http://dx.doi.org/10.3745/KIPSTD.2012.19D.2.127

서울 대도시권 버스 네트워크에서 픽업 위치 선정을 위한 흐름-포착 위치-할당 모델의 적용

박 종 수†

요 약

서울 대도시권 버스 네트워크에서 승객이 전자 상거래로 구매한 소화물을 집이나 사무실로 가는 도중의 버스 정류장에서 수령할 필요가 있을 수 있다. 흐름-포착 위치-할당 모델을 응용하면 포착되는 승객 흐름을 최대화시키는 버스 정류장들을 선정하는 것이 가능하다. 승객 흐름은 승객 이동의 기-종점 쌍을 나타낸다. 본 논문에서는 기-종점 행렬을 이용하여 빠르게 픽업 위치를 선정하는 휴리스틱 알고리즘을 제안한다. 실험을 위한 대용량의 기-종점 행렬은 오백만 교통카드 트랜잭션들로부터 추출된다. 실험 결과에서 승객 흐름에 관한 특성과 포착률을 이용해 픽업 위치로 선정된 버스 정류장들을 설명하고 상위 20개 픽업 위치의 공간적인 분포를 지도로 표시하였다.

키워드 : 픽업 문제, 버스 승객 흐름, 기-종점 행렬, 교통카드 트랜잭션

Application of the Flow-Capturing Location-Allocation Model to the Seoul Metropolitan Bus Network for Selecting Pickup Points

Jong Soo Park†

ABSTRACT

In the Seoul metropolitan bus network, it may be necessary for a bus passenger to pick up a parcel, which has been purchased through e-commerce, at his or her convenient bus stop on the way to home or office. The flow-capturing location-allocation model can be applied to select pickup points for such bus stops so that they maximize the captured passenger flows, where each passenger flow represents an origin-destination (O-D) pair of a passenger trip. In this paper, we propose a fast heuristic algorithm to select pickup points using a large O-D matrix, which has been extracted from five million transportation card transactions. The experimental results demonstrate the bus stops chosen as pickup points in terms of passenger flow and capture ratio, and illustrate the spatial distribution of the top 20 pickup points on a map.

Keywords : Pickup Problem, Bus Passenger Flow, Origin-Destination Matrix, Transportation Card Transaction

1. 서 론

서울 대도시권에서 2004년도에 버스 교통 시스템의 정비 및 교통 카드를 사용한 지하철 시스템과 환승으로 인하여 많은 시민들이 대중교통을 이용하게 되었고, 이런 대중교통 승객들의 이동 상황은 대용량의 교통카드 트랜잭션 데이터베이스로 관리되고 있다. 하루 천만 건 이상의 대용량 교통카드 트랜잭션 데이터베이스에서 지금까지 잘 알려져 있지 않은 주요한 정보와 지식을 추출하고 발견해내는 것[1, 2]은 새로운 교통정책 제안 등에 아주 중요하다. 본 논문에서는 교통카드 트랜잭션 데이터베이스에서 버스 승객들의 승객 흐름을 추출하여 픽업 문제에 응용하였다. 픽업 문제는 소비자들에게 어떤 서비스 시설을 제공할 때 어느 위치에 설치하면 그것을 이용할 수 있는 소비자들의 숫자를 최대화시키는 위치를 찾아내는 것이다[3, 4, 5]. 이런 서비스 시설에 해당되는 것은 전자 상거래 물품의 택배 취급 지점, 신선 농수산물 픽업 지점, 전기 자동차의 충전소, 전광판, 스마트폰 또는 태블릿의 AS 센터 등이 있다.

기존 연구에서는 실제 교통 데이터가 없어 대규모 버스 교통 네트워크에서 픽업 문제를 다루지 못하였지만, 본 논문에서는 서울 대도시권 대용량 교통카드 트랜잭션 데이터베이스로 부터 픽업 버스 정류장을 선정하는 알고리즘을 연구한다. 기존의 픽업 문제에서 다루었던 것은 버스 정류

※ 이 논문은 2011년도 성신여자대학교 학술연구조성비 지원에 의하여 연구되었음.
† 종신회원 : 성신여자대학교 IT학부 교수
논문접수 : 2012년 2월 21일
심사완료 : 2012년 4월 2일

장 등으로 픽업 위치로 선정될 수 있는 입력 위치들의 개수가 몇 십 개에서 300개 이내의 소규모 시스템에서 해결책을 찾으려고 하였고[3, 4, 5], 서울 수도권 지하철 시스템에서도 지하철역들의 개수는 400개 정도인 경우다[6]. 서울 대도시권 버스 네트워크는 15,000개 정도의 버스 정류장들로 구성되어 있고, 매일 시민들이 500만 번 넘게 버스로 통행하고 있다. 본 논문에서는 이와 같은 대규모 교통 네트워크에서 픽업 문제를 해결하는 빠른 알고리즘을 제시하고 이를 정교한 프로그래밍으로 구현하여 실험 결과를 보여주고 분석한다.

본 논문의 구성은 다음과 같다. 2장에서 버스 승객 흐름에 대한 픽업 위치 선정 모델을 기술하고 각 식과 변수에 대하여 설명한다. 3장에서는 버스 승객의 교통카드 트랜잭션의 속성들을 알아보고 버스 정류장과 버스 노선을 관리하는 자료구조를 설명하고, 기-종점 행렬을 사용하여 픽업 위치를 선정하는 휴리스틱 알고리즘을 상세히 설명한다. 4장에서는 교통카드 트랜잭션 데이터베이스를 적용하여 얻어진 결과를 분석하고 픽업 위치들을 지도 위에 보여준다. 5장에서는 결론을 제시한다.

2. 서울 대도시권 버스 승객 흐름에 대한 Pickup Model

흐름 포착 위치 할당 모델[3, 4], 픽업 문제의 모델[5], 그리고 서울 대도시권 지하철 승객의 픽업 모델[6]을 분석하였다. 이 모델들을 서울 대도시권 버스 네트워크의 승객들에 적용하여 픽업 버스 정류장을 결정하는 모델로 변환시킨다. 서울 대도시권 버스 네트워크에서 승객이 한 버스를 승차하고 하차한 후에 다시 같은 노선의 버스를 타면 환승으로 간주하지 않고 버스 요금을 다시 계산하고 있다. 이런 버스 시스템에서 픽업 버스 정류장을 선정하기 위해서는 버스 승객의 승차 위치와 하차 위치로 구성되는 승객 흐름을 파악해야 한다. 앞의 모델을 응용하여 p개의 버스 정류장을 픽업 위치(pickup point)로 선정하는 모델은 다음 식과 같다.

$$\text{최대화:} \quad Z = \sum_{q \in Q} f_q y_q \tag{1}$$

제한 조건;

$$\sum_{k \in N_q} x_k \geq y_q, \forall q, \qquad \sum_{k \in K} x_k = p,$$
$$x_k \in \{0,1\}, \forall k \in K, \qquad y_q \in \{0,1\}, \forall q \in Q.$$

식 (1)과 제한 조건에 나타난 변수들을 설명하고자 한다. q는 버스 승객이 한 버스 정류장에서 승차하여 다른 버스 정류장에서 하차하는 승객 흐름을 나타내며 기-종점 쌍(Origin- Destination pair)이라고 부른다[7]. m개의 버스 정류장들이 있다고 하면, 최대 $m \times m$개의 승객 흐름들이 있을 수 있다. Q는 모든 O-D 쌍들의 집합을 나타내고, f_q는 O-D 쌍으로 표시된 q에서 통행한 승객수를 나타낸다. y_q는 f_q에 해당되는 승객흐름의 한 정류장이 픽업 버스 정류장으로 선정되면 1이 되고, 선정되지 않으면 0이 된다. k는 잠재적인 픽업 위치인 버스 정류장의 위치를 나타내고, K는 모든 버스 정류장들의 위치들의 집합을 나타내고, $|K| = m$으로 둔다. 만약 픽업 버스 정류장의 위치가 k이면 x_k는 1이 되고, 그렇지 않으면 x_k는 0이 된다. N_q는 승객 흐름 q를 포착할 수 있는 잠재적인 픽업 버스 정류장 위치들로 승차 버스정류장과 하차 버스정류장을 포함한 집합으로 표시하고, 마지막으로 p는 선정되어야 할 픽업 버스 정류장들의 개수를 표시한다.

식 (1)의 목적 함수인 Z를 최대화하도록 결정 변수인 y_q와 x_k를 정하는 것이 이 모델의 해결책이 된다. 이러한 픽업 문제는 NP-hard 문제로 알려져 있고[3, 4], 주로 이진 선형 최적화 방법으로 상업용 소프트웨어 패키지로 최적해를 구하고 있다[5]. 본 논문에서는 m개의 버스 정류장들에서 승객 흐름이 최대화 되도록 p개의 픽업 버스 정류장들을 선정하는 알고리즘을 제안하고 실험으로 그 결과를 보여주고자 한다.

3. 교통카드 트랜잭션 DB에서 픽업 버스 정류장을 선정하는 알고리즘

본 논문에서 사용되는 교통카드 트랜잭션의 16개 속성들은 <표 1>에서 보여주고 있다. 카드번호는 승객의 교통 카드 번호가 아니고 임의로 부여된 일련번호에 해당된다. 트랜잭션ID는 한 승객이 하루 동안에 발생한 트랜잭션의 순서를 나타낸다. <표 1>의 트랜잭션 예제 1과 2는 한 승객이 버스를 두 번 사용했을 때 발생한 각각의 트랜잭션을 보여주고 있다. 트랜잭션 예제 1은 그 승객이 2007년 11월 13일 12시 28분 33초에 간선버스(115) 507번(석수역~동대문운동장) 버스노선의 정류장ID 10064인 “본동가칠목” 정류장에서 버스를 타서 12시 34분 55초에 버스정류장ID 9511인 “신길새마을금고” 정류장에서 내리고 요금은 900원을 지불했다는 것을 나타내고 있다. 트랜잭션 2는 카드번호가 같은 승객이 같은 날 17시 44분 57초에 지선버스(120)인 6411번(구로동~개포동) 버스노선의 정류장ID가 9426인 “대신시장” 정류장에서 버스를 타고 버스정류장ID 10010인 “사육신묘” 정류장에서 하차한 것을 나타내고 있다.

버스 승객의 트랜잭션에서 승차정류장과 하차정류장을 추출해내어 어느 버스 노선인지를 파악하려면, 사전 작업으로 서울 대도시권 버스 네트워크의 버스 노선에 관한 자료 구조를 설정해야 한다. 버스 정류장의 정보를 유지하는 자료구조는 (그림 1)에 설명되어 있다. (그림 1)의 버스 정류장의 class로 1차원 배열 bus_stops[]를 선언하여 각 버스 정류장의 정보를 이 배열에 저장한다. 버스 정류장의 정보를 배열에 먼저 순서대로 저장하고, (그림 1)의 노드를 가리키

〈표 1〉 교통 카드 트랜잭션의 속성과 예제 (2007년 기준)

속성 (attribute)	카드번호, 출발시간, 트랜잭션ID, 교통수단ID, 환승횟수, 버스노선ID, 교통사업자ID, 차량ID, 사용자구분코드, 운행출발일시, 승차정류장ID, 도착일시, 하차정류장ID, 이용객수, 승차금액, 하차금액
트랜잭션 예제 1	101356217, 20071113122833, 056, 115, 0, 41110205, 111009020, 111706796, 01, 20071113102300, 0010064, 20071113123455, 0009511, 1, 900, 0
트랜잭션 예제 2	101356217, 20071113174457, 057, 120, 0, 11110267, 111007500, 111748460, 01, 20071113170657, 0009426, 20071113175526, 0010010, 1, 900, 0

```
class   Busstop_node {                  // Bus stop (버스정류장) 노드, 기본 자료구조
public:
    int             busstop_ID;         // 버스정류장ID
    char            busstop_name[51];   // 버스 정류장 이름
    double          longitude;          // 경도
    double          latitude;           // 위도
    bool    flag;   // 인접 구역에서 대표 정류장의 여부 또는 pickup 정류장의 선정 여부
    int     num_of_same_name;           // 같은 이름의 버스 정류장들의 개수
    int     num_of_busroutes;  // 이 버스 정류장을 통과하는 버스 노선들의 개수
    Busroute_node_index  *busroutes; // 이 정류장을 통과하는 버스 노선 노드들의 Linked List
};
```

(그림 1) 버스 정류장의 기본 자료구조

```
FindPickupBusstops(int  p) {  // p 개의 pickup 버스 정류장을 선정
단계 1: 버스 노선 파일을 읽어서 버스 정류장과 버스 노선에 대한 자료를 해당 배열에 저장하고, 빠른 접근을
        위한 hash-table을 만든다.
단계 2: 교통카드 트랜잭션에서 버스 승객의 승차 정류장과 하차 정류장을 추출하여 hash-table을 이용하여 각
        정류장의 인덱스를 찾아서 OD-matrix를 만든다.
단계 3: 인접한 지역에서 동일 버스 정류장 이름을 사용하는 정류장들을 승객 흐름이 가장 많은 위치의 대표
        버스 정류장으로 통합한다.
단계 4: Pickup 버스 정류장을 선정한다:
    단계 4.1: 각 버스 정류장의 flag를 false로 설정한다.
    단계 4.2: OD-matrix에서 정류장의 flag가 false인 정류장들 중에서 승차승객수와 도착승객수의 합이 최대인
              정류장을 찾아 pickup 버스 정류장으로 선정한 후에 포착률을 계산하고 그 정류장의 flag를 true로
              설정한다.
    단계 4.3: OD-matrix의 나머지 정류장들의 각 원소의 값은 선정된 pickup 정류장에서 출발하는 승객수와
              도착하는 승객수 만큼 감소시킨다.
    단계 4.4: 주어진 p개의 pickup 버스 정류장을 찾지 못하면, 단계 4.2로 간다.
}
```

(그림 2) 픽업 버스 정류장을 선정하는 휴리스틱 알고리즘

는 pointer로 이루어진 버스정류장 hash-table에 busstop_ID로 해싱을 하여 해시 체인의 노드에 배열 bus_stops[]의 인덱스를 저장한다. 서울 대도시권 버스 네트워크의 버스 노선들도 버스 정류장의 자료구조와 유사한 구조로 만들어 찾으려고 하는 버스 노선과 그 노선에 속한 버스 정류장들의 정보를 저장한다.

(그림 2)는 입력으로 주어진 교통카드 트랜잭션 데이터베이스에서 픽업 버스 정류장을 선정하는 단계들을 설명한 것

이다. 단계 1에서 주어진 서울 대도시권 버스 노선에서 통과하는 버스 정류장들에 대한 정보를 1차원 배열에 저장한다. 버스 노선들과 버스 정류장들에 대한 자료의 빠른 접근을 위하여 hash-table을 사용한다. (그림 2)의 단계 2에서는 버스 승객의 교통카드 트랜잭션들을 읽어서 각 트랜잭션에서 추출된 승차정류장ID와 하차정류장ID를 이용하여 2차원 배열인 OD-matrix의 값을 구성한다. 버스 정류장들의 개수가 m이라 하면 OD-matrix는 $m \times m$ 2차원 배열로 이루어지고 각 원소의 초기값은 0으로 설정된다. 한 트랜잭션에서 승차정류장ID와 하차정류장ID의 인덱스가 i와 j이면, OD-matrix[i][j]의 값이 1 증가한다. 단계 2를 수행하면 인덱스 i와 j에 대한 모든 값에 대한 $m \times m$개의 기-종점 쌍에 대응되는 OD-matrix의 값이 정해진다. 인덱스 i인 한 버스정류장에서 승차하고 인덱스 j인 버스정류장에서 하차하는 기-종점 OD-matrix[i][j]의 값이 0 보다 크면, 식 (1)에서 기술된 승객흐름 q가 설정되고 f_q는 OD-matrix[i][j]의 값으로 표시된다.

(그림 2)의 단계 3은 버스 정류장 이름이 동일하고 위치가 인접한 정류장들을 승차승객수와 하차승객수를 합한 값이 가장 많은 버스 정류장 ID로 통합하는 과정을 설명한다. 버스 정류장 “서울역”은 인접 지역에서 버스 정류장 ID가 다른 14 곳에서 같은 이름으로 사용되고 있다. 단일 버스 정류장을 통과하는 승객수는 실험에서 전체 버스 트랜잭션들의 0.7%도 되지 않았다. 그러므로 인접한 지역 내의 동일한 버스 정류장 이름을 사용하는 서로 다른 버스 정류장 ID의 승객수를 통합된 대표 정류장 ID의 승객수로 묶어서 포착률을 높이도록 한다. 여기서 포착률(capture ratio)은 픽업 버스 정류장들을 통과하는 버스 승객수와 전체 버스 트랜잭션들의 개수의 비율로 나타낸다. (그림 2)의 단계 4에서는 승객 흐름을 나타내는 OD-matrix에서 flag를 사용하여 반복적으로 최대 승객 흐름을 갖는 버스 정류장을 픽업 위치로 선정하는 과정을 서술하고 있다. n이 교통카드 트랜잭션들의 개수, r이 버스노선들의 개수, m이 버스 정류장들의 개수일 때, 이 알고리즘의 각 단계별 시간 복잡도는 $O(rm)$, $O(n)$, $O(m^2)$, $O(pm)$으로 나타낼 수 있다. $r < m$이고 $p < m$이므로 전체적인 시간 복잡도는 $T(n,m) = O(n+m^2)$로 정리된다.

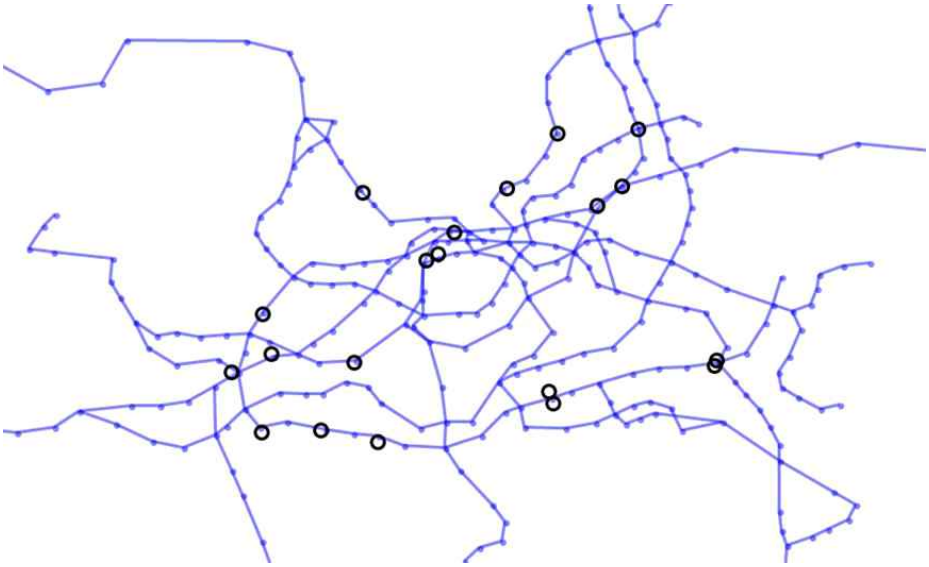
4. 실험 결과 및 분석

2007년 11월 기준의 서울 대도시권 버스 네트워크에서 교통카드 트랜잭션 데이터베이스를 입력으로 하여 얻어진 결과를 분석한다. 실험에 사용된 컴퓨터의 운영체제는 MS Windows 7 64 bit이고, 하드웨어의 주요 사양으로는 Intel i920 2.67GHz CPU, 220GB의 OCZ SSD, 메인 메모리로 16GB DDR3 RAM 등이다. 프로그램 개발 환경은 MS Visual Studio 2008 C++언어를 사용하였다. 2007년 11월의 서울 대도시권 버스 네트워크는 611개의 버스 노선들과 14,990개의 버스 정류장들로 구성된다. 실험에 사용된 입력 데이터는 2007년 11월 13일 하루 저장된 교통카드 트랜잭션 데이터베이스로 지하철 승객들을 포함하여 전체 11,212,392 트랜잭션들이고, 그 중에서 버스 승객 트랜잭션들의 개수는 5,303,421개이다.

버스 정류장들의 개수 m=14,990에서 (그림 2)에 기술된 알고리즘의 단계 2를 수행한 후에 식 (1)에서 기술된 승객 흐름들의 개수인 $|Q|$는 380,002개로 계산되어졌다. 이것은 서울 대도시권 버스 네트워크에서 승차와 하차를 할 수 있

〈표 2〉 픽업 버스 정류장의 승객수와 포착률 (2007년 11월 데이터)

순서	승하차 승객수	포착 승객수	포착률(%)	버스정류장이름	동명 정류장개수	순서	승하차 승객수	포착 승객수	포착률(%)	버스정류장이름	동명 정류장개수
1	48129	48129	0.91	구로디지탈단지역	2	11	29710	412099	7.77	삼선교(한성대입구)	4
2	45948	94077	1.77	청량리역	6	12	17751	429850	8.11	회기역	5
3	43165	137242	2.59	잠실역	4	13	28523	458373	8.64	당산역	7
4	39322	176564	3.33	서울역	14	14	29137	487510	9.19	석계역	17
5	39635	216199	4.08	미아삼거리역	3	15	28033	515543	9.72	홍제역	14
6	35500	251699	4.75	남대문시장	7	16	27302	542845	10.24	관악구청	3
7	33959	285658	5.39	종로2가	7	17	27211	570056	10.75	노량진역	8
8	32783	318441	6.00	신림사거리	2	18	26545	596601	11.25	강남역	13
9	33143	351584	6.63	잠실역롯데월드	2	19	25113	621714	11.72	신세계백화점	4
10	30805	382389	7.21	교보타워사거리	2	20	23661	645375	12.17	신도림역	10



(그림 3) <표 2>의 픽업 버스 정류장의 위치

는 모든 기-종점 쌍의 개수 $m \times m$ = 224,700,100의 0.169%의 비율이다. 이것은 실제 승객들이 주어진 버스 노선 상의 정류장에서만 승차와 하차를 하기 때문이다. <표 2>는 (그림 2)의 알고리즘에 따라 선정된 픽업 버스 정류장들 중에서 상위 20개를 나타내고 있고 약 12% 포착률을 보여주고 있다. 실험 결과에서 25%의 포착률에는 59개 픽업 정류장들, 50% 포착률에는 229 픽업 정류장들, 75% 포착률에는 1,671 픽업 정류장들, 100%의 포착률에는 7,117 픽업 버스 정류장들이 선정되었다. p가 20일 때, 알고리즘의 전체 실행 시간은 127.119초이었고, 각 단계별 실행시간은 3.12초, 112.626초, 11.341초이고, 단계 4는 0.032초에 실행되었다.

(그림 3)은 <표 2>에 기술된 픽업 버스 정류장들의 위치를 2007년도의 서울 수도권 지하철 노선도가 있는 지도 위에 원으로 표시하였다. 픽업 버스 정류장들의 공간 분포는 주로 지하철역에 인접해 있고 지하철 2호선에 인접한 정류장들은 10개나 된다. 버스 승객과 지하철 승객의 밀접한 연관성을 유추해 볼 수 있다.

5. 결 론

서울 대도시권 버스 네트워크는 15,000여개의 버스 정류장들로 구성된 대규모 대중교통 시스템으로 하루 500만 명 이상의 버스 승객들이 이용하고 있다. 버스 승객들의 트랜잭션 데이터베이스에서 승객 흐름을 최대화하는 버스 정류장을 찾아내는 픽업 문제에 대한 모델 제시와 문제 해결을 위한 휴리스틱 알고리즘을 제안하였다. 본문에서 제안된 알고리즘의 단계들을 상세히 기술하였고 실험 결과에서 알고리즘의 빠른 실행 시간을 보여주었다. 교통카드 트랜잭션 데이터베이스에서 알고리즘으로 찾아진 픽업 버스 정류장들에서 승객 흐름의 특성을 분석하고 포착률을 계산하였다. 픽업 버스 정류장들의 위치는 주로 서울 대도시권 지하철역들과 겹치는 것으로 버스 승객들의 지하철 환승을 추론해 볼 수 있다. 추후 연구 과제로는 지하철로 환승하는 승객들을 포함하여 서울 대도시권 대중교통 승객 흐름의 포착률이 높은 지하철역이나 버스 정류장을 찾아내는 것이다.

참 고 문 헌

[1] K. Lee, W.-S. Jung, J. S. Park, and M. Y. Choi, “Statistical analysis of the Metropolitan Seoul Subway System: Network structure and passenger flows,” Physica A: Statistical Mechanics and its Applications, Vol.387, Iss.24, pp.6231-6234, 2008.

[2] 박종수, 이금숙, “서울 수도권 지하철 교통망에서 승객 흐름의 분석,” 한국정보과학회 논문지: 컴퓨팅의 실제 및 레터, 제 16권 제 3호, pp.316-323, 2010.

[3] M. J. Hodgson, “A Flow-Capturing Location-Allocation Model,” Geographical Analysis, Vol.22, No.3, pp.270-279, 1990.

[4] M.J. Hodgson, K.E. Rosing, A. Leontien and G. Storrier, “Applying the flow-capturing location-allocation model to an authentic network: Edmonton, Canada”, European Journal of Operational Research 90, pp.427-443, 1996.

[5] W. Zeng, M. J. Hodgson, and I. Castillo, “The Pickup Problem: Consumers’ Locational Preferences in Flow Interception,” Geographical Analysis 41, pp.107 - 126, 2009.

[6] 박종수, 이금숙, “Pickup Point 최적 입지선정을 위한 Greedy Heuristic Algorithm 개발 및 적용: 서울 대도시권 지하철 시스템을 대상으로,” 한국경제지리학회지, 제 14권 제 2호, pp.116-128, 2011.

[7] 임강원, 임용택, 교통망 분석론, 서울대학교출판부, 2003.

박 종 수

e-mail : jpark@sungshin.ac.kr

1981년 부산대학교 전기기계공학과(학사)

1983년 한국과학기술원 전기및전자공학과 (석사)

1990년 한국과학기술원 전기및전자공학과 (박사)

1983년~1986년 국방부 군무설계기좌

1994년 7월~1995년 7월 IBM Watson 연구소 객원 연구원

1990년~현 재 성신여자대학교 IT학부 교수

관심분야 : 데이터베이스, 데이터마이닝, 교통지리