차량 간 통신을 이용한 유전자 알고리즘 기반 동적 차량 경로 탐색 알고리즘

오병화 ○ 배준성 양지훈 낭종호
서강대학교 컴퓨터공학과
mr-five@hanmail.net, hanuri13@sogang.ac.kr, yangjh@sogang.ac.kr, jhnang@sogang.ac.kr

A Genetic Algorithm-based Dynamic Vehicle Route Search Algorithm using Car-to-Car Communication

Byonghwa Oh ○ Junsung Bae Jihoon Yang Jongho Nang
Department of Computer Science and Engineering, Sogang University

요 약

차량 경로 탐색 문제를 해결하는 것은 차량 이동에 지체되는 시간 감소를 통한 개인적인 혜택 뿐만 아니라 교통 체증으로 인한 사고율 감소 및 사회적 비용 등의 감소를 가져온다. 본 논문에서는 기존에 사용되는 경로 탐색 알고리즘과 다른, 그러나 통신 가능한 네비게이션 시스템에 탑재가 용이한 새로운 알고리즘을 제시하고 이 알고리즘의 효용성을 시뮬레이션과 실험을 통해서 검증하였다. 제안된 알고리즘은 일반적인 경로 탐색 문제에서뿐만 아니라, 하드웨어의 고장 및 도로 구간의 교통 사고 발생 등의 경우에 대해서도 적응할 수 있음을 실험을 통해서 확인하였다.

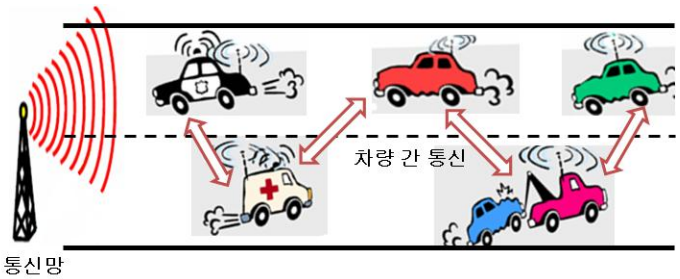
1. 서론

차량용 네비게이션은 그래픽 인터페이스를 이용하여 운전자에게 출발지부터 목적지까지의 최단 시간 및 거리 경로에 대한 정보를 제공함으로써 빠르고 안전한 운전을 할 수 있도록 도와줄 수 있기 때문에 최근 그 사용이 급격히 늘어나고 있다.

하지만, 기존의 네비게이션은 지도에서의 정적인 거리 및 운행 시간 정보를 이용하여 최적 경로를 계산하여 운전자에게 제공하기 때문에 교통량에 따라 동적으로 변하는 현 시점에서의 최적 경로를 제공하지 못하는 문제점을 가지고 있다. 이런 문제점을 해결하기 위하여 중앙에서 수집되어 방송된 교통 정보(예: DMB 방송에 포함된 TPEG 정보)를 반영하여 최적 경로를 동적으로 계산하여 제공하는 방법이 등장하였다.

그러나 중앙 집중적인 방송에 의하여 교통 정보를 얻는 방식은 수집 및 방송에 많은 시간이 소요되기 때문에 실시간 교통 정보라 할 수 없다. 그나마 수집되어 송출되는 정보 또한 큰 길 위주로 제한적이다. 또한 모든 네비게이션이 같은 정보를 이용하여 최적 경로를 계산하기 때문에 한적한 도로로 많은 차량이 집중되어 교통 체증을 더 유발 할 수 있는 문제점도 가지고 있다.

이러한 문제점을 해결하기 위하여서는 동적으로 변하는 교통 상황을 실시간으로 반영하여 빠른 시간에 최적 경로를 계산하고 실제 현재의 거리 상황을 이용하여 안내 정보를 제공할 수 있는 차량용 네비게이션의 개발이 요구된다. 이를 위해, 자신이 지나온 경로와 관련된 정보(거리 GPS 정보, 평균 속도 등)를 수집하여 주변 네비게이션과 서로 교환함으로써 실시간 교통 정보를 취득하며, 전달되는 교통 정보를 종합하여 최적 경로를 동적으로 빠르게 계산할 수 있는 종합적인 알고리즘을 개발하고, 실험을 통하여 성능 평가를 수행하였다.



[그림 1] 차량 간 통신을 이용한 정보 교환

본 연구에서는 유전자 알고리즘(Genetic Algorithm)과 통신 가능한 다양한 기능들을 사용하여, 이 네비게이션 알고리즘을 탑재한 자동차들끼리 서로 자신이 지나 온 최근의 교통 상황 정보를 근처에 있는 자동차에 전해줌으로써 자신이 가진 최신의 교통 정보를 업데이트하고, 그 정보를 바탕으로 자신이 갈 길을 계속적으로 수정하여 제공하는 알고리즘을 제안한다. 그리고 이 방법의 실험적인 우수성을 시뮬레이션을 통한 실험 및 시각화 프로그램을 이용하여 검증한다.

2. 유전자 알고리즘을 사용한 대체 경로 탐색 알고리즘의 설계

2.1. 유전자 알고리즘 기반 경로 탐색 알고리즘

유전자 알고리즘은 생물 진화의 원리에서부터 착안된 알고리즘으로서, 확률적 탐색(Probabilistic Search)이나 학습 및 최적화를 위한 기법 중의 하나이다. 각각의 후보 해(Candidate Solution)를 표현하는 개체들의 집합을 유지하면서, 개체들에 대해 선택, 교차, 돌연변이의 세 개의 연산자(Genetic Operators)를 사용하여 진화를 수행한다. 개체군의 크기가 항상 유지되는 가운데 우수한 개체는 유지되고 열등한 개체는 재생산된 다른 개체들로 교체되면서 우수한 해를 찾는다. 이 알고리즘은 여러 문제에 있어서 비교적 우수한 해를 단시간에 찾을 수 있다고 알려져 있다.

네비게이션에 기본적으로 탑재되는 경로 탐색 알고리즘은, 유전자 알고리즘을 사용하여 좋은 결과를 보인 논문의 알고리즘을 차용하였다[1]. 이 알고리즘은 기존의 발견적(Heuristic) 길 찾기 알고리즘보다 계산 시간 및 해의 정확성 측면에서 우수하며 그래프 토폴로지에 덜 민감하기 때문에 적합하다고 판단하였다.

이 알고리즘에서 길 찾기는 다음의 순서대로 진행된다. 먼저, 염색체는 가변 길이로 구성된다. 그리고 염색체군(Population)은 시작 지점(Vertex)에서 임의로 간선(Edge)을 이어 가며 찾아낸 도착 지점까지의 경로들로 초기화된다. 만약 시작 지점에서 임의로 간선을 연결하던 중 이 경로가 이전에 찾아낸 임의 경로보다 너무 길어지거나 도착 지점까지 연결되는 경로가 나타나지 않을 때는 이를 폐기하고 새롭게 염색체를 생성한다.

적합도(Fitness) 함수는 다음과 같이 정의된다. 여기서 f_i는 염색체의 적합도 값을 나타내고, m_i는 i번째 염색체의 길이를 의미하며, $g_i(j)$는 i번째 염색체의 j번째 위치의 유전자를 나타내고, C는 두 지점 간의 링크 비용을 의미한다.

$$f_i = \left(\sum_{j=1}^{m_i-1} C_{g_i(j),g_i(j+1)} \right)^{-1}$$

교배(Crossover)와 돌연변이(Mutation)은 염색체의 완성된 경로에 대해 부분 경로를 서로 교체하거나 또는 임의의 경로로 대체함으로써 수행될 수 있다. 교배시에는 두 부모 염색체에서 같은 지점을 찾아서 그 지점 바로 뒤의 경로들을 교체하는 방법을 사용하고, 돌연변이는 염색체 상에서 부분 경로를 선택한 후 초기화와 동일한 방법으로 그 부분 경로를 다른 새 경로로 대체하는 방법을 사용한다.

이렇게 했을 때 순환 경로(Loop)가 발생할 수 있다. 교배와 돌연변이 연산 후 모든 염색체에 대해 염색체의 처음부터 마지막 지점까지 살펴보면서 한 번 나왔던 지점이 또 나왔을 경우 그 중간의 내용을 삭제하는 방법으로 빠르게 순환 경로를 제거할 수 있다. 이러한 연산들의 자세한 내용은 [1]에 제시되어 있다.

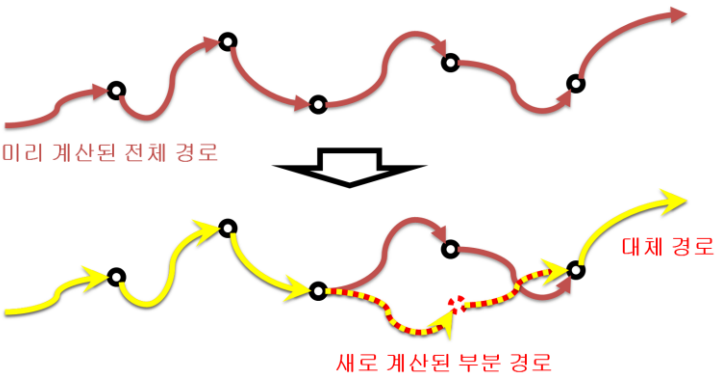
그러나, [1]에서는 일반적인 그래프 모두에 대해 우수한 성능을 보인다고 하였으나, 실험적으로는 비교적 작은 그래프(크기 100 이하)에서 우수함을 확인하였다. 왜냐하면 초기화 연산 및 돌연변이 연산에 사용되는 무작위 경로 탐색 부분의 계산 부담이 그래프가 커질수록 크게 증가하기 때문이다. 그러나 이는 작은 그래프에서는 문제가 되지 않는다. 그러므로 이후에 설명할 부분 대체 경로 탐색에서는 계산 시간의 문제 때문에 작은 부분 그래프에서의 경로 계산을 사용한다. 따라서 다른 알고리즘 대신 유전자 알고리즘 기반 경로 탐색 알고리즘을 사용하여 효율성을 높일 수 있다.

2.2. 대체 경로 탐색 알고리즘

2.2.1. 개요

대체 경로를 탐색하는 이유는 전체 경로의 탐색이 많은 계산을 야기하므로 현실적으로 네비게이션에서 수시로 수행하기 어렵기 때문이다. 그러나 대체 경로는 말 그대로 차선책으로 선택한 것이므로, 일단 시작점과 도착점 사이의 하나의 경로는 미리 주어져야 한다.

이는 임의로 지정할 수도 있고, 수행 속도가 짧은 알고리즘(예: A*)으로 구할 수도 있다. 또는 미리 네비게이션의 저장 장치에 각 시작점과 도착점의 쌍에 대응되는 최적 경로를 저장해 놓는 방법도 있다. 이 데이터는 오프라인 계산을 통해 전체 탐색 알고리즘을 사용하여 계산하거나 또는 네비게이션 제조사를 통해 제공받음으로써 확보한다. 본 연구에서는 네비게이션의 보조 저장소에 통계적으로 구한 평균 가중치(각 구간 사이의 통행 가능 시간)를 사용하여 전체 경로를 미리 계산한 정보가 저장되어 있다고 가정하였다.



[그림 2] 대체 경로 탐색 방법

이렇게 확보된 전체 경로에서, 시시각각 변하는 도로 정보를 반영한 후 유전자 알고리즘을 사용한 경로 탐색 알고리즘을 사용하여, 어떤 지점에서의 부분 그래프에서 최신 교통 데이터를 사용하여 찾아낸 부분 경로를 기존 경로의 부분 경로와 치환한다. 이렇게 하여 개선된 전체 경로를 얻을 수 있다. [그림 2]는 이러한 방법을 그림으로 나타낸 것이다.

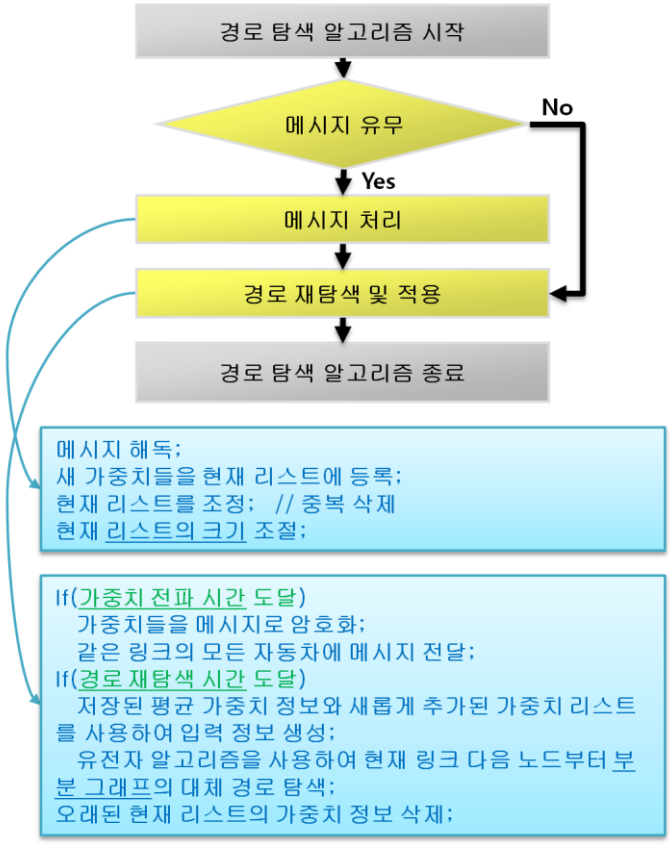
2.2.2. 알고리즘 설명

전체 도로를 중간 지점과 각각을 연결하는 간선으로 구성되는 그래프로 간주할 때, 그래프의 각 간선의 가중치는 이 거리를 자동차가 통과하는 데 걸리는 시간으로 정의할 수 있다. 이렇게 하면 풀고자 하는 문제는 시작점과 도착점 사이의 최소 시간을 구하는 문제가 된다. 건설교통부 지침[2]에 따라, 각각의 간선을 링크(Link), 간선 사이의 각각의 지점을 노드(Node)라고 지칭한다.

각 자동차에 장착된 네비게이션은 수시로 알고리즘을 실행한다. 먼저 다른 자동차의 네비게이션에서 어떤 링크의 통행 시간(가중치 정보)에 대한 정보가 전송되었는지 살펴본다. 있을 경우 현재 리스트(Current List)를 업데이트한다. 이 현재 리스트는 네비게이션의 특성상 제한된 저장 용량을 가진다고 가정한다. 유지되는 현재 리스트에 대해 만약 어떤 링크에 대해 더 최신의 정보가 수집된다면 그 정보로 기존 정보를 대체하고, 너무 오래 되어 현재 교통 상황을 더 이상 반영하지 못하는 정보는 삭제한다.

그 후 경로 재탐색 및 적용은 각 시간 주기마다 실행된다. 네비게이션은 자동차가 지금까지 지나온 링크 및 지나온 링크와 방향이 반대인 링크(즉, 어떤 길에서 상행을 타고 왔다면 하행의 링크)의 실제 교통류 속도와 통행 시간을 수집한다. 그리고 다음 링크로 건너갈 때마다 새로 만나는 자동차의 네비게이션에 그 정보를 전달한다. 이 정보는 같은 링크 상에 있는 자동차의 네비게이션에 전파(Propagate)되어 공유된다. 경로 재탐색 또한 정해진 시간 주기마다 실행되는데 이는 경로 재탐색의 계산 시간이 주기보다 길게 될 경우 서비스 품질에 문제가 생길 수 있기 때문이다. 주기는 계산 시간보다 조금 더 길게 할당한다. 그리고 경로 재탐색이 끝난 후에는 오랜 시간이 지나서 더 이상 의미가 없는 통행 시간 정보를 삭제한다.

[그림 3]에 이 모든 작업을 순서도와 유사 코드(Pseudo Code)로 나타내었다. 밑줄로 표시한 부분(리스트의 크기, 가중치 전파 시간, 경로 재탐색 시간, 부분 그래프)은 알고리즘에서 변경하여 적용할 수 있는 매개 변수들(Parameters)이다.



[그림 3] 대체 경로 탐색 순서도 및 유사 코드

3. 실험

3.1. 그래프 정보

본 연구에서 사용하는 도로 정보는 2.2.1절에서 설명한 건설교통부 지침에 따라 노드와 링크로 구분된 그래프에서 측정된 통계 정보를 사용한다. 2006년 9월 11일부터 2006년 12월 10일 사이의 서울시 링크의 10분 간격으로 측정된 속도 및 평균 링크 주행 시간 데이터를 이용하여, 그래프의 각 간선의 가중치를 결정하였다.

3.2. 시뮬레이션 환경

우선 실제 서울시 도로 교통 데이터를 바탕으로 하여 시가지를 구축하고 다수의 자동차를 생성하여 도로를 운행하게 함으로써 실제 도로 환경과 흡사한 실험 환경을 구축하였다. 그리고 개발된 프로그램의 변경 가능한 파라미터들을 설정할 수 있게 함으로써 이를 바꾸어 가면서 알고리즘을 테스트하여 구축된 시뮬레이션 환경과 알고리즘의 효용성을 검증하였다.

시가지를 구축하기 위해 먼저 3.1절의 정보를 사용하여 가상의 공간에 도로를 배치한 후, 도로의 모양에 따라 일반 도로, 3지 교차로, 4지 교차로 및 각

링크의 위치 관계를 추가로 생성하였다(위도 및 경도 데이터를 바탕으로 하여 각각의 링크의 각도 정보를 구해서 각도에 따라 어떤 링크가 어떤 링크의 좌회전으로 갈 수 있는지, 우회전으로 갈 수 있는지, 아니면 직진 도로인지 파악하는 작업이 추가적으로 필요). 그 후 필요한 부분에 신호등을 배치하고 적절한 주기를 주어서 신호에 따라 자동차들이 통제되거나 또는 통행할 수 있게 하였다.

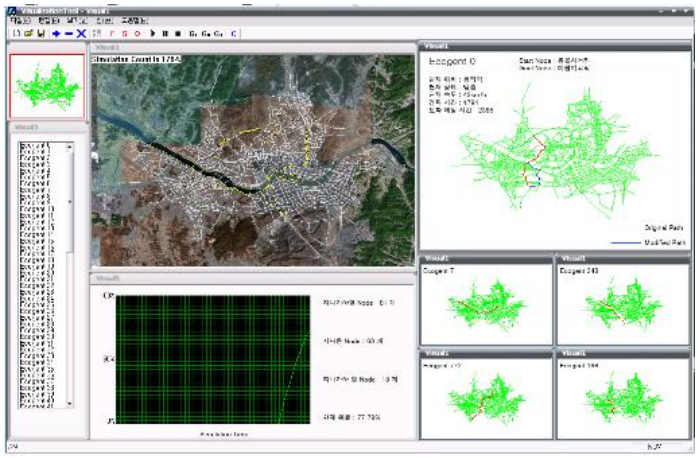
포화도(degree of saturation)는 "자동차의 수/한 링크 안에 들어갈 수 있는 최대 자동차의 수"로 표현된다. 자동차들이 한 링크 안에 많이 모일수록 포화도가 높아지고 링크 안에 있는 자동차들의 전체적인 속도는 줄어든다. 앞 차와의 간격 유지 및 교통 법규 준수, 기타 여러 가지 이유 때문에 속도가 줄어드는 것이다. 이를 교통공학에서는 통행 비용 함수식으로 모델링한다. 본 논문에서는 여러 연구에서 가장 널리 사용되는 미국공로국(Bureau of Public Road)의 BPR 함수[3]를 사용하여 교통 집중에 따라 어떻게 통행 시간 지체가 발생하는지 계산하여 실제로 적용하였다.

$$c_a(f_a) = t_a^0\left[1 + k\left(\frac{f_a}{u_a}\right)^{\beta}\right], \forall a \in L$$

위 식에서 f_a는 link a의 통행 흐름의 정도(자동차의 양), u_a는 link a의 용량(최대 들어갈 수 있는 자동차의 수), t_a^0는 평소 link a를 지나가는 데 걸리는 통행 시간, k와 β는 조절 가능한 매개 변수를 나타낸다.

k와 β는 국내 도로 교통 현실에 따라 적절한 값을 정해 주어야 하는데, 국토개발연구원에서 편찬된 문서에 의하면 모의실험과 경험에 근거하여 산출한 적절한 값은 각각 0.91과 3이다[4]. 이 값을 사용하여 모의 실험 환경을 구축하였다. 또한 모의 실험 을 통해 나온 결과값을 로그파일로 작성 후 실험 상황을 확인할 수 있는 시각화 프로그램을 제작하였다.

[그림 4]에서 보여지는 시각화 프로그램을 통하여 실시간과 같이 시뮬레이션을 수행함으로써 시간에 따라 변화되는 통과 자동차의 수와 교통상황을 쉽게 살펴볼 수 있다. 시각화 프로그램에서는 각 자동차들을 선택하면 미리 계산된 최적 운행 경로와 실시간으로 계산되는 운행경로가 각각 빨간색과 파란색으로 나타난다. 그리고 실시간으로 계산되는 운행경로에 영향을 미친 다른 자동차들의 경로 상황도 보여줄 수 있다. 아래에는 시간의 변화에 따라 도착하는 자동차의 수를 지속적으로 나타냄으로써 각 알고리즘에 따라 어떻게 성능 차이가 나는지 실시간으로 살펴볼 수 있게 설계하였다. 네트워크를 통해 여러 컴퓨터를 사용하여 다양한 결과에 대한 시각화를 동시에 수행할 수 있으며 결과를 한꺼번에 비교 분석할 수 있다.



[그림 4] 시각화 프로그램

3.3. 실험 계획

실험에 사용되는 비교군들은 길 찾기에 사용되는 알고리즘의 종류에 따라 크게 세 가지로 분류하였다.

첫 번째 알고리즘은 다익스트라(Dijkstra)의 최단 경로 알고리즘이다. 이 알고리즘은 계산 시간이 확보될 때 가장 짧은 길을 찾아낼 수 있으나 모든 노드에서 모든 노드로 가는 모든 경로를 다 탐색하기 때문에 계산 부담이 매우 큰 알고리즘이다. 기존 네비게이션은 현재 교통 정보 대신 맵 프로그램에 저장된 각 길의 정보를 사용해서 미리 계산하여 놓은 가장 짧은 길들의 집합에서 짧은 길을 찾아서 사용자에게 보여 주므로, 이와 같이 다익스트라의 알고리즘을 사용하여 미리 구해진 길을 그대로 따라가는 자동차들을 다수 생성하여 실제 교통 상황이 어떻게 변하는지 살펴보았다.

두 번째 알고리즘은 TPEG 유사 알고리즘이다. TPEG에서 유의해야 할 것은 현재 모든 길들(그러나 실제로 모든 길이 아닌 일부 큰 길들)의 정보가 5분마다 한 번씩 새로 업데이트된다는 것이다. 그러나 여러 사업자들을 거친 데이터가 부호화를 거쳐서 전송되는 복잡한 정보 전달 과정을 거치기 때문에, 이 모든 것을 고려하여 2분 정도 이전의 정보가 전파된다고 가정하였다. 역시 이 환경을 적용하여 자동차들이 5분마다 정보를 모두 새로 받게 하고 이 정보를 바탕으로 길을 바꾸어 찾아가게 하여 실제 교통 상황이 어떻게 변하는지 살펴보았다. 이 때는 다익스트라의 알고리즘을 사용하여 경로를 재탐색한다. 여기서, 다익스트라 알고리즘은 실제로 제한된 계산 자원을 가진 모바일 기기에서 수행하는 데 시간이 걸리기 때문에 알고리즘을 기기에서 수행한 평균값인 1분의 시간이 추가로 소요되게 하였다. 이렇게 되면 최악의 경우 8분 전의 정보를 사용하여 경로를 재탐색할 가능성이 존재한다. 또한 이렇게 예보하는 길은 실제로 큰 길들만 수행하므로 500개로 제한하였다.

세 번째 알고리즘은 본 연구에서 제시한 알고리즘이다. 2.2절에서 설명한 내용을 바탕으로 구현하고 이 상태에서 교통 상황이 어떻게 변하는지 살펴본다.

그리고 추가적으로 실험을 분할하였는데 이는 적응성(Adaptability)과 생존성(Survivability)을 알아보기 위해서이다. 특정 링크들에 사고를 발생시킴으로써 적응성을 파악하고, 또 본 연구의 알고리즘을 탑재한 자동차가 생태계 모방 길 찾기 기능을 사용할 수 없는 상황에서 다른 자동차들에는 영향이 없는지 알아보는 실험을 통해 생존성을 파악한다.

각 알고리즘의 우수성을 판별하기 위한 척도는 단위 시간에 목적지에 도착한 자동차의 수로 정하였다. 자동차들의 이동은 각각 정해진 어떤 위치에서 시작하여 어떤 위치에서 끝나게 하였고, 이동이 끝나면 다시 정해진 어떤 위치에서 새로 시작되도록 하였다. 이렇게 하면 계속하여 플랫폼 대수만큼의 트래픽을 유발시킬 수 있고 도착한 자동차의 대수를 비교하여 알고리즘의 우수성을 비교할 수 있다.

2.2.2절에서 제시한 매개 변수들에 대해, 리스트의 크기는 1000, 가중치 전파 시간은 15초, 경로 재탐색 주기는 60초, 부분 그래프의 크기는 중심점으로부터 5링크로 정하였다. 그리고 최신 정보를 유지하기 위해서 리스트의 크기를 조정할 때 30분 이전의 정보는 삭제하도록 하였다.

실험은 UNIX 운영체제 기반의 Sun Blade 서버 4대를 MPI 클러스터로 구성하여 진행하였다. DB와 파일을 통해 미리 정의된 포맷의 로그 데이터가 생성되게 하였고 이 로그 데이터를 시각화 프로그램에서 시각화하여 보여 줄 수 있도록 하였다.

3.4. 실험 결과

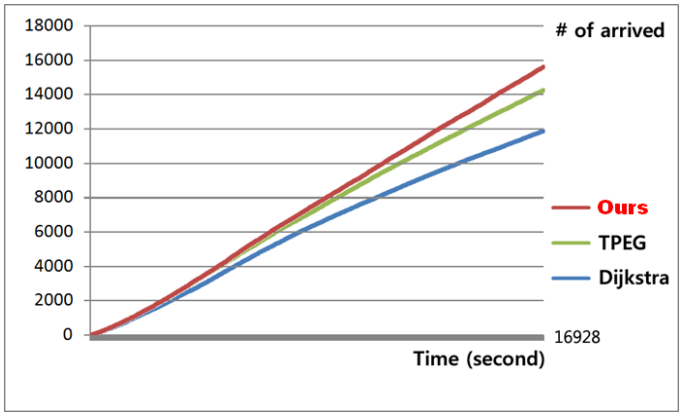
아래 [표 1]에서 자동차수는 시뮬레이터에서 항상 유지되는 자동차의 수를 의미한다.

[표 1] 알고리즘 및 상황 별 목적지에 정상적으로 도달한 차량 대수

상황	환경		알고리즘		
	자동차수	실험시간	Dijkstra	TPEG	Ours
보통	4900	10000	11866	14260	15606
생존성	10000	8000	16928	19163	21591
적응성	4900	10000	11696	13790	15181

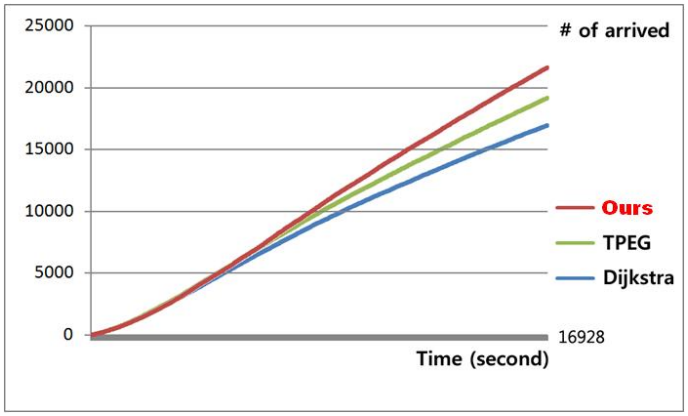
실험시간은 각 지점에서 자동차가 순서대로 출발한 이후 실험이 종료되기까지의 초 단위의 시간이다. 그리고 각 알고리즘 별로 측정된 숫자는 단위 시간에 대해 목적지까지 도착한 자동차의 대수이다. 세 상황에 대해 제시한 알고리즘(Ours)의 도착한 자동차의 수가 더 많고, 이를 통해 본 알고리즘을 사용했을 경우 교통류의 흐름이 더 원활하여 더 많은 자동차가 빨리 목적지에 도달함을 알 수 있다.

[그림 5]는 일반적인 교통 상황에서 각 알고리즘들의 성능을 평가한 그래프이다. 시뮬레이션 초기에는 각 자동차 사이에 교환되는 정보가 적기 때문에 기존 알고리즘보다 떨어지는 결과를 보이지만, 시간이 지나면서 다른 알고리즘을 능가하게 되고 점점 다른 알고리즘과 그 격차를 벌여 나가는 것을 볼 수 있다. 이는 시간에 따라 성능이 진화함을 보여 주는 부분이다.



[그림 5] Normal Case(보통 경우)

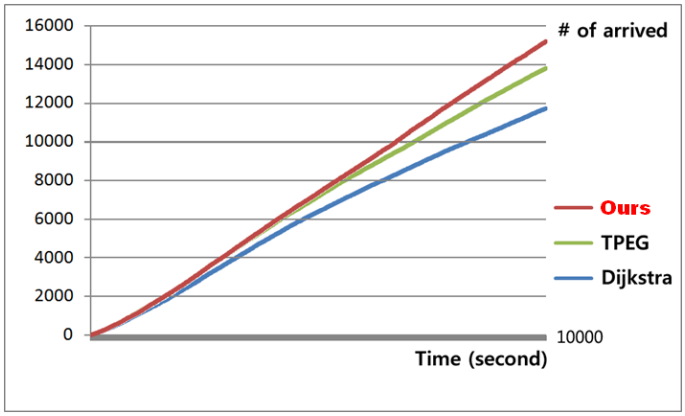
[그림 6]은 10%의 네비게이션이 통신 고장을 일으켰을 경우 나머지 90%가 정상적으로 목적지에 도달할 수 있는지를 도착 차량의 수로 검증한 그래프이다. 본 연구에서 제시한 알고리즘은 각각의 자동차가 적응적으로 정보를 얻고 행동하기 때문에 개개의 기기 고장에도 다른 정상적인 기기 상호 간의 정보 전달을 통해 나머지 기기들의 정상적 수행을 돕는다. 따라서 상대적으로 고장에 대한 위험 부담이 감소되며 생존성이 보장된다.



[그림 6] Survivability Case(고장 발생)

[그림 7]은 돌발사고 발생으로 특정 링크의 교통 혼잡도가 크게 증가하는 상황에서도 최적의 경로를 찾아낼 수 있는지를 검증한 그래프로, 다른 알고리즘의

경우 사고에 대해 적응하지 못해서 제안한 알고리즘보다 더 낮은 성능을 보인다. 이를 통해 알고리즘이 주어진 환경의 변화에 잘 적응한다는 것을 검증할 수 있다.



[그림 7] Adaptability Case(사고 발생)

4. 결론

본 연구에서 제시한 알고리즘은 일반적인 최단 경로 알고리즘과 또는 TPEG과 같은 중앙 집중형 정보 방송 시스템보다 여러 가지 장점을 가지고 있다. 계산 시간의 문제를 대체 경로 탐색 방법으로 해결할 수 있고, TPEG의 단점이 되는 초기 투자 비용 및 과금 문제를 저가의 하드웨어의 보급으로 해결할 수 있다. 그리고 전해지는 정보가 즉각적이며, 큰 길 뿐만 아니라 자동차가 지나갈 수 있는 모든 길, 즉 네비게이션에 저장되어 있는 모든 길에 대해 그 길을 지나쳐 온 자동차가 있다면 그 교통 상황을 파악할 수 있다. 이처럼 새로운 경로 탐색 방법의 하나로써 중앙 집중형 정보 방송 시스템의 문제를 해결하고 현재 교통 상황을 실시간으로 반영하는 길 찾기 알고리즘을 개발한 것은 궁극적으로 교통 흐름의 완화 및 긴 이동 시간으로 인한 사회적 비용 감소의 효과를 가져옴으로써 사회 전반에 큰 의미가 있다고 하겠다.

추후 매개변수의 조정 및 더 많은 자동차를 사용한 실험을 통해 알고리즘의 우수성을 추가로 검증할 수 있을 것이다. 뿐만 아니라 현재 문제가 되고 있는 제한된 정보 공유 문제를 해결하여 더 자세한 도로 정보 및 통행 시간 측정 정보 등을 확보할 수 있다면 더욱 더 현 교통 상황에 맞는 정밀한 실험을 수행할 수 있을 것이라 기대된다.

참고문헌

[1] 안창욱, R.S. Ramakrishna, 강충구. "최단 경로 라우팅을 위한 새로운 유전자 알고리즘", 한국통신학회논문지 제 27권 12C호. 2002.

[2] 건설교통부 건설교통종합정보센터, "http://nodelink.its.go.kr".

[3] A. Nagurney and Q. Qiang. "Robustness of transportation networks subject to degradable links", A Letters Journal Exploring the Frontiers of Physics, 68001-p1-p6, 2007.

[4] 정일호, 손동혁. "고속도로 사업효과 조사", 국토연구원, 1995.