ACE-BIS: 최적의 버스 노선을 선택하기 위한 비용 효율적인 알고리즘의 개발

(ACE-BIS: A Cost-Effective Bus Information System)

이 종 찬 † 박 상 현 †† 서 민 구 ††† 김 상 욱 ††††

(Jongchan Lee) (Sanghyun Park) (Minkoo Seo) (Sangwook Kim)

요 약 최근, 모바일 기술 및 GPS 기술의 발전으로 인하여 다양한 위치 기반 서비스가 크게 각광을 받고 있다. 이에 따라 본 논문에서는 모바일 기기를 사용하여 목적지에 도착할 수 있는 다양한 버스 노선 정보를 손쉽게 제공할 수 있는 모바일 대중교통 정보 시스템 ACE-BIS(A Cost-Effective Bus Information System)를 제안한다. 높은 통신비용과 서버의 과부하를 초래하는 기존의 교통 정보 시스템과는 달리, ACE-BIS는 모바일 기기 내에 저장된 버스 정류장 및 노선 데이타를 이용해 휴리스틱 알고리즘을 수행함으로써 목적지까지의 노선 정보 및 예상 소요 시간을 사용자에게 제공한다. 또한 별도의 통신비용을 부담하려는 사용자에게는 서버와의 통신을 통해 버스의 현재 위치 및 도로 체증 상황 등의 실시간 교통 정보를 반영한 좀 더 정확한 경로 정보를 제공한다. 아울러, 서버 내에서 관리되는 실시간 교통 정보를 이용하여 미래 시점의 경로 정보에 대한 서비스도 제공한다. 현실 세계의 특성을 반영한 가상 데이타를 대상으로 다양한 실험을 수행함으로써 제안된 시스템의 정확성과 효율성을 검증한다.

키워드 : 대중교통 정보 시스템, 휴리스틱 알고리즘, 위치 기반 서비스, 버스 정보 시스템

Abstract Due to the rapid development in mobile communication technologies, the usage of mobile devices such as cellular phones and PDAs becomes increasingly popular. One of the best ways to maximize the usability of mobile devices is to make them aware of their current locations and the locations of other fixed and mobile objects. In this paper, we propose a cost-effective Bus Information System, ACE-BIS, which utilizes a mobile device to retrieve the bus routes to reach a destination from the current location. To accomplish this task, ACE-BIS maintains a small amount of information on bus stops and bus routes in a mobile device and runs a heuristic routing algorithm based on such information. When a user asks more accurate route information or calls for a "leave later query", ACE-BIS entrusts the task to a server into which real-time traffic and bus location information is being collected. By separating the roles into a mobile device and a server, ACE-BIS is able to provide bus routes at the lowest cost for wireless communications, without imposing much burden to a server. The results of extensive experiments revealed that ACE-BIS is effective and scalable in most experimental settings.

Key words : Mobile public transportation system, Heuristic routing algorithm, Location-Based services, Bus Information System

1. 서 론

최근, GPS 및 모바일 기기 기술의 발전으로 인하여 사용자 위치 정보 안내[1] 및 인접 관광지 안내[2,3] 등과 같은 위치 기반 서비스(LBS: Location-Based Services)에 대한 관심이 늘고 있다. 또한, 교통 정보 시스템을 LBS에 결합하여 도로 네트워크를 관리하거나 목적지까지의 이동 경로를 제공하는 모바일 교통 정보 시스템에 대한 연구 및 애플리케이션의 개발 역시 활발하게 진행되고 있다[1,4-9]. 특히, BIS(Bus Information

· 본 연구는 제주 대학교를 통한 정보통신부 및 정보통신진흥원의 대학IT연구센터 지원사업의 부분적인 지원을 받았음(IITA-2005-C1090-0502-0009)

† 학생회원 : 연세대학교 컴퓨터과학과
jclee@cs.yonsei.ac.kr
†† 종신회원 : 연세대학교 컴퓨터과학과 교수
sanghyun@cs.yonsei.ac.kr
††† 학생회원 : 한국과학기술연구원(KAIST) 전산학과
mkseo@bulsai.ac.kr
†††† 종신회원 : 한양대학교 정보통신학과 교수
wook@hanyang.ac.kr

논문접수 : 2005년 11월 28일
심사완료 : 2006년 9월 15일

System)와 BMS(Bus Management System)는 기존의 버스운영체계에 정보, 통신, 컴퓨터, 전자, 제어 등의 첨단 IT 기술을 접목시켜 버스의 이용효율을 제고시키는 지능형 교통체계시스템의 한 분야로 크게 각광받고 있는 분야이다. 국내에는 98년 5월 처음으로 서울시에서 종로1가부터 동대문까지 약 6Km 구간에 대해 BIS를 도입하였고 송파구 관내의 롯데월드와 잠실 전화국 구간에 대해 시험운영을 시작했다. 이후 대중교통체계 개편의 일환으로 2003년 4월부터 '버스종합사령실' 설치공사를 시작하여 2005년 6월 서울시 소속 7,575대의 시내버스에 버스 차내 장치를 설치하고 BMS를 운영 중에 있다. 이러한 추세는 현재까지 부천시, 안양시를 비롯해 전국적으로 확산되고 있다.

한편, 자동차 수의 증가 및 이에 따른 교통 체증의 심화에 따라 도로 네트워크의 효율성을 증대시키기 위한 대중교통 정보 서비스의 확대가 요구되고 있다. 예를 들어 사용자는 "현재 사용자 위치와 가장 근접한 정류장에서 시청으로 가기 위해 이용 가능한 버스 노선들의 정보와 각 노선의 버스가 정류장에 도착할 예상 시간을 알려 달라"와 같은 서비스를 요청할 수 있으며, 이 경우 시스템은 "100번, 200번 버스 노선을 신촌역 앞 정류장에서 이용할 수 있으며, 각각 7분, 9분 뒤 정류장에 도착할 예정이다."를 결과로 제공하게 된다. 이러한 요구에 따라 대중교통 서비스에 대한 연구들이 최근 활발히 진행되고 있다[10-16]. 그러나 대중교통을 위한 기존의 모바일 교통 정보 시스템은 다음과 같은 한계를 지닌다.

첫째, 기존의 연구[10-13,15]에서는 모바일 기기의 무선 통신 비용 및 서버 측 부하에 대한 고려가 미흡하다. 기존의 방식은 모든 처리를 서버 측에 맡기고 모바일 기기는 질의 전송 및 결과 표시의 역할만 담당하였다. 따라서 모바일 기기 단독으로는 교통 정보 서비스의 이용이 불가능하며, 서비스를 요청할 때마다 서버와의 무선 통신 비용을 지불해야 한다. 현재까지 무선 통신 비용이 비교적 높다는 점과 모바일 서비스 이용자의 수가 더욱 늘어날 것이라는 점을 고려한다면, 낮은 통신비용의 확장성 있는(scalable) 모바일 교통 정보 시스템의 구축이 요구된다.

둘째, 실시간 버스 위치 정보를 고려한 라우팅 알고리즘이 발표된 바가 거의 없다. 기존의 최단 거리 알고리즘(Shortest Path Algorithm)과는 달리, 버스를 대상으로 하는 라우팅 알고리즘은 버스 노선도뿐만 아니라 유동적인 속도로 노선을 이동하고 있는 버스의 위치 정보를 추가로 사용해야 한다. 이에 대한 결과 값은 정류장까지의 이동 시간, 버스를 탈 때까지의 대기 시간, 총 이동 거리, 갈아타는 회수 등의 요소를 종합적으로 고려한 노선 정보가 되어야 한다. 그러나 기존의 라우팅 알고리즘에 대한 연구는 실시간 버스 위치 정보를 고려하지 않고 단지 버스의 정류장간 이동 시간에 대한 통계 수치만을 이용[10,11,13,15]하거나, 대중교통이 아닌 자가 차량을 위한 이동 경로[1,7-9]를 대상으로 하였다. 따라서 실시간 버스 위치 정보를 고려한 라우팅 알고리즘에 대한 연구가 필요하다.

셋째, 미래 시점 질의를 지원하지 않는다. 대부분의 경우, 질의를 던진 사용자가 항상 질의를 던진 시점에 바로 버스를 이용하고자 하지는 않을 것이다. 사용자는 특정 시간 뒤에 버스를 타겠다고 계획하고, 그 시점에 가장 적절한 노선이 무엇인지를 현재 시점에서 확인할 수 있는 형태인 미래 시점 질의(query for the future)를 선호할 것이다. 그러나 통계 수치를 기반으로 한 라우팅 알고리즘[13,15]은 시간대 별로 정류장 간의 이동 시간이 달라질 수 있다는 점을 고려하지 않아 미래 시점 질의에 부적합하며, 실시간 버스 정보를 입력으로 받는 MyBus[12]와 BusView[16]은 버스의 현재 위치 정보만을 보여줄 뿐, 미래 시점의 버스 위치에 대한 예측값은 제공하지 않는다. 따라서 미래 시점 질의가 가능한 교통 정보 서비스에 대한 연구가 필요하다.

본 논문에서는 이러한 문제점들을 해결하는 새로운 대중교통 정보 시스템 ACE-BIS(A Cost-Effective Bus Information System)을 제안한다. ACE-BIS의 주요 특징은 다음과 같다.

(1) 사용자는 모바일 기기를 사용하여 목적지에 도착할 수 있는 버스 노선 정보를 쉽게 제공받을 수 있다.
(2) 경로 탐색에 필요한 기본 정보로서 정류장 및 노선 데이타를 모바일 기기에 유지하고, 경로 탐색을 위한 휴리스틱 알고리즘을 이용하여 노선 정보, 목적지까지의 예상 소요 시간을 제공한다. 이를 통해 서버의 부담을 줄임과 동시에 최소한의 통신비용만으로 서비스를 제공할 수 있다.
(3) 서버 내에는 GPS를 통해 수집한 버스의 현재 위치 및 도로 체증 상황 등의 실시간 교통 정보를 추가적으로 관리한다. 통신비용을 부담하는 사용자에게는 이러한 실시간 교통 정보를 이용하여 해당 시점에서 더 나은 경로 정보를 제공할 수 있다.
(4) 서버 내에서 관리되는 실시간 교통 정보를 이용하여 미래 시점 질의를 처리할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로서 도로 네트워크 환경에 적합한 라우팅 알고리즘과 기존의 모바일 교통 정보 시스템에 대해 소개한다. 3장에서는 본 논문에서 제안하는 시스템의 전체적인 구조, 사용자 인터페이스, 데이타 저장 구조를 기술하고, 4장에서는 효율적인 라우팅 알고리즘을 제안한다. 5장에서는 제안된 시스템의 정확성 및 효율성 검증을 위한 성

능 분석 결과를 제시한다. 6장에서는 본 논문의 결론을 내리고, 향후 연구 방향을 제시한다.

2. 관련 연구

도로 네크워크 환경에 적합한 라우팅 알고리즘으로 Fast Lee 알고리즘[9]이 있다. 이 알고리즘은 도로의 정체(congestion) 정도를 도로의 최대 속도에 대한 현재의 소통 속도의 비로 정의하고, 이를 이용해 최단 시간에 목적지에 도달할 수 있는 이동 경로를 계산한다. 그러나 Fast Lee 알고리즘은 대중교통이 아닌 자가 차량에 대해서만 적용 가능하다는 점에서 본 논문에서 제안하는 알고리즘과는 그 목적이 다르다.

Datar와 Ranade는 통계 수치에 기반 한 라우팅 알고리즘을 제안하였다[15]. 이 알고리즘에서는 버스의 정류장간 이동 시간을 포아송 분포(Poisson distribution)로 가정하여 버스의 도착 시간을 예측하고, 이용할 버스 노선을 사용자에게 제공한다. 그러나 버스의 이동시간이 일정한 평균값을 갖는 포아송 분포를 따른다는 가정은 공학적 근사일 뿐, 실제적인 근거가 없다는 한계가 있다[13,15]. 이러한 문제를 해결하기 위해, Boyan과 Mitzenmacher는 버스의 정류장간 이동시간의 분포를 포아송 분포로부터 균등 분포(uniform distribution), 정규 분포(normal distribution), 감마 분포(Gamma distribution)까지로 확대하였다[13]. 그러나 Boyan과 Mitzenmacher의 방식은 알고리즘의 계산 비용이 지나치게 높다는 문제가 있다. 또한, 두 가지 방법 모두 실시간으로 수집되는 버스의 실제 위치 정보를 라우팅 과정에 반영할 수 있는 방법은 제시하지 않았다.

모바일 교통 정보 시스템 Bus Catcher[10,11]는 목적지에 도착할 수 있는 버스 노선 정보를 '버스 정류장까지의 도보 이동 시간이 짧은 순' 같은 미리 입력한 사용자의 선호에 따라 정렬하여 제공한다. 그러나 Bus Catcher에서는 알고리즘의 수행이 전적으로 서버에서 이루어져 무선 통신 비용을 지불하지 않고는 서비스를 이용할 수 없으며, 이에 따라 사용자 수에 비례하여 서버의 부하가 증가한다는 단점이 있다. 또한, 갈아타기를 고려하지 않으며 미래 시점 질의는 처리할 수 없다는 단점이 있다.

MyBus[12]와 BusView[16]는 버스의 위치를 실시간으로 보여주는 웹 애플리케이션이다. 특히, MyBus는 WAP으로도 구현되어 핸드폰 상에서도 정보를 제공한다. 그러나 이 시스템들은 라우팅 알고리즘이 구현되어 있지 않으므로 버스 노선 정보를 제공할 수 없다. 또한, 버스의 미래 위치를 예측하지 못하므로 미래 시점 질의도 지원할 수 없다.

"단기 통행시간예측 모형 개발에 관한 연구"[19]에서는 운전자에게 원하는 목적지까지 최적경로를 제공하거나 경로에 대한 통행시간 정보를 제공 또는 예측해 주는 시스템인 첨단여행자 정보체계의 통행시간 예측에 대한 새로운 모델을 제시하였다. 제안한 시스템은 실시간(real time) 정보가 아닌 예측된 교통정보를 제공하기 위해 확률과정 모형과 칼만 필터링 예측모형을 적용하였다. 하지만 이는 노선의 정보가 정해져 있는 대중교통에의 직접적인 적용이 불가능하고 통행 시간 예측에 있어 비싼 계산 비용이 들어 모바일 어플리케이션으로는 부적합하다.

"버스도착시간 정보에 대한 연구"[20]에서는 안내 도착예정시간의 함수로 버스도착을 안내된 도착예정시간보다 일찍 도착하는 조기도착과 늦게 도착하는 지연도착으로 구분하여 각각의 경우에 대한 대기시간의 합인 총 대기시간을 함수화하였다. 하지만 이 연구에서는 통계적 기법을 활용하여 버스의 도착예상시간을 정확하게 제공하는데 초점을 두고 있으며 버스 노선에 대한 정보는 고려하지 않았다. 또한 제안한 시스템이 복잡한 통계적 계산을 요구하기 때문에 모바일 어플리케이션에 적용되기에는 무리가 있다.

3. 시스템 개요

본 장에서는 논문에서 제안하는 모바일 대중교통 정보 시스템 ACE-BIS의 구조, 사용자 인터페이스, 데이타 저장 구조에 대해 기술한다.

3.1 시스템 구조

ACE-BIS는 크게 모바일 기기와 서버로 구성되며 모바일 기기에는 모바일 데이타베이스가, 서버에는 서버 데이타베이스가 구축되어 있다. 그림 1은 ACE-BIS의 개략적인 구조와 질의 처리에 수반되는 데이타의 흐름을 나타낸다.

그림 1에서 점선으로 표시된 부분은 무선 통신 비용이 필요한 작업을 나타내며, 실선으로 표시된 부분은 모바일 기기 상에서 수행되어 무선 통신 비용이 필요 없는 작업을 나타낸다. 시스템의 데이타 흐름은 다음과 같다.

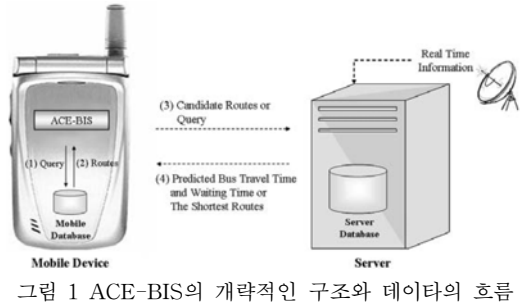

그림 1 ACE-BIS의 개략적인 구조와 데이타의 흐름

(1) 사용자가 '(목적 정류장, 갈아타기 최대 회수)'로 구성된 질의를 보낸다.
(2) 모바일 기기 상의 라우팅 알고리즘은 모바일 데이타베이스에 저장된 정보와 버스의 평균 이동 속도를 이용하여 목적지에 도달할 수 있는 복수 개의 이동 경로를 구한 후, 예상 소요 시간을 기준으로 정렬하여 사용자에게 보여준다.
(3) 만약 사용자가 특정한 경로에 대한 보다 정확한 소요 시간 정보를 요구하거나, 실시간 교통 정보를 반영한 라우팅 알고리즘을 수행하기를 원하거나 미래 시점에 대한 질의를 요청하면, 모바일 기기는 자신이 구한 특정 이동 경로 또는 질의의 내용을 서버로 전송한다.
(4) 서버에서는 특정 경로에 대한 정확한 소요 시간 정보나, 실시간 교통 정보를 반영한 라우팅 알고리즘의 실행 결과나, 미래 시점 질의에 대한 실행 결과를 모바일 기기로 전송한다.

만일 (1)에서 사용자가 모바일 기기내의 알고리즘이 아닌 실시간 교통 정보를 반영한 서버의 라우팅 알고리즘의 수행을 원한다면 (2)를 거치지 않고 (3)의 과정으로 바로 갈 수도 있다.

3.2 사용자 인터페이스

ACE-BIS의 사용자 인터페이스는 그림 2와 같다. 사용자가 최초 화면 (a)에서 목적지의 이름과 갈아타기의 최대 회수를 입력하고 'Route' 메뉴를 선택하면, 모바일 데이타베이스에 저장된 정보를 이용하여 라우팅 알고리즘을 수행한 후 그 결과를 화면 (b)에 보여준다. 만약 사용자가 경로에 대한 정확한 정보를 알고자 하면 'Predict' 메뉴를 선택하여 서버와의 통신을 수행한다. 'Predict' 메뉴는 화면 (c)에서 보는 것과 같이 두 개의 서브 메뉴로 구성되어 있다. 'Selected' 메뉴는 사용자가 원하는 경로 하나를 서버로 전송하여 정확한 시간 정보를 얻어 오는 것이고 'All' 메뉴는 사용자가 입력한 목적지의 이름과 갈아타기의 최대 회수를 서버에 전송하여 목적지로의 최단 시간 이동 경로를 얻어 오는 것이다.

서버와의 통신 후에는 그림 2(d)에 보인 것처럼, 정류장간 버스의 이동 시간(TT: Travel Time) 및 정류장에서 버스를 기다리는 시간(WT: Waiting Time)이 표시된다. 이 때, 화면 (d)에 보이는 경로 정보는 실시간 교통 정보를 반영한 결과이므로 화면 (b)의 경로 정보와 우선 순위가 다를 수 있다. 예를 들어, 화면 (b)에서는 7번 버스를 이용하여 Yonsei Univ.에서 Cine Theater로 이동하는 것을 첫 번째 우선순위로 추천하였으나, 화면 (d)에서는 실시간 교통 정보를 고려하여 Yonsei Univ. → City Hall → Cine Theater로 이동하는 경로를 첫 번째 우선순위로 추천하고 있음을 볼 수 있다. 또한, 사용자는 'FutureQuery' 메뉴를 선택하여 "30분 뒤 출발" 혹은 "14시 20분까지 도착"과 같은 미래 질의를 작성하는 화면 (e)로 이동할 수 있다. 화면 (f)는 미래 질의의 수행 결과이다.

3.3 데이타 저장 구조

3.3.1 모바일 기기의 데이타 저장 구조

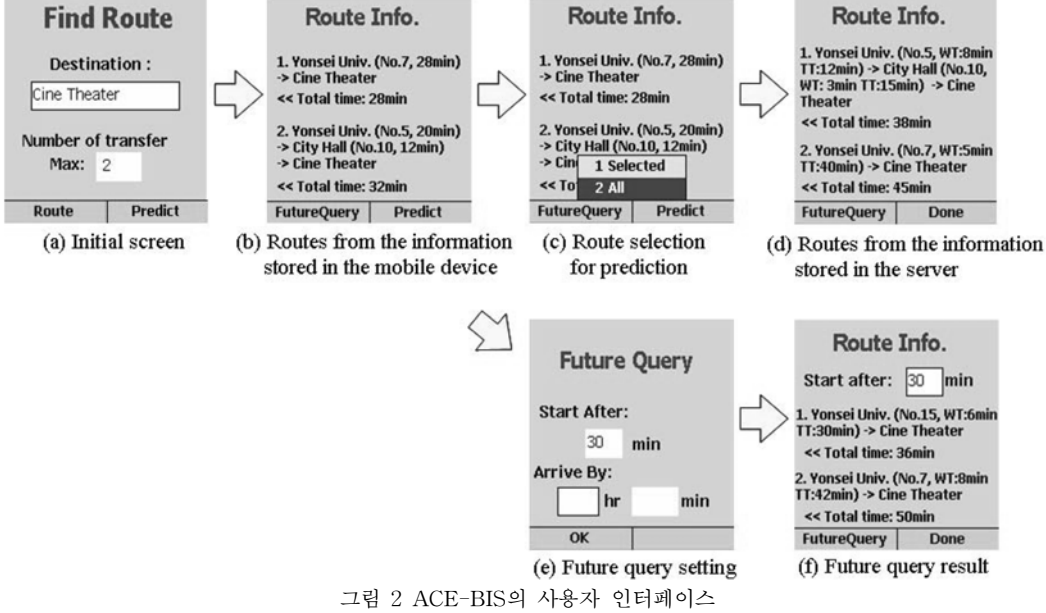

그림 2 ACE-BIS의 사용자 인터페이스

표 1 모바일 데이타베이스에 저장된 버스 노선도 및 정류장 테이블의 스키마

버스 노선도 테이블 (M_ROUTE)

컬 럼 명	의 미
ROUTE_NUM	노선 번호
SEQ	해당 정류장의 현재 노선내에서의 순서
SID	정류장 ID

정류장 테이블 (M_BUS_STOP)

컬 럼 명	의 미
SID	정류장 ID
LOC_X	정류장의 X 좌표
LOC_Y	정류장의 Y 좌표
SNAME	정류장 이름

ACE-BIS는 서버와의 무선 통신 없이도 목적지에 도달할 수 있는 이동 경로를 사용자에게 제공하기 위해 버스 노선과 정류장 정보를 테이블 형태로 모바일 데이타베이스에 저장한다. 표 1은 버스 노선도 테이블과 정류장 테이블에 대한 스키마를 보여준다. 여기서, 밑줄친 컬럼은 해당 테이블의 키를 나타낸다. 버스 노선도 테이블에는 각 버스 노선이 지나가는 정류장의 ID를 순서대로 유지하고 정류장 테이블에는 정류장의 이름과 위치를 저장한다. 정류장 테이블에 저장된 정류장 위치 정보를 이용하면 인접한 정류장 간의 거리를 계산할 수 있으며, 인접한 정류장 간의 거리와 버스의 평균 이동 속도를 이용하면 인접한 정류장 간의 이동 소요 시간을 예측할 수 있다.

3.3.2 서버의 데이타 저장 구조

ACE-BIS의 서버는 버스의 이동 상황을 실시간으로 수집하여 데이타베이스에 저장하며, 수집된 정보를 이용하여 정류장 간 버스의 이동 시간 및 사용자가 버스 정류장에서 기다려야 하는 시간을 예측한다. 또한, 사용자로부터 질의를 전송받아 서버 내의 라우팅 알고리즘을 실행하여 좀 더 정확한 경로 정보를 보여줄 수 있다. 서버의 데이타베이스에는 차량 테이블과 버스 노선도 테이블, 정류장 테이블을 저장한다. 서버에서 유지되는 정류장 테이블은 모바일 기기에서 유지되는 정류장 테이블과 동일한 스키마를 가지며, 차량 테이블 및 버스 노선도 테이블에 대한 스키마는 표 2와 같다.

차량 테이블은 일정한 간격으로 서버에 전달되는 버스의 실시간 위치 정보를 저장하기 위해서 사용된다. 즉, 버스의 실시간 위치 정보가 서버에 도착할 때 마다 차량 테이블에 새로운 튜플을 생성하여 컬럼 TIMESTAMP에는 수신 시각을, 컬럼 LOC_X와 컬럼 LOC_Y에는 버스의 현재 위치를, 컬럼 LAST_SID에는 가장 최근에 지나온 정류장의 ID를, 컬럼 TIME_FROM_LAST_BS에는 가장 최근 정류장을 출발한 뒤 경과한 시간을 저장한다.

표 2 서버 데이타베이스에 저장된 차량 테이블 및 버스 노선도 테이블의 스키마

차량 테이블 (S_BUS)

컬 럼 명	의 미
BUS_ID	버스 ID
ROUTE_NUM	노선 번호
TIMESTAMP	위치 정보의 수신 시각
LAST_SID	최근 지나온 정류장 ID
TIME_FROM_LAST_BS	최근 정류장을 출발한 뒤 경과한 시간
LOC_X	버스의 현재 X 좌표
LOC_Y	버스의 현재 Y 좌표

버스 노선도 테이블 (S_ROUTE)

컬 럼 명	의 미
ROUTE_NUM	노선 번호
SEQ	이 정류장의 현재 노선내에서의 순서
SID	정류장 ID
TIMESTAMP	PREDICTED_TT의 적용 시각
PREDICTED_TT	현재 노선의 다음 정류장까지의 예상 소요 시간

버스 노선도 테이블은 노선에 대한 기본적인 정보와 다음 정류장까지의 예상 소요 시간을 저장한다. 컬럼 TIMESTAMP은 다음 정류장까지의 예상 소요 시간인 PREDICTED_TT가 적용되는 시각을 나타내며 10분 단위로 구분하여 저장된다. 예를 들어, 5번 버스 노선의 첫 번째 정류장에서 두 번째 정류장까지의 예상 이동 시간이 [9:00, 9:10) 사이에는 7분, [9:10, 9:20) 사이에는 5분이라고 가정하면, 버스 노선도 테이블에는 두 개의 튜플(ROUTE_NUM, SEQ, SID, TIMESTAMP, PREDICTED_TT) = {(5, 1, STOP_1, 9:00, 7), (5, 1, STOP_1, 9:10, 5)}이 저장된다.

4. 질의 처리

본 장에서는 ACE-BIS의 질의 처리 알고리즘을 기술한다. 4.1절에서는 모바일 기기 상에서 수행되는 라우팅 알고리즘을 기술하고, 4.2절에서는 알고리즘에 사용되는 비용 함수에 대해 기술한다. 4.3절에서는 서버 상에서 수행되는 라우팅 알고리즘을 기술한다 4.4절에서는 정류장간 소요 시간을 예측하는 방법을 설명하고, 4.5절에서는 미래 질의를 처리하는 방법을 설명한다.

4.1 모바일 기기 내의 라우팅 알고리즘

ACE-BIS는 모바일 데이타베이스에 저장된 정보를 이용하여 목적지에 도달할 수 있는 경로를 제공한다. 이를 위해 버스 노선과 정류장 정보로부터 유향 다중 그래프(directed multi-graph)[1] G를 생성하고, 출발 노드

1) 유향 다중 그래프는 두 개의 노드 사이에 방향을 가지고 있는 에지가 하나 이상 존재하는 그래프를 의미한다[17].

와 목적 노드를 지정한 후, 라우팅 알고리즘을 호출한다. 유향 다중 그래프 G의 각 노드는 버스 정류장을 나타내며, 각 유향 에지는 버스의 정류장간의 이동을 의미한다. 각 노드에는 해당 정류장을 경유하는 모든 버스의 노선 번호가 저장되므로 두 노드간의 에지는 두 개 이상일 수 있다. 각 유향 에지에는 해당 노선을 운행하는 버스가 정류장간을 이동하는데 걸리는 시간이 기록된다.

알고리즘 1은 본 논문에서 제안하는 라우팅 알고리즘 Find_Routes를 나타낸다. 이 알고리즘은 유향 다중 그래프 G, 출발 정류장 S, 목적지 정류장 D, 최대 갈아타기 회수 MT를 입력으로 받아, S로부터 D에 도달할 수 있는 경로의 집합 R을 목적 정류장까지의 예상 소요 시간이 적은 순으로 정렬하여 사용자에게 전달한다. 이 알고리즘에서는 모바일 기기의 자원이 한정적이라는 것을 고려하여 검색 공간을 많이 사용하는 큐(queue) 보다는 비용이 적게 드는 노드를 먼저 방문함으로써 검색 공간을 최소화할 수 있는 우선순위 스택(priority stack)을 사용한다. 제안된 알고리즘에서는 모든 경로를 탐색하지 않고도 빠른 시간 내에 원하는 경로를 얻기 위해서 정답이 될 가능성이 낮은 경로들은 사전에 가지치기 하여 더 이상 탐색 대상에 포함시키지 않는다. 즉, 우선순위 스택을 사용하여 최적 우선(best first) 방식으로 그래프를 탐색함으로써 노드의 방문 여부를 결정하는 조건을 조기에 강화시켜 검색 공간을 줄이고 알고리즘의 수행 속도를 높이도록 한다.

우선순위 스택에는 위에 있는 노드의 비용이 아래에 있는 노드의 비용보다 언제나 작거나 같아야 한다는 제약을 부여한다. 우선순위 스택의 pop 연산은 기존 스택의 pop 연산과 동일하게 스택의 가장 위에 있는 노드를 꺼내서 반환한다. 반면, 우선순위 스택의 push 연산은 그 노드의 비용에 따라 스택의 적절한 위치에 노드를 삽입한다.

Find_Routes는 S를 출발하여 최적 우선 방식으로 유향 다중 그래프 G의 노드들을 방문한다. 현재 방문 중인 노드를 V라고 하자. 알고리즘은 먼저 V가 목적 정

알고리즘 1 모바일 기기 내의 라우팅 알고리즘 Find_Routes

```
Algorithm   Find_Routes
INPUT:      Directed multi-graph G, Start node S, Destination node D,
            Maximum number of transfers MT
OUTPUT:     Set of routes R

1:          SMALLEST_COST = ∞;
2:
3:          priority_stack.push(S);
4:          while(priority_stack.not_empty()) {
5:              V := priority_stack.pop();
6:              if (V is D) {
7:                  R.add(V.current_route);
8:                  continue;
9:              }
10:             ADJ := V.get_adjacent_nodes(G);
11:             foreach V' in ADJ {
12:                 V'.compute_num_transfers(V);
13:                 if (V'.num_transfers > MT) continue;
14:                 EC := V'.estimate_cost(V, D);
15:                 if (EC > BETA * SMALLEST_COST) continue;
16:                 else if (EC < SMALLEST_COST)
17:                     SMALLEST_COST := EC;
18:                 V'.compute_angle_of_vectors(V, S, D);
19:                 priority_stack.push(V');
20:             }
21:         }
22:         R.sort();
23:         Return R;
24:     }
```

류장인가를 점검한다. 만약 V가 목적 정류장이 아니라면 V의 각 인접 노드 V′에 대하여 다음을 수행한다.

(1) V에서 V′으로 이동한다고 가정하고 현재 경로의 갈아타기 회수를 계산한다. 만약 갈아타기 회수가 MT를 초과하는 경우에는 우선순위 스택으로부터 다음 노드를 pop해서 탐색을 계속한다.

(2) V에서 V′으로 이동한다고 가정하고 현재 경로의 비용을 예측한다. 비용을 예측하는 함수는 다음 절에서 설명한다. 만약, 예측된 비용이 BETA * SMALLEST_COST 보다 큰 경우에는 우선순위 스택으로부터 다음 노드를 pop해서 탐색을 계속한다. 이는 예측된 비용이 BETA * SMALLEST_COST 보다 큰 경우에는 V′까지의 현 경로를 계속 확장해도 D까지 도달할 가능성이 희박하다는 휴리스틱을 반영하기 위한 것이다. SMALLEST_COST는 현재까지 발견된 경로들의 비용 중 가장 작은 값을 나타낸다. 따라서 예측된 비용이 SMALLEST_COST의 BETA 배 이하인 경우에만 V′까지의 경로를 계속 고려한다. BETA는 1보다 크거나 같은 실수형의 변수로서 알고리즘의 수행 속도와 정확도 간의 트레이드오프(tradeoff)를 고려하여 설정한다.

(3) 아래 그림 3에서와 같이 S에서 D로의 벡터와 V′에서 D로의 벡터 사이의 각도를 계산한다. 이 각도가 작다는 것은 V′이 목적지 방향의 정류장임을 의미한다.

(4) V′을 우선순위 스택에 push한다. 우선순위 스택에서는 예측된 비용이 적은 노드가 위에 위치한다. 만일 예측된 비용이 동일하다면 (3)에서 구한 벡터 사이의 각도가 작은 것이 위에 위치하게 한다. 이것은 노드의 예측 비용이 동일하다면 목적지 방향의 노드를 먼저 방문하는 것이 유리하다는 휴리스틱을 반영하기 위한 것이다.

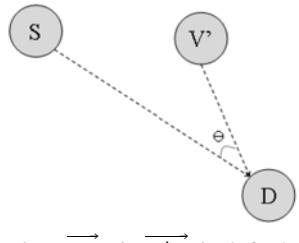

그림 3 $\overrightarrow{SD}$ 와 $\overrightarrow{V'D}$의 사이 각 θ

4.2 비용 예측 함수

노드 V에서 인접 노드 V′으로 이동한다고 가정하였을 때, 거리 및 갈아타기 여부를 이용하여 현재 경로의 비용을 예측하는 함수는 다음과 같다.

```
V'.current_cost(V, D) =
1:    V.current_cost +
2:    V.distance_to(V') / AVG_MOVING_SPEED +
3:    V.does_need_to_transfer(V') *
      AVG_TRANSFER_TIME;
```

```
V'.estimate_cost(V, D) =
1:    V'.current_cost +
2:    V'.distance_to(D) / AVG_MOVING_SPEED
```

V′.current_cost(V,D)의 1번째 줄은 출발 정류장으로부터 현재 경로를 따라 노드 V까지 도달하는데 걸린 시간을 나타낸다. 2번째 줄은 노드 V에서 노드 V′까지의 거리를 버스의 평균 속도로 나누는 수식으로서 V에서 V′으로 이동하는데 걸리는 시간을 예측한다. 버스의 평균 속도는 시간대 별로 다른 값을 사용하며, 이 값들은 매일 또는 매주 서버에 접속하여 갱신하게 된다. 3번째 줄은 만일 V′이 갈아타는 정류장일 경우 갈아타는 데 걸리는 평균 시간을 전체 비용에 더해주는 것이다. V′.estimate_cost(V,D)는 노드 V′의 현재 비용에 노드 V′에서 목적 정류장 D까지의 직선거리를 버스의 평균 속도로 나눈 값인 V′에서 D로 이동하는데 걸리는 최소 시간을 더하여 전체 경로에 대한 비용을 예측한다.

4.3 서버의 라우팅 알고리즘

사용자의 요구가 있을 경우 ACE-BIS는 두 가지 유형의 질의 중 하나를 서버에 전송한다. 첫 번째 유형은 모바일 기기에서 구한 경로를 선택해 그 경로에 대한 보다 정확한 정보를 서버에 요청하는 것이다. 서버는 차량 테이블에 저장된 버스의 실시간 위치 정보 및 버스 노선도 테이블에 저장된 정류장간 버스의 예상 소요 시간 정보를 이용하여 사용자가 정류장에서 버스를 기다려야 하는 시간, 버스의 정류장 간 이동 시간, 목적지까지의 총 소요 시간 등을 계산한다.

두 번째 유형은 시작 정류장 이름, 목적 정류장 이름, 갈아타기 최대 회수로 구성된 사용자 질의를 서버에 전송하는 것이다. 서버는 데이타베이스에 저장된 버스 노선과 정류장 정보로부터 유향 다중 그래프 G를 생성한 후 4.1에서 설명한 라우팅 알고리즘과 동일한 알고리즘을 수행하여 목적지까지의 경로를 탐색한다. 이 때 비용 함수는 버스의 정류장간 이동 시간과 갈아타는 정류장에서 버스를 기다려야 하는 시간의 합으로 정의된다. 이러한 정보는 최근에 수집된 실시간 교통 정보라는 측면에서 모바일 기기의 정보와는 차이가 있다.

서버에서 수행되는 알고리즘과 모바일 기기에서 수행되는 알고리즘은 비용 함수를 계산하는 부분을 제외하고는 동일하다. 하지만 일반적으로 서버가 모바일 기기보다 계산 능력이 월등히 우수하기 때문에 모바일 기기

에서 적용한 휴리스틱의 조건을 서버 측 알고리즘에서는 완화함으로써(즉, BETA의 값을 크게 함) 질의의 결과로 나온 경로의 정확성을 높일 수 있다.

4.4 정류장간 이동 시간의 예측

정류장간 버스의 이동 시간은 도로 상황이나 계절, 요일, 시간대 등 다양한 요인에 의해 영향을 받는다. 따라서 본 논문에서는 히스토리 데이타 기반 방법(history data-based approach)의 하나인 지수 이동 평균(exponential moving average)[18]을 사용하여 버스의 정류장간 이동 시간을 예측한다. 정류장간 이동 시간에 대한 n번째 예측 값과 이에 대한 실제 측정값을 각각 P_n과 T_n이라고 하면, n+1번째 예측 값 P_{n+1}은 다음과 같이 계산된다.

$$P_0 = T_0 \quad (1)$$

$$P_{n+1} = \alpha T_n + (1-\alpha) P_n \quad (n \geq 0,\ 0 \leq \alpha \leq 1) \quad (2)$$

위의 식에서 α는 n번째 실제 측정값이 n+1번째 예측값에 얼마나 영향을 주는가를 결정하는 매개 변수로서, α가 클수록 다음 예측 값(P_{n+1})은 지금까지의 히스토리(P_n)보다 현재의 실제 측정값(T_n)을 더 많이 반영하게 된다. 본 논문에서는 이동 시간에 대한 예측이 10분 단위로 이루어진다고 가정한다. 위의 지수 이동 평균을 이용하여 계산된 다음 예측 값(P_{n+1})은 3장에서 언급한 버스 노선도 테이블 S_ROUTE의 컬럼 PRETICTED_TT에 반영된다. 또한, 버스 노선도 테이블의 컬럼 PREDICTED_TT 값과 차량 테이블의 컬럼 TIME_FROM_LAST_BS 값의 차를 계산하면 해당 정류장에서 버스를 기다리는 시간을 예측할 수 있다.

4.5 미래 질의의 처리

미래 질의는 크게 두 가지 유형으로 나뉜다. 첫 번째 유형은 사용자가 목적지를 향해 특정한 시간 후에 출발하는 경우이며, 두 번째 유형은 지정된 시각까지 사용자가 목적지에 도착해야 하는 경우이다. 이러한 두 가지 유형의 질의를 처리하기 위해서 서버는 버스 테이블에 있는 최근 정류장을 출발한 뒤 경과한 시간과 버스 노선도 테이블에 있는 시간대별 정류장간 예상 소요 시간 정보를 이용해야한다.

이를 위해 다음의 표 3과 같이 현재 경로의 비용 예측 함수를 재정의 한다.

여기서 TIMESTAMP는 V에서 V′까지의 소요 시간 계산 시 언제의 시점을 적용할 것인지를 나타낸다. 다시 말해, 만약 출발시각이 T이고 V까지 왔을 때의 시각이 T′이라면, TIMESTAMP는 T′값을 갖는다. 따라서 이 함수는 4.2 절에 보인 비용 예측 함수와 달리, 노드 V로부터 V′까지의 소요 시간을 계산함에 있어, S_ROUTE 테이블로부터 TIMESTAMP값이 일치하는 PREDICTED_TT를 사용한다.

표 3 미래 질의 처리를 위한 비용 예측 함수

Input:	Current Node V, Destination node D, TIMESTAMP
Output:	Estimate_cost
1	V′.current_cost =
2	V.current_cost + PREDICTED_TT from
3	S_ROUTE bet. V and V′
4	TIMESTAMP mathes
5	V′.estimate_cost =
6	V′.current_cost +
7	V′.distance_to(D) / AVG_MOVING_SPEED + V.does_need_to_transfer(V′) * AVG_TRANSFER_TIME
8	Return V′.estimate_cost

다음에 보인 Future_Find_Routes는 미래 질의 처리를 위한 기본 알고리즘이다

알고리즘 2 미래 질의 처리 알고리즘 (Future_Find_Routes)

Input:	Directed multi-graph G, Start Node S, Destination node D, Maximum number of transfers MT, Time to Leave T
Output:	A route and expected time of arrival
1:	*R* := Find_Routes(*G*, *S*, *D*, *MT*) with redefined estimate_cost method
2	For each route *r* in *R*
3	Find out the nearest bus, *b*, of *r* at *T* using S_ROUTE and S_BUS
4	WT := Estimated the time for the *b* to arrive *S*
5	put a pair *r* and its total time, *WT* +cost of *r,* into *Candidates*
6	End For
7	Sort Candidates based on the total time of each route
8	Return the route whose total time is the minimum

이 알고리즘은 다음과 같이 수행된다. 먼저 Find_Routes를 앞서 보인 estimate_cost 함수를 사용해 수행하되, 출발시간 T에 출발지로부터 V까지 이동하는데 걸린 시간을 더해 estimate_cost 함수에 추가적으로 전달한다(행 1). 다음, 결과로 주어진 각 경로 r을 지나는 버스 중 시각 T에 출발 정류장 S 근처에 위치할 버스 b를 찾아 이 버스가 출발 정류장에 도착할 때까지 걸리는 시간 WT를 계산한다(행 3-4). 다음, 경로 r의 cost와

WT를 더해 총소요시간을 계산하고 (r, 총소요시간)을 Candidates에 저장한다(행 5). 최종적으로 Candidates를 정렬하여 총소요시간이 가장 작은 경로를 반환한다 (행 7-8).

Future_Find_Routes를 사용한 미래 질의 처리는 다음과 같이 수행된다. 예를 들어, 첫 번째 유형의 질의인 "15분 후에 집을 나서서 버스를 타고 시청에 가려면 어떤 경로가 최적인가?"라는 질의를 처리하기 위해서는 출발 시각 T를 현재 시간에서 15분 뒤로 설정한 뒤 알고리즘을 수행한다.

두 번째 유형의 미래 질의도 위와 유사한 방식으로 처리할 수 있다. 예를 들어, "1:00까지 시청에 도착하려면 언제까지 출발해야 하는 가?" 라는 질의는, 현재 시각부터 더 이상 질의 시각을 만족하는 버스 노선의 결과가 나오지 않을 때까지 T를 증가시킴으로써 처리할 수 있다.

5. 실 험

본 장에서는 ACE-BIS의 효율성 및 정확성 검증을 위한 실험 결과를 기술한다. 5.1절에서는 시스템 환경, 데이타 및 질의 생성 방법, 실험 결과의 비교 척도 등을 설명한다. 5.2절에서는 ACE-BIS의 효율성 및 정확성을 결정하는 시스템 파라미터인 BETA 값을 실험을 통하여 결정하고, 5.3절에서는 결정된 BETA 값을 사용해 ACE-BIS의 효율성 및 정확성을 검증한다.

5.1 실험 환경

5.1.1 시스템 환경

본 장의 실험은 Pentium Mobile Centrino CPU 1.0 GHz, RAM 512M를 장착한 PC에서 수행되었다. 알고리즘의 구현에는 JDK 1.4.2가 사용되었으며, 데이타베이스로는 MySQL 4.1.9를 사용하였다.

5.1.2 데이타 생성 방법

ACE-BIS의 정확한 검증을 위해서는 버스 정류장의 위치와 노선 정보에 대한 실 데이타를 사용하는 것이 가장 이상적이다. 그러나 현실적으로 이러한 데이타를 수집하는 데에는 많은 어려움이 따른다. 따라서 본 논문에서는 다음에서 설명하는 방법으로 실세계의 특성을 반영한 가상 데이타를 생성하여 실험에 사용하였다.

버스 정류장 데이타의 생성

실험에서 사용할 버스 정류장의 총 개수가 N이고, 정류장간 간격이 대체적으로 일정하다고 가정하자. 이를 위해, 눈금간의 간격이 100m이고 가로와 세로에 각각 $\sqrt{N}$ 개씩의 격자점이 있는 모눈을 작성한다.[2] 다음으로, 각각의 격자점 위에 버스 정류장 $b_{i,j}$를 위치시킨다 (단, $1 \leq i,j \leq \sqrt{N}$). 마지막으로, 정류장간 거리에 변화를 주기 위해 그림 4에서와 같이 각 격자점을 중심으로 반지름이 25m인 원 안에서 해당 정류장의 위치를 랜덤하게 이동시킨다.

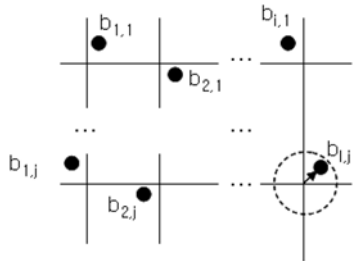

그림 4 정류장 위치의 결정(검은색 원은 격자점을 중심으로 반지름이 25m인 원 안에서 결정된 버스 정류장의 위치를 의미함)

버스 노선 데이타의 생성

먼저, 버스 정류장 중 임의로 선택한 두 정류장을 각각 노선의 시작 정류장과 종착 정류장으로 지정한다. 이때, 버스 노선의 길이가 너무 짧다면 현실성이 떨어질 것이다. 따라서 모눈의 눈금으로 볼 때 가로와 세로가 각각 $\sqrt{N}/2$ 이상 떨어진 위치에서 노선의 시작 정류장과 종착 정류장이 선택되도록 한다. 즉, 아래의 식 (3)과 4를 모두 만족시키는 범위 내에서 시작 정류장 $b_{sx,sy}$와 종착 정류장 $b_{ex,ey}$의 위치를 결정한다.

$$ex \leq sx - \frac{\sqrt{N}}{2} \quad or \quad ex \geq sx + \frac{\sqrt{N}}{2} \tag{3}$$

$$ey \leq sy - \frac{\sqrt{N}}{2} \quad or \quad ey \geq sy + \frac{\sqrt{N}}{2} \tag{4}$$

일단 노선의 시작과 종착 정류장이 정해지고 나면, 노선이 지나가는 중간 정류장들을 결정해야 한다. 이를 위해, 먼저 그림 5와 같이 시작 정류장 $b_{sx,sy}$에 인접한 총 8개의 정류장 $b_{sx\pm1,sy}, b_{sx,sy\pm1}, b_{sx\pm1,sy\pm1}$ 중 하나를 첫 번째 중간 정류장으로 지정한다. 다음으로, 첫 번째 중간 정류장 $b_{i,j}$에 인접한 총 8개의 정류장 $b_{i\pm1,j}$, $b_{i,j\pm1}$, $b_{i\pm1,j\pm1}$ 중 하나를 선택하여 두 번째 중간 정류장으로 지정한다. 단, 이미 지나온 중간 정류장은 또 다시 중간 정류장으로 선택되지 않도록 한다. 종착 정류장에 도달할 때까지 이러한 과정을 반복함으로써 중간 정류장들을 결정할 수 있다.

실세계의 버스 노선은 노선의 시작 정류장으로부터 노선의 종착 정류장으로 향하는 일정한 방향성을 갖는다. 따라서 중간 정류장 $b_{i,j}$에 인접한 8개의 정류장들이 다음 중간 정류장으로 선택될 확률을 모두 동일하게 설

2) 설명의 편의를 위해, 여기서는 $\sqrt{N}$ 이 정수라고 가정한다.

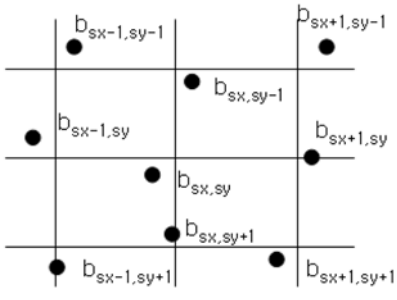

그림 5 노선의 시작 정류장 $b_{sx,sy}$에 인접한 8개의 정류장

정하는 것은 적절하지 않다. 예를 들어, 그림 6에서 V가 현재 선택된 중간 정류장, D가 현재 생성 중인 노선의 종착 정류장이라고 하자. 이 때, 버스 정류장 1, 4, 6, 7, 8 중 하나가 다음 중간 정류장으로 선택된다면 노선의 방향이 종착 정류장으로부터 멀어지게 된다. 즉, 버스 노선의 방향성을 고려한다면 2, 3, 5중 하나가 다음 중간 정류장으로 선택될 확률을 1, 4, 6, 7, 8 중 하나가 다음 중간 정류장으로 선택될 확률보다 높게 설정해야 한다. 본 실험에서는 2, 3, 5 중 하나가 선택될 확률을 70%, 1, 4, 6, 7, 8 중 하나가 선택될 확률은 30%로 설정하여 다음 중간 정류장들을 결정하였다.

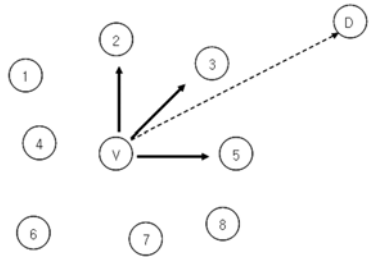

그림 6 노선의 방향성을 고려한 인접 정류장의 선택

질의 데이타의 생성

ACE-BIS에 대한 사용자의 질의는 출발 정류장 및 목적 정류장, 갈아타기 최대 회수로 구성된다. 갈아타기 최대 회수는 실험 목적에 따라 0, 1, 2 중에서 하나를 선택하며, 출발 정류장 및 목적 정류장은 다음과 같은 방식으로 지정한다. 먼저 N개의 버스 정류장 중 임의의 하나를 출발 정류장으로 선택한다. 다음으로 출발 정류장으로부터 눈금의 개수가 최소 $\sqrt{N}/3$, 최대 $\sqrt{N}/2$ 만큼 떨어진 정류장 중 임의의 하나를 선택하여 목적 정류장으로 선택한다.

5.1.3 실험 결과의 비교 척도

실험 결과의 비교 척도로서 결과 집합의 크기, 질의 처리 시간, 결과 집합의 정확도 등을 이용한다. 결과 집합의 크기란 질의를 수행한 후 시스템이 제시하는 경로의 수를 나타내며, 질의 처리 시간이란 질의를 시스템에 전달하고 나서 결과를 받을 때까지 소요되는 시간을 의미한다. 본 논문에서 제시하는 라우팅 알고리즘은 정답이 될 가능성이 낮은 경로들을 사전에 가지치기 하여 검색 공간을 줄이고 알고리즘의 수행 속도를 높이는 방식을 사용하고 있다. 이 과정에서 답이 될 수 있는 경로를 놓치는 경우가 발생할 수 있으므로 결과 집합의 정확도라는 척도를 사용하여 가지치기의 정확성을 평가한다.

가지치기를 수행하여 얻은 결과 집합의 정확도를 측정하기 위해서는 가지치기 없이 그래프를 탐색해 나가는 완전 탐색(exhaustive search)의 결과 집합과의 비교가 필요하다. PathSetE와 PathSetH를 각각 완전 탐색의 결과 집합과 가지치기를 수행하여 얻은 결과 집합으로 표기하자. PathSetH의 정확도를 정의하기 위해서 먼저 PathSetE에 포함된 각 경로 P_i의 원 점수 RawScore(P_i) 및 정규화 점수 NormScore(P_i)를 다음과 같이 계산한다.

$$\mathrm{RawScore}(P_i) = \frac{\mathrm{TimeSum} - T_i}{\mathrm{TimeSum}} \quad (5)$$

$$\mathrm{NormScore}(P_i) = \frac{\mathrm{RawScore}(P_i)}{|\mathrm{PathSetE}| - 1} \times 100 \quad (6)$$

위의 식 (5)와 식 (6)에서 TimeSum은 PathSetE 내의 모든 경로의 예상 소요 시간의 합, T_i는 경로 P_i의 예상 소요 시간, |PathSetE|은 PathSetE에 포함된 경로의 수를 표현한다. 식 (5)를 살펴보면, 원 점수는 0 이상 1 미만의 값을 가진다는 것과 해당 경로의 예상 소요 시간이 적을수록 높은 원 점수를 갖는다는 것을 알 수 있다. 또한, PathSetE 내의 모든 경로의 원 점수를 합하면 |PathSet|−1이 된다는 것을 알 수 있다. 식 (6)은 PathSetE 내의 모든 경로의 점수 합이 100이 되도록 원 점수를 정규화한 것을 나타낸다. PathSetH의 정확도는 아래 식 (7)과 같이 PathSetH에 속하는 각 경로 P_i가 PathSetE에서 가지는 정규 점수를 모두 합한 값으로 표현된다. 만약 PathSetH와 PathSetE가 동일한 원소로 구성되어 있다면 PathSetH의 정확도는 100%가 되며 PathSetE의 원소가 PathSetH에 하나도 포함되어 있지 않다면 정확도는 0%가 된다.

$$\mathrm{Accuracy}(\mathrm{PathSetH}) = \sum_{P_i \in \mathrm{PathSetH}} \mathrm{NormScore}(P_i) \quad (7)$$

5.2 BETA 결정 실험

본 논문에서 제안하는 라우팅 알고리즘에서 BETA는 가지치기 되는 경로의 개수를 조정하는 역할을 수행한다. BETA의 값이 작을수록 가지치기 되는 경로의 수가 증가하며, BETA의 값이 클수록 가지치기 되는 경로의 수가 감소한다. 따라서 어떤 값이 BETA에 할당되느냐

에 따라 알고리즘의 효율성 및 정확성이 많이 달라진다.

그러므로 ACE-BIS의 성능을 평가하기 전에, 본 절에서는 실험을 통하여 효율성 및 정확성을 결정하는 시스템 변수인 BETA의 값을 결정한다. 이 실험을 위해 우선 5.1절에서 기술한 방식으로 200개의 버스 정류장과 100개의 버스 노선을 생성하였다. 또한 최대 갈아타기 회수 MT가 0인 100개의 질의, 1인 100개의 질의, 2인 100개의 질의를 생성하였다. 다음으로 BETA 값을 1에서 10까지 순차적으로 증가시키면서 본 논문에서 제안하는 라우팅 알고리즘을 수행한 후, 최대 갈아타기 회수와 BETA 값의 각 쌍에 대해서 결과 집합의 크기, 질의 처리 시간, 결과 집합의 정확도를 측정한 후 각각의 평균값을 계산하였다.

결과 집합의 크기

그림 7은 0에서 2까지의 각각의 최대 갈아타기 회수 MT에 대하여 BETA 값의 증가에 따른 결과 집합의 평균 크기를 보여준다. 예상했던 대로 BETA의 값이 커질수록 결과 집합의 크기가 증가함을 볼 수 있다. 그림을 좀 더 자세히 살펴보면, BETA 값이 1에서 3 사이인 경우에는 결과 집합의 크기가 1이고 BETA 값이 4에서 6 사이인 경우에는 결과 집합의 크기가 약 15임을 알 수 있다. 질의 수행의 결과가 하나만 주어진다면 사용자 입장에서는 제공된 정보가 불충분하다고 느낄 가능성이 높다. 따라서 결과 집합의 크기만을 고려한다면 BETA 값을 4 이상으로 설정해야 한다는 결론을 얻을 수 있다.

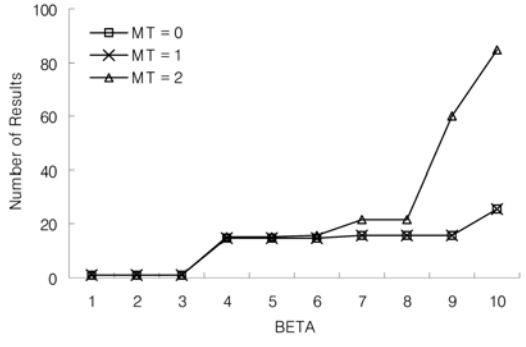

그림 7 BETA 값의 증가에 따른 결과 집합의 평균 크기 변화

질의 처리 시간

그림 8은 0에서 2까지의 각각의 최대 갈아타기 회수 MT에 대하여 BETA 값의 증가에 따른 평균 질의 처리 시간을 보여준다. MT가 0인 경우와 1인 경우의 결과는 거의 차이가 없어 하나의 그래프로 표현하였다. 결과 집합의 크기에 대한 실험의 경우와 마찬가지로 BETA 값이 커질수록 질의 처리 시간이 증가함을 볼 수 있다. 그림을 좀 더 자세히 살펴보면, BETA가 1에서 5 사이인 경우에는 MT 값에 관계없이 질의 처리 시간이 서서히 증가하고 있으며 MT가 0이나 1인 경우와 MT가 2인 경우의 질의 처리 시간의 차이가 거의 없음을 볼 수 있다. 그러나 BETA 값이 6 이상이 되면, 질의 처리 시간의 증가가 현저해지며 MT가 0이나 1인 경우와 2인 경우의 차이가 점점 커짐을 알 수 있다. 전반적으로 BETA 값과 MT 값의 모두 조합에 대해서 질의 처리 시간은 1초 이내였으며, 특히 BETA 값이 5 이하일 경우에는 100ms 미만의 질의 처리 시간을 가지는 것으로 나타났다.

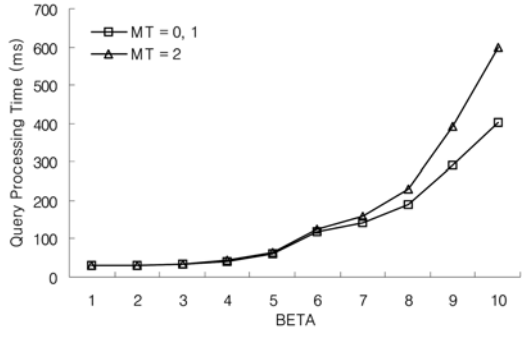

그림 8 BETA 값의 증가에 따른 평균 질의 처리 시간의 변화

결과 집합의 정확도

그림 9는 0에서 2까지의 각각의 최대 갈아타기 회수 MT에 대하여 BETA 값의 증가에 따른 결과 집합의 정확도를 보여준다. MT가 0인 경우와 1인 경우의 결과는 거의 차이가 없어 하나의 그래프로 표현하였다. 그림을 보면 BETA 값이 증가함에 따라 결과 경로의 정확도가 높아짐을 확인할 수 있다. 이는 BETA 값이 증가함에 따라 가지치기 조건이 완화되어 보다 많은 경로가 탐색되기 때문이다. 그림을 좀 더 자세히 살펴보면, BETA 값이 3에서 4로 변경될 때와 6에서 7로 변경될 때에 결과 집합의 정확도가 현저하게 증가함을 알 수 있다. 만약 MT에 관계없이 70% 이상의 정확도를 유지하려면 BETA 값을 4 이상으로 설정해야 한다는 결론을 얻을 수 있다.

BETA 값의 결정

위의 세 가지 실험 결과를 요약하면 다음과 같다. 첫째, 결과 집합의 크기만을 고려하면 MT에 관계없이 BETA 값을 4 이상으로 설정해야 한다. 둘째, 질의 처리 시간만을 고려하면 MT에 관계에 관계없이 BETA 값을 5 이하로 설정해야 한다. 셋째, MT에 관계없이 70% 이상

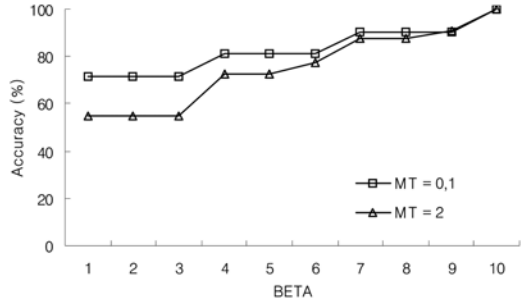

그림 9 BETA 값의 변화에 따른 결과 집합의 정확도 변화

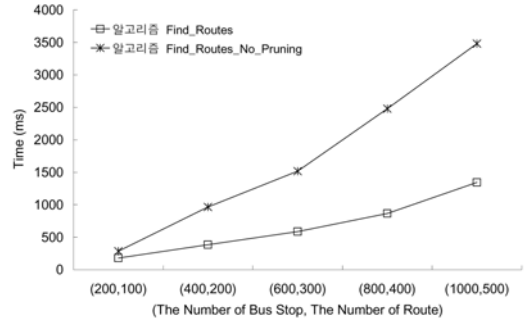

그림 10 데이타 크기 증가에 따른 질의 처리 시간의 변화

의 정확도를 유지하려면 BETA 값을 4 이상으로 설정해야 한다. 즉, BETA 값이 4 혹은 5인 경우에만 위의 세 가지 조건이 모두 만족된다는 결론에 도달하게 된다. 본 논문에서는 두 가지 값 중에서 4를 BETA 값을 선택하여 다음 실험을 진행하였다.

5.3 알고리즘의 효율성 및 정확성 실험

본 절에서는 결정된 BETA 값을 사용해 본 논문에서 제안한 라우팅 알고리즘의 효율성 및 정확성을 평가한다. 이 실험을 위해 버스 정류장의 수를 200에서 1000까지 차례로 증가시키면서 알고리즘 Find_Routes와 알고리즘 Find_Routes_No_Pruning의 질의 처리 시간 및 결과 집합의 정확도를 측정하여 비교하였다. 알고리즘 Find_Routes는 본 논문에서 제안한 라우팅 알고리즘을 나타내며 알고리즘 Find_Routes_No_Pruning은 알고리즘 Find_Routes에서 가지치기하는 부분(즉, 알고리즘 1의 15번째 라인에서 17번째 라인까지)을 제거한 버전을 나타낸다. 노선의 수는 버스 정류장 수의 1/2로 지정하였으며 갈아타기 회수 MT는 2로 고정하였다. 5.2절에서 기술한대로 BETA 값으로는 4를 사용하였다.

알고리즘의 효율성

그림 10은 데이타의 크기 증가에 따른 두 알고리즘의 질의 처리 시간 변화를 나타낸다. 그림을 보면 알고리즘 Find_Routes는 모든 경우에서 1.5초 이내의 빠른 응답 시간을 보인다는 것과 알고리즘 Find_Routes_No_Pruning에 비해서 최소 2배에서 최대 3배까지의 성능 향상을 보이는 것을 알 수 있다. 또한 데이타의 크기가 증가할수록 두 알고리즘의 질의 처리 시간의 차가 더욱 커짐을 알 수 있다. 이 실험을 통해 본 논문에서 제안하는 가지치기 기반의 라우팅 알고리즘 Find_Routes가 효율적이라는 것과 확장성이 높다는 것을 확인할 수 있었다.

알고리즘의 정확성

그림 11은 데이타의 크기 증가에 따른 두 알고리즘의 결과 집합의 정확도 변화를 나타낸다. 가지치기를 하지 않는 알고리즘 Find_Routes_No_Pruning의 경우에는

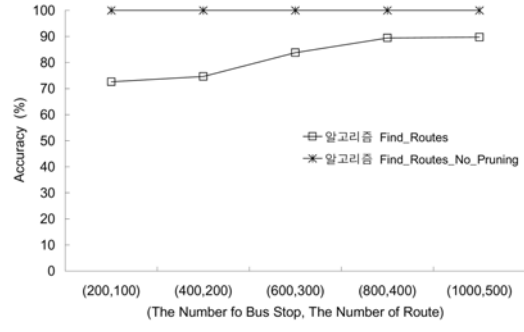

그림 11 데이타 크기 변화에 따른 결과 집합의 정확도 변화

가능한 모든 경로가 결과 집합에 포함되므로 데이타의 크기에 관계없이 언제나 100%의 정확도를 갖는다. 알고리즘 Find_Routes의 경우에는 최소 73%, 최대 90%, 평균 82%의 정확도를 보이며 이러한 정확도는 데이타의 크기 증가에 따라 소폭 상승하는 것으로 나타났다.

6. 결 론

본 논문에서는 고정된 노선을 이동하는 버스를 대상으로 목적지까지의 경로들을 최소한의 통신비용으로 제공할 수 있는 모바일 대중교통 정보 시스템 ACE-BIS를 제안하였다. 본 논문의 공헌은 다음과 같이 요약될 수 있다.

(1) 라우팅 알고리즘의 수행을 모바일 기기와 서버에 분산시킴으로서 서버의 부담과 통신 비용을 줄일 수 있는 시스템 구조를 제안하였다.

(2) 고정된 노선을 이동하는 버스의 특성을 고려하여 제안된 자원 하에서도 경로들을 탐색할 수 있는 휴리스틱 기반의 라우팅 알고리즘을 제시하였다

(3) 기존 연구에서는 충분히 논의되지 않았던 미래 질의에 대한 처리 방안을 제시함으로써 사용자의 다양한 요구를 반영할 수 있도록 하였다.

(4) 현실 세계의 특성을 반영한 실험 데이타를 생성하는 방안에 대하여 논의하였다.

(5) 생성된 데이타를 대상으로 가지치기의 정도를 결정하는 BETA 값을 체계적으로 결정하였으며, 실험을 통해 제안된 알고리즘의 효율성 및 정확성을 입증하였다. 실험 결과에 따르면 제안된 알고리즘은 1.5초 이내의 빠른 응답 속도 및 평균 82%의 정확도를 가지며 데이타의 크기가 커지는 경우에도 효율성 및 정확성을 유지하는 것으로 나타났다.

향후 연구 방향으로는 버스와 지하철의 연계 방안, 사용자 위치와 버스의 현재 위치를 고려하여 최적의 출발 정류장과 이동 경로에 대한 정보를 제공, 갈아타기를 위해 약간의 도보 이동을 허용하는 방안, 대중교통 이용에 있어 시간, 요금 등과 같은 개인의 취향을 사용자 프로파일(user profile)에 저장시켜 라우팅 알고리즘에 반영하는 방안 등에 대한 연구를 진행할 예정이다.

참 고 문 헌

[1] SK Telecom, http://www.sk.com/products/telecom/telecom_telecom.asp

[2] G. Abowd, C. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton, "Cyberguide: a Mobile Context-Aware Tour Guide," Wireless Networks, 3(5):421-433, 1997.

[3] O. Eriksson, "Location Based Destination Information for the Mobile Tourist," in Proc. Intl. Conf. on Information and Communication Technology in Tourism, pp. 22-25, 2002.

[4] F.-Y. Wang, S. Tang, Y. Sui, and X. Wang, "Toward Intelligent Transportation Systems for the 2008 Olympics," in Proc. IEEE Intelligent Systems, pp. 8-11, 2003.

[5] S. Q. Shi, X. Gong, and H. Mo, "Study on Real Time Traffic Flow Routing Algorithm Based on Case Based Reasoning," in Proc. Intl. Conf. on Intelligent Transportation Systems, pp. 163-167, 2003.

[6] J. F. Dillenburg, O. Wolfson, and P. C. Nelson, "The Intelligent Travel Assistant", in Proc. Intl. Conf. on Intelligent Transportation Systems, pp. 691-696, 2002.

[7] THINKWARE Systems Corp., http://www.thinkwaresys.com/

[8] The TomTom Website, http://www.TomTom.com

[9] J. Fawcett and P. Robinson, "Adaptive Routing for Road Traffic," IEEE Computer Graphics and Appliance, 20(3), pp. 46-53, 2000.

[10] M. Bertolotto, G. O'Hare, R. Strahan, A. Brophy, A. Martin, and E. McLoughlin, "Bus Catcher: a Context Sensitive Prototype System for Public Transportation Users," in Proc. IEEE Intl. Conf. on Web Information Systems Engineering, pp. 64-72, 2002.

[11] R. Strahan, C. Muldoon, G.M.P. O'Hare, M. Bertolotto, and R. W. Collier, "An Agent-Based Architecture for Wireless Bus Travel Assistants," in Proc. Wireless Information Systems, 2003.

[12] S. D. Maclean and D. J. Dailey, "Real-time Bus Information on Mobile Devices," in Proc. IEEE Intl. Conf. on Intelligent Transportation Systems, pp. 25-29, 2001.

[13] J. Boyan and M. Mitzenmacher, "Improved Results for Route Planning in Stochastic Transportation Networks," in Proc. 12th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 895-902, 2001.

[14] Jenks and Christopher, "Applications of Intelligent Transportation Systems to Public Transportation in Europe," TCRP Research Results Digest, 1998.

[15] M. Datar and A. Ranade, "Commuting with Dealy Prone Buses," in Proc. 11th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 22-29, 2000.

[16] D. J. Dailey, F. W. Cathey, S. D. Maclean, "Design and Realization of Multi-Modal/Multi-Agency Transit Management and Information System," in Proc. IEEE Conf. on Intelligent Transportation Systems, pp. 1664-1669, 2003.

[17] http://www.utm.edu/departments/math/graph/glossary.html

[18] http://www.investorwords.com/5561/exponential_moving_average.html

[19] 이승재, 김범일, 권혁, "단기 통행시간예측 모형 개발에 관한 연구," 한국 ITS 학회 논문지, 제3권, 제1호, 2004.

[20] 고승영, "버스도착시간 정보에 대한 연구," 대한교통학회지, Journal of Transportation Research Society of Korea, 1229-1366, 제20권 5호, pp.175-181, 2002.

이 종 찬

2004년 8월 연세대학교 컴퓨터과학과 졸업(학사). 2006년 8월 연세대학교 컴퓨터과학과 대학원 졸업(석사). 관심분야는 위치기반서비스, ITS, 데이타베이스 시스템, 데이타 마이닝, 바이오인포메틱스

박 상 현

정보과학회논문지 :
제 33 권 제 3 호 참조

서 민 구

정보과학회논문지 : 데이타베이스
제 33 권 제 3 호 참조

김 상 욱

정보과학회논문지 : 데이타베이스
제 33 권 제 2 호 참조