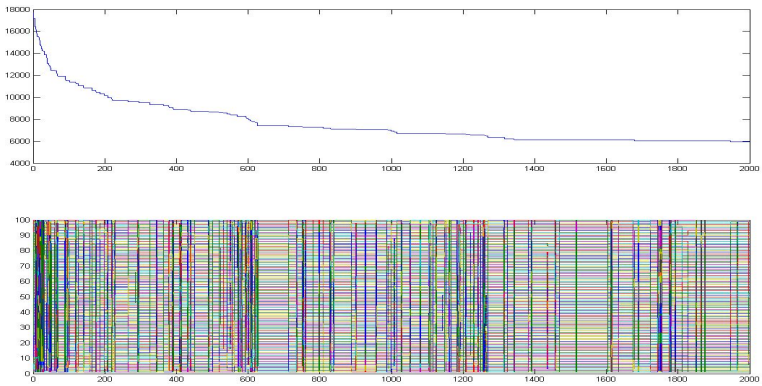
【그림 4-20】 제안한 알고리즘을 적용한 UI 실행 결과 4 (100개 도시)

그림 4-20은 제안한 알고리즘을 적용했을 때의 결과를 보여주고 있는 UI이다.

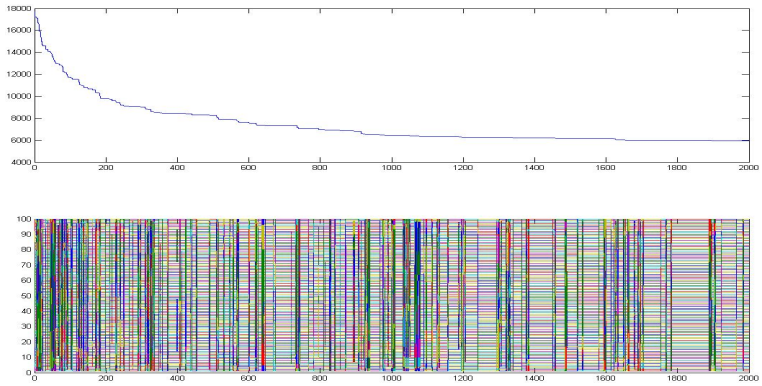
Collection @ chosun

5. 룰렛 휠(Roulette-Wheel) - OX - Inversion

다음은 교배 연산자로 순서 교배(Order Crossover, OX) 연산자를 사용하여 실험하였다. 그림 4-21과 표 4-11에서와 같이 이번 실험 결과 역시 제안한 알고리즘에서 우수한 결과를 얻었다.



▷ 제안한 알고리즘 적용 ◁



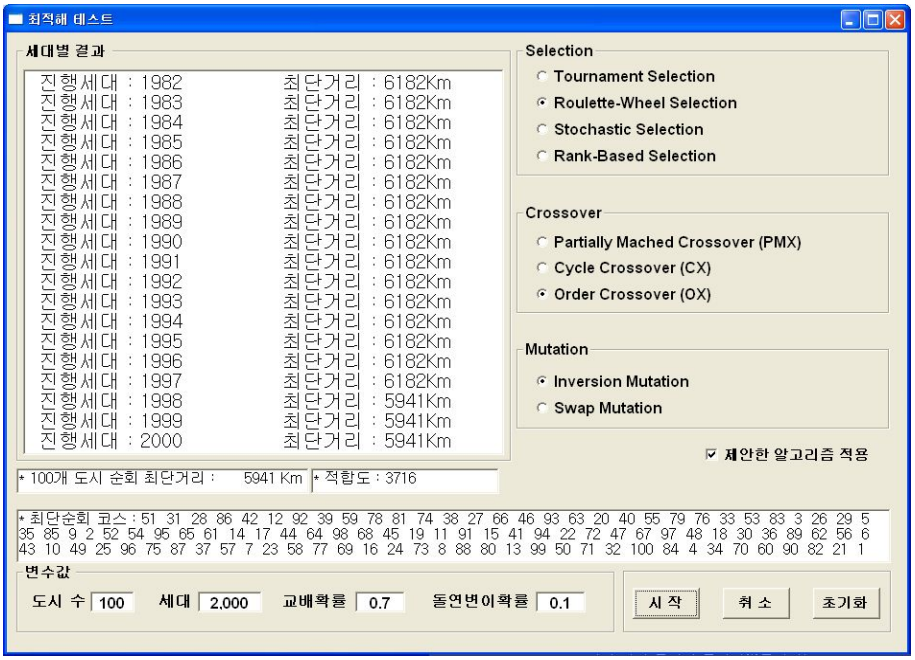
▷ 제안한 알고리즘 미적용 ◁

【그림 4-21】 룰렛 휠 연산자 실험 결과 5

Collection @ chosun

적용 연산자			제안한 알고리즘 적용			제안한 알고리즘 미적용		
선택	교배	돌연변이	최단거리	세대	적합도	최단거리	세대	적합도
룰렛 휠	OX	Inversion	5941	1998	3716	5953	1897	1010

【표 4-11】 룰렛 휠 연산자 실험 결과 비교 5



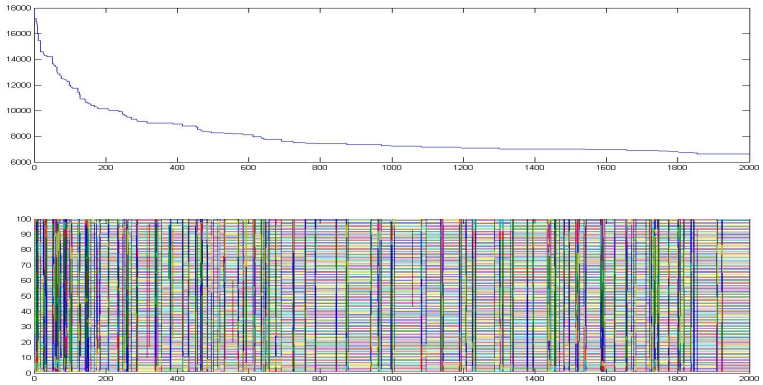
【그림 4-22】 제안한 알고리즘을 적용한 UI 실행 결과 5 (100개 도시)

그림 4-22는 제안한 알고리즘을 적용했을 때의 결과를 보여주고 있는 UI이다.

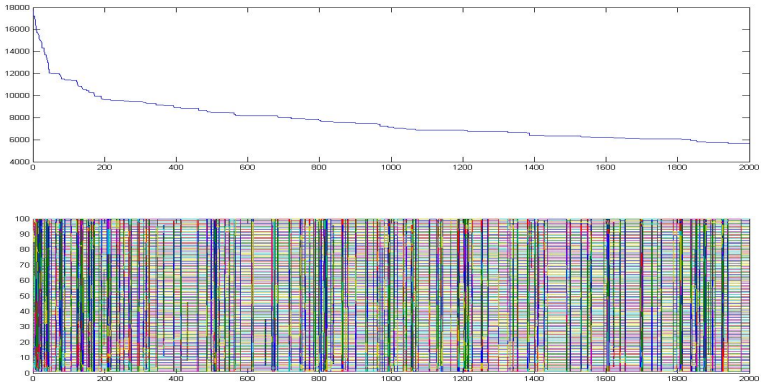
Collection @ chosun

6. 룰렛 휠(Roulette-Wheel) - OX - Swap

다음은 돌연변이 연산자로 Swap 연산자를 사용하여 실험하였다.



▷ 제안한 알고리즘 적용 ◁



▷ 제안한 알고리즘 미적용 ◁

【그림 4-23】 룰렛 휠 연산자 실험 결과 6

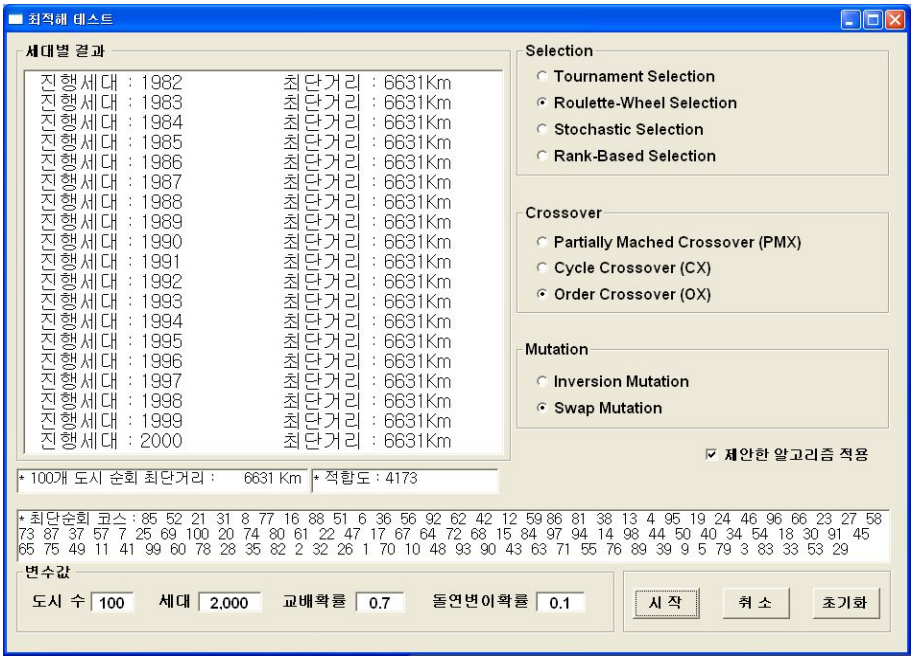
그림 4-23과 표 4-12에서 나타나 있듯 이번 실험에서는 앞선 결과와는 달리 본 논문에서 제안한 알고리즘을 적용했을 때 좋지 않은 결과가 나왔다.

Collection @ chosun

적용 연산자			제안한 알고리즘 적용			제안한 알고리즘 미적용		
선택	교배	돌연변이	최단거리	세대	적합도	최단거리	세대	적합도
룰렛 휠	OX	Swap	6631	1911	4173	5670	1942	3321

【표 4-12】 룰렛 휠 연산자 실험 결과 비교 6

하지만 룰렛 휠-OX-Swap 연산자 조합의 실험 결과가 타 연산자 조합의 실험결과 보다 최단거리가 상대적으로 좋지 않은 결과를 보이고 있는 만큼 이번 결과는 큰 의미가 없다고 봐도 무관하다.



【그림 4-24】 제안한 알고리즘을 적용한 UI 실행 결과 6 (100개 도시)

그림 4-24는 제안한 알고리즘을 적용했을 때의 결과를 보여주고 있는 UI이다.

Collection @ chosun

C. 실험에 대한 평가

지금까지 50개의 도시와 100개의 도시를 대상으로 최적의 TSP 해를 구하기 위한 실험을 진행하였다. 4개의 선택 연산자와 3개의 교배 연산자, 2개의 돌연변이 연산자의 조합으로 본 논문에서 제안한 혼합초기화법과 순차변환 기법을 적용한 경우와 적용하지 않은 경우에 대해 각각 24번, 총 48번의 실험을 실시하였다.

먼저 50개의 도시를 변수로 하여 실험한 결과 값은 표 4-13와 같다. 총 세대수와 집단의 크기는 1,000으로 하였고 교배 확률(P_c)과 돌연변이 확률(P_m)은 0.7과 0.1을 사용하였다.

총 24번의 실험 중 토너먼트-CX-Swap 연산자 조합과 룰렛 휠-OX-Inversion 연산자 조합을 제외한 22번의 실험에서 본 논문이 제안한 알고리즘을 적용했을 때 보다 우수한 결과를 얻었다.

이 중 토너먼트-CX-Swap 연산자 조합과 룰렛 휠-OX-Inversion 연산자 조합에서는 좋은 결과를 얻지 못했지만, 두 실험에서 최적의 해로 얻어진 값이 나머지 22번의 평균값에도 못 미칠 정도의 결과이기 때문에 토너먼트-CX-Swap 연산자 조합과 룰렛 휠-OX-Inversion 연산자 조합의 실험 결과에는 큰 의미를 두지 않아도 될 것으로 판단된다.

표 4-14은 100개의 도시를 변수로 사용하였을 경우에 실험 결과를 보여주는 표이다. 실험을 위한 변수는 50개 도시일 경우와 동일하며, 단 총 세대수를 2,000세대로 하였다.

이번 실험 역시 총 24번 진행을 하였고, 이중 토너먼트-CX-Swap 연산자 조합과 룰렛 휠-OX-Swap 연산자 조합, 확률균등-CX-Swap 연산자 조합에서 제안한 알고리즘을 적용했을 때 좋지 않은 결과가 나왔다.

하지만 이번 결과 역시 제안한 알고리즘에서 좋은 결과를 얻은 나머지 21번의

Collection @ chosun

평균값에도 못 미치는 결과를 보여주고 있기 때문에 의미를 두지 않아도 될 것이다.

dCollection @ chosun

적용 연산자			제안한 알고리즘 적용			제안한 알고리즘 미적용		
선택	교배	돌연변이	최단거리	세대	적합도	최단거리	세대	적합도
토너먼트	PMX	Inversion	2584	807	1394	3109	998	983
		Swap	3374	824	1044	3525	921	699
	CX	Inversion	2522	801	423	2658	975	586
		Swap	4285	982	1088	3543	991	993
	OX	Inversion	2476	928	528	2505	884	1025
		Swap	2505	606	7249	2760	655	588
룰렛 휠 선택	PMX	Inversion	2398	950	695	2458	869	1021
		Swap	3255	904	1201	3646	992	1001
	CX	Inversion	2596	955	611	2681	892	894
		Swap	3692	908	1426	3712	978	1532
	OX	Inversion	3079	878	2618	3031	848	1125
		Swap	2936	952	3558	3253	829	2689
확률 균등 선택	PMX	Inversion	2770	986	1462	2878	681	509
		Swap	3056	962	1218	3196	711	1322
	CX	Inversion	2814	984	1062	2822	946	998
		Swap	3591	980	924	3669	976	856
	OX	Inversion	3223	894	2822	3398	984	3003
		Swap	3126	890	3318	3568	926	2564
순위 기반 선택	PMX	Inversion	2439	774	421	2480	997	1089
		Swap	3095	957	598	3106	990	754
	CX	Inversion	2399	945	721	2420	891	645
		Swap	3164	898	788	3327	971	685
	OX	Inversion	2443	990	746	2560	849	886
		Swap	2554	365	1231	2788	831	1587

【표 4-13】 50개 도시 실험 결과

Collection @ chosun

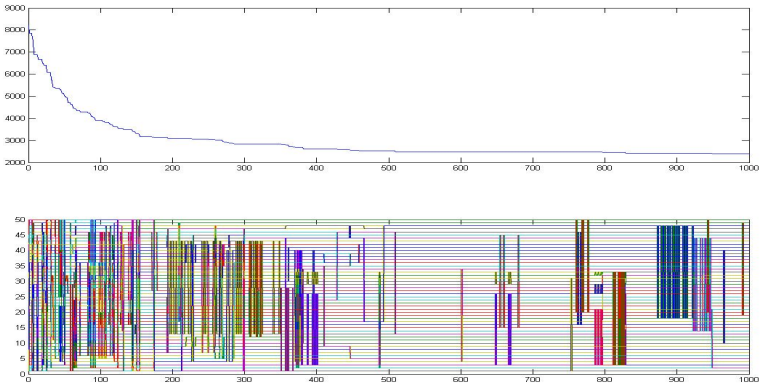
적용 연산자			제안한 알고리즘 적용			제안한 알고리즘 미적용		
선택	교배	돌연변이	최단거리	세대	적합도	최단거리	세대	적합도
토너먼트	PMX	Inversion	3566	1994	1056	3797	1988	951
		Swap	5442	1998	1222	5737	1863	983
	CX	Inversion	3675	1982	497	4086	1924	1003
		Swap	6095	1952	839	6021	1910	2543
	OX	Inversion	4071	1991	1010	4223	1989	754
		Swap	3962	1666	1336	4574	1988	2254
룰렛 휠 선택	PMX	Inversion	4618	1966	2098	4714	1899	2233
		Swap	6140	1983	1235	6499	1997	1985
	CX	Inversion	4430	1998	638	4772	1973	1021
		Swap	6751	1961	1479	7514	1966	2210
	OX	Inversion	5941	1998	3716	5953	1879	1010
		Swap	6631	1911	4173	5670	1942	3321
확률 균등 선택	PMX	Inversion	4878	1943	2613	5074	1979	1209
		Swap	5919	1988	876	6081	1990	620
	CX	Inversion	4507	1913	751	5019	1971	2505
		Swap	7382	1992	1094	7210	1979	2234
	OX	Inversion	5731	1956	2284	6705	1982	1876
		Swap	6194	1974	3573	6729	1909	937
순위 기반 선택	PMX	Inversion	3822	1916	693	3862	1951	995
		Swap	5586	1975	922	7029	1970	896
	CX	Inversion	3759	1812	671	4000	1885	1050
		Swap	5376	1929	2267	6001	1974	1451
	OX	Inversion	4161	1836	2601	5714	1894	3009
		Swap	4057	1892	1362	4267	1828	715

【표 4-14】 100개 도시 실험 결과

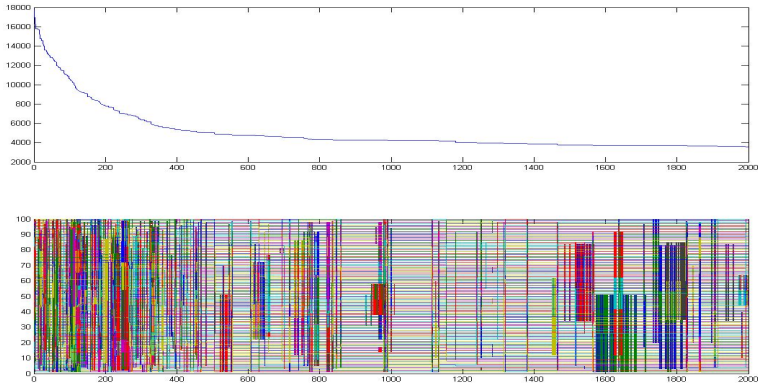
Collection @ chosun

실험 결과에서 알 수 있듯이 초기 집단을 생성함에 있어서 특별한 규칙 없이 집단을 생성하는 무작위 초기화법 보다는 사전에 알고 있는 도시들 간의 거리 정보를 활용하여 초기 집단을 생성하는 혼합 초기화법을 적용하였을 때 우수한 결과를 얻었다.

또한, 본 논문에서 제안한 또 다른 알고리즘인 순차변환 기법을 각 세대에 적용하여 우성인자들의 선택 확률을 높이는 것이 최적의 TSP 해를 구하는데 보다 효과적임을 실험을 통해 증명하였다.



【그림 4-25】 룰렛 휠-PMX-Inversion 연산자 조합 (50개 도시)



【그림 4-26】 토너먼트-PMX-Inversion 연산자 조합 (100개 도시)

Collection @ chosun

그림 4-25와 그림 4-26은 50개 도시와 100개 도시를 변수로 하여 제안한 알고리즘을 적용하여 실험한 결과 중 가장 우수한 결과를 나타낸 연산자 조합을 보여주고 있다.

50개 도시의 경우 룰렛 휠-PMX-Inversion 연산자 조합이 가장 우수한 결과를 보였고, 100개 도시의 경우엔 토너먼트-PMX-Inversion 연산자 조합이 가장 우수한 결과를 나타냈다.

Collection @ chosun

Ⅴ. 결론 및 향후 과제

정보통신의 발달과 함께 지리정보시스템(GIS), 대중교통망 설계, 네트워크 최적화, 택배시스템 등 최단거리 설정에 관한 연구가 활발히 진행되어 왔다. 이러한 최단거리 설정의 문제는 TSP에 근간을 두고 있는 경우가 대부분이다.

TSP란 주어진 N개의 도시와 각 도시 사이의 거리가 주어질 때, 임의의 한 도시에서 출발하여 모든 도시를 단 한 번만 방문하고 원래의 출발지로 되돌아오는 경로 중 최단거리를 찾는 문제이다. 하지만 도시의 수가 증가하게 되면 최적해를 찾는 계산량이 기하급수적으로 증가하기 때문에 NP-hard 문제로 분류되어 왔다.

이러한 문제를 해결하기 위해 John Holland에 의해 제안된 유전자 알고리즘은 TSP 최적의 해를 계산하기 위한 가장 대표적인 알고리즘으로 이용되고 있다. 자연계의 적자생존의 개념을 모델링한 유전자 알고리즘은 주어진 문제에 대해 조합 가능한 해의 영역을 병렬적으로 조사한다는 장점이 있다.

본 논문에서는 유전자 알고리즘을 이용하여 TSP 최적해를 찾고자 할 때 보다 효과적인 최적해를 찾을 수 있는 두 가지 방법을 제안하였다. 하나는 집단을 초기화함에 있어서 무작위 초기화법과 유도된 초기화법을 동시에 사용하는 혼합 초기화법이고, 다른 하나는 각 세대에서 선택 연산자가 적용되기 전에 개체 집단을 재배열하여 우성인자끼리 선택될 확률을 높여 주는 순차변환 기법이다.

TSP 최적해를 구하기 위해 유전자 알고리즘을 적용하는 데 있어 초기 집단 생성은 가장 중요한 요소 중 하나이다. 최초 집단이 어떻게 생성이 되느냐가 세대를 거듭해가며 최적해를 구해가는 과정에 있어서 중요한 변수이기 때문이다. 지금까지는 일반적으로 무작위 초기화법을 이용하여 초기 집단을 생성하였으나, 본 논문에서는 무작위 초기화법과 사전에 알고 있는 도시들 간의 거리 정보를 활용하는 유도된 초기화법을 동시에 적용하는 혼합 초기화법을 제안하였다.

Collection @ chosun

또한 각 세대에서 선택 연산자를 적용하기 직전에 개체 집단을 인위적으로 재배열하여 다음 세대에 우성인자끼리 선택될 확률을 높여줌으로써 세대가 거듭해갈수록 우수한 집단을 생성하도록 순차변환 기법을 제안하고 실험에 적용하였다.

제안한 알고리즘의 성능을 검증하기 위해 50개의 도시와 100개의 도시를 변수로 하여 각각 실험하였으며, 그 결과 50개 도시의 경우 평균 3.86%의 성능 개선 결과가 나타났고, 100개 도시의 경우에는 평균 6.97%의 성능 개선 결과가 나타났다.

즉, 사전에 이미 알고 있는 각 도시의 거리정보를 활용하여 집단을 초기화하는 방법과 개체를 재배열함으로써 우성인자끼리 선택될 확률을 높여주는 방법이 TSP 최적해를 구함에 있어서 성능 개선에 효과적임을 확인하였다. 또한 도시의 수가 더 많은 경우에 보다 우수한 성능 개선 효과를 나타냈다.

실험 결과에서 알 수 있듯이 TSP에 근간을 두고 있는 분야에서는 본 논문이 제안한 알고리즘을 다양한 방법으로 활용하여, 보다 빠르고 정확하게 최적해를 구할 수 있을 것이다.

정보통신 산업의 눈부신 발전으로 인해 사회 전반에 걸쳐 최단거리의 문제, 최소비용의 문제 등 TSP에 근간을 두고 있는 분야가 다양하게 생겨나고 있다.

본 논문에서 제안한 알고리즘을 향후 TSP 관련 산업에 효과적으로 적용하기 위해서는 현재도 꾸준히 연구되고 있는 유전자 알고리즘의 새로운 연산자와의 적용과 대규모 도시 또는 좀 더 복잡한 구조를 가진 환경을 대상으로 하는 추가적인 연구가 지속적으로 진행되어야 할 것이다.

Collection @ chosun

참 고 문 헌

[1] Schwefel, H.-P, "Numerical Optimization for Computer Models", John Wiley, Chichester, UK, 1981.

[2] 진강규. “유전알고리즘과 그 응용”, 교우사, 2002.

[3] C-Y. Lee, "Genetic Algorithms with Entropy-Boltzmann Samplings", Proceedings Congress on Evolutionary Computation, vol.1, 2001.

[4] 김대수. “신경망 이론과 응용(1)”. 하이테크 정보사. pp.211-223. 1992.

[5] Guarino, N., “Formal Ontology and Information Systems,” Proceedings of FOIS'98, 1998.

[6] Berners-Lee T., Hendler J. and Lassila, O., The Semantic Web, Scientific American, 2001.

[7] J. H. Holland, Adaptation In Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, 1975.

[8] Goldberg, D. E, "Genetic Algorithms in search, optimization, and machine learning", Addison Wesley, Reading, Massachusetts, 1989.

[9] Hue, X., "Genetic algorithm for Optimization: Background and Applications", Edinburgh Parallel Computing Centre The University of Edinburgh, Ver 1.0, 1997.

[10] 이건창, “사례기반추론과 유전자 알고리즘을 결합한 지식경영 방법론에 관한 연구: 신용평가를 중심으로”, 정보기술응용연구창간호, 1999.

[11] Lawer. E. L, Lenstra, J.K., Rinnooy Kan, A.H.G., and Shmoys, D.B, The Traveling Salesman Problem, John Wiley, Chichester, UK, 1985.

[12] Z. Michalewicz, "Genetic Algorithm + Data Structure = Evolution Programs", Springer-Verlag, Berlin Heidelberg, 1996.

Collection @ chosun

[13] 정은영, “고분자 제품의 최적 생산순서 결정을 위한 TSP의 적용과 생산 계획”, 2004.

[14] 이용권, “유전자 알고리즘을 이용한 일회용 인증 암호키 설계와 구현”, 강원대학교 박사학위논문, 2007.

[15] 문병로, “유전알고리즘”, 두양사, 2003.

[16] M. Gen and R. Cheng, Genetic Algorithms & Engineering Design, John Wiley & Sons, Inc., N.Y., 1997.

[17] 김준민, “생물학사전”, 창원출판사, 1986.

[18] 정재학, “우수유출 저류지 설계를 위한 다목적 유전자 알고리즘의 적용”, 경북대학교 박사학위논문, 2008.

[19] Kenneth A. DeJung, An analysis of the behavior of a class of genetic adaptive systems(Doctorial dissertation, University of Michigan), Dissertation Abstracts International 36(10), University of Michigan, Ann Arbor, 1975.

[20] D. Goldberg and J. Richardson, "Genetic algorithms whit sharing for multimodel function optimization," *International Conference on Genetic Algorithms*, 1987.

[21] Darwen and Yao, "Speciation automatic categorical modularization", *IEEE Trans. on Evolutionary Computation*, 1997.

[22] Sareni and Krahenbuhl, "Fitness sharing and niching methods revisited", *IEEE Trans. on Evolutionary Computation*, 1998.

[23] G. Syswerda, "Uniform crossover in genetic algorithm", *International Conference on Genetic Algorithms*, 1989.

[24] I. Oliver, D. Smith, and J.Holland, "A study of permutation crossover operators on the traveling salesman problem", *International Conference*

Collection @ chosun

on Genetic Algorithms, 1987.

[25] L. Davis, "Applying adaptive algorithm with epistatic domains", *International Joint Conference on Genetic Artificial Intelligence*, 1985.

[26] D. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem", *International Conference on Genetic Algorithms*, 1985.

[27] Z. Michalewicz, *"Genetic Algorithms + Data Structures = Evolutionary Programs"*, Springer, 1992.

[28] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, and C. Whitley, "A comparison of genetic sequencing operators", *International Conference on Genetic Algorithms*, 1991.

[29] J. Cohoon and W. Paris, "Genetic placement", *IEEE Trans. on Computer-Aided Design*, 1987.

[30] C. Anderson, K. Jones, and J. Ryan, “A two-dimesional genetic algorithm for the Ising problem", *Complex Systems*, 1991.

[31] A. Kang and B. Moon, "Toward more powerful recombinations", *International Conference on Genetic Algorithms*, 1995.

[32] S. Jung and B. Moon, "The natural crossover for the 2D Euclidean TSP", *Genetic and Evolutionary Computation Conference*, 2000.

[33] S. Jung and B. Moon, "Toward minimal restriction of genetic encoding and crossovers for the two-dimensional Educlidean TSP", *IEEE Trans. on Evolutionary Computation*, 2002.

dCollection @ chosun

부록 1 - 50개 도시 거리 정보

도시	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	107	241	190	124	80	316	76	152	157	283	133	113	297	228
2	107	0	148	137	88	127	336	183	134	95	254	180	101	234	175
3	241	148	0	374	171	259	509	317	217	232	491	312	280	391	412
4	190	137	374	0	202	234	222	192	248	42	117	287	79	107	38
5	124	88	171	202	0	61	392	202	46	160	319	112	163	322	240
6	80	127	259	234	61	0	386	141	72	167	351	55	157	331	272
7	316	336	509	222	392	386	0	233	438	254	202	439	235	254	210
8	76	183	317	192	202	141	233	0	213	188	272	193	131	302	233
9	152	134	217	248	46	72	438	213	0	206	365	89	209	368	286
10	157	95	232	42	160	167	254	188	206	0	159	220	57	149	80
11	283	254	491	117	319	351	202	272	365	159	0	404	176	106	79
12	133	180	312	287	112	55	439	193	89	220	404	0	210	384	325
13	113	101	280	79	163	157	235	131	209	57	176	210	0	186	117
14	297	234	391	107	322	331	254	302	368	149	106	384	186	0	69
15	228	175	412	38	240	272	210	233	286	80	79	325	117	69	0
16	129	176	349	121	232	226	187	98	278	132	161	279	75	191	122
17	348	265	422	152	314	362	313	344	360	193	165	415	231	59	122
18	276	199	356	86	287	296	266	289	333	127	141	349	165	35	56
19	188	182	355	68	238	232	154	177	284	100	95	285	81	125	56
20	150	67	204	70	155	164	282	216	201	28	187	217	85	167	108
21	65	42	182	137	65	85	321	141	111	95	254	138	92	255	175
22	341	278	435	151	366	375	298	346	412	193	103	428	230	44	113
23	184	271	417	239	300	249	168	108	321	241	279	310	184	309	240
24	67	146	292	135	175	147	249	57	221	131	215	200	74	245	176
25	221	251	424	137	307	301	95	190	353	169	117	354	150	169	125
26	169	105	116	242	57	118	437	245	72	200	359	169	208	327	280
27	108	191	337	165	220	188	190	43	266	161	216	241	104	246	177
28	45	139	273	228	121	60	314	81	132	189	308	112	158	335	266
29	167	79	77	205	97	185	435	243	111	163	322	238	206	288	243
30	29	80	151	222	84	71	102	25	117	29	20	457	19	98	159
31	11	107	241	190	124	80	316	76	152	157	283	133	113	297	228
32	107	12	148	137	88	127	336	183	134	95	254	180	101	234	175
33	241	148	13	374	171	259	509	317	217	232	491	312	280	391	412
34	190	137	374	14	202	234	222	192	248	42	117	287	79	107	38
35	124	88	171	202	15	61	392	202	46	160	319	112	163	322	240
36	80	127	259	234	61	16	386	141	72	167	351	55	157	331	272
37	316	336	509	222	392	386	17	233	438	254	202	439	235	254	210
38	76	183	317	192	202	141	233	18	213	188	272	193	131	302	233
39	152	134	217	248	46	72	438	213	19	206	365	89	209	368	286
40	157	95	232	42	160	167	254	188	206	20	159	220	57	149	80
41	283	254	491	117	319	351	202	272	365	159	21	404	176	106	79
42	133	180	312	287	112	55	439	193	89	220	404	22	210	384	325
43	113	101	280	79	163	157	235	131	209	57	176	210	23	186	117
44	297	234	391	107	322	331	254	302	368	149	106	384	186	24	69
45	228	175	412	38	240	272	210	233	286	80	79	325	117	69	25
46	129	176	349	121	232	226	187	98	278	132	161	279	75	191	122
47	348	265	422	152	314	362	313	344	360	193	165	415	231	59	122
48	276	199	356	86	287	296	266	289	333	127	141	349	165	35	56
49	188	182	355	68	238	232	154	177	284	100	95	285	81	125	56
50	150	67	204	70	155	164	282	216	201	28	187	217	85	167	108

dCollection @ chosun

도시	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	129	348	276	188	150	65	341	184	67	221	169	108	45	167	29
2	176	265	199	182	67	42	278	271	146	251	105	191	139	79	80
3	349	422	356	355	204	182	435	417	292	424	116	337	273	77	151
4	121	152	86	68	70	137	151	239	135	137	242	165	228	205	222
5	232	314	287	238	155	65	366	300	175	307	57	220	121	97	84
6	226	362	296	232	164	85	375	249	147	301	118	188	60	185	71
7	187	313	266	154	282	321	298	168	249	95	437	190	314	435	102
8	98	344	289	177	216	141	346	108	57	190	245	43	81	243	25
9	278	360	333	284	201	111	412	321	221	353	72	266	132	111	117
10	132	193	127	100	28	95	193	241	131	169	200	161	189	163	29
11	161	165	141	95	187	254	103	279	215	117	359	216	308	322	20
12	279	415	349	285	217	138	428	310	200	354	169	241	112	238	457
13	75	231	165	81	85	92	230	184	74	150	208	104	158	206	19
14	191	59	35	125	167	255	44	309	245	169	327	246	335	288	98
15	122	122	56	56	108	175	113	240	176	125	280	177	266	243	159
16	0	244	178	66	160	161	235	118	62	92	277	55	155	275	147
17	244	0	66	178	198	286	77	362	287	228	358	299	380	319	96
18	178	66	0	112	132	220	79	296	232	181	292	233	314	253	20
19	66	178	112	0	128	167	169	179	120	69	283	121	213	281	125
20	160	198	132	128	0	88	211	269	159	197	172	189	182	135	88
21	161	286	220	167	88	0	299	229	104	236	110	149	97	108	146
22	235	77	79	169	211	299	0	353	289	213	371	290	379	332	45
23	118	362	296	179	269	229	353	0	121	162	345	80	189	342	89
24	62	287	232	120	159	104	289	121	0	154	220	41	93	218	76
25	92	228	181	69	197	236	213	162	154	0	352	147	247	350	140
26	277	358	292	283	172	110	371	345	220	352	0	265	178	39	85
27	55	299	233	121	189	149	290	80	41	147	265	0	124	263	123
28	155	380	314	213	182	97	379	189	93	247	178	124	0	199	260
29	275	319	253	281	135	108	332	342	218	350	39	263	199	0	215
30	147	96	20	125	88	146	45	89	76	140	85	123	260	215	0
31	129	348	276	188	150	65	341	184	67	221	169	108	45	167	29
32	176	265	199	182	67	42	278	271	146	251	105	191	139	79	80
33	349	422	356	355	204	182	435	417	292	424	116	337	273	77	151
34	121	152	86	68	70	137	151	239	135	137	242	165	228	205	222
35	232	314	287	238	155	65	366	300	175	307	57	220	121	97	84
36	226	362	296	232	164	85	375	249	147	301	118	188	60	185	71
37	187	313	266	154	282	321	298	168	249	95	437	190	314	435	102
38	98	344	289	177	216	141	346	108	57	190	245	43	81	243	25
39	278	360	333	284	201	111	412	321	221	353	72	266	132	111	117
40	132	193	127	100	28	95	193	241	131	169	200	161	189	163	29
41	161	165	141	95	187	254	103	279	215	117	359	216	308	322	20
42	279	415	349	285	217	138	428	310	200	354	169	241	112	238	457
43	75	231	165	81	85	92	230	184	74	150	208	104	158	206	19
44	191	59	35	125	167	255	44	309	245	169	327	246	335	288	98
45	122	122	56	56	108	175	113	240	176	125	280	177	266	243	159
46	26	244	178	66	160	161	235	118	62	92	277	55	155	275	147
47	244	27	66	178	198	286	77	362	287	228	358	299	380	319	96
48	178	66	28	112	132	220	79	296	232	181	292	233	314	253	20
49	66	178	112	29	128	167	169	179	120	69	283	121	213	281	125
50	160	198	132	128	30	88	211	269	159	197	172	189	182	135	88

dCollection @ chosun

도시	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
1	11	107	241	190	124	80	316	76	152	157	283	133	113	297	228	129	348	276	188	150
2	107	12	148	137	88	127	336	183	134	95	254	180	101	234	175	176	265	199	182	67
3	241	148	13	374	171	259	509	317	217	232	491	312	280	391	412	349	422	356	355	204
4	190	137	374	14	202	234	222	192	248	42	117	287	79	107	38	121	152	86	68	70
5	124	88	171	202	15	61	392	202	46	160	319	112	163	322	240	232	314	287	238	155
6	80	127	259	234	61	16	386	141	72	167	351	55	157	331	272	226	362	296	232	164
7	316	336	509	222	392	386	17	233	438	254	202	439	235	254	210	187	313	266	154	282
8	76	183	317	192	202	141	233	18	213	188	272	193	131	302	233	98	344	289	177	216
9	152	134	217	248	46	72	438	213	19	206	365	89	209	368	286	278	360	333	284	201
10	157	95	232	42	160	167	254	188	206	20	159	220	57	149	80	132	193	127	100	28
11	283	254	491	117	319	351	202	272	365	159	21	404	176	106	79	161	165	141	95	187
12	133	180	312	287	112	55	439	193	89	220	404	22	210	384	325	279	415	349	285	217
13	113	101	280	79	163	157	235	131	209	57	176	210	23	186	117	75	231	165	81	85
14	297	234	391	107	322	331	254	302	368	149	106	384	186	24	69	191	59	35	125	167
15	228	175	412	38	240	272	210	233	286	80	79	325	117	69	25	122	122	56	56	108
16	129	176	349	121	232	226	187	98	278	132	161	279	75	191	122	26	244	178	66	160
17	348	265	422	152	314	362	313	344	360	193	165	415	231	59	122	244	27	66	178	198
18	276	199	356	86	287	296	266	289	333	127	141	349	165	35	56	178	66	28	112	132
19	188	182	355	68	238	232	154	177	284	100	95	285	81	125	56	66	178	112	29	128
20	150	67	204	70	155	164	282	216	201	28	187	217	85	167	108	160	198	132	128	30
21	65	42	182	137	65	85	321	141	111	95	254	138	92	255	175	161	286	220	167	88
22	341	278	435	151	366	375	298	346	412	193	103	428	230	44	113	235	77	79	169	211
23	184	271	417	239	300	249	168	108	321	241	279	310	184	309	240	118	362	296	179	269
24	67	146	292	135	175	147	249	57	221	131	215	200	74	245	176	62	287	232	120	159
25	221	251	424	137	307	301	95	190	353	169	117	354	150	169	125	92	228	181	69	197
26	169	105	116	242	57	118	437	245	72	200	359	169	208	327	280	277	358	292	283	172
27	108	191	337	165	220	188	190	43	266	161	216	241	104	246	177	55	299	233	121	189
28	45	139	273	228	121	60	314	81	132	189	308	112	158	335	266	155	380	314	213	182
29	167	79	77	205	97	185	435	243	111	163	322	238	206	288	243	275	319	253	281	135
30	29	80	151	222	84	71	102	25	117	29	20	457	19	98	159	147	96	20	125	88
31	0	107	241	190	124	80	316	76	152	157	283	133	113	297	228	129	348	276	188	150
32	107	0	148	137	88	127	336	183	134	95	254	180	101	234	175	176	265	199	182	67
33	241	148	0	374	171	259	509	317	217	232	491	312	280	391	412	349	422	356	355	204
34	190	137	374	0	202	234	222	192	248	42	117	287	79	107	38	121	152	86	68	70
35	124	88	171	202	0	61	392	202	46	160	319	112	163	322	240	232	314	287	238	155
36	80	127	259	234	61	0	386	141	72	167	351	55	157	331	272	226	362	296	232	164
37	316	336	509	222	392	386	0	233	438	254	202	439	235	254	210	187	313	266	154	282
38	76	183	317	192	202	141	233	0	213	188	272	193	131	302	233	98	344	289	177	216
39	152	134	217	248	46	72	438	213	0	206	365	89	209	368	286	278	360	333	284	201
40	157	95	232	42	160	167	254	188	206	0	159	220	57	149	80	132	193	127	100	28
41	283	254	491	117	319	351	202	272	365	159	0	404	176	106	79	161	165	141	95	187
42	133	180	312	287	112	55	439	193	89	220	404	0	210	384	325	279	415	349	285	217
43	113	101	280	79	163	157	235	131	209	57	176	210	0	186	117	75	231	165	81	85
44	297	234	391	107	322	331	254	302	368	149	106	384	186	0	69	191	59	35	125	167
45	228	175	412	38	240	272	210	233	286	80	79	325	117	69	0	122	122	56	56	108
46	129	176	349	121	232	226	187	98	278	132	161	279	75	191	122	0	244	178	66	160
47	348	265	422	152	314	362	313	344	360	193	165	415	231	59	122	244	0	66	178	198
48	276	199	356	86	287	296	266	289	333	127	141	349	165	35	56	178	66	0	112	132
49	188	182	355	68	238	232	154	177	284	100	95	285	81	125	56	66	178	112	0	128
50	150	67	204	70	155	164	282	216	201	28	187	217	85	167	108	160	198	132	128	0

dCollection @ chosun

부록 2 - 100개 도시 거리 정보

도시	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	107	241	190	124	80	316	76	152	157	283	133	113	297	228	129	348	276	188	150
2	107	0	148	137	88	127	336	183	134	95	254	180	101	234	175	176	265	199	182	67
3	241	148	0	374	171	259	509	317	217	232	491	312	280	391	412	349	422	356	355	204
4	190	137	374	0	202	234	222	192	248	42	117	287	79	107	38	121	152	86	68	70
5	124	88	171	202	0	61	392	202	46	160	319	112	163	322	240	232	314	287	238	155
6	80	127	259	234	61	0	386	141	72	167	351	55	157	331	272	226	362	296	232	164
7	316	336	509	222	392	386	0	233	438	254	202	439	235	254	210	187	313	266	154	282
8	76	183	317	192	202	141	233	0	213	188	272	193	131	302	233	98	344	289	177	216
9	152	134	217	248	46	72	438	213	0	206	365	89	209	368	286	278	360	333	284	201
10	157	95	232	42	160	167	254	188	206	0	159	220	57	149	80	132	193	127	100	28
11	283	254	491	117	319	351	202	272	365	159	0	404	176	106	79	161	165	141	95	187
12	133	180	312	287	112	55	439	193	89	220	404	0	210	384	325	279	415	349	285	217
13	113	101	280	79	163	157	235	131	209	57	176	210	0	186	117	75	231	165	81	85
14	297	234	391	107	322	331	254	302	368	149	106	384	186	0	69	191	59	35	125	167
15	228	175	412	38	240	272	210	233	286	80	79	325	117	69	0	122	122	56	56	108
16	129	176	349	121	232	226	187	98	278	132	161	279	75	191	122	0	244	178	66	160
17	348	265	422	152	314	362	313	344	360	193	165	415	231	59	122	244	0	66	178	198
18	276	199	356	86	287	296	266	289	333	127	141	349	165	35	56	178	66	0	112	132
19	188	182	355	68	238	232	154	177	284	100	95	285	81	125	56	66	178	112	0	128
20	150	67	204	70	155	164	282	216	201	28	187	217	85	167	108	160	198	132	128	0
21	65	42	182	137	65	85	321	141	111	95	254	138	92	255	175	161	286	220	167	88
22	341	278	435	151	366	375	298	346	412	193	103	428	230	44	113	235	77	79	169	211
23	184	271	417	239	300	249	168	108	321	241	279	310	184	309	240	118	362	296	179	269
24	67	146	292	135	175	147	249	57	221	131	215	200	74	245	176	62	287	232	120	159
25	221	251	424	137	307	301	95	190	353	169	117	354	150	169	125	92	228	181	69	197
26	169	105	116	242	57	118	437	245	72	200	359	169	208	327	280	277	358	292	283	172
27	108	191	337	165	220	188	190	43	266	161	216	241	104	246	177	55	299	233	121	189
28	45	139	273	228	121	60	314	81	132	189	308	112	158	335	266	155	380	314	213	182
29	167	79	77	205	97	185	435	243	111	163	322	238	206	288	243	275	319	253	281	135
30	29	80	151	222	84	71	102	25	117	29	20	457	19	98	159	147	96	20	125	88
31	11	107	241	190	124	80	316	76	152	157	283	133	113	297	228	129	348	276	188	150
32	107	12	148	137	88	127	336	183	134	95	254	180	101	234	175	176	265	199	182	67
33	241	148	13	374	171	259	509	317	217	232	491	312	280	391	412	349	422	356	355	204
34	190	137	374	14	202	234	222	192	248	42	117	287	79	107	38	121	152	86	68	70
35	124	88	171	202	15	61	392	202	46	160	319	112	163	322	240	232	314	287	238	155
36	80	127	259	234	61	16	386	141	72	167	351	55	157	331	272	226	362	296	232	164
37	316	336	509	222	392	386	17	233	438	254	202	439	235	254	210	187	313	266	154	282
38	76	183	317	192	202	141	233	18	213	188	272	193	131	302	233	98	344	289	177	216
39	152	134	217	248	46	72	438	213	19	206	365	89	209	368	286	278	360	333	284	201
40	157	95	232	42	160	167	254	188	206	20	159	220	57	149	80	132	193	127	100	28
41	283	254	491	117	319	351	202	272	365	159	21	404	176	106	79	161	165	141	95	187
42	133	180	312	287	112	55	439	193	89	220	404	22	210	384	325	279	415	349	285	217
43	113	101	280	79	163	157	235	131	209	57	176	210	23	186	117	75	231	165	81	85
44	297	234	391	107	322	331	254	302	368	149	106	384	186	24	69	191	59	35	125	167
45	228	175	412	38	240	272	210	233	286	80	79	325	117	69	25	122	122	56	56	108
46	129	176	349	121	232	226	187	98	278	132	161	279	75	191	122	26	244	178	66	160
47	348	265	422	152	314	362	313	344	360	193	165	415	231	59	122	244	27	66	178	198
48	276	199	356	86	287	296	266	289	333	127	141	349	165	35	56	178	66	28	112	132
49	188	182	355	68	238	232	154	177	284	100	95	285	81	125	56	66	178	112	29	128
50	150	67	204	70	155	164	282	216	201	28	187	217	85	167	108	160	198	132	128	30

dCollection @ chosun

도시	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	65	341	184	67	221	169	108	45	167	29	11	107	241	190	124	80	316	76	152	157
2	42	278	271	146	251	105	191	139	79	80	107	12	148	137	88	127	336	183	134	95
3	182	435	417	292	424	116	337	273	77	151	241	148	13	374	171	259	509	317	217	232
4	137	151	239	135	137	242	165	228	205	222	190	137	374	14	202	234	222	192	248	42
5	65	366	300	175	307	57	220	121	97	84	124	88	171	202	15	61	392	202	46	160
6	85	375	249	147	301	118	188	60	185	71	80	127	259	234	61	16	386	141	72	167
7	321	298	168	249	95	437	190	314	435	102	316	336	509	222	392	386	17	233	438	254
8	141	346	108	57	190	245	43	81	243	25	76	183	317	192	202	141	233	18	213	188
9	111	412	321	221	353	72	266	132	111	117	152	134	217	248	46	72	438	213	19	206
10	95	193	241	131	169	200	161	189	163	29	157	95	232	42	160	167	254	188	206	20
11	254	103	279	215	117	359	216	308	322	20	283	254	491	117	319	351	202	272	365	159
12	138	428	310	200	354	169	241	112	238	457	133	180	312	287	112	55	439	193	89	220
13	92	230	184	74	150	208	104	158	206	19	113	101	280	79	163	157	235	131	209	57
14	255	44	309	245	169	327	246	335	288	98	297	234	391	107	322	331	254	302	368	149
15	175	113	240	176	125	280	177	266	243	159	228	175	412	38	240	272	210	233	286	80
16	161	235	118	62	92	277	55	155	275	147	129	176	349	121	232	226	187	98	278	132
17	286	77	362	287	228	358	299	380	319	96	348	265	422	152	314	362	313	344	360	193
18	220	79	296	232	181	292	233	314	253	20	276	199	356	86	287	296	266	289	333	127
19	167	169	179	120	69	283	121	213	281	125	188	182	355	68	238	232	154	177	284	100
20	88	211	269	159	197	172	189	182	135	88	150	67	204	70	155	164	282	216	201	28
21	0	299	229	104	236	110	149	97	108	146	65	42	182	137	65	85	321	141	111	95
22	299	0	353	289	213	371	290	379	332	45	341	278	435	151	366	375	298	346	412	193
23	229	353	0	121	162	345	80	189	342	89	184	271	417	239	300	249	168	108	321	241
24	104	289	121	0	154	220	41	93	218	76	67	146	292	135	175	147	249	57	221	131
25	236	213	162	154	0	352	147	247	350	140	221	251	424	137	307	301	95	190	353	169
26	110	371	345	220	352	0	265	178	39	85	169	105	116	242	57	118	437	245	72	200
27	149	290	80	41	147	265	0	124	263	123	108	191	337	165	220	188	190	43	266	161
28	97	379	189	93	247	178	124	0	199	260	45	139	273	228	121	60	314	81	132	189
29	108	332	342	218	350	39	263	199	0	215	167	79	77	205	97	185	435	243	111	163
30	146	45	89	76	140	85	123	260	215	0	29	80	151	222	84	71	102	25	117	29
31	65	341	184	67	221	169	108	45	167	29	0	107	241	190	124	80	316	76	152	157
32	42	278	271	146	251	105	191	139	79	80	107	0	148	137	88	127	336	183	134	95
33	182	435	417	292	424	116	337	273	77	151	241	148	0	374	171	259	509	317	217	232
34	137	151	239	135	137	242	165	228	205	222	190	137	374	0	202	234	222	192	248	42
35	65	366	300	175	307	57	220	121	97	84	124	88	171	202	0	61	392	202	46	160
36	85	375	249	147	301	118	188	60	185	71	80	127	259	234	61	0	386	141	72	167
37	321	298	168	249	95	437	190	314	435	102	316	336	509	222	392	386	0	233	438	254
38	141	346	108	57	190	245	43	81	243	25	76	183	317	192	202	141	233	0	213	188
39	111	412	321	221	353	72	266	132	111	117	152	134	217	248	46	72	438	213	0	206
40	95	193	241	131	169	200	161	189	163	29	157	95	232	42	160	167	254	188	206	0
41	254	103	279	215	117	359	216	308	322	20	283	254	491	117	319	351	202	272	365	159
42	138	428	310	200	354	169	241	112	238	457	133	180	312	287	112	55	439	193	89	220
43	92	230	184	74	150	208	104	158	206	19	113	101	280	79	163	157	235	131	209	57
44	255	44	309	245	169	327	246	335	288	98	297	234	391	107	322	331	254	302	368	149
45	175	113	240	176	125	280	177	266	243	159	228	175	412	38	240	272	210	233	286	80
46	161	235	118	62	92	277	55	155	275	147	129	176	349	121	232	226	187	98	278	132
47	286	77	362	287	228	358	299	380	319	96	348	265	422	152	314	362	313	344	360	193
48	220	79	296	232	181	292	233	314	253	20	276	199	356	86	287	296	266	289	333	127
49	167	169	179	120	69	283	121	213	281	125	188	182	355	68	238	232	154	177	284	100
50	88	211	269	159	197	172	189	182	135	88	150	67	204	70	155	164	282	216	201	28

dCollection @ chosun

도시	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
1	283	133	113	297	228	129	348	276	188	150	2	107	241	190	124	80	316	76	152	157
2	254	180	101	234	175	176	265	199	182	67	107	10	148	137	88	127	336	183	134	95
3	491	312	280	391	412	349	422	356	355	204	241	148	57	374	171	259	509	317	217	232
4	117	287	79	107	38	121	152	86	68	70	190	137	374	66	202	234	222	192	248	42
5	319	112	163	322	240	232	314	287	238	155	124	88	171	202	74	61	392	202	46	160
6	351	55	157	331	272	226	362	296	232	164	80	127	259	234	61	11	386	141	72	167
7	202	439	235	254	210	187	313	266	154	282	316	336	509	222	392	386	5	233	438	254
8	272	193	131	302	233	98	344	289	177	216	76	183	317	192	202	141	233	125	213	188
9	365	89	209	368	286	278	360	333	284	201	152	134	217	248	46	72	438	213	114	206
10	159	220	57	149	80	132	193	127	100	28	157	95	232	42	160	167	254	188	206	10
11	21	404	176	106	79	161	165	141	95	187	283	254	491	117	319	351	202	272	365	159
12	404	22	210	384	325	279	415	349	285	217	133	180	312	287	112	55	439	193	89	220
13	176	210	23	186	117	75	231	165	81	85	113	101	280	79	163	157	235	131	209	57
14	106	384	186	24	69	191	59	35	125	167	297	234	391	107	322	331	254	302	368	149
15	79	325	117	69	25	122	122	56	56	108	228	175	412	38	240	272	210	233	286	80
16	161	279	75	191	122	26	244	178	66	160	129	176	349	121	232	226	187	98	278	132
17	165	415	231	59	122	244	27	66	178	198	348	265	422	152	314	362	313	344	360	193
18	141	349	165	35	56	178	66	28	112	132	276	199	356	86	287	296	266	289	333	127
19	95	285	81	125	56	66	178	112	29	128	188	182	355	68	238	232	154	177	284	100
20	187	217	85	167	108	160	198	132	128	30	150	67	204	70	155	164	282	216	201	28
21	254	138	92	255	175	161	286	220	167	88	65	42	182	137	65	85	321	141	111	95
22	103	428	230	44	113	235	77	79	169	211	341	278	435	151	366	375	298	346	412	193
23	279	310	184	309	240	118	362	296	179	269	184	271	417	239	300	249	168	108	321	241
24	215	200	74	245	176	62	287	232	120	159	67	146	292	135	175	147	249	57	221	131
25	117	354	150	169	125	92	228	181	69	197	221	251	424	137	307	301	95	190	353	169
26	359	169	208	327	280	277	358	292	283	172	169	105	116	242	57	118	437	245	72	200
27	216	241	104	246	177	55	299	233	121	189	108	191	337	165	220	188	190	43	266	161
28	308	112	158	335	266	155	380	314	213	182	45	139	273	228	121	60	314	81	132	189
29	322	238	206	288	243	275	319	253	281	135	167	79	77	205	97	185	435	243	111	163
30	20	457	19	98	159	147	96	20	125	88	29	80	151	222	84	71	102	25	117	29
31	283	133	113	297	228	129	348	276	188	150	11	107	241	190	124	80	316	76	152	157
32	254	180	101	234	175	176	265	199	182	67	107	12	148	137	88	127	336	183	134	95
33	491	312	280	391	412	349	422	356	355	204	241	148	13	374	171	259	509	317	217	232
34	117	287	79	107	38	121	152	86	68	70	190	137	374	14	202	234	222	192	248	42
35	319	112	163	322	240	232	314	287	238	155	124	88	171	202	15	61	392	202	46	160
36	351	55	157	331	272	226	362	296	232	164	80	127	259	234	61	16	386	141	72	167
37	202	439	235	254	210	187	313	266	154	282	316	336	509	222	392	386	17	233	438	254
38	272	193	131	302	233	98	344	289	177	216	76	183	317	192	202	141	233	18	213	188
39	365	89	209	368	286	278	360	333	284	201	152	134	217	248	46	72	438	213	19	206
40	159	220	57	149	80	132	193	127	100	28	157	95	232	42	160	167	254	188	206	20
41	0	404	176	106	79	161	165	141	95	187	283	254	491	117	319	351	202	272	365	159
42	404	0	210	384	325	279	415	349	285	217	133	180	312	287	112	55	439	193	89	220
43	176	210	0	186	117	75	231	165	81	85	113	101	280	79	163	157	235	131	209	57
44	106	384	186	0	69	191	59	35	125	167	297	234	391	107	322	331	254	302	368	149
45	79	325	117	69	0	122	122	56	56	108	228	175	412	38	240	272	210	233	286	80
46	161	279	75	191	122	0	244	178	66	160	129	176	349	121	232	226	187	98	278	132
47	165	415	231	59	122	244	0	66	178	198	348	265	422	152	314	362	313	344	360	193
48	141	349	165	35	56	178	66	0	112	132	276	199	356	86	287	296	266	289	333	127
49	95	285	81	125	56	66	178	112	0	128	188	182	355	68	238	232	154	177	284	100
50	187	217	85	167	108	160	198	132	128	0	150	67	204	70	155	164	282	216	201	28

dCollection @ chosun

도시	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
1	283	133	113	297	228	129	348	276	188	150	65	341	184	67	221	169	108	45	167	29
2	254	180	101	234	175	176	265	199	182	67	42	278	271	146	251	105	191	139	79	80
3	491	312	280	391	412	349	422	356	355	204	182	435	417	292	424	116	337	273	77	151
4	117	287	79	107	38	121	152	86	68	70	137	151	239	135	137	242	165	228	205	222
5	319	112	163	322	240	232	314	287	238	155	65	366	300	175	307	57	220	121	97	84
6	351	55	157	331	272	226	362	296	232	164	85	375	249	147	301	118	188	60	185	71
7	202	439	235	254	210	187	313	266	154	282	321	298	168	249	95	437	190	314	435	102
8	272	193	131	302	233	98	344	289	177	216	141	346	108	57	190	245	43	81	243	25
9	365	89	209	368	286	278	360	333	284	201	111	412	321	221	353	72	266	132	111	117
10	159	220	57	149	80	132	193	127	100	28	95	193	241	131	169	200	161	189	163	29
11	154	404	176	106	79	161	165	141	95	187	254	103	279	215	117	359	216	308	322	20
12	404	200	210	384	325	279	415	349	285	217	138	428	310	200	354	169	241	112	238	457
13	176	210	5	186	117	75	231	165	81	85	92	230	184	74	150	208	104	158	206	19
14	106	384	186	157	69	191	59	35	125	167	255	44	309	245	169	327	246	335	288	98
15	79	325	117	69	98	122	122	56	56	108	175	113	240	176	125	280	177	266	243	159
16	161	279	75	191	122	87	244	178	66	160	161	235	118	62	92	277	55	155	275	147
17	165	415	231	59	122	244	14	66	178	198	286	77	362	287	228	358	299	380	319	96
18	141	349	165	35	56	178	66	58	112	132	220	79	296	232	181	292	233	314	253	20
19	95	285	81	125	56	66	178	112	20	128	167	169	179	120	69	283	121	213	281	125
20	187	217	85	167	108	160	198	132	128	8	88	211	269	159	197	172	189	182	135	88
21	254	138	92	255	175	161	286	220	167	88	148	299	229	104	236	110	149	97	108	146
22	103	428	230	44	113	235	77	79	169	211	299	7	353	289	213	371	290	379	332	45
23	279	310	184	309	240	118	362	296	179	269	229	353	146	121	162	345	80	189	342	89
24	215	200	74	245	176	62	287	232	120	159	104	289	121	523	154	220	41	93	218	76
25	117	354	150	169	125	92	228	181	69	197	236	213	162	154	254	352	147	247	350	140
26	359	169	208	327	280	277	358	292	283	172	110	371	345	220	352	88	265	178	39	85
27	216	241	104	246	177	55	299	233	121	189	149	290	80	41	147	265	97	124	263	123
28	308	112	158	335	266	155	380	314	213	182	97	379	189	93	247	178	124	17	199	260
29	322	238	206	288	243	275	319	253	281	135	108	332	342	218	350	39	263	199	56	215
30	20	457	19	98	159	147	96	20	125	88	146	45	89	76	140	85	123	260	215	87
31	283	133	113	297	228	129	348	276	188	150	65	341	184	67	221	169	108	45	167	29
32	254	180	101	234	175	176	265	199	182	67	42	278	271	146	251	105	191	139	79	80
33	491	312	280	391	412	349	422	356	355	204	182	435	417	292	424	116	337	273	77	151
34	117	287	79	107	38	121	152	86	68	70	137	151	239	135	137	242	165	228	205	222
35	319	112	163	322	240	232	314	287	238	155	65	366	300	175	307	57	220	121	97	84
36	351	55	157	331	272	226	362	296	232	164	85	375	249	147	301	118	188	60	185	71
37	202	439	235	254	210	187	313	266	154	282	321	298	168	249	95	437	190	314	435	102
38	272	193	131	302	233	98	344	289	177	216	141	346	108	57	190	245	43	81	243	25
39	365	89	209	368	286	278	360	333	284	201	111	412	321	221	353	72	266	132	111	117
40	159	220	57	149	80	132	193	127	100	28	95	193	241	131	169	200	161	189	163	29
41	21	404	176	106	79	161	165	141	95	187	254	103	279	215	117	359	216	308	322	20
42	404	22	210	384	325	279	415	349	285	217	138	428	310	200	354	169	241	112	238	457
43	176	210	23	186	117	75	231	165	81	85	92	230	184	74	150	208	104	158	206	19
44	106	384	186	24	69	191	59	35	125	167	255	44	309	245	169	327	246	335	288	98
45	79	325	117	69	25	122	122	56	56	108	175	113	240	176	125	280	177	266	243	159
46	161	279	75	191	122	26	244	178	66	160	161	235	118	62	92	277	55	155	275	147
47	165	415	231	59	122	244	27	66	178	198	286	77	362	287	228	358	299	380	319	96
48	141	349	165	35	56	178	66	28	112	132	220	79	296	232	181	292	233	314	253	20
49	95	285	81	125	56	66	178	112	29	128	167	169	179	120	69	283	121	213	281	125
50	187	217	85	167	108	160	198	132	128	30	88	211	269	159	197	172	189	182	135	88

dCollection @ chosun

도시	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
1	11	107	241	190	124	80	316	76	152	157	283	133	113	297	228	129	348	276	188	150
2	107	12	148	137	88	127	336	183	134	95	254	180	101	234	175	176	265	199	182	67
3	241	148	13	374	171	259	509	317	217	232	491	312	280	391	412	349	422	356	355	204
4	190	137	374	14	202	234	222	192	248	42	117	287	79	107	38	121	152	86	68	70
5	124	88	171	202	15	61	392	202	46	160	319	112	163	322	240	232	314	287	238	155
6	80	127	259	234	61	16	386	141	72	167	351	55	157	331	272	226	362	296	232	164
7	316	336	509	222	392	386	17	233	438	254	202	439	235	254	210	187	313	266	154	282
8	76	183	317	192	202	141	233	18	213	188	272	193	131	302	233	98	344	289	177	216
9	152	134	217	248	46	72	438	213	19	206	365	89	209	368	286	278	360	333	284	201
10	157	95	232	42	160	167	254	188	206	20	159	220	57	149	80	132	193	127	100	28
11	283	254	491	117	319	351	202	272	365	159	21	404	176	106	79	161	165	141	95	187
12	133	180	312	287	112	55	439	193	89	220	404	22	210	384	325	279	415	349	285	217
13	113	101	280	79	163	157	235	131	209	57	176	210	23	186	117	75	231	165	81	85
14	297	234	391	107	322	331	254	302	368	149	106	384	186	24	69	191	59	35	125	167
15	228	175	412	38	240	272	210	233	286	80	79	325	117	69	25	122	122	56	56	108
16	129	176	349	121	232	226	187	98	278	132	161	279	75	191	122	26	244	178	66	160
17	348	265	422	152	314	362	313	344	360	193	165	415	231	59	122	244	27	66	178	198
18	276	199	356	86	287	296	266	289	333	127	141	349	165	35	56	178	66	28	112	132
19	188	182	355	68	238	232	154	177	284	100	95	285	81	125	56	66	178	112	29	128
20	150	67	204	70	155	164	282	216	201	28	187	217	85	167	108	160	198	132	128	30
21	65	42	182	137	65	85	321	141	111	95	254	138	92	255	175	161	286	220	167	88
22	341	278	435	151	366	375	298	346	412	193	103	428	230	44	113	235	77	79	169	211
23	184	271	417	239	300	249	168	108	321	241	279	310	184	309	240	118	362	296	179	269
24	67	146	292	135	175	147	249	57	221	131	215	200	74	245	176	62	287	232	120	159
25	221	251	424	137	307	301	95	190	353	169	117	354	150	169	125	92	228	181	69	197
26	169	105	116	242	57	118	437	245	72	200	359	169	208	327	280	277	358	292	283	172
27	108	191	337	165	220	188	190	43	266	161	216	241	104	246	177	55	299	233	121	189
28	45	139	273	228	121	60	314	81	132	189	308	112	158	335	266	155	380	314	213	182
29	167	79	77	205	97	185	435	243	111	163	322	238	206	288	243	275	319	253	281	135
30	29	80	151	222	84	71	102	25	117	29	20	457	19	98	159	147	96	20	125	88
31	28	107	241	190	124	80	316	76	152	157	283	133	113	297	228	129	348	276	188	150
32	107	3	148	137	88	127	336	183	134	95	254	180	101	234	175	176	265	199	182	67
33	241	148	32	374	171	259	509	317	217	232	491	312	280	391	412	349	422	356	355	204
34	190	137	374	97	202	234	222	192	248	42	117	287	79	107	38	121	152	86	68	70
35	124	88	171	202	7	61	392	202	46	160	319	112	163	322	240	232	314	287	238	155
36	80	127	259	234	61	71	386	141	72	167	351	55	157	331	272	226	362	296	232	164
37	316	336	509	222	392	386	22	233	438	254	202	439	235	254	210	187	313	266	154	282
38	76	183	317	192	202	141	233	80	213	188	272	193	131	302	233	98	344	289	177	216
39	152	134	217	248	46	72	438	213	1	206	365	89	209	368	286	278	360	333	284	201
40	157	95	232	42	160	167	254	188	206	214	159	220	57	149	80	132	193	127	100	28
41	283	254	491	117	319	351	202	272	365	159	108	404	176	106	79	161	165	141	95	187
42	133	180	312	287	112	55	439	193	89	220	404	337	210	384	325	279	415	349	285	217
43	113	101	280	79	163	157	235	131	209	57	176	210	157	186	117	75	231	165	81	85
44	297	234	391	107	322	331	254	302	368	149	106	384	186	421	69	191	59	35	125	167
45	228	175	412	38	240	272	210	233	286	80	79	325	117	69	21	122	122	56	56	108
46	129	176	349	121	232	226	187	98	278	132	161	279	75	191	122	22	244	178	66	160
47	348	265	422	152	314	362	313	344	360	193	165	415	231	59	122	244	64	66	178	198
48	276	199	356	86	287	296	266	289	333	127	141	349	165	35	56	178	66	19	112	132
49	188	182	355	68	238	232	154	177	284	100	95	285	81	125	56	66	178	112	113	128
50	150	67	204	70	155	164	282	216	201	28	187	217	85	167	108	160	198	132	128	27

dCollection @ chosun

도시	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
51	2	107	241	190	124	80	316	76	152	157	283	133	113	297	228	129	348	276	188	150
52	107	10	148	137	88	127	336	183	134	95	254	180	101	234	175	176	265	199	182	67
53	241	148	57	374	171	259	509	317	217	232	491	312	280	391	412	349	422	356	355	204
54	190	137	374	66	202	234	222	192	248	42	117	287	79	107	38	121	152	86	68	70
55	124	88	171	202	74	61	392	202	46	160	319	112	163	322	240	232	314	287	238	155
56	80	127	259	234	61	11	386	141	72	167	351	55	157	331	272	226	362	296	232	164
57	316	336	509	222	392	386	5	233	438	254	202	439	235	254	210	187	313	266	154	282
58	76	183	317	192	202	141	233	125	213	188	272	193	131	302	233	98	344	289	177	216
59	152	134	217	248	46	72	438	213	114	206	365	89	209	368	286	278	360	333	284	201
60	157	95	232	42	160	167	254	188	206	10	159	220	57	149	80	132	193	127	100	28
61	283	254	491	117	319	351	202	272	365	159	154	404	176	106	79	161	165	141	95	187
62	133	180	312	287	112	55	439	193	89	220	404	200	210	384	325	279	415	349	285	217
63	113	101	280	79	163	157	235	131	209	57	176	210	5	186	117	75	231	165	81	85
64	297	234	391	107	322	331	254	302	368	149	106	384	186	157	69	191	59	35	125	167
65	228	175	412	38	240	272	210	233	286	80	79	325	117	69	98	122	122	56	56	108
66	129	176	349	121	232	226	187	98	278	132	161	279	75	191	122	87	244	178	66	160
67	348	265	422	152	314	362	313	344	360	193	165	415	231	59	122	244	14	66	178	198
68	276	199	356	86	287	296	266	289	333	127	141	349	165	35	56	178	66	58	112	132
69	188	182	355	68	238	232	154	177	284	100	95	285	81	125	56	66	178	112	20	128
70	150	67	204	70	155	164	282	216	201	28	187	217	85	167	108	160	198	132	128	8
71	65	42	182	137	65	85	321	141	111	95	254	138	92	255	175	161	286	220	167	88
72	341	278	435	151	366	375	298	346	412	193	103	428	230	44	113	235	77	79	169	211
73	184	271	417	239	300	249	168	108	321	241	279	310	184	309	240	118	362	296	179	269
74	67	146	292	135	175	147	249	57	221	131	215	200	74	245	176	62	287	232	120	159
75	221	251	424	137	307	301	95	190	353	169	117	354	150	169	125	92	228	181	69	197
76	169	105	116	242	57	118	437	245	72	200	359	169	208	327	280	277	358	292	283	172
77	108	191	337	165	220	188	190	43	266	161	216	241	104	246	177	55	299	233	121	189
78	45	139	273	228	121	60	314	81	132	189	308	112	158	335	266	155	380	314	213	182
79	167	79	77	205	97	185	435	243	111	163	322	238	206	288	243	275	319	253	281	135
80	29	80	151	222	84	71	102	25	117	29	20	457	19	98	159	147	96	20	125	88
81	11	107	241	190	124	80	316	76	152	157	283	133	113	297	228	129	348	276	188	150
82	107	12	148	137	88	127	336	183	134	95	254	180	101	234	175	176	265	199	182	67
83	241	148	13	374	171	259	509	317	217	232	491	312	280	391	412	349	422	356	355	204
84	190	137	374	14	202	234	222	192	248	42	117	287	79	107	38	121	152	86	68	70
85	124	88	171	202	15	61	392	202	46	160	319	112	163	322	240	232	314	287	238	155
86	80	127	259	234	61	16	386	141	72	167	351	55	157	331	272	226	362	296	232	164
87	316	336	509	222	392	386	17	233	438	254	202	439	235	254	210	187	313	266	154	282
88	76	183	317	192	202	141	233	18	213	188	272	193	131	302	233	98	344	289	177	216
89	152	134	217	248	46	72	438	213	19	206	365	89	209	368	286	278	360	333	284	201
90	157	95	232	42	160	167	254	188	206	20	159	220	57	149	80	132	193	127	100	28
91	283	254	491	117	319	351	202	272	365	159	21	404	176	106	79	161	165	141	95	187
92	133	180	312	287	112	55	439	193	89	220	404	22	210	384	325	279	415	349	285	217
93	113	101	280	79	163	157	235	131	209	57	176	210	23	186	117	75	231	165	81	85
94	297	234	391	107	322	331	254	302	368	149	106	384	186	24	69	191	59	35	125	167
95	228	175	412	38	240	272	210	233	286	80	79	325	117	69	25	122	122	56	56	108
96	129	176	349	121	232	226	187	98	278	132	161	279	75	191	122	26	244	178	66	160
97	348	265	422	152	314	362	313	344	360	193	165	415	231	59	122	244	27	66	178	198
98	276	199	356	86	287	296	266	289	333	127	141	349	165	35	56	178	66	28	112	132
99	188	182	355	68	238	232	154	177	284	100	95	285	81	125	56	66	178	112	29	128
100	150	67	204	70	155	164	282	216	201	28	187	217	85	167	108	160	198	132	128	30

dCollection @ chosun

도시	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
51	65	341	184	67	221	169	108	45	167	29	11	107	241	190	124	80	316	76	152	157
52	42	278	271	146	251	105	191	139	79	80	107	12	148	137	88	127	336	183	134	95
53	182	435	417	292	424	116	337	273	77	151	241	148	13	374	171	259	509	317	217	232
54	137	151	239	135	137	242	165	228	205	222	190	137	374	14	202	234	222	192	248	42
55	65	366	300	175	307	57	220	121	97	84	124	88	171	202	15	61	392	202	46	160
56	85	375	249	147	301	118	188	60	185	71	80	127	259	234	61	16	386	141	72	167
57	321	298	168	249	95	437	190	314	435	102	316	336	509	222	392	386	17	233	438	254
58	141	346	108	57	190	245	43	81	243	25	76	183	317	192	202	141	233	18	213	188
59	111	412	321	221	353	72	266	132	111	117	152	134	217	248	46	72	438	213	19	206
60	95	193	241	131	169	200	161	189	163	29	157	95	232	42	160	167	254	188	206	20
61	254	103	279	215	117	359	216	308	322	20	283	254	491	117	319	351	202	272	365	159
62	138	428	310	200	354	169	241	112	238	457	133	180	312	287	112	55	439	193	89	220
63	92	230	184	74	150	208	104	158	206	19	113	101	280	79	163	157	235	131	209	57
64	255	44	309	245	169	327	246	335	288	98	297	234	391	107	322	331	254	302	368	149
65	175	113	240	176	125	280	177	266	243	159	228	175	412	38	240	272	210	233	286	80
66	161	235	118	62	92	277	55	155	275	147	129	176	349	121	232	226	187	98	278	132
67	286	77	362	287	228	358	299	380	319	96	348	265	422	152	314	362	313	344	360	193
68	220	79	296	232	181	292	233	314	253	20	276	199	356	86	287	296	266	289	333	127
69	167	169	179	120	69	283	121	213	281	125	188	182	355	68	238	232	154	177	284	100
70	88	211	269	159	197	172	189	182	135	88	150	67	204	70	155	164	282	216	201	28
71	148	299	229	104	236	110	149	97	108	146	65	42	182	137	65	85	321	141	111	95
72	299	7	353	289	213	371	290	379	332	45	341	278	435	151	366	375	298	346	412	193
73	229	353	146	121	162	345	80	189	342	89	184	271	417	239	300	249	168	108	321	241
74	104	289	121	523	154	220	41	93	218	76	67	146	292	135	175	147	249	57	221	131
75	236	213	162	154	254	352	147	247	350	140	221	251	424	137	307	301	95	190	353	169
76	110	371	345	220	352	88	265	178	39	85	169	105	116	242	57	118	437	245	72	200
77	149	290	80	41	147	265	97	124	263	123	108	191	337	165	220	188	190	43	266	161
78	97	379	189	93	247	178	124	17	199	260	45	139	273	228	121	60	314	81	132	189
79	108	332	342	218	350	39	263	199	56	215	167	79	77	205	97	185	435	243	111	163
80	146	45	89	76	140	85	123	260	215	87	29	80	151	222	84	71	102	25	117	29
81	65	341	184	67	221	169	108	45	167	29	28	107	241	190	124	80	316	76	152	157
82	42	278	271	146	251	105	191	139	79	80	107	3	148	137	88	127	336	183	134	95
83	182	435	417	292	424	116	337	273	77	151	241	148	32	374	171	259	509	317	217	232
84	137	151	239	135	137	242	165	228	205	222	190	137	374	97	202	234	222	192	248	42
85	65	366	300	175	307	57	220	121	97	84	124	88	171	202	7	61	392	202	46	160
86	85	375	249	147	301	118	188	60	185	71	80	127	259	234	61	71	386	141	72	167
87	321	298	168	249	95	437	190	314	435	102	316	336	509	222	392	386	22	233	438	254
88	141	346	108	57	190	245	43	81	243	25	76	183	317	192	202	141	233	80	213	188
89	111	412	321	221	353	72	266	132	111	117	152	134	217	248	46	72	438	213	1	206
90	95	193	241	131	169	200	161	189	163	29	157	95	232	42	160	167	254	188	206	214
91	254	103	279	215	117	359	216	308	322	20	283	254	491	117	319	351	202	272	365	159
92	138	428	310	200	354	169	241	112	238	457	133	180	312	287	112	55	439	193	89	220
93	92	230	184	74	150	208	104	158	206	19	113	101	280	79	163	157	235	131	209	57
94	255	44	309	245	169	327	246	335	288	98	297	234	391	107	322	331	254	302	368	149
95	175	113	240	176	125	280	177	266	243	159	228	175	412	38	240	272	210	233	286	80
96	161	235	118	62	92	277	55	155	275	147	129	176	349	121	232	226	187	98	278	132
97	286	77	362	287	228	358	299	380	319	96	348	265	422	152	314	362	313	344	360	193
98	220	79	296	232	181	292	233	314	253	20	276	199	356	86	287	296	266	289	333	127
99	167	169	179	120	69	283	121	213	281	125	188	182	355	68	238	232	154	177	284	100
100	88	211	269	159	197	172	189	182	135	88	150	67	204	70	155	164	282	216	201	28

dCollection @ chosun

도시	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
51	283	133	113	297	228	129	348	276	188	150	0	107	241	190	124	80	316	76	152	157
52	254	180	101	234	175	176	265	199	182	67	107	0	148	137	88	127	336	183	134	95
53	491	312	280	391	412	349	422	356	355	204	241	148	0	374	171	259	509	317	217	232
54	117	287	79	107	38	121	152	86	68	70	190	137	374	0	202	234	222	192	248	42
55	319	112	163	322	240	232	314	287	238	155	124	88	171	202	0	61	392	202	46	160
56	351	55	157	331	272	226	362	296	232	164	80	127	259	234	61	0	386	141	72	167
57	202	439	235	254	210	187	313	266	154	282	316	336	509	222	392	386	0	233	438	254
58	272	193	131	302	233	98	344	289	177	216	76	183	317	192	202	141	233	0	213	188
59	365	89	209	368	286	278	360	333	284	201	152	134	217	248	46	72	438	213	0	206
60	159	220	57	149	80	132	193	127	100	28	157	95	232	42	160	167	254	188	206	0
61	21	404	176	106	79	161	165	141	95	187	283	254	491	117	319	351	202	272	365	159
62	404	22	210	384	325	279	415	349	285	217	133	180	312	287	112	55	439	193	89	220
63	176	210	23	186	117	75	231	165	81	85	113	101	280	79	163	157	235	131	209	57
64	106	384	186	24	69	191	59	35	125	167	297	234	391	107	322	331	254	302	368	149
65	79	325	117	69	25	122	122	56	56	108	228	175	412	38	240	272	210	233	286	80
66	161	279	75	191	122	26	244	178	66	160	129	176	349	121	232	226	187	98	278	132
67	165	415	231	59	122	244	27	66	178	198	348	265	422	152	314	362	313	344	360	193
68	141	349	165	35	56	178	66	28	112	132	276	199	356	86	287	296	266	289	333	127
69	95	285	81	125	56	66	178	112	29	128	188	182	355	68	238	232	154	177	284	100
70	187	217	85	167	108	160	198	132	128	30	150	67	204	70	155	164	282	216	201	28
71	254	138	92	255	175	161	286	220	167	88	65	42	182	137	65	85	321	141	111	95
72	103	428	230	44	113	235	77	79	169	211	341	278	435	151	366	375	298	346	412	193
73	279	310	184	309	240	118	362	296	179	269	184	271	417	239	300	249	168	108	321	241
74	215	200	74	245	176	62	287	232	120	159	67	146	292	135	175	147	249	57	221	131
75	117	354	150	169	125	92	228	181	69	197	221	251	424	137	307	301	95	190	353	169
76	359	169	208	327	280	277	358	292	283	172	169	105	116	242	57	118	437	245	72	200
77	216	241	104	246	177	55	299	233	121	189	108	191	337	165	220	188	190	43	266	161
78	308	112	158	335	266	155	380	314	213	182	45	139	273	228	121	60	314	81	132	189
79	322	238	206	288	243	275	319	253	281	135	167	79	77	205	97	185	435	243	111	163
80	20	457	19	98	159	147	96	20	125	88	29	80	151	222	84	71	102	25	117	29
81	283	133	113	297	228	129	348	276	188	150	11	107	241	190	124	80	316	76	152	157
82	254	180	101	234	175	176	265	199	182	67	107	12	148	137	88	127	336	183	134	95
83	491	312	280	391	412	349	422	356	355	204	241	148	13	374	171	259	509	317	217	232
84	117	287	79	107	38	121	152	86	68	70	190	137	374	14	202	234	222	192	248	42
85	319	112	163	322	240	232	314	287	238	155	124	88	171	202	15	61	392	202	46	160
86	351	55	157	331	272	226	362	296	232	164	80	127	259	234	61	16	386	141	72	167
87	202	439	235	254	210	187	313	266	154	282	316	336	509	222	392	386	17	233	438	254
88	272	193	131	302	233	98	344	289	177	216	76	183	317	192	202	141	233	18	213	188
89	365	89	209	368	286	278	360	333	284	201	152	134	217	248	46	72	438	213	19	206
90	159	220	57	149	80	132	193	127	100	28	157	95	232	42	160	167	254	188	206	20
91	108	404	176	106	79	161	165	141	95	187	283	254	491	117	319	351	202	272	365	159
92	404	337	210	384	325	279	415	349	285	217	133	180	312	287	112	55	439	193	89	220
93	176	210	157	186	117	75	231	165	81	85	113	101	280	79	163	157	235	131	209	57
94	106	384	186	421	69	191	59	35	125	167	297	234	391	107	322	331	254	302	368	149
95	79	325	117	69	21	122	122	56	56	108	228	175	412	38	240	272	210	233	286	80
96	161	279	75	191	122	22	244	178	66	160	129	176	349	121	232	226	187	98	278	132
97	165	415	231	59	122	244	64	66	178	198	348	265	422	152	314	362	313	344	360	193
98	141	349	165	35	56	178	66	19	112	132	276	199	356	86	287	296	266	289	333	127
99	95	285	81	125	56	66	178	112	113	128	188	182	355	68	238	232	154	177	284	100
100	187	217	85	167	108	160	198	132	128	27	150	67	204	70	155	164	282	216	201	28

dCollection @ chosun

도시	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
51	283	133	113	297	228	129	348	276	188	150	65	341	184	67	221	169	108	45	167	29
52	254	180	101	234	175	176	265	199	182	67	42	278	271	146	251	105	191	139	79	80
53	491	312	280	391	412	349	422	356	355	204	182	435	417	292	424	116	337	273	77	151
54	117	287	79	107	38	121	152	86	68	70	137	151	239	135	137	242	165	228	205	222
55	319	112	163	322	240	232	314	287	238	155	65	366	300	175	307	57	220	121	97	84
56	351	55	157	331	272	226	362	296	232	164	85	375	249	147	301	118	188	60	185	71
57	202	439	235	254	210	187	313	266	154	282	321	298	168	249	95	437	190	314	435	102
58	272	193	131	302	233	98	344	289	177	216	141	346	108	57	190	245	43	81	243	25
59	365	89	209	368	286	278	360	333	284	201	111	412	321	221	353	72	266	132	111	117
60	159	220	57	149	80	132	193	127	100	28	95	193	241	131	169	200	161	189	163	29
61	0	404	176	106	79	161	165	141	95	187	254	103	279	215	117	359	216	308	322	20
62	404	0	210	384	325	279	415	349	285	217	138	428	310	200	354	169	241	112	238	457
63	176	210	0	186	117	75	231	165	81	85	92	230	184	74	150	208	104	158	206	19
64	106	384	186	0	69	191	59	35	125	167	255	44	309	245	169	327	246	335	288	98
65	79	325	117	69	0	122	122	56	56	108	175	113	240	176	125	280	177	266	243	159
66	161	279	75	191	122	0	244	178	66	160	161	235	118	62	92	277	55	155	275	147
67	165	415	231	59	122	244	0	66	178	198	286	77	362	287	228	358	299	380	319	96
68	141	349	165	35	56	178	66	0	112	132	220	79	296	232	181	292	233	314	253	20
69	95	285	81	125	56	66	178	112	0	128	167	169	179	120	69	283	121	213	281	125
70	187	217	85	167	108	160	198	132	128	0	88	211	269	159	197	172	189	182	135	88
71	254	138	92	255	175	161	286	220	167	88	0	299	229	104	236	110	149	97	108	146
72	103	428	230	44	113	235	77	79	169	211	299	0	353	289	213	371	290	379	332	45
73	279	310	184	309	240	118	362	296	179	269	229	353	0	121	162	345	80	189	342	89
74	215	200	74	245	176	62	287	232	120	159	104	289	121	0	154	220	41	93	218	76
75	117	354	150	169	125	92	228	181	69	197	236	213	162	154	0	352	147	247	350	140
76	359	169	208	327	280	277	358	292	283	172	110	371	345	220	352	0	265	178	39	85
77	216	241	104	246	177	55	299	233	121	189	149	290	80	41	147	265	0	124	263	123
78	308	112	158	335	266	155	380	314	213	182	97	379	189	93	247	178	124	0	199	260
79	322	238	206	288	243	275	319	253	281	135	108	332	342	218	350	39	263	199	0	215
80	20	457	19	98	159	147	96	20	125	88	146	45	89	76	140	85	123	260	215	0
81	283	133	113	297	228	129	348	276	188	150	65	341	184	67	221	169	108	45	167	29
82	254	180	101	234	175	176	265	199	182	67	42	278	271	146	251	105	191	139	79	80
83	491	312	280	391	412	349	422	356	355	204	182	435	417	292	424	116	337	273	77	151
84	117	287	79	107	38	121	152	86	68	70	137	151	239	135	137	242	165	228	205	222
85	319	112	163	322	240	232	314	287	238	155	65	366	300	175	307	57	220	121	97	84
86	351	55	157	331	272	226	362	296	232	164	85	375	249	147	301	118	188	60	185	71
87	202	439	235	254	210	187	313	266	154	282	321	298	168	249	95	437	190	314	435	102
88	272	193	131	302	233	98	344	289	177	216	141	346	108	57	190	245	43	81	243	25
89	365	89	209	368	286	278	360	333	284	201	111	412	321	221	353	72	266	132	111	117
90	159	220	57	149	80	132	193	127	100	28	95	193	241	131	169	200	161	189	163	29
91	21	404	176	106	79	161	165	141	95	187	254	103	279	215	117	359	216	308	322	20
92	404	22	210	384	325	279	415	349	285	217	138	428	310	200	354	169	241	112	238	457
93	176	210	23	186	117	75	231	165	81	85	92	230	184	74	150	208	104	158	206	19
94	106	384	186	24	69	191	59	35	125	167	255	44	309	245	169	327	246	335	288	98
95	79	325	117	69	25	122	122	56	56	108	175	113	240	176	125	280	177	266	243	159
96	161	279	75	191	122	26	244	178	66	160	161	235	118	62	92	277	55	155	275	147
97	165	415	231	59	122	244	27	66	178	198	286	77	362	287	228	358	299	380	319	96
98	141	349	165	35	56	178	66	28	112	132	220	79	296	232	181	292	233	314	253	20
99	95	285	81	125	56	66	178	112	29	128	167	169	179	120	69	283	121	213	281	125
100	187	217	85	167	108	160	198	132	128	30	88	211	269	159	197	172	189	182	135	88

dCollection @ chosun

도시	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
51	11	107	241	190	124	80	316	76	152	157	283	133	113	297	228	129	348	276	188	150
52	107	12	148	137	88	127	336	183	134	95	254	180	101	234	175	176	265	199	182	67
53	241	148	13	374	171	259	509	317	217	232	491	312	280	391	412	349	422	356	355	204
54	190	137	374	14	202	234	222	192	248	42	117	287	79	107	38	121	152	86	68	70
55	124	88	171	202	15	61	392	202	46	160	319	112	163	322	240	232	314	287	238	155
56	80	127	259	234	61	16	386	141	72	167	351	55	157	331	272	226	362	296	232	164
57	316	336	509	222	392	386	17	233	438	254	202	439	235	254	210	187	313	266	154	282
58	76	183	317	192	202	141	233	18	213	188	272	193	131	302	233	98	344	289	177	216
59	152	134	217	248	46	72	438	213	19	206	365	89	209	368	286	278	360	333	284	201
60	157	95	232	42	160	167	254	188	206	20	159	220	57	149	80	132	193	127	100	28
61	283	254	491	117	319	351	202	272	365	159	21	404	176	106	79	161	165	141	95	187
62	133	180	312	287	112	55	439	193	89	220	404	22	210	384	325	279	415	349	285	217
63	113	101	280	79	163	157	235	131	209	57	176	210	23	186	117	75	231	165	81	85
64	297	234	391	107	322	331	254	302	368	149	106	384	186	24	69	191	59	35	125	167
65	228	175	412	38	240	272	210	233	286	80	79	325	117	69	25	122	122	56	56	108
66	129	176	349	121	232	226	187	98	278	132	161	279	75	191	122	26	244	178	66	160
67	348	265	422	152	314	362	313	344	360	193	165	415	231	59	122	244	27	66	178	198
68	276	199	356	86	287	296	266	289	333	127	141	349	165	35	56	178	66	28	112	132
69	188	182	355	68	238	232	154	177	284	100	95	285	81	125	56	66	178	112	29	128
70	150	67	204	70	155	164	282	216	201	28	187	217	85	167	108	160	198	132	128	30
71	65	42	182	137	65	85	321	141	111	95	254	138	92	255	175	161	286	220	167	88
72	341	278	435	151	366	375	298	346	412	193	103	428	230	44	113	235	77	79	169	211
73	184	271	417	239	300	249	168	108	321	241	279	310	184	309	240	118	362	296	179	269
74	67	146	292	135	175	147	249	57	221	131	215	200	74	245	176	62	287	232	120	159
75	221	251	424	137	307	301	95	190	353	169	117	354	150	169	125	92	228	181	69	197
76	169	105	116	242	57	118	437	245	72	200	359	169	208	327	280	277	358	292	283	172
77	108	191	337	165	220	188	190	43	266	161	216	241	104	246	177	55	299	233	121	189
78	45	139	273	228	121	60	314	81	132	189	308	112	158	335	266	155	380	314	213	182
79	167	79	77	205	97	185	435	243	111	163	322	238	206	288	243	275	319	253	281	135
80	29	80	151	222	84	71	102	25	117	29	20	457	19	98	159	147	96	20	125	88
81	0	107	241	190	124	80	316	76	152	157	283	133	113	297	228	129	348	276	188	150
82	107	0	148	137	88	127	336	183	134	95	254	180	101	234	175	176	265	199	182	67
83	241	148	0	374	171	259	509	317	217	232	491	312	280	391	412	349	422	356	355	204
84	190	137	374	0	202	234	222	192	248	42	117	287	79	107	38	121	152	86	68	70
85	124	88	171	202	0	61	392	202	46	160	319	112	163	322	240	232	314	287	238	155
86	80	127	259	234	61	0	386	141	72	167	351	55	157	331	272	226	362	296	232	164
87	316	336	509	222	392	386	0	233	438	254	202	439	235	254	210	187	313	266	154	282
88	76	183	317	192	202	141	233	0	213	188	272	193	131	302	233	98	344	289	177	216
89	152	134	217	248	46	72	438	213	0	206	365	89	209	368	286	278	360	333	284	201
90	157	95	232	42	160	167	254	188	206	0	159	220	57	149	80	132	193	127	100	28
91	283	254	491	117	319	351	202	272	365	159	0	404	176	106	79	161	165	141	95	187
92	133	180	312	287	112	55	439	193	89	220	404	0	210	384	325	279	415	349	285	217
93	113	101	280	79	163	157	235	131	209	57	176	210	0	186	117	75	231	165	81	85
94	297	234	391	107	322	331	254	302	368	149	106	384	186	0	69	191	59	35	125	167
95	228	175	412	38	240	272	210	233	286	80	79	325	117	69	0	122	122	56	56	108
96	129	176	349	121	232	226	187	98	278	132	161	279	75	191	122	0	244	178	66	160
97	348	265	422	152	314	362	313	344	360	193	165	415	231	59	122	244	0	66	178	198
98	276	199	356	86	287	296	266	289	333	127	141	349	165	35	56	178	66	0	112	132
99	188	182	355	68	238	232	154	177	284	100	95	285	81	125	56	66	178	112	0	128
100	150	67	204	70	155	164	282	216	201	28	187	217	85	167	108	160	198	132	128	0

Collection @ chosun

저작물 이용 허락서

학 과	전산통계학과	학 번	20077321	과 정	박사
성 명	한글: 강 래 구 한문 : 姜 來 求 영문 : Rae-Goo Kang				
주 소	광주광역시 북구 임동 평화맨션 2동 508호				
연락처	E-MAIL : kangrg@mw.go.kr				
논문제목	한글 : 최적의 TSP 해결을 위한 유전자 알고리즘의 새로운 집단 초기화 및 순차변환 기법 제안과 구현 영어 : The Proposal and Implementation of New Population Initialization and Sequential Transformation Method for Genetic Algorithm to Find Optimal TSP				

본인이 저작한 위의 저작물에 대하여 다음과 같은 조건아래 조선대학교가 저작물을 이용할 수 있도록 허락하고 동의합니다.

- 다 음 -

1. 저작물의 DB구축 및 인터넷을 포함한 정보통신망에의 공개를 위한 저작물의 복제, 기억장치에의 저장, 전송 등을 허락함
2. 위의 목적을 위하여 필요한 범위 내에서의 편집·형식상의 변경을 허락함. 다만, 저작물의 내용변경은 금지함.
3. 배포·전송된 저작물의 영리적 목적을 위한 복제, 저장, 전송 등은 금지함.
4. 저작물에 대한 이용기간은 5년으로 하고, 기간종료 3개월 이내에 별도의 의사표시가 없을 경우에는 저작물의 이용기간을 계속 연장함.
5. 해당 저작물의 저작권을 타인에게 양도하거나 또는 출판을 허락을 하였을 경우에는 1개월 이내에 대학에 이를 통보함.
6. 조선대학교는 저작물의 이용허락 이후 해당 저작물로 인하여 발생하는 타인에 의한 권리 침해에 대하여 일체의 법적 책임을 지지 않음
7. 소속대학의 협정기관에 저작물의 제공 및 인터넷 등 정보통신망을 이용한 저작물의 전송·출력을 허락함.

동의여부 : 동의(O) 반대( )

2009년 8월

저작자 : 강 래 구 강래구 (인)

조선대학교 총장 귀하