개선된 파시클 기반 의미론적 압축 기법*

박형민[0] 심규석 장원준
서울대학교 전기 컴퓨터 공학부
hmpark[0]@kdd.snu.ac.kr, shim@ee.snu.ac.kr, wjchang@kdd.snu.ac.kr

Improved fascicle based semantic compression

Hyoungmin Park[0], Kyuseok Shim, WonjunChang
School of Electrical Engineering and Computer Science
Seoul National University

요 약

실제 데이터들은 압축이 필요할 정도로 큰 경우가 종종 있다. 일반적인 구조론적 데이터 압축 방법은 테이블을 긴 바이트 스트링으로 취급하여 바이트 레벨에서 압축을 시도한다. 이런 경우에 보통 압축의 효율과 검색의 용이성(특정 레코드의 애프리뷰트 값을 찾아내기 위해 압축을 풀어야 하는 부분의 크기)사이에 교환관계가 발생한다. 이런 점에서 검색을 위해 압축을 풀 필요가 없는 의미론적 압축 방법이 주목을 받고 있다.

이 논문에서는 기존의 파시클(fascicle) 알고리즘을 개선하는 새로운 의미론적 알고리즘을 제시하고 실험을 통하여 제안된 알고리즘의 우수성을 입증한다

1. 서 론

데이터 웨어하우스나 네트워크 모니터링 또는 바이오 정보시스템들과 연동하여 구축되는 많은 정보시스템은 대용량의 다차원 데이터베이스 테이블이 필요한 경우가 많다. 이러한 테이블의 크기는 테라바이트에 이르기 때문에, 이를 효율적이게 저장하는 문제가 대두되었다.

테이블의 크기를 줄이기 위한 방법으로는 고전적인 방법인 통계적이거나 사전적인 압축방법을 이용한, Lempel-Ziv[1,2] 또는 Hoffman[3]이 있다. 그러나 이러한 압축방법은 데이터 테이블을 단순한 크기가 큰 스트링으로 봄으로서 테이블이 가지고 있는 복잡한 연관관계를 설명하지 못하기 때문에, 대용량 데이터 테이블을 압축하는 방법으로 적절하지 못하다. 즉 테이블 전체를 하나로 압축하는 이 방법은 동적으로 질의를 처리하기 위해서는 매번 압축된 전체의 테이블을 압축을 풀어 질의를 수행할 대상인지 아닌지를 확인하는 것은 많은 시간과 공간의 비용이 든다. 따라서 압축을 푸는 일 없이 질의 처리를 할 수 있는 의미론적 압축 방법이 주목을 받게 되었다.

예를 들어, 그림 1의 테이블을 살펴보자. 첫 번째, 세 번째와 여섯 번째 레코드들이 'poor' credit을 가진 male의 그룹에 속한다. 또한 'good' credit을 가진 female들의 그룹은 두 번째, 네 번째, 다섯 번째 레코드들로 구성된다. 데이터 테이블에서 최대한 공통으로 묶을 수 있는 데이터 값들을 대표값으로 저장하여 필요저장공간을 줄이고, 동시에 데이터를 처리하는 추가비용을 최소화 할 수 있다. 이를 위한 데이터 부분집합을 형성하는 방법으로 [4]에서 파시클(*fascicle*)을 제안하였다. 파시클 $F(k,t)$는 데이터테이블 F에서 k개의 에트리뷰트를 압축하기 위해, 각각의 애트리뷰트에 대해 허용치 t를 주는데, 위의 예에서는 t_age=1, t_salary=1000, t_assets=2000, t_credit=0, t_gender=0을 허용하면 두 개의 2-D 파시클이 존재하게 된다. 그림 2는 이 파시클들을 이용하여 테이블을 압축한 결과이다.

age	salary	assets	credit	gender
20	36000	49000	poor	male
21	37000	50000	good	female
29	75000	120000	poor	male
30	76000	122000	good	female
40	119000	200000	good	female
41	120000	202000	poor	male

그림 1 예제 테이블

FID	Uncompressed values
1	20,36000,49000
1	29,75000,120000
1	41,120000,202000
2	21,37000,50000
2	30,76000,122000
2	40,119000,200000

(a) 압축된 테이블

FID	credit	gender
1	poor	male
2	good	female

(b) 파시클들

그림 2 파시클 알고리즘에 의해 압축된 테이블

그런데, 그림 1을 잘 살펴보면 압축시킬 수 있는 여지가 더 있다는 것을 알 수 있다. 예를 들어, 20, 36000, 49000

* 본 연구는 KT(Korea Telecom)의 연구비 지원으로 수행되었습니다.

의 값들은 은 첫 번째 레코드에 나타나고 두 번째 레코드에서 오차 허용치 내에서 다시 나타남을 알 수 있다. 따라서, 이 값들을 파시클로 만들어서 더 압축할 수 있다는 것은 자명하다. 그림 3은 이러한 파시클들을 이용하여 더욱 압축시킨 예제이다. 그러나, 기존의 파시클 알고리즘은 레코드 하나를 압축하는 데 하나의 파시클만을 이용해서 압축하도록 하고 있다. 따라서, 이미 첫 번째와 두 번째 레코드들은 1번 파시클을 이용하여 압축하고 있으므로 다른 파시클을 이용할 수 없다. 이 논문의 주된 아이디어는 하나의 레코드를 압축하는데 여러 개의 파시클을 사용하는 것을 허용함으로써 기존의 파시클의 압축률을 개선하는 것이다.

FIDs
1,3
1,4
1,5
2,3
2,4
2,5

(a) 압축된 테이블

FID	credit		gender
1	poor		male
2	good		female
FID	age	salary	assets
3	20	36000	49000
4	29	75000	120000
5	40	119000	200000

(b) 파시클들

□림 3 다중-파시클 알고리즘에 의해 압축된 테이블

본 논문의 나머지 구성은 다음과 같다. 2장에서 관련 연구에 대해 간단히 설명하고 3장에서 파시클에 대해 자세히 설명하겠다. 4장에서는 파시클을 개선하는 아이디어를 설명하고
5장에서는 실험 결과를 제시한다.

2. 관련 연구

이 장에서는 이미 알려진 의미론적 압축 알고리즘에 대해 간단하게 설명하겠다. [4]에서 제안된 파시클 알고리즘은 테이블에 대해 개발된 첫 번째 의미론적 압축 알고리즘이다. 이 알고리즘은 m개의 컬럼을 가진 어떤 테이블과 유저에 의해 정해지는 $u(u \leq m)$값과 각 컬럼에 대한 허용 오차가 주어지면 w개의 배타적인 레코드들의 집합들을 찾아낸다. 각각의 집합에 속한 모든 레코드들은 어떤 u개의 컬럼에 대해 어떤 대표 튜플로부터 허용오차 내의 값을 가진다. 이 때 이 대표 튜플을 파시클, u개의 컬럼을 컴팩트 애트리뷰트(compact attribute)라고 부른다. 그러면, 각각의 레코드를 해당 파시클에 대한 참조와 나머지 애트리뷰트로 표현하면, 파시클의 크기만큼 압축이 가능해진다. 그러기 위해 파시클 알고리즘은 먼저 테이블을 여러 번 스캔하여 파시클이 될 수 있는 레코드들의 후보 집합들을 찾아낸다. 이 후보 집합들의 집합 중에서 압축률을 최대로 하는 부분 집합을 찾아내는 것은 NP-Complete 문제로 알려져 있다. 그래서 파시클 알고리즘은 그리디 알고리즘을 사용하여 가장 압축을 많이 하는 집합 순으로 차례로 선택한다.

ItCompress[5] 알고리즘은 파시클 알고리즘과 유사하다. ItCompress도 파시클처럼 레코드들을 대표하는 대표 튜플들에 대한 참조를 이용함으로써 압축한다. 파시클 알고리즘과의 차이는 압축방법에 있다. 파시클 알고리즘에서 파시클들과 허용오차 내에 있지 않은 레코드들은 아웃라이어(outlier)로 취급되어 따로 저장되었다. 그러나, ItCompress에서는 대표 튜플들과 비교해서 일부 컬럼들이 허용 오차를 위반하더라도 전체 레코드를 아웃라이어로 저장하지 않고 위반하는 컬럼들만을 저장함으로써 압축률을 높이려고 시도한다. 이 때, 어떤 것이 아웃라이어인지 알기 위해 비트맵(bitmap)을 함께 저장한다. ItCompress 알고리즘은 처음에 테이블로부터 임의로 k개의 대표 튜플들을 선택한다. 각각의 레코드들은 압축이 가장 많이 되는 대표 튜플에 할당된다. 그리고 나서 같은 대표 튜플에 할당된 레코드들을 분석하여 압축률을 더 높이는 새로운 대표 튜플을 찾아낸다. 이 과정을 여러 번 반복하여 압축률을 높인다.

파시클 알고리즘이나 ItCompress 알고리즘이 레코드들 간의 상관관계를 이용하는 데 반해, 스파탄(SPARTAN) [6] 알고리즘은 컬럼간의 상관관계를 이용하여 압축을 시도한다. 즉, 전체 테이블에 대해 m개의 애트리뷰트를 예측하는 애트리뷰트들의 집합과 예측되는 애트리뷰트들의 집합으로 나누고, 예측하는 애트리뷰트들만을 저장함으로써 압축한다. 이 때, 예측되는 애트리뷰트들의 값들은 예측하는 애트리뷰트들의 값들로터 알아낼 수 있다. 스파탄 알고리즘은 베이지안 네트워크(Bayesian Network)[7]와 카트(CaRTs: classification and regression trees)[8,9]를 구축하여 예측하는 컬럼들을 찾아낸다. 게다가, 예측하는 애트리뷰트들의 집합을 파시클을 이용하여 더 압축한다. 따라서, 파시클의 성능을 향상시킴으로써 더 좋은 압축률을 보일 수 있다.

3. 파시클 알고리즘들과 개선된 다중-파시클 알고리즘

이번 장에서는 저장 공간의 최소화 문제를 해결하기 위한 해결책을 제시한다. [4]에서 저장 공간의 최소화 문제는 잘 알려진 최소 커버 문제(minimum cover problem)[10]에 해당하기 때문에 NP-complete이라는 것을 보이고, 근사 알고리즘들을 제시한다. 근사 알고리즘은 후보가 되는 파시클을 찾는 알고리즘을 개발하고 그 후보들 중에 어떤 값을 이용해서 저장 공간을 최소화할 수 있는지를 보여준다. [4]에서 나타난 단일-k와 다중-k 알고리즘을 먼저 설명하고 그것을 기반으로 개선된 다중-파시클 알고리즘을 제시한다.

3.1 단일-k 알고리즘

아래는 주어진 k값에 대해 k-D 파시클을 찾아내는 알고리즘을 제시하고 이후에는 파시클의 차원이 k이상인 파시

클을 찾는 다중-k 알고리즘을 제시한다.

3.1.1 기본적인 접근법 : 격자에 기초한 개념화

주어진 릴레이션의 모든 튜플의 부분 집합으로 구성된 격자를 고려하면 일반적으로 간선으로 연결된 부분 집합의 경우에는 $S \supset T$ 라면 $|S|=|T|+1$을 만족한다. 이 때 파시클의 정의에 의해 T가 S의 부분 집합이고 S가 j-D 파시클이라면 T는 $i \geq j$ 인 i-D 파시클이여야만 한다. 이러한 성질은 $1 \leq k \leq n$인 임의의 k에 대해서 격자의 꼭대기가 k-D 파시클이거나 아래의 성질을 만족하는 연속된 두 개의 집합 S_t, S_{t+1}가 존재함을 보장한다.

(1) S_t는 i-D 파시클이다.

(2) S_{t+1}는 j-D 파시클인데 $j < k \leq i$를 만족한다.

이러한 성질을 만족하는 S_t를 팁셋이라 하자. 팁셋을 이렇게 정의하면 최대 집합이라는 것을 정의할 수 있는데 이것은 S가 k-D 파시클이고 격자에서 S의 모든 조상인 T에 대해서 T는 j-D 파시클이면서 j<k를 만족하는 경우이다.

우리의 목표는 최대 집합을 찾는 것이다.

3.1.2 좋은 초기 팁셋 고르기

개념적으로 팁셋을 찾기 위해서는 무작위로 하나씩의 튜플을 골라서 넣으면서 해당 집합이 더 이상 k-D 파시클이 되지 않을 때까지 만들어 나가면 된다.

주어진 릴레이션이 메인 메모리로 처리하기에 너무 클 경우, 하나의 레코드 추출은 한 번의 디스크 접근을 의미하므로 볼록 샘플링이라는 기법을 이용하여 디스크 페이지를 메모리로 읽어 온 후 그 페이지에 있는 모든 튜플의 값을 사용한다. 하지만 기존의 방법과는 달리 우리의 경우에는 어떤 튜플의 순서가 매우 중요하므로 표본의 크기를 증가시키는 것은 그다지 의미가 없다. 따라서 b개의 페이지를 메모리에 읽어 온 후에 k-D 파시클을 찾고 다시 튜플의 순서를 무작위로 변경한 후에 다른 k-D 파시클을 찾는 방법을 이용하여 여러 팁셋을 찾는다.

3.1.3 팁셋을 최대 집합으로 키우기

이러한 팁셋은 k개의 압축 애트리뷰트를 가지므로 이것을 k-D의 조건으로 하여 나머지 릴레이션에 질의를 주게 되면 여전히 k-D 파시클인 팁셋을 만들 수 있다.

k개의 압축 애트리뷰트가 이미 t에 의해 허용되는 최대 너비를 차지하게 되었다면 위의 방법을 통해서 최대 집합을 찾을 수 있다. 하지만 아직 허용하는 공간이 남아 있다면 튜플을 추가하는 방법으로 최대 집합을 찾아야 한다.

3.2 다중-k 알고리즘

저장 공간을 최소화하기 위한 문제에서는 여러 개의 차원 k를 사용할 수 있지만 우리가 제시한 알고리즘은 하나의 k만을 사용하는 것이었다. k의 값이 커지면 그만큼 더 압축할 수 있는 여지가 생기는 것이므로 파시클의 부분 집합 중에 최대 차원을 가지는 값을 이용하여 압축하는 것이 저장 비용을 더욱 줄일 수 있다.

단일-k 알고리즘에서는 <⊥,S_1,S_2,…,S_t>의 경로를 찾는 동안 해당하는 파시클의 차원은 점차적으로 줄어든다. 하지만 다중-k 알고리즘에서는 같은 경로 <⊥,S_1,S_2,…,S_t>를 찾으면서 차원의 값을 최대로 하는 집합을 찾는다.

3.3 단일-k 알고리즘에서의 그리디 선택

위의 알고리즘을 통해서 찾아낸 후보 파시클들 중에서 적절한 파시클들을 선택하여 저장 공간 최소화 문제를 해결해야 한다. 이 문제가 어려운 이유는 문제 자체가 NP-complete인 셋 커버문제로 전환되기 때문이다. 따라서 알려진 방식대로 가장 많은 튜플을 포함하는 파시클을 먼저 선택하는 그리디 선택법을 사용한다. 이 때 k를 F의 차원으로 하고 고르는 파시클의 가중치를 wt(F)=k*|F|로 하면 wt값이 큰 순서로 고르면 커버하는 튜플이 많은 파시클의 순서로 고르게 된다. 매번 파시클을 고를 때마다 가중치가 wt(F/A)=k*|F-A|로 변경이 된다. 여기서 A는 이미 선택된 파시클이고 F는 현재 고려하고 있는 파시클이다. 이렇게 해서 주어진 릴레이션을 모두 커버할 때까지 파시클을 찾게 되면 압축이 끝난다.

3.4 다중-k에서의 그리디 선택 알고리즘

다중-k 알고리즘에서는 가중치를 재할당하는 방법이 조금 더 복잡하다. 예를 들어 $|F_1|=50$, $k_1=8$, $|F_2|=100$, $k_2=6$인 경우를 생각해 보자. F_2에 의한 공간 절약이 600이므로 F_2를 처음에 선택하겠지만 이후에 F_1을 이용해서도 100만큼의 절약이 가능하다. 하지만 wt값을 단일-k의 경우와 같이 사용하면 F_2를 선택한 후의 F_1의 가중치가 0이 되어서 선택이 되지 않는다. 그러므로 다중-k에 대해서는 아래와 같은 wt를 적용하기를 제안한다.

$$wt(F/A) = k_F * |F - A| + \max\{(k_F - k_A), 0\} * |F \cap A|$$

위의 식에서 k_F는 현재 고려하고 있는 파시클의 압축 애트리뷰트 집합이고 k_A는 이미 고려된 파시클의 압축 애트리뷰트 집합이다.

3.5. 다중-파시클(multi-fasicle) 알고리즘

다중-파시클 알고리즘은 다중-k 알고리즘과 유사하다. 차이점은 그리디 선택 알고리즘에 있다. 다중-파시클 알고리즘의 핵심은 하나의 레코드를 압축하는 데 여러 개의 파시클을 사용할 수 있다는 점이다. 따라서, 어떤 파시클을 이용해 어떤 레코드를 압축하기 위해서는 앞에서 선택된 파시클들에 의해 압축되는 애트리뷰트를 제외하고 얼마나 많은 애트리뷰트를 압축할 수 있는지 알아야 한다. 즉, 다중-k 알고리즘에서는 압축 애트리뷰트의 개수만을 고려하지만 다중-파시클 알고리즘에서는 어떤 애트리뷰트인지도 고려해야 한다. 그러므로, wt 함수는 다음과 같이 바뀐다.

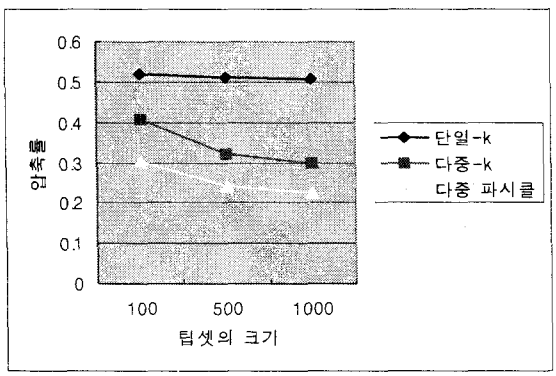
$$wt(F/A) = |k_F| * |F - A| + |k_F - k_A| * |F \cap A|$$

여기서 k_F와 k_A는 상수가 아니라 압축 애트리뷰트들의 집합이다. 즉, $k_F - k_A$는 앞에서 선택된 파시클들에 의해 압축되는 파시클을 제외하고 F에 의해 압축될 수 있는 애트리뷰트들의 집합을 의미한다. 만약 이 값이 공집합이면 F는 해당 레코드들을 압축할 수 없다는 것을 의미한다.

4. 실험 결과

이 장에서는 다양한 인자에 대한 다중-파시클 알고리즘의 성능을 평가한다. 실험은 2.4GHz Pentium 4 프로세서와 1GB의 메인 메모리를 가진 PC에서 수행되었다.
실험에 사용된 데이터는 [5]와 [6]에서도 사용된 이미지데이터 (http://kdd.ics.uci.edu/databases/CorelFeatures)이다. 이 데이터는 Corel. 이미지 모음으로부터 추출된 이미지의 특성들을 포함하고 있다. 우리가 사용하는 데이터는 68,040개의 이미지로부터 추출된 32개의 숫자로 된 애트리뷰트로 구성되어 있고 전체 크기는 20 MB이다.
사용된 알고리즘은 3장에서 설명된 3가지 알고리즘이다. 실험은 팁셋의 크기와 허용 오차 압축 애트리뷰트의 수를 변화시켜 다중-파시클 알고리즘의 압축률이 다른 두 알고리즘에 비해 우위에 있음을 보인다. 각 인자의 디폴트 값은 팁셋의 크기는 500, 허용 오차는 3%, 압축 애트리뷰트의 수는 15이다. 압축률은 원본 데이터에 비해 압축된 후의 데이터의 크기의 비율이다.

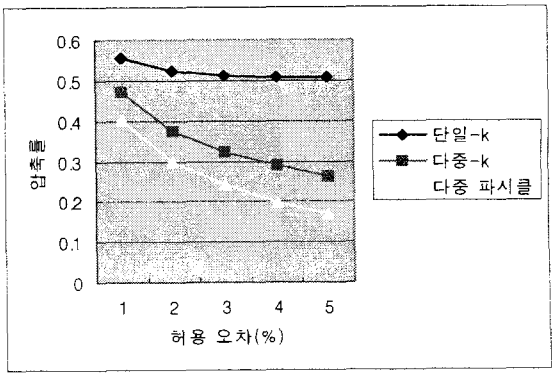
4.1 팁셋의 크기를 변화시킨 실험 결과



□림 4 팁셋의 크기를 변화시켰을 때 압축률의 변화

그림 4는 팁셋의 크기를 변화시켜 실험한 결과이다. 팁셋을 선택하는 과정에서 사용자가 정한 수만큼만 찾기 때문에 더 많은 팁셋을 찾을수록 더 좋은 팁셋이 선택될 가능성이 많아지므로 더 좋은 압축률을 보이고 있다. 결과는 다중-파시클 알고리즘이 가장 좋고 다중-k, 단일-k의 순서로 압축률이 좋다. 특히 다중-파시클 알고리즘을 사용했을 때 다중-k보다 거의 10%정도 압축률이 좋아졌다.

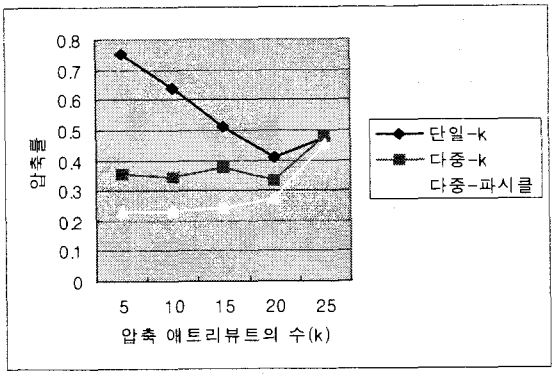
4.2 허용 오차를 변화시킨 실험 결과



□림 5 허용 오차를 변화시켰을 때 압축률의 변화

그림 5는 허용 오차를 1%~5%로 변화시켜 실험한 결과이다. 당연하게도 허용 오차가 커질수록 더 많은 레코드를 하나의 파시클로 압축할 수 있으므로 압축률은 좋아지고 있다. 결과는 항상 다중-파시클의 압축률이 가장 좋고 다중-k보다 10%정도 좋다.

4.3 압축 애트리뷰트의 수(k)를 변화시킨 실험 결과



□림 6 압축 애트리뷰트의 수를 변화시켰을 때 압축률의 변화

그림 6은 압축 애트리뷰트이 수를 변화시킨 실험 결과이다. 다중-파시클이 가장 좋은 압축률을 보인다. 다중-k나 다중-파시클은 압축 애트리뷰트의 수가 작을 수록 다양한 팁셋 중에서 선택할 수 있으므로 압축률이 좋다. 그러나 단일-k는 팁셋의 압축 애트리뷰트의 수가 고정되므로 수가 너무 적거나 크면 압축률이 좋지 않게 된다. 결과는 압축 애트리뷰트의 수가 25일 때 세 알고리즘이 비슷한 압축률을 보이다가 수가 줄어들면서 다중-파시클의 압축률이 상대적으로 훨씬 높아진다.

위의 세 실험 결과 항상 다중-파시클의 압축률이 좋고 거의 다중-k 보다 10%정도 압축률이 좋다. 위의 그래프는 원본 데이터에 대한 절대적인 압축률을 비교한 것이고, 상대

적으로는 다중-k의 압축률에 비해 평균적으로 20%정도 압축률이 좋아지고 경우에 따라 40% 이상 좋아지는 경우도 있다.

5. 결 론

최근 들어 데이터의 크기가 점점 커지고 있고 이를 관리하고 분석하는 것은 점점 어려운 일이 되고 있다. 의미론적 압축 기법은 이런 문제에 대한 한 가지 해결책으로서 주목을 받고 있고 그 성능을 개선하는 것은 중요한 문제가 되고 있다. 이런 이유로 본 논문은 기존의 파시클의 압축률을 개선하는 알고리즘을 제안하였다.

참고 문헌

[1] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. IEEE Transactions on Information Theory, 23:337-343, 1977.

[2] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. IEEE Transactions on Information Theory, 24:550-536, 1978.

[3] David A. Huffman. A method for the construction of minimum-redundancy codes. Proceedings of the IRE, 40(9):1098-1101, 1952.

[4] H.V.Jagadish, J.Madar, and R.Ng. Semantic compression and pattern extraction with fascicles. In Proc. 1999 Int. Conf. Very Large Data Bases (VLDB'99), pages 186-197, Edinburgh, UK, Sept. 1999.

[5] H. V. Jagadish, R. T. Ng, B. C. Ooi, and A. K. H. Tung. ItCompress: An iterative semantic compression algorithm. In 20th International Conference on Data Engineering, page 646, 2004.

[6] S. Babu, M. N. Garofalakis, and R. Rastogi. Spartan: A model-based semantic compression system for massive data tables. In Proceedings ofthe 2001 ACM SIGMOD International Conference on Management of Data, Santa Barbara, California, USA, May 2001.

[7] J. Pearl. Causality-Models, Reasoning and Inference. Cambridge University Press, 2000.

[8] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and Regression Trees. Chapman & Hall, 1984.

[9] R. Rastogi and K. Shim. PUBLIC: A Decision Tree Classifier that Integrates Building and Pruning. In Proc. of the 24th Intl. Conf. on Very Large Data Bases, 1998.

[10] R. Karp. Reducibility among combinatorial problems. Complexity of computer Computations, Plenum Press, pp 85-103, 1972.