유전 알고리즘을 이용한
지수귀문도 풀이

지도교수 **:** 장병탁

이 논문을 공학학사 학위 논문으로 제출함**.**

2002 년 **6** 월 **21** 일

서울대학교 공과대학

컴 퓨 터 공 학 부

김 동 민

2002 년 6 월

1. 서론

유전 알고리즘(Genetic Algorithm)은 자연 세계의 진화 과정을 모방해 문제 풀이 또는 모의 실험에 이용하는 연구의 한 방법이다. 유전 알고리즘은 풀고자 하는 문제에 대한 해들을 염색체로 표현한 다음 이들을 점차적으로 변형함으로써 점점 더 좋은 해들을 생성해 낸다. 각각의 해를 개체(individual)로 보고, 이 해들의 집합을 개체군(population)이라고 부른다. 세대가 지날수록 품질이 좋은 해를 생성하기 위해 유전 알고리즘에서는 교차와 돌연변이를 주된 연산자로 사용한다.

한편, 유전 알고리즘은 방대한 문제 공간 탐색 능력을 갖고 있으며 교차를 통해 이미 생성된 해들의 특성을 잘 활용할 수 있는 장점이 있다. 그러나, 교차와 변이 연산이 임의로 일어나기 때문에 품질은 더 좋아질 수도 있지만 더 나빠질 수도 있다. 이를 보완하기 위해 교차와 변이로 생성된 해에 지역 최적화 알고리즘을 적용하면 지역 최적점(local optimum)으로 접근하는 시간을 단축시킬 수 있다.

추상적으로 표현한 유전 알고리즘의 구조는 다음과 같다.

```
초기해 n 개를 임의로 생성;

{
   for i = 1 to k
   {
      두 염색체 p1, p2 선택;
      i 번째 자식해 = 교차(p1, p2);
      i 번째 자식해 = 변이(i 번째 자식해);
      i 번째 자식해 = 지역 최적화(i 번째 자식해);
   }

   이렇게 생성된 k 개의 자식해들을 개체군 내의 염색체들과 대치;

} (정지 조건을 만날 때까지 반복)
```

유전 알고리즘은 특히 결정론적인 방법으로 좋은 해를 구할 수 없는 NP-Hard 문제의 해결에 유용하다. 유전 알고리즘을 이용하여 풀 수 있는 대표적인 문제로는 TSP(Traveling Salesman Problem), 그래프 분할, 신경망 최적화, 퍼지 시스템 최적화, 베이지안망의 최적화 등의 문제를 꼽을 수 있다. 본 논문은 수학의 대표적인 난제인 ‘지수귀문도’ 문제에 유전알고리즘을 도입하여 최적의 답을 찾아 보는 것을 목표로 한다.

지수귀문도란, 동양 전래의 수학과 철학을 담은 마법진이다. 지수귀문도는 서로 맞물려 있는 육각형들의 구조를 가지고 있는데, 각각의 육각형을 구성하는 여섯 개 꼭지점 수의 합이 모두 일정하다. 꼭지점들은 모두 다른 수를 가지면서, 1 부터 총 꼭지점 수까지의 숫자가 정확히 한 번씩만 나타나야 한다. 이때 총 꼭지점 수가 같은 문제에 대해 조건을 만족시키는 지수귀문도가 다수 개 존재할 수도 있다.

지수귀문도는 꼭지점의 개수에 따라 수많은 문제의 종류가 존재하는데, 여기서는 그 중에서도 꼭지점이 각각 24 개, 54 개, 80 개, 110 개인 지수귀문도에 대해 실험을 하도록 한다. 앞의 두 문제는 교차와 변이만으로도 육각형의 합이 똑 같은 지수귀문도를 찾아낼 수 있지만, 뒤의 두 가지 문제는 단순히 유전알고리즘 뿐만 아니라 지역최적화 알고리즘을 적절히 병행해야만 좋은 해를 얻을 수 있는 정도의 적절한 문제 공간을 가지고 있다.

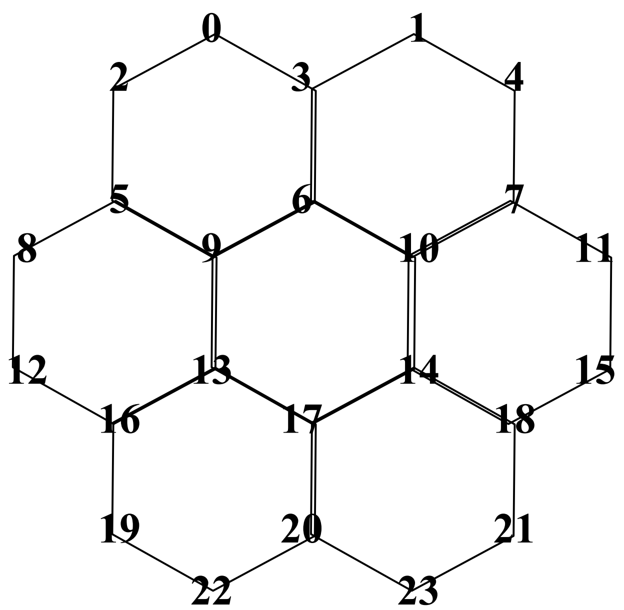
이 논문은 총 네 개의 장으로 이루어져 있다. 다음 장에서는 핵심 연구 내용에 대해 소개할 예정인데, 여기서는 유전알고리즘이 어떻게 지수귀문도 문제 풀이에 적용될 수 있는지 설명할 것이다. 제 3 장에서는 개체군의 크기, 교차율, 돌연변이율, 지역최적화 알고리즘 수행 횟수 등의 인자를 변화시켜가며 직접 실험을 해 보고, 그 결과에 대해 분석하였다. 제 4 장에서는 연구 결과를 요약하고 향후 과제에 대해 언급하는 것으로 끝을 맺도록 하였다.

2. 핵심 연구 내용

1) 염색체 표현 방법 (Representation)

하나의 지수귀문도를 나타내는 염색체는 일차원 배열로 저장된다. 이 배열의 N 번째 값은 논리적으로 지수귀문도의 왼쪽 위부터 N 번째 꼭지점의 숫자를 나타낸다.

[Fig.1] 24 지수귀문도의 염색체 인덱싱의 예

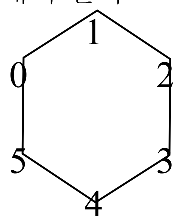


[Fig.2] 24 지수귀문도가 실제로 배열에 저장되는 모습

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**………..**	**22**	**23**

실제 저장되는 배열을 기준으로 하지 않고, 원하는 육각형을 기준으로 꼭지점에 접근할 때에는, 육각형의 인덱스를 가지고 할 수 있도록 내부적으로 구현하고 있다. 즉 육각형들도 역시 왼쪽 위부터 0 부터 시작되는 인덱스를 가지고 있고, 각 육각형에서는 오른쪽 그림과 같이 왼쪽 위의 꼭지점부터 다시 상대적인 인덱스를 가지고 있는 것이다.

[Fig.3] 육각형의 상대적 인덱스



2) 적합도 함수 (Fitness Function)

사이즈가 N 인 임의의 지수귀문도의 적합도는 적합도 = μ - σ$\sqrt{N}$ 로 정의한다. 여기서,

$$\sigma = \sqrt{\sum_{i=1}^{m} \frac{(Hi - \mu)^2}{m}}$$

m: 육각형의 총수
Hi: I 번째 육각형의 합
μ: 육각형들의 평균합

이다.

적합도의 정의에 따르면, 각 육각형들의 합이 평균값은 크되 표준편차는 작은 해일수록 좋은 품질을 갖는다.

3) 개체 선택법 (Selection)

선택 연산자는 교차에 쓰이는 두 개의 부모해를 고르기 위한 연산자이다. 해의 품질은 위에서 정의된 적합도 함수에 의해 평가되며, 우수한 해가 선택될 확률이 높게 해야 해집단의 평균 품질이 좋아지게 된다. 단, 우수한 해를 너무 차별하면 설익은 수렴(premature convergence)가 일어날 수도 있다. 선택 연산자에는 비례 선택법, 순위 선택법, 토너먼트 선택법이 있는데, 이 논문에서는 비례 선택법을 이용하도록 한다.

4) 교차 연산자 (Crossover)

기본적으로 일점 교차를 이용하는데, 0 부터 1 사이의 임의의 수를 발생시켜, 그 수가 교차율보다 작으면 교차가 일어난다. 개념적으로 염색체에서 임의의 한 점을 잡아 그 점을 중심으로 한 쪽은 부모 1 에서, 다른 쪽은 Parent 2 에서 가져와서 새로운

자식해를 만든다. 이 때 자식해에 나타나는 꼭지점 상의 숫자들은 중복될 가능성이 있다. 그래서 기존의 값을 최대한 반영하면서도 같은 숫자가 존재하지 않도록 수정해야 한다. 예를 들어, 값 2 가 3 개 존재하고, 동시에 값 3 이 존재할 경우에는 3 개의 값 2 를 각각 2, 3, 4 로 바꾸고, 값 3 은 그것을 고려해서 5 로 바뀌게 되며 그 외의 숫자들도 다 일정하게 증가하게 되어서 원래의 상대적 크기를 지켜주게 되는 것이다.

여기서는 염색체를 위의 왼쪽 꼭지점부터 순차적으로 1 차원 배열에 저장하고 있고, 이를 일점 교차하기 때문에, 실제적으로 교차점들은 지수귀문도의 모양에서 보자면 가로방향에 가까운 모양으로 일어나게 된다. 이는 교차 지점이 다양하지 못하게 하는 결과를 가져오므로, 이를 보안하기 위해서 염색체의 회전 기능을 추가하였다. 즉, 교차하기 전에 염색체를 임의의 각도만큼 회전시켜준 후, 가로로 자르기 때문에, 가로-세로 방향을 모두 포함하는 자름선을 확보할 수 있었다. 실제로 실험결과에서 회전을 구현한 후에 결과값이 상당히 좋아지는 것을 확인할 수 있었다.

하지만, 육면의 모양이 대칭인 24 와 54 개의 지수귀문도와는 다르게 80, 110 에서는 지수귀문도 전체 육면의 모양이 같지 않으므로 통상적인 회전을 사용할 수 없는 약점을 가진다.

5) 변이 연산자 (Mutation)

변이를 구현하기 위해, 임의로 선택된 꼭지점에 대해서 변이 확률에 따라서 vertex 개수의 최대값 내의 임의의 숫자로 바꾼다. 이 경우에도 숫자가 바뀐 후에는, 기존의 숫자로 겹쳐질 수 있으므로 값을 조정해야 한다.

6) 대치 연산자 (Replacement)

최종적으로 생성된 자식해들은 개체군 내의 염색체들과 대치되는데, 여기서는 적합도 값이 가장 낮은 염색체부터 순차적으로 대치하도록 했다.

7) 지역 최적화 알고리즘 (Local Optimization)

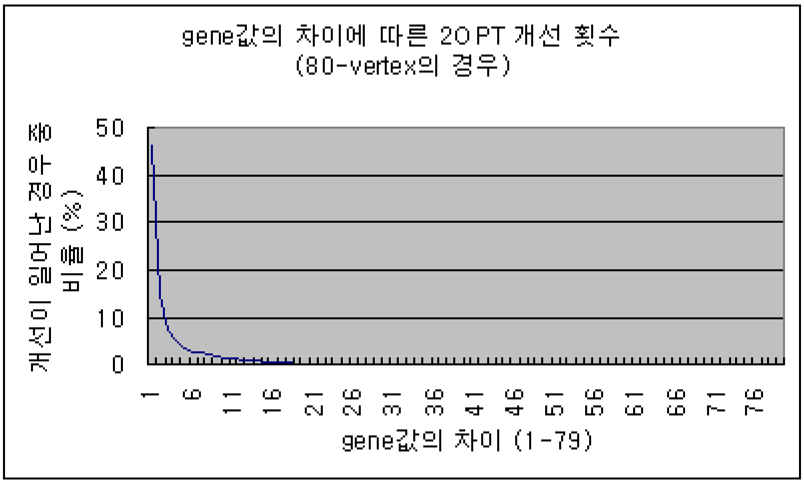
교차, 변이 연산을 한 자식해에 대해 지역 최적화 알고리즘을 적용할 수 있는데, 여기서는 2-OPT 를 비롯한 여러 알고리즘을 도입해 보았다.

2-OPT

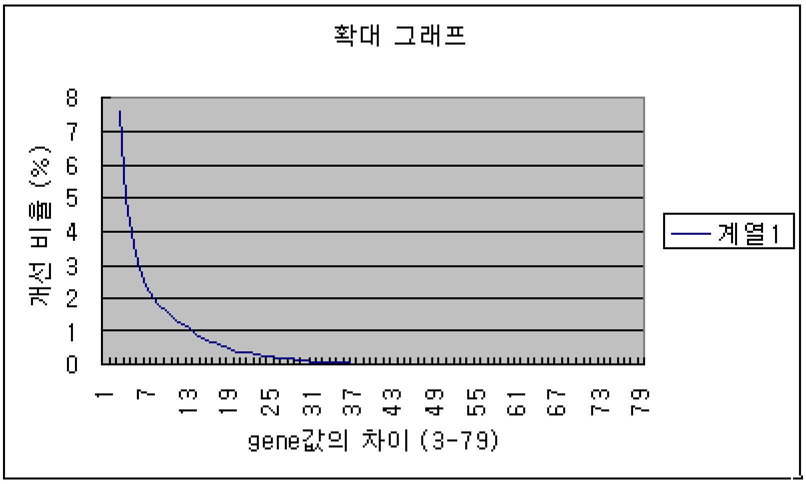
2-OPT 는 임의의 두 꼭지점을 바꾸어서 적합도를 계산해 본 후, 이전의 적합도보다 개선되면 그 결과를 반영하는 지역 탐색 알고리즘이다. 2-OPT 는 일정한 횟수만큼 반복해서 시도하도록 했는데, 여기에는 개선의 여지가 있다.

2-OPT 를 통해 품질 향상이 일어나는 경우의 패턴에 대해 생각해 보자. 세대가 거듭될수록 새롭게 생성되는 자식해는 이미 상당 부분 수렴되어 있는 부모해의 성질을 부분적으로 물려받게 되므로, 전역 최적해(global optimum)에 가까워져 있을 것이다. 이러한 상황에서 값의 차이가 큰 두 꼭지점을 바꾸어서 품질이 향상될 확률은 극히 작을 것이다. 이를 실험으로 증명한 것이 [Fig.4]에 나타나 있다.

[Fig.4] 2-OPT 에서 개선이 일어나는 경우



[Fig.5] 위의 그래프 중 일부를 확대



값이 1 만큼 차이나는 경우가 모든 경우의 50% 가까이를 차지함을 확인할 수 있었다. 유효한 차이값을 알아보니 위해 차이값이 1 인 경우를 제외하고 다시 그래프를 그려 보았다.

관찰된 바에 따르면, 값의 차이가 10 이상 나면 개선되는 경우의 빈도는 1%도 되지 않는다. 2-OPT 를 여러 번 적용할수록 최적해에 가까운 해로 변하므로 거기서 차이가 크게 나는 꼭지점들끼리는 바꾸어서 적합도가 좋아지기가 지극히 어렵기 때문이다. 따라서 지역 최적화 알고리즘을 수행할 때, 이러한 특성을 반영하는 것이 필요하다.

임의로 두 개의 꼭지점을 선택하여 2-OPT 를 시도하는 현재의 알고리즘을 비슷한 꼭지점들끼리 바꾸는 방향으로 개선하기 위해서는 우선 기존의 염색체에 들어 있는 인덱스-값의 대응 관계를 거꾸로 다른 배열에 저장해야 한다. 그런 다음, 그 배열에서 인덱스의 차이가 1 또는 2 만큼 나는 꼭지점(실제 염색체에서는 값의 차이가 1 또는 2 인 꼭지점) 두 개를 선택해서 2-OPT 알고리즘을 적용하는 방식으로 구현할 수 있다.

Hexagon Rotation Opt

각각의 작은 육각형을 회전시켜서, 모든 회전 가능성(6 가지)에 대해서 가장 적합도가 큰 모양으로 바꾼다. 위력이 2-OPT 보다는 많이 떨어진다. 실험에서는 2-OPT 후에 hexagon rotation opt 를 하도록 하였으며, 이 경우에 약 100~200 개의 해마다 한 번씩 최적화가 이루어지는 것을 확인할 수 있었다. 이는 2-OPT 를 한 후에 시행한 것이라, 약간의 성능 향상이 있었음을 알 수 있다.

Big Hexagon Rotation Opt

작은 육각형의 모양이 아니라, 지수귀문도 전체의 바깥 외각선을 따른 모양의 육각형을 회전시키는 것을 말한다. 또한 가장 바깥 외각선의 안쪽 외각선, 그 안쪽의 외각선 등 작은 육각형들을 포함하는 큰 육각형의 회전시켜서, 그 모든 가능성 중 가장 적합도를 크게 하는 모양으로 바꾸는 방법이다. 약간의 성능 향상이 있으나, 수행 시간에 비하면 그 효과는 미미하다.

기타

2 Hexagon Opt 라고 이름 붙힌 최적화를 시도하였으나, 논리적 오류가 있음을 발견하여 잠시 사용을 보류하였다. 이것은 2-OPT 를 응용하여, 2 개의 작은 육각형을 교환하여, 더 적합도가 좋을 경우에 바꾸는 것으로서, 지수귀문도의 특징을 살리면서도 2-OPT 가 확률적으로 일어나기 힘든 경우를 보완할 것으로 기대하였다. 그러나 서로 맞닿아있는 경우에 겹쳐져 있는 꼭지점의 값은 두 번 중복해서 복사가 일어나고, 일부 값은 손실이 되는 문제점이 일어남을 발견하여 더 이상 사용하지 않았다. 하지만 지수귀문도의 크기가 클 경우에는 서로 맞닿아있는 경우를 제외하고 이용하면 유용할 것으로 판단된다.

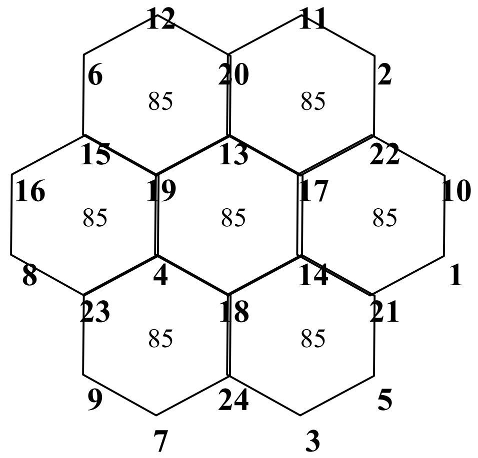
3. 실험 및 결과

꼭지점이 24 개, 54 개, 80 개일 때는 육각형의 합이 모두 똑같은 지수귀문도를 찾아낼 수 있었다. 110 개일 때는 답을 찾지 못했다.

1) 꼭지점 24 개 지수귀문도

10 번 수행한 결과 평균 22.9 초만에 85 를 찾아낸다. 이 때 나타날 수 있는 지수귀문도 중 하나를 제시하면 다음과 같다.

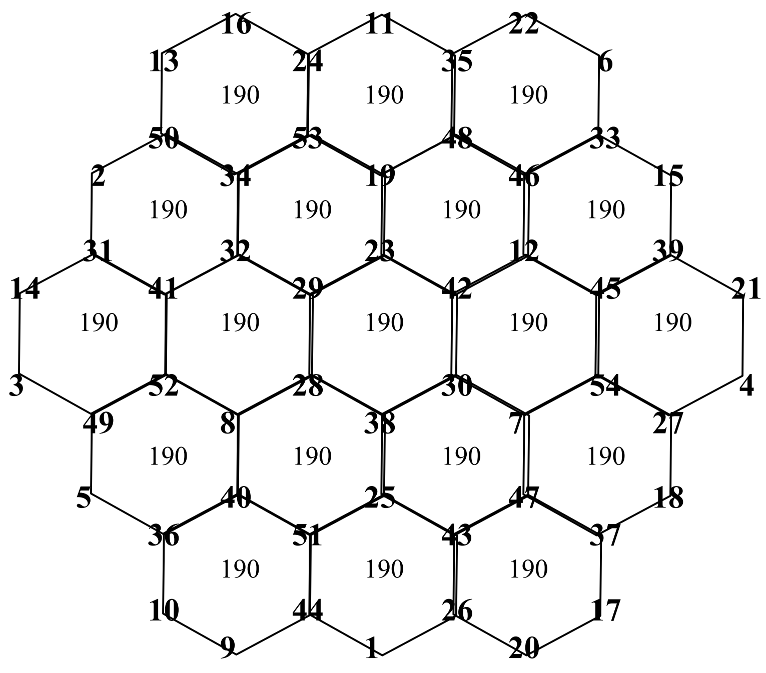
[Fig.6] 꼭지점 24 개 지수귀문도의 예 중 하나 (합이 85)



2) 꼭지점 54 개 지수귀문도

평균 2 분 47 만에 육각형의 합이 모두 190 인 지수귀문도를 찾아냈다.

[Fig.7] 꼭지점 54 개 지수귀문도의 예 중 하나 (합이 190)



3) 꼭지점 80 개 지수귀문도

이 문제에 대해서는 합이 281 인 지수귀문도를 찾아냈다. 하지만 앞의 두 문제처럼 수행할 때마다 최적화된 해를 찾아내지 못했다. 따라서 이 문제에 대해서는 여러 가지 인자값들을 변화시켜 가면서 수행해 보았다. 앞서 언급했듯이, 실험에 사용되는 주요 인자는 개체군의 크기, 교차율, 돌연변이율, 지역최적화 알고리즘 수행 횟수 등이다.

[Table 1] 개체군의 크기 (교차율 0.99, 변이율 0.03, 2-OPT 10000 회)

	평균
2	259.22
10	279.43
30	279.43
100	278.76

[Table 2] 교차율 (개체군의 크기 30, 변이율 0.03, 2-OPT 10000 회)

	평균
0.3	278.43
0.6	279.43
0.9	279.43
0.99	279.43

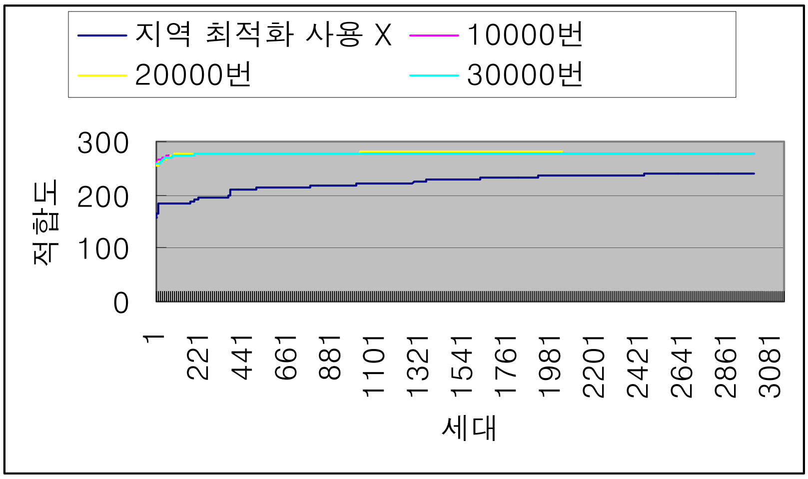
[Table 3] 변이율 (개체군의 크기 30, 교차율 0.99, 2-OPT 10000 회)

	평균
0	273.36
0.03	279.43
0.1	277.69
0.3	272.5

[Table 4] 2-OPT 수행횟수 (개체군의 크기 30, 교차율 0.99, 변이율 0.03)

	1	2	3	4	5	평균	수행 시간
0	241.55	235.19	242.03	236.71	240.39	239.174	04:37
10000	278.43	278.43	280	277.95	280	278.962	24:07
20000	281	280	278.43	281	280	280.086	46:33
30000	278.43	281	279.43	279.43	281	279.858	69:37

[Fig.8] 세대에 따른 적합도의 변화



크기가 80 인 지수귀문도의 경우, 개체군의 크기, 교차율, 변이율은 큰 영향을 미치지 못했다. 개체군의 크기가 너무 크거나(예를 들어 1000), 변이율이 아주 높은 경우(예를 들어 0.5)를 제외하고는 거의 비슷한 결과를 나타냈다. 반면, 하나의 자식해에 대해 수행하는 2-OPT 의 횟수는 평균 적합도 뿐만 아니라 수행 시간에도 결정적인 영향을 주는 인자였다. 특히 2-OPT 를 한 번도 수행하지 않은 경우와 10000 번 수행한 경우는 결과값에서 엄청난 차이를 보였다. 이는 지역 최적화 알고리즘이 해를 지역 최적점(local optimum) 근처로 이끄는 효과가 무척 강력함을 보여주는 증거이다.

4) 꼭지점 110 개 지수귀문도

꼭지점이 110 개인 경우에도 2-OPT 의 수행 횟수만이 유효한 변화를 유발하는 인자임을 확인할 수 있었다. 2-OPT 를 많이 수행할수록 결과는 당연히 더 향상되었다. (2-OPT 로 인해 적합도가 떨어지는 일은 없다.) 그러나 20000 번 수행할 때와 30000 번 수행할때는 거의 비슷한 결과임에도 수행 시간에서는 많은 차이가 났다.

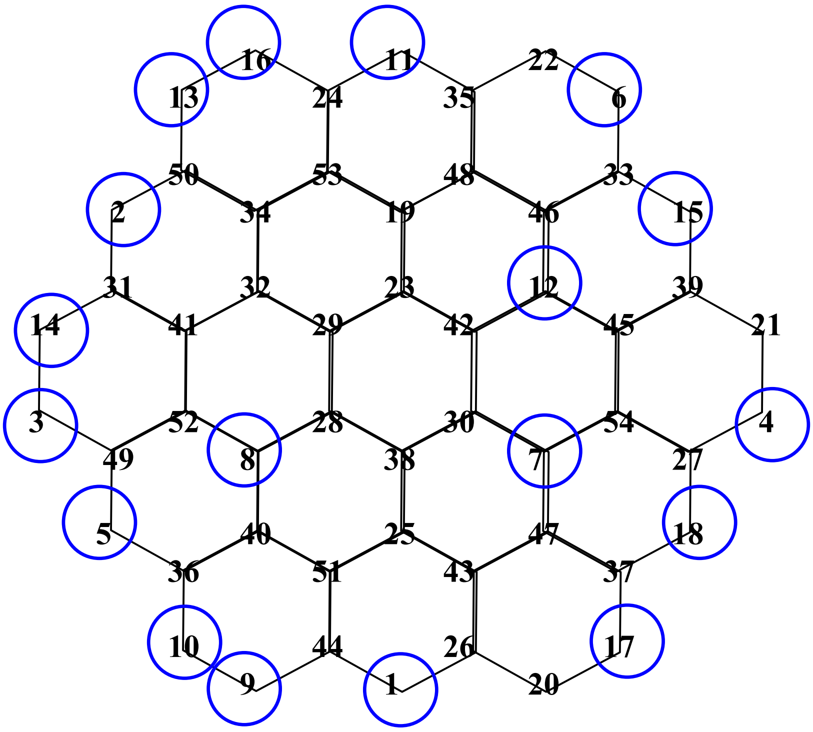
[Table 5] 2-OPT 수행횟수 (개체군의 크기 30, 교차율 0.99, 변이율 0.03)

	1	2	수행 시간
5000	363.1	361.6	18:31
10000	366.9	367.3	35:10
20000	374.2	373.5	62:47
30000	374.1	374.2	91:02

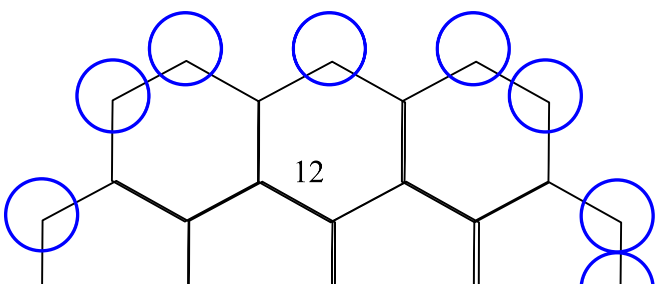
5) 최적해에 대한 고찰

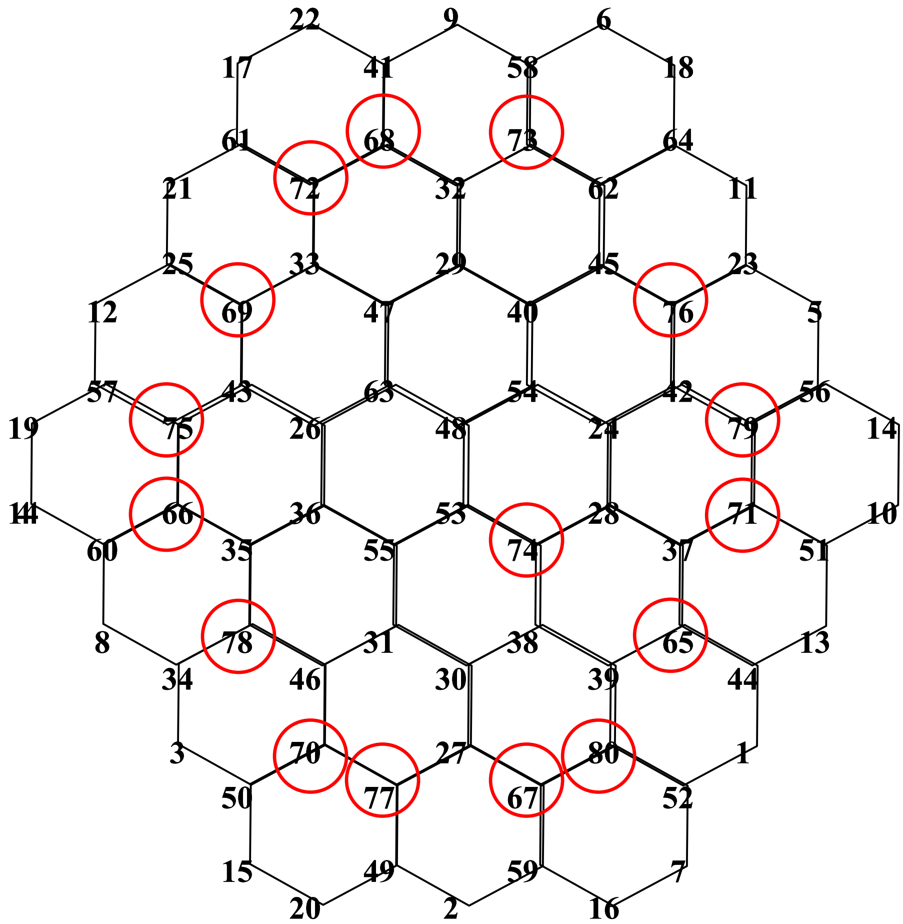
현재까지 최적해를 구한 54, 80 개의 지수귀문도를 살펴보자면, 최적해는 어느 정도 일정한 패턴을 보이고 있다. 먼저 알 수 있는 경향은 최소값을 가지는 꼭지점들은 지수귀문도의 바깥쪽에 분포된다는 것이다.

[Fig. 9] 크기 54 에서 가장 작은 값 18 개의 분포



[Fig.10] 크기 80 에서 가장 작은 값 23 개의 분포

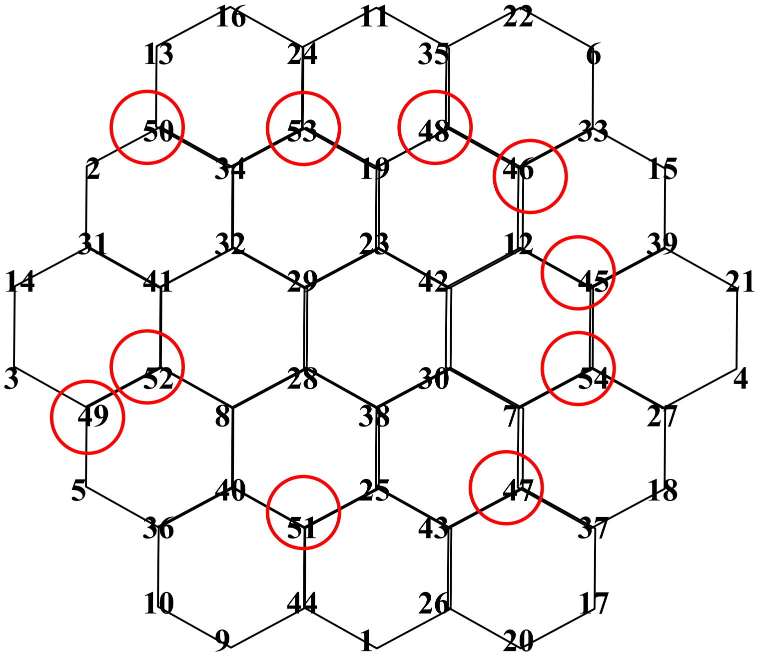




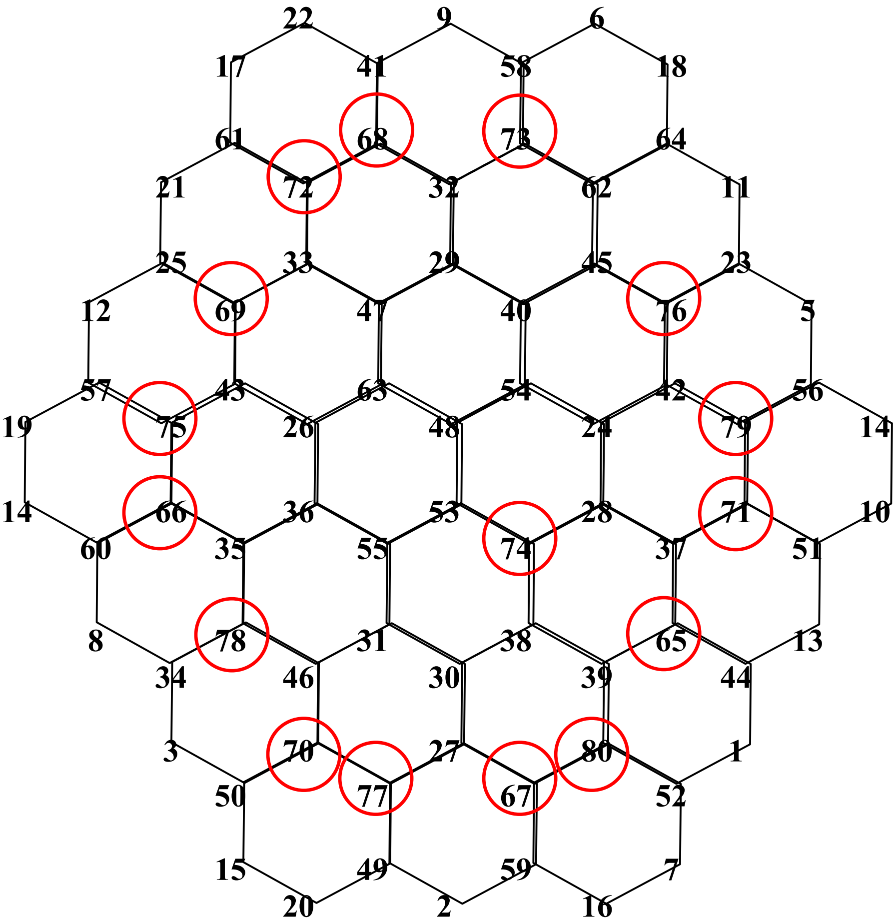
각 꼭지점들은 육각형의 합에 계산이 될 때, 한 개의 육각형 합에 들어갈 수도 있고, 두 개나 세 개의 육각형의 합에 더해질 수도 있다. 이를 각각 1-Hexa, 2-Hexa, 3-Hexa 인 꼭지점이라고 하자. 이를 가지고 위의 결과를 살펴보면, 최소값들은 거의 모두 1-Hexa 인 꼭지점이 되는 것을 알 수 있다. 적합도를 최대화시키기 위해서 작은 수는 전체 합에 안 좋은 영향을 가장 적게 미치도록 가능한 적은 육각형의 합에만 계산되어야 함을 알 수 있다.

반면에 가장 큰 수들의 분포를 살펴보도록 하자. 꼭지점이 54 개인 지수귀문도의 최적해를 살펴보면 흥미로운 현상을 알 수 있다. 적합도를 최대화시키기 위해서는 3-Hexa 인 꼭지점이 될 것으로 언뜻 생각할 수 있으나, 오히려 2-Hexa 인 Gene 이 되고 있다. 최대값들이 가운데로 다 몰리지 않고, 오히려 약간 퍼지는 이 현상은 꼭지점 80 개인 지수귀문도에서 더욱 두드러진다.

[Fig. 11] 크기 54 에서 가장 큰 값 10 개의 분포



[Fig. 12] 크기 80 에서 가장 큰 값 16 개의 분포



이

는 몇가지 이유로 생각해 볼 수 있다. 값이 큰 꼭지점이 너무 가운데로 몰리면 특정한 육각형들만 합이 커지기 때문에, 3-Hexa 면서도 가장 골고루 합을 증가시키는 경향을 보인다. 즉 지수귀문도를 하나의 원으로 생각해보면, 큰 원의 절반 정도의 지름을 가지는 절반 크기의 원으로 가장 큰 꼭지점들이 몰리게 되어, 원 전체의 무게를 가장 골고루 분포시킨다고 생각할 수 있다.

이러한 최적해의 패턴을 성능 향상에 활용할 수도 있을 것이다. 임의로 초기 해들을 생성하는 것 뿐만 아니라, 외곽에 가장 작은 값들을 배치하고 가장 큰 값들은 중간 깊이에 할당하는 방식으로 일부 해들을 생성해 해집단에 넣어줄 수도 있다.

4. 결론

순수한 Genetic Algorithm 을 사용한 경우와 비교했을 때, 지역 최적화 알고리즘의 역할이 차지하는 비율이 절대적인 것이 인상적이다. 하지만, 진화 알고리즘의 연산자들도 어느 정도 역할을 하고 있음을 확인할 수 있었다. 특히 변이율의 경우, 매우 작게 느껴지는 0.03 이라도 변이가 전혀 일어나지 않게 한 경우와 비교했을 때 훨씬 뛰어난 성능을 확인할 수 있다.

2-OPT 에서 적합도 개선이 일어나는 패턴을 발견한 후, 이를 도입하고 필요에 맞게 사용하느라 이후에는 다양한 지역 최적화 알고리즘들을 충분히 개선시켜보지 못한 것이 아쉬웠다. 앞에서 2-OPT 를 이용한 방식은 일정 횟수 동안 반복해서 2-OPT 를 수행하는 방식이었다. 그러나 이럴 경우 가지는 한계가 존재한다. 예를 들어 a-b-c 와 같은 배치가 존재한다고 하자. 2-OPT 의 경우 a 와 b 를 바꿔본 후 적합도 향상이 있으면 a-b-c 를 b-a-c 로 바꾼다. 하지만 실제로 b 와 c 를 바꿔서 나타나는 a-c-b 의 배치가 더 큰 적합도 향상을 가져올 수도 있다. 이러한 점을 보완하기 위해 3-OPT 나 4-OPT 를 도입할 수도 있을 것이고, 가장 큰 적합도 향상을 보장하는 두 개의 꼭지점을 교환하는 방법도 생각해 볼 수 있다.

그리고, 꼭지점들이 가지는 값의 지역성(비슷한 값끼리 바뀔 가능성이 높다)와 지리학적 지역성(비슷한 크기의 들은 지수귀문도의 중심에서 비슷한 거리를 가진다)인 특징들을 둘 다 결합한다면 더 빠르고 나은 성능을 가진 최적화 알고리즘의 개발이 가능할 것이다. 또한 작은 차이가 나는 꼭지점에 대해서만 2-OPT 를 시도하는 방식은 분명 그 효과가 놀랍긴 하지만, 여전히 한계를 지니고 있다. 즉 이미 지역적으로 최적화되어서 고착화된 육각형들이 뭉쳐있을 경우, 이를 통째로 옮기기가 쉽지 않다.

앞서 잠깐 언급했듯이 2-OPT 로 인해 세대가 지나면 해집단이 수렴되는 현상을 발견할 수 있다. 이는 지역 최적화 알고리즘의 강력한 효과 때문인데, 해집단의 다양성을 유지할 수 있도록 하는 효과적인 방법들이 요구된다고 할 수 있다. 시간이 지날수록 변이가 일어나는 확률을 감소시키는 비균등 변이(non-uniform mutation)과는 반대로, 세대가 흘러 일정한 비율 이상 같은 해가 해집단에 존재하는 경우 변이율을 높여주는 방법도 생각해 볼 수 있다.

참고문헌

문병로, *유전알고리즘*, 다성출판사, 2001.

"300 년만에 풀린 최석정의 마법진", 과학동아 1999 년 12 월호.

Chickering, D.M. *Learning Bayesian Networks is NP-Hard.*, Redmond, WA, Microsoft Research., 1994.

Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

Holland, J.H., *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, M.I., 1975.

Nills J. Nilsson, Morgan Kaufmann, *Artificial Intelligence : A New Synthesis*, 1998.

Syswerda, G., *Uniform Crossover in Genetic Algorithms*, Proceedings of the 3rd International Conference on Genetic Algorithms, Morgan Kaufman., 1989.

Winston, P.H., *Artificial Intelligence*, Addison-Wesely, 1992.

B.T. Zhang, *Learning and Optimization by Artificial Evolution* (in Korean), ICASE Magazine, 1(3):52-61, 1995.