Report from Dagstuhl Seminar 18071

Planning and Operations Research

Edited by

J. Christopher Beck[1], Daniele Magazzeni[2], Gabriele Röger[3], and Willem-Jan Van Hoeve[4]

1 **University of Toronto, CA,** jcb@mie.utoronto.ca
2 **King's College London, GB,** daniele.magazzeni@kcl.ac.uk
3 **Universität Basel, CH,** gabriele.roeger@unibas.ch
4 **Carnegie Mellon University – Pittsburgh, US,** vanhoeve@andrew.cmu.edu

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 18071 "Planning and Operations Research". The seminar brought together researchers in the areas of Artificial Intelligence (AI) Planning, Constraint Programming, and Operations Research. All three areas have in common that they deal with complex systems where a huge space of interacting options makes it almost impossible to humans to take optimal or even good decisions. From a historical perspective, operations research stems from the application of mathematical methods to (mostly) industrial applications while planning and constraint programming emerged as subfields of artificial intelligence where the emphasis was traditionally more on symbolic and logical search techniques for the intelligent selection and sequencing of actions to achieve a set of goals. Therefore operations research often focuses on the allocation of scarce resources such as transportation capacity, machine availability, production materials, or money, while planning focuses on the right choice of actions from a large space of possibilities. While this difference results in problems in different complexity classes, it is often possible to cast the same problem as an OR, CP, or planning problem. In this seminar, we investigated the commonalities and the overlap between the different areas to learn from each other's expertise, bring the communities closer together, and transfer knowledge about solution techniques that can be applied in all areas.

Seminar February 11–16, 2018 – https://www.dagstuhl.de/18071
2012 ACM Subject Classification Applied computing → Operations research, Computing methodologies → Planning and scheduling, Theory of computation → Constraint and logic programming
Keywords and phrases Artificial Intelligence, Automated Planning and Scheduling, Constraint Programming, Dynamic Programming, Heuristic Search, Mixed Integer Programming, Operations Research, Optimization, Real-world Applications, Reasoning under Uncertainty
Digital Object Identifier 10.4230/DagRep.8.2.26
Edited in cooperation with Florian Pommerening

1 Executive Summary

Florian Pommerening (Universität Basel, CH)

This seminar brought together leading experts in the fields of AI planning, constraint programming and operations research. These areas historically come from different roots but are all concerned with supporting decision making in complex systems which a huge space of interacting options. While the approach and focus is different, some concepts have been developed in multiple areas and some solution techniques are or could be transfered. There is also a growing intersection of the areas that considers hybrid problems or uses solvers developed in one area to solve problems from a different area, for example by compiling planning problems into MIP or by using CP for subproblems in a MIP solver. Solvers of a different community are often used as black boxes and the deeper understanding of each other's area of expertise that was developed in this seminar will help to foster collaboration and transfer knowledge between the areas.

The seminar started with eleven short but intense tutorials on Monday and Tuesday morning. The tutorials on Monday conveyed the basics of AI Planning, MIP, and CP. They also already introduced the main connections between the fields by talking about compilations from planning to CP and MIP and using LPs as heuristics in planning. The tutorials on Tuesday delved deeper into areas that became the focus of discussion later in the seminar, such as non-deterministic planning, Markov decision processes, and decision diagrams. Front-loading these tutorials worked well to bring everyone up to speed and created a good basis for the rest of the seminar.

The rest of the seminar was organized into working groups that included one to three short presentations followed by a longer discussion all focused on a central topic. Three of these sessions were organized as break-out sessions where the participants split into two groups, each discussion one topic and then reconvening to present the main points discussed in each group to each other. The schedule for each day was created on the evening before which kept the topics flexible and allowed the organizers to include topics that came up during the discussion. Notes on each of the working groups and abstracts of the tutorials are included in the rest of this report.

2 Table of Contents

Executive Summary
Florian Pommerening . . . 26

Overview of Tutorials

Introduction to Planning
Malte Helmert . . . 30

Introduction to MIP
Thorsten Koch . . . 30

Compiling Planning to Mixed Integer Linear Programming
Chiara Piacentini . . . 31

LP for Classical Planning Heuristics
Florian Pommerening . . . 31

Heuristics for Numeric Planning
Patrik Haslum . . . 32

Introduction to Constraint Programming
Pierre Schaus . . . 33

Compiling Planning to Constraint Programming
Roman Bartak . . . 33

Introduction to Non-deterministic Planning
Christian Muise . . . 34

Markov Decision Processes in Planning
Matthijs Spaan . . . 34

Decision Diagrams for Planning and Optimization
Scott Sanner . . . 35

Decision Diagrams for Optimization
Willem-Jan Van Hoeve . . . 35

Working groups

MINLP and Nonlinearities in Planning
J. Christopher Beck . . . 36

Benchmarking
Christina N. Burt . . . 37

Merge&Shrink and Decision Diagrams
Christina N. Burt and Florian Pommerening . . . 38

Teaching
Hadrien Cambazard . . . 42

Hybrids & Decomposition
Mathijs de Weerdt . . . 43

Non-traditional Objective Functions for MDPs
Patrik Haslum . . . 45

Open and Challenging Problems
Patrik Haslum 46

Explaining Solutions and Solution Methods
Christian Muise 48

Modeling
Christian Muise 49

Cross-Domain Example Problems
Florian Pommerening 50

Dealing with Uncertainty
Florian Pommerening 52

Connections Between our Fields
Mark Roberts 53

LP Heuristics: The Details
Gabriele Röger 55

Planning, Optimization, and Machine Learning
Scott Sanner 56

Multiagent Planning and Optimization
Matthijs Spaan 58

Exploiting Substructures
Charlotte Truchet 61

Participants 63

3 Overview of Tutorials

3.1 Introduction to Planning

Malte Helmert (Universität Basel, CH)

The talk gave a brief introduction to domain-independent automated planning. It introduced the classical (finite-domain, grounded, sequential) planning problem and generalizations to numerical state variables and to temporal planning. The talk discussed common representations used for classical planning, including finite-domain (SAS$^+$) representation, transition normal form (TNF), STRIPS, and ADL. It also pointed out the connection between classical planning and automata theory, where individual state variables can be interpreted as finite automata, and the overall planning problem corresponds to language intersection/product automata. Finally, the talk discussed the planning research community in order to help locate relevant research (published primarily at the ICAPS, IJCAI, AAAI and ECAI conferences and in the JAIR and AIJ journals), to understand benchmarking practices using the PDDL representation language and the benchmark collections from the International Planning Competitions (IPC), and to understand the historical emphasis of planning research of domain-independent algorithms and "neutral" models ("Physics, not advice").

3.2 Introduction to MIP

Thorsten Koch (Konrad-Zuse-Zentrum – Berlin, DE)

Joint work of The Mathematical Optimization Methods group at ZIB
Main reference Ambros Gleixner, Leon Eifler, Tristan Gally, Gerald Gamrath, Patrick Gemander, Robert Lion Gottwald, Gregor Hendel, Christopher Hojny, Thorsten Koch, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Jan Merlin Viernickel, Stefan Vigerske, Dieter Weninger, Jonas T. Witt, and Jakob Witzig: "The SCIP Optimization Suite 5.0". ZIB-Report, pp. 17–61, ISSN: 1438-0064, Zuse Institute Berlin, 2017.
URL http://nbn-resolving.de/urn:nbn:de:0297-zib-66297

The presentation gave an introduction to computational Mixed Integer Programming (MIP) and the algorithms behind it, in particular the Branch-and-Cut paradigm. This included notes about parallel implementations and extention to non-convex Mixed Integer Non-Linear Programming (MINLP).

3.3 Compiling Planning to Mixed Integer Linear Programming

Chiara Piacentini (University of Toronto, CA)

Joint work of Chiara Piacentini, Margarita P. Castro, André A. Ciré, J. Christopher Beck
Main reference Chiara Piacentini, Margarita P. Castro, André A. Ciré, J. Christopher Beck: "Compiling Optimal Numeric Planning to Mixed Integer Linear Programming", in Proc. of the Twenty-Eighth International Conference on Automated Planning and Scheduling, ICAPS 2018, Delft, The Netherlands, June 24-29, 2018., pp. 383–387, AAAI Press, 2018.
URL https://aaai.org/ocs/index.php/ICAPS/ICAPS18/paper/view/17770

Compilation techniques in planning reformulate a problem into an alternative encoding for which efficient, off-the-shelf solvers are available. In this tutorial, we presented mixed-integer linear programming (MILP) compilations for cost-optimal planning with instantaneous actions. We presented three encodings of classical planning taken from the literature and we show how they can be extended to handle problems with numeric state variables, numeric linear conditions, and numeric linear action effects.

References

1 Thomas Vossen, Michael O. Ball, Amnon Lotem, and Dana S. Nau. *On the Use of Integer Programming Models in AI Planning.* In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, pp. 304–309, 1999.
2 Menkes van den Briel, Thomas Vossen, and Subbarao Kambhampati. *Reviving Integer Programming Approaches for AI Planning: A Branch-and-Cut Framework.* In Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling, pp. 310–319, 2005.

3.4 LP for Classical Planning Heuristics

Florian Pommerening (Universität Basel, CH)

Main reference Florian Pommerening: "New Perspectives on Cost Partitioning for Optimal Classical Planning". Dissertation, University of Basel, Switzerland, 2017.

Linear Programs (LPs) are used in classical planning in different ways. This talk focused around their uses while computing admissible *heuristic functions*, which compute lower bounds during a search for an optimal plan. The use of LPs in such cases can be grouped into three classes: *computation*, *combination*, and *synthesis*.

In heuristic computation, heuristic values of heuristic functions like the state-equation heuristic [1] or the delete-relaxation heuristic [2] are computed as the objective value of an LP.

Heuristic combination is specifically interesting for admissible heuristics where they solve the problem of combining several lower bounds into a single lower bound while minimizing the loss of information from all bounds. Cost partitioning [3], posthoc optimization [4], and operator counting [5] are examples of this.

Finally, LPs can be used to synthesize a whole heuristic function. For example, potential heuristics [5] use linear constraints that characterize desired heuristic properties and optimize a measure of quality to select the best potential heuristic with the desired properties.

References

1 Menkes van den Briel, J Benton, Subbarao Kambhampati, and Thomas Vossen. *An LP-based heuristic for optimal planning.* In Proceedings of the Thirteenth International Conference on Principles and Practice of Constraint Programming, pp. 651–665, 2007.

2 Tatsuya Imai and Alex Fukunaga. *On a practical, integer-linear programming model for delete-free tasks and its use as a heuristic for cost-optimal planning.* Journal of Artificial Intelligence Research 54, pp. 631–677, 2015.

3 Michael Katz and Carmel Domshlak. *Optimal admissible composition of abstraction heuristics.* Artificial Intelligence 174(12–13), pp. 767–798, 2010.

4 Florian Pommerening, Gabriele Röger, and Malte Helmert. *Getting the most out of pattern databases for classical planning.* In Proceedings of the 23rd International Joint Conference on Artificial Intelligence, pp. 2357–2364, 2013.

5 Florian Pommerening, Malte Helmert, Gabriele Röger, and Jendrik Seipp. *From non-negative to general operator cost partitioning.* In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 3335–3341, 2015.

3.5 Heuristics for Numeric Planning

Patrik Haslum (Australian National University, AU)

Numeric planning extends the classical discrete state planning model with state variables whose values are rational numbers and actions whose preconditions and effects involve them. Unlike in classical setting, this makes the set of reachable states potentially non-finite, but apart from that, numeric planning keeps all the classical planning assumptions of determinism, observability and model completeness. Although semantically simple, this extension is quite powerful: the plan existence problem is semi-decidable even for very restricted fragments of numeric planning. Extending classical planning in a different way, "semantic attachments" are procedurally defined ("black box") functions or predicates used to model aspects of a planning problem that do not fit in the classcial model.

Recently, several relaxations, with associated heuristics, have been proposed both for general and restricted numeric planning and for planning with semantic attachments. This allows heuristic search methods, which have been very successful in classical planning, to address these extend models as well.

The aim of this tutorial was to give an overview of the numeric planning formalism, some of its relaxations, and draw parallels with ideas in OR.

3.6 Introduction to Constraint Programming

Pierre Schaus (UC Louvain, BE)

Main reference Laurent Michel, Pierre Schaus, Pascal Van Hentenryck: "Mini-CP: A Minimalist Open-Source Solver to teach Constraint Programming". 2017.
URL https://www.info.ucl.ac.be/~pschaus/minicp

We introduced constraint programming and its driving mantra CP=modeling+search. Standard examples such as job-shop models were used to illustrate the CP technology. Filtering algorithms for global constraints were briefly described for cumulative scheduling constraints, table constraints, element constraints, etc. Some implementation details about CP solvers internals were given: the search and state-restoration techniques (trails, DFS, fix-point computation). The differences between CP models and MIP/SAT models were emphasized. The importance of the search heuristics was illustrated with recent black-box searches able to learn from conflicts during the search. Several hybridization techniques were mentioned briefly: lazy-clause generation, Lagrangian-based filtering, column generation, multi-decision diagrams. Large Neighborhood Search was explained as a CP and Local-Search hybridization to make CP scale well and improve any-time behavior on large size problems.

3.7 Compiling Planning to Constraint Programming

Roman Bartak (Charles University – Prague, CZ)

Main reference Roman Barták: "On Constraint Models for Parallel Planning: The Novel Transition Scheme", in Proc. of the Eleventh Scandinavian Conference on Artificial Intelligence, SCAI 2011, Trondheim, Norway, May 24th – 26th, 2011, Frontiers in Artificial Intelligence and Applications, Vol. 227, pp. 50–59, IOS Press, 2011.
URL http://dx.doi.org/10.3233/978-1-60750-754-3-50

The tutorial described several methods of encoding the problem of finding a plan with at most k actions as a constraint satisfaction problem. First, three constraint models were introduced, a straightforward (textbook) model, a Graphplan-based model, and a model with successor-state axioms. For each model its reformulation using tabular constraints was suggested and experimental comparison of all models was given. The best model was then extended using techniques such as lifting, symmetry breaking, and singleton consistency and efficiency improvement was experimentally demonstrated. The planner based on this final model is called SeP (sequential planner). In the second part, we presented a constraint model based on finite state automata that supports parallel actions. The obtained planner is called PaP (parallel planner). The model was then reduced to achieve even better efficiency, which was demonstrated experimentally.

3.8 Introduction to Non-deterministic Planning

Christian Muise (IBM TJ Watson Research Center – Cambridge, US)

Main reference Christian Muise, Sheila A. McIlraith, J. Christopher Beck: "Improved Nondeterministic Planning by Exploiting State Relevance", in Proc. of the 22nd Int'l Conf. on Automated Planning and Scheduling, pp. 172–180, 2012.
URL http://www.aaai.org/ocs/index.php/ICAPS/ICAPS12/paper/view/4718

Non-deterministic planning [1] is a variant of planning where the action outcomes are uncertain but can be observed at execution time. Unlike classical planning, where a solution is a sequence of actions, a solution to a non-deterministic planning problem corresponds to a policy that maps the state of the world to an action for execution. In this talk, I presented the fully observable non-deterministic (FOND) planning formalism, outlined the main solution techniques that exist for FOND, discussed some of the applications of FOND planning, and concluded with some of the major open research challenges in the field.

References

1 Alessandro Cimatti, Marco Pistore, Marco Roveri, and Paolo Traverso. *Weak, strong, and strong cyclic planning via symbolic model checking*. Artificial Intelligence 147(1) pp. 35–84, 2003.

3.9 Markov Decision Processes in Planning

Matthijs Spaan (TU Delft, NL)

Joint work of Matthijs Spaan, Erwin Walraven
Main reference Erwin Walraven, Matthijs T. J. Spaan: "Accelerated Vector Pruning for Optimal POMDP Solvers", in Proc. of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA., pp. 3672–3678, AAAI Press, 2017.
URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14960

Planning under uncertainty can be formalized using Markov Decision Processes (MDPs) [1]. In this talk, I first introduced the MDP model for fully observable environments and basic algorithms for computing policies such as value iteration and policy iteration. Next, I considered Partially Observable Markov Decision Processes (POMDPs), which extend MDPs to handle planning for agents that cannot perfectly sense the state of the system. Finally, I discussed recent work that builds on a Benders decomposition to speed up solving of linear programs in optimal POMDP solving.

References

1 Matthijs T. J. Spaan. *Partially Observable Markov Decision Processes*. In Reinforcement Learning: State of the Art, pp. 387–414, Springer Verlag, 2012.

3.10 Decision Diagrams for Planning and Optimization

Scott Sanner (University of Toronto, CA)

Decision diagrams have proved to be a useful data structure for avoiding state enumeration in model checking, temporal verification, graphical model inference, classical planning, and value and policy optimization in factored MDPs and POMDPs. This tutorial covered the foundations of binary and algebraic decision diagrams (BDDs & ADDs) – their properties, their algorithms, their use in various automated planning settings. Beyond BDDs and ADDs, the tutorial also covered a variety of less well-known but important decision diagrams and their applications: Zero-suppressed DDs (ZDDs) for set representation, Affine ADDs (AADDs) for arithmetic function representation, and recent extensions of decision diagrams to continuous variables (XADDs) enabling novel exact solutions to a variety of continuous planning problems.

3.11 Decision Diagrams for Optimization

Willem-Jan Van Hoeve (Carnegie Mellon University – Pittsburgh, US)

Joint work of David Bergman, André A. Ciré, Willem-Jan van Hoeve, John N. Hooker
Main reference David Bergman, André A. Ciré, Willem-Jan van Hoeve, John N. Hooker: "Decision Diagrams for Optimization", Springer, 2016.
URL http://dx.doi.org/10.1007/978-3-319-42849-9

Binary decision diagrams have a long history in computer science, where they have been widely used as an efficient data structure for representing and solving circuit design and verification problems. Decision diagrams are a relatively new tool for optimization, however, and were introduced for this purpose about 10 years ago. This presentation described the development and application of decision diagrams as a framework for generic discrete optimization [1]: Decision diagrams can be used as discrete relaxations to generate dual bounds, as restrictions to generate primal solutions, and they form the basis for a new branch-and-bound search scheme. For several classical optimization problems, including the independent set, maximum cut, and maximum 2-SAT problems, decision diagrams can be competitive with or outperform state-of-the-art integer programming solvers such as CPLEX. Moreover, decision diagrams have been successfully applied to solve disjunctive scheduling and routing problems, and were able to close several open sequential ordering problems from the well-known TSPLIB benchmark set.

References

1 David Bergman, André A. Ciré, Willem-Jan van Hoeve, and John N. Hooker. *Decision Diagrams for Optimization*. Springer, 2016.

4 Working groups

4.1 MINLP and Nonlinearities in Planning

J. Christopher Beck (University of Toronto, CA)

The session was devoted to discussing the ways in which nonlinear constraints had been incorporated into planning models and solution techniques with the goals of exposing the work to the MINLP present.

4.1.1 Planning vs. Compilation for Linear and Non-linear Problems

Patrik Haslum discussed his experience with planning for complex systems where the dynamics of the environment could be described with linear or non-linear constraints. Such problems arise for example in determining power flow on a network (e.g., opening and closing switches to maintain network safety, power levels, etc.). While the power flow equations are non-linear, power engineers often use a linear relaxation.

There are (at least) two competing approaches: 1) use planning technology with an optimization sub-solver called at important points (often at each node in the planning search) and 2) unroll the whole problem in to an optimization problem ("unroll[ing]" is needed because there is a need for repeated decision making over time as decisions are taken and the network state is observed).

When a linear relaxation is used:

- Planning-with-LP: 50% of instances solved in < 1800 seconds with 1-3 faults (that is for problems with 1 to 3 simulated faults)
- MIP: 100 seconds for 5+ faults

However, if the nonlinear model is solved:

- Planning-with-NLP: 50% solved in 1000 seconds
- MINLP: 30% solved in 1000 seconds

So MIP beats planning-with-LP in the linear model but planning-with-NLP beats MINLP in the nonlinear case.

Conjecture: The MIP solver has benefited from last 2 decades of intense development but MINLP solver has not. But there is also the problems of comparing commercial MIP solvers to research MINLP code: MIP solvers are commercially software engineered, MINLP are not.

From a research perspective it is a general question to understand what decisions should be pushed to what technology with the unrolled solvers being an extreme version (everything in optimization). The answer depends not only on the (software) maturity of the solvers involved but also the problem solving leverage that planning might provide.

4.1.2 SMT for Planning with Non-linear Temporal Change

Michael Cashmore presented the problem of planning with non-linear change. The problem has discrete modes and nonlinear evolution while it is in a particular mode. It is a control problem with the goal of finding the sequence of the changes to the discrete state to achieve a goal.

The approach taken is SAT Modulo theory approach using a time-indexed model. Each index corresponds to a "happening" where discrete change takes place. Between the happenings there is continuous nonlinear change on the numeric variables.

One challenge is how do we check the numeric constraints between happenings. They have created a system that combines quantifier-free non-linear real arithmetic using a computer algebra based preprocessing system to do symbolic integration to derive expressions to include in the theory solver. The time between happenings is the maximum interval such that the numeric constraints are guaranteed to be satisfied. The second happening is placed at the earliest point where the constraints could be broken so that discrete change could deal with it.

Questions:

- How could this be solved with MINLP? Can we take the SMT model and solve it directly with an MINLP solver?

4.1.3 Planning with Non-linear (But Non-temporal) Change

Chiara Piacentini presented the problem of planning with nonlinear changes to the numeric state variables that do not depend on time - that is, with some nonlinear effects. This is not a problem with the progression of the planning because the state is known and so you can model the impact of the nonlinearities on the numeric variables getting you to the next step. A challenge arises in calculating the heuristic because as you calculate the heuristic distance estimate you do not have a state in the heuristic calculation.

As an example: power flow problem based on the discrete changes in power settings at the nodes in the power flow network. Linearizing the power flow equations can result in an invalid state but this can be good enough for the heuristic.

Another approach is a benders/semantic attachment approach where the planning-level relaxation is a hand-built linearization of the nonlinear power flow equations. The problem is how to automatically generate the linearization that could be used in heuristic.

Questions:

- Are there connections with spatial branching in MINLP?
- Is the concept of semantic attachments really a analogous to Benders? Refining the relaxation which Benders does would seem to require a re-evaluation of the open list.

4.2 Benchmarking

Christina N. Burt (Satalia – London, GB)

Speakers Malte Helmert (Universität Basel, CH) and Domenico Salvagnin (University of Padova, IT)

The benchmarking discussion focussed on commonalities and differences in both issues and solutions between the mathematical programming and automated planning communities. In both communities it was acknowledged that benchmarking pushes the state of the art and has a big impact on research.

Several issues were raised with benchmarking in general, with some proposed solutions. The first point related to comparing two algorithms, and determining what metric or point of comparison is appropriate, such as time to solve. First it must be ensured that the algorithms are comparable, such as having the same settings and parameters. When comparing algorithms based on time, the environment should be clean (e.g. on a cluster, ensure all machines are identical and each job is run exclusively on each machine). It was also highlighted that we should encourage researchers to state which CPU they use in their experiments, as this is more meaningful than processor speed. The second issue raised was how to be sure an improvement in algorithm performance is not due to an artefact. In MIP benchmarking, multiple runs of the algorithm are made on the same instance with differing random seeds in order to evaluate if changes to the algorithm are consistent.

This led to a discussion of how to select benchmarking instances in a fair way. e.g. finding instances that are hard for both solvers/algorithms. It is biased to take all the problems that are difficult for one algorithm as the test-bed. Finally, the topic of data aggregation was discussed, in particular that the way the results are aggregated can change the outcome. Using shifted geometric mean helps prevent bias toward smaller numbers. Truncated geometric mean has the same effect. However, it was suggested that we could move away from relative metrics, such as how far this solver is away from the fastest, but instead award points for achievements, such as solving in 1 second.

4.2.1 Talk: Benchmarking in MIP

Domenico Salvagnin (University of Padova, IT)

We surveyed the major challenges and pitfalls in algorithm benchmarking, and how they are currently handled by the MIP community.

4.3 Merge&Shrink and Decision Diagrams

Christina N. Burt (Satalia – London, GB) and Florian Pommerening (Universität Basel, CH)

Speakers Malte Helmert (Universität Basel, CH) and Willem-Jan Van Hoeve (Carnegie Mellon University – Pittsburgh, US)

In this discussion session moderated by Willem Van Hoeve we investigated the connections of merge&shrink [1], decision diagrams [2], and regular language constraints [3]. We started with an example planning task that has two variables x and y. Both variables can take values from $\{0, 1, 2\}$, are initially 0 and should get a value of 1 in a goal state. There are five operators a, b, c, d, e that change the variables as shown in Figure 1.

Consider only variable x for now. The behaviour of a variable in a planning task can be described with a deterministic finite automaton (DFA). The alphabet of the DFA contains all operators, and a word (i.e., an operator sequence) is in the language of the DFA iff executing the operator sequence in the projection to this planning variable leads to goal state. The state space of the projection almost directly corresponds to the transition function of the DFA: the only difference is that a DFA has exactly one outgoing transition for every symbol in every state. This can be easily satisfied by adding a dead state and route all missing transitions to this state. Other than that, the states of the DFA correspond to the value a variable has and appying an operator changes this value, so it brings the DFA into a new state.

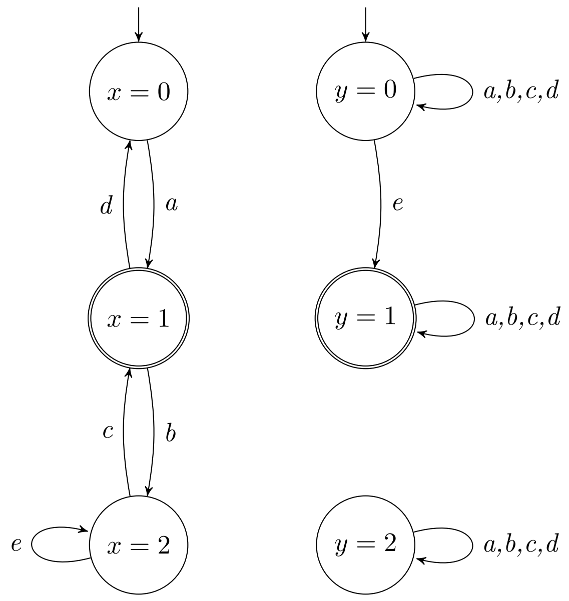

Figure 1 Example planning task. Each diagram shows the projection of the task to one of the variables, i.e. the effect of all operators on the variable.

A solution to the planning task has to bring all variables to their goal value, i.e. we have to find a single plan that is a plan in all projections. With the interpretation above this means that we have to find a word that is in the intersection of several regular languages. This yields a simple compilation of planning tasks to CP using the `regular` constraint for regular languages [3]: We use a fixed time horizon of n steps and introduce a CP variable X_t for every time step $1 \leq i \leq n$. The domain of each variable is the set of operators ($\{a, b, c, d, e\}$ in our example). There is one constraint $\texttt{regular}(X, \rho_j)$ for each planning variable j, where ρ_j is the regular expression for the DFA that describes the behavior of j. In our example task, ρ_x is the regular expression $a((da)|(be^*c))^*$ and ρ_y is $(a|b|c|d)^*e(a|b|c|d)^*$ (cf. Figure 1). By using the `cost-regular` constraint [4] instead, we can also support action costs in planning. Additional redundant constraints can be added using landmarks. For example, if a landmark analysis finds out that operator o is used at least k times, we can add the constraint $\sum_i(X_i = o) \geq k$. We could also use a global cardinality constraint [5] to express bounds on several operators: if the number of operator uses of operator o is between L_o and U_o then the global constraint $\texttt{gcc}(X, L, U)$ expresses the constraint $\bigwedge_o L_o \leq \sum_i(X_i = o) \leq U_o$. A CP-based planner would probably require such landmarks to be efficient but they can be extracted from the planning problem.

A `regular` constraint for finite sequences can be expressed with an MDD as shown in Figure 2. By intersection of these MDDs we can solve the planning problem for a bounded plan length. The shortest plan can then be found by iteratively solving for longer sequences. The MDD can be relaxed by bounding its width so the computation remains polynomial and returns a lower bound on the cost of a cheapest plan.

It is generally not possible (unless PSPACE = NP) to have a polynomial algorithm without the bounded length restriction because planning is PSPACE-complete. However, we can unroll levels only partially and use loops in the last or first layer to maintain completeness. This was under investigation in a Masters thesis in Basel [6].

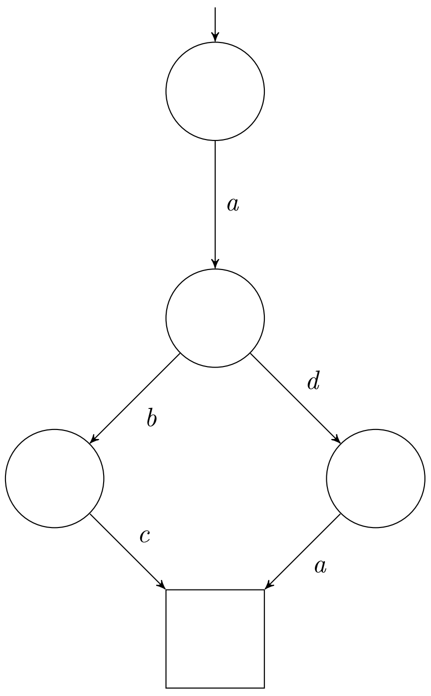

Figure 2 MDD expressing the behavior of variable x in our example task for sequences up to length 3.

The merge&shrink heuristic [1] is a planning heuristic that uses a similar approach to get a lower bound on the plan cost. The heuristic computation starts with the projections to all planning variables (i.e., the DFAs). All DFAs together preserve the original behavior of the system. Replacing two DFAs with their product automaton (called "merging") does not lose any information. Thus automata are merged while there is enough space to represent the resulting products. If a given space limit is reached, two states in one of the DFAs are combined into one (called "shrinking"). It would be possible to get a bound from each automaton but the bounds get better by combining all of them. Thus the algorithm continues merging and shrinking until there is only one automaton left. It is not so easy to keep track of which DFA state belongs to which states from the original problem after some merging and shrinking steps since we cannot keep track of all possibilities during the construction in a table. This would be as difficult as enumerating all states. Instead, trees are used to map partial states to abstract states. For the original DFAs, the tree consists of a root node labeled with the variable and one leaf for each abstract state and edges that are labeled with the variable values. The merging step combines two trees by appending one tree at each leaf state of the other tree (and then possibly simplifying the result). The shrinking step just relabels some leaf states (followed by simplifications). The representation at any time is an MDD. Since the leaves are the abstract states of all automata, its size is limited by the space limit of the heuristic.

The heuristic function of merge&shrink can be represented as an ADD by writing the heuristic values into the leaf nodes instead of the abstract states as for example done in symbolic merge&shrink heuristics. The interesting connection to the earlier discussion is that the memory bound in merge&shrink is essentially bounding the width of the MDD/ADD. While the bound is actually bounding the number of leaves, trees are combined by replacing leaves of one tree by another tree so limiting the number of leaves in turn limits the width of the decision diagram on all intermediate layers as well.

However, there are differences between the decision diagrams created by merge&shrink and those created for a regular language constraint since the variables do not represent the same objects. In the compilation to CP, variables represent operators used at specific time steps, unrolling the state space to some depth. In the heuristic on the other hand, the decision diagram stores the information to which abstract state an original state is mapped.

Apart from the use as a lower bound (heuristic), an MDD as build by merge&shrink could be useful for several other questions. If it were not relaxed, it would represent all solutions, so it could for example be used to count the number of solutions. It could also be limited to paths with a certain cost to represent all optimal paths. A relaxed MDD can give approximate answers to these questions.

Another interesting direction would be to use a CP model based on `regular` constraints for planning heuristics that extract more information than just a heuristic value. This way, it could for example easily be combined with operator-counting heuristics. Other suggestions that came up in discussion were to combine merge&shrink with more information such as mutexes (this is possible in general but takes a lot of memory) and using MDDs for proving unsolvability. Merge&shrink has been specialized for unsolvability where more abstract states can be merged without losing information. Unrolling the DFAs of all state variables up to a certain bound could also be used to prove that plans cannot have a certain length. However self-loops in the state space are common and lead to a problem because with loops it is always possible to unroll the state space one level further.

Relaxed MDDs can also be used to represent the whole problem. Ongoing work in Toronto builds an MDD and extracts relaxed solutions from it, unrolling the MDD until an actual solution is found. The difference to the methods described above is that this does not start from the DFAs describing the behavior of each variable.

References

1 Malte Helmert, Patrik Haslum, Jörg Hoffmann, and Raz Nissim. *Merge-and-Shrink Abstraction: A Method for Generating Lower Bounds in Factored State Spaces.* Journal of the ACM 61 (3), pp. 16:1–63. 2014.

2 David Bergman, André A. Ciré, Willem-Jan van Hoeve, and John N. Hooker. *Decision Diagrams for Optimization.* Springer, 2016.

3 Gilles Pesant. *A regular membership constraint for finite sequences of variables.* In Proceedings of the Tenth Conference on Principles and Practice of Constraint Programming, pp. 482–495, 2004.

4 Sophie Demassey, Gilles Pesant, Louis-Martin Rousseau. *A Cost-Regular based Hybrid Column Generation Approach.* Constraints 11 (4), pp. 315–333, 2006.

5 Jean-Charles Régin. *Generalized Arc Consistency for Global Cardinality Constraint.* In Proceedings of the Thirteenth National Conference on Artificial Intelligence, pp. 209–215, 1996.

6 Philipp Oldenburg. *Time Unrolling Heuristics.* Master Thesis, University of Basel, Switzerland, 2017.

4.4 Teaching

Hadrien Cambazard (Grenoble INP, FR) and Florian Pommerening (Universität Basel, CH)

Speakers Hadrien Cambazard (Grenoble INP, FR), Emmanuel Hebrard (LAAS – Toulouse, FR), Serdar Kadioglu (Fidelity Investments – Boston, US), and Pierre Schaus (UC Louvain, BE)

The goal of this session was to discuss the use of optimization techniques for teaching as well as discussing way to teach these techniques.

Pierre Schaus started the session by presenting MiniCP [1], a CP solver dedicated to teaching CP. The solver has the same design goal as the well-known MiniSat solver [2]: to provide a simple solver that can easily be investigated and used to learn about the most important techniques in a modern CP solver. MiniSat has largely contributed to the dissemination of SAT solvers and MiniCP should do the same for CP solvers. A solver like this can be used as a tool to help people get into the CP community and teach newcomers the fundamentals of how CP solvers are implemented. A suggestion that came up in discussion was to run a MiniCP competition.

Serdar Kadioglu then reported on his experience with gamification for a course on SAT, CP, LP, and MIP. The course was driven by projects which ran over three weeks. Each project started with a high level narrative description, a defined input/output format and students then designed their own solutions. A leader board was used to give immediate feedback on how well their strategy worked and the quality of results of the group are visible to all students but are kept anonymous. An interesting observation is that once a student managed to get a first feasible solution, others succeeded usually quickly afterwards. Because students were engaged and driven to improve their approach, solution qualities tended to converge and the projects became very hard to grade. Overall the engagement of students was very strong. The discussion focused on the advantages and disadvantages of this approach for teaching and academia. For example, this way of teaching might lead to a more academic orientation of students by encouraging them to search for innovative solutions by themselves. The difficulty in grading was also discussed in more detail. This issues is shared with massive open online courses (MOOCs) where techniques like this one are also used. The discussion then turned on whether and how online classes will affect the way universities work? Serdar argued that so far, we have not seen strong effects but these new techniques are likely to change our practices.

Next, Emmanuel Hebrard showed a Python solver called PySched [3] that provides state-of-the-art filtering algorithms in CP and can output diagrams in LaTeX to visualize the reasoning performed by the solver. Christian Muise asked if teaching materials like PySched or MiniCP are collected and shared. An interesting idea would be to include a demonstration about teaching in the ICAPS demo session.

Finally, Hadrien Cambazard presented Caseine [4], a learning platform in Industrial Engineering, Mathematics and Computer science. Its aim is to stimulate students' learning and autonomy while improving the quality of the time the teacher gives them. Based on Moodle, it allows to automatically evaluate the students' computer code and mathematical models, to monitor the students' progress, and to share content among the teachers through a community of users. It is meant to support "active classrooms" or "flipped classrooms". Caseine could be an option to share teaching materials for CP as suggested by Christian Muise earlier.

References

1 Laurent Michel, Pierre Schaus, Pascal Van Hentenryck. *Mini-CP: A Minimalist Open-Source Solver to teach Constraint Programming*. 2017. Available from www.info.ucl.ac.be/~pschaus/minicp.
2 Niklas Eén and Niklas Sörensson. *An extensible SAT-solver*. In Proceedings of the Sixth International Conference on Theory and Applications of Satisfiability Testing, pp. 502–518, 2003.
3 Emmanuel Hebrard https://github.com/ehebrard/PySched Accessed May 2018.
4 Hadrien Cambazard. http://caseine.org. Accessed May 2018.

4.4.1 Talk: Mini-CP – A Minimalist Open-Source Solver to teach Constraint Programming

Pierre Schaus (UC Louvain, BE)

Joint work of Laurent Michel, Pierre Schaus, Pascal Van Hentenryck
Main reference Laurent Michel, Pierre Schaus, Pascal Van Hentenryck: "Mini-CP: A Minimalist Open-Source Solver to teach Constraint Programming". 2017.
URL https://www.info.ucl.ac.be/~pschaus/minicp

The successful minisat solver has largely contributed to the dissemination of (CDCL) SAT solvers. Minisat has a neat and minimalist architecture that is well documented. We believe the CP community is currently missing such a solver that would permit new-comers to demystify the internals of CP technology. We introduce Mini-CP a white-box bottom-up teaching framework for CP implemented in Java. Mini-CP is voluntarily missing many features that you would find in a commercial or complete open-source solver. The implementation, although inspired by state-of-the-art solvers is not focused on efficiency but rather on readability to convey the concepts as clearly as possible. Mini-CP is small ($<$ 1500 LOC excluding tests) and well tested.

4.5 Hybrids & Decomposition

Mathijs de Weerdt (TU Delft, NL)

Speakers Jeremy D. Frank (NASA – Moffett Field, US), John N. Hooker (Carnegie Mellon University – Pittsburgh, US), and Louis-Martin Rousseau (Polytechnique Montreal, CA)

The goal of this session was to discuss commonalities of MIP, CP, and planning, and how ideas such as for efficient representations of the solution space, heuristics, upper and lower bounds, and decomposition can be more easily combined.

We had three main speakers in this session. John Hooker started with explaining (logic-based) Benders decomposition and the relation to similar ideas in MIP, CP, SAT, SMT, and planning. Jeremy Frank then talked about an architecture to allow combining many different techniques in solving real-life problems. Finally, Louis-Martin Rousseau explained how a combination of column generation and dynamic programming can be used to solve a complex vehicle routing problem (VRP).

4.5.1 Benders decomposition

Logic-based Benders decomposition is a method to facilitate communication between subproblems via a master problem. John Hooker explained this using two concrete examples. Here I give one.

Suppose the problem is to assign jobs with deadlines and processing times to agents that each have resource constraints (cumulative scheduling) and each agent may have different costs for executing a job. The master problem in this case is minimizing the allocation costs of assigning jobs to agents, and the subproblems are for each agent its resource-constrained scheduling problem. If an agent cannot fit all its allocated jobs, a constraint (Benders cut) is added to the master problem, which is subsequently resolved. A good cut would be a small subset of allocated jobs that together lead to infeasibility. This can be found using a dual formulation of the subproblem.

This approach has strong similarities to well-known successful approaches in various fields. In SAT solving, conflict clauses are exactly Benders cuts. In SMT the theory part is in fact a subproblem. Semantic attachments also represent a subproblem. This type of conflict analysis in MIP is called branch and check. In planning a relaxed graph or MDD can be seen as the master problem, and updating this structure can then be seen as adding Bender's cuts.

4.5.2 Architecture for hybrid approaches

Jeremy Frank talked about a planning/constraint reasoning architecture consisting of

- A temporal network
- Causal link and resources over timelines
- SAT/CSP/SMTs
- MILP
- Semantic attachments: physics of battery, power, orbit

This planning/constraint reasoning architecture has been used by several automated planners developed for applications with many different types of constraints, ranging from the usual causal constraints, to time and simple resources, to physical simulations.

The architecture splits constraints into categories for which efficient algorithms exist (e.g. temporal constraints). Doing this, however, requires both extracting the 'homogeneous' constraints from the planning model, transporting information between components that reason about a specific type of constraint, and transporting information to components that perform functions like heuristic evaluation. The architecture resembles the MILP architecture presented earlier by Thorsten Koch, but is somewhat more complex.

Generally, search in this architecture is incremental (commit to a single action or constraint at a time). This approach emphasizes the need for constraint reasoning and propagation, and efficient rollback; it facilitates many types of search (progression, regression, local search, and mixed initiative search). Unlike traditional constraint satisfaction and MILP techniques, the whole problem is not available for reasoning and heuristic decision making.

The discussion continued with the use of semantic attachments. If these are (costly) simulator runs, can we still extract useful information from this, such as the Bender cuts from John Hooker's talk? How to analyze the results from such semantic attachments to generate useful information, whether constraints or heuristic decisions, is an open question.

4.5.3 Column generation and DP for VRP

Louis-Martin Rousseau explained how at Hanalog they use a hybrid approach for a complex VRP problem with extra constraints such as on capacity and truck driver regulations, and continuity of care in the context of Homecare.

He first explained the approach of column generation in general: if a solution can be seen as a combination of solutions to subproblems, it may be more efficient to enumerate such solutions and repeatedly try to find a good combination of the ones constructed thus far. This works well for problems with some underlying structure, for example for Vehicle routing, Airline Scheduling, Shift Scheduling and Jobshop Scheduling.

For VRP the MIP constraints can lead to bad relaxations and column generation works quite well. In this case, the subproblem is finding a good route through a set of customers. For the problem at hand, there are multiple relevant resource constraints (such as the time already spent at other customers). This can however, be reasonably efficiently be expressed by a (multi-label) dynamic program. Perhaps some links can be established with the techniques used in planning here as well?

4.5.4 Discussion

We can indeed conclude that there are strong similarities between algorithms in the different paradigms (like Benders decomposition), and also concrete examples where hybrids are more successful than formulating the whole problem in one of the paradigms. It is decided that in a next session we will discuss also other similarities (e.g. with decision diagrams).

4.6 Non-traditional Objective Functions for MDPs

Patrik Haslum (Australian National University, AU)

The MDP model is used and solved both in OR (where it originated) and in AI. Sven posed two questions:

1. what has AI contributed to solving MDPs?
2. what kind of objective functions (for MDPs) can be solved, and which ones do we want to use?

The main answer to the first question was that AI contributed structured representations like probabilistic planning, factored MDPs, heuristic search, macro actions, decision diagram representations of value functions, and so on.

Concerning the second question, most AI methods work (well) with problems that are Markovian including objectives like expected (discounted) reward (for MDPs), end-state goals and sum of costs (in deterministic planning).

But utility functions, in particular for one-shot high-stakes decisions, are non-linear functions of accumulated reward (e.g., in economic decisions, attitude to risk is dependent on the amount at stake relative to total wealth), i.e., non-Markovian. Temporally extended goals (as expressed in LTL, CTL) are also non-Markovian. We can solve these by encoding enough history into the state to make the problem Markovian again, then apply (efficient) AI methods. On the other hand, encodings of planning (deterministic or not) into SAT, CP, MIP, etc, provide a representation of the space of (bounded-length) solution plans, where non-Markovian constraints (i.e., goals) and objectives can potentially be expressed directly.

There is an interesting link between this and multi-objective problems, in efficient (symbolic) representations of the Pareto front.

4.6.1 Talk: Non-Traditional Objective Functions for MDPs

Sven Koenig (USC – Los Angeles, US) and Yaxin Liu

Joint work of Yaxin Liu, Sven Koenig
Main reference Yaxin Liu, Sven Koenig: "Functional Value Iteration for Decision-Theoretic Planning with General Utility Functions", in Proceedings of the Twenty-First National Conference on Artificial Intelligence, July 16-20, 2006, Boston, Massachusetts, USA, pp. 1186–1193, AAAI Press, 2006.
URL http://www.aaai.org/Library/AAAI/2006/aaai06-186.php

We argue in this historical perspective on joint research with Yaxin Liu that research on probabilistic planning with Markov Decision Processes (MDPs) so far has been more interested in exploiting their structure for efficient planning for simple objective functions than in studying more realistic and thus also more complex objective functions. We try to understand the reasons for it, outline both some early and current AI research on non-traditional objective functions and explain why future AI research should focus more on realistic objective functions to make probabilistic planning more attractive for actual applications.

As an example, we discuss how to find plans that maximize the expected total utility for a given MDP, a planning objective that is important for decision making in high-stakes domains. The optimal actions can now depend on the total reward that has been accumulated so far in addition to the current state. We present our research from more than 10 years ago that extends the functional value iteration framework from one-switch utility functions to all utility functions that can be approximated with piecewise linear utility functions (with and without exponential tails) by using functional value iteration to find a plan that maximizes the expected total utility for the approximate utility function. Functional value iteration does not maintain a value for every state but a value function that maps the total reward that has been accumulated so far into a value. We describe how functional value iteration represents these value functions in finite form, how it performs dynamic programming by manipulating these representations and what kinds of approximation guarantees it is able to make.

See http://idm-lab.org/project-d.html for more information and published papers.

4.7 Open and Challenging Problems

Patrik Haslum (Australian National University, AU)

Speakers Malte Helmert (Universität Basel, CH), Thorsten Koch (Konrad-Zuse-Zentrum – Berlin, DE), Andrea Micheli (Bruno Kessler Foundation – Trento, IT), and Hana Rudová (Masaryk University – Brno, CZ)

Four speakers outlined challenging problems: Hanna Rudova, Andrea Micheli and Torsten Koch discussed application problems. Malte Helmert described the linear programs that arise in cost-partitioning planning heuristics, and raised the question what ways there are to exploit the particular structure of these LPs to solve them faster.

Hanna Rudova described the university course timetabling problem. Timetabling is a long-studied problem, but its practical application usually have additional complex constraints and/or objectives. In the university course timetabling context, these can include scheduling of linked small group activities (such as tutorials and labs) and large group lectures, with precedences (e.g., small group activities in a given week must occur either all before or all after a lecture). Problems are typically over-constrained, and objectives include maximising satisfaction of participants time and room preferences, and minimising student conflicts and other "soft" constraint violations.

Andrea Micheli described a production scheduling problem in an electroplating plant. The problem is similar to the known "hoist scheduling problem", but has additional constraints. Hoists move parts between different different chemical baths, and there are minimum and maximum constraints on the time that each part spends in as well as between stages. Several hoists move on the same track, so one cannot bypass another; moreover, there are no "parking" spots where parts can be temporarily stored or handed over from one hoist to another. The problem is not only an operational one, but also solved as part of the plant design, for example to determine what is the optimal number of hoists for a production line.

Torsten Koch described operational problems in the control of energy systems, with specific focus on gas networks. Pressure in gas networks is controlled with a range of regulators, compressors, and valves (so it is a switching problem). The main constraint is to maintain a minimum pressure at network exits and a maximum pressure in the pipes. Optimisation problems arise at many scales, in network size (from a single building to continent-wide), level of abstraction, time horizon, and precision. European gas networks are getting old and subject to increasing use: as a result, they are operated closer to the boundaries, and more often parts are out of service. Thus, an optimisation formulation that treats all constraints as hard is more and more often not feasible.

4.7.1 Talk: The MAIS P&S Problem: Hoist Scheduling Problem in the wild

Andrea Micheli (Bruno Kessler Foundation – Trento, IT)

Joint work of Andrea Micheli, Alessandro Cimatti, Marco Roveri, Enrico Scala

This talk presents the characteristics of the planning and scheduling problem we are currently working on. The problem is a generalization of the Hoist Scheduling Problem for electroplating factories. In particular, we need to synthesize the movement plans for a number of hoists that need to respect a given "recipe" for production, maximizing throughput. Differently from most of the literature, we are not aiming at cyclic plans for a fixed kind of production, but we want to generate plans for a given order of production. We have a solution technique that is able to quickly generate feasible plans, but we now need to focus on (sub-)optimality.

4.7.2 Talk: Complex University Course Timetabling

Hana Rudová (Masaryk University – Brno, CZ)

Joint work of Hana Rudová, Tomáš Müller, Zuzana Müllerová

Datasets of complex university course timetabling problems will be collected for the International Timetabling Competition 2019. Datasets are available thanks to the UniTime timetabling system which is applied for timetabling at many institutions worldwide. Our goal is the creation of rich real-world datasets with diverse characteristics which will stimulate further research in timetabling and scheduling. The aim is to find an optimal assignment of times, rooms, and students to events related to the set of courses. Various hard constraints ensure the feasibility of the timetable and soft constraints define optimization criteria and the quality of the timetable. The key novelty lies in the combination of student sectioning together with standard time and room assignment of events in courses. We encourage scheduling and planning community to consider the study of our complex real-life problems. More details will be provided at the International Conference on the Practice and Theory of Timetabling 2018.

4.8 Explaining Solutions and Solution Methods

Christian Muise (IBM TJ Watson Research Center – Cambridge, US)

Speakers Christina N. Burt (Satalia – London, GB), Jeremy D. Frank (NASA – Moffett Field, US), Thorsten Koch (Konrad-Zuse-Zentrum – Berlin, DE), and Daniele Magazzeni (King's College London, GB)

This session focused on issues surrounding the need to explain solutions and solution methods, and primarily focused on explanations geared towards those that need to work with the solutions (as opposed to solver developers producing the solutions).

Dan Magazzeni discussed key challenges and techniques for explaining planning solutions; an increasingly important issue with new EU regulations being enacted that will require companies to explain why certain decisions based on AI have been made. Ultimately, the building blocks for planning (e.g., causal link inference) provide a natural means for solution explanation.

Christina Burt described her experience working for Satalia and the need to explain and provide solutions to clients. Typically they do not want just a single solution, but a variety of good solutions to compare. The approach they plan on taking is to solicit a set of soft constraints from the user, and iteratively relax them until a solution is found (which simultaneously provides an explanation of what is feasible).

Jeremy Frank detailed the challenges that arise in the human-interaction interface for the space exploration setting. Typically, interaction must occur between several subject matter experts, and insights drawn across the entire spectrum of expertise. This makes the task of providing (tailored) explanations to the humans working with the systems particularly difficult.

Finally, Thorsten Koch described some of the techniques for explainable infeasibility in MIPs. Pre-trained infeasibility can be readily explained, but the explanations for irreducible subsets of constraints can be unwieldy to understand. A popular alternative is to introduce slack variables into the problem (making it feasible) and then optimizing. At times, a natural explanation may not exist, as we are trying (and failing) to solve problems that require super-human capability.

A common thread from the workshop is that we should strive to find ways to model the task of explanation explicitly into the problem representation itself. Doing so allows the strengths of modern solver technology to be leveraged both for computing solutions, and explaining them as well.

4.8.1 Talk: Explainable Planning

Daniele Magazzeni (King's College London, GB)

Main reference Maria Fox, Derek Long, and Daniele Magazzeni. *Explainable Planning.* IJCAI Workshop on XAI, 2017.
URL https://arxiv.org/abs/1709.10256

As AI is increasingly being adopted into application solutions, the challenge of supporting interaction with humans is becoming more apparent. Partly this is to support integrated working styles, in which humans and intelligent systems cooperate in problem-solving, but also it is a necessary step in the process of building trust as humans migrate greater responsibility to such systems. The challenge is to find effective ways to communicate the foundations of AI-driven behaviour, when the algorithms that drive it are far from transparent to humans. In this talk we consider the opportunities that arise in AI planning, exploiting the model-based representations that form a familiar and common basis for communication with users, while acknowledging the gap between planning algorithms and human problem-solving.

4.9 Modeling

Christian Muise (IBM TJ Watson Research Center – Cambridge, US)

Speakers Roman Bartak (Charles University – Prague, CZ), Thorsten Koch (Konrad-Zuse-Zentrum – Berlin, DE), Christian Muise (IBM TJ Watson Research Center – Cambridge, US), and Scott Sanner (University of Toronto, CA)

This workshop focused on the issues in planning and OR when it comes to modeling: what representation to use, what level of abstraction to use, how does the solver technology influence the model, etc. The discussions leaders were Christian Muise, Roman Bartak, Thorsten Koch, and Scott Sanner.

First, we discussed the hierarchy of complexity that appears in both fields: e.g., moving from deterministic to probabilistic planning or moving from MILP to MINLP. The general advice is that even if the model is less clear to the user, using a less rich language can often lead to substantial improvements in the scalability of the solver technology. Knowing how to model problems effectively should thus include recognizing the appropriate level of expressivity and abstraction that can capture the problem.

We then discussed how the solving technology can influence the modeling itself. This is reflected most prominently in OR where the inclusion of additional constraints can drastically improve the performance of certain solvers. Additionally, not all solvers have the same expressivity in terms of constraints (particularly in the area of CP), and so solver technology will largely influence the modeling choices that are made.

The discussion then moved towards the parallel between problem modeling and programming languages. Just like the levels abstraction that we see when moving from assembly to C to Java or Python, we have a similar hierarchy in fields such as CP where lower-level constraints are succinctly represented by higher level ones. The longer-term hope is that we will need to rely less on modeling ingenuity and domain-specific hand-tailored heuristics, and rely more on sophisticated solvers recognizing the techniques that should be applied in the same sense as we currently rely on the sophistication in modern compilers for programming languages.

Finally, we discussed some of the key issues that exist in traditional probabilistic planning settings, and a modern proposal to use a different style of problem modeling. Rather than use the traditional focus of planning for state change to be action-centric (modeling the impact individual actions can have), the newly introduced language focuses on what makes up the state of the world and how that may change based on the agent's choice of control. This opens the door to modeling far more complex phenomenon where the possible successors of a state may be on the order of the entire state space itself.

4.9.1 Talk: Is Modeling Important for Efficiency of Problem Solving?

Roman Bartak (Charles University – Prague, CZ)

Joint work of Roman Barták, Lukáš Chrpa, Agostino Dovier, Jindřich Vodrážka, Neng-Fa Zhou
Main reference Roman Barták, Lukáš Chrpa, Agostino Dovier, Jindřich Vodrážka, Neng-Fa Zhou: "Modeling and solving planning problems in tabled logic programming: Experience from the Cave Diving domain", Sci. Comput. Program., Vol. 147, pp. 54–77, 2017.
URL http://dx.doi.org/10.1016/j.scico.2017.04.007

The talk argues for importance of problem modeling on efficiency of problem solving. By showing several examples from different areas, I argue that the formal model plays a big role in problem solving. I used the Golomb-ruler problem to demonstrate influence of symmetry breaking, redundant constraints, and search strategy on efficiency when using constraint programing. I also compared efficiency of several constraint models of classical planning problems. Next, I presented the role of state representation, heuristics, and control knowledge when solving planning problems using search (iterative deepening and branch-and-bound) in the Picat planner. Finally, I presented a comparison of various solving approaches, namely the Picat system, pure SAT-based solver, ASP, and search-based solver for multi-agent path finding. The talk was concluded by a list of questions that might serve as a future research program. Do we agree that modeling is important for efficiency of solving? Do we know what the good practices to model problems are? Are these practices available to users/modelers? Can we formalize these practices such that we can automatically reformulate models? Can we learn the models automatically?

4.10 Cross-Domain Example Problems

Florian Pommerening (Universität Basel, CH)

Speakers Michael Cashmore (King's College London, GB), Thorsten Koch (Konrad-Zuse-Zentrum – Berlin, DE), and Louis-Martin Rousseau (Polytechnique Montreal, CA)

In a panel discussion with Michael Cashmore, Thorsten Koch, and Louis-Martin Rousseau we talked about the differences and commonalities of AI planning, Constraint Programming (CP), and Mixed Integer Programming (MIP). A main goal of the discussion was to find properties in the structure of problems that make them specifically suitable for one of these approaches.

One of the features that is specific to planning is its focus on sequential aspects. A^* search can find sequential solution without knowing a limit on the horizon beforehand, while MIP and CP often require unrolling or resolving with different horizons for sequential solutions. Planning problems can also handle state-specific constraints or costs more naturally than the other approaches. In contrast to scheduling with MIP or CP which *orders* activities, planning *choses and orders* activities. This means that the solution of a planning problem is a path, not just the goal state at the end of that path. Therefore, search strategies can be used that do not have a match in the MIP or CP world, like searching backwards from a solution. Puzzles like the 4-peg towers of Hanoi, Freecell, or Rubik's cube are simple examples that exhibit strong sequential aspects and state-dependent constraints. As more real-world examples, intrusion detection, software verification, and the tactical level of protocol verification were proposed as examples.

MIP and CP on the other hand offer a global view on the problem. In contrast to planning where the focus is on local effects along a transition from one state to another, the focus for MIP and CP is on the whole problem with global constraints. One key difference between the MIP and CP communities is how constraints are used to exclude invalid assignments: while MIP algorithms approach the solution space from "outside" (e.g., using cutting planes), CP algorithms can also exclude assignments from the "inside" of the space (e.g., by removing values from a domain).

Another difference between the three areas that was brought up is the different value of domain-independent solutions. While planning research almost exclusively focuses on domain-independent techniques – to the degree that solving a specific problem is not considered planning – MIP models typically contain a lot of problem-specific optimizations. Likewise, finding a good representation for a given problem has a low priority in planning research, whereas it is a main focus of CP and MIP. Using domain-independent solutions can slow down planning on specific problems but allows rapid prototyping and transfer of solution techniques. A final implementation can then be a specialization of such a prototype that is a lot faster but with a parallel performance curve. On the MIP side, an out-of-the-box implementation of general algorithms (e.g., Benders decomposition) is missing and algorithms have to be specialized for each new problem.

In the discussion about typical problems for CP, most suggestions were highly discrete and disjunctive problems like n-queens, Crossword puzzles, or Sudoku. But also complicated scheduling problems were mentioned where state-dependent constraints can be more easily handled by CP than by MIP. In contrast to planning, CP seems better suited for problems without a temporal aspect (or with a fixed horizon) like graph coloring. Interestingly, Sudoku can be solved quite efficiently by MIP and the reason for this is not perfectly clear. MIP solvers have an advantage here because they use CP solvers in their pre-solve step which will handle all cases that are well-suited for CP and leave only the hard combinatorial part for the core MIP solver. It is common that a large part of the work is already done in this pre-solve step.

In contrast to the mostly discrete suggestions for problems fitting CP and planning, MIP easily deals with continuous problems. Min-cost flow problems, for example, would likely be best solved with MIP. In fact only LP is needed to solve them. However, adding state-dependent restrictions can make a min-flow problem harder for MIP again and might require a combination of techniques.

In real-world applications like many robotics applications, the problem is often a mix of many subproblems that might require different solvers. A common approach is to decompose such problems and solve each subproblem with a specialized solver. Finding the right tool for each subproblem is currently still a task that requires expert knowledge so so knowing more about the strengths and weaknesses of each tool would be a big help.

One common theme in all areas is that they solve problems by exploiting structure contained in the problem and each solution method exploits a different kind of structure. Instead of finding a typical planning, MIP, or CP problem we can thus try to describe what kind of structure each method can exploit. A question that was brought up was if we can build a hierarchy of structures that can be added to or removed from problems where the best solution method changes with each added kind of structure. This would require modeling a problem (e.g., vehicle routing) with many variants in different frameworks. Compiling between the frameworks would also work but may lead to bad models. However, something could be learned from comparing the naively compiled models to hand-crafted models for the same problem as well.

4.11 Dealing with Uncertainty

Florian Pommerening (Universität Basel, CH)

Speakers Michele Lombardi (University of Bologna, IT), Matthijs Spaan (TU Delft, NL), and Scott Sanner (University of Toronto, CA)

We discussed ways of dealing with uncertainty in planning and optimization. Michele Lombardi presented work that handles uncertainty by creating robust partial order schedules. This was followed by a presentation of Matthijs Spaan and a implementation technique suggested by Scott Sanner.

4.11.1 Talk: On the Robustness of Partial Order Schedules

Michele Lombardi

Joint work of Michele Lombardi, Alessio Bonfietti, Michela Milano
Main reference Michele Lombardi, Alessio Bonfietti, Michela Milano: "Deterministic Estimation of the Expected Makespan of a POS Under Duration Uncertainty", in Proc. of the Principles and Practice of Constraint Programming – 21st Int'l Conference, CP 2015, Cork, Ireland, August 31 – September 4, 2015, Proceedings, Lecture Notes in Computer Science, Vol. 9255, pp. 279–294, Springer, 2015.
URL http://dx.doi.org/10.1007/978-3-319-23219-5_20

Partial Order Schedules (POSs) are a convenient method for representing solutions to scheduling problems with end-to-start precedence constraints and limited resources. Specifically, a POS is a directed acyclic graph that contains the original precedence relations, augmented with extra arcs that prevent the occurrence of resource conflicts. POSs are naturally robust against duration uncertainty, since they allow recomputing a feasible schedule in polynomial time. A POS can be obtained very efficiently from a classical, fixed-start, schedule. In this talk, we present empirical results that show how, for a large variety of POSs obtained from multiple scheduling problems, there exists a very strong linear correlation between the expected value of the makespan and the makespan obtained by fixing all durations to their expected value. The correlation is strong enough that optimizing the makespan with fixed durations is approximately equivalent to optimizing the expected makespan. The result has immediate practical applications, since the former problem is computationally much simpler than the latter – assuming that expected value optimization is appropriate (e.g. for activity sets that must be repeated multiple times). In case of statistically independent durations, the behavior is due to the fact that interference from multiple paths and variance compensation on sequences of activities tend to cancel each other. In case of statistically dependent durations, formal arguments show that the correlation can be much weaker, although this may require quite unrealistic probability distributions. When more realistic scenarios are empirically evaluated, interestingly, the correlation with statistically dependent durations is often at least as strong as with independent durations. As a secondary result, the talk contains a proof sketch that a tight upper bound on the expected makespan can always be obtained by taking into account at most $n + 1$ scenarios in the sampling space, where n is the number of activities.

4.12 Connections Between our Fields

Mark Roberts (Naval Research – Washington, US)

Speakers Christina N. Burt (Satalia – London, GB), John N. Hooker (Carnegie Mellon University – Pittsburgh, US), Christian Muise (IBM TJ Watson Research Center – Cambridge, US), and Florian Pommerening (Universität Basel, CH)

During this session, several speakers spoke about interrelationships between CP, Planning, and Optimization. John Hooker started with a diagram that highlighted the key linkages discussed during the week. Figure 3 illustrates the key connections. CP can be used in planning for modeling and propagation. MIP can be used for operator counts in planning; MIP and CP have relationships relating to search and inference. Bender's cuts can be used in many applications for solving, conflict analysis, and operator counts. Bender's cuts can

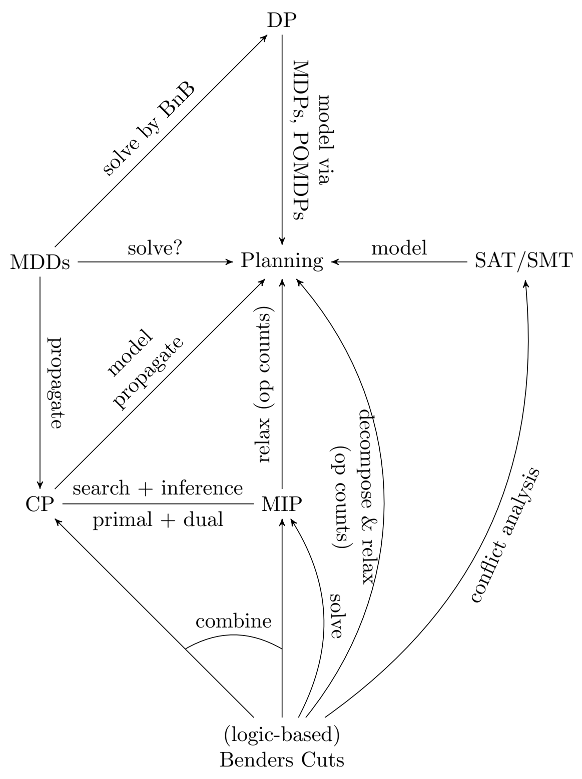

Figure 3 Interrelationships between CP, Planning, and Optimization.

also be used to decompose or relax planning problems. SAT and SMT can be used to model and solve planning problems. MDDs can be used to solve planning as well as for propagation in CP. MDDs can also be used to solve, via Branch and Bound, Dynamic Programming problems. DP can be used, via MDPs and POMDPs, to model and solve planning problems. Finally, planning can provide factored or structured representations for other techniques. During discussion, several issues were raised. One person noted that a central question related to pruning techniques, bounding, and solution set sizes. Another person noted the role of lower bounds under relaxations.

Florian Pommerening spoke about the the use of action or fact landmarks in planning. These serve a similar function to redundant constraints in CP. In terms of complexity, there can be exponentially many of these, and verifying any arbitrary one is as hard as planning. Some ideas that surfaced during the discussion related to the use of dead-end detection and optimization-based landmarks. It was also noted how landmarks relate to reductions in MIP. Specifically, knowing that an operator (or a fact) is a landmark is similar to knowing that a certain LP variable must take a certain value and can be fixed. For disjunctive action landmarks, the exact value of an LP variable may not be known, but knowing some information about it corresponds to a cut.

Christian Muise spoke about the relationship of determinism and observability in planning to various topics from the week. Figure 4 illustrated the relationship of POMDP, MDP, FOND, POND, POD, FOD. Along one axis, observability can be fully observable (FO) or partially observable (PO). On the other axis, determinism is varied from deterministic (D), non-deterministic (ND), and the Markovian Decision Problem (MDP). These types of planning relate to topics from the week, which included irreducible infeasible subsets in

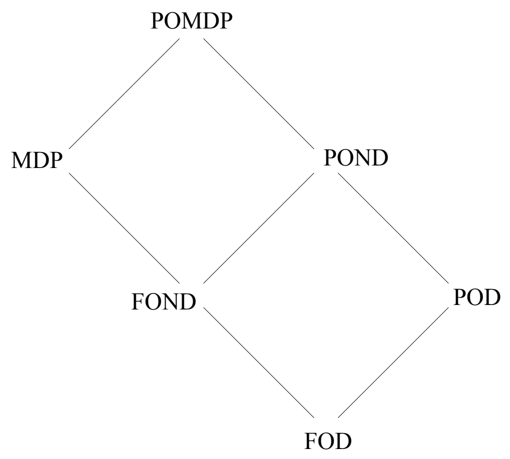

Figure 4 Relationship between different planning formalisms.

MIP, landmarks, strengthening benders cuts in CP, quick explain in CP, and multi-agent planning under uncertainty. Importantly, landmarks can play a central role in several of these techniques.

4.13 LP Heuristics: The Details

Gabriele Röger (Universität Basel, CH)

Speakers Florian Pommerening (Universität Basel, CH) and Malte Helmert (Universität Basel, CH)

The aim of this session was to understand LP-based cost-partitioning from an Operations Research perspective and to identify ways of improving the technique with common OR wisdom.

The core idea of cost partitioning in planning (using an additive cost function) is to distribute the action costs among several copies of the planning instance so that the sum of lower bounds on solution costs still gives a lower bound for the original problem. An optimal cost partitioning can be computed with linear programming. Florian Pommerening showed the structure of the LPs in a primal and a dual formulation. These show that reasoning about plans in the operator counting framework can also be cast as reasoning about operator costs (if allowing negative costs). The constraints in the operator counting framework have a declarative nature, describing necessary properties of all plans. We briefly spoke about how these constraints can be derived, which lead to the question whether it would be possible to discover landmarks with constraint optimization techniques and whether there is an analogy between landmarks and backbones. Also, is it possible to use linear approximations of heuristics that cannot directly be expressed as an LP?

We also wondered about the role of negative costs. Why are they possible, why are they necessary? Negative cost are used for extracting value of aspects that another relaxation ignores. From an intuitive point of view this seems to be similar to amortized cost analysis.

It seems that Lagrangian decomposition could improve the heuristic estimates. Would it also be possible to identify independent macro actions to get a faster computation and better values with flow constraints?

How does all this relate to things that go on in MIP/CP? How large is the gap to the (unrelaxed) MIP values? For landmark constraints, in the cases where we can solve the IP, the gap is often 0. It is also low for other heuristics that were tried.

Potential heuristic avoid re-computation for every state. To avoid an extreme point (as one gets it by simplex), one can use the interior point method. Writing out the properties of the splitting might help.

We also discussed the questions whether the actual partitioning is important or only the value, and what other information can we derive apart from lower bounds.

Some sets of constraints formulate shortest-path problems. How exactly do these look like? The combination is a minimax problem (maximizing the sum of minimal cost paths), formulated in one LP. To see this as a MIP we need to consider its dual.

Overall, we spend a significant amount of time discussing what we were actually talking about because the two communities do not share the same notion for highly related concepts. There seem to be close relationships to some MIP techniques but more detailed discussion is needed to match them. The most promising direction seems to be exploiting substructures and the cost of these substructures.

Although we did not find final answers to many of the questions, the session identified many interesting connecting factors for future research.

4.14 Planning, Optimization, and Machine Learning

Scott Sanner (University of Toronto, CA)

Speakers Serdar Kadioglu (Fidelity Investments – Boston, US), Michele Lombardi (University of Bologna, IT), Christian Muise (IBM TJ Watson Research Center – Cambridge, US), and Scott Sanner (University of Toronto, CA)

Michele Lombardi began the session with a discussion of how to optimize traffic light placement. He proposed learning a model $f(x)$ from data and then embedding the constraint $y = f(x)$ in the optimization model. The key question though is how to embed the constraint $y = f(x)$ in optimization models, when $f(x)$ is not in a convenient form such as a linear function but rather a complex empirically learned model like a neural network.

Michele mentioned various solutions to this problem depending on the type of optimization model. For the case of Constraint Programming, Michele suggested that intervals could be propagated through the neural net in both directions, or alternately a Lagrangian relaxation could be used to add the neural network constraints in the optimization objective [1].

However, if one is only verifying a property (constraint satisfaction) of the neural network for all inputs and the neural network uses only linear and rectified linear unit (ReLU) activation functions, one can use a modification of the simplex algorithm for ReLU activations termed ReLUplex [2]. There is also related work on verifying properties of neural networks by finding counterexamples via SMT [3].

Beyond embedding neural networks in optimization models, Michele mentioned that decision trees are fairly easily directly embedded in a variety of models.

Some final discussion centered on the problem of the "optimizer's curse" when embedding learned models in optimization models, namely the problem that if the learned model makes significant errors (often in input regions where the data for learning is sparse), the optimizer will exploit these errors (e.g., large extrapolations that lead to large objective values when maximizing). That is, when using the learned model for prediction, one can expect the prediction errors to be randomly distributed, but due to the optimizer's curse, the optimizer will zero in on precisely the prediction errors that lead to erroneous, but favorable objective values that are practically unachievable.

Scott Sanner presented next on the topic of planning in learned models. He started with an example of using neural networks to learn a deterministic transition model from data. If one then wants to plan in this model using a Mixed Integer Linear Program (MILP) with neural network constraints or objective and the neural network is trained with rectified linear unit (ReLU) activation functions, the neural network definition can be encoded directly as MILP constraints with additional redundant constraints to strengthen the LP relaxation [4]. However, Scott also pointed out that if we can learn neural networks with rectified linear unit (ReLU) activation functions then freezing the weights and optimizing the network activations themselves is a symmetric problem to learning, and furthermore the optimization problem for control is often only constrained with simple action bound constraints. Hence Scott suggested that Tensorflow with projected gradient descent (to project all action values to their nearest bound) could be used as the optimizer of actions for such control problems. He also remarked that similarly to the Tensorflow planning model of Wu et al. [5] using RMSProp as the optimizer with deep net learned transition models performed very well in these highly non-convex control optimization problems.

Finally, Serdar Kadioglu presented on the topic of learning for SAT solver selection. He started with the question of what the best solver for every instance is, and whether it always is the same. The answer was that the best solver per instance is almost never the best solver overall! In this case he proposed the problem of how to learn which solver to use on each instance. To address this problem, he suggested following standard methodology in machine learning to derive features of problem instances followed by a PCA projection of these features. He then suggested classifying or clustering instances according to their features should yield classifications or clusters of instances with similar solver behavior. Further, Michele suggested that beyond simple instance-specific selection of problem solvers, one could embed learning into different stages of a backtracking solver as done by Di Liberto et al. [6].

Serdar then went on to discuss machine learning for optimization and specifically for selecting rules. An example of rule learning was given for the purpose of scheduling jobs on a single machine. For example, Serdar mentioned that one could use learned rules to select different scheduling methods on the first job and remaining jobs.

Christian Muise spoke last to follow up on the first topic from Serdar's talk. Christian discussed the issue that occurs when no features are available in branching optimizers (such as for SAT). Chris Beck and colleagues worked on learning search rules and techniques that were dependent on the search depth with some very positive results [7].

References

1 Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. *Neuron constraints to model complex real-world problems.* In Proceedings of the Seventeenth International Conference on Principles and Practice of Constraint Programming, pp. 115–129, 2011.

2 Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer. *Reluplex: An efficient SMT solver for verifying deep neural networks.* In Proceedings of the Twenty-Ninth International Conference on Computer Aided Verification, pp. 97–117, 2017.

3 Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. *Safety verification of deep neural networks.* In Proceedings of the Twenty-Ninth International Conference on Computer Aided Verification, pp. 3–29, 2017.

4 Buser Say, Ga Wu, Yu Qing Zhou, and Scott Sanner. *Nonlinear hybrid planning with deep net learned transition models and mixed-integer linear programming*. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, pp. 750–756, 2017.
5 Ga Wu, Buser Say, and Scott Sanner. *Scalable Planning with Tensorflow for Hybrid Nonlinear Domains*. In Advances in Neural Information Processing Systems 30, pp. 6276–6286, 2017.
6 Giovanni Di Liberto, Serdar Kadioglu, Kevin Leo, and Yuri Malitsky. *DASH: Dynamic approach for switching heuristics*. European Journal of Operational Research 248(3), pp. 943–953, 2016.
7 Tom Carchrae and J. Christopher Beck. *Applying Machine Learning to Low-Knowledge Control of Optimization Algorithms*. Journal of Computational Intelligence 21(4), pp. 372–387, 2005.

4.14.1 Talk: Empirical Model Learning

Michele Lombardi

Joint work of Michele Lombardi, Andrea Micheli, Alessio Bonfietti, Andrea Bartolini, Tias Guns, Pascal Van Hentenryck, Michela Milano
Main reference Michele Lombardi, Michela Milano, Andrea Bartolini: "Empirical decision model learning", Artif. Intell., Vol. 244, pp. 343–367, 2017.
URL http://dx.doi.org/10.1016/j.artint.2016.01.005

This talk provided an overview of Empirical Model Learning (EML), a method to tackle optimization problems defined over complex systems by extracting an approximate model via Machine Learning, and then embedding it into a classical combinatorial optimization model. The talk focused on the case of embedding Neural Networks and Decision Trees, respectively in Constraint Programming and SAT Modulo Theories. As a case study, we considered a workload allocation problem on a many-core CPU, for which a thermal model cannot be compactly expressed in a declarative fashion. The talk presented the basic ideas of EML, and highlighted open questions.

4.15 Multiagent Planning and Optimization

Matthijs Spaan (TU Delft, NL)

Speakers Roman Bartak (Charles University – Prague, CZ), Adi Botea (IBM Research – Dublin, IE), and Mathijs de Weerdt (TU Delft, NL)

During this session, we discussed particular aspects of planning and optimization that arise in the context of multiagent systems.

First, Mathijs de Weerdt spoke about multi-party optimization, which refers to the problem of optimizing the decision making of multiple stakeholders whose objectives are not aligned. For instance, we can consider the interactions of multiple companies or different departments within a single company. Striving for economic efficiency either leads to mergers of companies or to optimization across the boundaries of organizations. Three specific challenges were introduced. First, it is important to deal with preferences, which relates to social choice theory. A way forward could be to study voting or aggregation rules in the context of optimization. Second, participants might demonstrate strategic behaviour, which can be addressed by using Groves mechanisms. Third, it is not yet clear how to deal with incomplete information and local autonomy. Here, decomposition techniques could be promising.

Next, Adi Botea presented recent work on multiagent path planning for strongly biconnected directed graphs. In this problem, a set of agents have to move from an initial configuration to a goal configuration. For the particular type of graphs, an open ear decomposition is used. If the graph satisfies certain conditions, then every instance with at least two blanks has a solution. For the latter case, a complete algorithm is presented. Future work is to generalize to other classes of directed graphs and to explore optimal solving with an optimization-based approach.

Finally, Roman Bartak spoke about modeling and solving the multiagent path finding problem in the Picat language. The problem can be solved using state-space or conflict-based search techniques, but the talk focused on compilation approaches. The abstract model is based on a layered graph describing the agent positions at each time step and the objective function can be either the makespan or the sum of costs (minimize the sum of end times). The problem can be represented in the Picat language, which allows you to solve it using SAT, MIP or CP. Picat makes it easy to swap solvers or adapt the model and for this problem, the efficiency is comparable to the state of the art.

4.15.1 Talk: Multi-Party Optimization

Mathijs de Weerdt (TU Delft, NL)

This talk introduced and motivated the importance of optimization problems across organizational boundaries. This brings interesting new challenges for optimization. A few known optimization techniques (like decomposition) that may be useful in this context were indicated, and some relevant concepts from game theory highlighted.

References

1. Frits de Nijs, Matthijs T. J. Spaan, and Mathijs de Weerdt. *Best-Response Planning of Thermostatically Controlled Loads under Power Constraints.* In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. 2015.
2. Frits de Nijs, Erwin Walraven, Mathijs de Weerdt, and Matthijs T. J. Spaan. *Bounding the Probability of Resource Constraint Violations in Multi-Agent MDPs.* In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence. 2017.
3. Mathijs de Weerdt. *Plan Merging in Multi-Agent Systems.* PhD Thesis, Delft University of Technology, Delft, The Netherlands. 2003.
4. Mathijs de Weerdt, Sebastian Stein, Enrico H. Gerding, Valentin Robu, and Nicholas R. Jennings. *Intention-Aware Routing of Electric Vehicles.* IEEE Transactions on Intelligent Transportation Systems, 17(5) pp.1472–1482, 2016.
5. Joris Scharpff, Matthijs T. J. Spaan, Leentje Volker, and Mathijs de Weerdt. *Planning under Uncertainty for Coordinating Infrastructural Maintenance.* In Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling, pp. 425–433, 2013.

4.15.2 Talk: Multi-agent path planning on strongly biconnected digraphs

Adi Botea (IBM Research – Dublin, IE)

License Ⓒ Creative Commons BY 3.0 Unported license
© Adi Botea
Joint work of Adi Botea, Davide Bonusi, Pavel Surynek
Main reference Adi Botea, Pavel Surynek: "Multi-Agent Path Finding on Strongly Biconnected Digraphs", in Proc. of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA., pp. 2024–2030, AAAI Press, 2015.
URL http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9653

In a common formulation of the problem of multi-agent path planning, a set of mobile agents share an environment represented as a graph. The time is discretized. At a given time, each agent is located at one graph vertex, under the constraint that two or more agents cannot share the same vertex at the same time. An agent located at a vertex a at the time t can move to an adjacent vertex b at time the $t + 1$, provided that no other agent will be at the vertex b at the time $t + 1$. An additional common condition is that the vertex b is free at time t as well. Several agents can move in parallel at a given time. The task is to compute a set of moves that will take the agents from their initial configuration to a goal configuration.

Many sub-optimal, polynomial-time algorithms are designed for undirected graphs, where the movement is allowed in both directions along an edge. Yet, directional constraints (e.g., allowing moves only along one direction of an edge) are relevant in several domains, including games and asymmetric communication networks.

We presented an approach to multi-agent path planning on strongly biconnected directed graphs. A directed graph is strongly biconnected if: it is strongly connected; and the undirected digraph obtained by ignoring the directions of the edges is biconnected.

In our work, we show that all instances with at least two unoccupied positions have a solution, except for a particular, degenerate sub-class where the graph has a cyclic shape. Our algorithm, called diBOX, runs in polynomial time, computes suboptimal solutions and is complete for instances on strongly biconnected digraphs with at least two unoccupied positions.

To our knowledge, our work is the first study of multi-agent path finding focused on directed graphs.

4.15.3 Talk: Modeling and Solving the Multi-Agent Pathfinding Problem in Picat

Roman Bartak (Charles University – Prague, CZ)

License Ⓒ Creative Commons BY 3.0 Unported license
© Roman Bartak
Joint work of Roman Barták, Neng-Fa Zhou, Roni Stern, Eli Boyarski, Pavel Surynek
Main reference Roman Barták, Neng-Fa Zhou, Roni Stern, Eli Boyarski, Pavel Surynek: "Modeling and Solving the Multi-agent Pathfinding Problem in Picat", in Proc. of the 29th IEEE Int'l Conference on Tools with Artificial Intelligence, ICTAI 2017, Boston, MA, USA, November 6-8, 2017, pp. 959–966, IEEE Computer Society, 2017.
URL http://dx.doi.org/10.1109/ICTAI.2017.00147

The multi-agent pathfinding (MAPF) problem has attracted considerable attention because of its relation to practical applications. We presented a constraint-based declarative model for MAPF, together with its implementation in Picat, a logic-based programming language. We showed experimentally that our Picat-based implementation is highly competitive and sometimes outperforms previous approaches. Importantly, the proposed Picat implementation is very versatile. We demonstrated this by showing how it can be easily adapted to optimize different MAPF objectives, such as minimizing makespan or minimizing the sum of costs, and for a range of MAPF variants. Moreover, a Picat-based model can be automatically compiled to several general-purpose solvers such as SAT solvers and Mixed Integer Programming solvers (MIP). This is particularly important for MAPF because some MAPF variants are solved more efficiently when compiled to SAT while other variants are solved more efficiently when compiled to MIP. We analyzed these differences and the impact of different declarative models and encodings on empirical performance.

4.16 Exploiting Substructures

Charlotte Truchet (University of Nantes, FR)

Speakers Malte Helmert (Universität Basel, CH), and Gilles Pesant (Ècole Polytechnique – Montréal, CA), Domenico Salvagnin (University of Padova, IT)

This session had three presentations: Gilles Pesant on "Counting in individual substructures", Domenico Salvagnin on "Global structures in MIP" and Malte Helmert.

The first presentation introduced different techniques for solution counting on global constraints, which represent a specific substructure of a combinatorial problem: based on a compact representation (e.g. regular constraint), based on sampling (e.g. all different constraint), based on domain relaxation (e.g. knapsack constraint) or based on theoretical results. There were several questions, with a discussion focused on ways to deal with the #P complexity of these problems in general.

The second presentation reviewed different ways of handling global structures in MIP problems, where the flat model is based on very simple primitives (linear and integrity constraints). The need for higher level representations was identified, either by extending the language expressivity (as in CP) or by doing reverse engineering on a flat model. Several global substructures like LP relaxation, symmetry groups, and graphs were presented. The audience had many questions and the discussions focused on automatic modeling, reified constraints, efficiency of global constraints, and the question whether the user should chose the substructures or they should be discovered by an educated guess by the solver.

The third presentation detailed an example in planning where global structures needed to be identified, based on two state variables and an implicit transition graph that is induced by domain transition graphs for each variable. What can be done on this representation was first presented by the speaker then discussed with the audience: apart from solving the the problem, properties of solutions can be extracted (e.g. mandatory actions); bounds on the number of times an operator is used in a solution can be used to extract information from the domain transition graphs; marginals of the actions in one of the automata can be computed; and landmarks can be discovered automatically.

4.16.1 Talk: Counting in Individual Substructures

Gilles Pesant (Ècole Polytechnique – Montréal, CA)

Main reference Gilles Pesant: "Getting More Out of the Exposed Structure in Constraint Programming Models of Combinatorial Problems", in Proc. of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA., pp. 4846–4851, AAAI Press, 2017.
URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14982

To solve combinatorial problems, Constraint Programming builds high-level models that expose much of the structure of the problem. The distinctive driving force of Constraint Programming has been this direct access to problem structure. This has been key to the design of powerful inference algorithms and branching heuristics. In particular, the ability to evaluate the number of solutions in such structures helps reduce the combinatorial search space, guide its exploration, but also has applications in model counting, uniform sampling, structure elicitation, and possibly planning.

This talk presented some of the algorithmic techniques used so far in order to perform solution counting on several important combinatorial substructures.

4.16.2 Talk: Global Structures in MIP

Domenico Salvagnin (University of Padova, IT)

We surveyed the global structures that are automatically constructed and exploited by MIP solvers to improve performance.

Participants

- Roman Bartak
Charles University – Prague, CZ
- J. Christopher Beck
University of Toronto, CA
- Adi Botea
IBM Research – Dublin, IE
- Christina N. Burt
Satalia – London, GB
- Hadrien Cambazard
Grenoble INP, FR
- Michael Cashmore
King's College London, GB
- Alessandro Cimatti
Bruno Kessler Foundation – Trento, IT
- Mathijs de Weerdt
TU Delft, NL
- Jeremy D. Frank
NASA – Moffett Field, US
- Patrik Haslum
Australian National University, AU
- Emmanuel Hebrard
LAAS – Toulouse, FR
- Malte Helmert
Universität Basel, CH
- John N. Hooker
Carnegie Mellon University – Pittsburgh, US
- Serdar Kadioglu
Fidelity Investments – Boston, US
- Michael Katz
IBM TJ Watson Research Center – Yorktown Heights, US
- Thorsten Koch
Konrad-Zuse-Zentrum – Berlin, DE
- Sven Koenig
USC – Los Angeles, US
- Michele Lombardi
University of Bologna, IT
- Daniele Magazzeni
King's College London, GB
- Andrea Micheli
Bruno Kessler Foundation – Trento, IT
- Christian Muise
IBM TJ Watson Research Center – Cambridge, US
- Eva Onaindia
Technical University of Valencia, ES
- Gilles Pesant
Ècole Polytechnique – Montréal, CA
- Chiara Piacentini
University of Toronto, CA
- Nicola Policella
Solenix – Darmstadt, DE
- Florian Pommerening
Universität Basel, CH
- Mark Roberts
Naval Research – Washington, US
- Gabriele Röger
Universität Basel, CH
- Louis-Martin Rousseau
Polytechnique Montreal, CA
- Hana Rudová
Masaryk University – Brno, CZ
- Domenico Salvagnin
University of Padova, IT
- Scott Sanner
University of Toronto, CA
- Pierre Schaus
UC Louvain, BE
- Matthijs Spaan
TU Delft, NL
- Charlotte Truchet
University of Nantes, FR
- Willem-Jan Van Hoeve
Carnegie Mellon University – Pittsburgh, US
- Parisa Zehtabi
King's College London, GB