ICCIT
INTERNATIONAL CONFERENCE
CREATIVE COMMUNICATION AND INNOVATIVE TECHNOLOGY

PERGURUAN TINGGI RAHARJA
ACCREDITED

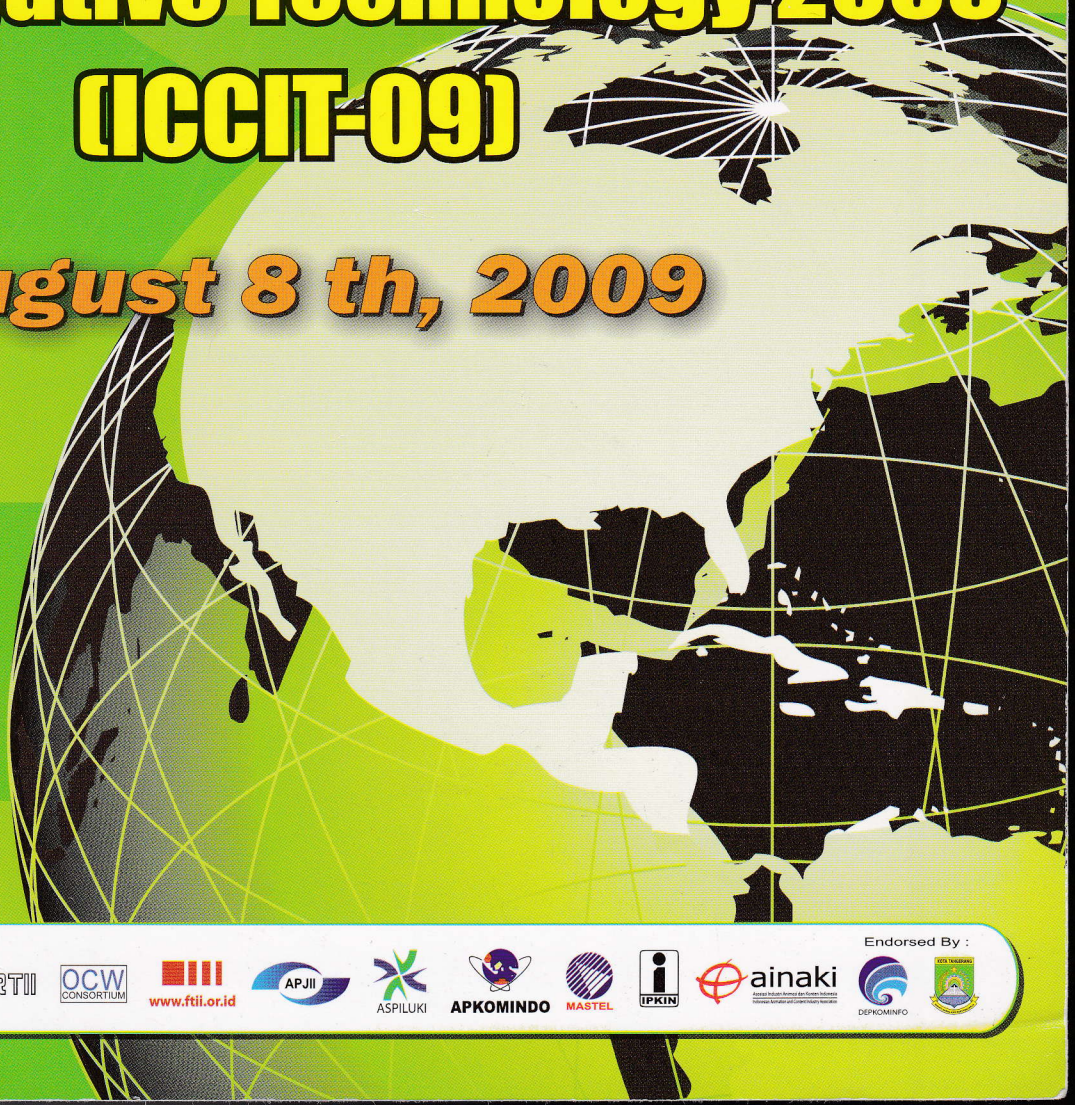
ISSN: 1978 - 8282

APTIKOM
ASOSIASI PERGURUAN TINGGI INFORMATIKA DAN KOMPUTER

PROCEEDINGS

International Conference on Creative Communication and Innovative Technology 2009 (ICCIT-09)

August 8 th, 2009



go green
With Raharja

Endorsed By :

REC IASA ISACA ID-SIRTII OCW CONSORTIUM www.ftii.or.id APJII ASPILUKI APKOMINDO MASTEL IPKIN ainaki DEPKOMINFO

ICCIT

Published by

CCIT Journal, Indonesia

The Association of Computer and Informatics Higher – Learning Instituition in Indonesia (APTIKOM) and STMIK Raharja Tangerang Section

CD – Conference Proceedong

CCIT Catalog Number:

ISSN: 1978 – 8282

Technically Co-Sponsored by

National Council Informatioan Techonology Of Indonesia

Raharja Enrichment Centre (REC)

The Institutional of Information Communication and Technology

Digital Syinztizer Laboratory of Computers system Processing

Design and typeset by:

Sugeng Widada & Lusyani Sunarya

Contents

Paper Participant

Gede Rasben Dantes
- Doctoral Student in Computer Science Department, University of Indonesia

Widodo Budiharto, DjokoPurwanto, Mauridhi Hery Purnomo
- Electrical Engineering Department Institue Technology Surabaya

Untung Rahardja, Valent
- STMIK RAHARJA Raharja Enrichment Centre (REC)
  Tangerang - Banten, Republic of Indonesia

Diyah Puspitaningrum, Henderi
- Information System, Faculty of Computer Science

Wiwik Anggraeni, Danang Febrian
- Information System Department, Institut Teknologi Sepuluh Nopember

Aan Kurniawan, Zainal A. Hasibuan
- Faculty of Computer Science, University of Indonesia

Untung Rahardja, Edi Dwinarko, Muhamad Yusup
- STMIK RAHARJA Raharja Enrichment Centre (REC) Tangerang - Banten, Republic of Indonesia
- GADJAH MADA UNIBERSITYFaculty of Mathematics and Natural SciencesYogyakarta,

Sarwosri, Djiwandou Agung Sudiyono Putro
- Department of Informatics, Faculty of Information Technology
- Institute of Technology Sepuluh Nopember

Chastine Fatichah, Nurina Indah Kemalasari
- Department, Faculty of Information Technology
- Institut Teknologi Sepuluh Nopember, Kampus ITS Surabaya

Untung Rahardja, Jazi Eko Istiyanto
- STMIK RAHARJA Raharja Enrichment Centre (REC)
  Tangerang - Banten, Republic of Indonesia
- GADJAH MADA UNIVERSITY Yogyakarta, Republic of Indonesia

Bilqis Amaliah, Chastine Fatichah, Diah Arianti
- Informatics Department – Faculty of Technology Information
- Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia

Tri Pujadi
- Information System Department – Faculty of Computer Study Universitas Bina Nusantara
  Jl. Kebon Jeruk Raya No. 27, Jakarta Barat 11530 Indonesia

Untung Rahardja, Retantyo Wardoyo, Shakinah Badar
- Faculty of Information System, Raharja University Tangerang, Indonesia
- Faculty of Mathematics and Natural Science, Gadjah Mada University Yogyakarta, Indonesia
- Faculty of Information System, Raharja UniversityTangerang, Indonesia

Paper Participant

Henderi, Maimunah, Asep Saefullah
- Information Technology Department – Faculty of Computer Study STMIK Raharja
  Jl. Jenderal Sudirman No. 40, Tangerang 15117 Indonesia

Yeni Nuaraeni
- Program Study Information Technology University Paramadina

Sfenrianto
- Doctoral Program Student in Computer Science University of Indonesia

Asep Saefullah, Sugeng Santoso
- STMIK RAHARJA Raharja Enrichment Centre (REC) Tangerang - Banten, Republic of Indonesia

Henderi, Maimunah, Aris Martono
- STMIK RAHARJA Raharja Enrichment Centre (REC) Tangerang - Banten, Republic of Indonesia

M. Tajuddin, Zainal Hasibuan, Abdul Manan, Nenet Natasudian, Jaya
- STMIK Bumigora Mataram West Nusa Tenggara
- Indonesia University
- PDE Office of Mataram City
- ABA Bumigora Mataram

Ermatita, Edi Dwinarko, Retantyo Wardoyo
- Information systems of Computer science Faculty Sriwijaya University
  (Student of Doctoral Program Gadjah Mada university)
- Computer Science of Mathematics and Natural Sciences Faculty Gadjah Mada University

Junaidi, Sugeng Santoso, Euis Sitinur Aisyah
- STMIK RAHARJA Raharja Enrichment Centre (REC) Tangerang - Banten, Republic of Indonesia

Ermatita, Huda Ubaya, Dwiroso Indah
- Information systems of Computer science Faculty Sriwijaya University
  (Student of Doctoral Program Gadjah Mada university)
- Computer Science Faculty of Sriwijaya University Palembang-Indonesia.

Mauritsius Tuga
- Jurusan Teknik Informatika Universitas Katolik Widya Mandira Kupang

Padeli, Sugeng Santoso
- STMIK RAHARJA Raharja Enrichment Centre (REC) Tangerang - Banten, Republic of Indonesia

M. Givi Efgivia, Safarudin, Al-Bahra L.B.
- Staf Pengajar STMIK Muhammadiyah Jakarta
- Staf Pengajar Fisika, FMIPA, UNHAS, Makassar
- Staf Pengajar STMIK Raharja, Tangerang

Primantara, Armanda C.C, Rahmat Budiarto, Tri Kuntoro P.
- School of Computer Sciences, Univeristi Sains Malaysia, Penang, Malaysia
- School of Computer Science, Gajah Mada University, Yogyakarta, Indonesia

Paper Participant

Hany Ferdinando, Handy Wicaksono, Darmawan Wangsadiharja
- Dept. of Electrical Engineering, Petra Christian University, Surabaya - Indonesia

Untung Rahardja, Hidayati
- STMIK RAHARJA Raharja Enrichment Centre (REC) Tangerang - Banten, Republic of Indonesia

Dina Fitria Murad, Mohammad Irsan
- STMIK RAHARJA Raharja Enrichment Centre (REC)
  Tangerang - Banten, Republic of Indonesia

Asep Saefullaf, Augury El Rayeb
- STMIK RAHARJA Raharja Enrichment Centre (REC)
  Tangerang - Banten, Republic of Indonesia

Richardus Eko Indrajit
- ABFI Institute, Perbanas

Azzemi Arifin, Young Chul Lee, Mohd. Fadzil Amiruddin, Suhandi Bujang, Salizul Jaafar, Noor Aisyah, Mohd. Akib
- AKIB#6#System Technology Program, Telekom Research & Development Sdn. Bhd., TMR&D Innovation Centre, Lingkaran Teknokrat Timur, 63000 Cyberjaya, Selangor Darul Ehsan, MALAYSIA
  Division of Marine Electronics and Communication Engineering, Mokpo National Maritime University (MMU) 571 Chukkyo-dong, Mokpo, Jeonnam, KOREA 530-729

Sutrisno
- Departement of Mechanical and Industrial Engineering, Gadjah Mada University,
  Jl. Grafika 2 Yogyakarta. 52281
- Faculty of Mathematics and Natural Sciences, Gadjah Mada University,
- Departement of Geodetical Engineering, Gadjah Mada University,

Saifuddin Azwar, Untung Raharja, Siti Julaeha
- Faculty Psychology, Gadjah Mada University Yogyakarta, Indonesia
- Faculty of Information System Raharja University Tangerang, Indonesia

Henderi, Sugeng Widada, Euis Siti Nuraisyah
- Technology Department – Faculty of Computer Study STMIK Raharja
  Jl. Jenderal Sudirman No. 40, Tangerang 15117 Indonesia

Panel of Reviewers

Abdul Hanan Abdullah, Prof.
Universiti Teknologi Malaysia

Arif Djunaidy, Prof.
Sepuluh November Institute of Technology, Indonesia

Djoko Soetarno, Ph.D
STMIK Raharja, Indonesia

Edi Winarko, Ph.D
Gajah Mada University, Indonesia

E.S. Margianti, Prof.
Gunadarma University, Indonesia

Iping Supriyana, Dr.
Bandung University of Technology, Indonesia

Jazi Eko Istiyanto, Ph.D
Gajah Mada University, Indonesia

K.C. Chan, Prof.
University of Glasgow, United Kingdom

Marsudi W. Kisworo, Prof.
Swiss-German University, Indonesia

Rahmat Budiharto, Prof.
Universiti Sains Malaysia

Stephane Bressan, Prof.
National University of Singapore

Suryo Guritno, Prof
Gajah Mada University, Indonesia

Susanto Rahardja, Prof.
Nanyang Technologycal University, Singapore

T. Basaruddin, Prof.
University of Indonesia,

Thomas Hardjono, Prof.
MIT, USA

Untung Rahardja, M.T.I.
STMIK Raharja, Indonesia

Wisnu Prasetya, Prof.
Utrecht University, Netherland

Y. Sutomo, Prof.
STIKUBANK University, Indonesia

Author Index

Author Index

Saturday, August 8, 2009
13:55 - 14:15 Room L-210

MINING QUERIES FASTER USING MINIMUM DESCRIPTION LENGTH PRINCIPLE

Diyah Puspitaningrum, Henderi

Department of Computing Science Universiteit Utrecht
diyah@cs.uu.nl

Information Technology Department – Faculty of Computer Study STMIK Raharja
Jl. Jenderal Sudirman No. 40, Tangerang 15117 Indonesia
email: henderi@pribadiraharja.com

Abstract

Ever since the seminal paper by Imielinski and Mannila [8], inductive databases have been a constant theme in the data mining literature. Operationally, an inductive database is a database in which models and patterns are first class citizens.

Having models and patterns in the database raises many interesting problems. One, which has received little attention so far, is the following: do the models and patterns that are stored help in computing new models and patterns? For example, if we have induced a classi_er C from the database and we compute a query Q. Does knowing C speed up the induction of a new classi_er on the result of Q? In this paper we answer this problem positively for one speci_c class of models, the code tables induced by our Krimp algorithm. The Krimp algorithm was built using minimum description length (MDL) principle. In Krimp algorithm, if we have the code tables for all tables in the database, then we can approximate the code table induced by Krimp on the result of a query, using only these global code tables as candidates; that is, we do not have to mine for frequent item sets one the query result. Since Krimp is linear in the number of candidates and this reduces the set of frequent item sets by many orders of magnitude, this means that we can speed up the induction of code tables on query results by many orders of magnitude.

Keywords: Inductive Database, Frequent Item Sets, MDL

1. Introduction

Ever since the start of research in data mining, it has been clear that data mining, and more general the KDD process, should be merged into DBMSs. Since the seminal paper by Imielinski and Mannila [8], the so-called inductive databases have been a constant theme in data mining research.

Perhaps surprisingly, there is no formal de_nition of what an inductive database actually is. In fact, de Raedt in [12] states that it might be too early for such a de_nition. There is, however, concensus on some aspects of inductive databases. An important one is that models and patterns should be _rst class citizens in such a database. That is, e.g., one should be able to query for patterns.

Having models and patterns in the database raises many interesting new problems. One, which has received little attention so far, is the following: do the models and patterns that are stored help in computing new models and patterns? For example, if we have induced a classi_er C from the database and we compute a query Q. Does knowing C speed up the induction of a new classi_er on the result Q?

In fact, this general question is not only interesting in the context of inductive databases, it is of prime importance in everyday data mining practice.

In the data mining literature, the usual assumption is that we are given some database that has to be mined.

formalise it we use MDL, which is briey discussed.

2.1 Preliminaries and Assumptions We assume that our data resides in relational databases. In fact, note that the union of two relational databases is, again, a relational database. Hence, we assume, without loss of generality, that our data resides in one relational database DB. So, the mining database is constructed from DB using queries. Given the compositionality of relational query languages, we may assume, again with out loss of generality, that the analysis database is constructed using one query Q. That is, the analysis database is Q(DB), for some relational algebra expression Q. Since DB is _xed, we will often simply write Q for Q(DB); that is we will use Q to denote both the query and its result.

2.2 The Problem Informally In the introduction we stated that knowing a model on DB should help in inducing a model on Q. To make this more precise, let A be our data mining algorithm. A can be any algorithm, it may, e.g., compute a decision tree, all frequent item sets or a neural network. LetMDB denote the model induced by A from DB, i.e, MDB = A(DB). Similarly, let MQ = A(Q). We want to transform A into an algorithm A_ that takes at least two inputs, i.e, both Q and MDB, such that:

1. A_ gives a reasonable approximation of A when ap plied to Q, i.e., A_(Q;MDB) tMQ

2. A_(Q;MDB) is simpler to compute than MQ. The second criterion is easy to formalise: the runtime of A_ should be shorter than that of A. The _rst one is harder. What do we mean that one model is an approximation of another? Moreover, what does it mean that it is a reasonable approximation? There are many ways to formalise this. For example, for predictive models, one could use the di_erence between predictions as a way to measure how well one model approximates. While for clustering, one could use the number of pairs of points that end up in the same cluster.

We use the minimum description length (MDL) principle [7] to formalise the notion of approximation. MDL is quickly becoming a popular formalism in data mining research, see, e.g., [5] for an overview of other applications of MDL.

2.3 Minimum Description Length MDL like its close cousin MML (minimum message length) [17], is a practical version of Kolmogorov Complexity [11]. All three embrace the slogan Induction by Compression. For MDL, this principle can be roughly described as follows.

Given a set of models1 H, the best model H 2 H is the one that minimizes L(H) + L(DjH) in which L(H) is the length, in bits, of the description of H, and _ L(DjH) is the length, in bits, of the description of the data when encoded with H. One can paraphrase this by: the smaller L(H) + L(DjH), the better H models D.

What we are interested in is comparing two al-gorithms on the same data set, viz., on Q(DB). Slightly abusing notation, we will write L(A(Q)) for L(A(Q)) + L(Q(DB)jA(Q)), similarly, we will write L(A_(Q;MDB)). Then, we are interested in comparing 1MDL-theorists tend to talk about hypothesis in this context, hence the H; see [7] for the details. L(A_(Q;MDB)) to L(A(Q)). The closer the former is to the latter, the better the approximation is. Just taking the di_erence of the two, however, can be quite misleading. Take, e.g., two databases db1 and db2 sampled from the same underlying distribution, such that db1 is far bigger than db2. Moreover, _x a model H. Then necessarily L(db1jH) is bigger than L(db2jH).

In other words, big absolute numbers do not necessar-ily mean very much. We have to normalise the di_er-ence to get a feeling for how good the ap proximation is. Therefore we de_ne the asymmetric dissimilarity mea-sure (ADM) as follows. Definition 2.1. Let H1 and H2 be two models for a dataset D. The asymmetric dissimilarity measure ADM(H1;H2) is de_ned by:
ADM(H1;H2) = jL(H1) _ L(H2)j L(H2) Note that this dissimilarity measure is related to the Normalised Com pression Distance. The reason why we use this asymmetric version is that we have a \gold standard". We want to know how far our approximate result A_(Q;MDB) deviates from the optimal result A(Q).

2.4 The Problem Before we can formalise our prob-lem using the notation introduced above, we have one more question to answer: what is a reasonable approx-imation? For a large part the answer to this questionis, of course, dependent on the application in mind. An ADM in the order of 10% might be perfectly alright in one application, while it is unacceptable in another.
Hence, rather than giving an absolute number, we make it into a parameter _. Problem:
For a given data mining algorithm A, devise an algo-rithm A_, such that for all relational algebra expressions Q on a database DB:

1. ADM(A_(Q;MDB);A(Q)) __
2. Computing A_(Q;MDB) is faster than computing A(Q)

2.5 A Concrete Instance: Krimp The ultimate solution to the problem as stated in above would be an algorithm that transforms any data mining algorithm A in an algorithm A_ with the requested properties. This is a rather ambitious, ill-de_ned (what is the class of all data mining algo-rithms?), and, probably, not attain able goal. Hence, in this paper we take a more modest approach: we trans-form one algorithm only, our Krimp algorithm.
The reason for using Krimp as our problem instance is threefold. Firstly, from earlier research we know that Krimp characterises the underlying data distribution rather well; see, e.g., [14, 16]. Secondly, from earlier research on Krimp in a multi-relational setting, we al ready know that Krimp is easily transformed for joins [10]. Finally, Krimp is MDL based. So, notions such as L(A(Q)) are already de_ned for Krimp.

3. Introducing Krimp For the convenience of the reader we provide a brief introduction to Krimp in this section, it was originally introduced in [13] (although not by that name) and the reader is referred to that paper for more details.
Since Krimp is selects a small set of representative item sets from the set of all frequent item sets, we _rst recall the basic notions of frequent item set mining [1].

3.1 Preliminaries Let I = fI1; : : : ; Ing be a set of binary (0/1 valued) attributes. That is, the domain Di of item Ii is f0; 1g. A transaction (or tuple) over I is an element of Qi2f1;:::;ng Di. A database DB over I is a bag of tuples over I. This bag is indexed in the sense that we can talk about the i-th transaction. An item set J is, as usual, a subset of I, i.e., J _ I. The item set J occurs in a transaction t 2 DB if 8I 2 J : _I(t) = 1. The support of item set J in database DB is the number of transactions in DB in which J occurs. That is, suppDB(J) = jft 2 DBj J occurs in tgj. An item set is called frequent if its support is larger than some user-de_ned threshold called the minimal support, min-sup. Given the A Priori property, 8I; J 2 P(I) : I _ J suppDB(J) _ suppDB(I) frequent item sets can be mined e_ciently level wise, see [1] for more details. Note that while we restrict ourself to binary databases in the description of our problem and algo-rithms, there is a trivial generalisation to categorical databases. In the experiments, we use such categorical databases.

3.2 Krimp The key idea of the Krimp algorithm is the code table. A code table is a two-column table that has item sets on the left-hand side and a code for each item set on its right-hand side. The item sets in the code table

are ordered descending on 1) item set length and 2) support size and 3) lexicographically. The actual codes on the right-hand side are of no importance: their lengths are. To explain how these lengths are computed the coding algorithm needs to be intro duced.

A transaction t is encoded by Krimp by searching for the _rst item set c in the code table for which c _ t. The code for c becomes part of the encoding of t. If t nc 6= ;, the algorithm continues to encode t n c. Since it is insisted that each code table contains at least all singleton item sets, this algorithm gives a unique encoding to each (possible) transaction over I.

The set of item sets used to encode a transaction is called its cover. Note that the coding algorithm implies that a cover consists of non-overlapping item sets.

The length of the code of an item in a code table CT depends on the database we want to compress; the more often a code is used, the shorter it should be. To compute this code length, we encode each transaction in the database DB. The frequency of an item set c 2 CT, denoted by freq(c) is the number of transactions t 2 DB which have c in their cover That is, freq(c) = jft 2 DBjc 2 cover(t)gj The relative fre quency of c 2 CT is the probability that c is used to encode an arbitrary t 2 DB, i.e.

P(c) = freq(c) Pd2CT freq(d)

For optimal compression of DB, the higher P(c), the shorter its code should be. Given that we also need a code for unambiguous decoding, we use the well-known optimal Shannon code [4]:

lCT (c) = _log(P(cjDB)) = _log_ freq(c) Pd2CT freq(d)_ The length of the encoding of a transaction is simply the sum of the code lengths of the item sets in its cover. Therefore the encoded size of a transaction t 2 DB compressed using a speci_ed code table CT is calculated as follows:

lCT (t) = X c2cover(t;CT) lCT (c)

The size of the encoded database is the sum of the sizes of the encoded transactions, but can also be computed from the frequencies of each of the ele ments in the code table:

L(CTDB) = Xt2DB LCT (t)

= _Xc2CT freq(c) log_ freq(c) Pd2CT freq(d)_

To _nd the optimal code table using MDL, we need to take into account both the compressed database and Figure 1: Krimp in action as described above, as well as the size of the code table. For the size of the code table, we only count those item sets that have a non-zero frequency.

The size of the right-hand side column is obvious; it is simply the sum of all the di_erent code lengths. For the size of the left-hand side column, note that the simplest valid code table consists only of the singleton item sets. This is the standard encoding (st), of which we use the codes to compute the size of the item sets in the left-hand side column. Hence, the size of code table CT is given by: L(CT) = X c2CT:freq(c)6=0 lst(c) + lCT (c) In [13] we de_ned the optimal set of (frequent) item sets as that one whose associated code table minimises the total compressed size: L(CT) + LCT (DB)

Krimp starts with a valid code table (only the collection of singletons) and a sorted list of candidates (frequent item sets). These candidates are assumed to be sorted descending on 1) support size, 2) item set length and 3) lexicographically. Each candidate item set is considered by inserting it at the right position in CT and calculating the new total compressed size. A candidate is only kept in the code table i_ the resulting total size is smaller than it was before adding the candidate. If it is kept, all other elements of CT are reconsidered to see if they still positively contribute to compression. The whole process is illustrated in Figure 1. For more details see [13].

4 The Hypothesis for Krimp If we assume a _xed minimum support threshold for a database, Krimp has only one essential parameter:
the database. For, given the database and the (_xed) minimum support threshold, the candidate list is also speci_ed. Hence, we will simply write CTDB and Krimp(DB), to denote the code table induced by Krimp from DB. Similarly CTQ and Krimp(Q) denote
the code table induced by Krimp from the result of applying query Q to DB.
Given that Krimp results in a code table, there is only one sensible way in which Krimp(DB) can be re-used to compute Krimp(Q): provide Krimp only with the item sets in CTDB as candidates. While we change nothing to the code, we'll use the notation Krimp_to indicate that Krimp got only code table elements as candidates. So, e.g., Krimp_(Q) is the code table that Krimp induces from Q(DB) using the item sets in CTDB only.
Given our general problem statement, we have now have to prove that Krimp_ satis_es our two require-ments for a transformed algorithm. That is, _rstly, we have to show that Krimp_(Q) is a good approximation of Krimp(Q). That is, we have to show that ADM(Krimp_(Q); Krimp(Q)) = jL(Krimp_(Q)) _ L(Krimp(Q))j L(Krimp(Q))j _ _ for some (small) epsilon. Secondly, we have to show that it is faster to compute Krimp_(Q) than it is to compute Krimp(Q). Given that Krimp is a heuristic algorithm, a formal proof of these two requirements is not possible. Rather, we'll report on extensive tests of these two requirements.
5 The Experiments In this section we describe our experi-

mental set-up.
First we briey describe the data sets we used. Next we discuss the queries used for testing. Finally we describe how the tests were performed.
5.1 The Data Sets . To test our hypothesis that Krimp_ is a good and fast approximation of Krimp, we have performed extensive test on 8 well-known UCI [3]
data sets, listed in table 1, together with their respective numbers of tuples and attributes. These data sets were chosen because they are well suited for Krimp. Some of the other data sets in the UCI repository are simply too small for Krimp to perform well. MDL needs a reasonable amount of data to be able to function.
Some other data sets are very dense. While Krimp performs well on these data sets, choosing them would have turned our extensive testing prohibitively time-consuming.
Note that all the chosen data sets are single table Dataset #rows #attributes Heart 303 52
Iris 150 19 Led7 3200 24 Pageblocks 5473 46 Pima 786 38
Tictactoe 958 29 Wine 178 68

Table 1: UCI data sets used in the experiments.
data sets. This means, of course, that queries involving joins can not be tested in the experiments. The reason for this is simple: we have already tested the quality of Krimp_ in earlier work [10]. The algorithm introduced in that paper, called R-Krimp, is essentially Krimp_;
we'll return to this topic in the discussion section.
5.2 The Queries To test our hypothesis, we need to consider randomly generated queries. On _rst sight this appears a daunting task. Firstly, because the set of all possible queries is very large. How do we determine a representative set of queries? Secondly, many of the generated queries will have no or very few results. If the query has no results, the hypothesis is vacuously true.
If the result is very small, MDL (and Krimp) doesn't perform very well.
Generating a representative set of queries with a non-trivial result set seems an almost impossible task.
Fortunately, relational query languages have a useful property: they are compositional. That is, one can combine queries to form more complex queries. In fact, all queries use small, simple, queries as building blocks.
For the relational algebra, the way to de_ne and combine queries is through well-known operators: pro-jection (_), selection (_), join (on), union ([), intersec-tion (\), and setminus (n). As an aside, note that in principle the Cartesian product (_) should be in the list of operators rather than the join. Cartesian prod-ucts are, however, rare in practical queries since their results are often humonguous and their interpretation is at best di_cult. The join, in contrast, su_ers less from the _rst disadvantage and not from the second. Hence, our ommission of the Cartesian products and addition of the join.
So, rather than attempting to generate queries of arbitrary complexity, we generate simple queries only.
That is, queries involving only one of the operators _, _, [, \, and n. How the insight o_ered by these exper-iments coupled with the compositionality of relational algebra queries o_ers insight in our hypothesis for more general queries is discussed in the discussion section.
5.3 The Experiments The experiments preformed for each of the operators on each of the data sets were generated as follows.
Projection: The projection queries were generated by randomly choosing a set X of n attributes, for n 2 f3; 5g. The generated query is then _X. For this case, the code table elements generated on the complete data set were also projected on X.
The rationale for using a small sets of attributes rather than larger ones is that these projections are the most disruptive. That is, the larger the set of attributes projected on, the more the structure of the table remains intact. Given that Krimp induces this structure, projections on small sets of attributes are the best test of our hypothesis.
Selections: The random selection queries were again generated by randomly choosing a set X of n attributes, with n 2 f1; 2; 3; 4g. Next for each random attribute Ai a random value vi in its domain Di was chosen. Finally, for each Ai in X a random _i 2 f=; 6=g was chosen The generated query is thus _(VAi2X Ai_ivi).
The rationale for choosing small sets of attributes in this case is that the bigger the number of attribute sets selected on, the smaller the result of the query becomes. Too small result sets will make Krimp perform badly.
Union: For the union queries, we randomly split the dataset D in two parts D1 and D2, such that D = D1 [D2; note that in all experiments D1 and D2 have roughly the same size. The random query generated is, of course, D1 [D2.
Krimp yields a code table on each of them, say CT1 and CT2. To test the hypothesis, we give Krimp_the union of the item sets in CT1 and CT2.

In practice, tables that are combined using a union may not be disjoint. To test what happens with various level of overlap between D1 and D2, we tested at overlap levels from f0%; 33:3%; 50%g.
intersection: For the intersection queries, we again randomly split the data set D into two overlapping parts D1 and D2. Again, such that D = D1 [D2 and again in all experiments D1 and D2 have roughly the same size. The random query gener-ated is, of course, D1 \ D2.
Again Krimp yields a code table on each of them, say CT1 and CT2. To test the hypothesis, we give Krimp_ the union of the item sets in CT1 and CT2.
The union of the two is given as either of one might have good codes for the intersection. The small raise in the number of candidates is o_set by this potential gain.

the overlap levels tested were from f33:3%;

Selection queries can, of course, be seen as a of setminus queries. They are special, though, in the that they remove a well described part of the data-

well structured setminus operations, we sim- random subsets of the data set. The sizes of chosen from f33:3%; 50%; 66:6%g. these experiments is performed ten times on each

In this section we give an overview of the of the experiments described in the previous sec- relational algebra operator is briey discussed in

The projection results are given in Table 2. scores listed are the average ADM score over experiments performed on that data set _ deviation of those 10 ADM scores. Similarly, scores. Note that Size stands for the reduction of candidates.

score of 0.2, means that Krimp_ got only the number of candidates that Krimp got for the

that most of the ADM scores are in the percent, whereas the Size scores are gener- The notable exceptions are the scores for and, in one case, for Led7. Note, however, these averages, the standard deviation is One would expect, this is caused by a That is, if one looks at the individual scores one or two are very high. Random experi- their disadvantages.

should note, however, that these_gures selected sets of attributes. Such are about as disruptive of the data possible. In other words, it is impressive still manages to do so well.

the bigger the result set is, the smaller One can see that, e.g., by comparing the data set for projections on 3 and 5 Clearly, these di_erences are in and the trend doesn't always hold. we will also see in the other experi-

The results are given in Ta-ble 3. the averages _ the standard devia- The resulting ADM scores are now of just a few percent. This is all the considers the Size scores. were reached while Krimp_ got of the number of candidates than often it got less than 1% of the num-

ber of candidates.

The fact that Krimp_ performs so well for selections means that while Krimp models the global underlying data distribution, it still manages capture the \local" structure very well. That is, if there is a pattern that is important for a part of the database, it will be present in the code table.

The fact that the results improve with the number of attributes in the selection, though mostly not sig-ni_cantly, is slightly puzzling. If one looks at all the experiments in detail, the general picture is that big-ger query results give better results. In this table, this global picture seems reversed. We do not have a good explanation for this observation.

6.3 Union The projection results are given in Ta-ble 4. The general picture is very much as with the previous experiments. The ADM score is a few percent, while the reduction in the number of candidates is often impressive.

The notable exception is the Iris database. The explanation is that this data set has some very local structure that (because of minsup settings) doesn't get picked up in the two components; it only becomes apparent in the union. Note that this problem is exaggerated by the fact that we split the data sets at random. The same explanation very much holds for the_rst Led7 experiment.

We already alluded a few times to the general trend that the bigger the query results, the better the results.

This trend seems very apparent in this table. For, the higher the overlap between the two data sets, the bigger the two sets are, since their union is the full data set.

However, one should note that this is a bit misleading, for the bigger the overlap the more the two code tables\know" about the \other" data distribution.

6.4 Intersection The projection results are given in Table 5. Like with for the union, the reduction of the number of candidates is again huge in general. The ADM scores are less good than for the union, however, still mostly below 0.1. This time the Heart and the Led7 databases that are the outliers. Heart shows the biggest reduction in the number of candidates, but at the detriment of the ADM score. The explanation for these relative bad scores lies again in local structures, that have enough support in one or both of the components, but not in the intersec-tion. That is, Krimp doesn't see the good candidates for the tuples that adhere to such local structures. This is witnessed by the fact that some tuples are compressed better by the original code tables than by the Krimp generated code table for the intersection. Again, this problem is, in part, caused by the fact that we split our data sets at random.

The ADM scores for the other data sets are more in line with the numbers we have seen before. For these, the ADm score is below 0.2 or (much) lower.

6.5 Setminus The projection results are given in Table 6.

Both the ADM scores and the Size scores are very good for all of these experiments. This does make sense, each of these experiments is computed on a random subset of the data. If Krimp is any good, the code tables generated from the complete data set should compress a random subset well.
It may seem counter intuitive that the ADM score grows when the size of the random subset grows. In fact, it is not. The bigger the random subset, the closer its underlying distribution gets to the \true" underlying distribution. That is, to the distribution that underlies the complete data set. Since Krimp has seen the whole data set, it will pick up this distribution better than Krimp_.

7 Discussion
First we discuss briey the results of the experiments.
Next we discuss the join. Finally we discuss what these experiments mean for more general queries.

7.1 Interpreting the Results The Size scores re-ported in the previous section are easy to interpret.
They simply indicate how much smaller the candidateset becomes. As explained before, the runtime complex-ity of Krimp is linear in the number of candidates. So, since the Size score is never below 0.4 and, often, con-siderably lower, we have established our _rst goal for Krimp_. It is faster, and often far faster, than Krimp.
In fact, one should also note that for Krimp_, we do not have to run a frequent item set miner. In other words, in practice, using Krimp_ is even faster than suggested by the Size scores.
But, how about the other goal: how good is the approxima-tion? That is, how should one interpret ADM scores? Ex-cept for some outliers, ADM scores are below 0.2. That is, a full-edged Krimp run compresses the data set 20% better than Krimp_. Is that good?
In a previous paper [15], we took two random sam-ples from data sets, say D1 and D2. Code tables CT1and CT2 were induced from D1 and D2 respectively.
Next we tested how well CTi compressed Dj . For the four data sets also used in this paper, Iris, Led7, Pima and, PageBlocks, the \other" code table compressed 16% to 18% worse than the \own" code table; the _g-ures for other data sets are in the same ball-park. In other words, an ADM score on these data sets below 0.2 is on the level of \natural variations" of the data distri-bution. Hence, given that the average ADM scores are often much lower we conclude that the approximation by Krimp_ is good.
In other words, the experiments verify our hypoth-esis: Krimp_ gives a fast and good approximation of Krimp. At least for simple queries.

7.2 The Join In the experiments, we did not test the join operator. We did, however, already test the join in a previ-ous paper [10]. The R-Krimp algorithm introduced in that paper is Krimp_ for joins only.
Given two tables, T1 and T2, the code table is induced on both, resulting in CT1 and CT2. To compute the code table on T1 on T2, R-Krimp only uses the item sets in CT1 and CT2. Rather than using, the union of these two sets, for the join one uses pairs (p1; p2), with p1 2 CT1 and p2 2 CT2.
While the ADM scores are not reported in that paper, they can be estimated from the numbers reported there. For various joins on, e.g., the well known _nancial data set, the ADM can be estimated as to be between 0.01 and 0.05. The Size ranges from 0.3 to 0.001; see
[10] for details.
In other words, Krimp_ also achieves its goals for the join operator.

7.3 Complex Queries For simple queries we know that Krimp_ delivers a fast and good approximation.
How about more complex queries?
As noted before, these complex queries are built from sim-pler ones using the relational algebra operators.
Hence, we can use error propagation to estimate the error of such complex queries.
The basic problem is, thus, how do the approxima-tion errors propagate through the operators? While we do have no de_nite theory, at worse, the errors will have to be summed. That is, the error of the join of two se-lections will be the sum of the errors of the join plus the errors of the selections.
Given that complex queries will only be posed on large database, on which krimp performs well. The initial errors will be small. Hence, we expect that the error on complex queries will still be reasonable; this is, however, subject to further research.
8 Related Work While there are, as far as the authors know, no other papers that study the same problem, the topic of this paper falls in the broad class of data mining with back-ground knowledge. For, the model on the database, MDB, is used as background knowledge in computing MQ. While a survey of this area is beyond the scope of this paper, we point out some papers that are related to one of the two aspects we are interested in, viz., speed-up and approx-imation.
A popular area of research in using background knowl-edge is that of constraints. Rather than trying to speed up the mining, the goal is often to produce mod-els that ad-here to the background knowledge. Examples are the use of constraints in frequent pattern mining, e.g. [2], and mono-tonicity constraints [6]. Note, how-ever, that for frequent patter mining the computation can be speeded up consid-erable if the the constraints can be pushed into the mining algorithm [2]. So, speed-up is certainly a concern in this area. However, as far
as we know approximation plays no role. The goal is still to _nd all patterns that satisfy the constraints.

[3] Frans Coenen. The LUCS-KDD discretised/normalised ARM and CARM data library: http://www.csc.liv.ac.uk/~frans/ KDD/Software/LUCS KDD DN/. 2003.

[4] T.M. Cover and J.A. Thomas. Elements of Informa-tion Theory, 2nd ed. John Wiley and Sons, 2006.

[5] C. Faloutsos and V. Megalooikonomou. On data mining, compression and kolmogorov complexity. In Data Mining and Knowledge Discovery, volume 15, pages 3{20. Springer Verlag, 2007.

[6] A. J. Feelders and Linda C. van der Gaag. Learning bayesian network parameters under order constraints.Int. J. Approx. Reasoning, 42(1-2):37{53, 2006.

[7] Peter D. Grnunwald. Minimum description length tutorial. In P.D. Grnunwald and I.J. Myung, editors, Advances in Minimum Description Length. MIT Press, 2005.

[8] Tomasz Imielinski and Heikki Mannila. A database perspective on knowledge discovery. Communications of the ACM, 39(11):58{64, 1996.

[9] Szymon Jaroszewicz and Dan A. Simovici. Interestingness of frequent itemsets using bayesian networks as background knowledge. In Proceedings KDD, pages 178{186, 2004.

[10] Arne Koopman and Arno Siebes. Discovering relational item sets e_ciently. In Proceedings SDM 2008, pages 585{592, 2008.

[11] M. Li and P. Vit_anyi. An introduction tp kolmogorov complexity and its applications. Springer-Verlag, 1993.

[12] Luc De Raedt. A perspective on inductive databases. SIGKDD Explorations, 4(2):69{77, 2000.

[13] A. Siebes, J. Vreeken, and M. van Leeuwen. Item sets that compress. In Proceedings of the SIAM Conference on Data Mining, pages 393{404, 2006.

[14] Matthijs van Leeuwen, Jilles Vreeken, and Arno Siebes. Compression picks item sets that matter. In Proceed-ings PKDD 2006, pages 585{592, 2006.

[15] J. Vreeken, M. van Leeuwen, and A. Siebes. Preserving privacy through data generation. In Proceedings of the IEEE International Conference on Data Mining, pages 685{690, 2007.

[16] Jilles Vreeken, Matthijs van Leeuwen, and Arno Siebes.

Preserving privacy through data generation. In Proceedings ICDM, 2007.

[17] C.S. Wallace. Statistical and inductive inference by minimum message length. Springer, 2005.

Dataset 3 attr 5 attr ADM Size ADM Size
Heart 0.06 _ 0.09 0.2 _ 0.13 0.03 _ 0.03 0.2 _ 0.13
Iris 0.24 _ 0.28 0.17 _ 0.12 0.21 _ 0.18 0.14 _ 0.12
Led7 0.05 _ 0.1 0.31 _ 0.23 0.38 _ 0.34 0.25 _ 0.19
PageBlocks 0.04 _ 0.06 0.23 _ 0.21 0.08 _ 0.06 0.2 _ 0.17
Pima 0.04 _ 0.05 0.14 _ 0.13 0.08 _ 0.07 0.23 _ 0.17
TicTacToe 0.12 _ 0.09 0.11 _ 0.17 0.09 _ 0.1 0.17 _ 0.11
Wine 0.16 _ 0.2 0.10 _ 0.09 0.1 _ 0.11 0.1 _ 0.09

Table 2: The results of the projection experiments. The ADM and Size scores are averages _ standard deviation

Dataset 1 attr 2 attr 3 attr 4 attr
ADM Size ADM Size ADM Size ADM Size
Heart 0.04 _ 0.03 0.04 _ 0.11 0.04 _ 0.03 0.02 _ 0.002 0.04 _ 0.03 0.003 _ 0.003 0.02 _ 0.02 0.001 _ 0.0004
Iris 0.04 _ 0.04 0.09 _ 0.01 0.05 _ 0.05 0.1 _ 0.02 0.04 _ 0.01 0.1 _ 0.01 0.01 _ 0.03 0.1 _ 0.01
Led7 0.04 _ 0.06 0.02 _ 0.001 0.04 _ 0.01 0.02 _ 0.001 0.03 _ 0.02 0.02 _ 0.001 0.03 _ 0.03 0.02 _ 0.01
PageBlocks 0.09 _ 0.07 0.007 _ 0.008 0.05 _ 0.04 0.002 _ 0.0002 0.03 _ 0.02 0.002 _ 0.0002 0.02 _ 0.02 0.002 _ 0.0002
Pima 0.1 _ 0.14 0.01 _ 0.003 0.03 _ 0.02 0.01 _ 0.003 0.03 _ 0.02 0.01 _ 0.002 0.03 _ 0.02 0.01 _ 0.001
TicTacToe 0.16 _ 0.09 0.01 _ 0.002 0.1 _ 0.028 0.01 _ 0.002 0.12 _ 0.04 0.02 _ 0.02 0.08 _ 0.03 0.01 _ 0.005
Wine 0.03 _ 0.03 0.02 _ 0.02 0.02 _ 0.02 0.02 _ 0.02 0.02 _ 0.01 0.01 _ 0.006 0.02 _ 0.01 0.01 _ 0.005

Table 3: The results of the selection experiments. The ADM and Size scores are averages _ standard deviation

Dataset 0% 33.3% 50%
ADM Size ADM Size ADM Size
Heart 0.07 _ 0.02 0.0001 _ 0.0001 0.04 _ 0.02 0.001 _ 0.00004 0.03 _ 0.05 0.001 _ 0.0002
Iris 0.36 _ 0.11 0.07 _ 0.01 0.37 _ 0.1 0.07 _ 0.007 0.34 _ 0.12 0.07 _ 0.006
Led7 0.38 _ 0.31 0.02 _ 0.005 0.05 _ 0.02 0.03 _ 0.002 0.03 _ 0.02 0.03 _ 0.002
PageBlocks 0.06 _ 0.01 0.002 _ 0.0001 0.04 _ 0.01 0.003 _ 0.0001 0.02 _ 0.01 0.003 _ 0.0001
Pima 0.04 _ 0.03 0.01 _ 0.0006 0.03 _ 0.02 0.02 _ 0.002 0.03 _ 0.02 0.02 _ 0.002
TicTacToe 0.07 _ 0.01 0.009 _ 0.0005 0.03 _ 0.02 0.01 _ 0.0003 0.01 _ 0.002 0.01 _ 0.0002
Wine 0.03 _ 0.01 0.006 _ 0.0003 0.03 _ 0.01 0.008 _ 0.0006 0.02 _ 0.01 0.008 _ 0.0003

Table 4: The results of the union experiments. The percentages denote the amount of overlap between the two data sets. The ADM and Size scores are averages _ standard deviation

Dataset 33.3% 50% 66.6%
ADM Size ADM Size ADM Size
Heart 0.39 _ 0.14 0.0002 _ 0.0001 0.36 _ 0.05 0.0002 _ 0.0001 0.42 _ 0.17 0.0001 _ 0.0001
Iris 0.09 _ 0.08 0.1 _ 0.02 0.08 _ 0.07 0.09 _ 0.02 0.03 _ 0.09 _ 0.01
Led7 0.5 _ 0.14 0.005 _ 0.002 0.42 _ 0.1 0.007 _ 0.001 0.12 0.01 _ 0.001
PageBlocks 0.13 _ 0.07 0.001 _ 0.0002 0.09 _ 0.06 0.0001 0.07 _ 0.05 0.002 _ 0.0001
Pima 0.09 _ 0.06 0.01 _ 0.002 0.09 _ 0.09 0.01 _ 0.06 0.01 _ 0.002
TicTacToe 0.2 _ 0.05 0.007 _ 0.002 0.22 _ 0.04 0.24 _ 0.04 0.004 _ 0.0007
Wine 0.1 _ 0.02 0.01 _ 0.005 0.12 _ 0.03 0.005 _ 0.04 0.002 _ 0.0006

Table 5: The results of the intersection experiments. The percentages denote the amount of overlap between the two data sets. The ADM and Size scores are averages _ standard deviation

Dataset 33.3% 50% 66.6%
ADM Size ADM Size ADM Size
heart 0.01 _ 0.01 0.001 _ 0.00007 0.01 _ 0.01 0.03 _ 0.02 0.002 _ 0.0004
iris 0.003 _ 0.006 0.11 _ 0.007 0.005 _ 0.008 0.12 _ 0.02 0.14 _ 0.01
led7 0.02 _ 0.02 0.02 _ 0.0002 0.02 _ 0.02 0.02 _ 0.03 0.02 _ 0.001
pageBlocks 0.01 _ 0.004 0.002 _ 0.00004 0.02 _ 0.00003 0.03 _ 0.01 0.003 _ 0.00007
pima 0.02 _ 0.01 0.02 _ 0.001 0.01 _ 0.01 0.02 _ 0.02 0.02 _ 0.001
ticTacToe 0.06 _ 0.02 0.01 _ 0.0003 0.07 _ 0.02 0.08 _ 0.02 0.02 _ 0.002
wine 0.01 _ 0.007 0.01 _ 0.002 0.02 _ 0.01 0.02 _ 0.02 0.04 _ 0.01

Table 6: The results of the setminus experiments. The percentages denote the size of the remaining data set. The ADM and Size scores are averages _ standard deviation