

Constraint Programming meets Machine Learning and Data Mining

Edited by

Luc De Raedt¹, Siegfried Nijssen², Barry O’Sullivan³, and Pascal Van Hentenryck⁴

1 K.U. Leuven, BE, luc.deraedt@cs.kuleuven.be

2 K.U. Leuven, BE, siegfried.nijssen@cs.kuleuven.be

3 University College Cork, IE, b.osullivan@cs.ucc.ie

4 Brown University – Providence, US, pvh@cs.brown.edu

Abstract

This report documents the programme and the outcomes of Dagstuhl Seminar 11201 *Constraint Programming meets Machine Learning and Data Mining*. Our starting point in this seminar was that machine learning and data mining have developed largely independently from constraint programming till now, but that it is increasingly becoming clear that there are many opportunities for interactions between these areas: on the one hand, data mining and machine learning can be used to improve constraint solving; on the other hand, constraint solving can be used in data mining in machine learning. This seminar brought together prominent researchers from both communities to discuss these opportunities.

Seminar 15.–20. May, 2011 – www.dagstuhl.de/11201

1998 ACM Subject Classification F.4.1. [Mathematical Logic: Logic and constraint programming]; H.2.8. [Database applications: Data mining]; I.2.6. [Learning]

Keywords and phrases Machine learning, data mining, constraint programming, constraints

Digital Object Identifier 10.4230/DagRep.1.5.61


1 Executive Summary

Luc De Raedt

Siegfried Nijssen

Barry O’Sullivan

Pascal Van Hentenryck

License  Creative Commons BY-NC-ND 3.0 Unported license
© Luc De Raedt, Siegfried Nijssen, Barry O’Sullivan, Pascal Van Hentenryck

Goal of the Seminar

Over the past two decades the fields of constraint programming, machine learning and data mining have become well-established research fields within computer science. They have contributed many foundational techniques that are routinely applied in real-life scientific and industrial applications. At the same time, awareness has grown that constraints can be very useful during mining and learning, and also that machine learning and data mining may allow one to automatically acquire constraints from data.

Both the data mining and machine learning communities have been interested in *constraint-based mining and learning*, that is, the use of constraints to formalize mining and learning problems. Examples are the specification of desirable properties of patterns to be mined,



Except where otherwise noted, content of this report is licensed under a Creative Commons BY-NC-ND 3.0 Unported license

Constraint Programming meets Machine Learning and Data Mining, *Dagstuhl Reports*, Vol. 1, Issue 5, pp. 61–83

Editors: Luc De Raedt, Siegfried Nijssen, Barry O’Sullivan, and Pascal Van Hentenryck



DAGSTUHL
REPORTS

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

or clusters to be found. The task of the data mining or machine learning system is to generate all patterns or to compute the optimal clustering satisfying the constraints. A wide variety of constraints for local pattern mining, clustering and other machine learning problems exist and they have been implemented in an even wider range of specific data mining and machine learning systems for supporting such constraints. Some of these methods are based on mathematical programming techniques, such as linear programming or quadratic programming; other problems, however, cannot be modeled using these techniques. So far, the machine learning and data mining communities have been unable to develop general solvers that are applicable to a wide range of machine learning and data mining problems.

On the other hand, the artificial intelligence community has studied several types of constraint-satisfaction solvers. The most general systems are now gathered in the area of *constraint programming*. In constraint programming, the user specifies the model, that is, the set of constraints to be satisfied and constraint solvers generate solutions. Thus, the goals of constraint programming and constraint based mining and learning are similar; it is only that constraint programming targets *any* type of constraint satisfaction problem, whereas constraint-based mining and learning *specifically* targets data mining and machine learning applications. Therefore, it is surprising that despite the similarities between these two endeavours, the two fields have evolved independently of one another, and also, that – with a few recent exceptions – constraint programming tools and techniques are not yet applied to data mining and machine learning, and, vice versa, that problems and challenges from data mining and machine learning have not yet been taken up by the constraint programming community. Exploring the possibilities for exploiting constraint programming in data mining and machine learning was one goal of this seminar.

The second goal was to study the use of machine learning and data mining in constraint programming. Practitioners of constraint programming have to formulate explicitly the constraints that underly their application. This is often a difficult task. Even when the right constraints are known, it can be challenging to formalize them in such a way that the constraint programming system can use them efficiently. This raises the question as to whether it is possible to (semi)- automatically learn such constraints or their formulations from data and experience. Again, some initial results in this direction exist, but we are away from a complete understanding of the potential of this approach.

In this seminar, we aimed at bridging the gap between these two fields by investigating, on the one hand, how standard constraint-programming techniques can be used in data mining and machine learning, and on the other hand, how machine learning and data mining can contribute to constraint programming. Therefore, this workshop brought together researchers in the areas of constraint programming, machine learning and data mining to discuss these issues, to identify interesting opportunities and challenges for research, and to consolidate and strengthen a promising line of research.

Seminar Organization

Given the various backgrounds of the participants, the seminar started with several introductory presentations. The focus was first on constraint programming and the use of machine learning and data mining in constraint programming; Helmut Simonis provided an introduction of constraint programming, while Barry O’Sullivan presented the possibilities for using machine learning and data mining in constraint programming.

Subsequently, the focused moved to data mining and machine learning and the possibilities

for using constraint solvers in data mining and machine learning. Bart Goethals provided an introduction to data mining; subsequently, Ian Davidson discussed uses of constraint solving in clustering, Dan Roth uses of solvers in inference problems, Siegfried Nijssen uses of solvers in pattern mining, and Tijl De Bie uses of solvers in statistical machine learning.

The remainder of the program consisted of a mix of technical presentations as well as meetings of discussion groups, each of which focused in more detail on the various possibilities for combining machine learning, data mining and constraint programming.

The topics of the discussion groups were determined after discussion at the first day of the seminar and were the following:

- declarative data mining, to discuss questions regarding the use of constraint programming solvers and declarative modelling to solve data mining problems;
- programming languages for machine learning, to discuss questions regarding the development of languages specifically for machine learning;
- applications, to discuss applications in which machine learning, data mining and constraint programming can be used;
- challenges, to discuss the possibilities for setting up benchmark problems –both real-life and artificial– that can be used to determine the success of combining machine learning, data mining and constraint programming;
- learning to solve, to discuss the use of machine learning and data mining to improve the efficiency and quality of constraint solving;
- learning constraints, to discuss the use of machine learning and data mining to learn appropriate models from examples.

The seminar was concluded with plenary presentations in which the results of the discussion groups were summarized.

The intention is to publish the results of the seminar in an edited book.

2 Table of Contents

Executive Summary

Luc De Raedt, Siegfried Nijssen, Barry O’Sullivan, Pascal Van Hentenryck 61

Overview of Talks

Fast Algorithms for Finding Extremal Sets

Roberto Bayardo 66

Declarative Experimentation

Hendrik Blockeel 66

Discovering knowledge by combining primitives in the constraint programming framework

Bruno Cremilleux 67

Bayesian network learning with cutting planes

James Cussens 67

Continuous Optimization – Problems and successes

Tijl De Bie 68

Reinforced Adaptive Large Neighborhood Search

Yves Deville 68

Two combinatorial optimization problems in Machine Learning

Pierre Dupont 69

Data Uncertainty and Constraint Programming Models

Carmen Gervet 70

Sequence Classification in High Dimensional Predictor Space

Georgiana Ifrim 71

Using and Learning Constraints in Inductive Process Modeling

Pat Langley 71

Search and Black-Box Constraint-Programming Solvers

Laurent Michel 72

MIME: Discovering Interesting Patterns Visually and Interactively

Sandy Moens 72

Constraint programming for Itemset Mining

Siegfried Nijssen 73

Integrations of Machine Learning and Data Mining in Constraint Satisfaction

Barry O’Sullivan 74

Constrained Conditional Models: Integer Linear Programming Formulations for Natural Language Understanding

Dan Roth 75

Gecode – an open constraint solving library

Christian Schulte 76

Lifelong learning in Constraint Solving


Michele Sebag 76

A Constraint Seeker: Finding and Ranking Global Constraints from Examples <i>Helmut Simonis</i>	77
APOPCALEAPS: Automatic Pop Composer And Learner of Parameters <i>Jon Sneyers</i>	78
MiniZinc – Towards a standard CP modelling language <i>Guido Tack</i>	78
On Learning Constraint Problems <i>Christel Vrain</i>	79
Working Groups	
Declarative Data Mining <i>Siegfried Nijssen</i>	79
Learning Constraint Satisfaction Problems <i>Luc De Raedt</i>	80
Learning in Constraint Solvers <i>Barry O’Sullivan</i>	81
Acknowledgements	82
Participants	83

3 Overview of Talks

3.1 Fast Algorithms for Finding Extremal Sets

Roberto Bayardo (Google Inc. – Mountain View, US)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Roberto Bayardo

Joint work of Bayardo, Roberto; Panda, Biswanath

Main reference Roberto J. Bayardo, Biswanath Panda, “Fast Algorithms for Finding Extremal Sets,” Proc. of 11th SIAM International Conference on Data Mining (SDM’11), pp. 25–34.


URL <http://siam.omnibooksonline.com/data/papers/089.pdf#page=1>

Identifying the extremal (minimal and maximal) sets from a collection of sets is an important subproblem in the areas of data-mining and satisfiability checking. For example, extremal set finding algorithms are used in the context of mining maximal frequent itemsets, and for simplifying large propositional satisfiability instances derived from real world tasks such as circuit routing and verification. We describe two new algorithms for the task and detail their performance on real and synthetic data.

Each algorithm leverages an entirely different principle – one primarily exploits set cardinality constraints, the other lexicographic constraints. Despite the inherent difficulty of this problem (the best known worst-case bounds are nearly quadratic), we show that both these algorithms provide excellent performance in practice, and can identify all extremal sets from multi-gigabyte itemset data using only a single processor core. Both algorithms are concise and can be implemented in no more than a few hundred lines of code.

3.2 Declarative Experimentation

Hendrik Blockeel (K.U. Leuven, BE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Hendrik Blockeel

Main reference Hendrik Blockeel, Joaquin Vanschoren, “Experiment Databases: Towards an Improved Experimental Methodology,” in 11th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD’07), pp. 6–17, 2007


URL http://dx.doi.org/10.1007/978-3-540-74976-9_5

Data mining and machine learning are highly experimental scientific areas: many results are backed up with empirical data. With the current state of the art, to find the answer to certain experimental questions or testing experimental hypotheses, one needs to specify the experiments in a completely procedural way.

This makes the process of setting up experiments and interpreting the results very error-prone. We believe it should be possible to describe questions or hypotheses declaratively, and let the computer set up the right experiments. To make this possible, a formal language for experimental questions or hypotheses is required. We expect that elements from constraint programming will play a major role in such a language. In this talk we will discuss a number of issues that arise in languages for declarative experimentation, and point out the links with constraint-based data mining.

3.3 Discovering knowledge by combining primitives in the constraint programming framework

Bruno Crémilleux (Université de Caen, FR)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Bruno Crémilleux

Joint work of Boizumault, Patrice; Crémilleux, Bruno; Khiari, Mehdi; Loudni, Samir; Métivier, Jean-Philippe
Main reference Patrice Boizumault, Bruno Crémilleux, Mehdi Khiari, Samir Loudni, Jean-Philippe Métivier, “Discovering Knowledge using a Constraint-based Language,” Dagstuhl Preprint Archive, arXiv:1107.3407v1 [cs.LG]
URL <http://arxiv.org/abs/1107.3407v1>

Discovering pattern sets or global patterns is an attractive issue from the pattern mining community in order to provide useful information. By combining local patterns satisfying a joint meaning, this approach produces patterns of higher level and thus more useful for the end-user than the usual and well-known local patterns, while reducing the number of patterns. In parallel, recent works investigating relationships between data mining and constraint programming (CP) show that the CP paradigm is a nice framework to model and mine such patterns in a declarative and generic way. In this talk, we present a set of primitives which enables us to define queries addressing patterns sets and global patterns.


We provide several examples of such queries. The usefulness to propose a declarative approach is highlighted by the example of the clustering based on associations. We express the primitives in the CP framework by using numeric constraints and set constraints. Then a CP solver performs a sound and complete resolution of the queries. Finally, we discuss of the implementation of our approach.

References

- 1 L. De Raedt, T. Guns, and S. Nijssen. Constraint Programming for Itemset Mining. In *ACM SIGKDD Int. Conf. KDD’08*, Las Vegas, Nevada, USA, 2008.
- 2 L. De Raedt, T. Guns, and S. Nijssen. Constraint programming for data mining and machine learning. In *Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, pages 1671–1675, 2010.
- 3 M. Khiari, P. Boizumault, and B. Crémilleux. Local constraint-based mining and set constraint programming for pattern discovery. In *From Local Patterns to Global Models (LeGo-09), ECML/PKDD-09 Workshop*, pages 61–76, Bled, Slovenia, 2009.
- 4 M. Khiari, P. Boizumault, and B. Crémilleux. Constraint programming for mining n-ary patterns. In *16th Int. Conf. on Principles and Practice of Constraint Programming (CP’10)*, volume 6308 of *LNCS*, pages 552–567. Springer, 2010.

3.4 Bayesian network learning with cutting planes

James Cussens (University of York, GB)

License  Creative Commons BY-NC-ND 3.0 Unported license
© James Cussens


Main reference James Cussens, “Bayesian network learning with cutting planes,” Proc. 27th Conference on Uncertainty in Artificial Intelligence (UAI 2011).
URL <http://www-users.cs.york.ac.uk/~jc/research/uai11>

The problem of learning the structure of Bayesian networks from complete discrete data with a limit on parent set size is considered. Learning is cast explicitly as an optimisation problem where the goal is to find a BN structure which maximises log marginal likelihood (BDe score). Integer programming, specifically the SCIP framework, is used to solve this

optimisation problem. Acyclicity constraints are added to the integer program (IP) during solving in the form of *cutting planes*. Finding good cutting planes is the key to the success of the approach—the search for such cutting planes is effected using a sub-IP. Results show that this is a particularly fast method for exact BN learning.

3.5 Continuous Optimization – Problems and successes

Tijl De Bie (University of Bristol, GB)

License  Creative Commons BY-NC-ND 3.0 Unported license
 © Tijl De Bie
URL <http://www.tijldebie.net>

Continuous optimization (or mathematical programming), and convex optimization in particular [1], has had a profound impact on certain areas of machine learning research in the past decade [2, 3]. This talk is intended to provide an overview of the reasons for these successes, as well as some of the limitations.

The focus of the talk is on similarities with constraint programming as a largely declarative way of formulating search problems through large parameter spaces.

References

- 1 Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press (2004).
- 2 Kristin Bennet and Emilio Parrado-Hernandez. The interplay of optimization and machine learning research. *Journal of Machine Learning Research* 7 (2006).
- 3 Tijl De Bie. Deploying SDP for machine learning. *Proceedings of the 15th European Symposium on Artificial Neural Networks (ESANN 2007)*, Bruges, April 2007.

3.6 Reinforced Adaptive Large Neighborhood Search

Yves Deville (Université Catholique de Louvain, BE)

Joint work of Mairy, Jean-Baptiste; Deville, Yves; Van Hentenryck, Pascal
License  Creative Commons BY-NC-ND 3.0 Unported license
 © Yves Deville

Large Neighborhood Search (LNS) is a hybrid search paradigm that combines local search (LS) and constraint programming (CP). At each step of LNS, a subset of variables of the current solution are relaxed. The potential solutions of the relaxed problems form the neighborhood. CP is used to solve the relaxed problem and tries to find a neighbor improving the current solution.

The CP part has a search limit. If no improving solution is found, the current solution is unchanged. In LNS, one has to fix the size of the relaxed fragment and the search limit. These two parameters are crucial in LNS. One has also to decide how the relaxed variables are chosen. Usually, a random selection is performed.


In this talk, we report preliminary results on applying reinforcement learning in LNS. Reinforcement learning is also used to guide the selection of the variables to relax at each LNS step.

References

- 1 Laurent Perron, Paul Shaw and Vincent Furnon. *Propagation Guided Large Neighborhood Search*. CP 2004, LNCS 3258, (2004) 468–481.
- 2 Laurent Perron and Paul Shaw. *Combining Forces to Solve the Car Sequencing Problem*. CPAIOR 2004, LNCS 3011, (2004) 225–239.
- 3 Richard S. Sutton, Andrew G. Barto. *Reinforcement learning: an introduction*. Cambridge, MA: MIT Press, 1998.

3.7 Two combinatorial optimization problems in Machine Learning

Pierre Dupont (UC Louvain-la-Neuve, BE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Pierre Dupont

In this talk we present two classical ML problems which can be formulated as combinatorial optimization problems. The open question is to which extent constraint programming could help to better address them.

Regular grammar induction aims at learning a formal regular language from sets of positive and negative strings. Learning a regular grammar is classically restated as the minimal DFA consistency problem: finding a minimal deterministic finite state machine accepting all positive strings and rejecting all negative ones.

We briefly describe the search space of this problem, present standard state-merging techniques and sketch the key properties of the winning algorithm of the recent Stamina competition.

Many supervised classification problems are characterized by a very large number (p) of features, but only a limited number (n) of samples. Transcriptome analysis from DNA microarray data is a typical example.

In such a small n big p setting, the risk of overfitting a classification model to the training sample is particularly present.

Feature selection aims at reducing, sometimes drastically, the number of actual features on which the classifier is effectively built.

The simplest approach selects top-ranked features according to a relevance index, such as the mutual information between each input feature and the response variable.

Such an approach is particularly efficient from a computational viewpoint but it is univariate and hence lacks to model the dependence between various features. A more advanced alternative aims at selecting a subset of maximally relevant but minimally redundant features.

Finding the optimal feature subset is computationally hard and typical approximations relies on a greedy search with no global optimum guarantee.


References

- 1 D. Angluin, *On the complexity of minimum inference of regular sets*, Information and Control **39** (1978), 337–350.
- 2 F. Coste and J. Nicolas, *How considering incompatible state mergings may reduce the DFA induction search tree*, Grammatical Inference, ICGI’98 (Ames, Iowa), Lecture Notes in Artificial Intelligence, no. 1433, Springer Verlag, 1998, pp. 199–210.
- 3 P. Dupont, B. Lambeau, C. Damas, and A. van Lamsweerde, *The QSM algorithm and its application to software behavior model induction*, Applied Artificial Intelligence **22** (2008), 77–115.

- 4 P. Dupont, L. Miclet, and E. Vidal, *What is the search space of the regular inference ?*, Grammatical Inference and Applications, ICGI'94 (Alicante, Spain), Lecture Notes in Artificial Intelligence, no. 862, Springer Verlag, 1994, pp. 25–37.
- 5 E.M. Gold, *Complexity of automaton identification from given data*, Information and Control **37** (1978), 302–320.
- 6 I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh (eds.), *Feature extraction, foundations and applications*, Series Studies in Fuzziness and Soft Computing, vol. 207, Springer, 2006.
- 7 T. Helleputte and P. Dupont, *Feature selection by transfer learning with linear regularized models*, European Conference on Machine Learning, Lecture Notes in Artificial Intelligence, no. 5781, 2009, pp. 533–547.
- 8 T. Helleputte and P. Dupont, *Partially supervised feature selection with regularized linear models*, International Conference on Machine Learning, 2009.
- 9 M. Heule and S. Verwer, *Exact DFA identification using SAT solvers*, International Colloquium on Grammatical Inference, Lecture Notes in Artificial Intelligence, vol. 6339, Springer Verlag, 2010, pp. 66–79.
- 10 B. Lambeau, C. Damas, and P. Dupont, *State-merging DFA induction algorithms with mandatory merge constraints*, Lecture Notes in Artificial Intelligence, vol. 5278, 2008, pp. 139–153.
- 11 G. Obozinski, B. Taskar, and M.I. Jordan, *Joint covariate selection and joint subspace selection for multiple classification problems*, Statistics and Computing (2009), 1–22.
- 12 H. Peng, F. Long, and C. Ding, *Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy*, IEEE Transactions on Pattern Analysis and Machine Intelligence **27** (2005), no. 8, 1226–1238.
- 13 Y. Saeys, I. Inza, and P. Larranaga, *A review of feature selection techniques in bioinformatics*, Bioinformatics **23** (2007), no. 19, 2507–2517.
- 14 N. Walkinshaw, K. Bogdanov, C. Damas, B. Lambeau, and P. Dupont, *A framework for the competitive evaluation of model inference techniques*, 1st International workshop on Model Inference In Testing, 2010.

3.8 Data Uncertainty and Constraint Programming Models

Carmen Gervet (German University in Cairo, EG)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Carmen Gervet

Joint work of Gervet, Carmen; Saad, Aya; Abdennadher, Slim
Main reference Aya Saad, Carmen Gervet, Slim Abdennadher, “Constraint Reasoning with Uncertain Data Using CDF-Intervals,” Proc. 7th Int’l Conf. on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR’10), pp. 292–306, 2010.
URL http://dx.doi.org/10.1007/978-3-642-13520-0_32

The classical CP framework has been extended in the past 10 years to address ill-defined, uncertain real-world optimisation problems. Extensions include probabilistic models, quantified models, interval data models. Each tackles different aspects of data uncertainty, due to data errors, incompleteness, forecasting and aims at ensuring that the problem is faithfully represented with what is known for sure about the data. The solution sought vary from robust solutions, to reliable solutions and covering set solutions. In all cases, the model does have a great impact on the algorithm complexity, and new approaches have to be implemented.

We summarize existing approaches and introduce a novel approach based on a new definition of the confidence interval .

3.9 Sequence Classification in High Dimensional Predictor Space

Georgiana Ifrim (*University College Cork, IE*)

License © © ⊖ Creative Commons BY-NC-ND 3.0 Unported license
© Georgiana Ifrim

Joint work of Ifrim, Georgiana; Wiuf, Carsten;

Main reference G. Ifrim, C. Wiuf, “Bounded Coordinate-Descent for Biological Sequence Classification in High Dimensional Predictor Space,” Proc. 17th ACM SIGKDD Int’l Conf. on Knowledge Discovery and Data Mining (SIGKDD’11), pp. 708–716, 2011.

URL <http://doi.acm.org/10.1145/2020408.2020519>

In this talk I present a framework for sequence classification where we build linear classifiers on a rich but very high dimensional feature space. I present an optimisation algorithm based on coordinate gradient-descent coupled with branch-and-bound to efficiently build such classifiers.

Furthermore, I discuss accuracy, interpretability and scalability of this learning framework. This technique can be applied to any classification problem where the input data can be represented in symbolic form as a sequence of tokens (e.g., text, biological sequences, time series).

References

- 1 G. Ifrim. Gradient-bounded coordinate-descent for sequence regression. <http://www.4c.ucc.ie/ifrim/seql/data>, 2010.
- 2 G. Ifrim, C. Wiuf. Bounded Coordinate-Descent for Biological Sequence Classification in High Dimensional Predictor Space In *KDD*, USA, 2011.
- 3 G. Ifrim, G. Bakir, and G. Weikum. Fast logistic regression for text categorization with variable-length n-grams. In *KDD*, pages 354–362, USA, 2008.
- 4 T. Kudo, E. Maeda and Y. Matsumoto. An Application of Boosting to Graph Classification. In *NIPS*, 2004.

3.10 Using and Learning Constraints in Inductive Process Modeling

Pat Langley (*ASU – Tempe, US*)

License © © ⊖ Creative Commons BY-NC-ND 3.0 Unported license
© Pat Langley

URL <http://www.isle.org/process/>

Most research on computational knowledge discovery has focused on descriptive models that only summarize data and utilized formalisms developed in AI or statistics. In contrast, scientists typically aim to develop explanatory models that make contact with background knowledge and use established scientific notations. In this talk, I present an approach to computational discovery that encodes scientific models as sets of processes that incorporate differential equations, simulates these models’ behavior over time, uses background knowledge to guide construction of model structures, and fits the model structures to time-series data. I illustrate this framework on data and models from a number of fields but focusing on aquatic ecosystems. One important form of background knowledge – constraints – is used to rule out implausible model structures. I also report an extension that learns new constraints based on the success and failure of candidate models. In closing, I discuss intellectual influences on the work and directions for future research.

This talk describes joint work at Stanford University and ISLE with Kevin Arrigo, Stuart Borrett, Matt Bravo, Will Bridewell, and Ljupco Todorovski under funding from the National


Science Foundation. The URL <http://www.isle.org/process/> includes a list of publications on inductive process modeling.

References

- 1 Chunki Park, Will Bridewell, Pat Langley: *Integrated Systems for Inducing Spatio-Temporal Process Models*. AAAI 2010
- 2 Will Bridewell, Ljupco Todorovski: *The Induction and Transfer of Declarative Bias*. AAAI 2010
- 3 Will Bridewell, Pat Langley, Ljupco Todorovski, Saso Dzeroski: *Inductive process modeling*. Machine Learning 71(1): 1-32 (2008)

3.11 Search and Black-Box Constraint-Programming Solvers

Laurent Michel (University of Connecticut – Storrs, US)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Laurent Michel

Joint work of Michel, Laurent; Van Hentenryck, Pascal

Main reference P. Van Hentenryck, L. Michel. *Constraint-Based Local Search*. MIT Press, Cambridge, London, 2009.

URL <http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=11963>


Constraint Programming Tools traditionally rely on the combination of a modeling and a search component to solve challenging combinatorial optimization problems. Advances in modeling are often aimed at delivering stronger propagation and filtering algorithms while progress on the search components focus on declarative control abstractions that deliver clear, modular and reusable search procedures.

The purpose of this talk is to review the state of the art and recent advances in search. In particular, it illustrates how control abstractions like non-deterministic tryall, selectors, explorers, controllers and labeling heuristics found in the COMET language are instrumental in delivering simple, elegant and fully reusable search procedures.

The talk reviews how such abstractions can be used to build classic generic search procedures found in black-box constraint-programming solvers. It discusses in details a recent addition: activity-based search, the idea of using the activity of variables during propagation to guide the search. ABS is compared experimentally to impact-based search and the WDeg heuristics on a variety of benchmarks to illustrate its robustness and performance.

3.12 MIME: Discovering Interesting Patterns Visually and Interactively

Sandy Moens (University of Antwerp, BE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Sandy Moens

Joint work of Goethals, Bart; Moens, Sandy; Vreeken, Jilles

Main reference Goethals, B., Moens, S., and Vreeken, J., “MIME: A Framework for Interactive Visual Pattern Mining,” In Proc. 17th ACM SIGKDD Int’l Conf. on Knowledge Discovery and Data Mining (SIGKDD’11), pp. 757–760, 2011

URL <http://doi.acm.org/10.1145/2020408.2020529>

Pattern mining is concerned with discovering interesting patterns from data. Formalizing interestingness, however, is notoriously difficult, as it is inherently subjective. We propose to discover subjectively interesting patterns. In our system, users can visually browse

through data and patterns. To assist users in identifying interesting patterns, a toolbox of interestingness measures and algorithms for mining and post-processing are provided. All statistics can be combined in interconnected views, and results of actions in one view can be directly used in others. All information is computed on-the-fly, where priority is given to what is currently shown to the user, while off-screen results are calculated in the background.

By making the process interactive, we enable the user to combine their subjective notion of interestingness, and background knowledge, with a wide variety of objective measures in order to easily mine the most interesting patterns. Basically, we enable the user to become an essential part of the mining algorithm, by allowing the construction of patterns, deciding what algorithms to run and how to analyze what patterns in detail.

References

- 1 Goethals, B., Moens, S. and Vreeken, J. *MIME: A Framework for Interactive Visual Pattern Mining*. In Proc. KDD, ACM, 2011.
- 2 Goethals, B., Moens, S. and Vreeken, J. *MIME: A Framework for Interactive Visual Pattern Mining*. In Proc. ECML PKDD, Springer, 2011.

3.13 Constraint programming for Itemset Mining

Siegfried Nijssen (K.U. Leuven, BE)

License © ⓘ ⊖ Creative Commons BY-NC-ND 3.0 Unported license
© Siegfried Nijssen

Joint work of Nijssen, Siegfried; Guns, Tias; De Raedt, Luc

Main reference Tias Guns, Siegfried Nijssen, Luc De Raedt, “Itemset mining: A constraint programming perspective,” *Artif. Intell.*, 175(12–13): 1951–1983 (2011)

URL <http://dx.doi.org/10.1016/j.artint.2011.05.002>

The field of data mining has become accustomed to specifying constraints on patterns of interest. A large number of systems and techniques has been developed for solving such constraint-based mining problems, especially for mining itemsets. The approach taken in the field of data mining contrasts with the constraint programming principles developed within the artificial intelligence community. While most data mining research focuses on algorithmic issues and aims at developing highly optimized and scalable implementations that are tailored towards specific tasks, constraint programming employs a more declarative approach. The emphasis lies on developing high-level modeling languages and general solvers that specify what the problem is, rather than outlining how a solution should be computed, yet are powerful enough to be used across a wide variety of applications and application domains.

Here we present a declarative constraint programming approach to data mining.

More specifically, we show that it is possible to employ off-the-shelf constraint programming techniques for modeling and solving a wide variety of constraint-based itemset mining tasks, such as frequent, closed, discriminative, and cost-based itemset mining. In particular, we develop a basic constraint programming model for specifying frequent itemsets and show that this model can easily be extended to realize the other settings. This contrasts with typical procedural data mining systems where the underlying procedures need to be modified in order to accommodate new types of constraint, or novel combinations thereof. Even though the performance of state-of-the-art data mining systems outperforms that of the out-of-the-box constraint programming approach on some standard tasks, we show that by developing specialized CP systems, it is possible to overcome most of the differences in efficiency. Furthermore, we show that there exist problems where the out-of-the-box


constraint programming approach already leads to significant performance improvements over state-of-the-art methods in data mining and leads to new insights into the underlying data mining problems. Many such insights can be obtained by relating the underlying search algorithms of data mining and constraint programming systems to one another. We provide an illustration of the approach on a problem in bioinformatics and finally discuss a number of interesting new research questions and challenges raised by the declarative constraint programming approach to data mining.

References

- 1 Luc De Raedt, Tias Guns, Siegfried Nijssen: Constraint programming for itemset mining. KDD 2008: 204-212
- 2 Tias Guns, Hong Sun, Kathleen Marchal, Siegfried Nijssen: Cis-regulatory module detection using constraint programming. BIBM 2010: 363-368
- 3 Tias Guns, Siegfried Nijssen, Luc De Raedt: Itemset mining: A constraint programming perspective. Artif. Intell. 175(12-13): 1951-1983 (2011)
- 4 Siegfried Nijssen, Tias Guns: Integrating Constraint Programming and Itemset Mining. ECML/PKDD (2) 2010: 467-482
- 5 Siegfried Nijssen, Tias Guns, Luc De Raedt: Correlated itemset mining in ROC space: a constraint programming approach. KDD 2009: 647-656

3.14 Integrations of Machine Learning and Data Mining in Constraint Satisfaction

Barry O’Sullivan (University College Cork, IE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Barry O’Sullivan

Main reference Barry O’Sullivan, “Automated Modelling and Solving in Constraint Programming,” Proc. 24th AAAI Conf. on Artificial Intelligence (AAAI’10), pp. 1493-1497, 2010.

URL <http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1899>

Constraint programming can be divided very crudely into modeling and solving. Modeling defines the problem, in terms of variables that can take on different values, subject to restrictions (constraints) on which combinations of variables are allowed. Solving finds values for all the variables that simultaneously satisfy all the constraints. However, the impact of constraint programming has been constrained by a lack of “user-friendliness”. Constraint programming has a major “declarative” aspect, in that a problem model can be handed off for solution to a variety of standard solving methods. These methods are embedded in algorithms, libraries, or specialized constraint programming languages. To fully exploit this declarative opportunity however, we must provide more assistance and automation in the modeling process, as well as in the design of application-specific problem solvers. Automated modelling and solving in constraint programming presents a major challenge for the artificial intelligence community. Artificial intelligence, and in particular machine learning, is a natural field in which to explore opportunities for moving more of the burden of constraint programming from the user to the machine. This talk presents technical challenges in the areas of constraint model acquisition, formulation and reformulation, synthesis of filtering algorithms for global constraints, and automated solving. We also present the metrics by which success and progress can be measured.

References

- 1 Barry O’Sullivan: Automated Modelling and Solving in Constraint Programming. AAAI 2010
- 2 Hadrien Cambazard, Tarik Hadzic, Barry O’Sullivan: Knowledge Compilation for Itemset Mining. ECAI 2010: 1109-1110
- 3 Christian Bessière, Emmanuel Hebrard, Barry O’Sullivan: Minimising Decision Tree Size as Combinatorial Optimisation. CP 2009: 173-187
- 4 Xuan-Ha Vu, Barry O’Sullivan: A Unifying Framework for Generalized Constraint Acquisition. International Journal on Artificial Intelligence Tools 17(5): 803-833 (2008)
- 5 Christian Bessière, Remi Coletta, Frédéric Koriche, Barry O’Sullivan: A SAT-Based Version Space Algorithm for Acquiring Constraint Satisfaction Problems. ECML 2005: 23-34

3.15 Constrained Conditional Models: Integer Linear Programming Formulations for Natural Language Understanding

Dan Roth (University of Illinois at Urbana-Champaign, USA)

License © ⓘ ⊖ Creative Commons BY-NC-ND 3.0 Unported license
© Dan Roth

URL www.aaai.org/Papers/AAAI/2008/AAAI08-252

Main reference M. Chang, L. Ratinov, N. Rizzolo and D. Roth, “Learning and Inference with Constraints,” Proc. of the National Conference on Artificial Intelligence (AAAI), 2008.


Intelligent Information Access and Extraction suggest significant challenges for Natural Language analysis. Tasks of interest include semantic role labeling (determining who did what to whom, when and where), information extraction (identifying events, entities and relations), transliteration of names, and textual entailment (determining whether one utterance is a likely consequence of another). A computational approach to these challenges often involves assigning values to sets of interdependent variables and thus frequently necessitate performing global inference that accounts for these interdependencies. This talk presented research on Constrained Conditional Models (CCMs), a framework that augments probabilistic models with declarative constraints as a way to support such decisions. We presented a framework we introduced a few years ago, formulating decision problems in NLP as Integer Linear Programming problems, but focused on new algorithms for training these global models using indirect supervision signals. Learning models for structured tasks is difficult partly since generating supervision signals is costly. We showed that it is often easy to obtain a related indirect supervision signal, and discussed several options for deriving this supervision signal, including inducing it from the world’s response to the model’s actions. Our learning framework is “Constraints Driven” in the sense that it allows and even gains from global inference that combines statistical models with expressive declarative knowledge (encoded as constraints), modeled via Constrained Conditional Models. Experimental results showed the significant contribution of easy-to-get indirect supervision on several NLP tasks including information extraction, Transliteration and Textual Entailment.

See also the tutorial from NAACL’10:

<http://l2r.cs.uiuc.edu/%7Edanr/Talks/ILP-CCM-Tutorial-NAACL10.ppt>.

3.16 Gecode – an open constraint solving library

Christian Schulte (KTH – Kista, SE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Christian Schulte

Joint work of Gecode team


URL <http://www.gecode.org>

Gecode is a widely used toolkit for developing constraint-based systems and applications. Gecode provides a constraint solver with state-of-the-art performance while being modular and extensible. Gecode is: open (documented interfaces support tasks from modeling to implementing new variables, constraints, search engines, ...), free (MIT license), portable (standard C++), accessible (extensive tutorial and reference documentation, efficient (excellent performance, has won the 2008-2010 MiniZinc challenges in all categories), and parallel (exploits today's multi-core hardware).

The talk provides an overview of what Gecode is, what one can do with it, and what are the basic ideas behind its architecture.

3.17 Lifelong learning in Constraint Solving

Michele Sebag (LIMSI – CNRS, FR)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Michele Sebag

Joint work of Arbelaez, Alejandro; Hamadi, Youssef; Sebag, Michele

Main reference Alejandro Arbelaez, Youssef Hamadi, Michèle Sebag, “Continuous Search in Constraint Programming,” Proc. 22nd IEEE Int'l Conf. on Tools with Artificial Intelligence (ICTAI'10), pp. 53–60, 2010.

URL <http://dx.doi.org/10.1109/ICTAI.2010.17>

The goal is to allow any user to eventually get top performance from his constraint solver. This goal is reached by customizing the solver to the user's problem instance distribution, using the computer idle time to launch exploratory experiments on the user's problem, observing the results and learning the most appropriate tunings of the heuristic portfolio.

This approach can be viewed as an instance of Meta-Learning problem, with the difference that many descriptive features have been proposed in the CP literature to characterize the problem to solve and the current search state [4]. The novelty compared to the state of the art (e.g.

CPHydra, [5]) is that the computer does not require a set of problems, representative of the user's activity, to be available beforehand.

Instead, the computer uses an exploration/exploitation approach (lifelong learning) to gradually become an expert into the user's problem distribution.

Experimental validation suggests that Continuous Search can design efficient mixed strategies after considering a moderate number of problem instances.

References

- 1 Alejandro Arbelaez. *Learning During Search*. PhD thesis, Université Paris-Sud, France, 2011.
- 2 Alejandro Arbelaez, Youssef Hamadi, Michèle Sebag: *Continuous Search in Constraint Programming*. ICTAI (1) 2010: 53–60.

- 3 Alejandro Arbelaez and Youssef Hamadi and Michele Sebag: *Autonomous Search*. In: Youssef Hamadi and Eric Monfroy and Frederic Saubion (editors): *Continuous Search in Constraint Programming*, Springer, 2011.
- 4 Frank Hutter, Holger H. Hoos, Thomas Stützle: *Automatic Algorithm Configuration Based on Local Search*. *AAAI 2007*: 1152–1157.
- 5 Eoin O’Mahony, Emmanuel Hebrard, Alan Holland, Conor Nugent and Barry O’Sullivan: *Using Case-based Reasoning in an Algorithm Portfolio for Constraint Solving*. *Proceedings of the 19th Irish Conference on Artificial Intelligence and Cognitive Science*, 2008.

3.18 A Constraint Seeker: Finding and Ranking Global Constraints from Examples

Helmut Simonis (University College Cork, IE)

License © ⓘ ⊖ Creative Commons BY-NC-ND 3.0 Unported license
© Helmut Simonis

Joint work of Beldiceanu, Nicolas; Simonis, Helmut

Main reference N. Beldiceanu and H. Simonis, “A Constraint Seeker: Finding and Ranking Global Constraints from Examples,” In *Principles and Practice of Constraint Programming (CP2011)*, Perugia, Italy, September 2011.

The global constraint catalog provides a valuable repository of global constraint information for both researchers in the constraint field and application developers searching for the right modelling abstraction. The catalog currently describes over 350 constraints on more than 2800 pages. This wealth of information is also its main weakness.

For a novice (and even for an experienced) user it can be quite challenging to find a particular constraint, unless the name in the catalog is known. As a consistent naming scheme (like the naming scheme for chemical compounds http://en.wikipedia.org/wiki/IUPAC_nomenclature, for example) does not exist in the constraint field, and different constraint systems often use incompatible names and argument orders for their constraints, there is no easy way to deduce the name of a constraint and the way its arguments are organized from its properties. The catalog already provides search by name, or by keywords, and provides extensive cross-references between constraints, as well as a classification by the required argument type. All of these access methods can be helpful, but are of limited use if one does not know too much about the constraint one is looking for.


The Constraint Seeker (<http://seeker.mines-nantes.fr>) provides another way of finding constraint candidates, sorted by potential relevance, in the catalog.

The user provides positive and/or negative ground instances for a constraint, and the tool provides a ranked list of possible candidates which match the given examples.

Besides introducing the Constraint Seeker as an application, we want to illustrate the power of meta data and meta programming in the context of future constraints platforms. We use meta data for explicitly describing different aspects of global constraints such as argument types, restrictions on the arguments, typical use of a constraint, symmetries w.r.t. the arguments of a constraint, links between constraints (e.g. implication, generalisation). The electronic version of the global constraint catalog provides such information in a systematic way for the different global constraints. The Constraint Seeker illustrates the fact that the same meta data can be used for different purposes, unlike ad-hoc code in a procedural language which is designed for a specific (and unique) usage and a single system.

3.19 AOPCALEAPS: Automatic Pop Composer And Learner of Parameters

Jon Sneyers (K.U. Leuven, BE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Jon Sneyers

Joint work of Sneyers, Jon; De Schreye, Danny

Main reference Jon Sneyers and Danny De Schreye, “AOPCALEAPS: Automatic Music Generation with CHRiSM,” 22nd Benelux Conf. on Artificial Intelligence (BNAIC’10), Luxembourg, October 2010.

URL http://people.cs.kuleuven.be/~jon.sneyers/papers/chris_m_music.pdf

AOPCALEAPS is a system for automatic pop music generation and learning, implemented in CHRiSM [1]. CHRiSM is a new programming language, combining features of the existing languages CHR [2] and PRISM [3]. It is a high-level rule-based formalism for probabilistic-logic learning, allowing a powerful mix of domain-specific constraints and probabilistic inference. The current implementation of CHRiSM is based on the K.U.Leuven CHR system in B-Prolog. The user interface of AOPCALEAPS was made using ftk; its output is rendered using LilyPond.

The goal of the AOPCALEAPS project is to create a personal automatic music generator. The user creates music, selects the examples that are good according to his or her taste, and then these selected examples are used as a training set for learning. After several iterations, the result is a music generator that produces personalized music.

References

- 1 Jon Sneyers, Wannes Meert, Joost Vennekens, Yoshitaka Kameya and Taisuke Sato. *CHR(PRISM)-based Probabilistic Logic Learning*. 26th International Conference on Logic Programming (ICLP’10), Edinburgh, UK, July 2010.
- 2 Jon Sneyers, Peter Van Weert, Tom Schrijvers, and Leslie De Koninck. *As Time Goes By: Constraint Handling Rules – A Survey of CHR Research between 1998 and 2007*. *Theory and Practice of Logic Programming*, 10(1):1-47, 2010.
- 3 Taisuke Sato. *A glimpse of symbolic-statistical modeling by PRISM*. *Journal of Intelligent Information Systems*, Vol.31, No.2, pp.161-176, 2008.

3.20 MiniZinc – Towards a standard CP modelling language

Guido Tack (K.U. Leuven, BE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Guido Tack

Joint work of Nethercote, Nicholas; Stuckey, Peter J.; Becket, Ralph; Brand, Sebastian; Duck, Gregory J.; Tack, Guido

Main reference N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, and G. Tack, “MiniZinc: Towards a standard CP modelling language,” . *Proc. 13th Int’l Conf. on Principles and Practice of Constraint Programming (CP’07)*, pp. 529–543, 2007.


URL http://dx.doi.org/10.1007/978-3-540-74970-7_38

MiniZinc arose as a response to the extended discussion at CP’06 of the need for a standard modelling language for CP. Since 2006, MiniZinc and its ecosystem have grown and matured considerably. Today, in addition to several CP solvers that handle MiniZinc (through the FlatZinc solver interface language), there is support for SMT, SAT, and MIP solvers as well. This makes MiniZinc an ideal candidate language for experimentation and model exchange.

The talk consists of an introduction to the MiniZinc language using example models, a quick overview of the compilation and execution approach, and the current plans for MiniZinc’s future.

3.21 On Learning Constraint Problems

Christel Vrain (Université d’Orleans, FR)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Christel Vrain

Joint work of Lallouet, Arnaud; Lopez, Matthieu; Martin, Lionel; Vrain, Christel

Main reference Arnaud Lallouet, Matthieu Lopez, Lionel Martin, Christel Vrain, “On Learning Constraint Problems,” Proc. 22nd IEEE Int’l Conf. on Tools with Artificial Intelligence (ICTAI’10), pp. 45–52, 2010.

URL <http://dx.doi.org/10.1109/ICTAI.2010.16>

We address the problem of learning constraint models expressed in a higher level language. For learning it, we use Inductive Logic Programming and we develop a new search strategy to tackle the problem.

We start from the observation that some novice users have difficulties to model with constraints and it would be desirable to provide them tools that help them to come with a constraint model by using only the informal knowledge they have on their problem.

Often users can provide examples and counter-examples for the problem to solve or answer questions about it. But often the problems they tackled in the past were not exactly the same but only similar. For example, it may happen that the user has solved the 4-queens and 5-queens problem but wants to solve n-queens.

How to reuse this knowledge to provide a general model?

In this work, we learn a high level model in a first-order logic language retaining some of the features of middle-level constraint modeling languages like Minizinc. The user provides some historical data and the model is learned by ILP techniques. We found that existing ILP learners were subject to blind search and we had to provide a new bidirectional learning algorithm to find solutions.

Then the model is transformed into a CSP through a rewriting phase which also takes as input the actual variables of the new problem to solve. Experiments have been conducted on graph coloring, timetable and jobshop scheduling and n-queens.


References

- 1 Arnaud Lallouet, Matthieu Lopez, Lionel Martin, Christel Vrain: On Learning Constraint Problems. ICTAI (1) 2010: 45-52
- 2 Matthieu Lopez, Lionel Martin, Christel Vrain: Learning Discriminant Rules as a Minimal Saturation Search. ILP 2010: 146-157

4 Working Groups

4.1 Declarative Data Mining

Siegfried Nijssen (K.U. Leuven, BE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Siegfried Nijssen

The aim of this working group was to consider how declarative approaches to problem solving, such as constraint programming, may be used to improve the principles and practice of data mining. The discussion started by compiling a list of questions:

- What is a declarative approach to data mining?
- What is currently missing in declarative solvers if we wish to use them in data mining?

- Which data mining problems are good candidates for a declarative approach to data mining?
- Is there a catalog of data mining problems that could give constraint programmers a better idea what the challenges are?
- What are expected benefits of a declarative approach data mining approach?

Subsequently, these questions were addressed in the further discussion.

Improved flexibility and easiness were identified as main reasons for declarative data mining. Declarative data mining should allow to implement new data mining tasks with only a small number of lines of code, hence making the implementation of new data mining tasks easier. Furthermore, declarative data mining system should ideally be compositional. This would make it possible to combine data mining tasks in novel ways, making data mining more flexible.


A declarative data mining language would be a language that allows to express *what* a data mining task is, and would decouple this from *how* the data mining task is solved. Hence, it was argued that such a language could be solver independent, where the Zinc language was mentioned as an example from the constraint programming community. A solver would ideally process a model in the language automatically, but the model may also be used a starting point for manual tuning.

A topic that attracted extensive discussion was how easy would be to make such an approach sufficiently efficient. It was argued that declarative approaches have been successful in the database community as any statement written down in a language such as (traditional) SQL will always be executed in polynomial time. This is not the case for statements written in a constraint programming language. It was argued that a library should be constructed that makes clear the complexity of the primitives in a language, such that users have a clear idea how to model problems efficiently. Convex optimization in machine learning was used as an example: certain mathematical operations are known to be more efficient than others, and hence machine learning researchers focus on using these.

The working group was concluded with a discussion on problems involving probabilistic and/or soft constraints. Many probabilistic inference problems in machine learning are hard to solve exactly and would require inexact solvers or solvers based on sampling. From a constraint programming perspective addressing such problems is still a challenge.

4.2 Learning Constraint Satisfaction Problems

Luc De Raedt (K.U. Leuven, BE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Luc De Raedt

This working group discussed how machine learning techniques could help to improve constraint programming. A constraint satisfaction problem is usually defined as a triplet (V, D, C) , with V the variables, D the domain of the variables and C the constraints. In addition, there can be a preference function f that specifies which solutions are to be preferred.


The working group concluded that there were at least two possible roles for machine learning. The first role is for learning the model of the problem, that is, the triple (V, D, C) and possibly the preference function f from data. The most direct task is that of learning the constraints C from data, that is, from possible assignments to the variables V that

satisfy the constraints. This is essentially a task that is akin to that of concept-learning. Learning constraints has been addressed within the database and inductive logic programming communities. Several variations on the task can be imagined: learning from positives only, learning from positive as well as negative examples, active learning (where the learner may ask questions), and learning *soft constraints*. Learning soft constraints would typically involve the learning of the weights or parameters of the constraints, though also the structure might be learned. This setting is akin to that of learning probabilistic models. As an alternative to learning soft constraints, one might learn the preference function f from examples of the form assignment x is better than assignment y . This is known in the machine learning literature as preference learning.

The second role for machine learning is to learn how to improve the performance of the solvers using experience, that is, speed-up learning. In this case it is usually assumed that the model (V, D, C) is given and passed runs of the solver are used to learn additional constraints or heuristics. This role for machine learning was extensively discussed in the talk by Michele Sebag. One can basically use deductive as well as inductive techniques for speed-up learning. Inductive techniques would gather data from previous test-runs and learn constraints or heuristics in a similar way as above, the key difference being that the learned hypotheses have to be tested against the model rather than against the data. On the other hand, deductive techniques would employ methods of explanation based learning to analyse traces and proofs and deductively obtain constraints that are implied by the given model.

4.3 Learning in Constraint Solvers

Barry O’Sullivan (University College Cork, IE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Barry O’Sullivan

This working group discussed how machine learning techniques could be used to improve the constraint solving process and, in particular, how machine learning techniques could be employed in the design and development of constraint solvers. The group discussed the opportunities for applying machine learning at a variety of levels, such as:

- solving specific problem instances, e.g. how can machine learning be used to solve a specific instance of a class of CSPs.
- designing portfolios of constraint solvers, building on existing approaches such as SATzilla (runtime prediction), CPhydra (case-based reasoning), ACME (solver selection as classification).
- lifelong learning, e.g. how can constraint solving experience be generalised and applied in an automated way?
- using learning to exploit problem structure, e.g. neighborhood identification for large-neighbourhood search methods.
- solver configuration, e.g. inspired by the MULTITAC system which parameterizes constraints solvers in terms of search heuristics, propagators, etc. Which technology should be used and when?

The key challenges for applying learning to solver design/development are: to identify the appropriate learning task that defines an appropriate success metric; to define an appropriate set of features to represent the training data; to select an appropriate distribution of scenarios/problem instances to use as training data; and, to select an appropriate learning

method. As evidenced in these working notes, there has been some progress in these areas. However, many rather mundane, but complex, challenges remain, e.g. on what basis should we claim that one solving procedure is better than another? Clearly solving time is important, but so is robustness if that solver must solve many instances within a problem class, or across several classes.

Many interesting opportunities for classification were discussed. For example, one can frame a classification task around the identification of “good” vs “bad” search trees to inform a randomized search algorithm which uses restarts. If a search tree is considered “bad” the algorithm should restart. A related discussion considered the opportunities for reinforcement learning to react to the context defined at a node during search. For example, should the problem be decomposed, should we propagate the state? These are all complex challenges which experienced constraint programmers deal with, and which could be aided by the appropriate use of machine learning.

Many researchers were interested in the use of learning for designing new search heuristics. There are many opportunities to gather statistics during search which can be used as training data for credit assignment methods and rule learners (both symbolic and numeric) upon which new heuristics can be learned.

A key concern for constraint programmers is to learn how to store and add constraints from dead-ends, often referred to as nogood learning. Challenges here relate to the identification, generalization, and forgetting of nogoods.

Finally, there was a long discussion about the use of learning for problem decomposition. An interesting question is how to generalize branching on assignments to branching on constraints. Machine learning can help us learn from past experience about the utility of solving particular problems using stream-lining constraints whereby a decision is made to impose a constraint that solutions should have specific structure. Similarly, machine learning can help learn grammars for composing interesting constraints.

In conclusion there was agreement that research at the interface between machine learning and constraint solving offered considerable promise and had the potential to yield many high-impact and valuable techniques.

5 Acknowledgements

We would like to thank the staff of Schloss Dagstuhl for their excellent help in organising this seminar.

Participants

- Rolf Backofen
Universität Freiburg, DE
- Roberto Bayardo
Google Inc. –
Mountain View, US
- Michele Berlingerio
CNR – Pisa, IT
- Hendrik Blockeel
K.U. Leuven, BE
- Jean-François Boulicaut
INSA – Lyon, FR
- Maurice Bruynooghe
K.U. Leuven, BE
- Toon Calders
TU Eindhoven, NL
- Bruno Crémilleux
Université de Caen, FR
- James Cussens
University of York, GB
- Ian Davidson
Univ. of California – Davis, US
- Tijl De Bie
University of Bristol, GB
- Luc De Raedt
K.U. Leuven, BE
- Yves Deville
UC Louvain-la-Neuve, BE
- Pierre Dupont
UC Louvain-la-Neuve, BE
- Pierre Flener
Uppsala University, SE
- Paolo Frasconi
University of Firenze, IT
- Alan Frisch
University of York, GB
- Carmen Gervet
German University in Cairo, EG
- Bart Goethals
University of Antwerp, BE
- Tias Guns
K.U. Leuven, BE
- Georgiana Ifrim
University College Cork, IE
- Kristian Kersting
Fraunhofer IAIS –
St. Augustin, DE
- Arnaud Lallouet
GREYC – Caen, FR
- Pat Langley
ASU – Tempe, US
- Laurent Michel
University of Connecticut –
Storrs, US
- Sandy Moens
University of Antwerp, BE
- Barry O'Sullivan
University College Cork, IE
- Dan Roth
Univ. of Illinois – Urbana, US
- Salvatore Ruggieri
University of Pisa, IT
- Christian Schulte
KTH – Kista, SE
- Michele Sebag
LIMSI – CNRS, FR
- Arno Siebes
Utrecht University, NL
- Helmut Simonis
University College Cork, IE
- Jon Sneyers
K.U. Leuven, BE
- Guido Tack
K.U. Leuven, BE
- Pascal Van Hentenryck
Brown Univ. – Providence, US
- Peter Van Roy
UC Louvain-la-Neuve, BE
- Christel Vrain
Université d'Orleans, FR
- Toby Walsh
The University of New South
Wales, Australia, AU

