도로망의 계층성을 이용한 휴리스틱 주행 경로 탐색

김기홍 유승원 강현민 차상균
서울대학교 전기공학부 E-mail: {next, mokmon, lovjesus, chask}@kdb.snu.ac.kr

A Heuristic Path-Finding Scheme Based on the Road Hierarchy

Kihong Kim Seungwon Yoo Hyun M. Kang Sang K. Cha
School of Electrical Engineering, Seoul National University

요약

주행 경로 안내는 최근 국내에서도 활발히 연구되고 있는 지능형 교통 시스템(ITS)의 주요 기능 중 하나이다. 주행 경로 안내를 위해서는 대규모 도로망에서 신속하게 경로를 찾는 방법과 찾은 경로를 음성 또는 기호로 운전자에게 효율적으로 안내하는 방법 등이 필요하다. 본 논문에서는 도로망의 계층성을 휴리스틱 정보로 활용하여 최단시간 경로를 효율적으로 찾는 방법을 제안한다. 제안된 방법은 고속국도, 국도 등의 고속 주행용 도로만으로 소규모 상위 계층 도로망을 만들고 이를 기존 도로망에 덧붙이는 방식으로 통합한다. 이 통합망에 상위 계층 도로망을 우선적으로 찾도록 구성한 A* 알고리즘을 수행하여 최단시간 경로를 찾는다. 또 경로 탐색용 그래프가 디스크에 저장된 경우에, 디스크 접근을 최적화하기 위한 데이터베이스 설계 및 디스크 접근 방법을 기술한다.제안된 방법의 효율성을 검증하기 위해 서울시 도로망 데이터를 이용하여 실험한 결과, 제안된 방법을 통해 경로 탐색 소요 시간, 디스크 입출력 회수, 메모리 사용량 등을 75% 이상 줄일 수 있었다.

1 서론

주행 경로 안내는 최근 국내에서도 활발히 연구되고 있는 지능형 교통 시스템(Intelligent Transportation System)의 주요 기능 중 하나이다. 주행 경로 안내를 위해서는 도로망에서 신속하게 경로를 찾는 방법과 찾은 경로를 음성 또는 기호로 운전자에게 효율적으로 안내하는 방법 등이 필요하다. 특히, 도로망은 대규모 그래프로 모델되기 때문에 효율적인 경로 탐색 알고리즘이 중요하다. 예로, 한 서울시 수치 지도는 71,176 개의 교차로와 106,533개의 도로 구간을 가지며, 미국 캘리포니아 주의 경우 교차로만 8백만 개를 넘는다

경로 탐색은 기본적인 그래프 문제의 하나로 다양한 연구 결과가 있지만, 이를 대규모 도로망에 대한 경로 탐색에 그대로 적용하기는 어렵다. 노드 수를 n, 링크 수를 m이라고 할 때, 널리 알려진 Dijkstra's 알고리즘이나 A* 알고리즘의 가장 좋은 성능은 $O(m+n\log n)$이다[1][2] 경로를 미리 계산하여 저장하는 방법을 이용하면 $O(1)$의 성능을 얻을 수 있지만, 모든 노드 간의 경로를 저장하기 위해 $O(n^2)$의 저장 공간을 필요로 한다[3] 최근에는 그래프를 계층적으로 구성하여, 미리 계산된 경로 저장에 필요한 공간을 줄이는 방법도 제안되었다[4][5]. 그러나, 이 방법도 대규모 도로망에 대해서는 여전히 대량의 저장 공간이 필요하며, 링크 비용이 바뀔 때 계층 그래프를 재구성하는 비용이 크다는 문제가 있다[6][7].

본 논문에서는 도로망의 계층성을 휴리스틱 정보로 활용하여 최단시간 경로를 찾는 방법을 제안한다. 제안된 방법은 고속국도, 국도, 지방도 등과 같은 고속 주행용 도로만으로 상위 계층 도로망을 만들고 이를 기존 도로망과 통합한다. 이 통합망에 상위 계층 도로망을 우선적으로 찾도록 구성한 A* 알고리즘을 수행하여 최단시간 경로를 찾는다 또 경로 탐색용 그래프가 디스크에 저장된 경우에, 디스크 접근을 최적화하기 위해 사용한 데이터베이스 설계 및 디스크 접근 방법을 기술한다. 끝으로, 서울시 도로망 데이터를 이용하여 제안된 방법의 효율성을 검증하는 실험을 수행하였다. 제안된 방법을 통해 경로 탐색 소요 시간, 디스크 입출력 회수, 메모리 사용량 등을 75% 이상 줄일 수 있었다.

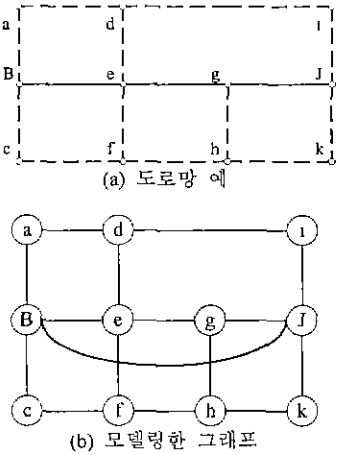
본 연구는 "국가 GIS 기술 개발 사업"의 지원을 받았습니다

2 휴리스틱 주행 경로 탐색

2.1 계층성 휴리스틱

자동차로 이동할 경우 사람들은 대부분 사전에 계층적으로 주행 경로를 계획하며, 특히 장거리 이동의 경우 이러한 현상이 두드러진다. 예로, 1) 고속도로, 국도 등의 큰도로만으로 출발지 인근과 목적지 인근을 연결하는 개략적인 큰도로 경로를 구성한 후, 2) 이 경로를 출발지와 목적지 인근의 작은도로와 함께 고려하여 완전한 경로를 구성한다. 이때 출발지나 목적지가 큰도로 경로에 효율적으로 연결되지 않을 경우, 또 다른 큰도로 경로를 찾아서 단계 2를 반복한다.

이처럼 두 단계 또는 그 이상의 단계로 주행 경로를 계획하는 이유는, 고려할 도로의 양이 크게 줄어들 뿐만 아니라, 많은 경우 큰도로만 고려하여도 최적의 경로를 찾을 수 있기 때문이다 이는 큰도로일수록 고속 주행용으로 건설되었고 실제 주행 속도도 빨라서, 큰도로를 통해 조금 돌아가더라도 작은도로로 주행하는 것보다 더 빨리 목적지에 도착할 수



(a) 도로망 예

(b) 모델링한 그래프

그림 1. 계층성 휴리스틱을 이용한 그래프 모델링

있다는 경험에 근거한다. 이러한 경험적 사실을 앞으로 계층성 휴리스틱이라 부른다.

2.2 최단시간 경로 탐색 알고리즘

본 논문에서 제안하는 경로 탐색 방법은 1) 계층성 휴리스틱을 활용할 수 있도록 도로망을 그래프 구조로 모델한 후, 2) 모델된 그래프에 A* 알고리즘을 적용하여 최단시간 경로를 찾는다

그림 1은 본 논문에서 제안하는 계층성 휴리스틱을 이용하여 도로망을 그래프로 모델하는 예이다. 알고리즘 1을 사용하여 그림 1(a)의 도로망을 그림 1(b)의 그래프로 모델한다. 그림 1(a)에서 실선은 큰도로를, 파선은 작은 도로를 나타낸다.

이렇게 만들어진 그래프에 A* 알고리즘을 적용하면 최단시간 경로를 효율적으로 탐색할 수 있다. A* 알고리즘은 Dijkstra's 알고리즘의 변형으로, 추측한 목적지까지의 잔여비용과 현재까지 계산된 비용의 합을 기준으로 방문 후보 노드의 순서를 정한다 A* 알고리즘은 추정한 잔여비용을 실제값보다 항상 작을 때, 최적의 해를 찾을 수 있다[8]. 예로, 잔여비용을 목적지까지의 직선 거리로 잡으면 최단거리 경로를 찾을 수 있다. 최단시간 경로를 찾기 위해서는 목적지까지의 직선 거리를 최고 제한속도로 나눈 값을 잔여 비용으로 사용한다.

알고리즘 2는 계층성 휴리스틱을 이용하여 경로를 찾을 수 있도록 A* 알고리즘을 수정한 것이다. 차이는 현재 노드에서 목적지까지의 직선 거리를 미리 정해진 속도 V로 나눈 값을 추정 잔여 비용으로 사용한다는 점이다. 직선 거리를 계산하기 위해 노드가 자신의 위치를 저장하고 있다고 가정한다. 속도 V의 값은 알고리즘의 효율을 결정하는 인자로 경험적으로 결정하는 값이다. V 값이 클수록 A* 알고리즘에 가까워지며, 작을수록 큰도로를 이용하여 경로를 빨리 찾지만 최적의 해를 찾을 확률도 떨어진다. 참고로, 본 논문의 실험에서는 V값으로 작은도로의 최고 속도를 사용하였다.

알고리즘 1: 계층성 휴리스틱을 이용한 그래프 모델링

1. 큰도로에 해당하는 도로 구간과 교차로를 선택한다 그림 1에서는 교차로 B, e, g, J와 이들 사이의 도로 구간을 선택한다.
2. 선택된 교차로 중에서 두 개의 큰 도로만 연결하는 교차로는 지우고, 연결된 도로 구간은 병합한다. 그림 1에서는 교차로 e와 g를 지운다
3. 과정 2에서 병합한 도로 구간을 원래 그래프에 추가한다. 그림 1에서는 굵은 선으로 표시된 도로 구간 BJ를 추가하였다.

알고리즘 2: 계층성 휴리스틱을 이용한 경로 탐색

1. 출발지를 후보 노드 집합에 넣는다
2. 후보 노드 집합에서 최소 비용 노드 N을 빼낸다. N이 목적지이면 알고리즘을 종료한다.
3. N에서 갈 수 있는 모든 노드 노드 $\{M_i\}$에 대해 N을 경유하는 비용과 목적지까지의 추정 비용을 계산한다. 전자는 N의 비용에 링크 NM_i의 비용을 더한 값이며, 후자는 M_i에서 목적지까지의 직선 거리를 미리 정해진 속도 V로 나눈 값이다
4. 새로 계산된 두 값의 합이 M_i에 저장된 값보다 작으면 M_i의 비용을 갱신하고, M_i를 후보 노드 집합에 추가하고, 과정 2로 돌아간다.

예로 교차로 c에서 i까지 가려는 경우 알고리즘 2가 경로를 찾는 과정을 밝아보자.

1. 그림 1(b)의 c 노드에서 출발하여 노드 B와 노드 f를 후보 노드 집합에 넣는다.
2. 그림에서는 구분하기 어렵지만 B와 f중에서 B의 비용이 더 낮다고 가정하면, B에 연결된 노드 a, e, J를 방문하게 된다.
3. 후보 노드 a, c, f, J 중에서 J의 비용이 가장 작으므로, J와 연결된 노드 g, i, k를 방문한다. 예로 J와 a의 비용을 비교하면 다음과 같다. J의 비용은 링크 cB와 링크 BJ의 소요시간에 J와 i 사이의 직선 거리를 V로 나눈 값이다. a의 비용은 링크 cB와 Ba의 소요 시간에 a와 i사이의 직선 거리를 V로 나눈 값이다. Ba의 소요 시간은 J와 i 사이의 직선 거리를 V로 나눈 값보다 크며, a와 i 사이의 직선 거리를 V로 나눈 값은 BJ의 소요시간보다 크다. 따라서 J의 비용이 a의 비용보다 작다.
4. 후보 노드 a, e, f, g, i 중에서 i의 비용이 최소이므로 알고리즘을 종료한다. 참고로, 현재 g의 비용은 링크 cB, BJ, Jg 소요시간의 합에 g에서 i까지의 직선 거리를 V로 나눈 값을 더한 것이다. 따라서, i의 비용이 g의 비용보다 작다.

계층성 휴리스틱을 적용하기 위해서는 도로를 큰도로와 작은도로로 나눌 수 있어야 한다. 일반적으로 도로 구간은 고속국도, 국도, 지방도로, 일반도로, 이면도로 등과 같이 분류되며, 이러한 분류의 앞 단계일수록 장거리 이동에 사용된다. 따라서 이를 기준으로 도로를 구분할 수 있으며, 예로 일반도로 이상을 큰도로로 그 이하를 작은도로로 나눌 수 있다 또 다른 방법으로 도로의 폭, 차선 수, 제한속도 등을 기준으로 나눌 수도 있다.

3 디스크 접근 최적화

자동차 항법 시스템(Car Navigation System)과 같이 경로 탐색

이 빈번하지 않은 환경에서는, 도로망 데이터를 디스크에 저장하여 두고 경로를 탐색할 때 필요한 데이터만 메모리에 올리는 방법이 요구된다. 이 경우 디스크 접근 시간을 최소화할 수 있도록, 도로망 데이터베이스를 구축하고 경로 탐색에 필요한 도로망 데이터를 효율적으로 읽어오는 방법이 필요하다.

필요한 데이터만 메모리에 올리기 위해, 본 논문에서는 디스크에 저장된 도로망 데이터로부터 경로 탐색용 그래프를 주메모리상에 점진적으로 구성한다. 이를 위해 메모리로 읽어온 노드의 ID를 해쉬 테이블로 관리한다. 또, 알고리즘 2의 과정 3에서 노드 폴트(fault)가 일어나면, 1) 해당 노드를 디스크에서 읽어 메모리에 저장하고, 2) 그 식별자를 해쉬 테이블에 등록한 후, 3) 그 노드에 저장된 연결 노드의 식별자를 스위즐링(swizzling)한다[9] 이때 연결 노드가 메모리 상에 없을 때는, 아직 스위즐되지 않았음을 표시한다.

제안한 알고리즘을 포함한 대부분의 경로 탐색 알고리즘은 출발점을 기준으로 검색 범위를 원형 또는 타원형으로 넓혀간다. 따라서, 전술한 바와 같이 점진적으로 경로 탐색용 그래프를 주메모리에 구성하면 디스크 접근 시간이 길어진다. 이런 이유로 도로망 데이터를 디스크 페이지 단위로 클러스터링한 후, 한 페이지를 읽을 때 그 페이지에 저장된 모든 노드를 메모리에 올리는 방법이 제안되었다[10]. 본 논문에서는 더 나아가 검색 범위를 예상하여 이 범위에 속하는 큰도로를 한 번에 메모리에 올리는 방법을 함께 사용한다. 예상 범위는 출발지와 목적지를 초점으로 하는 타원으로 잡는다.

제안된 방법에서는 예상 영역에 속하는 노드를 빨리 찾기 위해, 노드 좌표를 기준으로 공간 색인을 구축한다. 구현한 공간 색인은 그리드 파일과 Hilbert R-tree이다[11][12]. 또, 도로망 데이터에 공간 클러스터링을 직용하여 영역을 기준으로 검색된 결과를 디스크에서 신속하게 읽을 수 있도록 한다. 사용된 방법은 노드 데이터를 그리드 파일의 데이터 페이지 또는 R-tree의 리프 노드에 직접 저장하는 일차 색인 방법이다 또 데이터 페이지를 Hilbert 값의 순서대로 전역적으로 클러스터링하였다

4 성능 평가

본 논문에서 제안한 경로 탐색 알고리즘의 성능을 측정하기 위한 실험 결과를 기술한다. 이 실험에서는 계층성 휴리스틱을 사용하여 그래프를 구성한 경우와 그렇지 않은 그래프를 대상으로 A* 알고리즘의 수행 시간을 비교한다. 실험에 사용한 데이터는 서울시 도로망으로, 106.533개의 도로 구간과 71,176개 정도의 교차로로 이루어진다

실험에서는 4차선 이상의 일반 도로 및 지방도 이상의 도로를 큰도로로, 나머지를 작은 도로로 분류하였다. 또, 도로 구간의 이동 속도는 제한 속도의 90%로 하였다. 또 그리드 파일을 사용하여 공간 색인을 구축하고, 이를 이용한 일차 색인 방법과 공간 순서화 방법을 사용하여 클러스터링하였다. 그리드 파일의 페이지 크기는 4KB이다 그리드 파일 패킹 방법을 사용하여 파일의 크기를 줄였으며, 줄어든 크기가 약 10MB이다.

표 1은 "서대문구 남가좌동 명지대 사거리"에서 "잠실종합 운동장"까지의 주행 경로를 찾는 경우에 대한 실험 결과이다. 실험 결과는 계층성 휴리스틱을 이용하면 경로 탐색 알고리즘의 성능을 약 4배 향상시킬 수 있음을 나타낸다. 이 경우, 소요 시간이 약 23%로 줄었으며, 디스크 접근 회수는 30%로 줄었다 방문한 노드 개수는 메모리 사용량을 나타내는 지표인데, 계층성 휴리스틱을 사용하지 않는 경우에 비해 약 25%의 메모리만을 사용하여 경로를 찾았다.

	계층성 휴리스틱을 사용한 경우	계층성 휴리스틱을 사용하지 않은 경우
소요시간	(cold) 3.01초 (hot) 2.94초	(cold) 13 66초 (hot) 12.42초
읽은 페이지 개수	850회	2,872회
방문한 노드 개수	7,929개	31,871개

표 1 성능 평가 실험 결과

5 결론

본 논문에서는 계층적으로 주행 경로를 계획하는 인간의 사고 패턴을 반영하는 휴리스틱 경로 탐색 알고리즘을 제안하였다. 또, 자동차 항법 시스템과 같이 경로 탐색용 그래프가 디스크에 저장된 경우에, 디스크 접근을 최적화하기 위해 사용한 데이터베이스 설계 및 접근 방법을 기술하였다. 끝으로, 성능 평가 실험을 통해 제안된 방법이 경로 탐색 시간, 디스크 접근 회수, 메모리 사용량 등의 측면에서 A* 알고리즘에 비해 4배 이상 효율적임을 보였다.

참고문헌

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows. Theory, Algorithms, and Applications*, Prentice Hall, 1993.

[2] M. L. Fredman and R E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms", *Journal of the ACM*, Vol. 34, No. 3, pp 596-615, July 1987

[3] R. Agrawal and H. V. Jagadish, "Materialization and incremental update of path information", in *Proceedings of ICDE*, 1989, pp. 374-383.

[4] S Jung and S. Pramanik, "HiTi graph model of topographical road maps in navigation systems", in *Proceedings of ICDE*, 1996, pp. 76-84

[5] Y.-W. Huang, N. Jing, and E. Rundensteiner, "Hierarchical optimization of optimal path finding for transportation applications", in *Proceedings of ACM CIKM*, 1996, pp. 261-268.

[6] S. Shekhar, A. Fetterer, and B Goyal, "Materialization trade-offs in hierarchical shortest path algorithms", in *Proceedings of SSD*, 1997, pp. 94-111.

[7] K. Kim, S Yoo, and S. K. Cha, "A partitioning scheme for hierarchical path finding robust to link cost update", in *Proceedings of 5th World Congress of ITS*, 1998.

[8] D. Galperin, "On the optimality of A*", *Artificial Intelligence*, Vol. 8, No. 1, pp. 69-76, 1977.

[9] J. E.B. Moss. "Working with Persisten Objects To Swizzle or Not to Swizzle", *IEEE Transaction on Software Engineering*. Vol. 18, No. 8, pp. 657-673, August 1998.

[10] Y.-W. Huang, N Jing, and E. A. Rundensteiner, "Effective graph clustering for path queries in digital map databases", in *Proceedings of ACM CIKM*, pp. 215-222, 1996

[11] J. Nievergelt. H. Hinterberger, and K. C. Sevcik, "The grid file: An adaptable, symmetric multikey file structure", *ACM Transaction on Database Systems*, Vol. 9, No. 1, pp 38-71, 1984.

[12] I. Kamel and C. Faloutsos, "Hilbert R-tree: An improved R-tree using fractals", in *Proceedings of VLDB*, pp. 500-509, 1994.