Lin-Kernighan 휴리스틱에 기반한 차량 텔레매틱스 운행 경로의 결정 1)

이정훈, 홍영신, 박경린
제주대학교 전산통계학과
e-mail:{jhlee, yshong, glpark}@cheju.ac.kr

Path finding for vehicular telematics based on the Lin-Kernighan heuristic

Junghoon Lee, Youngshin Hong, Gyung-Leen Park
Dept of Computer Science and Statistics, Cheju National University

요 약

본 논문은 차량 텔레매틱스 시스템에서 중요한 응용중의 하나인 다중 목적지 방문을 위한 경로 결정 방식을 구현하기 위하여 Lin-Kernighan 휴리스틱을 텔레매틱스 시스템에 결합하는 방법에 대해 기술하고 다중 목적지 결정 서버를 구현한다. 서버는 클라이언트는 로드 네트워크에 대한 자료구조를 공유하고 있으며 클라이언트가 목적지 리스트를 요청하면 1) 서버가 A* 기법에 의해 각 목적지간의 비용을 계산하고 2) Lin-Kernighan 프로그램의 인자로 변환하여 3) 경로 결정 모듈을 수행시킨다. 이 경로의 순서는 클라이언트에게 정해진 메시지 포맷에 의해 전달되며 클라이언트는 각 인접한 목적지간에 A* 기법에 의해 실제 도로 네트워크 상에서의 경로를 결정하여 사용자에게 제공한다. 성능측정 결과 방문지 수가 많더라도 수초 이내에 경로를 결정할 수 있으며 그 정확성도 거의 100%에 근접한다.

1. 서론

차량 텔레매틱스 시스템에서 다중목적지 경로를 결정하는 서비스는 유용한 위치기반 서비스 중의 하나로서 관광객이 여러 관광지를 결정하고 그 방문순서를 결정하는 과정에서도 사용될 수 있으며 택배회사에서도 고객들간의 방문 순서를 결정하기 위해서도 사용된다. 다중 목적지 방문 문제는 결국 차량이 원래의 출발지로 돌아와야 한다는 점에서 TSP(Traveling Salesman Problem)의 솔루션이 적용된다. 점대점 경로 결정은 Dijkstra 방식이 최적 솔루션으로 알려져 있는데 반해 TSP 문제는 O(n!)의 복잡도를 갖기 때문에 다양한 휴리스틱들이 개발되어 왔다.

TSP 휴리스틱들 중에 Lin-Kernigan 방식이 성능이 일반적으로 우수한 것으로 알려져 있다. 이 방식은 출발지부터 시작하여 아직 방문되지 않은 가장 가까운 노드를 다음 순서로 정하려고 한다. 이 과정에서 국지 개선 기법을 거치는데 이는 $(i_0, \ldots i_{n-1})$의 p 번째 및 q 번째 목적지에 대해 다음의 조건을 만족시키는지 검사한다.

$$C(i_{p-1}, i_p) + C(i_q, i_{q+1}) \leq C(i_{p-1}, i_q) + C(i_p, i_{q+1}) \quad (1)$$

이 조건을 만족시키지 못하는 쌍이 있으면 (2-1)을 (2-2)에서 보는 바와 같이 변경하며 이 과정을 모든 쌍들이 (1)을 만족시킬 때까지 반복한다.

$$(i_0, \ldots, i_{p-1}, i_p, \ldots i_q, i_{q+1}, \ldots, i_{n-1}) \quad (2\text{-}1)$$

$$(i_0, \ldots, i_{p-1}, i_q, i_{q-1}, \ldots, i_{p+1}, i_p, i_{q+1}, \ldots, i_{n-1}) \quad (2\text{-}2)$$

이 방법은 널리 응용되고 있을 뿐만 아니라 concorde 싸이트에 소스 프로그램이 공개되어 다양한 솔루션에 이용될 수 있다[1]. 다운로드된 프로그램을 설치하여 구동시키기 위해서는 프로그램이 정의한 파일의 포맷에 맞추어 노드간 비용 행렬을 입력한다. 또 이 서비스를 차량의 텔레매틱스 장치에서 요청하기 위해서는 서버와 클라이언트간에 같은 로드 네트워크 자료구조를 공유하고 있어야 하며 클라이언트측에서 목적지를 선정하여 서버에 보내면 서버는 목적지들간의 비용 행렬을 구하여야 한다. 본 논문에서는 차량 텔레매틱스 클라이언트에게 다중 목적지 방문 스케줄링을 제공하는 구조를 제안 및 구현한다.

2. 시스템 구성

2.1 서버의 구성

서버는 리눅스 운영체제 상에 구현되어 클라이언트들과의 통신과 비용 행렬 계산 및 Link-Kernighan 프로그램 구동 등의 기능을 갖고 있다. 먼저 클라이언트와 서버간에는 (그림 1)과 같은 메시지 포맷에 따라 데이터를 교환하는데 Command 필드는 Request (클라이언트->서버) 혹은 Reply (서버->클라이언트)의 값을 가질 수 있다. 첫 필드는 노드의 수를 나타내는 필드로서 뒤따라오는 노드 번호, 즉 방문지의 개수를 지정한다. 이어 이 개수만큼의 노드들이 지정된다.

Command	Length	Node1	Node2	...	NodeN

(그림 1) 메시지 포맷

1) 본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (IITA-2008-C1090-0801-0040)

서버가 TSP 문제의 답을 얻기 위해서는 노드의 개수를 n이라 할 때 이 메시지를 받아 각 노드들간의 비용을 계산하는 n×n 비용 행렬을 계산한다. 이 과정에서 다익스트라 알고리즘은 최적의 답을 찾아내기는 하지만 n×(n-1) 번의 계산을 하기엔 그 수행시간이 길어지게 되어 적용이 불가능하다. 따라서 고속으로 허용가능한 오차 이내의 비용을 계산할 수 있는 A* 알고리즘을 적용한다. 이 알고리즘은 데이터에 의존성이 없으므로 병렬 프로그램으로 구현되어 속도를 향상시킬 수 있다[3]. 계산된 비용 행렬을 Concorde 패키지에 적용하기 위해서 (그림 2)의 예에서 보는 바와 같은 파일 포맷으로 변환한다. 각 필드는 노드의 개수 (11개), 요구되는 솔루션 타입(TSP), 비용입력타입 (비용명시), 포맷(상위삼각행렬) 등을 설정한다. 이에 의해 EDGE_WEIGHT_SECTION에 각 링크의 비용이 순서대로 입력된다. 단 노드의 아이디를 입력할 수는 없으므로 입력 순서와 노드 아이디에 대한 mapping 정보를 추가로 관리하여야 한다.

```
NAME: tour.tsp
TYPE: TSP
COMMENT: Tour Planning
DIMENSION: 11
EDGE_WEIGHT_TYPE: EXPLICIT
EDGE_WEIGHT_FORMAT: UPPER_ROW
EDGE_WEIGHT_SECTION
2707 345 4124 2083 3949 431 5156 3847 3958 3677
2936 3415 4885 4343 2808 6210 6387 4992 4638
4369 1731 3839 490 4916 3515 3960 3567
5501 937 3885 2257 5734 1262 1154
5075 2311 3361 1901 5202 4803
3553 1181 5631 330 220
4751 4090 3644 3251
4017 833 1070
4430 4558
315

EOF
```

(그림 2) Concorde 파일 포맷

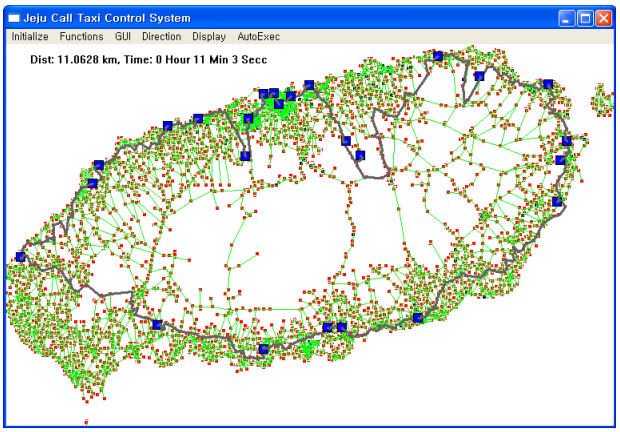
이 파일은 tour.tsp로 저장되게 되며 이는 다음과 같은 함수 호출에 의해 Lin-Kernighan 프로그램과 연동된다.

system("./linkern -o outputfile tour.tsp");

결국 수행결과가 outputfile에 저장되어 네트워크를 통해 클라이언트에게 반환된다.

2.2 클라이언트 기능

클라이언트는 기본적으로 사용자의 인터페이스를 제공하는 부분과 네트워크 연산을 수행하는 부분으로 나뉜다. 사용자의 입력 처리 부분은 사용자에게 기본적인 지도 인터페이스를 제공하고 마우스에 의해 선택된 좌표에 의해 최근점 노드를 찾아 노드 아이디로 변환한다. 이 리스트에 의해 (그림 1)에서 보는 Request 메시지를 구성하여 서버에게 요청을 하며 서버의 계산에 의해 결정된 방문 순서를 Reply 메시지로 수신한다. 이후 각 방문순서상 인접노드간에 역시 A* 알고리즘에 기반한 경로를 구하여 사용자에게 제공한다. (그림 3)은 결과 화면을 도시한 것이며 이 인터페이스는 Pocket PC 버전으로도 구현되었다.



(그림 3) 클라이언트 화면 및 경로 탐색 결과

3. 성능 측정 결과 및 결론

일반적으로 휴리스틱을 사용하면 최적을 해답을 얻을 수 있는 Brute-force 방식보다 수행시간은 현저히 향상된다. 특히 O(n!)을 갖는 TSP 문제의 경우 코어2듀오 2.2Ghz, 메모리 2GB일반적인 PC 와 Linux 운영체제 플랫폼에서 n이 11를 넘게 되면 수행시간이 이미 수초를 넘게 되어 사용자가 감내할 수 없다. 본 시스템은 Lin-Kernighan 휴리스틱에 의해 구현되었기 때문에 수행시간은 상당히 빠르다. 실험 결과는 노드의 수가 10이면 1초 이내, 20이면 3초이내, 30이면 7초 이내에 응답을 받을 수 있었으며 이 경우에도 TSP 휴리스틱이 수행되는 시간은 1초 이내인 것으로 출력되고 있다. 따라서 비용 행렬을 구성하기 위한 A* 알고리즘의 수행시간이 전체 응답시간에 더욱 큰 영향을 주고 있다.

이와 아울러 휴리스틱의 정확성에 대한 분석을 위해 brute-force 방식을 구현하여 성능을 비교하였다. 실험에 있어서 노드 9와 12 사이의 값을 갖는 목적지 집합을 40개씩 생성하였으며 이 경우 오직 노드의 수가 9인 경우 하나의 불일치가 발생할 뿐이며 그 외에는 모두 최소비용 방문순서를 찾아내는 것을 발견하였다.

결국 본 시스템에서 구현된 TSP 결정 서비스는 텔레매틱스 서비스에서 다중 목적지의 방문 순서를 결정하는데 효율성을 기할 수 있으며 추후 사용자의 다양한 요구사항을 반영하는 경로를 찾을 수 있도록 발전할 예정이다.

참고문헌

[1] http://www.tsp.gatech.edu/concorde/downloads/codes/src/co031219.tgz

[2] A. Goldberg, H. Kaplan, and R. Werneck, "Reach for A*: Efficient point-to-point shortest path algorithms," MSR-TR-2005-132. Microsoft, 2005.

[3] J. Lee, G., Park, H., Kim, Y., Yang, P., Kim, P., and S. Kim, "A telematics service system based on the Linux cluster," LNCS, Vol. 4490, 2007, pp. 660-667.