Journal of Korea Multimedia Society Vol. 10, No. 8, August 2007(pp. 945-957)

병렬처리를 이용한 효율적인 수량 연관규칙

이혜정[†], 박두순[††], 홍 민[†††]

요 약

수량 연관규칙은 대량의 데이터베이스에 존재하는 데이터 중 수량적 속성이 강한 데이터를 항목으로 만들어 이진 연관규칙에 적용한다. 만약 중요한 의미를 내포하는 수량 데이터의 정의역 범위가 넓을 경우 정의역을 최소지지도에 만족하는 적정 구간으로 분할하여 빈발구간 항목을 생성하는 작업이 필요하다. 이러한 빈발구간 항목은 어떻게 생성되었느냐에 따라 생성된 규칙의 신뢰도에 큰 영향을 미치게 된다. 따라서 본 논문에서는 빈발구간 항목을 효율적으로 생성하는 방법을 제시한다. 본 논문에서 제안하는 방법은 기존 방법들에 비해 의미가 있는 구간을 분실하지 않고 최소지지도에 근접하는 세밀한 빈발구간을 생성하기 때문에 데이터가 가진 특성의 손실을 최소화할 수 있는 효율적인 방법이다. 또한 병합이 불필요한 곳에서는 병합을 시도하지 않고 빈도가 높은 구간만을 취해 병합하므로 수량의 정의역이 넓을 경우 기존 방법에 비해 실행속도가 월등히 빠른 효율적인 방법이다. 그리고 인구센서스와 같은 실제로 사용되는 데이터를 이용하여 클루닉스 HPC 시스템에서 병렬처리 수행을 통하여 제안 방법이 우수함을 보였다.

Efficient Quantitative Association Rules with Parallel Processing

Hye-Jung Lee[†], Doo-Soon Park[††], Min Hong[†††]

ABSTRACT

Quantitative association rules apply a binary association to the data which have the relatively strong quantitative attributions in a large database system. When a domain range of quantitative data which involve the significant meanings for the association is too broad, a domain requires to be divided into a proper interval which satisfies the minimum support for the generation of large interval items. The reliability of formulated rules is enormously influenced by the generation of large interval items. Therefore, this paper proposes a new method to efficiently generate the large interval items. The proposed method does not lose any meaningful intervals compared to other existing methods, provides the accurate large interval items which are close to the minimum support, and minimizes the loss of characteristics of data. In addition, since our method merges data where the frequency of data is high enough, it provides the fast run time compared with other methods for the broad quantitative domain. To verify the superiority of proposed method, the real national census data are used for the performance analysis and a Clunix HPC system is used for the parallel processing.

Key words: Data mining(데이터 마이닝), Quantitative Association Rules(수량 연관규칙), Parallel Processing(병렬처리), Large Interval Items(빈발구간항목)

※ 교신저자(Corresponding Author) : 박두순, 주소 : 충남 아산시 신창면 읍내리 순천향대학교 컴퓨터학부(336-745), 전화 : 041)530-1317, FAX : 041)530-1548, E-mail : parkds@sch.ac.kr
접수일 : 2007년 1월 29일, 완료일 : 2007년 6월 21일

[†] 준회원, 순천향대학교 교양과정부 강사
(E-mail : mean1218@sch.ac.kr)
[††] 종신회원, 순천향대학교 컴퓨터학부 교수
[†††] 종신회원, 순천향대학교 컴퓨터학부 전임강사
(E-mail : mhong@sch.ac.kr)

1. 서 론

데이터 마이닝은 대용량화된 데이터베이스에서 데이터들 사이의 연관성을 정보로 변환하는 기술로서 데이터베이스에서 반영하지 못하고 감추어져 있는 규칙성을 발견하여 유용한 정보를 추출하는 것이다. 따라서 데이터 마이닝은 시장 전략 수립, 수요예측, 의료진단 등 광범위한 분야에 유용하게 적용되고 있다[1]. 데이터 마이닝은 목표에 따라 예측과 기술로 나눌 수 있는데, 연관규칙은 기술로 표현되며 정보의 직접적인 활용도가 높고, 고급 정보 가공의 기반 데이터로 활용되기 때문에 많은 연구가 진행되고 있다[2].

연관규칙(association rules)은 하나의 트랜잭션이 다수의 항목을 포함하고 있을 때, 데이터베이스 내의 모든 트랜잭션들에 대해 한 항목들의 그룹과 다른 항목들의 그룹 사이의 연관성을 찾아내는 것이다. 그러나 이러한 연관규칙은 항목의 유무만을 고려한 이진 연관규칙에 관한 연구가 대부분이기 때문에 인구센서스 데이터와 같이 수량적 속성(quantitative attribute)이 강한 자료의 경우, 수량 데이터를 무시한 채 규칙을 생성한다면 의미가 없는 규칙이 된다. 그래서 이진 연관규칙을 수량적 속성을 가진 데이터에도 적용시키는 수량 연관규칙에 대한 연구가 진행되고 있다. 수량 연관규칙은 수량항목의 정의역이 작을 경우엔 각 항목의 값을 독립적인 항목으로 보고 규칙을 생성해도 되지만 정의역 구간이 넓을 경우는 이를 사용자 최소지지도에 만족하는 구간으로 분할하여 빈발구간 항목(large-interval item)을 생성한 후, 각각의 구간을 이진항목으로 취급하여 연관규칙을 탐사하는 방법을 사용한다[3]. 최근에 항목들을 범주데이터(categorical data)와 수량 데이터(quantitative data)로 구분하여 접근하는 연구가 소개되고 있으며[3,4], 수량연관규칙을 위해 수량 항목을 이진 형태로 변환하는 기존 연구[3-5]들이 있다. 본 논문에서는 수량적 속성을 지닌 데이터를 수량 연관규칙에 적용하기 위하여 정의역이 넓을 경우 빈발구간 항목을 효율적으로 생성하는 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 연관규칙과 수량 연관규칙, 그리고 빈발구간 항목 생성 방법에 대한 기존 연구에 대해 살펴보고, 3장에서는 기존 연구의 단점을 보완하여 성능을 개선한 효과적인 빈발구간 항목 생성 방법에 대해 서술한다. 4장에서는 본 논문에서 제안한 방법에 대한 성능평가 환경 및 실제로 사용하는 데이터를 이용한 성능평가 결과에 대해 설명하고 5장에서 결론을 맺는다.

2. 관련 연구

2.1 연관규칙 탐사

연관규칙은 데이터 항목들 간의 조건-결과 식으로 표현되는 유용한 패턴을 말한다. 연관규칙 탐사는 마케팅에서 고객들의 구매행태에 대한 분석 등을 가능하게 하므로 다양한 형태로 많은 곳에서 응용되고 있다. 연관규칙은 n개의 트랜잭션 데이터와 전체 m개의 항목으로 구성되고 이를 I 라 하면, 연관규칙 R은 조건부와 결과부로 구성되며 항목인 X와 Y에 대하여 'X가 발생하면 Y도 발생 한다' 는 의미로 식 (1)과 같이 표현된다.

$$R : X \Rightarrow Y \text{ (단, } X \subset I,\ Y \subset I \text{ 이고 } X \cap Y = \varnothing) \quad (1)$$

따라서 연관규칙은 적절한 항목 X와 Y를 선택하는 문제로 볼 수 있으며 이를 위해 최소지지도(minimum support, S_{min})와 최소신뢰도(minimum confidence, C_{min})라는 척도를 사용한다. 최소지지도는 $S_{min}(R) = supp(X \cup Y)/n$으로 표현되며 집합 X의 항목을 동시에 포함하는 트랜잭션 수의 전체 수(n)에 대한 비율을 의미하고, 최소신뢰도는 $C_{min}(R) = supp X \cup Y)/supp(X)$로 표현되어지며 연관규칙 R의 가치를 평가할 때 통상 최소신뢰도를 사용한다. 이 신뢰도는 조건부 확률의 개념으로 트랜잭션에 X의 항목들을 포함하는 경우 Y의 항목들도 동시에 포함할 확률을 나타내며, 신뢰도가 큰 규칙일수록 의미가 크다고 할 수 있다.

연관규칙을 탐사하기 위해서는 데이터베이스에 있는 모든 빈발 항목들의 지지도를 계산하여 최소지지도를 만족하는 빈발 항목을 찾고 이로부터 주어진 신뢰도를 바탕으로 실제의 규칙을 탐사하는 과정으로 이루어진다. 빈발항목을 찾는 대표적인 알고리즘으로 Apriori가 있다[6]. 연관규칙은 빈발항목을 찾는 단계에서 전체성능이 결정되며 이는 항목의 수가 증가함에 따라 항목의 크기가 기하급수적으로 증가하여 상당량의 처리시간과 메모리를 요구한다. 따라

서 이러한 문제 해결을 위한 연구로 AprioriTid[7], AprioriHybrid[7], DHP[8], 트리 구조를 이용한 방법[9], 그리고 최근에는 다차원 연관규칙 알고리즘으로 일반 데이터베이스에도 이용되고 있다[10]. 또한 Partition[11], DIC[2], Direct Sampling[11], Sampling Approach[12]등의 알고리즘에 대한 연구가 진행 되고 있다. 연관규칙의 대표적인 알고리즘으로 알려진 Apriori는 해당 항목의 발생 유무만을 고려한 이진 연관규칙 탐사 알고리즘이다. 이 알고리즘은 탐사 원리가 간편하고 이해가 용이하여 많은 알고리즘에서 응용하고 있다. 그 외에도 범주적 속성 데이터 뿐 아니라 수량적 속성 데이터의 연관규칙을 찾는 알고리즘[3,4]가 있어 수량 연관규칙에 적용할 수 있다. 또한, 항목 간의 순차적 특성을 찾는 알고리즘[13], 갱신과 유지를 위한 알고리즘, 사용자의 제약을 고려한 알고리즘, 상대적으로 패턴의 길이가 긴 데이터에 대한 연관규칙 알고리즘[14] 등이 있다.

2.2 수량 연관규칙

데이터베이스 내에는 판매 수량이나, 나이, 인구센서스 등 대부분이 수량으로 표현된 데이터들도 적지 않다. 이러한 수량적 속성을 포함하는 규칙을 수량 연관규칙이라 한다. 수량 데이터에 대한 연관규칙탐사가 가능하기 위해서는 먼저 수량 데이터 항목을 이진 항목형태로 변환할 수 있다면 기존의 탐사 알고리즘으로 이진 연관규칙탐사가 가능하다. 따라서 수량연관규칙을 위해 수량 항목을 이진 형태로 변환하는 기존 연구[3,4]들이 있다. 수량적 속성은 구간으로 표현이 가능하고 값으로도 표현이 가능하다. 수량의 정의역의 크기가 작으면 각각의 수량을 독립적인 항목으로 인식하고 값으로 처리해도 되지만 그렇지 않을 경우에는 정의역을 구간으로 분할하여 빈발구간 항목을 생성한 다음 이를 이진 항목으로 대응시켜야 한다. 따라서 의미 있는 일정 구간을 빈발구간 항목으로 처리하는 효율적인 방법이 필요하다. 이를 위한 기존 연구로는 일정간격 분할 및 병합 방법[3]과 유동적 분할 및 병합방법[3], 그리고 최빈수를 이용한 병합방법[4]가 있다.

일정간격 분할 및 병합 방법은 수량항목의 정의영역을 일정한 범위의 작은 구간으로 분할한 후, 이웃한(adjacent) 작은 구간 분할을 병합하여 최소지지도를 만족하는 빈발구간 항목을 생성한다. 이 경우에는 수량 항목의 정의 영역에 데이터가 골고루 분포된 경우에는 효과적이지만, 일부 영역에 집중된 경우는 효율적이지 못하다. 따라서 이에 대한 해결 방법으로 분포도에 따라 분할하는 유동적 분할 및 병합 방법이 있다. 이들 두 가지 방법은 2 단계의 절차를 거쳐 최소지지도를 만족하는 빈발구간 항목을 생성한다. 첫 번째 단계에서는 작은 구간 분할을 생성하며, 두 번째 단계에서는 최소지지도를 만족할 때까지 이웃 작은 구간을 합병한다. 분할과 병합에 사용되는 기준은 일정범위 분할법에서는 최소지지도만 사용하고, 유동적 분할법은 최소지지도와 최소분할지지도를 사용하여 분할 및 병합을 실시한다. 그리고 분할과정이 필요 없이 병합만으로 최소지지도에 만족하는 빈발구간을 생성하는 최빈수를 이용한 병합 방법이 있다. 이 방법은 빈도수가 가장 높은 최빈수 단위구간을 중심으로 좌-우의 단위구간을 병합해나가는 방법이기 때문에 분할기준을 따로 설정할 필요가 없다. 뿐만 아니라 최소지지도를 만족하면 병합을 멈추기 때문에 앞선 두 방법보다 좀 더 밀도 높은 빈발구간 항목을 생성할 수 있는 방법이다.

2.3 빈발구간 항목 생성 방법에 대한 기존 연구

일정간격 분할 및 병합 방법[3]은 수량적 속성을 갖는 정의역을 일정 간격의 작은 구간으로 일괄 분할한 후 분할된 작은 구간을 이웃 작은 구간과 병합하여 빈발구간 항목을 생성하는 방법이다. 여기서 일괄 분할기준은 사용자에 의하여 주어지고, 병합된 구간 중에 사용자가 지정한 최소지지도를 만족하는 구간은 빈발구간 항목으로 처리하고 그렇지 못한 구간은 빈발구간 항목이 아닌 것으로 처리한다. 이 방법은 수량항목의 정의역을 사용자가 지정한 일정한 구간으로 일괄 분할한 후 이웃한 구간을 병합하는 방법을 사용하므로 입력된 수량 항목의 데이터 발생 빈도가 일정한 경우에는 효과적인 방법이다. 그러나 데이터 발생 빈도가 어느 한쪽으로 집중되어 있거나 분산된 데이터일 경우 이들 데이터간의 특성을 반영하지 않고 고정간격으로 분할과 병합을 하기 때문에 분할기준과 데이터 분포에 따라 병합된 구간의 결과가 다르게 나타날 수 있으며 지지도가 월등히 높은 구간을 생성하게 되거나 의미가 있는 구간의 데이터를 분실할 수 있는 문제가 발생하게 된다.

유동적 분할 및 병합 방법[3]은 데이터가 지닌 특

성에 따라 유동적인 간격으로 분할하고 데이터의 집중도를 고려하여 병합하는 방법을 사용하여 데이터의 특성을 반영하도록 하였다. 이 방법은 분할과 병합의 방법을 달리하는데 분할은 이분법을 사용하여 전체 데이터를 한 덩어리로 보고 두 개로 분할하여 최소분할지지도 미만인 부분은 분할하지 않고 최소분할지지도 이상인 부분을 분할한다. 여기서 최소분할지지도는 어떤 구간을 분할할 것인지 하지 않을 것인지 구분하는 지지도이다. 최소분할지지도는 항상 최소지지도 보다 적은 지지도를 설정한다. 분할 과정은 모든 부분이 최소분할지지도 미만인 간격이 될 경우 종료한다. 따라서 넓은 구간이더라도 최소분할지지도 보다 낮은 구간은 더 이상 분할하지 않게 하여 불필요한 분할을 줄인다. 이 방법의 분할은 데이터의 특성을 반영한 분할 방법인 것에 비하여 매 분할시마다 최소분할지지도와 지지율을 비교해야하는 단점이 있다.

최빈수를 이용한 병합 방법[4]는 데이터의 특성을 반영하고 최소지지도에 가장 가까운 밀도 있는 빈발구간 항목을 생성하는 방법이다. 최빈수는 정의역 내 임의의 구간 중의 하나에서 나타나는 가장 높은 빈도를 의미한다. 데이터의 발생이 지역성이 있다고 가정했을 때, 최빈수의 기본구간을 중심으로 하는 주변 데이터의 발생 빈도가 높은 개념을 이용하여 최빈수를 기준으로 최소지지도를 만족할 때까지 기본구간을 병합하여 빈발구간 항목을 생성한다. 빈발구간 항목 생성 방법은 1차 최빈수의 단위구간을 선택한 후, 이를 기준으로 인근(좌-우) 단위구간을 최소지지도를 만족할 때까지 병합한다. 빈발구간 생성방법은 단위구간부터 출발하여 빈발구간 항목을 생성하기 때문에 타 방법과 달리 분할 과정을 별도로 하지 않는다. 그러므로 정의역이 넓은 데이터인 경우에는 병합 과정의 수행시간이 문제가 될 수 있다.

3. 효과적인 빈발구간 항목 생성 방법

3.1 기존 연구의 문제점

일정 간격 분할 및 병합방법은 정의역을 일정 간격으로 일괄 분할한 후 이웃 구간을 병합하는 방법으로 데이터의 분포도가 일정할 경우에는 좋은 성능을 보인다. 그러나 데이터의 분포가 편향적인 지역성을 가지고 있는 데이터일 경우 다음과 같은 문제가 발생하게 된다. 첫째, 지지도가 높은 영역에서 병합할 경우 지지도가 월등히 높은 빈발구간을 생성하여 빈발구간의 개수가 감소하게 되고 결과적으로 데이터의 특성을 반영하지 못하게 된다. 둘째, 빈도가 낮은 영역에서는 불필요한 분할과 병합을 시도하게 되어 효율적이지 못하고 셋째, 빈도가 높은 의미가 있는 구간을 분실할 수 있다. 넷째, 데이터를 분할할 때 분할기준을 어떻게 설정하느냐에 따라 빈발구간 생성 결과가 달라져 분할기준을 설정하기 난해하다.

유동적 분할 및 병합 방법은 일정구간 분할 및 병합 방법이 가지고 있는 단점을 해결하기 위해 제안된 방법으로 불필요한 분할을 줄이고 데이터가 지닌 특성에 따라 유동적인 간격으로 분할하기 위해 이분법을 이용하여 최소분할지지도 이상인 구간만을 분할하고 분할이 가장 많은 구간부터 병합하는 방법을 택하고 있다. 따라서 이 방법은 데이터의 특성을 반영한 방법이다. 그러나 최소분할지지도를 어떻게 설정하느냐에 따라서 생성되는 빈발구간의 결과가 다르기 때문에 다음과 같은 문제점을 가지고 있다. 첫째, 최소분할지지도를 설정하기 난해하고 둘째, 유동적으로 정의역을 분할할 때마다 매번 최소분할지지도와 비교해야하고 분할 횟수를 기억해야 하므로 복잡하다. 셋째, 최악의 경우 생성된 빈발구간의 지지도가 최대 최소분할지지도 만큼 높아지게 되고 그 결과 생성된 빈발구간 항목의 개수가 감소하게 된다. 넷째, 데이터의 특성에 따라 생성된 빈발구간이 이웃한 구간에 위치할 경우 사이에 놓인 구간은 빈도가 높은 구간임에도 불구하고 분실될 수 있는 가능성이 있다.

최빈수를 이용한 병합 방법은 정의역의 분할 없이 빈도수가 제일 높은 최빈수 단위구간을 중심으로 최소지지도에 만족할 때까지 양옆 구간을 병합해나가는 방법으로 세밀한 빈발구간을 생성하고 데이터의 특성을 반영하는 데는 우수한 방법이다. 그러나 이 방법은 첫째, 이미 생성된 빈발구간이 최빈수 단위구간의 양옆에 존재할 경우 의미가 없는 구간임에도 불구하고 분실되는 구간이 다수 발생할 수 있는 문제점이 있으며 둘째, 최소지지도에 만족할 때까지 구간을 병합하기 때문에 지지도가 낮은 의미가 없는 구간에서 병합을 시도할 경우 구간 간격이 지나치게 넓은 의미가 없는 빈발구간 항목을 생성하게 된다. 셋째, 빈발구간을 생성할 때마다 최빈수를 찾아야 하기 때

문에 정의역이 넓을 경우 실행 시간이 오래 걸리는 단점이 있다.

3.2 성능을 개선한 효율적인 빈발구간 항목 생성 방법

본 논문에서 제안하는 방법은 기존 방법의 문제점 보완하여 빠르고 효과적으로 빈발구간 항목을 생성하는 방법이다. 본 방법은 우선 정의역을 이분법으로 분할한 후 분할된 구간을 하나씩 취하여 병합을 시도한다. 이때 병합은 최빈수를 이용한 병합 방법을 이용하되 분할된 구간과 병합된 구간이 일치하지 않을 경우 분할된 구간을 재구성한다. 본 방법의 분할과 병합 그리고 분할 구간의 재구성에 대한 자세한 내용은 다음과 같다.

3.2.1 분할

본 방법에서의 정의역 분할은 병합을 위한 분할이 아니라 최빈수 검색의 시간을 단축하기 위한 분할이다. 정의역 전체를 1로 보고 이분법을 사용하여 분할하되 분할된 구간의 지지도가 [최소지지도*2]보다 크면 분할을 계속 진행하고 [최소지지도*2]보다 작으면 1회만 더 분할하고 더 이상 분할하지 않는다. 왜냐하면 최소지지도에 가까운 구간으로 분할함으로써 불필요한 분할을 줄이고 병합의 효율을 최대한 증대시키기 위해서이다. 분할된 구간은 빈도가 높은 영역에서는 구간의 간격이 좁고 지지도가 높게 나타난다. 따라서 빈발구간 생성 시 구간 간격이 좁고 지지도가 높게 나타나는 구간을 우선적으로 취하여 최빈수를 검색하게 하므로 검색시간을 단축할 수 있을 뿐만 아니라 구간의 빈도가 높은 곳에서 빈발구간을 생성하므로 데이터의 특성을 좀 더 반영한 빈발구간을 생성할 수 있게 된다.

3.2.2 병합

병합은 분할된 구간 내에서 최빈수 단위구간을 검색하여 이를 중심으로 최소지지도에 만족할 때 까지 병합한다. 본 방법에서는 최빈수의 단위구간의 검색 시간과 횟수를 줄이기 위해 분할된 분할구간의 정보를 이용한다. 분할된 구간 중 구간의 지지도가 가장 높고 구간 간격이 가장 작은 밀도 높은 구간을 취한 후 해당 구간 내에서만 최빈수 단위구간을 검색한다. 따라서 정의역 전체에서 최빈수를 검색하여 병합하는 것 보다 실행 시간을 상당량 절약 할 수 있을 뿐만 아니라 병합의 효율성을 극대화 할 수 있다. 즉 밀도 높은 구간에서 병합을 시도하므로 좀 더 많은 수의 빈발구간을 생성하고 의미가 있는 구간을 분실하지 않기 때문에 데이터의 특성을 최대로 반영할 수 있게 된다.

그림 3-1은 분할된 구간 중 지지도가 가장 높고 구간간격이 가장 작은 구간을 취하는 Get_block 함수의 알고리즘이다. 우선 현재 구간의 지지도(*cmp*_sum)와 다음 구간의 지지도(*Block*[i].sum)를 비교하여 지지도가 가장 큰 구간을 검색하고 만약 지지도가 동일할 경우 구간 간격(*cmp*_size, *Block*[i].size)을 비교하여 간격이 가장 작은 구간의 위치정보(*small*_idx)를 리턴 한다. 선택된 구간은 flag값을 1로 수정하여 다음에 구간을 검색할 때 검색 대상에서 제외시킨다.

위와 같이 Get_block 함수를 통해 취한 구간에서 지지도가 가장 높은 최빈수 단위구간을 검색하게 되면 병합을 시도한다. 병합은 최빈수 단위구간을 중심으로 좌-우의 단위구간을 병합하여 최소지지도(S_{min})를 만족하면 중단한다.

그림 3-2는 최빈수의 단위구간을 중심으로 최소지지도를 만족할 때까지 인접 좌-우의 단위구간을 병합하는 절차를 수행하는 알고리즘을 나타낸다[4]. 이때 상한과 하한 경계의 인접 여부에 따라 4가지 경우를 고려하고 있다. Case 1은 상한 및 하한의 한계가 없는 경우이다. 좌-우의 단위구간(lq_{i-j}, lq_{i+j})의

```
Function Get_block
//blcok_count : 분할 구간의 개수 //cmp_sum:구간 지지도 //cmp_size:구간 간격
  for (i=0 ; i < blcok_count ; i++) do begin
    if (Block[i].flag == 0)
        case 1 : cmp_sum < Block[i].sum
                 cmp_sum = Block[i].sum;
                 cmp_size = Block[i].size; small_idx = i
        case 2 : cmp_sum == array_block[i].sum
                 if (cmp_size > Block[i].size)
                     cmp_sum = Block[i].suml;
                     cmp_size = Block[i].size;
                     small_idx = i
                 endif
    endif
  end
  if (small_idx != -1) Block[small_idx].flag = 1
  return small_idx;
```

그림 3-1. 밀도 높은 구간을 취하는 Get_block함수

```
Function Gen_FL
 for (j=1; Max_lq ≥ Smin, j++) do begin
 case 1 : lq_ti-j, and lq_ti+j are not tagged
      if (f(lqi-j)+f(lqi+j)) ≤ (Smin-Max_lq)
                then Max_lq = Max_lq + f(lqi-j)
                +f(lqi+j);
                      flk merge lqi-j, lqi+j lq_ti-j,
                      lq_ti+j = tag
           else if (f(lqi-j)≤f(lqi+j) and (Smin- Max_lq)
           ≤ f(lqi-j))
                then Max_lq = Max_lq + f(lqi-j); flk
                merge lqi-j lq_ti-j = tag
                else Max_lq = Max_lq + f(lqi+j); flk
                merge lqi+j lq_ti+j = tag
           endif
      endif
 case 2 : lq_ti-j is not tagged,lq_ti+j tagged
      Max_lq = Max_lq + f(lqi-j) ; flk merge
      lqi-jlq_ti-j = tag
 case 3 : lq_ti-j is tagged,lq_ti+j not tagged
                Max_lq = Max_lq + f(lqi+j) ; flk merge
                lqi+j lq_ti+j = tag
 case 4 : lq_ti-j andlq_ti+jis tagged
                return // flk is not large
 end
Return
```

그림 3-2. 빈발구간 생성을 위한 단위구간 병합 함수

지지도를 병합하되, 최소지지도에 근접하는 단위구간을 병합한다. Case 2는 상한 경계가 있으므로 하한쪽의 단위구간만을 병합하는 과정이다. Case 3은 Case 2의 반대 경우이며, Case 4는 상하 모두 경계가 있으므로 더 이상 진행할 수 없으므로 리턴 한다. 이때 해당 구간의 지지도가 최소지지도를 만족하면 for 문의 조건에 의하여 빈발 구간이 되지만, 그렇지 않으면 빈발 구간이 아닌 것으로 처리한다. 그러나 빈발 구간이 아닌 구간이 의미가 있는 항목임에도 불구하고 상한과 하한의 경계 때문에 빈발 구간이 아닌 것으로 처리 된 경우라면 그것의 사용여부를 판단하여 의미가 있는 경우 해당 구간을 분할하여 이미 생성된 양 옆의 구간에 병합 시킨다. 만약 의미가 없는 경우 그대로 빈발 구간이 아닌 것으로 처리한다. 만약 병합된 구간의 정보와 분할된 구간의 정보가 불일치 할 경우 분할 구간의 재구성이 필요하다. 분할 구간과 병합구간이 일치하지 않을 경우 구간이 분실되기 때문이다.

3.2.3 분할 구간의 재구성

Get_block 함수를 통해 분할된 구간 중 가장 밀도 높은 구간을 취해 빈발구간을 생성했을 때 분할된 구간의 길이와 병합된 구간의 길이가 불일치하는 경우 분할 구간의 재구성이 필요하다. 그림 3-3은 불일치하는 최악의 경우를 나타낸다. 그림 3-3을 보면 사용된 구간 A가 있고 사용되지 않는 구간인 B, C, D가 있다. 구간 B를 취해 빈발구간 B′를 생성하였을 경우 병합된 빈발구간 B′가 사용 되지 않는 구간인 C의 일부를 침범하고 있다. 그림에서 번호는 작업의 순서를 나타내며, 이 경우 분할 구간의 재구성은 다음과 같이 한다.

구간 B를 취해 최빈수 단위구간을 중심으로 빈발구간 B′를 생성한 결과 사용되지 않는 구간 C를 침범하여 빈발구간이 생성되었으므로 구간 D와 구간 C

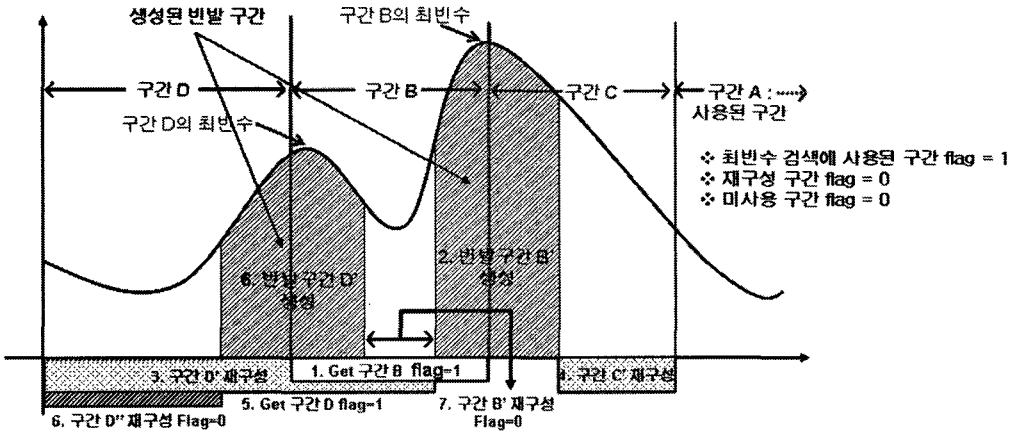

그림 3-3. 분할 구간의 재구성 예

를 재구성한다. 구간 B에서 병합된 구간을 제외시킨 나머지 구간을 구간 D에 병합하여 구간 D'를 재구성한 후 구간 C에서 병합된 구간을 제외한 나머지 구간을 구간 C'로 재구성한다. 그 다음은 재구성된 구간 D'가 Get_block함수를 통해 선택되어지고 빈발구간 D'가 생성되어지면 구간 D'의 양쪽 구간이 사용되지 않는 구간으로 남게 되므로 이들 구간이 다시 구간 D''와 구간 B'로 재구성되어지고 각각의 flag값은 다시 0으로 바꾸어 사용되지 않은 구간으로 표시한다.

분할 구간을 재구성하는 Mod_block 함수는 그림 3-4와 같다.

그림 3-4에서 생성된 빈발구간이 이웃한 구간을 침범하였을 경우 이웃한 구간이 아직 한 번도 취하지 않은 구간이면 구간의 시작 위치와 끝 위치를 재구성하고, 이미 사용된 구간이면 남은 구간을 재구성한 후 사용이 안 된 구간으로 표시한다. 사용된 구간과 사용이 안 된 구간은 flag값으로 표시한다. 사용이 안 된 구간은 0으로 사용된 구간은 1로 표시하고 블록을 취할 경우 flag값이 0인 구간만 검색한다.

```
Function Mod_block
// Block[offset] 현재 최빈수 검색 구간
// start_pos : 현재 구간의 시작 위치 // end_pos : 현재 구간의 끝 위치
// l_low : 현재 구간에서 생성된 빈발항목의 시작 위치
// l_up : 현재 구간에서 생성된 빈발항목의 끝 위치

  if(start_pos != l_low)
      if(Block[offset-1].flag == 0)
          then Block[offset-1].end_pos = l_low -1
              if(Block[offset-1].start_pos > l_low -1)
                then Block[offset-1].flag=1  endif
          else Block[offset].end_pos = l_low -1
              Block[offset].flag = 0  endif
  endif

  if(end_pos != l_up)
      if(block[offset+1].flag == 0)
          then block[offset+1].start_pos =l_up +1
              if(Block[offset+1].end_pos < l_up +1)
                then Block[offset+1].flag=1  endif
          else block[offset].start_pos = l_up +1
                block[offset].flag = 0  endif
  endif
```

그림 3-4. 분할 구간의 재구성 알고리즘

그림 3-5는 제안한 방법으로 빈발구간 항목을 생성하는 전체적인 알고리즘이다. 먼저 Get_block함수를 통해 분할 구간의 정보를 받아 최빈수 단위구간을 검색한다. 선정된 최빈수 단위구간은 Gen_F_L함수에 그 값을 전달하여 최소지지도를 만족할 때까지 인접 좌-우의 단위구간을 병합하는 절차를 수행한다. 만약 병합된 구간이 비빈발구간일 경우 해당 구간이 의미가 있는지 여부를 판단하여 의미가 있으면 병합된 구간을 분할하여 생성되어진 이웃한 빈발구간에 병합시킨다. 모든 절차 수행 후 병합된 구간과 Get_block을 통해 얻어진 구간의 정보가 불일치 할 경우 Mod_block함수를 통해 구간을 재구성한다. 재구성된 구간은 사용여부에 따라 flag=1 또는 flag=0으로 하여 최빈수 검색 시 검색 대상 구간인지 아닌지를 판단한다.

```
// User specifies Minimum Support (Smin)
// Create f(lq) through search DB
 FL= ∅

 CALL Partition_block   // 정의역 분할
 for (k=1 ; Lq≠∅ k++) do begin
    CALL Get_blcok // 밀도 높은 구간 선택
    Max_lq = MAX(f(lqi)) ; (not tagged, Get_block[].
    start_pos ≤i≤ Get_block[].end_pos )
    //해당 블록 내에서 최빈수(Max_lq) 검색
    flk merge lqi
    CALL Gen_FL // 빈발구간 생성 // 단위구간 병합
    FL = ∪ flk // 빈발구간 항목(FL)에 생성된 빈발구간(flk) 포함
    Max_lq = 0
        if 병합된 구간(lqs) < Smin  //
          then lqs가 의미가 있는 구간인지 판단
            if lqs가 의미가 있는 구간이면
              then lqs를 두 구간으로 분할
                  FL = ∪ FLi, FLj //양옆에 위치
                  한 빈발구간(FLi, FLj)에 병합
            else lqs가 의미가 없는 구간이면
              lqi = tagged (1≤i≤n) // not large
              Interval
          endif
        CALL Mod_block //분할된 구간 재구성
    endif
 end
```

그림 3-5.제안 방법의 빈발구간 항목 생성 알고리즘

4. 성능평가

4.1 성능평가 환경

본 논문에서 제안한 방법의 우수성을 보이기 위하여 4가지 방법, 즉 일정간격 분할 및 병합 방법(M1), 유동적 분할 및 병합 방법(M2), 최빈수를 이용한 병합 방법(M3), 본 논문에서 제안한 방법(M4)을 사용하여 생성되는 빈발구간 항목 수와 그들의 구간 평균 간격, 그리고 실행속도를 비교한다. 성능평가 결과 생성된 빈발구간 항목의 수가 많고, 이들의 간격이 좁게 나타나는 방법이 우수한 방법이기 때문이다. 또한 각 방법들의 실행시간을 측정하여 각 방법들이 얼마나 효율적인지를 알아볼 수 있다.

성능평가에는 실제로 사용되는 다음의 데이터들을 사용하였다.

i) 인구주택 총 조사 중 대전광역시의 연령별 인구 데이터 1,214,327 레코드

ii) 인구주택 총 조사 중 대전광역시 가구의 주택 연건평 데이터 269,535 레코드

iii) 사업체 기초통계조사 중 대전광역시 종사자 규모별사업체 데이터 88,869 레코드

이들 데이터는 정의역의 어느 한쪽에 빈도가 편중된 특성을 가지고 있다. 이를 지역성(locality)라고 하는데 이들의 분포는 그림 4-1, 그림 4-2, 그림 4-3과 같다. 그림 4-1의 연령별 인구 분포는 정의역의 50%인 '1세~52세'까지의 빈도가 전체 빈도의 71%를 차지하고 있다. 그림 4-2는 정의역의 50%가 전체 빈도의 97%를 차지하고 있으며 그림 4-3은 정의역의 50%가 99.5%를 차지하는 데이터의 편중이 가장 심한 데이터이다. 따라서 이들 데이터의 발생빈도 지역성은 그림 4-1, 그림 4-2, 그림 4-3 순서로 지역성이 크다. 특히 그림 4-2의 데이터는 빈도가 낮은 의미가 없는 구간이 정의역의 대부분을 차지하고 있기 때문에 의미가 없는 빈발구간 항목을 생성할 가능성이 있는 데이터이다. 본 논문의 성능평가에 사용한 최소지지도는 9가지(40%, 35%, 30%, 25%, 20%, 15%, 10%, 5%, 3%)를 사용하였으며, M1에 대하여는 분할간격을 2로 하고 M2에 사용하는 최소분할지지도는 최소지지도의 1/2로 하였다.

4.2 생성된 빈발구간 항목의 수 비교

생성된 빈발구간 항목의 수가 많다는 것은 그만큼

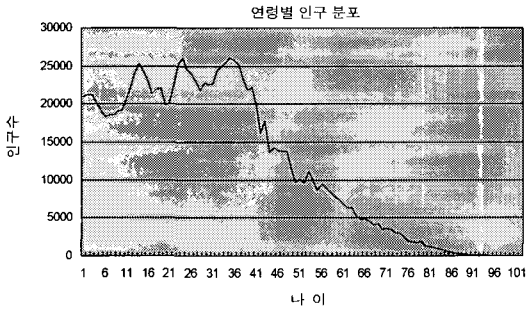

그림 4-1. 연령별 인구 데이터

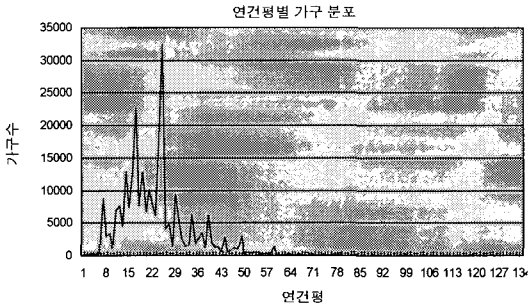

그림 4-2. 가구의 주택 연건평 데이터

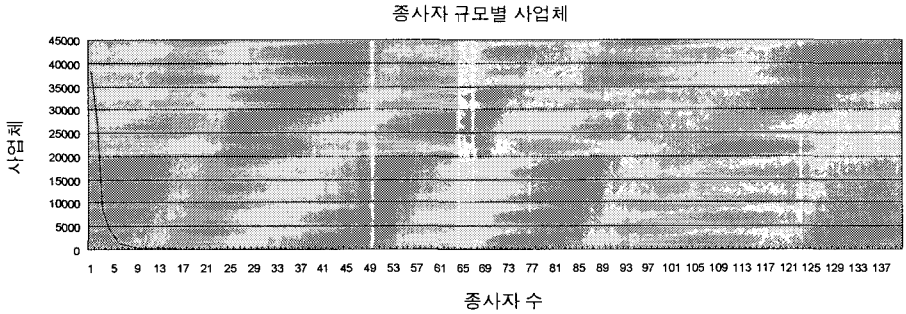

그림 4-3. 종사자 규모별 사업체 데이터

세밀한 빈발구간을 생성하는 우수한 방법임을 뜻한다. 각 방법으로 생성한 빈발구간 항목의 수는 그림 4-4, 그림 4-5, 그림 4-6과 같다. 이들 그래프의 전체적인 양상을 보면 최소지지도가 낮을수록, 특히 15% 이하에서 M3과 제안한 방법인 M4가 타 방법에 비해 많은 수의 빈발구간 항목을 생성하는 것을 볼 수 있다. 이는 최빈수를 이용한 방법이 빈발구간을 생성하는 효과적인 방법임을 보인다. 그러나 지역성이 강한 데이터인 그림 4-5와 그림 4-6에서 M3이 제안한 방법인 M4보다 많은 수의 빈발구간 항목을 생성한 것처럼 보이는데 이는 M3이 빈도수가 낮은 정의역에서 생성된 의미가 없는 병합구간조차 빈발구간 항목으로 처리하였기 때문이다. 이렇게

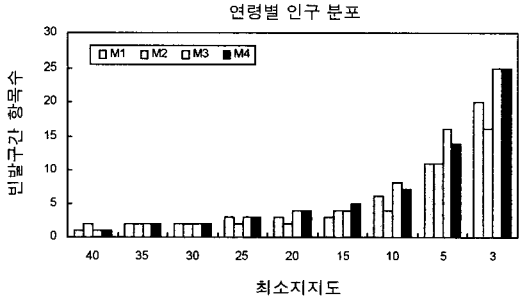

그림 4-4. 데이터 i)에 대한 생성된 빈발구간 항목의 수

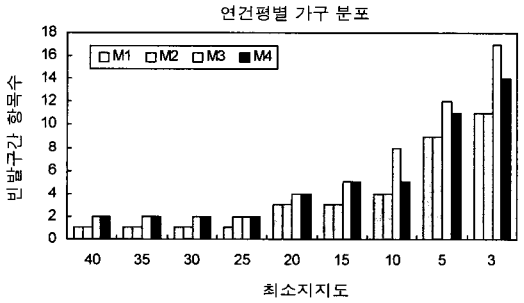

그림 4-5. 데이터 ii)에 대한 생성된 빈발구간 항목의 수

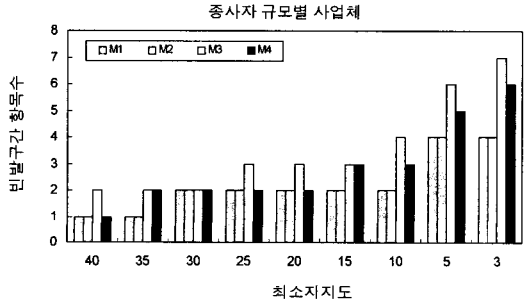

그림 4-6. 데이터 iii)에 대한 생성된 빈발구간 항목의 수

구간의 길이가 지나치게 넓은 병합구간은 의미가 없어 빈발 구간이 아닌 구간으로 처리해야 한다. 특히 정의 영역이 비교적 고루 분포된 데이터인 그림 4-4의 경우 최소지지도 15%에서 제안한 방법인 M4가 가장 많은 수의 빈발구간 항목을 생성하고 있는데 이는 제안 방법이 효율적인 방법임을 의미한다.

이상과 같은 성능평가를 통하여 제안한 방법이 기존의 방법보다 좀 더 세밀하고 최소지지도에 가까운 빈발구간 항목을 생성하고 있으므로 우수한 방법이며 지역성이 심하고 최소지지도가 낮을수록 더욱더 좋은 성능을 보임을 알 수 있다.

4.3 구간 평균 간격 비교

생성된 빈발구간 항목의 수로 얼마나 세밀한 빈발구간을 생성했는지를 예측할 수 있지만 생성된 빈발구간 항목의 구간 평균 간격을 비교함으로써 더욱더 정확한 성능평가를 할 수 있다. 생성된 빈발구간 항목의 평균 간격이 좁게 나타난다는 것은 그 만큼 최소지지도에 가까운 세밀한 빈발구간 항목을 생성한 것이며 데이터의 특성을 반영한 방법임을 뜻한다. 각 방법으로 생성된 빈발구간 항목의 구간 평균 간격은 그림 4-7, 그림 4-8, 그림 4-9와 같다. 그래프의 결과는 각 방법에서 생성된 빈발구간 항목의 개수에 상관없이 해당 최소지지도에서 생성된 빈발구간 항목들의 평균 구간 간격을 나타낸다. 따라서 생성된 빈발구간 항목의 수를 감안하여 비교해야한다. 구간의 평균 간격이 넓다는 것은 그만큼 생성된 빈발구간의 수가 적고 지정한 최소지지도 보다 높은 빈발구간 항목이 생성되었다는 것을 의미하기 때문이다. 그러나 생성된 빈발구간 항목의 수가 많음에도 불구하고 구간의 평균 간격이 넓게 나타났다는 것은 간격이 지나치게 넓은 의미가 없는 빈발구간 항목을 생성했다는 것을 의미한다.

이들 그래프를 보면 대체로 데이터의 분포 특성에 따라 각 방법이 큰 차이가 있음을 알 수 있다. 데이터의 특성을 고려하지 않은 방법인 M1은 전체적으로 구간의 평균 간격이 넓게 나타나고 있다. 특히 지역성이 가장 심한 경우를 제외하고 최소지지도가 높을수록 가장 넓은 빈발구간을 생성하고 있다. 따라서 M1은 지역성이 크고 최소지지도가 높을수록 비효율적인 방법임을 알 수 있다. 데이터의 특성을 고려한 방법인 M2는 M1에 비해 비교적 양호한 구간 간격을

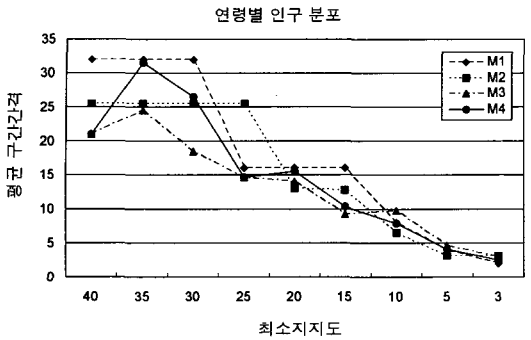

그림 4-7. 데이터 i)에 대한 빈발구간 항목의 평균 구간 간격

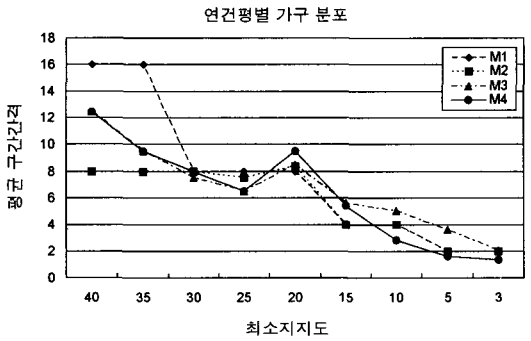

그림 4-8. 데이터 ii)에 대한 빈발구간 항목의 평균 구간 간격

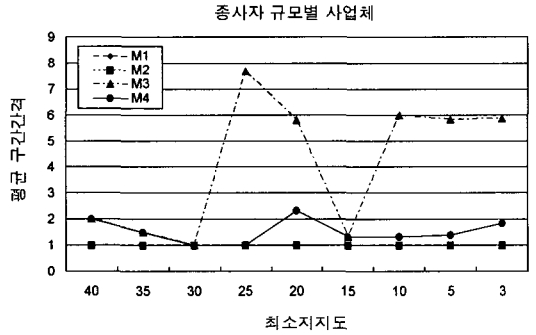

그림 4-9. 데이터 iii)에 대한 빈발구간 항목의 평균 구간 간격

보이고 있다. 따라서 M2는 유동적인 분할과 병합과정에서 이미 병합된 구간 사이의 데이터를 분실할 수 있는 가능성을 배재한다면 지역성에 관계없이 적용시킬 수 있는 방법이다. M3은 병합 방법의 특성상 최소지지도를 만족하면 병합을 중단하기 때문에 대체적으로 구간의 간격이 가장 좁게 나타나지만 그림 4-7의 경우 생성되는 빈발구간의 수에 비해 구간의 간격이 좁은 것으로 보아 분실되는 항목이 다소 있음을 알 수 있다. 또한 지역성이 심한 경우 빈도가 현저히 낮은 영역에서 빈발구간 항목을 생성함으로써 구간의 간격이 월등히 높게 나타나는 양상을 보이고 있다.

제안한 방법인 M4는 전체적으로 타 방법에 비해 생성된 빈발구간 항목의 수가 많고 분실되는 구간이 없음에도 불구하고 비교적 세밀한 평균 간격을 보이고 있다. 특히 지역성의 정도가 심하지 않은 경우에는 최소지지도가 낮을수록 좀 더 세밀한 빈발구간을 생성하고 있다. 그림 4-9의 경우 타 방법인 M1과 M2에 비해 구간의 평균 간격이 약간 넓게 나타나는데 이는 정의역의 대부분을 차지하는 낮은 빈도와 생성된 빈발구간의 수를 많은 것을 감안하면 효율적인 방법임을 알 수 있다.

이상과 같은 구간 평균 간격의 성능평가를 통하여 제안한 방법 M4가 의미가 있는 데이터를 분실하지 않고 세밀한 빈발구간을 생성하는 우수한 방법임을 보였다. 특히 지역성이 심하고 최소지지도가 낮을수록 제안한 방법이 가장 우수한 성능을 나타내고 있다. 반면 데이터의 특성에 따라 차이는 있지만 빈도가 낮은 정의역 구간이 넓게 분포되어있는 데이터에서는 M3이 가장 부적합한 것으로 나타났고 데이터의 특성을 고려하지 않은 M1은 높은 최소지지도에서 구간의 간격이 넓은 빈발구간을 생성하므로 부적합하다. 반면 M2는 데이터의 특성과 최소지지도에 따라 차이는 있지만 데이터의 분실로 생성된 빈발구간 항목의 개수가 적은 것을 배재하면 구간 평균 간격에 있어서는 전체적으로 양호한 성능을 보이고 있다.

4.4 실행 속도 비교

본 절에서는 각 방법에 대해 실행 시간을 비교한다. 프로세서 한 개일 때 실행 시간을 측정하여 비교하고, 프로세서 2개일 때와 4개일 때, 그리고 8개일 때로 프로세서 수를 늘려가며 표 4-1과 같은 클루닉스 HPC 시스템을 사용하여 측정하였다.

속도 측정값에 대한 부드러운 결과를 얻기 위해 본 논문에서 사용한 실제 사용되는 데이터인 인구주택 총 조사의 대전지역 나이별 인구분포, 동일 지역의 주택의 연건평별 가구분포, 사업체 기초통계조사의 종사자규모별 사업체분포에 대한 데이터의 항목수를 랜덤(random) 수를 이용하여 22만 여개로 늘렸다. 그 결과 생성된 데이터의 항목의 수는 데이터 (i)의 경우 204,000개, 데이터 (ii)의 경우 221,100개, 그

표 4-1. 병렬처리 시뮬레이션 환경

구분	항 목	사 양		수량
H/W	Computing Cluster	CPU	AMD-Opteron 246	8
		RAM	DDR-SDRAM(PC2700) 1GB	8
		inter-connection	Fast Gigabyte (T/1000) 1Gbps port	4
		HDD	SATA-HDD 120GB - 총 600 GB 보조기억장치 (공유 스토리지 240 GB)	5
		AUX-D	DVD-RW	1
S/W	ISV compiler, library	compiler	Inter C compiler	1
		library	라이브러리	

리고 데이터 (iii)의 경우 224,000개로 입력 데이터의 항목의 수가 변경되었다. 또한 실행속도 측정값에 대한 오차를 적게 하기 위해 최소지지도마다 각각 4번씩 측정 시간대를 달리 하여 테스트하였고, 결과 값은 측정된 4번의 시간에 대한 평균값으로 하였다. 또한 생성된 프로세스의 개수만큼 데이터를 분할하여 각 프로세스에게 할당한 후 이를 각 프로세스에서 처리하도록 구성하였다. 데이터를 분할하여 각 프로세스에게 나누어 주고 필요한 변수를 각 프로세스에게 전송하는 방법은 일대일 통신을 이용하였고 각 프로세스에서 생성된 빈발구간과 실행시간 등의 결과 취합은 집합통신 서브루틴 방법을 사용하여 병렬 프로그램을 구성하였다.

처리한 결과 실행 시간은 그림 4-10, 그림 4-11, 그림 4-12와 같다. 프로세서 1개로 실행했을 경우 전체적으로 M3의 실행 시간이 월등히 높고 제안한 방법 M4의 실행 시간이 월등히 낮은 것을 볼 수 있다. 프로세서 개수를 증가시킴에 따라 기존의 방법들은 실행 시간이 월등히 감소하는 추세를 보이고 있으나

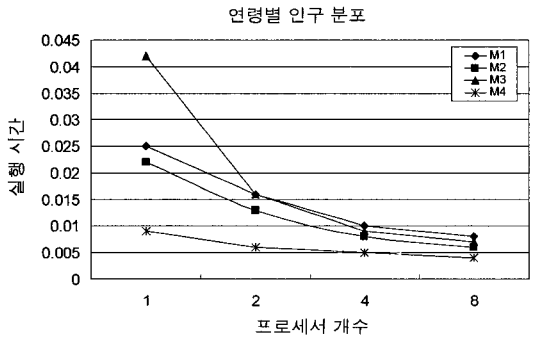

그림 4-10. 데이터 i)에 대한 프로세서 개수에 따른 평균 실행 시간

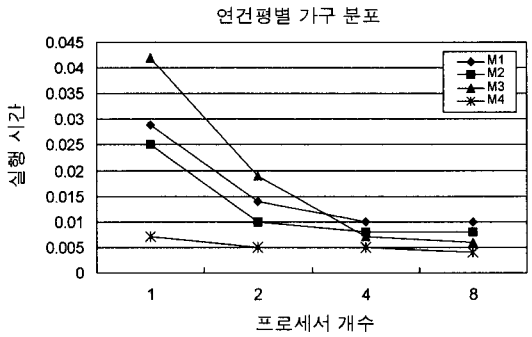

그림 4-11. 데이터 ii)에 대한 프로세서 개수에 따른 평균 실행 시간

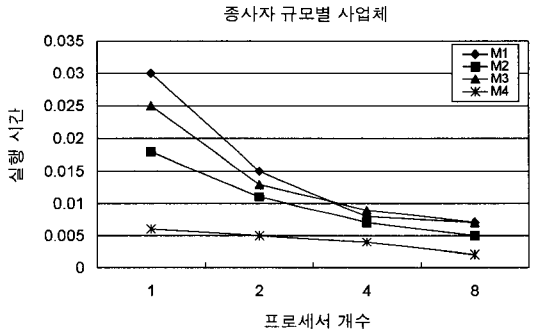

그림 4-12. 데이터 iii)에 대한 프로세서 개수에 따른 평균 실행 시간

M4의 경우, 프로세서 한 개일 때의 실행 시간이 타 방법에 비해 월등히 빠르기 때문에 프로세서의 개수가 증가하더라도 감소하는 비율이 낮게 나타나고 있다. 특히 M3은 프로세서의 수가 증가함에 따라 최빈수를 검색해야 하는 정의역 구간이 1/n로 축소되면서 실행 시간의 감소폭이 크게 나타나고 있다. 그러나 M2의 경우 M1에 비해 불필요한 분할과 병합을 줄였다 할지라도 매번 최소분할지지도와 비교해야

하는 단점을 가지고 있고 약간은 복잡한 알고리즘으로 인하여 실행 시간의 감소 비율이 가장 낮게 나타나고 있다. 반면 M1은 데이터의 특성을 반영하지 않는 방법으로 데이터의 특성과 무관하게 일정한 감소폭을 보이고 있다.

결과적으로 불필요한 분할을 줄인 방법인 M2와 M4가 속도적인 측면에서 가장 우수한 성능을 보이고 있다. 그러나 제안한 방법인 M4의 분할 횟수가 M2보다 적고 병합 시에도 고려할 사항이 M2보다 적기 때문에 M4가 월등히 좋은 성능을 보이고 있다.

5. 결 론

본 논문에서는 수량 항목을 포함하는 대용량의 데이터베이스에서 수량 연관규칙 탐사를 위해 효과적으로 빈발구간 항목을 생성하는 방법을 연구하였다. 본 논문에서 제안하는 방법은 정의역을 최소지지도에 만족하는 구간으로 분할한 후 빈도가 가장 높은 구간을 취해 해당 구간에서만 최빈수를 검색하여 이를 중심으로 빈발구간을 생성하는 방법이다. 따라서 제안 방법은 데이터의 특성을 잃지 않을 뿐만 아니라 빈도가 낮은 의미가 없는 구간에서의 병합을 시도하지 않으므로 실행시간을 상당량 단축할 수 있다. 또한 의미가 있는 구간의 데이터를 분실하지 않기 위해 병합된 구간이 최소지지도에 만족하지 않더라도 의미가 있는 구간일 경우 이미 생성된 이웃한 빈발구간에 이를 병합하도록 하였으며, 비록 빈발구간이라도 간격이 지나치게 넓어 의미가 없는 구간일 경우 빈발구간이 아닌 구간으로 처리하여 규칙에 적용하기 곤란한 빈발구간 항목을 생성하지 않도록 하였다. 그리고 제안하는 방법의 우수함을 보이기 위해 제안 방법과 기존의 방법들을 인구주택 총 조사와 같이 실제 사용되는 데이터에 대해서 생성되는 빈발구간 항목수, 그들의 구간 평균 간격, 그리고 클루닉스 HPC 시스템을 이용한 병렬처리로 실행속도를 성능 평가하였다. 본 논문에서 제안한 방법은 적용된 데이터가 편중된 지역성을 가진 데이터일수록, 그리고 지정된 최소지지도가 낮을수록 좀 더 세밀하고 많은 수의 빈발구간 항목을 생성하였고 실행속도 역시 타 방법에 비해 월등히 빠른 효율적인 방법임을 보였다. 특히 단일 프로세서로 실행했을 경우 타 방법에 비해 4배 이상 실행속도의 단축 효과를 볼 수 있었고 지역성이 심한 데이터일수록 실행속도가 5배 이상으로 단축되는 매우 효율적인 방법을 제안하였다.

앞으로는 효율적인 병렬 수량 연관규칙 방법을 연구할 것이다.

참 고 문 헌

[1] R. Srikant and R. Agrawal, "Mining Generalized Association Rules," *Proc. of the 21st Int'l Conference on Very Large Databases*, Zurich, Switzerland, Sep. 1995.

[2] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining*, AAAI Press, pp. 1-34, 1996.

[3] R. Srikant and R. Agrawal, "Mining Quantitative Association Rules in Large Relational Tables," *Proc. of the ACM SIGMOD Conference on Management of Data*, June 1996.

[4] 박원환, 박두순, "데이터의 지역성을 이용한 빈발구간 항목집합 생성방법," *한국멀티미디어학회 논문지*, 제4권, 제5호, pp. 101-111, 2001.

[5] 최영희, 장수민, 유재수, 오재철, "수량적 연관규칙탐사를 위한 효율적인 고빈도 항목열 생성기법," *한국정보처리학회 논문지*, 제6권 제10호, pp. 2597-2607, 1999.

[6] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large database," *Proc. of the ACM SIGMOD Conference on Management Data*, pp. 207-216, 1993.

[7] R. Agrawal and R. Srikant, "Fast Algorithms for mining association rules," *Proc. of the 20th VLDB Conference*, Santigo, Chile, Sept. 1994.

[8] J. S. Park, M. S. Chen, and P. S. Yu, "An Effective case-based algorithm for mining association rules," *Proc. of ACM SIGMOD Conference on Management of Data*, pp. 175-186, May 1995.

[9] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without candidate Generation," *SIGMOD'00, Dallas*, TX, May 2000.

[10] L. Wenmin, J. Han, and J. Pei, "CMAR: Accurate and Efficient Classification Based on Multiple Class Association Rules," *Proc. of 2001 International Conference on Data Mining*, San Jose, CA, Nov. 2001.

[11] A. Savasere, E. Omiencinsky, and S. Navathe, "An Effective algorithm for mining rules in large databases," *Proc. of the 21st VLDB Conference*, pp. 432-444, 1995.

[12] J. S. Park, P. S. Yu, and M. S. Chen, "Mining Association Rules with Adjustable Accuracy," *Proc. of ACM CIKM 97*, pp. 151-160, Nov. 1997.

[13] M. Zaki, "Efficient Enumeration of Frequent Sequences," *Proc. of ACM CIKM 98*, pp. 278-280, Nov. 1998.

[14] R. Jr. Bayardo, "Efficiently Mining Long Patterns from Databases," *Proc. of the ACM SIGMOD Int'l Conference on Management of Data*, pp. 85-93, June. 1998.

이 혜 정

1997년 순천향대학교 전산학과 (공학사)
2000년 순천향대학교 전산학과 (공학석사)
2006년 순천향대학교 전산학과 (공학박사)
2001년~현재 순천향대학교 교양과정부 강사
관심분야 : 데이터마이닝, 병렬처리, 컴퓨터교육, 멀티미디어 정보처리

박 두 순

1988년 고려대학교 전산학전공 (이학박사)
2004년~2005년 미국 콜로라도대학교 전산학과 객원교수
2002년~2003년 순천향대학교 공과대학 학장
2006년~2007년 순천향대학교 u-Healthcare 연구센터 소장
2000년~현재 한국멀티미디어학회 편집위원 및 이사
1985년~현재 순천향대학교 컴퓨터학부 교수
관심분야 : 병렬처리, 데이터 마이닝, 컴퓨터 교육, 멀티미디어 정보 처리

홍 민

1995년 순천향대학교 전산학과 (공학사)
2001년 미국 콜로라도대학교 전산학과 (공학석사)
2005년 미국 콜로라도대학교 바이오 인포메틱스 전공 (공학박사)
2005년~2006년 미국 콜로라도대학교 컴퓨터공학과 Post-Doc.
2006년~현재 순천향대학교 컴퓨터학부 전임강사
관심분야 : 컴퓨터 시뮬레이션, 바이오 인포메틱스, 컴퓨터 그래픽스