An Automata-Based Constraint Programming Framework for Optimal Classical Planning

Damien Van Meerbeeck
Linköping University, Sweden

Arnaud Lequen
Linköping University, Sweden

Gilles Pesant
Polytechnique Montréal, Canada

Jendrik Seipp
Linköping University, Sweden

Abstract

2012 ACM Subject Classification Computing methodologies → Planning and scheduling

Keywords and phrases Optimal Classical Planning, Landmark Constraints, Automata

Digital Object Identifier 10.4230/LIPIcs.CP.2026.22

1 Introduction

Optimal classical planning is the challenge of finding one of the shortest sequences of actions (called a *plan*) that transforms a given initial situation into one that satisfies a given goal description in a deterministic environment [7]. While the current strongest optimal classical planners are based on state-space search with admissible heuristics [8], we instead solve planning tasks using Constraint Programming (CP) which provides a flexible language for combining declarative information and global constraints [14].

Our approach builds on recent automata-based representations of planning tasks [2]. Starting from an input planning task, we automatically derive a factored set of deterministic finite automata and use them as the backbone of a CP model. Finding a plan is then reduced to finding a word accepted by all automata through Regular [12] constraints. With this framework in place, we can then extend the model with constraints from the planning literature that provide declarative information about the task. Concretely, we focus on two families of constraints from the operator-counting framework [13]: *landmark constraints* [9, 5] and *net change constraints* [4].

2 Pipeline and CP Model Overview

Figure 1 illustrates our pipeline. First, the planning task is translated from PDDL [10] to SAS^+ [3], and preprocessed [1]. Then, the operator-counting constraints are extracted as well as the set of automata, one for each state variable of the SAS^+. Next, we iteratively build and solve the CP model until a plan is found, starting from a lower bound given by an admissible heuristic on the initial state.

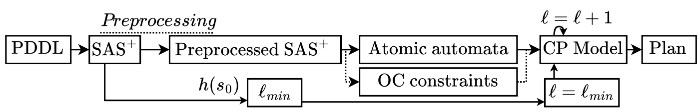

Figure 1 Pipeline of our approach. Dotted lines depict optional components of the pipeline.

32nd International Conference on Principles and Practice of Constraint Programming (CP 2026).
Editor: Nicolas Beldiceanu; Article No. 22; pp. 22:1–22:3
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

	DataNetwork (20)	Logistics (63)	Miconic (150)	ParcPrinter (50)	Pegsol (50)	Rovers (40)	Satellite (36)	Scanalyzer (50)	Tetris (17)	Trucks (30)	Woodworking (50)	Zenotravel (20)	*Other* (1210)	Total (1786)
GP-WCSP	11	13	32	28	22	**7**	5	7	3	3	32	7	231	401
Base$^{\text{IPG}}$	**17**	13	58	30	38	4	7	23	5	5	35	9	446	690
LMC$^{\text{IPG}}$	15	**28**	**102**	34	38	6	**9**	**25**	5	5	**50**	**11**	443	**771**
NCC$^{\text{IPG}}$	12	13	38	**50**	**42**	5	5	11	**9**	**8**	45	8	323	569

Table 1 Number of solved tasks per domain for GP-WCSP and our models Base$^{\text{IPG}}$, LMC$^{\text{IPG}}$ and NCC$^{\text{IPG}}$. Domain sizes are shown in parentheses. A domain is detailed in the table if the absolute difference between LMC$^{\text{IPG}}$ and Base$^{\text{IPG}}$ is at least 2 or if the difference between NCC$^{\text{IPG}}$ and Base$^{\text{IPG}}$ is more than 2. The remaining domains are compiled in the *Other* column.

For a given plan length, we start the construction of our base model by defining an array of variables for the actions of the plan. Then, we reduce each automaton by merging together its transitions that are equivalent. After that, for each automaton, we use a Regular constraint to enforce acceptance by the reduced automaton of an additional sequence of actions linked to the main plan with Table according to the reduction. Finally, we use the operator-counting constraints to enrich the model with additional variables and constraints. After an iterative refinement process over the variables counting the occurrences of each action, reducing the total number of variables, we add the resulting constraints to the model.

3 Experimental Evaluation and Results

We evaluate our approach on all IPC optimal track benchmarks (modified with unit action costs) using Intel Xeon Gold 6130 processors with a 30-minute cutoff and a memory limit of 8 GiB. We used Scorpion [15] for all planning-related components of the pipeline, and CP-SAT [11] to build and solve our CP models.

Using the best empirical preprocessing, we compare the base model Base$^{\text{IPG}}$ against models enriched with landmark constraints LMC$^{\text{IPG}}$ and net change constraints NCC$^{\text{IPG}}$. We also compare our models against GP-WCSP [6] adapted to use CP-SAT.

Table 1 shows that our base model Base$^{\text{IPG}}$ outperforms GP-WCSP in almost all domains suggesting that the automata-based encoding is a strong baseline for optimal planning. Table 1 also shows that the landmark constraints are the most successful extension overall and improve the total number of solved tasks over Base$^{\text{IPG}}$, which suggests that they provide valuable information to the solver. Net change constraints also improve performance in a reduced number of domains, but they are less effective overall and can even hurt performance. Indeed, these constraints tend to add more computational overhead than landmark constraints, and their effectiveness heavily depends on the structure of the task.

4 Conclusion

We presented an end-to-end, automated CP framework for optimal classical planning, which leverages an automata-based representation of planning tasks and outperforms the state-of-the-art CP-based planner. We showed that our framework is extensible with redundant operator-counting constraints derived from the planning literature and that the resulting models can improve solver performance in a number of domains.

References

1. Vidal Alcázar and Álvaro Torralba. A reminder about the importance of computing and exploiting invariants in planning. In Ronen Brafman, Carmel Domshlak, Patrik Haslum, and Shlomo Zilberstein, editors, *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS 2015)*, pages 2–6. AAAI Press, 2015.
2. Behrouz Babaki, Gilles Pesant, and Claude-Guy Quimper. Solving classical AI planning problems using planning-independent CP modeling and search. In Daniel Harabor and Mauro Vallati, editors, *Proceedings of the 13th Annual Symposium on Combinatorial Search (SoCS 2020)*, pages 2–10. AAAI Press, 2020.
3. Christer Bäckström and Bernhard Nebel. Complexity results for SAS$^+$ planning. *Computational Intelligence*, 11(4):625–655, 1995.
4. Blai Bonet. An admissible heuristic for SAS$^+$ planning obtained from the state equation. In Francesca Rossi, editor, *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, pages 2268–2274. AAAI Press, 2013.
5. Blai Bonet and Malte Helmert. Strengthening landmark heuristics via hitting sets. In Helder Coelho, Rudi Studer, and Michael Wooldridge, editors, *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, pages 329–334. IOS Press, 2010.
6. Martin C. Cooper, Marie de Roquemaurel, and Pierre Régnier. A weighted CSP approach to cost-optimal planning. *AI Communications*, 24(1):1–29, 2011.
7. Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, 2004.
8. Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: What's the difference anyway? In Alfonso Gerevini, Adele Howe, Amedeo Cesta, and Ioannis Refanidis, editors, *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, pages 162–169. AAAI Press, 2009.
9. Erez Karpas and Carmel Domshlak. Cost-optimal planning with landmarks. In Craig Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 1728–1733. AAAI Press, 2009.
10. Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL – The Planning Domain Definition Language – Version 1.2. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, Yale University, 1998.
11. Laurent Perron and Frédéric Didier. CP-SAT. `https://developers.google.com/optimization/cp/cp_solver/`, 2025. Version v9.12, Google.
12. Gilles Pesant. A regular language membership constraint for finite sequences of variables. In Mark Wallace, editor, *Proceedings of the Tenth International Conference on Principles and Practice of Constraint Programming (CP 2004)*, volume 3258 of *Lecture Notes in Computer Science*, pages 482–495. Springer-Verlag, 2004.
13. Florian Pommerening, Gabriele Röger, Malte Helmert, and Blai Bonet. LP-based heuristics for cost-optimal planning. In Steve Chien, Alan Fern, Wheeler Ruml, and Minh Do, editors, *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2014)*, pages 226–234. AAAI Press, 2014.
14. Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 2006.
15. Jendrik Seipp, Thomas Keller, and Malte Helmert. Saturated cost partitioning for optimal classical planning. *Journal of Artificial Intelligence Research*, 67:129–167, 2020.