IE Interfaces
Vol. 20, No. 4, pp. 504-514, December 2007.

멀티캐스트 라우팅을 위한 다목적 마이크로-유전자 알고리즘

전성화† · 한치근

경희대학교 컴퓨터공학과

Multi-Objective Micro-Genetic Algorithm for Multicast Routing

Sunghwa Jun · Chigeun Han

Department of Computer Engineering, KyungHee University, Yongin, 446-701

The multicast routing problem lies in the composition of a multicast routing tree including a source node and multiple destinations. There is a trade-off relationship between cost and delay, and the multicast routing problem of optimizing these two conditions at the same time is a difficult problem to solve and it belongs to a multi-objective optimization problem (MOOP).
A multi-objective genetic algorithm (MOGA) is efficient to solve MOOP. A micro-genetic algorithm(μGA) is a genetic algorithm with a very small population and a reinitialization process, and it is faster than a simple genetic algorithm (SGA).
We propose a multi-objective micro-genetic algorithm (MOμGA) that combines a MOGA and a μGA to find optimal solutions (Pareto optimal solutions) of multicast routing problems. Computational results of a MOμGA show fast convergence and give better solutions for the same amount of computation than a MOGA.

Keyword: micro-genetic algorithm, multicast routing problem, multi-objective optimization

1. 서 론

최근 통신망의 발달과 함께 인터넷을 이용한 다양한 서비스가 제공되고 있다. 과거에는 인터넷을 통한 데이터 중심의 서비스가 주를 이루었지만, 최근에는 화상 회의, VOD 서비스 등과 같은 멀티미디어 서비스가 중심이 되고 있어 QoS(Quality of Service)가 무엇보다도 중요하게 되었다(Wang and Hou, 2000). 특히 VOD 서비스, 동영상 강의와 같은 인터넷 서비스는 하나의 단말에서 다수의 단말로 서비스 되는 멀티캐스트 기법을 사용한다. 그래프로 모델링 되었을 때 시작 노드와 목적지 노드를 연결하는 트리 형태의 멀티캐스트 경로를 선택하는 문제를 멀티캐스트 라우팅이라고 한다.

멀티캐스트에서 QoS를 판단하는 파라미터로는 혼잡도, 평균 트래픽비율, 네트워크 거리, 비용, 실시간 서비스를 위한 최소 지연 조건 등이 있으며, 이것들은 에지의 가중치로 표현될 수 있다. 이들 파라미터들은 각각의 서비스에 따라 중요도가 달라질 수 있으며, 그 파라미터들은 서로에게 영향을 주는 Trade-off 관계에 있다. 따라서 멀티캐스트 문제는 다수의 목적함수를 동시에 최적화 시키는 다목적 최적화 문제(Multi-Objective Optimization Problem: MOOP)에 속한다. 지금까지의 대부분의 연구는 일부 목적을 제약식으로 변환하거나, 각 목적 함수에 가중치를 부여하여 하나의 목적 함수로 표현하는 방법으로 처리하였다. 하지만 단일목적 최적화 문제로 변환하여 처리하는 방법은 의사결정자가 목적들 간의 상대적 중요도를 알고 있다는 것을 전제로 하고 있다. 따라서 목적이 많아지면 의사결정자가 목적들의 가중치를 구할 때 일관성을 유지하기가 쉽지 않다. 그래서 최근에는 의사결정자에게 다양한 대안 해를 제시하기 위해 다목적 유전자 알고리즘(Multi-Objective Genetic

†연락저자 : 전성화 박사과정, 446-701 경기도 용인시 기흥구 서천동 1번지 경희대학교 컴퓨터공학과, Fax : 031-202-1723,
E-mail : zolac@khu.ac.kr
2006년 07월 접수, 1회 수정 후 2007년 07월 게재확정.

Algorithm: MOGA)을 이용하여 MOOP를 처리하여 다수의 목적이 동시에 최적화된 파레토 최적해를 구하는 방법들이 연구되고 있다.

본 논문에서는 비용, 지연의 2가지의 파라미터를 최소화하는 멀티캐스트 라우팅 문제에 대해서 소수의 모집단으로 진화를 시켜서 단순 유전자 알고리즘(Simple Genetic Algorithm:SGA)보다 수렴 속도 효율이 좋은 마이크로 유전자 알고리즘(Micro-Genetic Algorithm: μGA)과 다목적 유전자 알고리즘(MOGA)가 결합된 다목적 마이크로 유전자 알고리즘(Multi-Objective Micro-Genetic Algorithm: MOμGA)을 적용하여 문제해결의 수렴 속도를 향상 시킬 수 있는 방법을 제안한다.

본 논문의 구성은 제 2장에서는 멀티캐스트 라우팅 문제에 대해서 정의하고, 제 3장에서 일반적인 유전자 알고리즘에 대해 설명한다. 제 4장에서 멀티캐스트 라우팅 문제를 위해 제안된 MOGA에 대해서 설명하고 제 5장에서는 다양한 문제에서 MOGA과 MOμGA 성능을 실험 계산을 통해 비교하며, 제 6장에서 결론 및 추후 연구 방향에 대해서 논의한다.

2. 멀티캐스트 라우팅 문제

2.1 문제 정의

멀티캐스트 라우팅 문제는 서비스를 제공하는 하나의 단말에서 서비스를 제공받는 다수의 단말로의 서비스 경로를 선택하는 문제이다.

멀티캐스트 라우팅 문제의 해법을 위한 네트워크는 그래프 G = (V, E, s, D)로 모델링 되며, 그래프 G = (V, E, s, D)는 노드의 집합 V와 에지의 집합 E, 시작 노드 s (∈V), 목적지 노드 D (⊂V)로 이루어진 무향, 연결, 가중그래프이다.

각각의 에지(e∈E)는 지연시간(Delay), 비용(Cost)을 가중치로 갖는다. 멀티캐스트 라우팅 문제는 전체 네트워크 비용이 최소가 되는 시작 노드와 목적지 노드를 포함한 식 (1), 식 (2)의 목적 함수를 동시에 만족하는 최소 스타이너 트리(Minimum Steiner Tree)를 구하는 문제와 같다. <Figure 1>은 멀티캐스트 라우팅 문제를 위한 전체 네트워크를 나타낸 하나의 예이다.

$$Minimize \sum_{e \in T} C(e) \quad (1)$$

$$Minimize \sum_{e \in T} D(e) \quad (2)$$

$C(e)$: Cost of e,
$D(e)$: Delay of e,
T : Multicast Tree

총 13개의 노드를 포함하고 있으며, 하나의 시작 노드(v0)와 세 개의 목적지 노드(v1, v2, v3)를 갖고 있으며, 에지에 나타난 가중치는 비용, 지연시간을 쌍으로 표현한 것이다. <Figure 2>는 <Figure 1>의 예에서 구성할 수 있는 멀티캐스트 라우팅 트리의 한 예를 나타낸다.

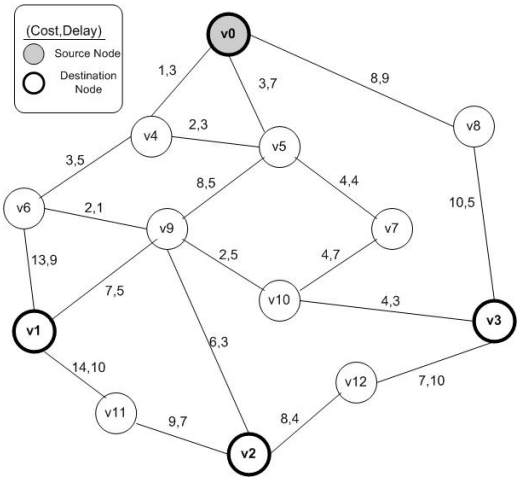

Figure 1. A Sample Network

이 예에서는 시작 노드에서 목적지 노드로 메시지를 전달하기 위해 v4, v6, v9, v10의 중간 노드를 경유하게 된다. 그리고 시작 노드에서 목적지 노드까지 에지들의 총 비용과 총 지연시간은 각각 25가 된다.

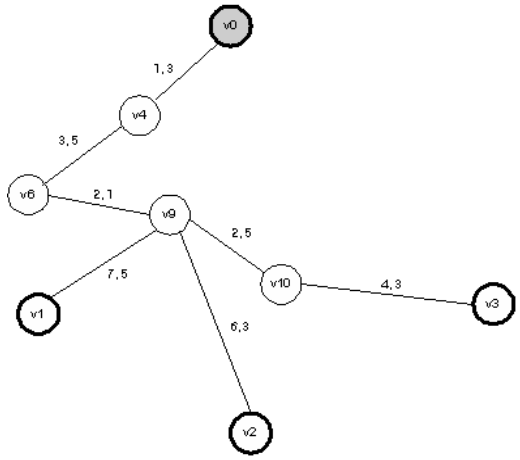

Figure 2. A Multicast Routing Tree of <Figure 1>

2.2 기존 해법

멀티캐스트 라우팅 문제는 전체 네트워크 비용을 최소화하는 스타이너 트리를 찾는 문제이며, 이는 NP-Complete 문제이다(Winter, 1987).

최소 스타이너 트리 문제를 해결하는 효율적인 휴리스틱 알고리즘으로 Kou, Markowsky 그리고 Berman에 의해서 제안된 KMB 알고리즘이 있다(Kou *et al.,* 1981). 또 지연 시간을 제한조건으로 두고 최소 스타이너 트리를 구성하는 KMB 알고리즘 기반의 KPP 알고리즘이 있다(Kompella *et al.,* 1993). 하지만 이러한 알고리즘은 지연, 노드의 차수 및 홉 수 등의 제한 조건을 이용하여 비용만을 최적화하는 알고리즘으로 두 개의 목적을 동시에 최적화시키지 못한다는 단점을 가지고 있다.

또한 멀티캐스트 라우팅 문제를 해결하기 위해 하나의 해가 아닌, 해 공간을 운용하는 방식의 다점 탐색을 함으로써 최적화 문제에 적합한 유전자 알고리즘이 제안되었다. (Esbensen, 1995)에서 유전자 알고리즘을 적용하였지만 최소 비용만을 고려한 해법이었고, (Tsai and Tasi, 2001; Ying and Jianping, 2002)에서는 지연 시간을 제한조건으로 두고 유전자 알고리즘을 적용하는 방법이 제안되었다. 이들 연구는 다점 탐색을 하는 유전자 알고리즘의 장점을 가졌지만, 멀티캐스트 서비스에 필요한 특성과 요구사항을 동시에 최적화 하지 못하는 한계가 있다.

이러한 다목적 최적화 문제의 다양한 파레토 최적해를 구해내는 MOGA가 연구되었다(Hyun and Kim, 1996). 하지만 MOGA는 파레토 최적해를 판단하고, 해 집단을 운용하는 과정에서 SGA보다 많은 계산량이 필요하다.

본 논문에서는 MOOP에 적합한 MOGA의 수렴 속도를 향상시키기 위해 기존 방법(MOGA)에 마이크로 유전자 알고리즘(μGA)을 적용한 다목적 마이크로 유전자 알고리즘(MOμGA)을 제안하였다.

3. 유전자 알고리즘

3.1 유전자 알고리즘

유전자 알고리즘은 다윈의 진화론과 적자생존을 기반으로 개발된 탐색 기법으로, 고전적인 최적화 기법에서 하나의 해를 운용하여 해 공간을 탐색하는 방법과는 달리 임의의 다수 해집합에서 시작하여 진화 탐색해 나가는 다점 탐색 방법을 사용하고 있다. 유전자 알고리즘은 주로 최적화 문제를 해결하는데 널리 사용되었고, NP-hard인 조합 최적화 문제를 해결하는데 효과적으로 적용되었다.

일반적으로 해결하고자 하는 문제의 특성에 따라 스트링으로 코드화된 유전자를 만들고, 각 세대는 일정한 수의 유전자들로 구성된다. 각각의 유전자는 문제에 대한 적합도를 이용하여 수치화된 값으로 우성과 열성을 나타내며 확률적으로 다음 세대로의 진화 여부를 결정한다. 결국, 우성의 성질이 다음 세대로 전파될 확률이 높으며, 다음 세대에서는 더 많은 우성의 유전자들이 존재하게 된다. 위와 같은 과정을 처리하기 위해서 선택연산(Selection Operation), 교차연산(Crossover Operation), 돌연변이연산(Mutation Operation)과 같은 연산자가 존재한다.

모집단은 매 세대마다 일정수의 개체를 유지하고 매 세대에서 각 개체의 적합도를 평가하고 다음세대에 생존할 개체들을 확률적으로 선별한다. 선별된 개체들을 임의로 짝을 지어 교차연산과 돌연변이연산을 통해 자식을 생성하고 그것들이 새로운 세대를 이룬다.

3.2 마이크로 유전자 알고리즘

μGA는 (Krishnakumar, 1989)에서 처음으로 실제적인 최적화 문제에 적용되었으며, 기존의 SGA와의 차이는 모집단의 크기에 있다.

SGA에서 모집단 수가 적다면 다양한 형질을 갖지 못하기 때문에 지역 최적해에 빠질 수 있는 단점이 있다. 따라서 일반적으로 30개에서 200개정도의 유전자로 모집단을 구성한다. 하지만 μGA는 모집단 수를 줄이면서도 지역 최적해에 빠지지 않게 하여 수렴 속도를 증가시킨다. 일반적인 μGA의 흐름도는 <Figure 3>과 같다.

μGA의 전체 모집단은 직접적으로 진화하지 않는다. 전체 모집단에서 선별된 소수의 유전자를 선별하여 마이크로 모집단을 생성하고, 생성된 마이크로 모집단만으로 SGA의 연산자들을 그대로 이용하여 진화를 시킨다.

마이크로 모집단이 수렴하면 수렴된 해들 중 가장 좋은 해를 저장하고, 전체 모집단을 갱신하고 다시 마이크로 모집단을 생성한다. 전체 모집단을 갱신하는 방법으로는, 모집단의 유전자 전체를 임의로 다시 생성하는 방법, 가장 좋은 해는 보존하고 나머지 유전자 전체를 임의로 다시 생성하는 방법, 일부는 보존하면서 일부는 다시 생성하는 방법 등이 있다.

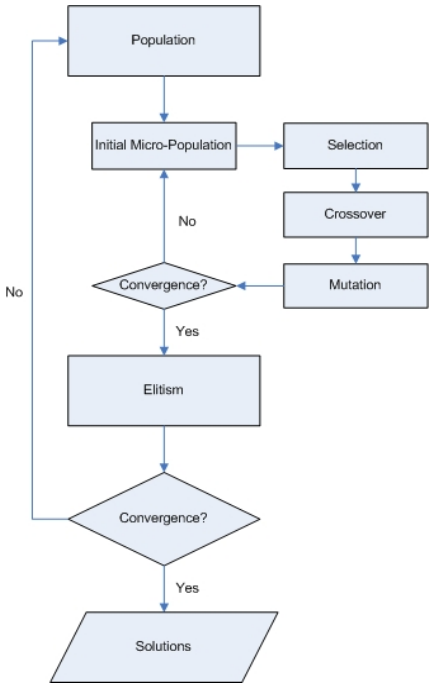

Figure 3. General Micro-Genetic Algorithm's Flow

3.3 파레토 최적

현실 세계에서는 목적 함수가 하나인 최적화 문제보다 두 개 이상의 목적들이 Trade-off 관계에 있는 다목적 최적화 문제(MOOP)가 일반적이다. MOOP의 해결은 어느 하나만을 최적화 시키는 것이 아니라 목적 함수 모두를 최적화 시키는 해들을 찾는 것이지만 다양한 조건을 만족시키는 여러 가지 해가 존재한다.

MOOP를 해결하는 방법으로 각 목적에 가중치를 부여하는 단순가중치 방법, 일부 목적을 제약식으로 변환하는 방법 등이 있으나, 이러한 방법들은 MOOP를 하나의 목적 함수로 변환하여 단일 목적 최적화 문제로 해결하는 방법들이다.

단일 목적 최적화 문제로의 변환이 아닌 다수의 목적을 동시에 최적화 시키는 해를 구하는 방법으로 파레토 최적 방법이 있다. 파레토 최적 방법은 특정 목적에 최적화된 단일 해를 구하는 것이 아니라 다수의 목적을 고려하여 우열을 가릴 수 없는 해집합을 구하는 방법을 말한다. <Figure 4>는 f_1, f_2의 두 가지의 목적 함수를 최소화시키는 MOOP의 해들을 예로 보여주고 있다.

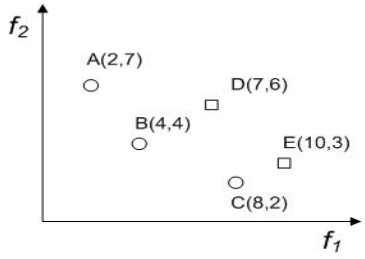

Figure 4. Pareto Optimal

<Figure 4>에서 해 A와 B를 비교하면, A는 f_1에 대해 B보다 더 좋은 해이지만, f_2에 대해서는 B보다 좋지 못한 해이다. 따라서 두 가지 모두를 최소화 시키는 문제에서 A와 B중 어떤 해가 더 좋은 해라고 말할 수 없다. 하지만 B와 D의 경우를 보면, f_1과 f_2에 대해 B가 더 좋은 해임을 알 수 있다. 그래서 B는 D보다 좋은 해라고 말할 수 있다. 이런 경우에 D는 B에 '지배된다'라고 표현하고, C와 E의 경우도 마찬가지이다. 그래서 C는 E를 지배한다. 그리고 A, B, C와 같이 어떤 해에도 서로 지배되지 않는 해를 파레토 최적해 또는 비지배해라고 말한다.

MOOP를 해결하는 것은 모든 목적을 동시에 최적화하는 파레토 최적해 집합을 구하는 것이라고 할 수 있다.

3.4 다목적 유전자 알고리즘(MOGA)

MOOP를 SGA로 해결하기 위해서는 단일목적 최적화 문제로 변환한 뒤 해결해야만 한다. MOOP에서 파레토 최적해 집합을 구하기 위한 유전자 알고리즘을 다목적 유전자 알고리즘(MOGA)이라고 한다(Murata and Ishibushi, 1995; Khare, 2003).

MOGA의 기본적인 구조는 SGA와 동일하나, 적합도 평가 시 다수의 목적을 동일하게 평가한다는 것과 단일해가 아닌 해집합을 저장한다는 차이가 있다.

MOGA는 비지배해들을 선택하는 방법에 따라 다음 두 가지로 나뉘게 된다. 첫째, 선별 시 각 유전자들의 지배여부에 따라 순위를 매기고, 그 순위를 바탕으로 적합도를 책정하는 파레토 유전자 알고리즘이 있고, 둘째로 선별을 위해 경쟁할 두 후보 유전자를 임의로 선택한 후 이 후보 유전자들과 비교할 임의의 비교집단을 선별하고, 두 후보 중 하나의 후보만 비교집단에 지배되지 않는다면 그 후보를 선택하고 그렇지 않다면 적소 매커니즘(Niche Mechanism)을 이용해 개체를 선별하는 적소 파레토 유전자 알고리즘이 그것이다(Hyun and Kim, 1996). 다양한 파레토 최적해를 찾기 위해서는 해가 집중된 곳보다는 넓게 흩어진 곳에서 해를 선택하는 것이 유리하다. 그러한 선택을 하는 방법을 적소 매커니즘이라고 한다.

파레토 유전자 알고리즘은 단순히 지배관계에 의한 순위만을 평가하지만, 적소 파레토 유전자 알고리즘은 적소 매커니즘이 존재하여 파레토 최적해가 고루 분포하기 때문에 다양한 대안 해를 제공할 수 있다.

4. 제안된 다목적 마이크로 유전자 알고리즘

4.1 다목적 마이크로 유전자 알고리즘(MOμGA)

MOμGA는 기존의 MOGA의 수렴 속도 향상을 위해 MOGA와 μGA가 결합된 알고리즘이다(Coello and Pulido, 2001).

MOμGA는 초기의 모집단 메모리(Population Memory)를 이용하여 적은 크기의 모집단을 생성하고 외부메모리에 최적해들을 저장하고 있어 유전자 알고리즘의 효율성을 높이며, 다양한 파레토 최적해를 찾도록 설계되어야 한다. MOμGA의 흐름도는 <Figure 5>와 같다.

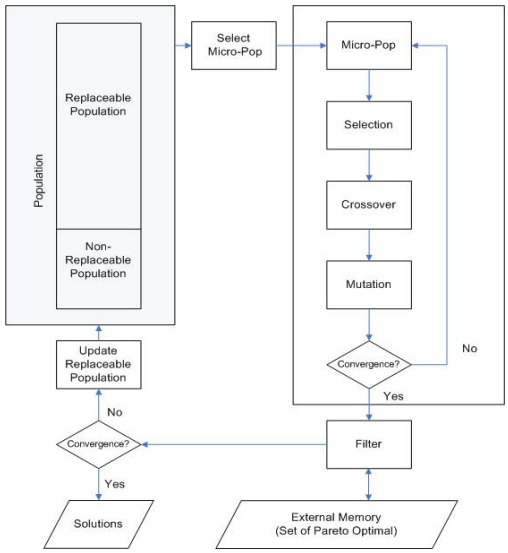

Figure 5. Multi-Objective Micro-Genetic Algorithm's Flow

4.2 유전자 구성

본 논문에서 제안하는 유전자는 그래프(G)에서 시작노드(s)와 목적 노드들(D)을 제외한 중간 노드들로 구성된다. 멀티캐스트 트리에 포함된다면 1을 그렇지 않으면 0값을 가지며, 유전자의 크기는 (|V|-1-|D|)가 된다. <Figure 6>은 <Figure 2>

의 트리를 유전자로 나타낸 것이다.

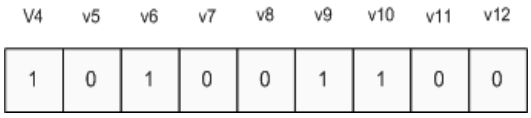

Figure 6. Chromosome Structure

(v5, v7, v8, v11, v12) 노드의 경우는 선택되지 않았기 때문에 유전자에 0으로 나타나고, 나머지 노드(v4, v6, v9, v10)의 경우 트리를 구성하는데 선택된 노드들이기 때문에 1로 나타난다. 시작 노드(v0)와 목적지 노드(v1, v2, v3)는 유전자에 포함되지 않았다.

4.3 디코딩

주어진 유전자를 사용하여 최소 스타이너 트리를 구성하고, 이를 해결하기 위해 가중치 있는 프림 알고리즘(Weighted Prim's Algorithm)을 이용한다. 가중치 있는 프림 알고리즘의 의사코드는 다음과 같다.

```
Weighted Prim
        G′ = (V′, E′)
        T ← {s}
        while V' - T ≠ ∅ do
                (v, u) ← Min2T()
                where v ∈ T, u ∈ V′-T
                add edge (v, u) to the tree
                T ← T + {u}
End Prim
```

위 알고리즘에서 시작노드, 염색체의 원소가 1인 중간노드, 그리고 목적지 노드들의 집합을 V′라고 할 때, G′은 V′과 각 노드사이의 에지인 E′으로 이루어진 G의 서브그래프이다. T는 현재까지 만들어진 최소신장트리에 포함되어 있는 노드들의 집합을 나타낸다. Min2T()는 T에 속한 노드와 V′-T에 속한 노드 사이에서 $[\alpha C(e)+\beta D(e)]$의 값이 최소가 되는 에지를 찾는다. α, β는 각각 비용가중치와 지연가중치를 나타낸다. α, β가 일정하다면, 선택된 중간 노드가 같은 경우 항상 같은 트리를 구성하게 될 것이며 이것은 다목적 문제를 단일목적 문제로 변환하는 것이다. 두 가지의 목적에 대해 다양한 해를 찾기 위해 본 논문에서는 각각의 유전자는 임의의 α, β를 갖게 하였다. 또한 선택된 중간노드가 같다고 할지라도 경우에 따라 다른 트리를 구성할 수 있게 되어 보다 많은 트리를 구성할 수 있기 때문에 보다 다양한 해를 찾게 된다.

4.4 모집단 메모리 생성 및 초기화

본 논문에서 해의 다양성을 보장하기 위해 모집단 메모리 기법(Coello and Pulido, 2001)을 사용한다. MOμGA의 모집단은 모집단 메모리와 마이크로 모집단으로 구분된다.

모집단 메모리는 일정 비율의 교체가능 메모리와 교체불가 메모리로 구성되며 초기 모집단에서 임의로 선별하여 교체가능 메모리와 교체불가 메모리로 나누어진다. 모집단 메모리의 두 부분에서 일정 비율로 유전자를 선별한 소수의 모집단이 마이크로 모집단이다. 실제로 교차, 돌연변이연산에 의해 진화가 일어나는 부분은 이 마이크로 모집단이고, 모집단 메모리는 마이크로 모집단을 구성하기 위한 해를 저장하는 역할을 한다.

모집단 메모리의 교체가능 모집단은 선택된 마이크로 모집단이 수렴 조건을 만족한 후 진화된 개체가 교체되어 점진적인 진화가 일어난다. 하지만, 모집단 메모리의 교체불가 메모리는 진화 과정동안에도 변하지 않는다. 만약 수렴된 해들로만 모집단 메모리가 채워진다면, 지역 최적해에 빠질 수 있다. 그러므로 모집단 메모리의 교체불가 메모리의 유전자가 변하지 않으면서 지역 최적해 문제를 회피하고, 다양한 해를 보장할 수 있게 된다.

4.5 선택연산

MOGA는 그 목적식 수만큼의 적합 함수가 존재하며, 본 논문에서는 비용 함수, 지연시간 함수의 2가지의 목적함수가 존재하는 것으로 가정 하였다.

본 논문에서는 토너먼트 선별법을 이용하여 진화될 유전자를 선택한다. 토너먼트 선별법은 선별을 위해 경쟁할 두 후보 유전자를 임의로 선택하고, 이 후보 유전자들과 비교할 임의의 비교집단을 선별한다. 두 후보 중 하나의 후보만 비교집단에 지배되지 않는다면 그 후보를 선택하고, 두 후보 모두 지배되거나, 두 후보 모두 지배되지 않는다면 적소 매커니즘을 이용해 개체를 선별하는 적소 파레토 기법을 적용한다. 적소 매커니즘은 보다 다양한 해를 구하기 위한 방법으로 후보 유전자와 비교집단의 각 해들과 유클리드 거리를 측정하여 더 큰 값을 갖는 후보를 선택한다. 유클리드 거리가 크다는 것은 비교집단과 멀리 떨어져 있다는 것을 뜻하고, 진화되는 해의 분포가 다양하게 된다.

4.6 유전자 연산자

본 논문에서는 교차 연산자와 돌연변이 연산자로 이점 교차 연산자와 단일점 돌연변이 연산자를 각각 이용한다.

이점 교차연산자는 두 개의 부모 유전자에서 임의의 두 점을 선택하여 두 점 사이의 염색체를 교환하는 것으로 <Figure 7>과 같다. 단일점 돌연변이 연산자는 한 개의 부모 유전자에서 임의의 한 점을 선택하여, 그 염색체가 0이라면 1로, 1이라면 0으로 변환하는 방법이며 <Figure 8>과 같다.

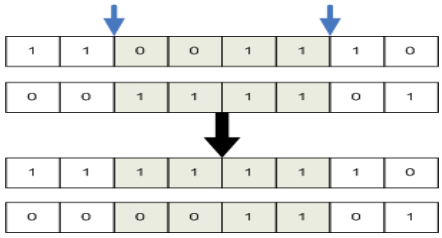

Figure 7. Two-point Crossover Operator

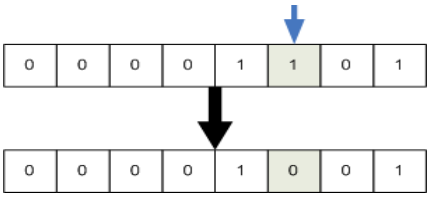

Figure 8. One-point Mutation Operator

4.7 보정연산

교차연산과 돌연변이연산을 통해 생성된 유전자들이 항상 가능해 집합에 속하지는 않는다. 이러한 경우 생성된 유전자가 제약 사항을 만족하는 해가 되도록 수정해야 한다. 이것을 보정연산이라고 한다.

본 논문에서 보정연산이 필요한 경우는 다음 두 가지 경우로 나누어진다. 첫째는 하나의 유전자를 이용하여 멀티캐스트 트리를 구성하지 못하는 경우가 있다. 즉, 가중치 있는 프림 알고리즘을 이용하여 트리를 구성할 경우에 선택된 중간 노드들만으로 트리를 구성할 수 없는 경우이다.

이를 보정하기 위해서 주어진 그래프(G)에서 선택된 중간 노드, 시작 노드 , 목적 노드만으로 구성된 subgraph(G′)를 확인한 후 G′이 두 개 이상의 component로 구성되어 있다면, 각 component 사이의 노드들의 최단 경로를 구하여 최단 경로 상에 있는 노드를 추가함으로써 G′을 connected graph로 만들게 된다. 물론 이 때의 최단 경로 역시 4.3절의 가중합을 통해서 구하게 된다. 추가된 노드들에 해당하는 유전자 값을 1로 보정해 준다.

둘째로 단말 노드가 선택된 경우이다. 단말 노드는 자식 노드를 갖지 않는 노드를 말한다. 최소 스타이너 트리를 구성하고 난 후, 중간 노드들 중 단말노드가 존재 한다면 그 중간 노드들은 목적지 노드로 가는 경로와 무관하기 때문에 중간 노드가 단말 노드라면 그에 해당하는 유전자 값을 0으로 보정하여 중간 노드가 선택되지 않도록 한다.

4.8 필터

본 논문에서는 마이크로 모집단이 진화되어 형성한 유전자들 중 기존의 외부메모리에 저장되어 있는 유전자들에 지배되지 않은 유전자들을 외부메모리에 저장하고, 외부메모리 내의 유전자들 중 어느 유전자에라도 지배되는 유전자는 외부메모리에서 제외시킨다. 외부메모리가 가득 찬 경우는 적소 매커니즘을 통해 외부메모리에서 제외시킬 유전자를 선별한다.

유전자 진화 과정에서 파레토 최적해들을 진화시키지만 다양한 대안 해를 제공하기 위해서는 특정 부분에 편중되기 보다는 일정하게 분포한 파레토 최적해를 찾아야 한다. 따라서 파레토 최적해의 다양성을 보장하기 위해 적소 매커니즘을 사용한다. 적소 매커니즘은 적소 반경을 이용하며, 적소 값은 수식 (3)을 사용하여 결정한다.

$$m(i) = \sum_{j \in R_i} (r - d(i,j)) \tag{3}$$

r: 적소 반경
i: 비교할 유전자
R_i: i와의 거리가 r이하인 모든 유전자 집합
$d(i,j)$: 유전자 i와 유전자 j의 유클리드 거리
$m(i)$: 유전자 i의 적소 값

각 유전자의 적소 값은 적소 반경 안에 있는 유전자들만 고려하여 계산하며, 거리가 멀수록 낮은 적소 값을 가진다. 따라서 적소 반경 안에 유전자가 많다면 적소 값이 커지고, 적소 반경안의 유전자가 비교할 유전자와 가까울수록 적소 값이 커진다. 즉, 외부메모리에서 제외시킬 유전자는 적소 값이 가장 큰 값이다. 해의 종류가 다양하여 파레토 최적해들이 다수 예상될 경우에는 r의 값을 증가시켜 파레토 최적해가 밀집해서 발견되지 않게 하며, 해의 종류가 어느 정도 제한되어 있다면, r의 값을 감소시켜 가능한 한 많은 해들이 살아남게 한다.

4.9 교체가능 모집단 갱신

모집단 메모리에서 선별된 마이크로 모집단이 수렴하고 나면, 수렴된 해는 필터를 통해 외부 메모리에 저장된다. 또한 교체가능 모집단의 유전자들 중에 수렴된 해에 지배되는 유전자가 있다면 지배되는 유전자 중의 임의의 하나의 유전자를 수렴된 해로 교체한다.

마이크로 모집단은 적은 수의 모집단이며, 그 적은 수의 모집단이 진화하고, 마이크로 모집단에서 수렴된 해들로 전체 모집단을 교체해 나가면서 모집단 메모리도 진화된 해들을 보유하게 된다. 따라서 모집단 메모리의 진화 세대수가 증가할수록 모집단 메모리는 진화된 해들을 보유하게 되며, 그로써 마이크로 모집단 자체도 좋은 해를 가지고 진화를 시작하게 된다.

모집단 메모리의 모든 해들이 진화된 해들로 교체된다면 지역 최적해에 빠질 가능성이 있지만, 교체불가 메모리를 두어 지역 최적해가 되는 것을 방지하고, 다양한 해를 보장할 수 있다.

5. 실험 및 성능평가

5.1 실험 환경

본 논문은 MOGA의 성능 개선을 제안하였기 때문에 동일한

그래프 구성으로 MOGA와 MOμGA를 비교 분석하였다.

실험에 사용된 문제의 그래프는 <Table 1>과 같은 값을 갖는 임의의 연결 그래프로 구성하였으며 서로 다른 10개의 그래프를 생성하여 결과를 관찰하였고, 문제 1에 대해서는 10회의 실험을 하였다. 각 에지의 비용과 지연시간은 10에서 300사이의 정수 값을 무작위로 설정하였다.

Table 1. Problems

Problem	number of nodes	number of destination nodes	number of edges
1~10	100	10	400

본 논문에서는 MOGA와 MOμGA 모두 동일하게 적소 파레토 유전자 알고리즘을 적용하여 성능 분석을 하였다. 또한 MOGA의 모집단 수를 N1, 세대수를 G1이라고 할 때, MOGA의 총 연산수(교차, 변이, 보정)는 (N1 * G1)이다. MOμGA에서 모집단 메모리의 크기를 M, 마이크로 모집단 수를 N2, 마이크로 모집단의 세대수를 mG, 전체 세대수를 G2라고 할 때, MOμGA에서의 총 연산수(교차, 변이, 보정)는 (N2 * mG * G2)가 된다. 즉, (N1 * G1)과 (N2 * mG * G2)가 같다면, 두 알고리즘은 같은 연산수를 처리하므로 수행 시간이 같다고 할 수 있다.

위의 연산수에 적소 매커니즘은 포함되어 있지 않다. 그 이유는 하나의 유전자가 생성될 때마다 MOGA와 MOμGA 모두에서 동일한 연산을 하기 때문에 포함하지 않아도 무관하기 때문이다. 따라서 본 논문에서는(M = 100, N1 = 100, G1 = 1000, N2 = 4, mG = 25, G2 = 1000)으로 파라미터를 설정하였다. 또한 MOμGA에서 교체불가 모집단은 M의 10%인 10으로 설정하였다.

5.2 결과 분석

<Table 1>의 서로 다른 그래프에 대한 MOGA의 해와 MOμGA의 해를 나타낸 것이 <Figure 9>~<Figure 18>이다.

각각의 해들은 멀티캐스트 트리의 총 비용과 총 지연시간을 의미하며 X축은 비용을 Y축은 지연시간을 나타낸다. 각 결과의 ◆는 MOGA의 해를 나타내며, ■는 MOμGA의 해를 나타낸다.

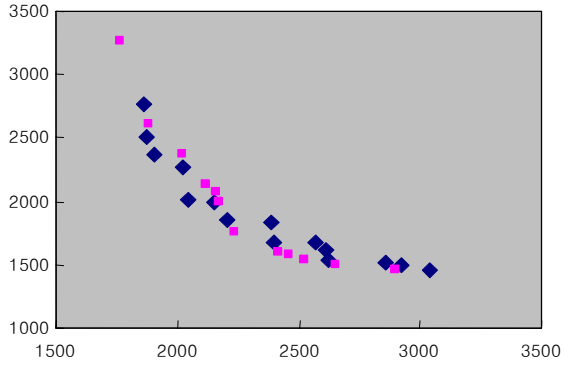

Figure 9. Pareto Optimals of Problem 1

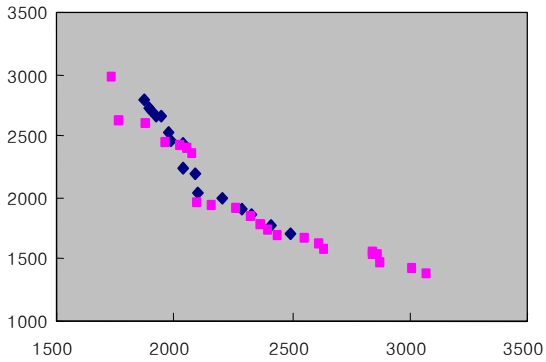

Figure 10. Pareto Optimals of Problem 2

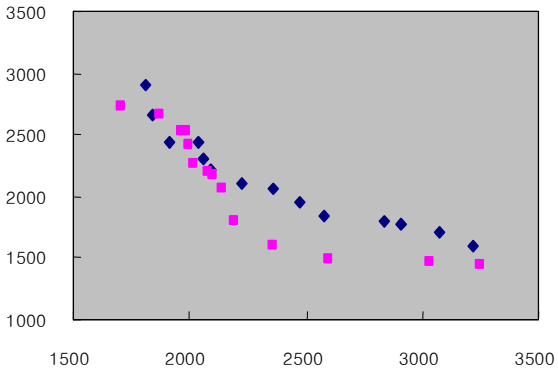

Figure 11. Pareto Optimals of Problem 3

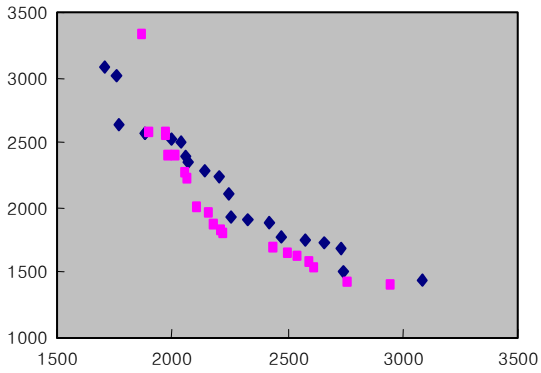

Figure 12. Pareto Optimals of Problem 4

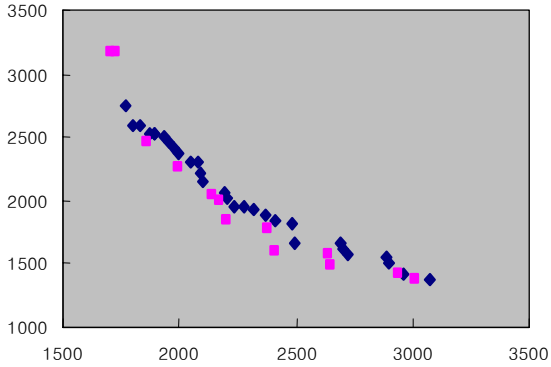

Figure 13. Pareto Optimals of Problem 5

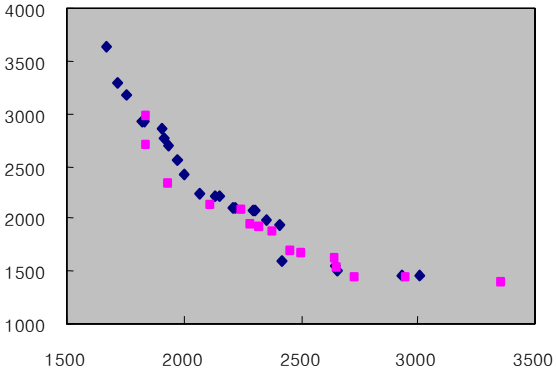

Figure 14. Pareto Optimals of Problem 6

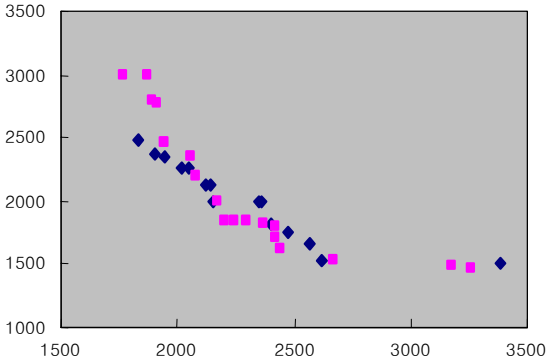

Figure 15. Pareto Optimals of Problem 7

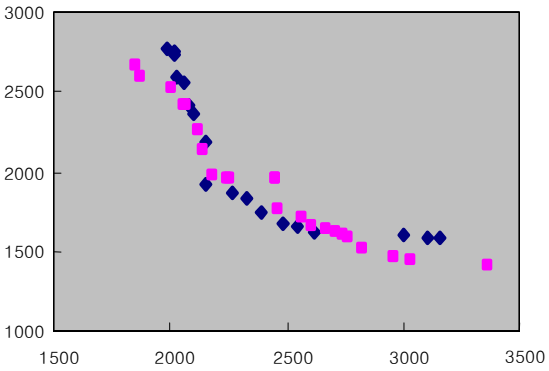

Figure 16. Pareto Optimals of Problem 8

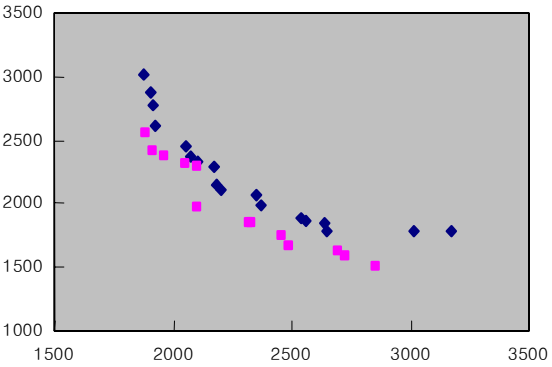

Figure 17. Pareto Optimals of Problem 9

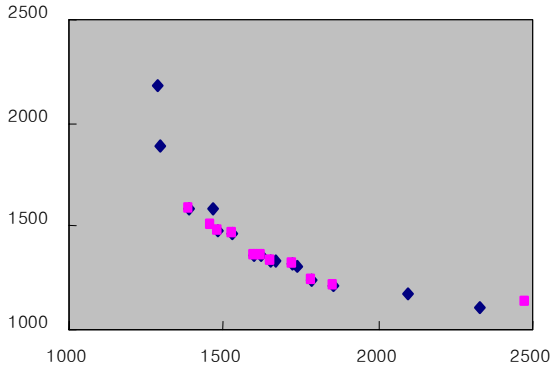

Figure 18. Pareto Optimals of Problem 10

비용과 지연의 최소화 문제이기 때문에 해가 원점에 가까울수록 좋은 해라고 말할 수 있다.

문제 2, 3, 4, 5, 6, 9, 10의 7가지의 문제에 대해서는 제안된 MOμGA의 해가 MOGA의 해보다 좋다는 것을 확인할 수 있다. 나머지 3가지의 문제에 대해서도 MOμGA와 MOGA는 비슷한 해를 구해내고 있다. <Figure 9>에 나타나 있는 문제 1의 결과는 MOμGA와 MOGA 중 어느 것이 좋은 해인지 판단하기 힘들다.

<Figure 19>~<Figure 28>은 Problem 1에 대해서 10번을 수행한 결과를 나타낸다.

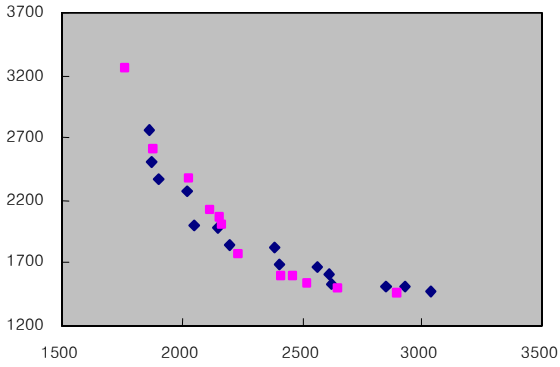

Figure 19. Pareto Optimals of Problem 1~1

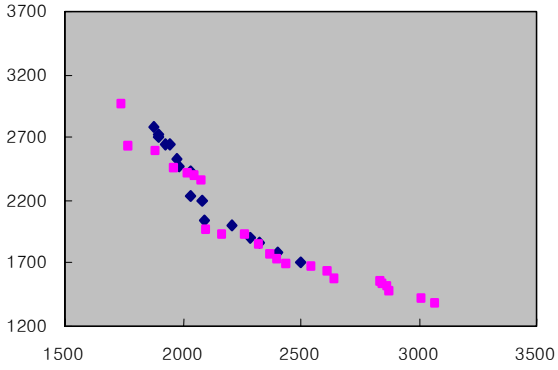

Figure 20. Pareto Optimals of Problem 1~2

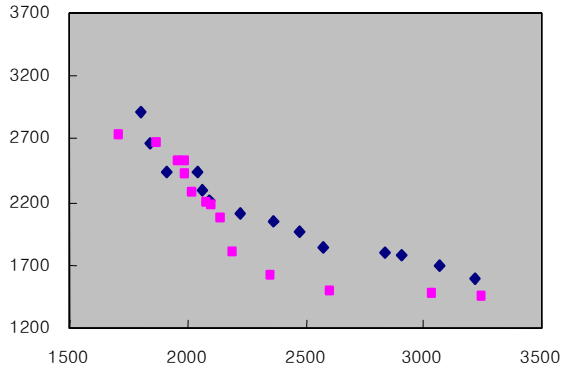

Figure 21. Pareto Optimals of Problem 1～3

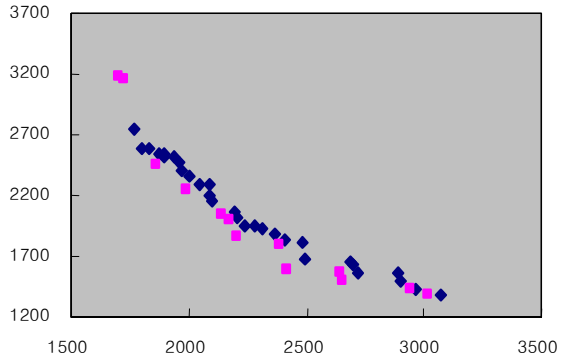

Figure 22. Pareto Optimals of Problem 1～4

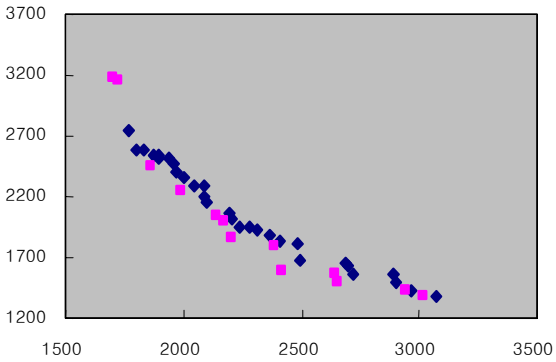

Figure 23. Pareto Optimals of Problem 1～5

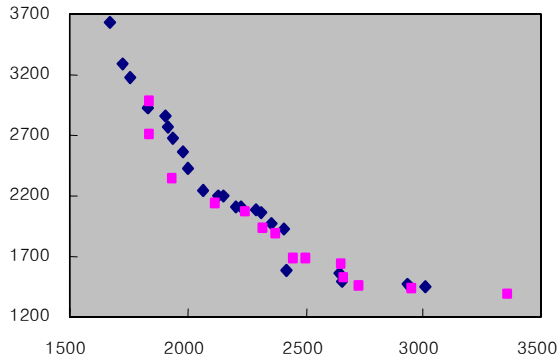

Figure 24. Pareto Optimals of Problem 1～6

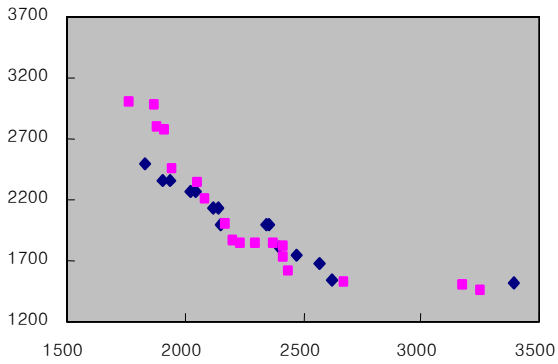

Figure 25. Pareto Optimals of Problem 1～7

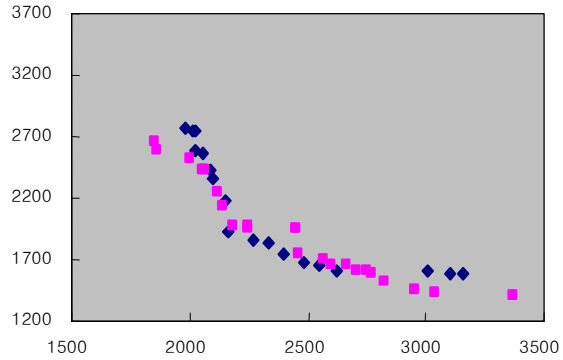

Figure 26. Pareto Optimals of Problem 1～8

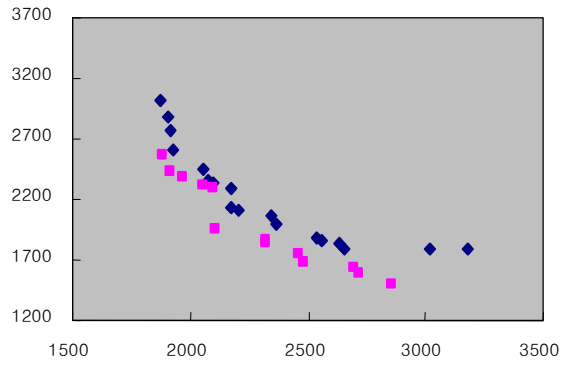

Figure 27. Pareto Optimals of Problem 1～9

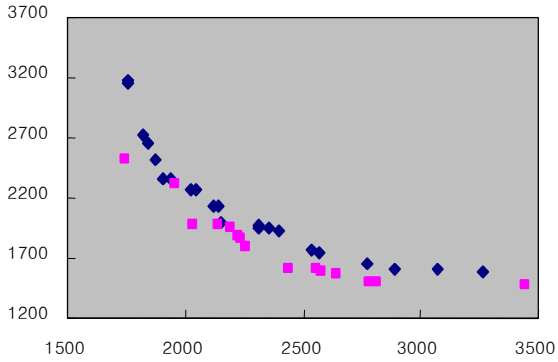

Figure 28. Pareto Optimals of Problem 1～10

<Figure 9>에서 MOμGA와 MOGA의 해가 유사한 값을 보인 문제 1에 대한 10번의 실험 결과 많은 경우에 MOμGA의 해가 MOGA의 해보다 좋다는 것을 확인할 수 있다. <Figure 29>는 문제 1에 대한 결과인 <Figure 19>~<Figure 28>을 한꺼번에 나타낸 것이다. MOμGA의 해들이 MOGA의 해들보다 전체적으로 아래쪽에 위치하는 것을 확인할 수 있다. 따라서 MOμGA의 해가 MOGA의 해보다 좋다는 것을 볼 수 있다. 같은 문제를 풀었을 때 각각의 결과가 다른 것은 진화 연산이 확률적으로 일어난다는 것과 초기 모집단의 구성이 각기 다르기 때문이다.

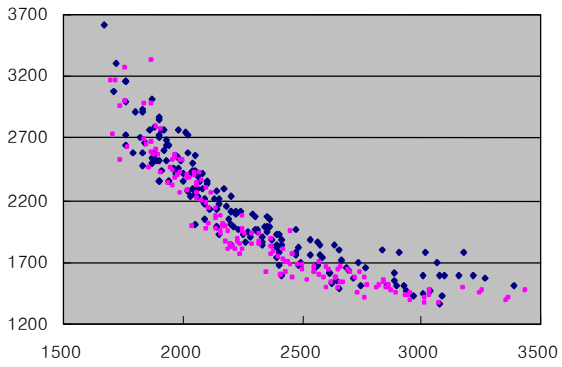

Figure 29. All Solutions of <Figure 19>~<Figure 28>

앞 절에서 MOμGA와 MOGA의 비교를 위해 동일한 수의 유전자를 생성하게 하여 동일한 연산수를 갖도록 파라미터를 설정하였다. 따라서 실험에 대한 실제 수행 시간의 비교가 필요하다. <Table 2>는 각 문제에 대한 실제 수행 시간을 나타낸다.

Table 2. Running Time

Problem	Running Time(hour)	
	MOμGA	MOGA
1	8.913676	8.915932
2	9.05665	8.621201
3	9.053411	9.232665
4	9.09569	9.523288
5	9.04132	9.710223
6	9.125538	9.425824
7	9.060945	9.352319
8	9.091836	9.062905
9	9.077075	8.964106
10	8.870752	9.474725
Average	9.038689	9.228319

각 문제의 MOμGA와 MOGA에서 생성하는 유전자 수는 동일하며 실제 평균 수행 시간은 MOμGA가 MOGA보다 짧다는 것을 확인 할 수 있다.

6. 결론 및 향후 연구

본 논문에서는 멀티캐스트 라우팅 문제에 기존의 MOGA의 성능 향상을 위하여, 모집단 메모리의 교체가능 모집단과 교체불가 모집단에서 소수의 모집단을 추출하고, 추출된 마이크로 모집단을 진화시켜서 수렴된 해를 교체가능 모집단의 해와 교체하여 모집단 메모리도 점진적으로 진화하는 MOμGA를 적용하였다.

MOμGA는 전체 모집단이 동시에 진화하는 것이 아니고, 소수의 모집단을 선별하여 그들만으로 진화를 시켜서 짧은 시간에 최적해를 찾아낸다. 하지만 그러한 최적해는 지역 최적해일 가능성이 높다. 이러한 지역 최적해를 막기 위해 모집단 메모리의 교체가능 모집단과 교체불가 모집단에서 일정 비율로 선별하여 마이크로 모집단을 구성하여 다양한 해를 보장한다.

기존의 MOGA와 제안된 MOμGA의 비교, 분석을 위하여 동일한 임의의 그래프를 이용하였으며, 단순 수행 시간 비교가 아닌 동일한 연산수를 실행하는 조건으로 비교하였으며, 실제 수행 시간이 약 2% 단축되었다는 것을 관찰하였다. 이러한 비교 분석을 통해 동일한 연산수에 대해서 제안된 MOμGA의 해가 기존의 MOGA보다 더 우수하다는 것을 보였다.

앞으로 교체가능 메모리와 교체불가 메모리 비율에 따른 비교와 수행 시간의 많은 부분을 차지하는 보정연산의 속도 개선을 위해 휴리스틱 알고리즘에 관한 연구가 필요하다.

참고문헌

Coello, C. A. and Pulido, G. T. (2001), A Micro-Genetic Algorithm for Multiobjective Optimization., *Lecture Notes in Computer Science*, **1993**, 126-140.

Esbensen, H. (1995), Computing Near-Optimal Solutions to the Steiner Problem in a Graph Using a Genetic Algorithm, *Network*, **26**, 173-185.

Hyun, C. J. and Kim, Y. G. (1996), A Genetic Algorithm for Multiple Objective Sequencing Problems in Mixed Model Assembly Lines, *Journal of the Korean Institute of Industrial Engineers*, **22**(4), 533-549.

Khare, V. (2003), Performance Scaling of Multi-Objective Evolutionary Algorithms, *EMO 2003*, **8**(11), 376-390.

Kompella, V. P., Pasquale, J. C., and Polyzos, G. C. (1993), Multicast Routing for Multimedia Communications, *IEEE/ACM Trans. on Networking*, 1(3), 286-292.

Kou, L., Markowsky, G. and Berman, L. (1981), A Fast Algorithms for Steiner Tree, *Acta Informatica*, **15**, 141-145.

Krishnakumar, K. (1989), Micro-Genetic Algorithms for Stationary and Non-Stationary Function Optimization., *SPIE Proceedings: Intelligent Control and Adaptive Systems*, **1196**, 289-296.

Lee, Y. G. and Han, C. G. (2004), Multi Objective Genetic Algorithm for Multi-Objective Multicast Routing, *Master's Thesis Kyung Hee Univ.*

Murata, T. and Ishibushi, H. (1995), MOGA: Multi-Ojbective Genetic Algorithms, *Proc. 2nd IEEE Int. Conf. on Evolutionary Computation*, 289-294.

Osyczka, A. (1985), Multicriteria Optimization for Engineering Design, *Design*

Optimization, 193-227.
Tsai, C. F. and Tasi, C. W. (2001), A Nobel Multicast Routing Algorithm with Delay Constraint for Multimedia Application in Wide Area Network, *Info-tech and Info-net*, **5**, 301-305.
Wang, B. and Hou, J. C. (2000), Multicast Routing and Its QoS Extension: Problems, Algorithms, and Protocols, *IEEE Networks*, **14**(1), 22-36.
Winter, P. (1987), Steiner Problem in Network: a Survey, *Networks*, **17**, 129-167.
Ying, L. and Jianping, W. (2002), A Genetic Algorithm for the Degree-Constrained Multicasting Problem, *High Speed Networks and Multimedia Communications 5th IEEE International Conference on*, 315-319.

전 성 화
경희대학교 컴퓨터공학전공 학사
경희대학교 컴퓨터공학과 석사
현재: 경희대학교 컴퓨터공학과 박사과정
관심분야: 그래프 이론, 유전자 알고리즘, 게임 이론

한 치 근
서울대학교 산업공학과 학사
서울대학교 산업공학과 석사
미국 Pennsylvania 주립대학 Computer Science 석사
미국 Pennsylvania 주립대학 Computer Science 박사
현재: 경희대학교 컴퓨터공학 전공 교수
관심분야: 계산 이론, 알고리즘, 수치 해석, 병렬 처리