진화 알고리즘을 이용한 초고속 통신망에서의 멀티캐스트 경로배정 방법에 관한 연구

이 창 훈[†] · 장 병 탁[††] · 안 상 현[†††] · 곽 주 현[††††] · 김 재 훈[††††]

요 약

화상 회의, 원격 진료 및 교육 시스템, CSCW 등과 같은 그룹 응용을 지원하기 위해서는 망에 의해서 멀티캐스트 기능이 제공되어야 한다. 멀티캐스트 경로배정의 방법으로는 보통 최단 경로 트리 방식과 최소 비용 스타이너 트리 방식이 사용되는데 최소 비용 스타이너 트리 방식은 그룹당 한 개의 트리만을 사용하면 되는 장점이 있다. 그러나 최소 비용 스타이너 트리를 발견하는 문제 자체가 NP-complete이므로 효율적인 경험적 알고리즘을 개발하는 것이 주요 문제점으로 대두된다. 본 논문에서는 최적해에 보다 가까운 최소 비용 스타이너 트리를 찾기 위해 유전자 알고리즘을 사용하는 진화적 최적화 방법을 제안하고자 한다. 특히 스타이너 트리의 표현에 있어, 일반적인 유전자 알고리즘에서 사용되는 이진 스트링의 개체 표현 대신 트리를 사용하여 개체를 표현하는 방법을 제안함으로써 최적화의 효율을 개선하는 방식을 제안한다. 실험을 통하여 제안된 방법이 최적해가 알려진 네트워크상에서의 실제로 최적해를 찾을 수 있음을 보여주며, 또한 기존의 경험적 알고리즘과의 비교를 통하여 진화방식에 의한 최적화가 기존의 방법보다 최적해에 더 가까이 수렴할 수 있음을 보여준다.

Multicast Routing On High Speed Networks Using Evolutionary Algorithms

Chang-Hoon Lee[†] · Byoung-Tak Zhang[††] · Sang-Hyun Ahn[†††] · Ju-Hyun Kwak[††††] · Jae-Hoon Kim[††††]

ABSTRACT

Network services, such as teleconferencing, remote diagnostics and education, and CSCW require multicasting. Multicast routing methods can be divided into two categories. One is the shortest path tree method and the other is the minimal Steiner tree method. The latter has an advantage over the former in that only one Steiner tree is needed for a group. However, finding a minimal Steiner tree is an NP-complete problem and it is necessary to find an efficient heuristic algorithm. In this paper, we present an evolutionary optimization method for finding minimal Steiner trees without sacrificing too much computational efforts. In particular, we describe a tree-based genetic encoding scheme which is in sharp constast with binary string representations usually adopted

※이 연구는 1995년도 한국학술진흥재단 대학부설연구소 과제 연구비에의하여 연구되었음.

† 종신회원:건국대학교 컴퓨터공학과
†† 정 회 원:서울대학교 컴퓨터공학과
††† 정 회 원 서울시립대학교 전산통계학과
†††† 준 회 원:건국대학교 컴퓨터공학과

논문접수:1997년 10월 23일, 심사완료:1997년 12월 19일

in convetional genetic algorithms. Experiments have been performed to show that the presented mehtod can find optimal Steiner trees for given network configurations. Comparitive studies have shown that the evolutionary method finds on average a better solution than other conventional heustric algorithms.

1. 서 론

고속 통신망이 발달함에 따라 여러 종류의 데이터 즉 텍스트, 음성, 정지화상(still image), 동화상(live video) 등을 제공하는 서비스가 늘어나고 있으며 같은 요건에서도 더 나은 서비스를 요하게 되었다. 이러한 요구에 맞게 통신망에 있어 데이터의 양적 증가도 가속화되고 있으며 이에 따른 통신망의 부담도 더욱 증가하고 있다. 또한 다수의 관련된 네트워크 사용자들이 그룹을 형성하여 데이터를 주고받는 그룹방식의 응용이 부각되고 있으며 이러한 상황에서 효율적인 데이터 전송의 중요성은 매우 커지고 있다.

이러한 그룹단위에서 멀티미디어 서비스를 이용하는 응용에는 화상 회의 및 원격 진료/교육 시스템, CSCW(Computer Supported Cooperative Work) 등과 같은 것들이 있으며, 이들 그룹 응용을 지원하기 위해서는 망에 의해서 멀티캐스트 기능이 제공되어야 한다.

멀티캐스트 기능은 한 지점에서 다수의 목적지 또는 사용자(그룹 구성원)들에게 데이타를 전송하는 것으로, 그룹 구성원들 간에 멀티캐스트 연결이 설정되어야 한다. 지점간(point-to-point) 망에서 멀티캐스트 기능은 멀티캐스트 트리를 사용함으로써 효율적으로 지원될 수 있으며, 이에 대한 연구는 최단 경로 트리(shortest path tree) 방식과 최소 비용 스타이너 트리(minimal Steiner tree) 방식을 중심으로 수행되어 왔다[1, 2, 3, 4, 5, 7]. 최단 경로 트리 방식의 경우, 각 그룹 구성원을 루트로 하는 최단 경로 트리가 설정되어야 하며, 결과적으로 그룹 내의 구성원 수 만큼의 트리를 필요로 하는 단점을 지닌다[4]. 최소 비용 스타이너 트리는 트리를 구성하는 링크들의 비용의 합이 최소가 되는 트리로서, 이 방식의 경우는 그룹당 한 개의 트리만이 사용된다. 따라서 최소 비용 스타이너 트리 방식이 최단 경로 트리 방식보다 훨씬 효율적이지만, 최소 비용 스타이너 트리를 발견하는 문제 자체가 NP-complete[6]이므로 효율적인 경험적(heuristic) 알고리즘을 개발하는 것이 주요 문제점으로 대두된다. 현재 최소 비용 스타이너 트리를 발견하는 경험적 알고리즘은 다수 제안되었으며, 이들의 성능에 대한 분석도 수행되었다[5, 7].

본 연구에서는 진화 알고리즘을 사용하여 보다 최적해에 가까운 최소 비용 스타이너 트리를 구하는 방법을 개발하고자 한다. 최소 비용 스타이너 트리를 찾기 위하여 주어진 그래프에서 어떻게 스타이너 트리를 생성해 내며, 또한 어떠한 방법으로 진화 알고리즘을 적용하는가를 보이며 그 효용성을 실험적으로 검토한다. 제2절에서는 먼저 진화 알고리즘에 대한 개요를 설명하고, 제3절에서는 주어진 네트워크상에서 스타이너 트리의 표현 방법과 생성 과정을 기술한다. 제4절에서는 진화 방식에 의해 멀티캐스트 경로 배정을 하기 위한 유전 연산자를 기술하며 전반적인 알고리즘의 흐름도는 제5절에서 기술된다. 제6절에서는 실험 결과를 기술하고 그 성능을 분석한다. 제7절에서는 본 연구의 결론과 앞으로의 과제에 대하여 기술한다.

2. 진화 알고리즘

진화 알고리즘은 자연계의 진화(evolution) 과정에 기반한 새로운 계산 모델로서 특히 아주 어려운 최적화(optimization) 문제를 해결하는데 성공적으로 응용되고 있다. 진화 알고리즘에서는 문제 해결을 위해 시도할 수 있는 여러 방법들을 원소로 하는 집합을 구성하여 탐색을 함으로써 최적해를 구한다. 자연 생태계에 비유하여 각각의 원소를 흔히 개체(individual) 그리고 이들의 집합을 개체군(population)이라 한다. 각각의 개체는 보통 스트링의 형태로 코딩되며 이를 염색체(chromosome)이라 한다. 탐색은 염색체의 구조와 내용을 변경함으로써 수행되는데 이 때 사용되는 연산자를 유전 연산자(genetic operators)라 한다.

흔히 사용되는 연산자로는 교차(crossover)와 돌연변이(mutation)가 있다. 교차 연산자는 두 개의 부모

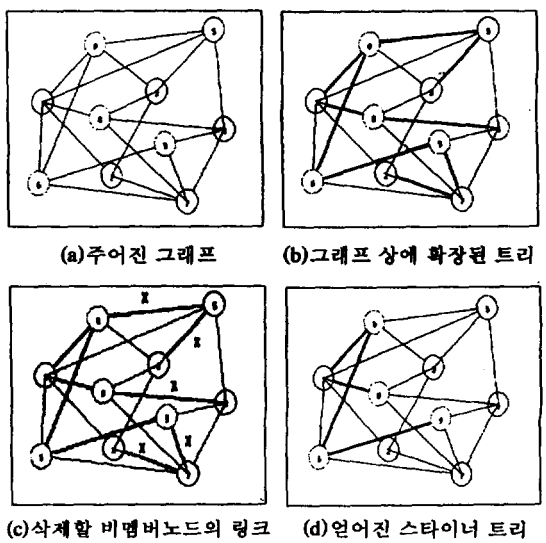
염색체의 일부를 끊어서 상호 교환함으로써 수행한다. 돌연변이 연산자는 부모염색체의 유전자(gene)를 임의로 변경하는 방법이다. 이러한 연산자들을 일정한 또는 가변적인 확률 즉 교차율과 돌연변이율에 의해 적용하여 좀더 우수한 개체 즉 최적 해를 찾아간다. 진화의 과정은 초기화된 어떤 개체군으로 시작하여 새로운 개체군들을 반복적으로 생성하며 세대(generation) 교체를 반복하는 일종의 집단에 의한 탐색과정이다. 세대 교체 시에는 선택(selection) 연산자가 있어 적합도(fitness)가 높은 개체들이 다른 개체들보다 다음 세대의 개체군 생성에 더욱 많은 영향력을 미치도록 함으로써 우성 인자들이 계속 유전할 가능성을 높여준다.

진화 알고리즘에 의한 최적화는 군(population)에 의한 탐색 방법으로서 한 세대에서 찾은 가능한 좋은 해를 다음 세대에 간접적으로 보존함으로써 전역적인 해를 찾아내는데(global search)에 적합하다. 이것은 기존의 탐색 방법이 하나의 탐색점을 이용하여 최적해를 찾고자 하는(point-based search) 것과는 대조되는 방식이다. 또한 진화 알고리즘은 상태 공간에 대한 제약, 예를 들어 공간의 연속성과 같은 제약 조건을 갖지 않기 때문에 여러 가지 문제에 적용이 가능하고 또 주어진 상황에 적응적으로 대처해서 탐색을 한다는 특징을 갖고 있다. 특히 multimodal 공간에서의 탐색, 최적화 및 학습 문제에 탁월한 성능을 보일 수 있음이 여러 연구에 의하여 확인되었다[9, 10].

3. 주어진 네트워크상에서 스타이너 트리 구축

본 연구에서는 최소비용 스타이너 트리를 구하기 위한 진화 알고리즘을 응용하는 방법을 고찰하고자 한다. 즉, 각 세대의 개체들은 스타이너 트리가 되며, 진화 알고리즘을 적용하여 세대를 반복할수록 이전 세대의 것보다 우수한 개체를 형성하여 나가는 진화 방식을 탐구하고자 한다. 이 과정을 효율적으로 수행하기 위해 가장 중요한 점 중의 하나는 스타이너 트리를 어떠한 자료구조로 표현하는가이다.

진화 알고리즘을 사용하여 최소비용 스타이너 트리를 찾는 방법은 Esbensen에 의해서 이미 제안된 바 있으며[13], 이 방법에서는 주어진 그래프의 각 노드를 특정 비트로 매핑시켜 이진 스트링 형태로 표현하고 있다. 스타이너 트리를 이진스트링 형태로 표현하는데 있어서는 트리형태의 계속적인 유지가 어렵고, 또한 이진스트링에 의한 표현은 유전 연산자의 구현이 복잡해지기 쉽고 유효하지 못한 트리형태를 생성할 가능성이 많다고 본다. 따라서 좀더 효율적인 알고리즘의 구현을 위해 본 연구에서는 네트워크상에서 멀티캐스트 경로를 의미하는 스타이너 트리를 이진스트링의 자료구조가 아닌 트리 자료구조를 통해 알고리즘에 적용하였다. 스타이너 트리형태를 프로그램상에서도 같은 모습을 갖는 트리 자료구조로 표현함으로써 네트워크 정보 유지 및 알고리즘의 구현을 효율적으로 수행 할 수 있었다. 트리 자료구조에 있어 각 노드는 네트워크 상의 실제 노드를 의미하며 따라서 한 부모 노드와 자식노드들상의 링크는 실제 네트워크상의 이웃노드를 의미하게 된다. 트리 자료구조에 있어 또한 네트워크 각 노드간의 링크 경비를 의미하는 정보를 포함시킴으로써 평가 함수의 수행을 손쉽게 구현할 수 있었다. 이러한 스타이너 트리 표현을 통하여 주어진 네트워크상에서 처음 세대를 구성하는 개체군을 만들고 진화 알고리즘의 교차(Crossover), 돌연변이(Mutation) 선택(Selection) 연산자를 적용하여 다음 세대를 계속 반복적으로 생성한다. 주



(a)주어진 그래프 (b)그래프 상에 확장된 트리

(c)삭제할 비멤버노드의 링크 (d)얻어진 스타이너 트리

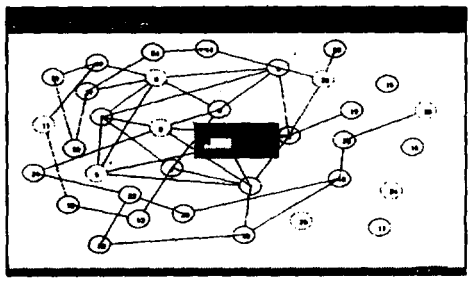
(그림 1) 한 개체로서 스타이너 트리를 얻는 과정
(Fig. 1) Constructing a steiner tree

어진 네트워크에서 처음 세대의 구성은 실험 인자로 넘겨받은 개체군의 크기(population size)만큼 무작위적으로 만들어진다.

한 개체 즉 스타이너 트리를 만드는 방법은 먼저 임의의 한 멀티캐스트 멤버 노드로부터 모든 네트워크 노드에 이르는 스패닝 트리를 우선 랜덤하게 만든다.((그림 1)의 (b)) 다음으로 얻어진 스패닝 트리를 순환함으로써 단말 노드가 멀티캐스트 멤버 노드가 되게 필요없는 노드를 삭제한다.((그림 1)의 (c)) 이것은 최소 비용 스타이너 트리를 구성함에 있어 단말 노드가 통신의 비 멤버 노드가 되는 것은 아무런 의미가 없기 때문이다.

(그림 1)과 같이 구성된 각각의 객체 즉 스타이너 트리는 자료구조에 있어 트리 형태를 유지하게 되며, 일어날 수 있는 임의의 멀티캐스트 경로를 의미한다. 이후 유전자 알고리즘에 의해 처음 세대중 좋은 부모를 선택하여 다음 세대를 구성할 자식을 만들어 가게 되며 세대를 반복할수록 더욱 좋은 경로를 나타내는 스타이너 트리를 찾아가게 된다.

구현된 프로그램상에서 사용자는 임의의 네트워크 구조를 마우스를 사용하여 직접 생성할 수 있게 하였다. 또한 프로그램이 랜덤 네크워크를 자동으로 생성할 수 있는 기능도 포함하고 있다.



(그림 2) 프로그램에서 마우스를 통하여 네트워크를 수작업으로 구성하는 경우
(Fig. 2) Manual construction of a network using a mouse.

(그림 2)는 수작 업으로 구성하는 네트워크 모형을 보여 주고 있으며, 후의 실험에 있어 최적값을 미리 정하고 구성한 네트 워크에 대하여 최적값을 발견하는지를 검토하는 실험에 아주 유용하게 사용되었다.

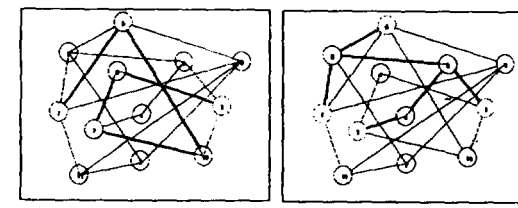
4. 진화에 의한 멀티캐스트 라우팅

본 절에서는 멀티캐스트 라우팅을 위해 교차(Crossover), 선택(Selection), 돌연변이(Mutation) 연산자가 어떻게 구현되었는지에 대해서 기술하고 그 효용성을 검토해 보고자 한다.

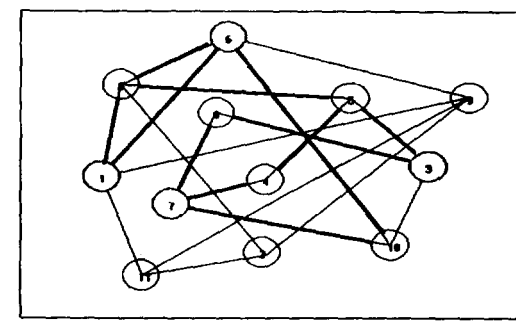
4.1 교차(Crossover) 연산자

교차 연산자는 유전자 알고리즘에 있어 두 개의 부모 염색체의 일부를 끊어서 상호 교환함으로써 두 부모의 유전 인자를 갖는 자식을 만드는 것을 말한다. 개체 표현이 이진스트링 형태인 일반적인 경우에 있어서는 두 부모를 의미하는 이진스트링을 각각 끊어서 합침으로 수행 할 수 있다.

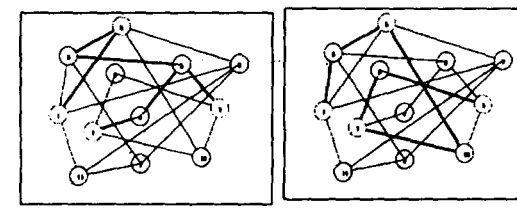
(a) 2개의 부모 트리



(b) 부모를 혼합하였을 경우 생성된 서브 네트워크



(c)임의로 생성된 2개의 자식



(그림 3) Croossover 연산
(Fig. 3) Crossover operation

한 개체의 표현이 트리 형태가 되는 본 실험에 있어서는 이와는 다른 방법이 필요하였으며 좀더 효율적인 교차 연산 수행을 위하여 다음과 같은 방법을 사용하였다. 교차 연산자로 스타이너 트리의 특성을 유지시키면서도 두 트리의 정보를 계속 유지할수 있도록 하기위하여 일종의 '합병-재분배' 방법을 사용해 보았다. 이 방법은 먼저 이전 세대의 트리 중에서 선택 연산자에 의해 두 개의 부모 트리를 선택하여 혼합함으로써 새로운 서브 네트워크를 생성한다. 이러한 과정에 의해 생성된 서브 네트워크는 cycle을 지니는 그래프를 생성하며 원래 네트워크 형태의 일부로서 그 일부분에 대해서 원래 네트워크와 같은 정보를 유지하게 된다. 서브 네트워크가 생성되었다면 이를 다시 처음 세대를 구성하기위한 초기화 과정에 사용한 방법을 이용하여 다시 무작위로 두 개의 스타이너 트리를 생성하고, 이것은 자손으로서 다음 세대에 추가된다.

(a)는 선택된 두 개의 부모이고 두 부모를 합병하였을 때 (b)와 같은 서브 네트워크가 생성된다. (c)는 (b)에서 생성된 서브 네트워크를 초기화 과정에 넣었을 경우 얻어진 두 자식을 보여준다. 만일 부모가 같은 형태의 트리인 경우 두 자식 역시 필연적으로 부모와 같은 두개의 자손이 생성된다.

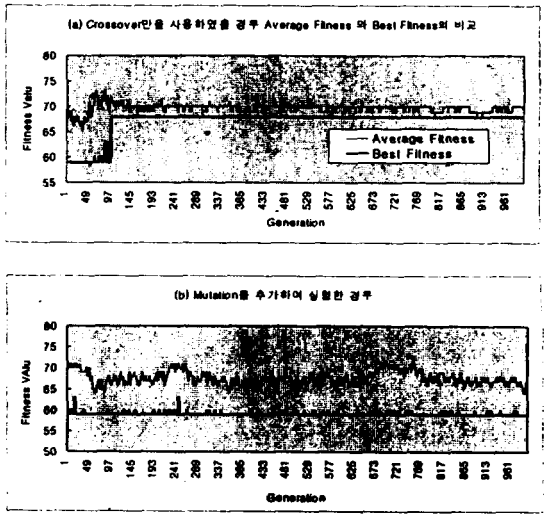
4.2 선택(Selection) 연산자

이전 세대에서 다음세대의 자손을 만들기위한 부모를 선택하는 것은 우수한 적합도를 가진(적합도값이 낮은 즉 통신 코스트 비용이 낮은)개체에 높은 확률을 부여하여 두 개의 부모 개체를 선택한다. 여기서 실험한 네트워크의 적합도 값은 통신 코스트 비용을 랜덤하게 부여한 수치에 의해서 계산되어진다.

4.3 돌연변이(Mutation) 연산자

돌연변이 연산자는 부모 염색체의 일부를 임의로 변경하는 방법으로 일반적인 이진스트링 표현에 있어서는 돌연변이율에 따라 임의의 비트를 변경함으로써 수행 할 수 있다. 한 개체의 표현이 트리가 되는 본 연구에 있어서는 트리 구조에 대한 돌연변이 연산자가 필요하며 여러 가지 생각될 수 있는 방법중 다음과 같은 방법에 착안하였다.

돌연변이의 방법으로, 교차 연산에있어 배제되어지는 노드들에 촛점을 두어 실험하였다. 즉 우수한 부모개체에 의해 만들어진 자식 트리는 다음 세대에 있어 자신이 가지고 있는 노드들의 합병에 의해서만 이루어진다. 하지만 이것은 최적값을 찾아가기 위해 사용되어지는 노드들을 고정시킬 위험성을 내포한다. 한 번 나빠진 세대는 노드의 고정화로 인하여 다음 세대에서 더욱 나쁜 세대로 만들어 갈 수도 있다. 실험 결과 중간 세대에 있어 한 번 나빠진 세대는 위의 이유로 인해 다음세대에 계속 나빠지는 경우가 실험 횟수의 1/4의 확률로 발생했다. 따라서 교차연산에 의해 만들어진 각 세대의 트리 개체에 있어 배제되어진 노드들을 돌연변이 연산자를 사용하여 일정 비율 추가 시킨다면 다음 세대를 구성하는데 있어 좀더 변화를 줄 수 있고 위 와 같이 나빠지는 현상은 일어나지 않게 된다.



(그림 4) Mutation을 추가하여본 경우와 하지 않은 경우의 비교

(Fig. 4) Comparison of performance with and without mutation.

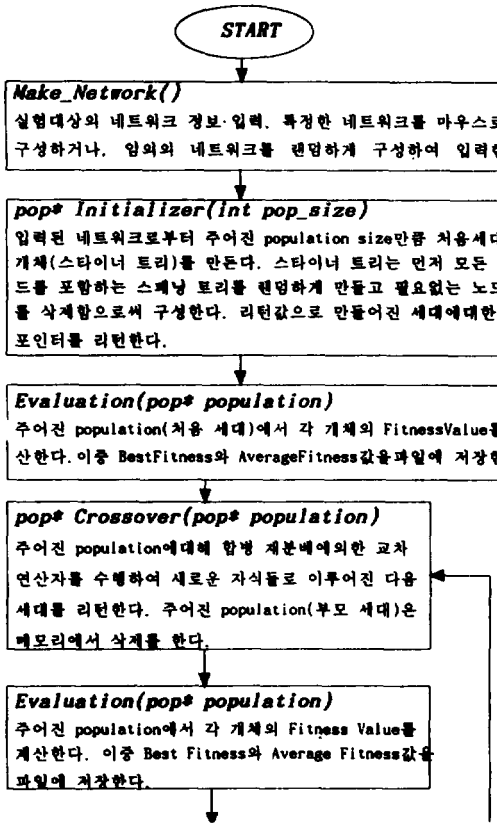
(그림 4)는 돌연변이 연산자를 사용할 경우 좀 더 좋은 세대를 구성하여 나갈 수 있음을 실험적으로 보여준다. (그림 4(a))는 Crossover만을 사용한 경우로서 알고리즘상의 노드의 고정화의 문제가 발생할 경우 극히 나빠지는 상태를 보여준다. (b)의 경우는 똑같은 상황에 대해 Mutation을 추가하였을 경우 그러한 문

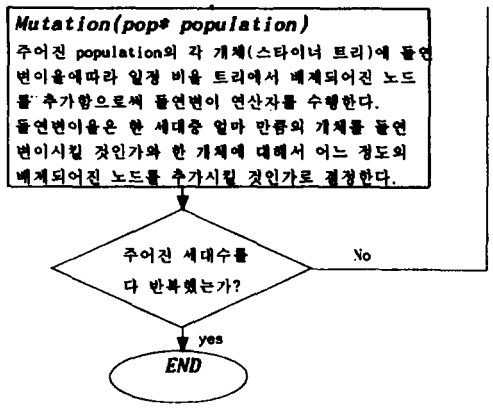
제가 해결되는 것을 보여준다. 여러 번의 실험 결과 Mutation을 사용함으로써 Crossover 연산만을 사용한 경우의 문제점을 해결할 수 있음을 확인할 수 있었다.

5. 실험을 위해 사용된 프로그램의 전반적 알고리즘

본 절에서는 위의 3, 4절에서 설명한 네트워크 구성 방법 및 유전 연산자를 이용하여 구현한 프로그램의 수행과정을 순서도로 요약하여 기술한다.

프로그램은 Windows 95와 Windows NT를 사용한 Pentium 166Mhz, Memory 32M, H.D. 2G 상에서 Visual C++ 4.0로 구현되었다.





(그림 5) 구현된 프로그램의 수행 과정을 기술하는 순서도
(Fig. 5) Flow chart description of the implemented program

6. 실험 결과 및 고찰

여기서는 3가지의 실험을 통하여 진화 알고리즘을 이용한 멀티캐스트 경로배정 방식의 효용성을 알아보고자 한다.

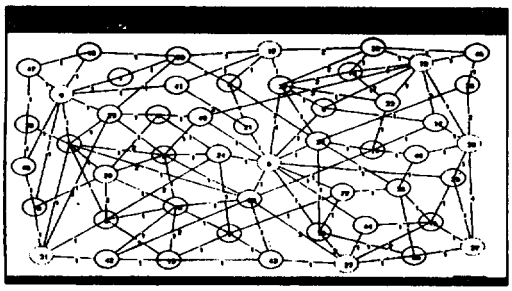
최적값을 아는 문제에 대해 위의 방법을 적용할 경우찾은 해에 대한 최적화 정도를 확인 할 수 있으므로 NP-complete이지만 즉 시간이 많이 걸리지만 모든 경우의 수(모든 경우의 최소비용 스타이너 트리)를 다 따져보는 프로그램을 따로 작성하였다. 계산 시간 문제를 줄이기 위해 수퍼컴퓨터에서 프로그램을 수행해 보았으며 대략 25개 정도의 노드를 가진 네트워크에서는 최적값을 얻어 낼 수 있었다. 차후 몇몇의 실험은 이러한 정보를 가지고 그 효용성을 검토할 수 있었다. 또한 위의 방법 외에도 의도적으로 최적값을 정하고 마우스로 수작업으로 구성한 특정한 네트워크에 대해서도 실험을 함으로써 알고리즘의 정확도를 검토할 수 있었다.

6.1 실험 1

실험 1은 마우스로 최적의 값을 먼저 구성하고 최적값보다 조금씩 큰 값들을 마우스로 구성한 특정 네트워크에 대하여 실험하였다.

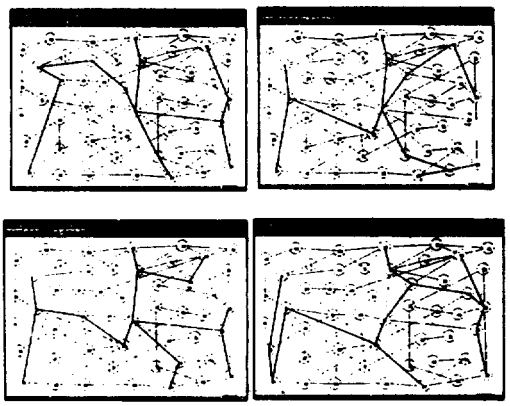
(그림 6)은 사용된 네트워크의 모양과 각 링크에 할당된 경비를 보여주고 있다. 네트워크에 포함된 노드 수는 50이고 각 노드당 링크 수는 평균 5개이며 멀티

캐스트에 참여하는 멤버노드(그림에서 흐리게 표시된 노드)는 8개로 하였다. 실험에서 population size는 100으로 하였고 돌연변이율은 각 세대에서 40%로만큼의 개체를 돌연변이 시켰으며 돌연변이의 대상이 되는 개체에 있어서는 30%로 개체(스타이너 트리)에 속하지 않은 노드를 연결시켰다. 의도적으로 작성한 최소 비용 스타이너 트리의 적합도는 12이다. (그림 7)은 (그림 6)의 네트워크에 대해서 스타이너 트리가 진화하는 과정을 보여준다.



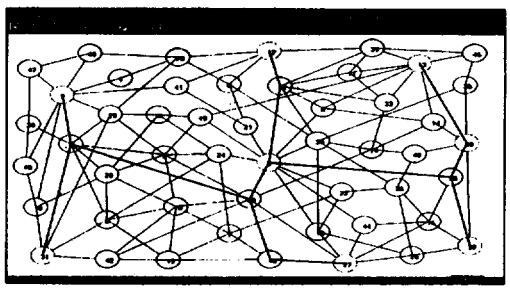
Network node = 100, Member node = 8, Average #link per node = 5, Population size = 100, Mutation rate = 0.4~0.3

(그림 6) 실험에 사용된 네트워크
(Fig. 6) The network used in the experiment

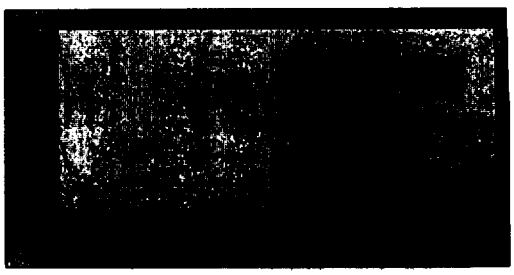


(그림 7) 진화하는 스타이너 트리
(Fig. 7) Steiner trees in evolution.

(그림 8)은 진화 과정 중 최적값으로 얻어진 스타이너 트리를 보여주며 Fitness value는 12로서 실험이 의도한 최적값을 찾을 수 있었다. (그림 9)의 그래프는 위의 실험에서 1000 세대까지 진화시켰을 경우 100세대의 진화 중의 각 세대별 평균 Fitness value와 세대에서 가장 좋은 Fitness value의 비교 및 변화를 보여준다. 100회의 실험을 하여 얻어진 데이터의 평균값을 그래프화 하였다. 100회의 실험 모두 최적값을 찾으며 1000세대를 진화시켰을 경우 걸린 시간은 평균적으로 30초 정도였다. 또한 모든 실험에서 10세대내에 최적값을 찾았고 Average Fitness와 Best Fitness가 가까와 지는 것을 확인할 수 있었다.



(그림 8) 진화에 의해 구해진 최적 스타이너 트리
(Fig. 8) The optimal Steiner tree obtained by evolution.



(그림 9) Average fitness와 best fitness의 비교
(Fig. 9) Comparison of average fitness and best fitness.

6.2 실험 2

실험 2는 랜덤하게 구성한 네트워크에 대해서 실험하여 보았다. 이 경우에 있어 최적값을 알 수 없으므로 따로 제작한 모든 경우의 수(모든 스타이너 트리)를 따져보는 프로그램을 통해 최적값을 알고 실험을 해보았다. 이와 같은 경우 시간상의 문제로 인하여 노드 수에 있어 20개 정도까지 밖에 실험을 할 수 밖에 없었다. 추가로 멀티캐스트 경로 배정을 위하여 사용

되어진 이미 나와있는 다른 알고리즘을 비교해 보면 좋을 것 같아 그중 Takahashi등의 경험적 알고리즘을 택하여 프로그램으로 구현하여 비교해 보았다.

Takahashi등의 경험적 알고리즘[14]은 서브트리를 구성해 나감으로써 최소 비용 스타이너 트리를 찾아내는 방법이다. 초기에 서브트리는 단 하나의 멀티캐스트 구성원만으로 형성되며, 이후 그 서브트리에 최소 비용의 경로로 연결될 수 있는(아직 서브트리에 속하지 않은) 구성원이 하나씩 추가됨으로써 트리를 구성해 나간다. 이와 같은 과정은 모든 구성원이 서브트리에 포함될 때 까지 계속된다.

〈표 1〉은 실험을 통하여 Takahashi등과 진화 알고리즘을 비교해 본 결과이며 보는 바와 같이 두 알고리즘에 있어 노드 수 20이하로 실험해본 결과 둘다 최적값을 발견하는 것을 알 수 있었다. 진화 알고리즘에 있어 population size는 50으로 하였고, 돌연변이율은 0.4*0.3이며 100세대까지 진화시켰다.

〈표 1〉 최적값을 아는 경우 Takahashi 등의 방법과 진화 알고리즘의 비교

〈Table 1〉 Comparison of the methods by Takahashi and the evolutionary algorithm : optimal solution known

수행된 실험		최적값	Takahashi 등의 경우	진화 알고리즘의 경우
A	네트워크 노드 수:15 멀티캐스트 멤버 노드수:4	24	24	24
B	네트워크 노드 수:20 멀티캐스트 멤버 노드수:4	21	21	21
C	네트워크 노드 수:20 멀티캐스트 멤버 노드수:4	31	31	31
D	네트워크 노드 수:20 멀티캐스트 멤버 노드수:8	59	59	59

6.3 실험 3

실험 3은 랜덤하게 구성된 네트워크에 있어 네트워크 노드 수가 50개, 100개인 경우를 실험하여 보았다. 이 경우에 있어 문제가 되는 것은 최적값을 모르고 실험을 하게 되므로 실험 결과가 과연 최적값을 찾는지를 확인할 수 없다는 것이다. 이와 같은 문제를 위해서 본 실험에서는 실험 2에서 비교된 Takahashi등의 경험적 방법을 같이 비교해 봄으로써 그 효용성을 검토해 보고자 하였다. 실험은 네트워크 노드 수 50개, 100개로 구성된 네트워크를 각각 50, 100개씩 랜덤하게 구성하여 실험을 해보았다.

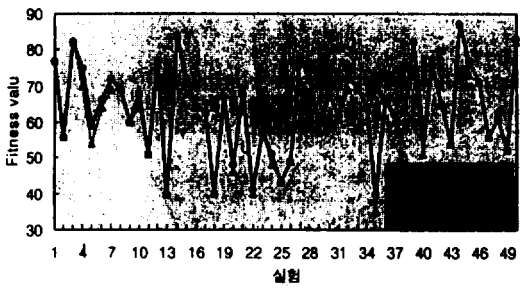
〈표 2〉는 주어진 네트워크에 대하여 Takahashi등의 경험적 방법과 진화 알고리즘을 이용한 방법에 대해 승패를 따져본 결과이다. 50노드의 경우 50번의 실험중 Takahashi등이 0번 이기고 진화 알고리즘의 경우 24번 이기고 나머지 26번은 비겼다. 그리고 매 회마다 얻어진 Fitness값을 더해 평균한 값은 Takahashi등이 63.94이었고 진화 알고리즘의 경우 62.56이었다. 그리고 수행된 시간에있어서는 50노드의 경우 Takahashi등이 평균적으로 대략 1초정도 걸렸으며, 진화 알고리즘의 경우 50세대까지 진화하여 걸린 시간은 약 29초였다. 실험 3도 실험 1, 2에서처럼 각 조건에

〈표 2〉 최적값을 모르는 경우 Takahashi 등의 방법과 진화 알고리즘의 비교

〈Table 2〉 Comparison of the methods by Takahashi and the evolutionary algorithm : optimal solution unknown

경 우 조 건	Takahashi 등의 경우	진화 알고리즘의 경우
[네트워크 조건] 네트워크 노드수:50 멀티캐스트 멤버 노드수:8 한 노드당 평균 link수:5 [진화 알고리즘 조건] population size:500 돌연변이율:0.4*0.4 진화한 세대수:50	50번중 0번 이김 50번의 평균 Fitness값:63.94 평균 수행 시간:1초	50번중 24번 이김 50번의 평균 Fitness값:62.56 평균 수행 시간:29초
[네트워크 조건] 네트워크 노드수:100 멀티캐스트 멤버 노드수:12 한 노드당 평균 link수:5 [진화 알고리즘 조건] population size:500 돌연변이율:0.4*0.4 진화한 세대수:50	100번중 26번 이김 100번의 평균 Fitness값:110.1 평균 수행 시간:2.5초	100번중 64번 이김 100번의 평균 Fitness값:109.4 평균 수행 시간:2분 7초

있어 평균 20세대안에서 최적의 값으로의 수렴이 나타나는 현상을 볼 수 있었고, population size를 500으로 놓고 이 시점까지의 수행시간을 계산해 보면 평균적으로 대략 50노드가 10초, 100노드가 23초 걸림을 알 수 있었다. (그림 10)의 그래프는 50노드의 경우 50번동안의 Takahashi등과 진화 알고리즘이 찾은 Fitness의 비교이다.



(그림 10) 50노드의 경우 50번 실험의 두 알고리즘의 fitness비교

(Fig. 10) Fitness comparison of two algorithms for 50 runs for the 50-node network problem

7. 결 론

본 논문에서는 멀티캐스트 경로배정을 위해 진화 알고리즘 방식을 적용하여 최적의 스타이너 트리를 찾는 문제를 다루었다. 실험 1, 2, 3에서와 같이 진화 알고리즘을 통한 멀티캐스트 경로 배정은 대체로 만족할 만한 결과를 제시해 주고 있으며, 또한 Takahashi 등의 경험적 알고리즘과 비교하였을 때, 뒤지지 않는 성능을 보여 주고 있다. 물론 최적화에 걸리는 시간상의 문제가 제시되고 있지만 진화 알고리즘에 있어 세대수를 반복함으로써 일정의 시간을 요하게 되는 것은 당연한 것이라 사료되며, 이제까지의 실험한 모든 실험에 있어 평균적으로 20세대 내에 최적값에 수렴하고 이 때까지의 시간은 대략 25초 정도 걸리는 것을 보면 이를 어느 정도 보완해 줄 수 있다고 본다.

지금까지의 실험은 멀티캐스트 경로 배정시 진화 알고리즘을 사용하는 방법과 그 유용성에 초점을 맞추었다. 이러한 이유 때문에 방대한 양의 네트워크 크기, 각 노드간의 링크의 가중치에 있어 네트워크의 traffic, bandwidth 등의 고려와 같은 다른 실용적인 조건들에 대해서는 실험하지 못하였다.

향후 연구에서는 문제의 폭을 더욱 확대하고 이렇게함으로써 추가적으로 발생 가능한 문제점과 이의 개선 및 성능 향상을 위한 방안을 모색해야 할 것이다.

참 고 문 헌

[1] Frank, A.J., Wittie, L.D., and Bernstein, A.J., "Multicast Communication on Network Computers", IEEE Software, Vol.2, No.3, pp.46-61, 1985.

[2] Kompella, V.P., Pasquale, J.C., and Polyzos, G. C., "Multicast Routing for Multimedia Communication", *IEEE/ACM Trans. on Networking*, Vol. 1, No.3, pp.286-292, 1993.

[3] Wall, D.W., "Mechanisms for Broadcast and Selective Broadcast", *Ph.D. dissertation, Department of Electrical Engineering and Computer Science*, Stanford University, 1980.

[4] Dalal, Y.K., and Metcalfe, R.M., "Reverse Path Forwarding of Broadcast Packets", *Comm. of the ACM*, Vol.21, No.12, pp.1040-1048, 1978.

[5] Winter, P., "Steiner Problem in Networks: A Survey", *Networks*, Vol.17, pp.129-167, 1987.

[6] Karp, R.M., "Reducibility among Combinatorial Problems", *Introduction to Complexity of Computer Computations (R.E. Miller, J.W. Thatcher eds.)*, pp.85-104, New York: Plenum Press, 1972.

[7] Hwang, F.K., and Richards, D.S., "Steiner Tree Problems", *Networks*, Vol.22, pp.55-89, 1992.

[8] T. Baeck and H.-P. Schwefel, "An Overview of Evolutionary Algorithms for Parameter Optimization", *Evolutionary Computation*, 1(1):1-23, 1993.

[9] D. E. Goldberg, "Genetic and Evolutionary Algorithms Come of Age", *Communications of the ACM*, 39(3):113-119, 1994.

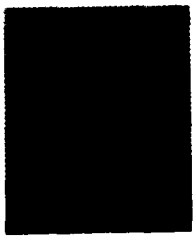
[10] M. Srinivas and L. M. Patnaik, "Genetic algorithms: a survey", *IEEE Computer*, June 1994, pp. 17-26.

[11] Feasibility Study Committee of the Real-World Computing Program, *The Master Plan for the Real-World Computing Program*, MITI, Japan, May 1992.

[12] B.T. Zhang and H. Muehlenbein, "Balancing Accuracy and Parsimony in Genetic Programming", *Evolutionary Computation*, 3(1):17-38, 1995.

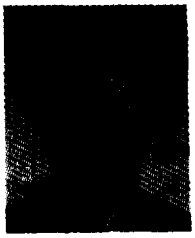
[13] H. Esbensen, "Finding (Near-)Optimal Steiner Trees in Large Graphs", *Proc. Sixth Int. Conf. on Genetic Algorithms*, San Francisco, CA:Morgan Kaufmann, pp.485-491, 1995.

[14] Takahashi, H., and Matsuyama, A., "An Approximate Solution for the Steiner Problem in Graphs", Math. Japonica, Vol.24, pp.573-577, 1980.



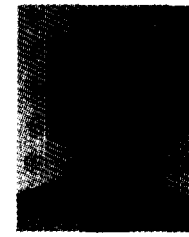
이 창 훈

1975년 연세대학교 수학과 학사
1977년 한국과학기술원 전산과 석사
1993년 한국과학기술원 전산과 박사
1977년~1980년 고려 System (System Engineer)
1980년~현재 건국대학교 컴퓨터공학과 교수
관심분야:인공지능, 전문가 시스템, 에이전트 시스템, 정보검색



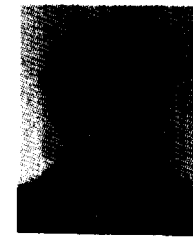
장 병 탁

1986년 서울대학교 컴퓨터공학과 학사
1988년 서울대학교 컴퓨터공학과 석사
1992년 독일 Bonn 대학교 컴퓨터과학과 박사
1988년~1992년 Bonn 대학교 AI Lab. 연구원
1992년~1995년 독일국립전산학연구소(GMD) 연구원
1995년~1997년 건국대학교 컴퓨터공학과 조교수
1997년~현재 서울대학교 컴퓨터공학과 조교수
관심분야:인공지능, 기계학습, 신경망, 진화 알고리즘



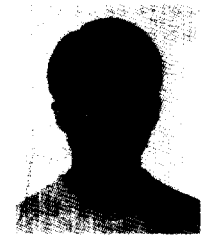
안 상 현

1986년 서울대학교 컴퓨터공학과 학사
1988년 서울대학교 컴퓨터공학과 석사
1988년~1989년 데이콤 재직(연구원)
1993년 University of Minnesota 전산학과 박사
1994년~1997년 세종대학교 컴퓨터공학과 교수
1998년~현재 서울시립대학교 전산통계학과 교수
관심분야:ATM, 멀티캐스트 라우팅, 인터넷 등



곽 주 현

1995년 건국대학교 컴퓨터공학과 학사
1997년 건국대학교 컴퓨터공학과 석사
1997년~현재 건국대학교 컴퓨터공학과 박사과정
관심분야:GA, 자연어 처리, VRML



김 재 훈

1997년 건국대학교 컴퓨터공학과 학사
1997년~현재 건국대학교 컴퓨터공학과 석사
관심분야:인공지능, GA, 데이타베이스