모바일 시스템에서 Top-down 방식의 위치데이터베이스 클러스터링 알고리즘

이광조*, 송진우*, 한정석*, 양성봉
*연세대학교 컴퓨터과학과
e-mail : {kjlee5435,fantaros,leohan,yang}@cs.yonsei.ac.kr

Location Database Clustering using Top-down Approach in Mobile Computing Systems

Kwang-Jo, Lee*, JinWoo, Song*, JungSuk, Han*, Sung-Bong, Yang*
*Dept. of Computer Science, Yonsei University

요 약

최근 모바일 기기 사용자의 수가 증가함에 따라 모바일 기기 사용자의 위치정보를 관리하기 위한 기법들이 활발히 연구되고 있다. 기존의 모바일 시스템에서 위치정보를 관리하기 위한 방법으로 two-tier 방식과 two-tier 방식을 개선한 구조적 기법이 제시되었다. 구조적 기법에서는 어떻게 위치 데이터베이스를 군집화시키는 것이 매우 중요하다. 왜냐하면 데이터베이스를 군집하는 방법에 따라 업데이트 비용의 차이가 크기 때문이다. 구조적 기법을 위한 이전 연구는 set-cover 알고리즘을 기반한 bottom-up 방식의 시스템 이다. 본 논문에서는 구조적 기법의 데이터베이스 군집화를 위해 K-means clustering 알고리즘을 기반한 top-down 방식의 시스템을 사용하였고, 실험을 통해 본 논문에서 제시된 방식의 시스템이 기존 방식의 시스템보다 데이터베이스 업데이트측면에서 13.67%의 성능이 향상되었음을 보였다.

1. 서론

최근의 모바일 기기 사용자들은 W-CDMA, UMTS, IMT-2000 과 같은 통신기술에 기반한 다양한 모바일 서비스들을 이용하고 있다[1]. 이러한 모바일 서비스를 위해서 모바일 기기의 효율적인 위치관리가 필수적이다. [2]에서는 기존의 two-tire 방식을 개선한 구조적 관리기법을 제시하였으며, 이 기법은 위치 데이터베이스 (location database, LDB)를 tree 형태로 군집화하는 방식으로서, LDB 의 위치정보 업데이트 비용을 감소시키는 결과를 보여주었다 [2].

[2]에서는 사용자들의 이동패턴을 고려하여 set-cover 를 찾는 greedy 알고리즘을 사용하여 LDB 를 군집화하였다. 이 시스템은 bottom-up 방식으로 군집에 포함될 cell 을 선택한다. 그러나 한번 선택된 cell 들은 다음 선택에서 제외되어 더 '좋은' 군집을 위한 군집간의 조정을 할 수 없는 것이 단점이다.

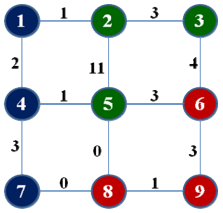
본 논문에서 제안하는 시스템은 K-means 알고리즘을 기반하여 top-down 방식으로 문제를 접근한다. 실험을 통해 본 논문에서 제시된 시스템이 [2]에서 제안된 시스템보다 13.67%의 LBD 업데이트 성능이 향상되었음을 보인다.

본 논문의 구성은 다음과 같다. 2 장에서는 [2]에서 제안된 시스템을 설명하고, 3 장에서는 본 논문에서 제안하는 K-means 알고리즘 기반한 시스템을 살펴본다. 4 장에서는 실험을 통해 성능을 비교 분석한다.

2. 관련연구

[2]에서 제안된 구조적인 LDB 관리 시스템의 군집 형성과정은 set-cover 를 찾는 알고리즘에 기반한다. 일반적으로 set-cover 는 주어진 subset 들 중 몇 개의 subset 을 선택하여, 선택된 subset 들의 합집합이 전체의 합집합과 동일하도록 subset 들을 선택하는 문제이다. 이 set-cover 문제는 NP-완전 문제로서, greedy approximation 알고리즘[3]을 이용하여 근사해를 얻을 수 있다.

본 연구에서는 네트워크를 그래프로 표현하는데, 각 cell 은 그래프의 정점이고, 간선은 두 개의 cell 이 인접됨을 의미한다. 또한 간선에 가중치가 주어지는데 이는 cell 간의 이동횟수를 나타낸다. [그림 1]은 네트워크의 한 예를 그래프로 보여주고 있다.



[그림 1] 네트워크에 대응된 그래프

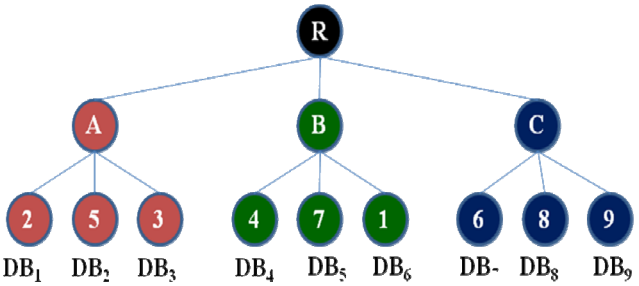
[2]에서 제안한 시스템의 Set-Cover 알고리즘은 다음과 같으며, 이 알고리즘을 최하위 레벨로부터 위의 레벨로 각각 적용하여, 즉 bottom-up 방식으로 tree 를 만든다.

//입력: 그래프 G=(S,E), S ={1,2,3, …, ***n***}이며 E 는 두 인접한 cell 사이의 간선들의 집합, 그리고 ***k***=child 개수
//출력: C_i 집합, 단, ***i*** = 1, 2, … , ***n***/***k***
[1] ***i***=1;
[2] S 로 형성되는 그래프에서 가중치가 가장 큰 간선을 선택하여 간선의 양 끝 cell 들을 C_i 에 추가하고, 추가된 cell 들은 S 에서 제거 한다. 단, S 에서 cell 이 제거되면 그래프상에서 그 cell 에 인접한 간선도 같이 제거된다.
[3] S 에 있는 각 cell 과 C_i에 포함된 모든 cell 들과의 이동횟수를 더한 값 SUM 를 구한다.
[4] SUM 값이 가장 큰 cell 을 C_i에 추가한다. 만약 동일한 값이 여러 개인 경우, 그 중에서 랜덤하게 하나의 cell 을 선택하여 C_i 에 추가한다. 추가된 cell 은 S 에서 제거된다. C_i에 포함된 cell 의 개수가 child 개수보다 작으면 더 추가하기 위해 go to step [3].
[5] S 가 NULL 이 아니면 ***i*** ++, go to step [2].

[알고리즘 1] Set-Cover 알고리즘

[그림 1]의 그래프를 [알고리즘 1]을 통하여 child 개수가 3 인 LDB 를 생성하는 과정을 살펴보면 다음과 같다. 초기의 S = {1,2,3,4,5,6,7,8,9}이고, S 로 이루어진 그래프에서 가장 큰 가중치를 가진 간선은 (2,5)이므로 C_1={2,5}, S={1,3,4,6,7,8,9}가 된다. C_1 의 개수가 child 개수인 3 보다 작으므로, S 의 각 cell 과 C_1의 cell 과의 이동횟수 SUM 을 구한다. 예를 들어, 1 과 {2,5}와의 이동횟수 SUM 은 1 이고, 3 과 {2,5}와의 이동횟수의 SUM 은 3 이며, 6 과 {2,5}와의 이동횟수 SUM 은 3 이다. 또한 7, 8, 9 각각의 {2,5}에 대한 SUM 은 0 이다. 여기서 가장 큰 SUM 값을 가진 cell 은 3 과 6 인데, 둘 중 임의로 3 이 C_1에 추가되며, 동시에 S 에서 제거된다. 즉, C_1={2,5,3}, S={1,4,6,7,8,9}가 된다.

C_1의 개수가 3 이 되었으므로 ***i*** 를 1 증가시키고 C_2를 [알고리즘 1]의 step [2]-[4]를 따라서 구해보면 C_2={4,7,1}이 되고, S={6,8,9}이 된다. 마지막으로 C_3는 그래프에 남아있는 3 개의 cell 들로 구성된다. 따라서 [알고리즘 1]의 최종출력은 C_1={2,5,3}, C_2={4,7,1}, C_3={6,8,9}가 된다. 이 결과는 [그림 2]의 tree 에서 최하위 레벨과 같이 표현된다. 상기 과정을 그 다음 레벨에 대하여 반복 수행한다. 즉, {A,B,C}가 있는 레벨에 대해서도 [알고리즘 1]을 적용한다.



[그림 2] [알고리즘 1]을 [그림 1]에 적용한 군집 결과

구조적인 방법의 시스템에서의 LDB 의 위치정보 업데이트 비용 계산은 [2]에서 제시한대로 다음과 같이한다. 단, 최하위 레벨의 DB 는 좌에서 우로 DB_1, DB_2, DB_3, … , DB_n 으로 놓는다.

$$Update\_Cost = 2 * DIST(i,j) * Moves(i,j)$$

LDB(*i*) = cell***i*** 가 속해있는 DB 이며,
LCA(DB_i, DB_j) = tree 상에서 DB_i와 DB_j 로부터 최초의 공통된 조상 노드(Least Common Ancestor) 까지의 높이,
DIST(*i*,*j*) = LCA(LDB(*i*),LDB(*j*))이다. 또한,
Moves(*i*,*j*) = cell***i*** 와 cell***j*** 간의 이동횟수이다.

[그림 2]를 예로 들어 Update-Cost 구해보면, Moves(3,6) = 4, Moves(4,1) = 2 이며, LDB(3) = DB_3, LDB(6) = DB_7, LDB(4) = DB_4, LDB(1) = DB_6 이다. DIST 값은 실제 cell 이 속해있는 DB 사이의 최초의 공통된 조상 노드까지의 높이이므로 DIST(3,6) = 2, DIST(4,1) = 1 이다. $Cell_3$ 과 $Cell_6$ 의 Update_Cost = 2*2*4 = 16, $Cell_1$ 과 $Cell_4$ 의 Update_Cost = 2*1*2 = 4 이다.

3. **K-means 를 이용한** Top-Down 방식 시스템

[2]에서 제안된 bottom-up 방식의 시스템은 선택된 cell 들을 그래프에서 제거하기 때문에 [알고리즘 1]의 step [2]의 선택과정에서 제거된 간선들 고려될 수 없다. 본 논문에서 제시한 시스템은 top-down 방식으로 접근하여 처음부터 cell 사이의 거리와 이동횟수를 고려하여 전체적으로 관련이 있는 cell 끼리 군집을 수행하고, 다시 나눠진 군집에서 다시 분류를 해 나아가는 방식을 사용하였다. Top-down 방식을 시스템에 적용하기 위해서 K-means clustering 알고리즘을 이용하였다.

//입력: 그래프 G=(S,E), S ={1,2,3, …, ***n***}이며 E 는 두 인접한 cell 사이의 간선들의 집합, 그리고 ***k***=child 개수
//출력: 군집화된 tree
[1] ***k*** 개의 군집 center 들 선택
먼저 그래프에서 랜덤하게 하나의 cell 을 첫 center 로 선택한 후, 첫 center 로부터 가장 먼 cell 을 다음 center 로, 그 다음 center 는 두 개의 center 들로부터 가장 먼 cell 을 center 로 선택하여, 총 ***k*** 개의 군집 center 들을 선택한다. 단, 두 정점 사이의 거리는 그래프상에서 두 점 사이의 최단경로상의 간

선 수를 의미한다.
[2] k 개의 center 각각에 대해 상대적으로 가까운 cell 을 [식 1]을 이용하여 군집화 한다.
[3] [2]에서 군집화된 결과가 이전의 군집화된 결과를 [식 2]를 이용하여 비교하며, Update_Cost 가 작은 cell 을 선택한다.
[4] k 개의 center 각각에 대하여 같은 군집내의 이웃한 cell (그래프상에서 현 center 의 상, 하, 좌, 우에 위치한 cell 들 중 하나)로 center 를 교체한다. 만일 이웃한 cell 이 없는 경우에는 center 는 교체되지 않는다.
[5] Step[2]-[4]를 δ 회 반복한다. (δ 는 실험을 통하여 정함)
[6] Step[5]가 끝나면서 만들어진 k 개의 군집에 대해서, 각각의 군집을 하나의 cell 로 여겨서 새로운 그래프를 만들어 step [2]-[6]을 수행 한다. 단, 군집에 포함된 cell 의 개수가 1 이면 알고리즘을 중단 한다.

[알고리즘 2] K-means 알고리즘

위의 알고리즘 step [2]에서 사용되는 거리 계산은 아래와 같다.

각 center U_i 와 그래프상의 각 cell x 에 대해서, $K\_dist(U_i, x)$ = U_i 와 x 사이의 거리/ U_i 가 속한 군집에 포함된 cell 들과 x 와의 이동횟수의 합 [식 1]

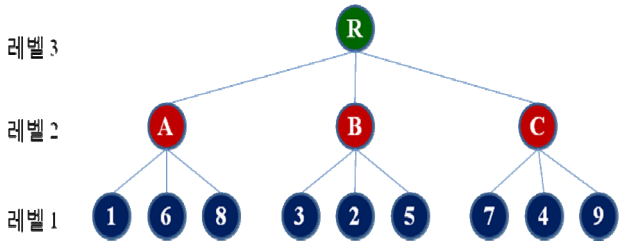
K_dist 는 실제거리와 비례하고, 이동횟수 합에 반비례한다. 즉, 실제로 가까이 있는 cell 일수록 cell 간 이동이 많을 가능성을 반영하기 위함이며, 이동횟수가 많을수록 같은 군집에 포함이 되는 것이 Update_Cost 측면에서 유리하기 때문이다.

Temp_Cost =모든 a,b 에 대하여, $\sum$ 2* Moves (a,b) * (군집화가 수행된 레벨 - 1). 단, a 는 U_i가 중심인 군집의 하나의 cell, b 는 U_i 군집에 포함되지 않은 하나의 cell, k = child 개수, n 은 cell 의 개수, $1 \le i \le k$. [식 2]

Step[3]에서 center 변경 이전과 이후의 Update_Cost 를 비교할 때, 군집화가 최하위 레벨까지 이루어지지 않은 상태에서는 정확한 비용을 계산할 수는 없다. 왜냐하면 기존의 Update_Cost 를 구하는 식에서 DIST(i,j)는 두 cell 의 최초의 공통된 조상까지의 높이이므로, 완성되기 이전의 트리에서는 DIST(i,j)의 값을 알 수가 없기 때문이다. 하지만 만들어지는 레벨 X 에서 만들어지는 군집의 최후의 공통된 부모는 레벨 X 에 존재하므로 cell 간의 최대 높이 차이는 X-1 라는 것을 알 수 있다. 이 최대 높이 차이를 이용하여 worst-case Update_Cost 인 Temp_Cost 를 구하여 step[3]에서 비교하는 것이다.

K-means 알고리즘을 [그림 1]의 네트워크를 입력으로 δ= 1, k=3 일 때 수행해보면 다음과 같다. 초기 입력집합 S={1,2,3,4,5,6,7,8,9} 이다. Step[1]에서 랜덤하게 하나의 center U_1=3 을 선택되었다고 가정하자. U_2 는 U_1 로부터 가장 거리가 먼 cell 로서 7 이 두 번째 center 로 선택되며, 세 번째 center U_3 은 3 과 7 로부터 가장 거리가 먼 cell 로 선택한다. 그러나 이 경우에는 1, 5, 9 가 해당되며, 이 중 하나인 1 을 임의로 U_3 으로 선택하였다고 가정하자.

Step[2]에서는 3 개의 center 1, 3, 7 에 대하여 다음과 같이 군집화를 수행한다. 초기의 세 개의 군집은{**1**}, {**3**}, {**7**}이며, S={2,4,5,6,8,9}로서, S 의 각 cell 과 각 군집의 center 에 대하여 [식 1]을 계산하여 군집화를 수행한다. 예를 들어 cell 2 에 대하여 각각의 center 에 대한 K_dist 값을 구해보면, $K\_dist(1,2)$ = 1/1=1, $K\_dist(3,2)$ = 1/3, $K\_dist(7,2)$ = 3/0=∞이므로, 이 중 가장 가까운 center 는 cell 3 이 되어서 cell 2 는 cell3 이 center 인 군집에 속하게 된다. 이러한 방식으로 S 에 있는 각각의 cell 에 대하여 군집화를 수행하며, 그 결과는 3 개의 군집 {1,6,8}, {3,2,5}, {7,4,9}이다.



[그림 3] K-means 알고리즘의 군집화되는 과정

위의 그림으로 군집화 과정을 살펴보면, 먼저 R(레벨 1)에서 A, B, C 3 개로 군집을 형성하고(레벨 2), 형성된 각 A, B, C 로부터 다시 3 개의 군집을 형성하여 (레벨 3) 전체적인 알고리즘이 진행된다. 다음 장에서는 본 논문에서 제안된 시스템이 Set-Cover 방식의 시스템보다 Update-Cost 측면에서 보다 효율적임을 실험을 통해 보여준다.

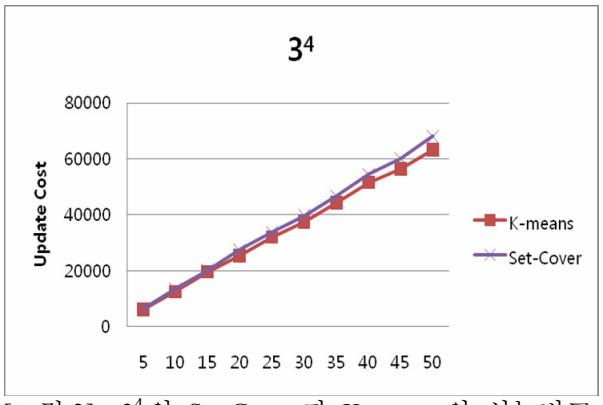
4. 실험

실험을 통해 본 논문에서 제안한 K-means 방식의 시스템과 기존 Set-Cover 방식의 시스템에서의 LDB 업데이트 비용을 비교하였다. [2]에서는 cell 간 평균 이동횟수는 1~10 회, tree 레벨은 4, 6, 8, child 개수는 3 으로 하여 실험하였고, 각 cell 당 평균 사용자의 수는 15 명이다. 본 논문에서도 이와 유사하게 [표 1]에 제시된 환경하에 실험을 하였다. 실험은 Pentium4 Core2Quad Q6600 2.4Ghz CPU 상에서, RAM 은 8GB, OS 는 윈도우 Vista 64Bit 인 시스템을 사용하였다.

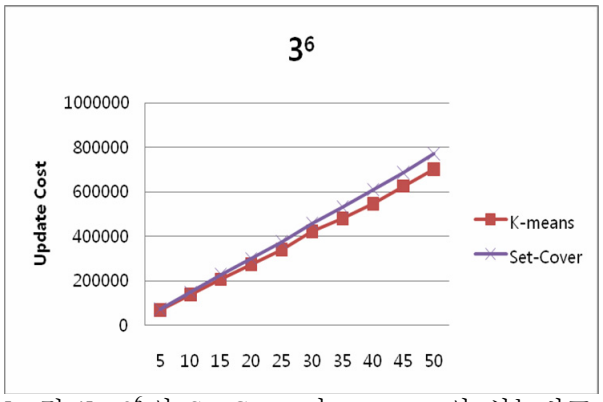
[표 1] 실험 환경 변수

cell 간 평균 이동횟수	5,10,15,20,25,30,35,40,45,50
tree 레벨 수	4,6,8
child 의 수	3
cell 개수	3^4 , 3^6, 3^8
사용자 수	cell 당 평균 15 명

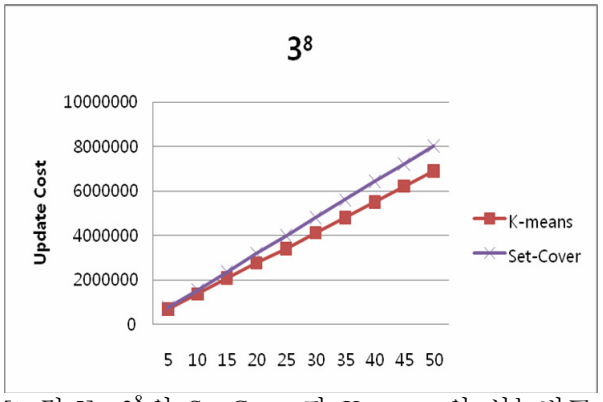
[그림 3]-[그림 5]는 각 레벨 수에 따라 두 시스템의 LDB 업데이트 비용에 대한 실험 결과이다.



[그림 3] 3^4의 Set-Cover과 K-means의 성능비교

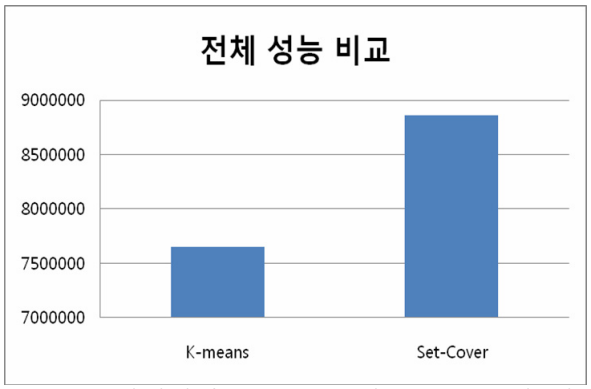


[그림 4] 3^6의 Set-Cover과 K-means의 성능비교



[그림 5] 3^8의 Set-Cover과 K-means의 성능비교

전체적으로 cell 간 이동횟수가 많을수록, cell 의 개수가 많아질 수록 두 시스템간의 성능 차이가 커졌다. Cell 개수가 3^4인 경우 K-means 방식의 시스템이 Set-Cover 방식의 시스템보다 평균 5.87%정도 성능이 향상되었고, 3^6의 경우에는 9.08%, 3^8의 경우에는 14.03% 성능이 향상되었다.



[그림 6] 전체적인 Set-Cover과 K-means의 성능비교

[그림 6]은 전체적으로 두 시스템의 평균 성능을 비교한 결과를 보여주고 있다. 그 결과는 본 논문에서 제안된 K-means 방식의 시스템이 Set-Cover 방식의 시스템의 LDB 업데이트 비용을 13.67% 절감시켰음을 보여준다.

5. 결론

본 논문에서는 기존의 모바일 기기의 위치정보를 관리하는 시스템에서 LDB 업데이트 비용을 감소시키기 위하여 "어떻게 데이터베이스를 관리할 것인가"에 대하여 새로운 방식의 시스템을 제안하였으며, Set-Cover 방식의 bottom-up 군집화 진행상에서 발생되는 문제점을 개선하기 위하여, 보다 융통성 있는 top-down 방식과 K-means Clustering 알고리즘을 이용하여 LDB 업데이트 비용을 13.67% 를 감소시켰다.

참고문헌

[1] A. Samukic, "UMTS Universal Mobile Telecommunications System: Development of Standards for the Third Generation," *IEEE Trans. Vehicular Technology*, Vol. 47, No. 4, pp. 1099-1104, Nov. 1998.

[2] Chen Jixiong, Li Guohui, Xu Huajie, Cai Xia, and Yang Bing, "Location Database Clustering to Achieve Location Management Time Cost Reduction in a Mobile Computing System," *Wireless Communications, Networking and Mobile Computing*, Vol. 2, pp.23-26 , 1328-1332 , 2005.

[3] Sanjoy Dasgupta, Christos Papadimitriou, and Umesh Vazirani, *Algorithms*, McGraw Hill, pp.279-281, 2008.