유전자 알고리즘을 이용한 멀티캐스팅 경로 설정

조 병 헌*, 오 하 령*, 성 영 락*, 이 면 섭**, 안 현 식*
*국민대학교 전자공학과
**인천전문대학교 컴퓨터정보과
E-mail : d995552@hanmail.net

A Genetic Algorithm for Multicasting Routing Problem

Byeong Heon Cho*, Ha Ryoung Oh*, Yeong Rak Seong*
Myeon Seob Lee**, Hyun Sik Ahn*
*Dept. of Electronics Engineering, Kookmin University
**Dept. of Computer & Information Science

요 약

본 논문에서는 멀티캐스팅 경로 설정에 많이 사용되는 스타이너 트리 문제를 위한 새로운 유전자 알고리즘을 제안하였다. 제안된 알고리즘에서는 휴리스틱 방법으로 스타이너 트리를 구성하고, 여기에 유전자 알고리즘을 반복 수행하여, 보다 최적에 가까운 스타이너 트리를 구하도록 하였다. 제안 알고리즘을 OR-라이브러리의 스타이너 트리 문제 중 38개의 예제에 대하여 실험한 결과, 21개의 예제에서 최적 해를 찾아내었다. 또한 반복 계산 초기에 빠르게 수렴하는 성질을 볼 수 있었다.

1. 서론

최근 들어 멀티캐스팅 통신에 대한 연구가 활발하다. 멀티캐스팅은 화상 및 음성 회의, 실시간 비디오 전송, 소프트웨어 업데이트, 웹 캐싱, 웹 클러스터링 등의 분야에서 사용된다. 이러한 멀티캐스팅 서비스의 경로 설정을 위해 최소비용 스패닝 트리에 의한 방법, 근원지 기반 방법, 공유 트리 방법, 통계에 의한 방법, 스타이너 트리를 이용한 방법 등이 이용되고 있다[1,2].

통신의 많은 문제에서처럼 멀티캐스팅 문제는 그래프로 표현될 수 있다. 그래프에서 각 노드는 통신망내의 라우터나 컴퓨터이고, 링크는 이들간의 직접적인 연결 망을 나타내며, 연결 망의 비용이 가중치로 표시된다. 그래프에서 임의의 점들만을 연결하면서 링크 비용의 합이 최소가 되도록 하는 트리를 그 임의의 점들에 대한 최소 비용 스패닝 트리라고 한다. 또한 여기에 전체 링크 비용 감소에 도움이 된다면 새로운 노드들을 추가한 트리를 스타이너 트리라고 한다. 이런 측면에서 스타이너 트리는 멀티캐스팅을 위한 최상의 경로이다. Ramanathan은 멀티캐스트 트리를 생성하는 문제를 스타이너 트리 문제의 하나의 유형으로 공식화하였다[3].

최적의 스타이너 트리를 찾는 문제는 완전-NP 문제로서 다항 시간 내에 해결하는 알고리즘이 존재하지 않는다. 그래서 지금까지 이 문제를 해결하기 위해서 휴리스틱 방법이 많이 연구되었다. 휴리스틱 방법은 다항 시간 내에 해를 찾아내기는 하지만, 최적해를 찾을 수 없다는 단점을 가진다. 그래서 유전자 알고리즘을 이용하여 최적해에 보다 근사한 해를 찾으려는 연구들이 진행되었다[4,5].

본 논문에서는 유전자 알고리즘을 이용하여 스타이너 트리를 생성하는 새로운 알고리즘을 제안한다. 1975년 Holland에 의해 소개된 유전자 알고리즘은 자연 생태계의 진화이론을 이용한 탐색과정이다[6]. 유전자 알고리즘은 탐색공간에서 찾고자 하는 해의 후보를 나타내는 염색체(chromosome), 염색체들 사이의 정보교환을 위한 연산자인 교배 연산자와 돌연변이 연산자, 그리고 염색체의 적합성을 나타내는

함수(fitness function) 등으로 정의된다.

어떤 그래프에서 특정 노드들을 목적 노드로 하는 스타이너 트리는 매우 다양하게 존재한다. 그러나 효율적인 멀티캐스팅을 위해서는 이들 중에서 최적의 스타이너 트리를 구해야 한다. 본 논문에서 제안하는 알고리즘은 이러한 과정을 표현하는 것이다. 즉, 수행 초기에는 임의의 스타이너 트리를 가지지만 계산이 진행됨에 따라 점차 최적에 가까운 스타이너 트리를 구한다.

2. 제안 알고리즘

염색체 알고리즘을 위해서는 염색체의 표현과 적합도 함수, 교배 연산자, 돌연변이 연산자가 정의되어야 한다.

2.1 염색체

유전자 알고리즘에서 염색체는 해결하고자 하는 문제에 대한 해를 표현한다. 그러므로 본 논문에서는 멀티캐스트 경로가 염색체로 표현된다. 제안하는 알고리즘에서 염색체는 그림 1과 같이 노드들의 배열로 나타낸다. 염색체와는 별도로 각 염색체와 관련 있는 링크 정보도 함께 저장된다.

n1	n2	n3	n4	n5	n6	n7	n8	n9	n10

그림 1 염색체 표현 방법

노드들의 배열로 염색체를 표현했고, 이들을 이용하여 스타이너 트리를 생성하기 때문에 모든 염색체에 대하여 유효한 스타이너 트리를 얻을 수 있는 장점이 있다.

2.2 염색체 초기화

염색체는 다음과 같은 과정을 통해 초기화된다.

a. 목적지 노드들을 경유하는 스타이너 트리 계산
b. a에서 얻은 링크에서 목적지 노드를 포함하지 않는 링크 제거
c. 최종적으로 계산된 링크들을 염색체 별로 저장
d. 저장된 링크에 포함된 노드들 중에서 목적지 노드를 제외한 나머지 노드들을 염색체 배열에 저장

예를 들어 그림 2의 상황을 가정해 보자. 여기에서 목적지 노드는 1, 3, 7, 9번 노드이다. 그림 3(a)는 스타이너 트리를 계산한 것이다. 여기서 목적지 노드를 포함하지 않는 4개의 링크를 제거하면 그림 3(b)의 형태를 갖는다. 또한 아무런 링크와도 연결되지 않은 노드들을 제거한다(그림 3(c)). 최종적으로 남은 4개의 링크들은 염색체와 별도로 저장되고, 링크에 포함된 노드들 중 목적지 노드를 제외한 2, 4, 6, 8번 노드가 염색체로 표현된다(그림 3(d)).

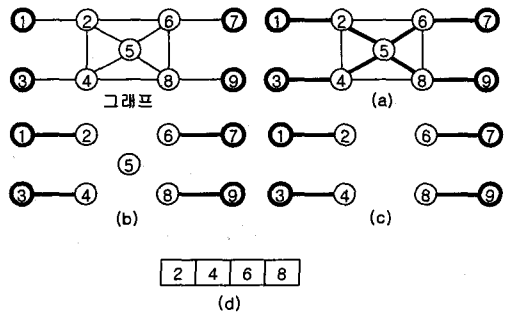

그림 2 염색체 초기화

위에서 초기의 스타이너 트리를 생성할 때 적용 가능한 알고리즘으로는 Glover 등이 제안한 Tabu Heuristic 방법[7], Takahashi와 Matsuyama가 제안한 Shortest Path Heuristic(SPH) 방법[8]이 있다. 본 논문에서는 두 가지 방법 중 상대적으로 속도가 빠르면서도 유사한 계산 결과를 보이는 SPH 방법을 이용한다.

2.3 적합도 함수

염색체의 적합도는 각 염색체가 다음 세대로 유전되는 확률에 영향을 미친다. 그림 3은 본 논문에서 사용하는 적합도 함수이다.

```
1   for each ch in chromosomes
2     if (!ch.IsConnected())
3       ch.MakeConnected(); // add intermediate nodes
4     endif
5     ch.CalcTotalCost();
6   endfor
7   chromosomes.sort();          // ascending order of cost
8   for each ch in chromosomes
9     ch.fitness(2^(N-r));       // r: rank, N: population
10  endfor
```

그림 3 적합도 함수

앞서 언급한 바와 같이 염색체는 계산 도중의 해

이므로, 하나의 완전한 스타이너 트리가 되어야 한다. 그러나 이후에 설명할 교배 및 돌연변이 연산자들을 적용하는 과정에서 염색체가 트리를 구성하지 못하는 일이 발생한다. 그러므로 적합도를 계산하기 위해서는 우선 각 염색체에 저장된 링크만을 이용하여 목적지 노드들과 염색체에 포함되어 있는 노드들을 모두 연결할 수 있는지 검사해야 한다(그림 3(2)). 이때 만약 연결되어 있지 않다면 목적지 노드들과 염색체의 노드들을 연결되어 있는 노드들을 하나의 수퍼 노드로 간주하고 이들을 최단 거리로 연결하기 위한 노드들을 염색체에 추가한다(그림 3(3)). 저장된 링크 비용의 합을 염색체의 비용으로 저장되며(그림 3(5)) 이에 따라 염색체들이 정렬된다(그림 3(7)). 최종적으로 2^{N-r}(N:염색체의 총 개수, r: 염색체의 순위)의 수식을 이용하여 각 염색체의 적합도를 계산한다(그림 3(9)).

2.4 교배 연산자

앞서 설명한 적합도 함수에 기반하여 두 개의 부모 염색체를 선택된다. 선택된 염색체들은 교배 연산자를 통하여 새로운 두 개의 자식 염색체를 생성한다. 일반적인 교배 연산자로는 1-point, 2-point 교배 연산자 등이 있다. 전자는 두 염색체의 임의의 위치 이후의 내용을 교환하는 방법이고, 후자는 두 개의 임의의 위치 사이의 내용을 교환하는 방법이다. 그러나 멀티캐스팅 경로 설정 문제나 TSP(Traveling Salesman Problem) 문제와 같은 경로 문제에는 무효한 해를 만들 가능성이 있기 때문에 적용하기 곤란하다.

본 논문는 스타이너 트리를 생성할 때 특정 노드를 반드시 포함시켜야 총 비용을 줄일 수 있을 것이라는 아이디어를 기초하여 교배 연산자를 구현하였다. 그래서 비용이 작은 두 염색체에 공통으로 포함되어 있는 노드들은 반드시 포함되어야 할 노드로 간주하여 자식 염색체로 유전시켰다.

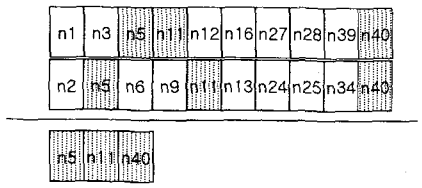

그림 5 교배 연산자

2.5 돌연변이 연산자

돌연변이 연산자는 조기수렴을 방지하기 위해, 또한 검색 영역을 확대하기 위해 사용된다. 일반적인 돌연변이 연산자는 비트 스트링에서 임의의 위치의 내용을 반전(0 ⇒ 1, 1 ⇒ 0)시킨다. 그러나 제안하는 알고리즘의 염색체 표현방법을 사용하면 존재하지 않는 노드가 나타날 가능성이 존재하기 때문에 임의의 염색체에서 임의의 노드를 추가하거나 삭제하도록 돌연변이 연산자를 구현하였다.

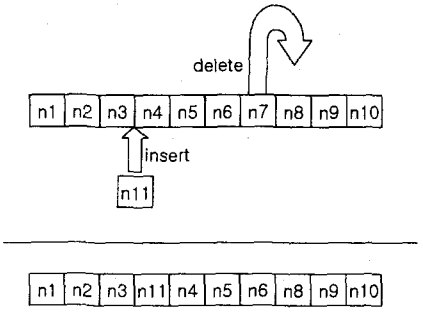

그림 6 돌연변이 연산자

3. 실험결과

본 논문에서 제안한 알고리즘을 OR-라이브러리[10]의 스타이너 트리 문제들에 대해 적용해 보았다. 총 38개의 예제를 적용하였고, 약 55%인 21개의 예제에 대하여 최적해를 얻을 수 있었다. 모든 예제들에 대해서는 최적해와 평균 3.1%의 차이가 있었다. 또한 기존에 제안된 알고리즘과의 성능 비교를 위해서 가장 성능 좋은 휴리스틱 방법으로 알려져 있는 SPH 반복 수행 알고리즘[9]의 결과와 비교하였다. 비교 결과 38개의 예제에 대하여 평균 2.5% 정도 성능이 떨어지는 것으로 나타났다.

또한 제안된 알고리즘의 수렴 특성에 대해서 살펴보았다. 그림 7은 steinc2의 문제에 대해서 계산이 진행되는 도중에 염색체로 표현된 스타이너 트리의 비용을 도식화한 것이다. 결과를 보면 처음 염색체가 초기화될 때 단순히 SPH 방법에 의해 구한 스타이너 트리에 비해서 연산이 반복될수록 비용이 감소함을 보여준다. 즉 연산이 반복됨에 따라 점차 최적의 스타이너 트리에 가까운 스타이너 트리가 계산되는 것이다. 또한 제안하는 알고리즘은 실행 초기에 빨리 수렴하는 특성을 보인다.

표 1 실험 결과

	Opt	SPH-I	GA	ΔOpt (%)	ΔSPH-I (%)
Steinb1	82	82	82	0.0	0.0
Steinb2	83	83	83	0.0	0.0
Steinb3	138	138	138	0.0	0.0
Steinb4	59	59	59	0.0	0.0
Steinb5	61	61	61	0.0	0.0
Steinb6	122	122	126	3.3	3.3
Steinb7	111	111	111	0.0	0.0
Steinb8	104	104	104	0.0	0.0
Steinb9	220	220	220	0.0	0.0
Steinb10	86	86	86	0.0	0.0
Steinb11	88	88	88	0.0	0.0
Steinb12	174	174	181	4.0	4.0
Steinb13	165	168	165	0.0	-1.8
Steinb14	235	235	235	0.0	0.0
Steinb15	318	318	321	0.9	0.9
Steinb16	127	127	127	0.0	0.0
Steinb17	131	131	131	0.0	0.0
Steinb18	218	218	225	3.2	3.2
Steinc1	85	85	85	0.0	0.0
Steinc2	144	144	144	0.0	0.0
Steinc3	754	754	769	2.0	2.0
Steinc4	1079	1081	1112	3.1	2.9
Steinc5	1579	1579	1622	2.7	2.7
Steinc6	55	55	55	0.0	0.0
Steinc7	102	102	102	0.0	0.0
Steinc8	509	512	536	5.3	4.7
Steinc9	707	714	769	8.8	7.7
Steinc10	1093	1098	1180	8.0	7.5
Steinc11	32	32	32	0.0	0.0
Steinc12	46	46	46	0.0	0.0
Steinc13	258	263	275	6.6	4.6
Steinc14	323	327	349	8.0	6.7
Steinc15	556	558	605	8.8	8.4
Steinc16	11	11	12	9.1	9.1
Steinc17	18	18	18	0.0	0.0
Steinc18	113	121	129	14.2	6.6
Steinc19	146	155	170	16.4	9.7
Steinc20	267	267	300	12.4	12.4
			avg	3.1	2.5

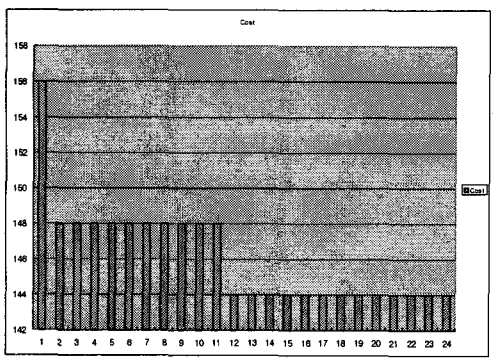

그림 7 steinc2 예제의 세대별 비용 변화 추이

4. 결론

본 논문에서는 멀티캐스팅 경로 설정에 많이 사용되는 스타이너 트리 문제를 새로운 유전자 알고리즘을 이용하여 해결하였다. 본 알고리즘에서는 SPH를 이용하여 스타이너 트리를 생성하고 여기에 교배 및 돌연변이 연산자를 반복 적용시켜 최적해에 가까운 해를 구하도록 하였다. 제안 알고리즘을 OR-라이브러리의 스타이너 문제에 적용하여 실험하였다. 실험 결과 적용된 38개의 예제 중에서 55%에 달하는 21개의 예제에서 최적의 해를 찾아내었다. 전체 문제에 대하여 최적해와 평균 3.1%의 차이가 있었다. 또한 기존에 가장 성능이 좋은 휴리스틱 알고리즘인 SPH-I와 비교한 결과 평균 2.5% 차이를 보였다. 제안 알고리즘은 몇몇 예제에 대해서는 수행시간이 상당히 오래 소요되는 단점이 있지만, 추후 최적의 해에 더 근접시키고 계산 속도 향상시키기 위한 연구를 진행할 예정이다.

참고 문헌

[1] Kosiur. D., IP Multicasting, Wiley, 1998.

[2] Huitema. C., Routing in the Internet, Prentice Hall, 1995.

[3] S. Ramanathan, Multicast Tree Generation in Networks with Asymmetric Links, IEE/ACM Transactions on Networking, Vol. 4, No. 4, Au gust 1996

[4] A. Kapsalis, V. J. Rayward-Smith, G. D. Smit h, Solving the Graphical Steiner Tree Problem Using Genetic Algorithms," Journal of the Ope rational Research Society, Vol. 44, No. 4, pp. 3 97-406, 1993.

[5] H. Esbensen. Computing near-optimal solution s to the Steiner problem in a graph using a g enetic algorithm. Networks, 26:173-185, 1995.

[6] David E. Goldberg, Genetic Algorithms in Sear ch, Optimization & Machine Learning, Addison Wesley, 1989.

[7] J. Xu, S. Y. Chiu, and F. Glover, "Probabilistic tabu search for telecommunications network de sign", Telecommun Syst 6, pp.117-125, 1996

[8] H. Takahashi and A. Matsuyama. "An approxi mate solution for the Steiner problem in graph s", Math. Japonica, 24, pp.573-577, 1980.

[9] Pawel Winter, J. MacGregor Smith,"Path-Dista nce Heuristics for the Steiner Problem in Undi rected Networks,"Algorithmica, Vol. 7, pp. 309 -327, 1992.

[10] http://www.ms.ic.ac.uk/info.html