Sašo Džeroski
Jan Struyf (Eds.)

LNCS 4747

Knowledge Discovery in Inductive Databases

5th International Workshop, KDID 2006
Berlin, Germany, September 2006
Revised Selected and Invited Papers

Springer

Sašo Džeroski Jan Struyf (Eds.)

Knowledge Discovery in Inductive Databases

5th International Workshop, KDID 2006
Berlin, Germany, September 18, 2006
Revised Selected and Invited Papers

Volume Editors

Sašo Džeroski
Jožef Stefan Institute
Department of Knowledge Technologies
Jamova 39, 1000 Ljubljana, Slovenia
E-mail: saso.dzeroski@ijs.si

Jan Struyf
Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A, 3001 Leuven, Belgium
E-mail: jan.struyf@cs.kuleuven.be

Library of Congress Control Number: 2007937944

CR Subject Classification (1998): H.2, I.2

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

ISSN 0302-9743
ISBN-10 3-540-75548-9 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-75548-7 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12171675 06/3180 5 4 3 2 1 0

Lecture Notes in Computer Science 4747

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Preface

The 5th International Workshop on Knowledge Discovery in Inductive Databases (KDID 2006) was held on September 18, 2006 in Berlin, Germany, in conjunction with ECML/PKDD 2006: The 17th European Conference on Machine Learning (ECML) and the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD).

Inductive databases (IDBs) represent a database view on data mining and knowledge discovery. IDBs contain not only data, but also generalizations (patterns and models) valid in the data. In an IDB, ordinary queries can be used to access and manipulate data, while inductive queries can be used to generate (mine), manipulate, and apply patterns. In the IDB framework, patterns become "first-class citizens", and KDD becomes an extended querying process in which both the data and the patterns/models that hold in the data are queried.

The IDB framework is appealing as a general framework for data mining, because it employs declarative queries instead of ad-hoc procedural constructs. As declarative queries are often formulated using constraints, inductive querying is closely related to constraint-based data mining. The IDB framework is also appealing for data mining applications, as it supports the entire KDD process, i.e., nontrivial multi-step KDD scenarios, rather than just individual data mining operations. The goal of the workshop was to bring together database and data mining researchers interested in the areas of inductive databases, inductive queries, constraint-based data mining, and data mining query languages.

This workshop followed the previous four successful KDID workshops organized in conjunction with ECML/PKDD: KDID 2002 held in Helsinki, Finland, KDID 2003 held in Cavtat-Dubrovnik, Croatia, KDID 2004 held in Pisa, Italy, and KDID 2005 held in Porto, Portugal. Its scientific program included nine regular presentations and two short ones, as well as an invited talk by Kiri L. Wagstaff (Jet Propulsion Laboratory, California Institute of Technology, USA). This volume bundles all papers presented at the workshop and, in addition, includes three contributions that cover relevant research presented at other venues. We also include an article by one of the editors (SD) that attempts to unify existing research in the area and outline directions for further research towards a general framework for data mining.

We wish to thank the invited speaker, all the authors of submitted papers, the program committee members and additional reviewers, and the ECML/PKDD organization committee. KDID 2006 was supported by the European project IQ ("Inductive Queries for Mining Patterns and Models", IST FET FP6-516169).

July 2007

Sašo Džeroski
Jan Struyf

Organization

Program Chairs

Sašo Džeroski — Department of Knowledge Technologies
Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
saso.dzeroski@ijs.si
http://www-ai.ijs.si/SasoDzeroski/

Jan Struyf — Department of Computer Science
Katholieke Universiteit Leuven
Celestijnenlaan 200A, 3001 Leuven, Belgium
jan.struyf@cs.kuleuven.be
http://www.cs.kuleuven.be/~jan/

Program Committee

Hiroki Arimura, Hokkaido University, Japan
Hendrik Blockeel, Katholieke Universiteit Leuven, Belgium
Francesco Bonchi, ISTI-C.N.R., Italy
Jean-François Boulicaut, INSA Lyon, France
Toon Calders, University of Antwerp, Belgium
Luc De Raedt, Katholieke Universiteit Leuven, Belgium
Minos N. Garofalakis, Intel Research Berkeley, USA
Fosca Giannotti, ISTI-C.N.R., Italy
Bart Goethals, University of Antwerp, Belgium
Jiawei Han, University Illinois at Urbana-Champaign, USA
Ross D. King, University of Wales, Aberystwyth, UK
Giuseppe Manco, ICAR-C.N.R., Italy
Rosa Meo, University of Turin, Italy
Ryszard S. Michalski, George Mason University, USA
Taneli Mielikäinen, University of Helsinki, Finland
Shinichi Morishita, University of Tokyo, Japan
Siegfried Nijssen, Katholieke Universiteit Leuven, Belgium
Céline Robardet, INSA Lyon, France
Arno Siebes, Utrecht University, The Netherlands
Takashi Washio, Osaka University, Japan
Philip S. Yu, IBM Thomas J. Watson, USA
Mohammed Zaki, Rensselaer Polytechnic Institute, USA
Carlo Zaniolo, UCLA, USA

Additional Reviewers

Annalisa Appice
Marko Bohanec
Emma L. Byrne
Hong Cheng
Amanda Clare
Francesco Folino
Gemma Garriga
Kenneth A. Kaufman
Elio Masciari
Riccardo Ortale
Jimeng Sun
Janusz Wojtusiak

Lecture Notes in Computer Science

Sublibrary 3: Information Systems and Application, incl. Internet/Web and HCI

For information about Vols. 1– 4312
please contact your bookseller or Springer

Vol. 4797: M. Arenas, M.I. Schwartzbach (Eds.), Database Programming Languages. VIII, 261 pages. 2007.

Vol. 4796: M. Lew, N. Sebe, T.S. Huang, E.M. Bakker (Eds.), Human–Computer Interaction. X, 157 pages. 2007.

Vol. 4777: S. Bhalla (Ed.), Databases in Networked Information Systems. X, 329 pages. 2007.

Vol. 4761: R. Obermaisser, Y. Nah, P. Puschner, F.J. Rammig (Eds.), Software Technologies for Embedded and Ubiquitous Systems. XIV, 563 pages. 2007.

Vol. 4747: S. Džeroski, J. Struyf (Eds.), Knowledge Discovery in Inductive Databases. X, 301 pages. 2007.

Vol. 4740: L. Ma, M. Rauterberg, R. Nakatsu (Eds.), Entertainment Computing – ICEC 2007. XXX, 480 pages. 2007.

Vol. 4730: C. Peters, P. Clough, F.C. Gey, J. Karlgren, B. Magnini, D.W. Oard, M. de Rijke, M. Stempfhuber (Eds.), Evaluation of Multilingual and Multi-modal Information Retrieval. XXIV, 998 pages. 2007.

Vol. 4723: M. R. Berthold, J. Shawe-Taylor, N. Lavrač (Eds.), Advances in Intelligent Data Analysis VII. XIV, 380 pages. 2007.

Vol. 4721: W. Jonker, M. Petković (Eds.), Secure Data Management. X, 213 pages. 2007.

Vol. 4718: J. Hightower, B. Schiele, T. Strang (Eds.), Location- and Context-Awareness. X, 297 pages. 2007.

Vol. 4717: J. Krumm, G.D. Abowd, A. Seneviratne, T. Strang (Eds.), UbiComp 2007: Ubiquitous Computing. XIX, 520 pages. 2007.

Vol. 4715: J.M. Haake, S.F. Ochoa, A. Cechich (Eds.), Groupware: Design, Implementation, and Use. XIII, 355 pages. 2007.

Vol. 4714: G. Alonso, P. Dadam, M. Rosemann (Eds.), Business Process Management. XIII, 418 pages. 2007.

Vol. 4704: D. Barbosa, A. Bonifati, Z. Bellahsène, E. Hunt, R. Unland (Eds.), Database and XML Technologies. X, 141 pages. 2007.

Vol. 4690: Y. Ioannidis, B. Novikov, B. Rachev (Eds.), Advances in Databases and Information Systems. XIII, 377 pages. 2007.

Vol. 4675: L. Kovács, N. Fuhr, C. Meghini (Eds.), Research and Advanced Technology for Digital Libraries. XVII, 585 pages. 2007.

Vol. 4674: Y. Luo (Ed.), Cooperative Design, Visualization, and Engineering. XIII, 431 pages. 2007.

Vol. 4663: C. Baranauskas, P. Palanque, J. Abascal, S.D.J. Barbosa (Eds.), Human-Computer Interaction – INTERACT 2007, Part II. XXXIII, 735 pages. 2007.

Vol. 4662: C. Baranauskas, P. Palanque, J. Abascal, S.D.J. Barbosa (Eds.), Human-Computer Interaction – INTERACT 2007, Part I. XXXIII, 637 pages. 2007.

Vol. 4658: T. Enokido, L. Barolli, M. Takizawa (Eds.), Network-Based Information Systems. XIII, 544 pages. 2007.

Vol. 4656: M.A. Wimmer, J. Scholl, Å. Grönlund (Eds.), Electronic Government. XIV, 450 pages. 2007.

Vol. 4655: G. Psaila, R. Wagner (Eds.), E-Commerce and Web Technologies. VII, 229 pages. 2007.

Vol. 4654: I.-Y. Song, J. Eder, T.M. Nguyen (Eds.), Data Warehousing and Knowledge Discovery. XVI, 482 pages. 2007.

Vol. 4653: R. Wagner, N. Revell, G. Pernul (Eds.), Database and Expert Systems Applications. XXII, 907 pages. 2007.

Vol. 4636: G. Antoniou, U. Aßmann, C. Baroglio, S. Decker, N. Henze, P.-L. Patranjan, R. Tolksdorf (Eds.), Reasoning Web. IX, 345 pages. 2007.

Vol. 4611: J. Indulska, J. Ma, L.T. Yang, T. Ungerer, J. Cao (Eds.), Ubiquitous Intelligence and Computing. XXIII, 1257 pages. 2007.

Vol. 4607: L. Baresi, P. Fraternali, G.-J. Houben (Eds.), Web Engineering. XVI, 576 pages. 2007.

Vol. 4606: A. Pras, M. van Sinderen (Eds.), Dependable and Adaptable Networks and Services. XIV, 149 pages. 2007.

Vol. 4605: D. Papadias, D. Zhang, G. Kollios (Eds.), Advances in Spatial and Temporal Databases. X, 479 pages. 2007.

Vol. 4602: S. Barker, G.-J. Ahn (Eds.), Data and Applications Security XXI. X, 291 pages. 2007.

Vol. 4601: S. Spaccapietra, P. Atzeni, F. Fages, M.-S. Hacid, M. Kifer, J. Mylopoulos, B. Pernici, P. Shvaiko, J. Trujillo, I. Zaihrayeu (Eds.), Journal on Data Semantics IX. XV, 197 pages. 2007.

Vol. 4592: Z. Kedad, N. Lammari, E. Métais, F. Meziane, Y. Rezgui (Eds.), Natural Language Processing and Information Systems. XIV, 442 pages. 2007.

Vol. 4587: R. Cooper, J. Kennedy (Eds.), Data Management. XIII, 259 pages. 2007.

Vol. 4577: N. Sebe, Y. Liu, Y.-t. Zhuang, T.S. Huang (Eds.), Multimedia Content Analysis and Mining. XIII, 513 pages. 2007.

Vol. 4568: T. Ishida, S. R. Fussell, P. T. J. M. Vossen (Eds.), Intercultural Collaboration. XIII, 395 pages. 2007.

Vol. 4566: M.J. Dainoff (Ed.), Ergonomics and Health Aspects of Work with Computers. XVIII, 390 pages. 2007.

Vol. 4564: D. Schuler (Ed.), Online Communities and Social Computing. XVII, 520 pages. 2007.

Vol. 4563: R. Shumaker (Ed.), Virtual Reality. XXII, 762 pages. 2007.

Vol. 4561: V.G. Duffy (Ed.), Digital Human Modeling. XXIII, 1068 pages. 2007.

Vol. 4560: N. Aykin (Ed.), Usability and Internationalization, Part II. XVIII, 576 pages. 2007.

Vol. 4559: N. Aykin (Ed.), Usability and Internationalization, Part I. XVIII, 661 pages. 2007.

Vol. 4558: M.J. Smith, G. Salvendy (Eds.), Human Interface and the Management of Information, Part II. XXIII, 1162 pages. 2007.

Vol. 4557: M.J. Smith, G. Salvendy (Eds.), Human Interface and the Management of Information, Part I. XXII, 1030 pages. 2007.

Vol. 4541: T. Okadome, T. Yamazaki, M. Makhtari (Eds.), Pervasive Computing for Quality of Life Enhancement. IX, 248 pages. 2007.

Vol. 4537: K.C.-C. Chang, W. Wang, L. Chen, C.A. Ellis, C.-H. Hsu, A.C. Tsoi, H. Wang (Eds.), Advances in Web and Network Technologies, and Information Management. XXIII, 707 pages. 2007.

Vol. 4531: J. Indulska, K. Raymond (Eds.), Distributed Applications and Interoperable Systems. XI, 337 pages. 2007.

Vol. 4526: M. Malek, M. Reitenspieß, A. van Moorsel (Eds.), Service Availability. X, 155 pages. 2007.

Vol. 4524: M. Marchiori, J.Z. Pan, C.d.S. Marie (Eds.), Web Reasoning and Rule Systems. XI, 382 pages. 2007.

Vol. 4519: E. Franconi, M. Kifer, W. May (Eds.), The Semantic Web: Research and Applications. XVIII, 830 pages. 2007.

Vol. 4518: N. Fuhr, M. Lalmas, A. Trotman (Eds.), Comparative Evaluation of XML Information Retrieval Systems. XII, 554 pages. 2007.

Vol. 4508: M.-Y. Kao, X.-Y. Li (Eds.), Algorithmic Aspects in Information and Management. VIII, 428 pages. 2007.

Vol. 4506: D. Zeng, I. Gotham, K. Komatsu, C. Lynch, M. Thurmond, D. Madigan, B. Lober, J. Kvach, H. Chen (Eds.), Intelligence and Security Informatics: Biosurveillance. XI, 234 pages. 2007.

Vol. 4505: G. Dong, X. Lin, W. Wang, Y. Yang, J.X. Yu (Eds.), Advances in Data and Web Management. XXII, 896 pages. 2007.

Vol. 4504: J. Huang, R. Kowalczyk, Z. Maamar, D. Martin, I. Müller, S. Stoutenburg, K.P. Sycara (Eds.), Service-Oriented Computing: Agents, Semantics, and Engineering. X, 175 pages. 2007.

Vol. 4500: N.A. Streitz, A.D. Kameas, I. Mavrommati (Eds.), The Disappearing Computer. XVIII, 304 pages. 2007.

Vol. 4495: J. Krogstie, A. Opdahl, G. Sindre (Eds.), Advanced Information Systems Engineering. XVI, 606 pages. 2007.

Vol. 4480: A. LaMarca, M. Langheinrich, K.N. Truong (Eds.), Pervasive Computing. XIII, 369 pages. 2007.

Vol. 4473: D. Draheim, G. Weber (Eds.), Trends in Enterprise Application Architecture. X, 355 pages. 2007.

Vol. 4471: P. Cesar, K. Chorianopoulos, J.F. Jensen (Eds.), Interactive TV: A Shared Experience. XIII, 236 pages. 2007.

Vol. 4469: K.-c. Hui, Z. Pan, R.C.-k. Chung, C.C.L. Wang, X. Jin, S. Göbel, E.C.-L. Li (Eds.), Technologies for E-Learning and Digital Entertainment. XVIII, 974 pages. 2007.

Vol. 4443: R. Kotagiri, P. Radha Krishna, M. Mohania, E. Nantajeewarawat (Eds.), Advances in Databases: Concepts, Systems and Applications. XXI, 1126 pages. 2007.

Vol. 4439: W. Abramowicz (Ed.), Business Information Systems. XV, 654 pages. 2007.

Vol. 4430: C.C. Yang, D. Zeng, M. Chau, K. Chang, Q. Yang, X. Cheng, J. Wang, F.-Y. Wang, H. Chen (Eds.), Intelligence and Security Informatics. XII, 330 pages. 2007.

Vol. 4425: G. Amati, C. Carpineto, G. Romano (Eds.), Advances in Information Retrieval. XIX, 759 pages. 2007.

Vol. 4412: F. Stajano, H.J. Kim, J.-S. Chae, S.-D. Kim (Eds.), Ubiquitous Convergence Technology. XI, 302 pages. 2007.

Vol. 4402: W. Shen, J.-Z. Luo, Z. Lin, J.-P.A. Barthès, Q. Hao (Eds.), Computer Supported Cooperative Work in Design III. XV, 763 pages. 2007.

Vol. 4398: S. Marchand-Maillet, E. Bruno, A. Nürnberger, M. Detyniecki (Eds.), Adaptive Multimedia Retrieval: User, Context, and Feedback. XI, 269 pages. 2007.

Vol. 4397: C. Stephanidis, M. Pieper (Eds.), Universal Access in Ambient Intelligence Environments. XV, 467 pages. 2007.

Vol. 4380: S. Spaccapietra, P. Atzeni, F. Fages, M.-S. Hacid, M. Kifer, J. Mylopoulos, B. Pernici, P. Shvaiko, J. Trujillo, I. Zaihrayeu (Eds.), Journal on Data Semantics VIII. XV, 219 pages. 2007.

Vol. 4365: C.J. Bussler, M. Castellanos, U. Dayal, S. Navathe (Eds.), Business Intelligence for the Real-Time Enterprises. IX, 157 pages. 2007.

Vol. 4353: T. Schwentick, D. Suciu (Eds.), Database Theory – ICDT 2007. XI, 419 pages. 2006.

Vol. 4352: T.-J. Cham, J. Cai, C. Dorai, D. Rajan, T.-S. Chua, L.-T. Chia (Eds.), Advances in Multimedia Modeling, Part II. XVIII, 743 pages. 2006.

Vol. 4351: T.-J. Cham, J. Cai, C. Dorai, D. Rajan, T.-S. Chua, L.-T. Chia (Eds.), Advances in Multimedia Modeling, Part I. XIX, 797 pages. 2006.

Vol. 4328: D. Penkler, M. Reitenspiess, F. Tam (Eds.), Service Availability. X, 289 pages. 2006.

Vol. 4321: P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), The Adaptive Web. XII, 763 pages. 2007.

Vol. 4317: S.K. Madria, K.T. Claypool, R. Kannan, P. Uppuluri, M.M. Gore (Eds.), Distributed Computing and Internet Technology. XIX, 466 pages. 2006.

Table of Contents

Invited Talk

Contributed Papers

Discussion Paper

Value, Cost, and Sharing: Open Issues in Constrained Clustering

Kiri L. Wagstaff

Jet Propulsion Laboratory, California Institute of Technology,
Mail Stop 126-347, 4800 Oak Grove Drive, Pasadena CA 91109, USA
kiri.wagstaff@jpl.nasa.gov

Abstract. Clustering is an important tool for data mining, since it can identify major patterns or trends without any supervision (labeled data). Over the past five years, semi-supervised (constrained) clustering methods have become very popular. These methods began with incorporating pairwise constraints and have developed into more general methods that can learn appropriate distance metrics. However, several important open questions have arisen about which constraints are most useful, how they can be actively acquired, and when and how they should be propagated to neighboring points. This position paper describes these open questions and suggests future directions for constrained clustering research.

1 Introduction

Clustering methods are used to analyze data sets that lack any supervisory information such as data labels. They identify major patterns or trends based on a combination of the assumed cluster structure (e.g., Gaussian distribution) and the observed data distribution. Recently, semi-supervised clustering methods have become very popular because they can also take advantage of supervisory information when it is available. This supervision often takes the form of a set of pairwise *constraints* that specify known relationships between pairs of data items. Constrained clustering methods incorporate and enforce these constraints. This process is not just a fix for suboptimal distance metrics; it is quite possible for different users to have different goals in mind when analyzing the same data set. Constrained clustering methods permit the clustering results to be individually tailored for these different goals.

The initial work in constrained clustering has led to further study of the impact of incorporating constraints into clustering algorithms, particularly when applied to large, real-world data sets. Important issues that have arisen include:

1. Given the recent observation that some constraint sets can *adversely* impact performance, how can we determine the utility of a given constraint set, prior to clustering?
2. How can we minimize the effort required of the user, by active soliciting only the most useful constraints?

S. Džeroski and J. Struyf (Eds.): KDID 2006, LNCS 4747, pp. 1–10, 2007.

3. When and how should constraints be propagated or shared with neighboring points?

This paper begins with a description of the constrained clustering problem and surveys existing methods for finding satisfying solutions (Section 2). This overview is meant to be representative rather than comprehensive. Section 3 contributes more detailed descriptions of each of these open questions. In identifying these challenges, and the state of the art in addressing them, we highlight several directions for future research.

2 Constrained Clustering

We specify a clustering problem as a scenario in which a user wishes to obtain a partition $\mathcal{P}$ of a data set $\mathcal{D}$, containing n items, into k clusters or groups. A *constrained clustering* problem is one in which the user has some pre-existing knowledge about their desired $\mathcal{P}^*$. Usually, $\mathcal{P}^*$ is not fully known; if it were, no clustering would be necessary. Instead, the user is only able to provide a partial view $\mathcal{V}(\mathcal{P}^*)$. In this case, rather than returning $\mathcal{P}$ that best satisfies the (generic) objective function used by the clustering algorithm, we require that the algorithm adapt its solution to accommodate $\mathcal{V}(\mathcal{P}^*)$.

2.1 Pairwise Constraints

A partition $\mathcal{P}$ can be completely specified by stating, for each pairwise relationship (d_i, d_j) where $d_i, d_j \in \mathcal{D}$ and $d_i \neq d_j$, whether the pair of items is in the same cluster or split between different cluster. When used to specify requirements about the output partition, we refer to these statements as *must-link* and *cannot-link* constraints, respectively [1,2]. The number of distinct constraints ranges from 1 to $\frac{1}{2}n(n-1)$, since constraints are by definition symmetric. It is often the case that additional information can be automatically inferred from the partial set of constraints specified by the user. Cluster membership is an equivalence relation, so the must-link relationships are symmetric and transitive. Cannot-link relationships are symmetric but not necessarily transitive. When constraints of both kinds are present, an *entailment* relationship permits the discovery of additional constraints implied by the user-specified set [2,3].

The first work in this area proposed a modified version of COBWEB that enforced pairwise must-link and cannot-link constraints [1]. It was followed by an enhanced version of the widely used k-means algorithm that could also accommodate constraints, called COP-KMEANS [2]. Table 1 reproduces the details of this algorithm. COP-KMEANS takes in a set of must-link ($Con_=$) and cannot-link ($Con_\neq$) constraints. The essential change from the basic k-means algorithm occurs in step (2), where the decision about where to assign a given item d_i is constrained so that no constraints in $Con_=$ or $Con_\neq$ are violated. The satisfying condition is checked by the VIOLATE-CONSTRAINTS function. Note that it is possible for there to be no solutions that satisfy all constraints, in which case the algorithm exits prematurely.

Table 1. Constrained K-means Algorithm for hard, pairwise constraints [2]

COP-KMEANS(data set D, number of clusters k, must-link constraints $Con_{=} \subset D \times D$, cannot-link constraints $Con_{\neq} \subset D \times D$)

1. Let $C_1 \ldots C_k$ be the k initial cluster centers.
2. For each point $d_i \in D$, assign it to the closest cluster C_j such that VIOLATE-CONSTRAINTS($d_i, C_j, Con_{=}, Con_{\neq}$) is false. If no such cluster exists, fail (return $\{\}$).
3. For each cluster C_i, update its center by averaging all of the points d_j that have been assigned to it.
4. Iterate between (2) and (3) until convergence.
5. Return $\{C_1 \ldots C_k\}$.

VIOLATE-CONSTRAINTS(data point d, cluster C, must-link constraints $Con_{=} \subset D \times D$, cannot-link constraints $Con_{\neq} \subset D \times D$)

1. For each $(d, d_{=}) \in Con_{=}$: If $d_{=} \notin C$, return true.
2. For each $(d, d_{\neq}) \in Con_{\neq}$: If $d_{\neq} \in C$, return true.
3. Otherwise, return false.

A drawback of this approach is that it may fail to find a satisfying solution even when one exists. This happens because of the greedy fashion in which items are assigned; early assignments can constrain later ones due to potential conflicts, and there is no mechanism for backtracking. As a result, the algorithm is sensitive to the order in which it processes the data set D. In practice, this is resolved by running the algorithm multiple times with different orderings of the data, but for data sets with a large number of constraints (especially cannot-link constraints), early termination without a solution can be a persistent problem. We previously assessed the hardness of this problem by generating constraint sets of varying sizes for the same data set and found that convergence failures happened most often for problems with an intermediate number of constraints, with respect to the number of items in the data set. This is consistent with the finding that 3-SAT formulas with intermediate complexity tend to be most difficult to solve [4].

In practice, however, this algorithm has proven very effective on a variety of data sets. Initial experiments used several data sets from the UCI repository [5], using constraints artificially generated from the known data labels. In addition, experimental results on a real-world problem showed the benefits of using a constrained clustering method when pre-existing knowledge is available. In this application, data from cars with GPS receivers were collected as they traversed repeatedly over the same roads. The goal was to cluster the data points to identify the road lanes, permitting the automatic refinement of digital maps to the individual lane level. By expressing domain knowledge about the contiguity of a given car's trajectory and a maximum reasonable separation between lanes in the form of pairwise constraints, lane-finding performance increased from 58.0% without constraints to 98.6% with constraints [2]. A natural follow-on to this work was the development of a constrained version of the EM clustering algorithm [6].

Soft Constraints. When the constraints are known to be completely reliable, treating them as hard constraints is an appropriate approach. However, since the constraints may be derived from heuristic domain knowledge, it is also useful to have a more flexible approach. There are two kinds of uncertainty that we may wish to capture: (1) the constraints are noisy, so we should permit some of them to be violated if there is overwhelming evidence against them (from other data items), and (2) we have knowledge about the *likelihood* that a given constraint should be satisfied, so we should permit the expression of a probabilistic constraint. The SCOP-KMEANS algorithm is a more general version of COP-KMEANS algorithm that treats constraint statements as soft constraints, addressing the issue of noise in the constraints [7]. Rather than requiring that every constraint be satisfied, it instead trades off the objective function (variance) against constraint violations, penalizing for each violation but permitting a violation if it provides a significant boost to the quality of the solution. Other approaches, such as the MPCK-means algorithm, permit the specification of an individual weight for each constraint, addressing the issue of variable per-constraint confidences [3]. MPCK-means imposes a penalty for constraint violations that is proportional to the violated constraint's weight.

Metric Learning. It was recognized early on that constraints could provide information not only about the desired solution, but also more general information about the metric space in which the clusters reside. A must-link constraint (d_i, d_j) can be interpreted as a hint that the conceptual distance between d_i and d_j is small. Likewise, a cannot-link constraint implies that the distance between d_i and d_j is so great that they should never be clustered together. Rather than using a modified clustering algorithm to enforce these individual constraints, it is also possible to use the constraints to learn a new metric over the feature space and then apply regular clustering algorithms, using the new metric. Several such metric learning approaches have been developed; some are restricted to learning from must-link constraints only [8], while others can also accommodate cannot-link constraints [9,10]. The MPCK-means algorithm fuses both of these approaches (direct constraint satisfaction and metric learning) into a single architecture [3].

2.2 Beyond Pairwise Constraints

There are other kinds of knowledge that a user may have about the desired partition $\mathcal{P}^*$, aside from pairwise constraints. Cluster-level constraints include existential constraints, which require that a cluster contain at least c_{min} items [11,12] and capacity constraints, which require that a cluster must have less than c_{max} items [13].

The user may also wish to express constraints on the features. *Co-clustering* is the process of identifying subsets of items in the data set that are similar with respect to a subset of the features. That is, both the items and the features are clustered. In essence, co-clustering combines data clustering with feature selection and can provide new insights into a data set. For data sets in which the

features have a pre-defined ordering, such as a temporal (time series) or spatial ordering, it can be useful to express interval/non-interval constraints on how the features are selected by a co-clustering algorithm [14].

3 Open Questions

The large body of existing work on constrained clustering has achieved several important algorithmic advances. We have now reached the point where more fundamental issues have arisen, challenging the prevailing view that constraints are always beneficial and examining how constraints can be used for real problems, in which scalability and the user effort required to provide constraints may impose an unreasonable burden. In this section, we examine these important questions, including how the utility of a given constraint set can be quantified (Section 3.1), how we can minimize the cost of constraint acquisition (Section 3.2), and how we can propagate constraint information to nearby regions to minimize the number of constraints needed (Section 3.3).

3.1 Value: How Useful Is a Given Set of Constraints?

It is to be expected that some constraint sets will be more useful than others, in terms of the benefit they provide to a given clustering algorithm. For example, if the constraints contain information that the clustering algorithm is able to deduce on its own, then they will not provide any improvement in clustering performance. However, virtually all work to date values constraint sets only in terms of the number of constraints they contain. The ability to more accurately quantify the utility of a given constraint set, prior to clustering, will permit practitioners to decide whether to use a given constraint set, or to choose the best constraint set to use, when several are available.

The need for a constraint set utility measure has become imperative with the recent observation that some constraint sets, even when completely accurate with respect to the evaluation labels, can actually decrease clustering performance [15]. The usual practice when describing the results of constrained clustering experiments is to report the clustering performance averaged over multiple trials, where each trial consists of a set of constraints that is randomly generated from the data labels. While it is generally the case that average performance does increase as more constraints are provided, a closer examination of the individual trials reveals that some, or even many, of them instead cause a drop in accuracy. Table 2 shows the results of 1000 trials, each with a different set of 25 randomly selected constraints, conducted over four UCI data sets [5] using four different k-means-based constrained clustering algorithms. The table reports the fraction of trials in which the performance was lower than the default (unconstrained) k-means result, which ranges from 0% up to 87% of the trials.

The average performance numbers obscure this effect because the "good" trials tend to have a larger magnitude change in performance than the "bad" trials do. However, the fact that any of the constraint sets can cause a decrease in

Table 2. Fraction of 1000 randomly selected 25-constraint sets that caused a drop in accuracy, compared to an unconstrained run with the same centroid initialization (table from Davidson et al. [15])

	Algorithm			
	CKM [2]	**PKM [3]**	**MKM [3]**	**MPKM [3]**
Data Set	Constraint enforcement	Constraint enforcement	Metric learning	Enforcement and metric learning
Glass	28%	1%	11%	0%
Ionosphere	26%	77%	0%	77%
Iris	29%	19%	36%	36%
Wine	38%	34%	87%	74%

performance is unintuitive, and even worrisome, since the constraints are known to be noise-free and should not lead the algorithm astray.

To better understand the reasons for this effect, Davidson et al. [15] defined two constraint set properties and provided a quantitative way to measure them. *Informativeness* is the fraction of information in the constraint set that the algorithm cannot determine on its own. *Coherence* is the amount of agreement between the constraints in the set. Constraint sets with low coherence will be difficult to completely satisfy and can lead the algorithm into unpromising areas of the search space. Both high informativeness and high coherence tend to result in an increase in clustering performance. However, these properties do not fully explain some clustering behavior. For example, a set of just three randomly selected constraints, with high informativeness and coherence, can increase clustering performance on the `iris` data set significantly, while a constraint set with similarly high values for both properties has no effect on the `ionosphere` data set. Additional work must be done to refine these measures or propose additional ones that better characterize the utility of the constraint set.

Two challenges for future progress in this area are: 1) to identify other constraint set properties that correlate with utility for constrained clustering algorithms, and 2) to learn to predict the overall utility of a new constraint set, based on extracted attributes such as these properties. It is likely that the latter will require the combination of several different constraint set properties, rather than being a single quantity, so using machine learning techniques to identify the mapping from properties to utility may be a useful approach.

3.2 Cost: How Can We Make Constraints Cheaper to Acquire?

A single pairwise constraint specifies a relationship between two data points. For a data set with n items, there are $\frac{1}{2}n(n-1)$ possible constraints. Therefore, the number of constraints needed to specify a given percentage of the relationships (say, 10%) increases quadratically with the data set size. For large data sets, the constraint specification effort can become a significant burden.