ISSN 1229-800X The Transactions of the Korean Institute of Electrical Engineers Vol. 64P, No. 4, pp. 315~320, 2015
http://dx.doi.org/10.5370/KIEEP.2015.64.4.315

유전 알고리즘의 연산처리를 통한 개선된 경로 탐색 기법

Improved Route Search Method Through the Operation Process of the Genetic Algorithm

지 홍 일* · 서 창 진†
(Hong-il Ji · Chang-jin Seo)

Abstract - Proposal algorithm in this paper introduced cells, units of router group, for distributed processing of previous genetic algorithm. This paper presented ways to reduce search delay time of overall network through cell-based genetic algorithm. As a result of performance analysis comparing with existing genetic algorithm through experiments, the proposal algorithm was verified superior in terms of costs and delay time. Furthermore, time for routing an alternative path was reduced in proposal algorithm, in case that a network was damaged in existing optimal path algorithm, Dijkstra algorithm, and the proposal algorithm was designed to route an alternative path faster than Dijkstra algorithm, as it has a 2nd shortest path in cells of the damaged network. The study showed that the proposal algorithm can support routing of alternative path, if Dijkstra algorithm is damaged in a network.

Key Words : Genetic algorithm, Mobile agent, Route search, Networks

1. 서 론

인터넷 프로토콜의 지속적인 발전과 확장으로 인터넷을 이용한 서비스와 콘텐츠는 과거의 일반적인 데이터만을 주고받는 응용과 달리 비디오 및 오디오와 같은 실시간 멀티미디어 데이터의 교환으로 확대되었으며 동시에 전송률 보장 및 안정화 요구가 더욱 높아지고 있다. 이러한 멀티미디어 콘텐츠 응용은 재전송 요청과 수신을 반복하는 일반 데이터 트래픽과 달리 전송 지연에 민감한 특성을 가지며, End-To-End 형태로 구성되어 있는 현재 인터넷 환경에서는 전송되는 전송 스트림의 QoS를 보장하기 위한 여러 연구들이 진행되고 있다[1, 2].

인터넷의 경우 신뢰성이 보장된 최단경로 알고리즘으로 Dijkstra의 알고리즘을 주로 사용해 왔다. 그러나 네트워크의 구성노드의 수가 많아지거나 복잡한 트리구조를 가지게 되는 멀티미디어 스트리밍과 같은 전송에서는 높은 계산시간의 복잡도에 비례하여 네트워크 내의 전송 지연이 증가하기 때문에 수신 노드의 증가 및 감소와 전송량의 변화, 그리고 네트워크 전송 트래픽의 변화 등과 같은 문제에 능동적으로 대응하기 어려운 실정이다. 이와 같은 문제의 해결을 위해 신경망을 모방한 네트워크의 구성과 유전 알고리즘을 통한 네트워크의 최단경로 구성과 같은 또 다른 해결책이 대두되고 있으며, 이에 관련된 연구가 활발히 진행되고 있다[3, 4].

본 논문에서는 먼저, 기존 최단 거리 알고리즘인 Dijkstra 알고리즘을 사용하여 방향성을 결정하고 이동 에이전트 환경에서 셀 단위의 유전 알고리즘을 적용하여 Dijkstra 알고리즘에 수렴정도를 측정하였으며, 기존 유전 알고리즘과의 성능 비교를 통하여 제안한 알고리즘의 지연시간이나 비용(Cost)값에 대해 비교 분석하였다.

† Corresponding Author : Dept. of National Defense Intelligence Engineering, Sangmyung University, Korea
E-mail : cjseo@smu.ac.kr
* Dept. of Automotive Software, YoungDong University, Korea
접수일자 : 2015년 8월 3일
수정일자 : 2015년 11월 4일
최종완료 : 2015년 11월 10일

2. 경로 탐색 알고리즘

2.1 기존 경로 탐색 알고리즘

2.1.1 Dijkstra 알고리즘

Dijkstra의 최단경로 알고리즘은 한 정점에서 다른 모든 정점으로의 최단경로를 구하는 알고리즘으로, 최단경로와 최단거리를 모두 얻을 수 있다. OSPF(Open Shortest Path First) 라우팅 프로토콜은 Dijkstra 알고리즘을 사용하여 네트워크를 경유하는 최적화된 경로를 계산하며, 대략적인 계산 방법은 다음과 같다.

① 시작 라우터에서 가장 인접한 라우터(네이버 라우터)를 찾고, 그 라우터까지의 경로를 최단경로로 설정한다.
② 현재 최단경로 상의 라우터는 2개(자기 자신과 지금 찾은 라우터)가 된다. 최단경로가 알려진 라우터들의 집합을 S라 하면, 집합 S에 포함되지 않은 라우터 중에서 시작 라우터로부터 가장 가까운 라우터를 찾는다. 이 새로운 라우터는 집합 S에 바로 이웃한 라우터들 중 하나이고 그 라우터까지의 경로는 최단경로이며, 그 라우터를 집합 S에 포함시킨다.
③ 같은 영역 내에 새로운 라우터가 없을 때까지, 즉 모든 라우터가 집합 S에 포함될 때까지 ①에서 ③까지의 과정

을 반복한다.

링크 상태 라우팅 프로토콜은 Dijkstra의 알고리즘을 사용하여 네트워크를 경유하는 최적화된 경로를 계산한다[5, 6]. 네트워크의 각 링크마다 코스트를 할당하고, 트리 구조의 루트에 특수한 노드를 두고, 각 수신지로 가는 코스트를 합하여 트리의 분기를 계산한다. OSPF(Open Shortest Path First)의 경우 설정된 대역폭을 근거로 인터페이스 비용(Cost)을 계산하며 OSPF의 코스트를 각 인터페이스에서 직접 설정할 수 있으며, 이 값은 설정된 대역폭을 바탕으로 계산되는 코스트 값에 우선한다.

2.1.2 Munetomo 알고리즘

최근에 유전 알고리즘을 이용하여 최단 경로 문제를 접근한 Munetomo는 가변길이의 염색체 구조를 설계하고, 그에 따른 유전자 부호화 기법을 설계하여 주어진 출발지-목적지 노드 사이에서 최단 경로를 찾는 방법을 제안하였다. 가변길이 염색체는 출발지 노드부터 목적지 노드에 이르는 일련의 노드-열(Node-sequence)을 유전자의 대립형질에 순차적으로 할당함으로써 부호화된다[7].

한편, 교배를 수행하기 위한 교차점은 동일한 대립형질(동일한 노드를 의미)을 갖는 유전자가 염색체 상에서 동일한 위치(Locus)에 있는(경로 상에서 출발지 노드로부터 동일한 홉의 수만큼 떨어진 경우를 의미함) 경우에만 형성된다. 따라서 Munetomo가 제안한 교배 기법은 유전자의 위치에 절대적인 의존성을 갖게 되어서 교차점의 개수 및 교배가 가능한 염색체의 조합에 대한 경우의 수를 감소시키기 때문에, 해-표면을 효과적으로 탐색할 수 없게 되어 발견해의 최적성이 저하되는 문제점이 있다. 즉, 각 염색체가 탐색하는 경로들 중에서 동일 노드가 출발지에서부터 동일한 홉의 수만큼 떨어진 거리에 위치할 때만이 부분 염색체(부분 경로)를 서로 교환할 수 있기 때문에 해-표면의 탐색 능력이 열악해진 것이다. 또한, 돌연변이 동작에 있어서 염색체(경로) 상의 한 유전자(노드)를 무작위로 선택한 후에 이와 연결된 노드들 중에 한 노드를 역시 무작위로 선택하고 이를 “돌연변이 노드(Mutation node)”이라 한다. 그 후, 출발지 노드에서 돌연변이 노드에 이르는 최단 경로를 Dijkstra 알고리즘을 이용하여 찾고, 돌연변이 노드에서 목적지 노드에 이르는 최단 경로도 동일한 알고리즘에 의해 발견하는 절차에 의해 돌연변이 염색체를 생성한다. 하지만, 계산 시간의 복잡성이 높은 Dijkstra 알고리즘을 이용하고 있으며, 동시에 돌연변이 동작의 가장 핵심적인 역할인 개체군의 유전적 다양성 유지 역할을 수행하지 못하는 문제점이 있다[8, 9].

2.2 제안한 경로 탐색 알고리즘

유전 알고리즘을 이용한 이동 에이전트 경로 탐색기법은 최단거리 알고리즘의 방향성을 이용하여 라우팅 시 최적 경로를 탐색하는 알고리즘이다. 클라이언트로부터 요청된 정보를 전달하기 위해 서버쪽에서는 클라이언트로 최단 경로를 탐색하기 위해 에이전트 컨트롤러를 이용하여 경로에 위치한 라우터들을 검색하고 그들을 통해 서로 간의 최단거리를 검색하도록 하였다. 그리고 노드 장애시 클라이언트 쪽의 ″Path Modification″을 통해 장애노드를 포함하고 있는 셀에서 재설정 작업을 시행하도록 하였다.

서버는 그림 1과 같이 서비스와 그에 따른 에이전트 컨트롤 그리고 각 셀에서 구한 엘리트 집단, 즉 최적 경로들을 저장하는 “Cell Reiterate Index”로 구성된다. 클라이언트는 서버와 같이 각 셀에서 구한 최적 경로들을 저장하는 “Cell Reiterate Index”와 별도로 네트워크 안에서 장애 시 복구를 위한 “Path Modification” 모듈로 구성되어 있다.

네트워크 내에서 일정 변심거리에 의해 클러스터형식의 셀을 증식시켜 그 안에서 최저비용을 지불하는 경로를 선택하는데 그림 1의 ″Internet Roue′s″ 안에 보이는 C_1, C_2, C_3, C_4,…, C_n, 들이 제안한 알고리즘에 의해 생성된 셀이며 Cs_1, Cs_2, Cs_3, Cs_4,…, Cs_n는 각 셀 간의 중복된 라우터들의 그룹이다.

2.3 제안한 경로 탐색 알고리즘 수행 과정

2.3.1 동작 모델

본 논문에서 제안한 알고리즘의 기본 동작은 에이전트에 의해 초기화를 진행한 후 셀을 증식하며 각 셀 내의 에이전트들을 통해 셀 내의 유전알고리즘을 이용하여 최적경로를 구하고 인접 셀과 최적경로 비교를 통해 서로 간의 우열을 가리는 과정을 거쳐 목적지까지 도달하는 과정으로 이루어진다. 그림 1은 이러한 과정을 나타내며 그림 2는 그에 대한 Pseudo code의 일부이다.

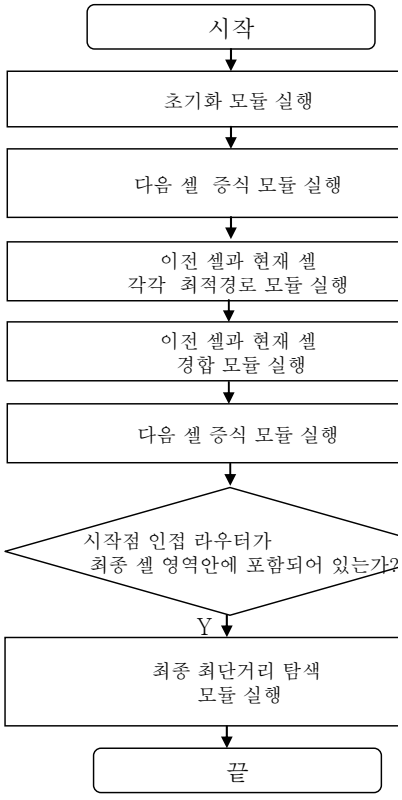


그림 1 제안한 경로 탐색 알고리즘의 순서도

Fig. 1 Flowchart of the proposed algorithm

```
Find_GASPF(){
    While  not (Start_point && Cn+1.) do
     initialize(GA_Cell);
      Creation(GA_Cell_First);
      Creation(GA_Cell_Second);
      If GA_Code(GA_Cell_first)
       < GA_Code(GA_Cell_second) then
        Cell_Reiterate_Index.add =
         GA_Code(GA_Cell_first)
           Else
Cell_Reiterate_Index.add =GA_Code(GA_Cell_second)
    End If
    End While }
```

그림 2 경로 탐색 알고리즘의 Pseudo Code
Fig. 2 Pseudo code of pathway search algorithm

2.3.2 초기화 이전 선행 절차

유전 알고리즘을 이용한 최적경로 탐색 알고리즘은 다음과 같은 기존 라우팅 알고리즘을 통해 목적지에 도달하기까지의 셀 생성의 방향성을 교정하게 된다. 즉 새롭게 생성되는 셀의 경우에 그 셀의 중앙부에 위치하는 에이전트 노드는 자신이 생성되기 이전의 에이전트 노드에게 자신이 생성해야 할 셀의 범위 및 위치를 명령받게 되는데, 에이전트 자신이 목적지의 방위를 알아낼 수 있는 방법이 없으므로 방향성의 교정을 위해서는 에이전트들이 참조할 수 있는 로드맵이 구성되어야 한다. 따라서 기존 Dijkstra 알고리즘을 이용한 OSPF 프로토콜에 의해 구성되어 있는 테이블을 참조하여 경로를 설정하여 셀을 증식하도록 한다.

만약 목적지 방향성을 위한 초기화 생성 과정을 생략한다면 새로 생성되는 셀의 에이전트 노드들은 목적지가 어디인지 감지할 수 없고 예상치 못한 곳으로 향하여 셀을 형성할 수 있는 문제가 발생하게 되며, 그로 인해 필요 없는 연산을 수행할 경우 및 그에 따른 위험에 노출되게 된다. 이와 같은 문제를 해결하기 위한 방편으로 제안한 최적경로 탐색 알고리즘의 초기 실행 단계에서는 다음의 기존 라우팅 알고리즘들을 이용해 시작 노드와 종단 노드 간에 경로 설정을 선행한다. 그 후 새로이 생성되는 셀의 에이전트들은 앞서 수행된 라우팅 경로의 위치를 참조해 예상치 않은 방향으로 셀의 위치를 잡는 오류를 미리 방지한다. 그림 3은 제안한 알고리즘에서의 선행된 초기 기존 Dijkstra 알고리즘 적용 경로를 도식화하여 나타낸 것이다.

2.3.3 초기화 과정

초기화 모듈 부분은 목적지점으로부터 가장 인접한 라우터를 찾는 과정부터 시작한다. 먼저 제안한 알고리즘의 초기 설정으로서 셀에서 셀의 경계선을 생성하기 위한 실수 상수 ε을 지정한다. ε는 상수이나 처음 구성시 네트워크 관리자에 의하여 결정된다. 이후 경계선 상수 ε로부터 각 셀들의 중복 영역을 설정을 위한 셀 블록의 직접 관할 구역을 배정하기 위한 식을 생성한다. 이 때 초기화 모듈 부분에서 형성되는 셀의 에이전트 간 거리는 삼각형으로 이루어진다. 제안한 알고리즘에서는 정삼각형만을 고려하여 실험 환경으로 적용하였다.

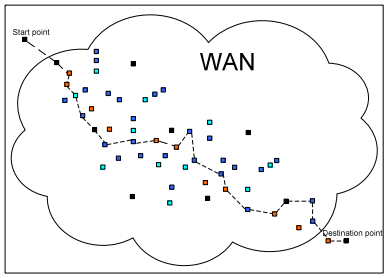


그림 3 라우팅 알고리즘 적용 경로
Fig. 3 Routing algorithm pathway

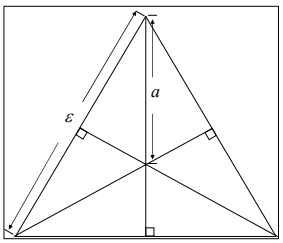


그림 4 삼각형의 무게중심
Fig. 4 Centroid of triangle

그림 4는 제안한 알고리즘에서 적용한 정삼각형의 무게중심을 결정하는 그림이며 이에 대한 삼각형의 꼭지점에서 중심까지의 거리 α의 값을 구하면 식(1)은 다음과 같다.

$$\alpha = \sqrt{\frac{\varepsilon^2}{3}} \tag{1}$$

ε는 셀의 경계선을 생성하는 값이다. 셀의 경계선을 찾는 이유는 다음 셀의 중심을 찾을 때 위치를 잡기 위해서이다. α는 셀의 경계선 외에 셀이 직접 관할하고 셀 간의 라우터들의 중복성을 적절히 유지하기 위한 값으로서 사용된다.

그림 5는 셀 구역내의 에이전트 관할 영역을 α 값을 이용하여 설정하는 과정을 보여준다.

2.3.4 셀 간 우성 평가 및 경로 설정

그림 6은 최적해 평가 함수를 통해 C_1과 C_2 셀에서 각각 구해진 최적해를 A_0에이전트의 관점에서 비교하게 되며, C_1과 C_2셀 중 우성으로 판단된 최적해 경로를 따라 Cs_3 혹은 Cs_4셀의 경계까지 진행하게 된다. 이러한 절차를 통해 각 셀 안에서의 최적해를 맹목적으로 따르지 않고 셀 구간별로 경합을 통해 최적해를 찾을 확률을 높이기 위한 절차이다. A_0에이전트 까지 도달된 최적해 경로를 뒤이어 빗금으로 표시된 영역의 C_1과 C_2 셀 내에서는 각각 유전 알고리즘에 의한 염색체 교배 과정을 통해 각 셀의 최적해 경로를 산출하고 그 결과를 A_0에이전트에게 통보하게 된다. 통보된 결과를 받은 A_0에이전트는 최적해 평가함수를 통해 두 후보 경로에 대한 평가를 진행하고 그 결과를 바탕으로 후보군 중

우성인 경로를 선정하며, 그 경로를 통해 자신의 셀(A_0) 까지 진행 되었던 전체 경로의 흐름을 우성으로 선발된 셀의 A_1에이전트에게 통보한다. A_1에이전트는 자신의 셀 구간의 최적해 경로를 $Cs_2 \rightarrow Cs_4$ 까지 적용한다. 이러한 과정을 통하여 각 셀 간의 최적해 값에 대하여 한번 더 경합을 통해 성능이 우수한 최적해를 구할 수 있고, 이와 같은 과정은 목적지까지 반복 적용된다.

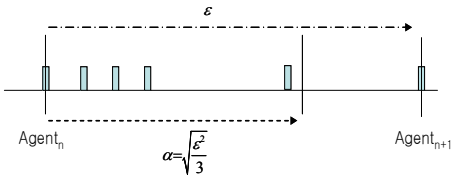


그림 5 셀 구역 내의 에이전트의 영역
Fig. 5 Agent area of cell region

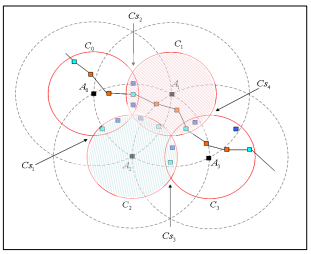


그림 6 셀 간 우성 평가 범위 및 경로 설정
Fig. 6 inter cell range and pathway

3. 실험 및 성능 분석

3.1 실험에 필요한 셀의 크기 결정

본 논문에서는 300개의 라우터를 갖는 하이브리드 토폴로지의 환경의 시뮬레이션 모델에 적합한 셀의 크기를 결정하여야 한다. 제안한 알고리즘에서는 4개 이상의 셀이 형성되어야 실험이 가능하며, 이를 본 실험에 사용된 토폴로지에 적용하면 10홉 이하의 홉 수가 형성된다. 11홉 이상의 홉 수에서는 본 실험에서 요구된 4개 이상의 셀이 형성되지 않기 때문에 본 실험환경에 적합하지 않다. 또한, 4홉 이하 경우는 Control 셀이 셀 간의 중첩부분이 생기지 않으므로 Control 셀이 형성 되지 않으므로 실험 환경을 만족시킬 수 없다. 이에 본 실험을 만족하기 위해서는 홉 수가 5홉에서 10홉 사이에서 셀의 크기를 결정하여야 한다. 실험에서 연산시간이 적을수록 신뢰도가 높아지므로 적용한 셀의 크기는 연산시간이 가장 작은 5홉으로 결정하였다.

3.2 셀 간 우성 평가 및 경로 설정 실험

A_1과 A_2 간의 최적 경로 탐색하는 과정을 나타내는 그래프가 그림 8의 (a)이다. A_1과 A_2 각 각 최적경로를 탐지하는데 걸리는 지연시간으로 앞서 논의 한 최단거리 판단 과정의 최적해 탐지 함수로 2초단에서 A_2 셀의 최단경로가 우성으로 결정된다. 그에 대한 영역이 그림 7이며 그림 8의 (a)(b)(c)는 위 과정의 3차례 시행에 대한 그래프이다.

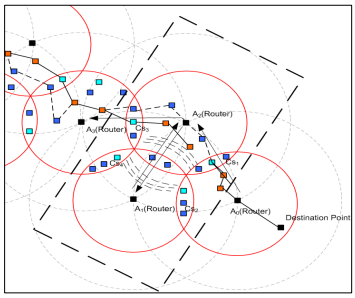
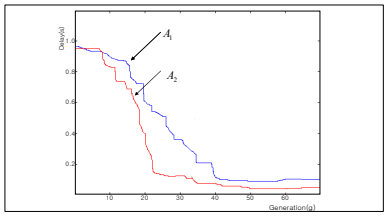
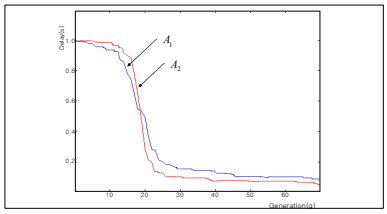


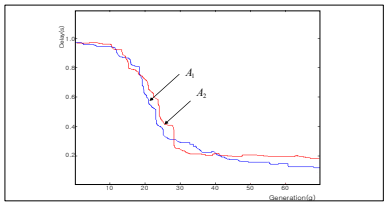
그림 7 셀 범위 지정
Fig. 7 Cell range determination



(a) Run 1



(b) Run 2



(c) Run 3

그림 8 에이전트 구역 별 연산 지연
Fig. 8 Operation delay in agent region

표 1는 2-블럭에서 A_1과 A_2가 속한 그룹에서 각각의 에이전트가 최단 경로 알고리즘 시행시 지연시간을 나타낸다. 표 1에서 Run 1 의 경우에는 A_1보다 A_2가 우월한 것을 보여주며 표 2에서 Run 2의 경우에도 A_2가 우월하며 표 3에

표 1 에이전트 구역 별 연산 지연(Run 1)
Table 1 Operation delay in agent region(Run 1)

Generation	A_1	A_2
10	0.912	0.812
20	0.614	0.401
30	0.381	0.173
40	0.121	0.057
50	0.113	0.039
60	0.102	0.038

서 Run 3의 경우는 약 25번 이상을 시행 했을 경우 A_2가 우월하다가 40 이상에서 다시 바뀜을 확인할 수 있다.

표 2 에이전트 구역 별 연산 지연(Run 2)
Table 2 Operation delay in agent region(Run 2)

Generation	A_1	A_2
10	0.952	0.998
20	0.521	0.351
30	0.072	0.156
40	0.067	0.078
50	0.056	0.041
60	0.051	0.037

표 3 에이전트 구역 별 연산 지연(Run 3)
Table 3 Operation delay in agent region(Run 3)

Generation	A_1	A_2
10	0.956	0.967
20	0.631	0.713
30	0.273	0.231
40	0.216	0.212
50	0.176	0.211
60	0.152	0.199

3.3 제안한 알고리즘의 성능 비교 분석

3.3.1 연산 지연 시간 비교

본 논문의 실험에서는 Dijkstra 알고리즘에 의해 결정된 최적해와 제안 알고리즘이 도출하는 해를 비교하여 최적성에 대한 성능 비교와 각 알고리즘에 대한 수렴 속도 성능에 대한 비교도 함께 수행하였다. 그리고 제안하는 유전 알고리즘을 이용한 라우팅 알고리즘의 성능을 네트워크 토폴로지를 구성하여 Munetomo가 제안한 유전 알고리즘과 성능을 비교 분석하였다. 제안한 알고리즘의 성능 분석 결과는 그림 9와 같다.

제안하는 유전 알고리즘과 그 비교 대상인 Munetomo의 유전 알고리즘 간 성능 분석의 척도로 Dijkstra 알고리즘을 선정하였다. 그림에서 보는바와 같이 40세대까지는 제안한 알고리즘의 경로 설정 지연이 비교적 큰 값을 가지고 있으며, Munetomo 알고리즘에 비해 초반 지연이 큰 것으로 나타난다. 하지만, 50세대가 지나면서 Munetomo 알고리즘에 비해 지연이 큰 폭으로 감소되며, 결과적으로 성능의 척도로 자리하고 있는 Dijkstra 알고리즘의 라우팅 지연에 근접하게 수렴함을 볼 수 있다. 80세대 이후에서는 기준 경로에 수렴함을 확인 할 수 있다. 비교대상인 Munetomo 알고리즘보다 초반 지연이 큰 이유는 제안하는 알고리즘이 초반 라우팅 경로 설정에 필요한 에이전트군의 선출에 의한 지연과 각 에이전트가 관리하는 노드들의 상태를 파악하는 전송 지연의 결과이며, 이 과정이 종료된 후로는 Munetomo 알고리즘에 비해 적은 지연을 가지는 것을 확인 할 수 있다. 그림 10과 표 4는 라우팅 알고리즘의 비교를 나타낸다.

3.3.2 경로 처리 비용(cost) 비교

경로 처리 비용(cost) 비교를 위해 각 노드에 가중치를 110부터 160까지 각각 부여하여 초기 값을 설정하였다. 기존 Dijkstra 알고리즘에서 전체 경로 비용으로써 가중치의 합이 2,282이 나왔으며 따라서 제안한 알고리즘이 Dijkstra 알고리즘의 경로 처리 비용 2,282에 수렴하도록 최적해 값을 부여하여 설계하였다. Munetomo 알고리즘의 경우 전체 경로 비용은 2,341가 나왔으며 제안한 알고리즘은 전체 경로 비용이 2,297가 나왔으며 99.34%의 최적성을 가지고 있으며 Munetomo 알고리즘의 경로 비용이 97.47%의 최적성을 가지고 있음을 표 5에서 확인 할 수 있다. 따라서 제안한 알고리즘의 수렴 성능에 대한 값은 Munetomo 알고리즘보다 높게 나타났으며 수렴 성능이 Munetomo 알고리즘보다 강건함을 나타낸다.

표 4 라우팅 알고리즘의 성능 비교
Table 4 Comparison with the performance of routing algorithm

Generation (g)	제안한 알고리즘 (delay time)	Dijkstra 알고리즘 (delay time)
20	3.225	1.551
40	2.897	1.544
60	2.225	1.523
80	1.525	1.533

표 5 라우팅 알고리즘의 최적성 비교
Table 5 comparison with optimality of routing algorithm

일련번호	Dijkstra 알고리즘		Munetomo 알고리즘		제안한 알고리즘	
	이동경로	비용	이동경로	비용	이동경로	비용
1	45	111	45	111	45	111
2	46	126	46	126	40	134
3	47	113	47	113	48	114
4	53	117	53	117	49	128
5	52	126	64	144	50	112
6	65	117	153	136	51	124
7	102	123	102	116	67	111
8	100	113	100	113	100	136
9	177	126	177	126	103	115
10	179	127	179	127	104	126
11	107	113	119	135	107	126
12	108	124	118	132	109	123
13	117	116	117	115	117	114
14	233	117	233	117	125	134
15	236	127	236	127	244	114
16	243	133	243	133	243	122
17	287	114	287	114	287	114
18	294	126	294	126	294	126
19	D	113	D	113	D	113
전체경로 비용	2282		2341(97.47%)		2297(99.34%)	

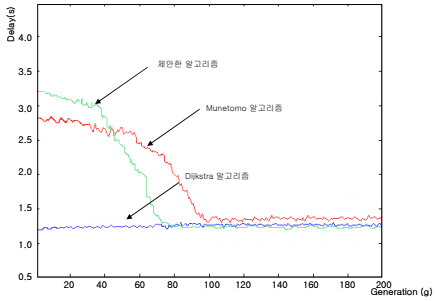


그림 9 제안한 알고리즘과 기존 알고리즘 비교

Fig. 9 Comparison between proposed and previous algorithm

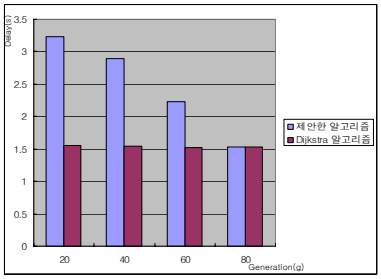


그림 10 라우팅 알고리즘 별 비교

Fig. 10 Comparison of each routing algorithms

4. 결 론

본 논문에서는 기존 유전 알고리즘과 이동에이전트에 대해 분석하고 기존 유전 알고리즘보다 향상된 유전 알고리즘을 이용하여 대체 경로 탐색 기능을 향상 시키는 알고리즘을 제안하였다. 또한, 제안된 경로탐색 알고리즘은 유전 알고리즘의 연산을 셀 단위로 처리하므로 유전 연산 처리 시간을 분산시켜 연산 시간을 단축하는 효과도 얻었다.

본 논문에서 제안한 알고리즘의 수행 절차를 살펴보면 첫 번째 셀을 만들고 그 위치에서 두 번째 셀과 세 번째 그리고 네 번째 셀을 차례로 만들며 그 포인트에 에이전트를 복제 이전시키고 에이전트로 하여금 각 셀마다 최단 경로를 구하도록 하고 그 후 경쟁하여 최적의 요소를 찾도록 하였다.

제안한 알고리즘이 설정과정에서는 셀 단위 유전 연산이 이루어짐으로 Munetomo 유전 알고리즘보다 50세대가 지나면서 지연시간이 큰 폭으로 감소함을 보였다. 이로써 실제 네트워크 환경에서 TCP 세션 형성을 통한 전송 시 초반 경로 탐색에서 기준시간동안 비교적 긴 초반 지연을 갖지만 전송측면에서의 효율은 우월할 것으로 판단된다.

제안한 알고리즘은 시뮬레이션 실험을 통하여 기존 유전 알고리즘인 Munetomo 알고리즘과의 성능 분석에 대한 결과는 전체 경로 비용에서는 1.87%정도 우위에 있는 것으로 나타났다. 이는 기존 유전 알고리즘인 Munetomo 알고리즘보다 본 논문에서 제안한 알고리즘이 우수하다는 것으로 판단되며 최적 경로 알고리즘이 Dijkstra 알고리즘과 비교하여 볼 때 지연시간에서는 초반 연산시간은 유전 연산 과정에서 지연시간이 길어지나 8초가 지난 후부터는 Dijkstra 알고리즘에 수렴하는 것을 확인할 수 있으며, 제안한 알고리즘이 Dijkstra 알고리즘에 전체 경로 비용의 99.34% 도달하여 비교적 최적 경로의 값에 수렴하는 것을 알 수 있다.

References

[1] Filipe Araújo, Bernardete Ribeiro, Luís Rodrigues, "A neural network for shortest path computation", IEEE Transactions on Neural Networks, Volume 12, Issue 5, pp. 1067 - 1073, Sep. 2001.

[2] W. Stalling, HIGH-SPEED NETWORKS TCP/IP AND ATM DESIGN PRINCIPLES, Prentice Hall, 2000.

[3] J. H. Holland, Adaptation in Natural and Artificial Systems, The MIT press, 1992.

[4] N. M. Karnik and A. R. Tripathi, "Design Issues in Mobile-Agent Programming Systems", IEEE Concurrency, pp.52-61, 1998.

[5] B. Liu, S. Choo, S. Lok, S. Leong, S. Lee, F. Poon, H. Tan, "Integrating case-based reasoning, knowledge-based approach and Dijkstra algorithm for route finding", Artificial Intelligence for Applications, 1994, Proceedings of the Tenth Conference, pp. 149 - 155, 1994.

[6] C. Xi, F. Qi, L. Wei, "A New Shortest Path Algorithm based on Heuristic Strategy", Proceedings of the 6th World Congress on Intelligent Control and Automation, 2006.

[7] M. Munemoto, Y. Takai, and Y. Sato, "A migration scheme for the genetic adaptive routing algorithm", in Proc. IEEE Int. Conf. Systems, Man, and Cybernetics, pp. 2774 - 2779, 1998.

[8] J. Inagaki, M. Haseyama, and H. Kitajima, "A genetic algorithm for determining multiple routes and its applications", in Proc. IEEE Int. Symp. Circuits and Systems, pp. 137 - 140, 1999.

[9] C. Ahn, R. S. Ramakrishna, "A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations", IEEE Transactions on Evolutionary Computation, Vol. 6, No. 6, December 2002.

저 자 소 개

서 창 진 (徐 彰 辰)

1999년 부산대 대학원 멀티미디어 석사과정 졸업 2003년 동대학원 박사과정 졸업(공학박사), 현재 상명대 국방정보공학과 조교수,

E-mail : cjseo@smu.ac.kr

지 홍 일 (池 弘 壹)

2002년 충북대 대학원 컴퓨터공학과 석사과정 졸업 2007년 동대학원 박사과정 졸업(공학박사), 현재 영동대 자동차소프트웨어학과 조교수

E-mail : jihi61@yd.ac.kr

Copyright ⓒ The Korean Institute of Electrical Engineers

This is an Open-Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.