선형배열 기지국을 위한 위치정보 서버의 최적할당 방식

임 경 식†

요 약

이동 통신환경에서 이동노드들은 기지국(base station)을 통하여 통신하므로, 기지국은 트래픽의 발신지 및 착신지로서 데이터 트래픽, 위치정보 트래픽 등과 같이 처리비용이 다른 이종의 트래픽을 발생한다. 그런데, 하나의 위치정보서버는 물리적으로 제한된 용량을 가지므로 여러 대의 위치정보서버를 분산시켜 구축할 필요가 있는데, 이때 기지국을 최적으로 클러스터링하고 각 클러스터마다 하나의 위치정보서버를 할당해야 한다.

본 논문에서는 기지국이 선형으로 배열된 이동 통신망에서, 서로간에 다양한 형태의 트래픽을 발생하는 n 개의 기지국이 주어졌을 때, 전체 네트워크에 대한 통신비용을 최소화하기 위하여 이들을 m ($1 \le m \le n$) 개의 인접한 클러스터(disjoint cluster)로 분리하는 문제를 고려한다. 이를 위하여, 본 논문에서는 주어진 트래픽이 클러스터 내부에서 발생할 때와 클러스터간에 발생할 때의 통신비용 차이를 반영한 상대비용(relative cost) 개념을 도입하여 $O(mn^2)$의 동적 프로그래밍(dynamic programming) 알고리즘을 제시한다. 또한, 이 알고리즘은 하나의 클러스터에 대한 크기제한과 전체 네트워크에 허락된 총 통신비용이 제약조건으로 주어질 경우, 같은 계산시간 내에 모든 유효한 클러스터를 찾아 낼 수 있음 보인다.

An Optimal Allocation Mechanism of Location Servers in A Linear Arrangement of Base Stations

Kyung-Shik Lim†

ABSTRACT

Given a linear arrangement of n base stations which generate multiple types of traffic among themselves, we consider the problem of finding a set of disjoint clusters to cover n base stations so that a cluster is assigned a location server. Our goal is to minimize the total communication cost for the entire network where the cost of intra-cluster communication is usually lower than that of inter-cluster communication for each type of traffic. The optimization problem is transformed into an equivalent problem using the concept of relative cost, which generates the difference of communication costs between intra-cluster and inter-cluster communications. Using the relative cost matrix, an efficient algorithm of $O(mn^2)$, where m is the number of clusters in a partition, is designed by dynamic programming. The algorithm also finds all the valid partitions in the same polynomial time, given the size constraint on a cluster and the total allowable communication cost for the entire network.

† 정 회 원 : 경북대학교 컴퓨터과학과 교수
논문접수 : 1999년 10월 20일, 심사완료 : 2000년 2월 1일

1. 서 론

이동 통신망에서 특정 이동노드와 통신채널을 설정해야 할 때마다 네트워크는 우선 기지국들 중 어느 것이 그 노드와 통신할 수 있는지를 알아내야 한다. 이러한 이동노드의 위치추적 문제와 관련하여 여러 가지 연구결과가 발표되었는데, 대표적인 것으로는 그래프 이론과 영역매칭(regional matching) 기술을 사용한 계층적인 위치정보 서버[1,2], 중앙 센터(reporting center) 개념을 기반으로 한 부분적인 위치정보 보고[3], 무선 환경에서 질의어를 이용하여 사용자의 위치정보를 파악하는 방법[4] 등을 들 수 있다.

이제 특정한 무선통신망에 적합한 하나의 위치정보 추적방식이 결정되고, 네트워크는 이 방식을 사용하여 이동중인 노드를 추적한다고 가정하자. 이때, 이동노드의 위치추적에는 기본적으로 논리적인 두 가지 기능인 *move*와 *find*가 사용된다. 전자는 노드가 이동할 때, 기지국들과 위치정보서버들 사이에서 교환되는 위치수정 메시지를 통하여 이동중인 노드의 위치정보를 네트워크에 등록하는 기능이며, 후자는 네트워크가 통신채널 설정 요청을 받을 경우 목적지 이동노드를 담당하고 있는 현재의 기지국을 발견하는 기능이다. 본 논문에서는 n 개의 기지국이 선형으로 배열된 이동통신망 환경에서 효율적인 하나의 위치추적 방식이 결정되고 기지국간에 *move*와 *find* 기능의 빈도가 주어졌을 때, m $(1 \le m \le n)$ 개의 위치정보서버를 기지국에 최적으로 할당하는 문제를 다룬다.

위치정보서버는 보통 고정망(fixed network)으로 상호 연결되어 있으며, 관할 기지국과도 고정망으로 상호 연결되어있다. 하나의 위치정보서버는 보통 제한된 수의 기지국을 관리하고, 그 기지국에 처음으로 가입된 이동노드를 추적한다. 위치정보서버의 관점에서 보면, 기지국은 이동노드와 네트워크 사이를 연계시켜주는 역할을 함과 동시에 *move*와 *find* 트래픽의 발생지와 목적지가 된다. 이때, 하나의 위치정보서버가 관리하는 영역 내에 있는 이동노드간의 통신비용은 다른 위치정보서버의 관리하에 있는 이동노드간의 통신비용보다 일반적으로 훨씬 적다. 그러므로, 어떤 위치정보 추적방식이 사용되든지 위치정보서버의 최적할당 문제는 네트워크 전체 성능을 향상시키기 위해 해결해야 할 근본 문제인데, 이는 결국 기지국의 최적분할 문제로 변환될 수 있다. 즉, 기지국들간의 *move*와 *find* 빈도를 완전한 방향성 그래프(complete directed graph)로 표현할 때, 네트워크의 전체 통신비용을 최소화하도록 기지국을 최적 분할하는 문제로 변환될 수 있다. 또한, 이 문제는 분할된 각 클러스터(cluster) 내에 있는 기지국들이 물리적으로 인접하게 연결되도록 기지국들의 물리적인 배치에 따른 위상을 고려해야 한다. 따라서, 본 논문에서는 두 가지 종류의 그래프를 사용하는데, 하나는 통신비용의 최소화 계산을 위해 사용하는 빈도 그래프(frequency graph)이고, 다른 하나는 물리적인 위상을 고려하여 기지국을 분할하는데 사용하는 위상 그래프(topology graph)이다.

일반적으로 선형 그래프에서 n 개의 노드를 임의의 m $(1 \le m \le n)$ 클러스터로 노드의 순서를 지켜서 분할하는 방법의 수는 $N(n,m) = \binom{n-1}{m-1}$이고, 모든 m에 대해 n 개의 노드를 순서를 지켜서 분할하는 총 수는 2^{n-1}이다. 그러므로, 모든 가능한 분할을 검사하는데 소비되는 시간은 각 m에 대해 $O(N(n,m)n^2)$이고, 모든 m에 대해서는 $O(2^n n^2)$이다. 따라서, 필요한 계산시간(time complexity)과 저장공간(space complexity)을 줄이기 위한 효율적인 알고리즘이 필요한데, 일반 그래프에 대한 기존의 알고리즘들은 본 논문에서 다루는 최적분할 문제를 해결하는데 부적합하다. 예를 들면, 다른 클러스터에 있는 두 개의 노드를 연결하는 비용의 합이 주어진 양의 정수보다 작거나 같도록 n 개의 노드를 인접한 클러스터로 분할하는 문제는 NP-complete이다[5]. 또한, 클러스터들 사이의 총 비용이 최소가 되도록, 주어진 노드들을 하나 이상의 노드를 포함하는 k 개의 클러스터들로 분할하는 문제도 임의의 k에 대해 NP-complete이다. 그러나, 임의의 노드가 지정될 때, 이 문제는 2-cut 문제의 확장인데, 이는 $k \ge 3$에 대하여 NP-hard이다[6]. 이러한 분할 문제는 하나의 고정된 k에 대한 평면 그래프(planar graph)에서는 해결될 수 있으나, k가 고정되지 않는 문제에 대해서는 NP-hard가 된다[7].

본 논문에서는 원래의 문제를 상대비용함수(relative cost function)를 이용한 최적화 문제로 전환한다. 상대비용함수는 *move*와 *find* 빈도수가 주어진 한 쌍의 기지국이 같은 클러스터 내에 있을 경우의 통신비용과 다른 클러스터에서 있을 경우의 통신비용과의 차이를

생성한다. 상대비용함수로 생성한 n 개 기지국간의 상대비용행렬(relative cost matrix)을 이용하여 누적비용행렬(cumulative cost matrix)을 구하고, 이를 기반으로 $O(mn^2)$의 동적 프로그래밍(dynamic programming) 알고리즘을 개발한다. 우선 클러스터의 크기에 제한이 없는 일반적인 경우를 고려한 알고리즘을 개발하고, 마지막으로 이 알고리즘이 클러스터의 크기 제한과 네트워크 전체 통신비용 제한이 있을 경우도 같은 시간 내에 모든 유효한 분할을 찾을 수 있음을 보인다.

2. 최적화 문제의 형식화

2.1 최적화 문제 정의

기지국 네트워크는 논리적으로 $|V|=n$인 완전한 방향성 그래프(complete directed graph) $G=(V,E)$로 나타낼 수 있다. 집합 V에 포함된 노드들은 기지국을 나타내며, 물리적인 위상에 따라 순서대로 $1,2,\ldots,n$으로 표현된다. 집합 E에 포함된 에지(edge)는 $f_m: V\times V \rightarrow R^+$에 의해 *move* 빈도수, $f_f: V\times V \rightarrow R^+$에 의해 *find* 빈도수가 할당된다. w_m^1과 w_m^2은 각각 클러스터 내부에서의 *move* 연산과 클러스터간 *move* 연산의 가중치이며, w_f^1와 w_f^2는 각각 클러스터 내부에서의 *find* 연산과 클러스터간 *find* 연산의 가중치를 가리킨다. 비용함수를 $c_i^j: V\times V \rightarrow R^+$로 정의하면, i = m 혹은 f 이고, j = 1 또는 2 일 때, 모든 $(u,v)\in V\times V$ 에 대해 다음과 같이 표현할 수 있다.

$$\begin{aligned} c_m^1(u,v) &= f_m(u,v)w_m^1, \\ c_m^2(u,v) &= f_m(u,v)w_m^2, \\ c_f^1(u,v) &= f_f(u,v)w_f^1, \\ c_f^2(u,v) &= f_f(u,v)w_f^2. \end{aligned} \quad (1)$$

수식 (1)은 한 쌍의 기지국에 연산의 빈도수가 할당되면 그 비용은 고정되는 것이 아니라, 분할 결과로 그 기지국들이 같은 클러스터 내에 위치하는지 아니면 서로 다른 클러스터에 위치하는지에 따라 그 비용이 동적으로 변한다는 것을 보이고 있다. $\{1, \ldots, n\}$으로 구성된 n 개의 노드들이 $P_i \cap P_j = \emptyset$ 이고 $\bigcup_i P_i = V$ $(i,j=1,\ldots,n \text{ and } i\neq j)$ 인 m 개의 클러스터 $\Pi=\{P_1, \ldots, P_m\}$으로 분할된다고 가정하자. 그러면, Π에 대한 클러스터 내부의 총 통신비용은

$$Cost_{intra}(\Pi) = \sum_{i=1}^{m} \sum_{u,v\in P_i} (c_m^1(u,v) + c_f^1(u,v)) \quad (2)$$

이고, Π에 대한 클러스터간 총 통신비용은

$$\begin{aligned} Cost_{inter}(\Pi) = \sum_{i=1}^{m-1} \sum_{u\in P_i, v\in P_j, i<j} ((c_m^2(u,v) \\ + c_f^2(u,v)) + (c_m^2(v,u) + c_f^2(v,u))) \end{aligned} \quad (3)$$

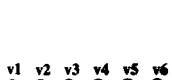
이다. (2)와 (3)에서 Π에 대한 전체 통신비용을 구하면 다음과 같다.

$$Cost_{total}(\Pi) = Cost_{intra}(\Pi) + Cost_{inter}(\Pi). \quad (4)$$

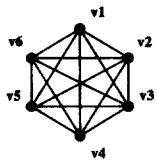
우리의 목적은 $i=1,\ldots,m$인 모든 i에 대해 $1\le|P_i|\le n$인 조건하에 $Cost_{total}$을 최소화하는 m 개의 인접한 클러스터(disjoint cluster)로 구성된 최적분할 Π를 찾는 것이다. 여기서 $|P_i|$는 i 번째 클러스터에 있는 노드의 수이다.

(그림 1)은 두 개의 빈도행렬(frequency matrix) f_m과 f_f를 가진 크기 6 (6개의 기지국)인 네트워크 모델을 나타낸다. (그림 2)에 있는 비용행렬은 수식 (1)에서 $w_m^1=w_f^1=1$, $w_m^2=3$, $w_f^2=2$를 사용하여 생성되었다. 이때, 만일 하나의 분할 $\Pi=\{P_1, P_2, P_3\}$에서 $P_1=\{1,2\}$, $P_2=\{3,4\}$, $P_3=\{5,6\}$이라면, $Cost_{intra}(\Pi)$와 $Cost_{inter}(\Pi)$는 각각 (그림 2)(a)와 (그림 2)(b)의 어두운 영역에 있는 모든 원소의 합이 된다. 따라서, $Cost_{total}(\Pi) = Cost_{intra}(\Pi) + Cost_{inter}(\Pi)$ = 44 + 493 = 537이다.

<표 1>은 n 개의 노드를 순서를 지켜서 m 개의 클러스터로 분할하는 방법의 수를 보여준다. <표 1>에 있는 수들은 파스칼 삼각형을 구성하고 있기 때문에, 임의의 n과 m이 주어졌을 때, 분할 방법의 수는 $N(n,m)=\binom{n-1}{m-1}$이 된다. 예를 들면, n=9이고 m=5이면 $N(9,5)=\binom{8}{4}=70$이 된다. <표 1>의 n 번째 행의 합은 이항식 이론(binomial theorem)에 따라 2^{n-1}이기 때문에, n 개 노드의 순서를 지키면서 분할하는 전체 경우의 수는 $n\ge1$일 경우 2^{n-1}이다. 따라서, 검색에 소비되는 시간은 각 m에 대해 $O(N(n,m)n^2)$이고, 모든 m에 대해서는 $O(2^n n^2)$이다.



기지국의 물리적인 배열



기지국의 논리적인 배열

	1	2	3	4	5	6
1	0	4	2	5	8	6
2	2	0	6	3	3	5
3	4	3	0	1	5	3
4	3	7	3	0	2	7
5	5	5	4	4	0	4

move 빈도 행렬

	1	2	3	4	5	6
1	0	2	4	6	3	2
2	4	0	7	2	2	1
3	3	5	0	7	5	3
4	2	6	3	0	1	5
5	6	2	2	4	0	3

find 빈도 행렬

(그림 1) 크기가 6인 네트워크 모델의 예

C_m^1

	1	2	3	4	5	6
1			2	5	8	6
2			6	3	3	5
3	4	3			5	3
4	3	7			2	7
5	5	5	4	4		
6	2	3	7	5		

C_f^1

	1	2	3	4	5	6
1			4	6	3	2
2			7	2	2	1
3	3	5			5	3
4	2	6			1	5
5	6	2	2	4		
6	4	3	5	3		

(a) 클러스터 내부 비용행렬

C_m^2

	1	2	3	4	5	6
1	0	12				
2	6	0				
3			0	3		
4			9	0		
5					0	12
6					15	0

C_f^2

	1	2	3	4	5	6
1	0	4				
2	8	0				
3			0	14		
4			6	0		
5					0	6
6					12	0

(b) 클러스터간 비용행렬

(그림 2) 클러스터 내부 및 클러스터간 비용행렬

〈표 1〉 분할 방법의 수

n	클러스터 수								
	m = 1	m = 2	m = 3	m = 4	m = 5	m = 6	m = 7	m = 8	m = 9
1	1								
2	1	1							
3	1	2	1						
4	1	3	3	1					
5	1	4	6	4	1				
6	1	5	10	10	5	1			
7	1	6	15	20	15	6	1		
8	1	7	21	35	35	21	7	1	
9	1	8	28	56	70	56	28	8	1

2.2 상대비용함수를 이용한 문제 변환

수식 (1)에서 보듯이 이 문제는 여러 종류의 트래픽을 다루면서 그 중 한 종류의 트래픽에 대해 여러 가지 비용함수를 상황에 따라 동적으로 적용해야하기 때문에, 계산시간과 저장공간 측면에서 효율적인 알고리즘을 개발하기가 매우 어렵다. 그러므로 동적비용함수를 이용한 원래의 최적화 문제를 정적비용함수를 이용한 새로운 최적화 문제로 전환하는 것이 바람직하다. 따라서, 본 논문에서는 모든 $(u, v) \in V \times V$에 대하여 수식 (5)를 만족하는 상대비용함수 $r: V \times V \rightarrow R^+$를 도입한다.

$$r(u, v) = f_m(u, v)(w_m^2 - w_m^1) + f_f(u, v)(w_f^2 - w_f^1). \quad (5)$$

상대비용함수 $r(u, v)$는 $f_m(u, v)$ *move* 연산과 $f_f(u, v)$ *find* 연산 모두에 대하여, 두 노드 u와 v가 동일 클러스터 내에 있을 경우와 다른 클러스터에 있을 경우 각각에 대한 통신비용의 차이를 생성한다. 따라서, 임의의 Π에 대하여 클러스터간 상대비용의 전체 합인 $Rcost_{inter}(\Pi)$는 다음과 같이 정의할 수 있다.

$$Rcost_{inter}(\Pi) = \sum_{i=1}^{m-1} \sum_{u \in P_i, v \in P_j, i<j} (r(u, v) + r(v, u)). \quad (6)$$

수식 (6)의 정의를 이용하면, $i = 1, \ldots, m$인 모든 i에 대해 $1 \le |P_i| \le n$의 제약조건 하에서 $Cost_{total}$을 최소화하기 위한 원래의 최적화 문제는 동일한 제약조건 하에서 $Rcost_{inter}$를 최소화하기 위한 새로운 최적화 문제로 변환된다. 이러한 변환은 수식 (7)과 (8)의 정의를 이용한 Lemma 1과 Theorem 1에 의하여 증명된다.

$$Upper_{intra}(\Pi) = \sum_{i=1}^{m-1} \sum_{u \in P_i, v \in P_j, i<j} (c_m^1(u, v) + c_f^1(u, v)). \quad (7)$$

$$Lower_{intra}(\Pi) = \sum_{i=1}^{m-1} \sum_{u \in P_i, v \in P_j, i<j} (c_m^1(v, u) + c_f^1(v, u)). \quad (8)$$

LEMMA 1.

$Cost_{constant}(\Pi) = Cost_{intra}(\Pi) + Upper_{intra}(\Pi) + Lower_{intra}(\Pi)$로 정의하면, $Cost_{constant}(\Pi)$는 분할방법과 무관하게 항상 일정한 값을 가지게 되며, $Cost_{total}(\Pi) = Cost_{constant}(\Pi) + Rcost_{inter}(\Pi)$가 된다.

PROOF. 수식 (1), (2), (3), (4)를 이용하면,

$$Cost_{total}(\Pi) = \sum_{i=1}^{m} \sum_{u,v \in P_i} (f_m(u,v)w_m^1 + f_f(u,v)w_f^1) + \sum_{i=1}^{m-1} \sum_{u \in P_i, v \in P_j, i<j} (f_m(u,v)w_m^2 + f_f(u,v)w_f^2) + \sum_{i=1}^{m-1} \sum_{u \in P_i, v \in P_j, i<j} (f_m(v,u)w_m^2 + f_f(v,u)w_f^2). \quad (9)$$

수식 (9)의 우변에 같은 값을 더하거나 빼도 등식은 변하지 않는다. 따라서, 수식 (9)의 우변에 있는 첫 번째 항에 $Upper_{intra}(\Pi)$ (7)과 $Lower_{intra}(\Pi)$ (8)을 더하고, 두 번째 항에서 $Upper_{intra}(\Pi)$를, 세 번째 항에서 $Lower_{intra}(\Pi)$를 빼면 다음의 수식을 얻을 수 있다.

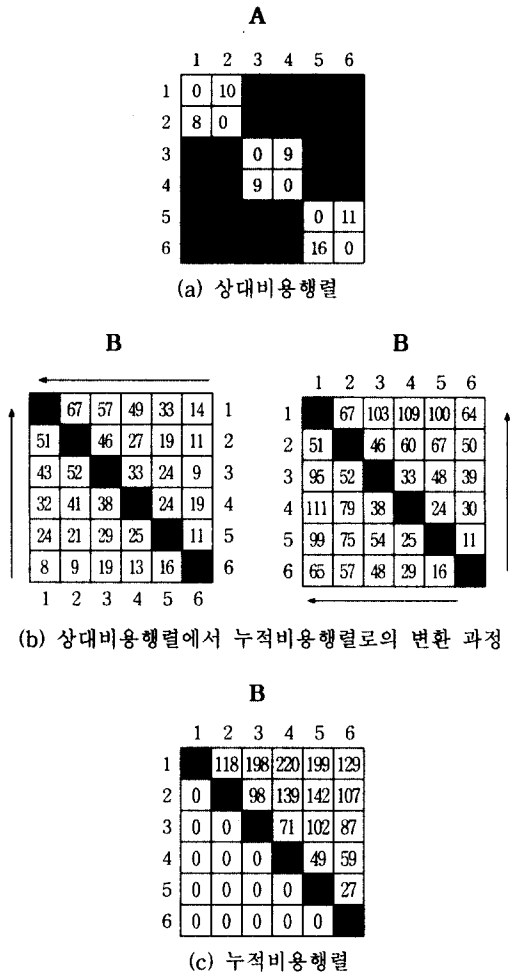
$$Cost_{total}(\Pi) = Cost_{constant}(\Pi) + \sum_{i=1}^{m-1} \sum_{u \in P_i, v \in P_j, i<j} (f_m(u,v)(w_m^2 - w_m^1) + f_f(u,v)(w_f^2 - w_f^1)) + \sum_{i=1}^{m-1} \sum_{u \in P_i, v \in P_j, i<j} (f_m(v,u)(w_m^2 - w_m^1) + f_f(v,u)(w_f^2 - w_f^1)). \quad (10)$$

수식 (5)와 (6)을 수식 (10)에 순서대로 적용하면,

$$Cost_{total}(\Pi) = Cost_{constant}(\Pi) + \sum_{i=1}^{m-1} \sum_{u \in P_i, v \in P_j, i<j} (r(u,v) + r(v,u)) = Cost_{constant}(\Pi) + Rcost_{inter}(\Pi). \quad \square$$

(그림 3)(a)는 $w_m^2 - w_m^1 = 2$, $w_f^2 - w_f^1 = 1$인 경우, 수식 (5)를 사용해 (그림 1)에서 주어진 빈도행렬로부터 만들어진 상대비용행렬이다. 따라서, $\Pi = \{\{1,2\}, \{3,4\}, \{5,6\}\}$에 대하여 원래의 문제와 동일한 최적화 문제는 어두운 영역의 모든 원소의 합인 $Rcost_{inter}(\Pi)$를 최소화하는 문제와 동일하게 되는데, 위의 예에서는 $Rcost_{inter}(\Pi) = 300$이다.

이는 $Cost_{constant}(\Pi) = Cost_{intra}(\Pi) + Upper_{intra}(\Pi) + Lower_{intra}(\Pi) = 44 + 96 + 97 = 237$ 이므로 $Cost_{total}(\Pi) = Cost_{constant}(\Pi) + Rcost_{inter}(\Pi) = 237 + 300 = 537$ 이 되기 때문이다.



(그림 3) 누적비용행렬의 생성

THEOREM 1. 만일 Π가 $Rcost_{inter}$를 최소화하는 최적분할이라면, Π는 $Cost_{total}$을 최소화하는 최적분할이다.

PROOF. Lemma 1에 의해 자명하다. $\square$

2.3 누적비용행렬을 이용한 문제 변환

(그림 3)(a)에서 보듯이, 모든 $(u,v) \in V \times V$에 대하여 $A(u,v) = r(u,v)$를 만족하는 크기 n의 상대비용행렬 A가 주어졌을 때, 원래의 문제는 행렬 A를 다음의 조건을 만족하는 부행렬 A_{ij} $(i,j = 1, \ldots, m)$으로 분할하는 문제와 같다. 즉, 대각선 방향으로 있는 부행렬 A_{ii}가 정방형이고,

$$Rcost^{A}_{inter}(\Pi) = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} (A^{*}_{ij} + A^{*}_{ji}) \qquad (11)$$

가 최소화되는 조건을 만족하는 부행렬 A_{ij} $(i, j = 1, \ldots, m)$으로 분할하는 문제와 같게 된다. 여기에서 A^{*}_{ij}는 부행렬 A_{ij}에 있는 모든 원소들의 합인데, i 번째 클러스터에서 j 번째 클러스터로의 상대비용을 나타낸다. $P_1 = \{1,2\}$, $P_2 = \{3,4\}$, $P_3 = \{5,6\}$을 갖는 (그림 3)(a)의 예에서, 클러스터간 전체 통신비용은 다음과 같다.

$$\begin{aligned} Rcost^{A}_{inter}(\Pi) &= A^{*}_{12} + A^{*}_{13} + A^{*}_{23} + A^{*}_{21} + A^{*}_{31} + A^{*}_{32} \\ &= 51 + 52 + 48 + 50 + 45 + 54 \\ &= 300 = Rcost_{inter}(\Pi). \end{aligned}$$

A^{*}_{ij}와 A^{*}_{ji}는 모든 가능한 분할에 대해 반복적으로 계산되어야 하므로, 임의의 분할에 대해 수식 $\sum_{j=i+1}^{m} (A^{*}_{ij} + A^{*}_{ji})$의 값을 구할 때 간단히 행렬에 있는 하나의 대응하는 원소로부터 그 수식의 값을 직접 구할 수 있어야 한다. 이를 위하여 본 논문에서는 상대비용행렬 A를 이용하여 누적비용행렬 B를 구한다. 행렬 A의 중앙 대각선상에 있는 원소의 위쪽에 있는 원소들에 대해서는 각 행에 대해 오른쪽에서 왼쪽으로 원소 값을 누적시키고 난 다음, 다시 각 열에 대해 밑에서 위로 원소 값을 누적시킨다. 반면, A의 중앙 대각선상에 있는 원소의 아래쪽에 있는 원소들에 대해서는 각 열에 대해 밑에서 위로 원소 값을 누적시키고 난 다음, 다시 각 행에 대해 오른쪽에서 왼쪽으로 원소 값을 누적시킨다. (그림 3)(b)는 이러한 과정을 보여주고 있는데, 이때 B에 있는 원소는 A의 원소들에 대한 식으로 아래와 같이 표현될 수 있다.

$$B(s,t) = \sum_{k=s}^{t-1} \sum_{l=t}^{n} A(k,l) \quad \text{if } s < t; \qquad (12)$$

$$B(t,s) = \sum_{k=s}^{t-1} \sum_{l=t}^{n} A(l,k) \quad \text{if } s < t. \qquad (13)$$

LEMMA 2. P_i의 크기를 $\sum_{i=1}^{m} d_i = n$을 만족하는 d_i로 두면, $1 \le i \le m-1$인 정수 i가 주어졌을 때,

1) $B(s,t) = \sum_{j=i+1}^{m} A^{*}_{ij}$ if $s < t$,
where $s = d_1 + \ldots + d_{i-1} + 1$
and $t = d_1 + \ldots + d_{j-1} + 1$.

2) $B(t,s) = \sum_{j=i+1}^{m} A^{*}_{ji}$ if $s < t$,
where $s = d_1 + \ldots + d_{i-1} + 1$
and $t = d_1 + \ldots + d_{j-1} + 1$

PROOF. 수식 (12)는 다음과 같이 $(m-i+1)$ 개의 항으로 나누어질 수 있다.

$$\begin{aligned} B(s,t) &= \sum_{k=s}^{t-1} \sum_{l=t}^{n} A(k,l) \\ &= \sum_{k=s}^{t-1} \Big(\sum_{l=t}^{t+d_{i+1}-1} A(k,l) + \sum_{l=t+d_{i+1}}^{t+d_{i+1}+d_{i+2}-1} A(k,l) \\ &\quad + \ldots + \sum_{l=t+d_{i+1}+\ldots+d_{m-1}}^{n} A(k,l) \Big). \end{aligned}$$

각 항은 A_{ij} $(j = i+1, \ldots, m)$에 있는 모든 원소들의 합이므로, 1)의 좌변은 다음과 같이 표현될 수 있다.

$$\begin{aligned} B(s,t) &= A^{*}_{i(i+1)} + A^{*}_{i(i+2)} + \ldots + A^{*}_{im} \\ &= \sum_{j=i+1}^{m} A^{*}_{ij}. \end{aligned}$$

1)과 같은 방식으로 2)를 증명할 수 있다. □

i 번째 클러스터 P_i가 주어졌을 때, Lemma 2는 모든 $j = i+1, \ldots, m$에 대해 P_i에서 P_j까지 상대비용의 합은 $B(s,t)$이고, P_j에서 P_i까지 상대비용의 합은 $B(t,s)$라는 것을 의미한다. 그런데, $B(s,t)$와 $B(t,s)$는 중앙 대각선에 대해 대칭이므로, $B(s,t)$에 $B(t,s)$를 더하여 최적화 문제를 더욱 간단히 할 수 있다. 따라서, $s < t$에 대해서는 $B(s,t) = b_{st} > 0$ 이고, $s \ge t$에 대해서는 $B(s,t) = 0$이 된다. 그러므로, 수식 (11)은 아래와 같이 변환된다.

$$Rcost^{B}_{inter}(\Pi) = \sum_{i=1}^{m-1} b_{st}. \qquad (14)$$

여기서 모든 i에 대해 $s = \sum_{k=1}^{i} d_{k-1} + 1$, $d_0 = 0$, 그리고 $t = \sum_{k=1}^{i} d_k + 1$이다. (그림 3)(c)의 예에서, $Rcost^{B}_{inter}(\Pi) = b_{13} + b_{35} = 198 + 102 = 300 = Rcost^{A}_{inter}(\Pi)$가 된다. 이때,

b_{13}는 클러스터 {1, 2}와 {3, 4, 5, 6} 사이의 비용이고, b_{35}는 클러스터 {3, 4}와 {5, 6}사이의 비용이다. 그러므로, $b_{13}+b_{35}$는 클러스터 {1, 2}, {3, 4}, {5, 6} 사이의 총 비용이 된다.

3. 최적화 알고리즘

3.1 최적 부구조 및 중복 부문제

2절에서 기술한바와 같이, 이제 원래의 문제는 누적 비용행렬 $B=(b_{ij})$ $(1\le i<j\le n)$에서 $(m-1)$ 개의 원소로 구성된 수열 중에서 합이 최소가 되는 수열을 찾는 문제로 변환되었다. 즉, 첫 번째 원소는 b_{1j}들 중에서 선택하고, 만일 b_{ij}가 r 번째 원소로 선택된다면, 다음 가능한 원소는 $j+1\le k\le n-(m-1-r)+1$인 b_{jk}들 중에서 하나가 선택된다. 이 최적 수열에 대한 문제를 LPART(1, n, m)로 표기하고, $\langle b_{1j}, b_{jk}, \ldots, b_{st}\rangle$를 LPART(1, n, m)에 대한 $(m-1)$개 원소들로 구성된 최적 수열이라 하자. 그러면, $\langle b_{jk}, \ldots, b_{st}\rangle$는 LPART($j$, n, $m-1$)의 최적 수열이 되어야 한다. 만약 그렇지 않다면, 합이 $\langle b_{jk}, \ldots, b_{st}\rangle$보다 작은 $(m-2)$ 개 원소들로 구성된 다른 수열 $\langle b_{jl}, \ldots, b_{xy}\rangle$가 존재하게 되고, $\langle b_{1j}, b_{jl}, \ldots, b_{xy}\rangle$는 더 작은 값을 가지는 $(m-1)$ 개의 원소들로 구성된 수열이 되는데, 이는 모순이다. 따라서, LPART(1, n, m) 문제는 최적 부구조 성질(optimal substructure property)과 중복 부문제(overlapping subproblem)들을 가진다. $p\le m-1$인 경우, $1<p<m$인 p에 대하여 LPART(j, n, p)는 $j'=1,2,\ldots,j-1$인 j'에 대하여 LPART(j', n, $p+1$)을 계산하는 동안 N($j-1$, $m-p$) 번 참조하게 된다. 예를 들어, n=9 와 m=5일 때, 노드 1에서 4까지를 분할하는 세 가지 방법인 {{1 ,2, 3}, {4}}, {{1, 2}, {3, 4}}, {{1}, {2, 3, 4}}와 각각 대응되는 LPART(4, 9, 4), LPART(3, 9, 4), LPART(2, 9, 4)를 계산하는 동안에 LPART(5, 9, 3)은 $N(4,2)=\binom{3}{1}=3$번 참조된다. 따라서 우리의 최적 분할문제는 최적 부구조 성질과 중복 부문제를 가지므로, 동적 프로그래밍을 사용하여 최적화 알고리즘을 구현하는 것이 바람직하다.

3.2 순환 정의

이 절에서는 동적 프로그래밍 기법을 사용하기 위해 최적 해를 순환적으로 정의(recursive definition)한다. 노드 $\{j, j+1, \ldots, n\}$이 p 개의 클러스터들로 선형적으로 분할되었을 때의 최적 비용을 $\Phi_j(p)$로 정의하면, 최적 해는 $\Phi_1(m)$이 된다. 또한, 최적 비용 $\Phi_j(p)$는 다음과 같이 정의되어질 수 있다. 만일 $p=1$이면, 노드 j에서 n까지가 하나의 클러스터가 되어 클러스터간 통신비용을 고려할 필요가 없다. 만일 $p\ge 2$이면, $j+1\le k\le n-p+2$인 k에 대하여 각 b_{jk}는 노드 k에서 n까지로 구성된 나머지 $(p-1)$ 개의 클러스터에 대해 첫 번째 클러스터 $\{j, j+1, \ldots, k-1\}$에 대한 비용을 나타내는 첫 번째 원소로 정의된다. 따라서, $\{j, j+1, \ldots, n\}$을 p 개의 클러스터로 분할하는 최적 비용은 모든 k에 대해 b_{jk}와 $\Phi_k(p-1)$의 합들 중에서 최소의 값이 된다. $1\le j\le n-p+1$에 대한 동적 프로그래밍 식은 다음과 같다.

$$\Phi_j(p)=\begin{cases} 0, & \text{if } p=1 \\ \min_{j+1\le k\le n-p+2}\{b_{jk}+\Phi_k(p-1)\}, & \text{if } p\ge 2 \end{cases}. \quad (15)$$

지금부터는 굳이 명시하지 않아도 j와 k는 이 절에서 주어진 것과 같은 범위의 값을 가진다고 가정한다.

3.3 최적비용 계산

이 절에서는 수식 (15)의 순환 정의를 기반으로 최적 비용을 계산하는 알고리즘을 기술한다. 다음 의사코드(pseudocode)는 $1\le i<j\le n$인 누적비용행렬 $B=(b_{ij})$와 클러스터 개수인 m을 입력으로 받는다. $1\le p\le m$인 각 p에 대해 수행하는 동안, 의사코드는 최적비용 $\Phi_j(p)$를 중앙 대각선상에 있는 원소인 b_{jj}에 저장하고, 그 때의 k 인덱스들을 중앙 대각선상에 있는 원소의 아래쪽에 있는 원소인 $b_{(n-p+2)j}$에 저장한다.

```
LPART(1, n, m)

1  for j ← 1 to n
2      b_jj ← 0
3  for p ← 2 to m
4      for j ← n−p+1 downto 1
5          begin
6              b_jj ← ∞
7              for k ← j+1 to n−p+2
8                  begin
9                      sum ← b_jk + b_kk
```

```
10                  if  sum < b_jj
11                      begin
12                          b_jj ← sum
13                          b_(n-p+2)j ← k
14                      end
15              end
16      end
```

LPART(1, n, m) 알고리즘은 우선 라인 1-2에서 $j=1,\ldots,n$인 j에 대해 $b_{jj}\leftarrow 0$을 계산하는데, 이는 $\{j,\ldots,n\}$을 하나의 클러스터로 만들 경우의 최소비용인 $\Phi_j(1)$이 된다. 이후, 라인 3-16에 있는 코드를 수행하면 수식 (15)를 기반으로 $\{j,\ldots,n\}$을 두 개의 클러스터로 분할할 경우의 최소비용인 $\Phi_j(2)$를 계산하고, 같은 방식으로 라인 3-16까지의 코드를 반복 수행하면서 $\Phi_j(m)$까지의 값을 계산한다. 이때, 클러스터의 크기가 적어도 1인 것으로 가정하고 $1\le j\le n-p+1$에 대한 최소비용을 계산한다. p와 j를 어떻게 선택하느냐에 따라 라인 5-16에서 계산되는 b_{jj}는 라인 3-16의 이전 실행에서 이미 계산된 b_{jk}와 b_{kk}의 합에서 구해진다. 이때 일단 k가 결정되면 그것은 $b_{(n-p+2)j}$에 저장된다.

(그림 4)는 $1\le p\le 6$와 $n=6$일 경우 알고리즘의 동작 과정을 보여준다. 특정한 p에 대한 행렬에서 검게 칠한 엔트리들의 값은 $(p-1)$에 대한 행렬에서 중앙 대각선상에 있는 원소와 그 위쪽에 있는 원소들로부터 계산된다. 실제 계산은 하나의 행렬을 사용해서 이루어지나, (그림 4)에서는 p가 증가하면서 행렬이 변화하는 단계를 보여주기 위하여 다른 행렬을 사용한다. 초기화 과정에서 노드간의 누적비용을 가리키는 중앙 대각선 위쪽의 원소들은 변화하지 않고 아래쪽의 원소들은 0으로 초기화된다. $p=1$일 때 모든 대각 원소는 0으로 설정된다. 다음은 $p\neq 1$인 경우, $p=3$에 대하여 b_{11} 원소를 계산하는 방법인데, 이는 $\Phi_1(3)$와 동일하다.

$$\Phi_1(3)=\min\{b_{12}+\Phi_2(2),\ b_{13}+\Phi_3(2),\ b_{14}+\Phi_4(2),\ b_{15}+\Phi_5(2)\}.$$

$k=2$일 때, $p=2$에 대한 행렬로부터

$$\Phi_1(3)=\min\{b_{12}+b_{22},\ b_{13}+b_{33},\ b_{14}+b_{44},\ b_{15}+b_{55}\}$$
$$=\min\{118+98,\ 198+71,\ 220+49,\ 199+27\}=216.$$

따라서, b_{11}=216이고, $b_{51}=2$이다.

3.4 최적분할 계산

LPART(1, n, m)은 단지 최적비용 $\Phi_1(m)$을 계산하므로, 중앙 대각선 원소 아래에 있는 원소에 저장된 k 값을 이용하여 m 개의 클러스터로 구성된 최적 집합을 구해야 한다. $\langle k_1,k_2,\ldots,k_{m-1}\rangle$을 m 개 클러스터로 구성된 최적 분할을 위한 k의 수열이라고 하면, $b_{(n-m+2)1}$은 k_1값을 주고, $b_{(n-(m-1)+2)k_1}$은 k_2 값을, 그리고 $b_{(n-(m-2)+2)k_2}$는 k_3 값을 갖고 있다. 일단 $\langle k_1,k_2,\ldots,k_{m-1}\rangle$이 결정되면, 주어진 노드들은 m 개의 클러스터 $\{1,\ldots,k_1-1\},\{k_1,\ldots,k_2-1\},\ldots,\{k_{m-1},\ldots,n\}$으로 나누어진다. 이를 위한 의사코드 CLUSTERING(1, n, m)은 다음과 같다.

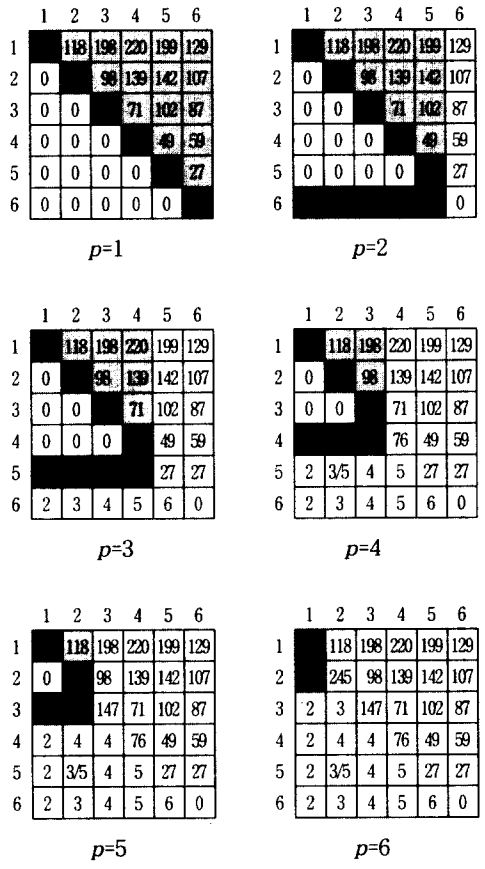
CLUSTERING(1, n, m)

```
1  LPART(1, n, m)
2  CLUSTER ← ∅
3  k_0 ← 1
4  i ← n-m+2
5  j ← 1
6  for l ← 1 to m-1
7      begin
8          k_l ← b_ij
9          CLUSTER ←
           CLUSTER ∪ {k_(l-1), ..., k_l - 1}
10         i ← i-1
11         j ← k_l
12     end
13 CLUSTER ←
   CLUSTER ∪ {k_l, ..., n}
```

(그림 4)의 예에서, 함수 CLUSTERING(1, 6, 3)은 처음에 $k_1=2$를 갖는 b_{51}을 참조한 다음에, k의 다음 값인 $k_2=3$을 얻는다. 따라서, 세 개의 클러스터로 구성된 최적분할은 {{1}, {2}, {3, 4, 5, 6}}이 된다.

일단 $O(n^2)$ 시간에 누적비용행렬을 계산하면, LPART 알고리즘은 $O(mn^2)$ 시간에 최적비용을 계산한다. 라인 6-12에 있는 다른 처리는 $O(m)$을 요구하

므로, 알고리즘 전체 계산시간은 $O(mn^2)$이 된다.



(그림 4) $1 \le p \le 6$인 p에 대하여 LPART(1, 6, p)의 계산 과정

3.5 클러스터 크기의 제약

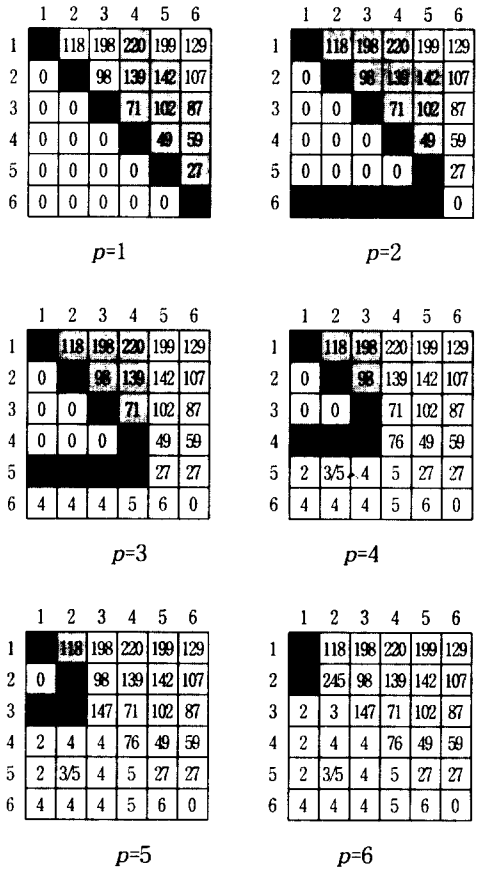
이제 클러스터 크기에 대한 제약조건이 가해질 경우를 고려한다. $m \times s \ge n$을 만족하도록 분할하는 과정에서 허용되는 한 클러스터의 최대크기를 s로 두자. 아래의 의사코드 MLPART(1, n, m, s)는 LPART(1, n, m) 알고리즘의 라인 9에 있는 k 값을 간단히 조정해서 얻을 수 있다. 즉, 제약조건을 만족하는 유효한 k 값을 생성하기 위해, 가장 작은 k 값은 $(j+1)$에서 $\max(j+1,\ n-(p-1)s+1)$까지 변화하고, 가장 큰 k 값은 $(n-p+2)$에서 $\min(n-p+2,\ j+s)$까지 변화한다.

MLPART(1, n, m, s)

```
1   if ms < n
2     return
3   for j ← 1 to n
4       b_jj ← 0
5   for p ← 2 to m
6       for j ← n−p+1 downto 1
7           begin
8               b_jj ← ∞
9               low ← max(j+1, n−(p−1)s+1)
10              high ← min(n−p+2, j+s)
11              if low ≤ high
12                  for k ← low to high
13                      begin
14                          sum ← b_jk + b_kk
15                          if sum < b_jj
16                              begin
17                                  b_jj ← sum
18                                  b_(n−p+2)j ← k
19                              end
20                      end
21          end
```

(그림 5)는 $1 \le p \le 6$에 대해 MLPART(1, 6, p, 3) 알고리즘의 동작 과정을 보여준다. 특정한 p에 대한 행렬에서 회색으로 칠한 원소들은 어떤 클러스터의 크기도 3보다 작거나 같은 제약조건을 만족시키는 유효한 k 값을 나타내는데, 이는 다시 $(p+1)$ 행렬을 계산하는데 사용된다. 예를 들어, $p=1$에 대한 행렬의 첫 번째 행은 $p=2$에 대한 행렬의 b_{11}에 저장되는 $\Phi_1(2)$를 계산하기 위해 사용된다. (그림 4)와는 달리, b_{14}만이 $\Phi_1(2)$를 계산하기 위해 사용될 수 있는데, 이는 다른 원소들을 사용하면 3보다 큰 크기를 가진 클러스터를 만들기 때문이다. 즉, 변수 *low*는 $n-(p-1)s+1=6-(2-1)\times 3+1=4$로 설정되고, 변수 *high*는 $j+s=1+3=4$로 설정됨을 의미한다. 따라서, $\Phi_1(2)=b_{14}+b_{44}=220+0=220$이므로 p=2에 대한 최적분할은 {{1, 2, 3}, {4, 5, 6}}이 된다. $p \ge 4$인 각 p에 대해서는, 제약조건이 k 값의 범위를 제한하지 않기 때문에 최적비용 $\Phi_1(p)$와 최적분할 Π는 (그림 4)의 예와 같다. 이것은 $n-(p-1) \le s$ 조건을 만족할 때, 항상 성립한다.

또한, 한 클러스터의 최대크기인 s와 전체 네트워크에 허용된 최대 통신비용인 정수 K가 주어졌을 때, MLPART(1, n, m, s) 알고리즘은 n 노드들의 모든 가능한 최적분할을 생성한다. (그림 5)의 예에서, n=6, s=3, K=500인 경우를 고려하면, MLPART 알고리즘은 상대비용을 최적화하기 때문에, 우선 상대비용 $K' = K - Cost_{constant} = 500 - 237 = 263$을 계산한 후, MLPART 알고리즘을 사용하여 $\Phi_1(p) \geq K'$일 때까지 $2 \leq p \leq n$인 p에 대하여 $\Phi_1(p)$를 계산한다. $\Phi_1(2) = 220$와 $\Phi_1(3) = 257$이 K'보다 작기 때문에, 가능한 최적분할은 p=2에 대해서는 {{1, 2, 3}, {4, 5, 6}}이고, p=3에 대해서는 {{1}, {2, 3}, {4, 5, 6}}이다.



(그림 5) $1 \leq p \leq 6$인 p에 대하여 MLPART(1, 6, p, 3)의 계산 과정

4. 결 론

본 논문에서는 선형으로 배열된 n 개의 이종 트래픽 소스가 다른 처리비용을 요구하는 여러 종류의 트래픽을 발생할 때, 전체 시스템의 처리 비용을 최소화할 수 있도록 트래픽 소스를 최적으로 분할하는 알고리즘을 제시하였다. 이 알고리즘은 다양한 응용분야에 적용될 수 있는데, 본 논문에서는 이동통신망을 구축할 때 전체 네트워크에 대한 통신비용을 최소화하기 위하여 n 개의 기지국에 m ($1 \leq m \leq n$) 개의 위치정보 서버를 최적으로 할당하는 방법을 논하였다. 이를 위하여, 본 논문에서는 주어진 트래픽이 클러스터 내부에서 발생할 때와 클러스터간에 발생할 때의 통신비용 차이를 반영한 상대비용 개념을 도입하여 $O(mn^2)$의 동적 프로그래밍 알고리즘을 제시하였다. 또한, 이 알고리즘을 이용하여 하나의 클러스터에 대한 크기제한과 전체 네트워크에 허락된 총 통신비용이 제약조건으로 주어질 경우, 같은 계산시간 내에 모든 유효한 클러스터를 찾아 낼 수 있음을 보였다.

참 고 문 헌

[1] Baruch Awerbuch and David Peleg, "Concurrent online tracking of mobile users," *ACM SIGCOM '91*, pp. 221-233, 1991.

[2] Baruch Awerbuch and David Peleg, "Sparse partitions," *1990 IEEE 31th Annual Symp. on Foundations of Computer Science*, pp. 503-513, 1990.

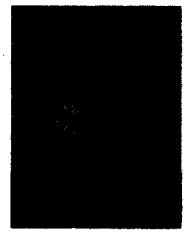
[3] Amotz Bar-Noy and Ilan Kessler, "Tracking mobile users in wireless communications networks," *INFOCOM'93*, pp. 1232-1239, 1993.

[4] T. Imielinski and B. R. Badrinath, "Querying locations in wireless environments," *The 3rd Workshop on Third Generation Wireless Information Networks*, 1992.

[5] Michael R. Garey and David S. Johnson, *Computers and Interactability - A Guide To The Theory of NP-Completeness*, ISBN 0-7167-1045-5, W.H. Freeman and Company, 1979.

[6] Oliver Goldschmidt and Dorit S. Hochbaum, "Polynomial algorithm for the k-cut problem," *The 29th Annual Symp. on Foundations of Computer Science*, pp. 445-451, 1988.

[7] E. Dahlhaus, D. S. Johnson, and C. H. Papadimitriou, "The complexity of multiway cuts," *Proc. of The 24th Annual ACM Symp. on The Theory of Computing*, pp.241-251, 1992.



임 경 식

e-mail : kslim@knu.ac.kr

1982년 경북대학교 전자공학과 졸업(공학사)

1985년 한국과학기술원 전산학과 졸업(공학석사)

1994년 University of Florida at Gainesville, 전산학과 졸업(공학박사)

1985년~1998년 한국전자통신연구원(ETRI) 책임연구원, 실장

1998년~현재 경북대학교 컴퓨터과학과 전임강사

관심분야 : 이동컴퓨터통신, 이동컴퓨팅, 무선인터넷, 고속통신망