End-to-End Classical Planning using CP and Belief Propagation

Damien Van Meerbeeck[1], Gilles Pesant[2], and Jendrik Seipp[1]

[1] Linköping University, Sweden
damien.van.meerbeeck@liu.se, jendrik.seipp@liu.se
[2] Polytechnique Montréal, Canada
gilles.pesant@polymtl.ca

Classical planning is one of the original core AI research areas. It is the challenge of finding a sequence of actions that transforms a given initial situation into one that satisfies a given goal description [7]. In *optimal* classical planning, the plan must minimize the sum of its action costs. Most of today's strongest planners are based on state-space search with goal distance estimators, called *heuristics*. The predominant algorithm is A* [9] for optimal planning [6, 16]. In this work, we leave the beaten path of state-space search planning and instead solve planning tasks with constraint programming (CP). We do so firstly in order to obtain a planning system that allows users to input their plan constraints in a declarative way. Our second motivation is to use the power of CPBP solvers [14] to obtain a system that scales better to large planning tasks.

The literature features several papers using constraint programming for planning [17, 5, 18, 4, 8]. Following the seminal work of Kautz and Selman [11], they typically cast the problem as a succession of fixed-length planning tasks but differ both in the choice of constraints and in the branching strategy for the CP solver. Babaki, Pesant, and Quimper [2] build on the work of Zanarini, Pesant, and Milano [18] to manually derive CP models for three planning domains. With each planning object they associate an automaton describing how actions affect its state, enforced using a REGULAR or COSTREGULAR constraint. The main novelty is that they compare several branching strategies and find that *maxSD*, an instance of counting-based search [15], performs vastly better than the other strategies. We build our work on their approach. However, instead of manually designing the automata and consequently the CP model, we aim to compute them automatically from the input planning tasks thereby providing an end-to-end approach. We also stay close in spirit to the idea behind their successful branching heuristic by exploiting the marginal probabilities provided by the CPBP framework [14]. So far, no CP-based planner has reached the performance of planners based on state-space search, which is something we aim to change with this line of work.

Our Approach So Far

On a high level, our pipeline automatically transforms a given input task into a set of automata which MiniCPBP [13] uses to model and solve the task. In more detail, we proceed as follows. First, we use the translator component of the Fast Downward planning system [10] to translate the input planning task, formulated in the first-order planning domain definition language (PDDL) [12], into a ground finite-domain SAS$^+$ task [3]. Then we construct an automaton for each variable v in the SAS$^+$ task by projecting the task onto v. The resulting transition system forms a deterministic finite-state automaton that describes how the actions in the task influence the value of v.

		NoGroup	Group	Group+Prune	Manual
Miconic (150)	solved opt.	38	42	41	49
	out of memory	60	28	28	0
	out of time	52	80	81	101
Scanalyzer (41)	solved opt. (solved)	5 (11)	5 (15)	5 (21)	33 (33)
	out of memory	18	18	12	0
	out of time	18	18	24	8

Table 1: Per-domain aggregated planner outcomes: tasks solved optimally, tasks solved (possibly suboptimally), runs that hit the 8 GB memory limit and runs that exceed 30 minutes of runtime.

Without further adjustments, the CP models obtained through this baseline approach (which we call NoGroup) will not scale to large planning tasks. One issue is the large number of automata produced. We could combine several single-variable automata by computing their *product automaton* but deciding which to combine is not trivial and we have not attempted it yet. Another issue is that the number of *ground* actions in large SAS^+ planning tasks can be in the tens of thousands. Since our encoding uses one variable per plan step, each choosing among all ground actions, we obtain very large variable domains that are out of reach for current CP solvers. To address this, we use two methods that can drastically reduce the domain sizes of the CP variables. First, we group actions within each automaton that incur only parallel transitions (Group). Second, we prune irrelevant actions, i.e., those that can never be part of an optimal plan, with an off-the-shelve preprocessing tool [1] (Group+Prune).

Preliminary Experiments and Discussion

We use MiniCPBP with belief propagation, the maximum marginal branching heuristic on the actions of the plan, and limited discrepancy search. We fix the plan length to the length of an optimal plan, which we precomputed with the Scorpion planner [16]. (Obviously, in the final system we will need to iterate over several plan lengths.) We consider three common planning domains: Miconic, Scanalyzer, and Floortile. Since none of the algorithms solves any task in Floortile, we focus on results for the other two domains here.

Table 1 compares our automated approaches to the manually designed CP models from Babaki et al. [2] (Manual). We observe that our action-reduction efforts have a positive impact on the number of tasks solved but fall short of reaching the performance of the manual approach. Comparing the models used by Group+Prune to those of Manual, we observe on average a factor 14 and 230 increase in the number of actions for Miconic and Scanalyzer, respectively. This could explain why our approaches run out of memory for some tasks, in contrast to Manual. We also observe a factor 2 increase in the number of automata, hinting at models with less of a global view.

This initial investigation already points to areas for improvement in the generation of the automata encoding. We also plan to study CP variable-selection heuristics that do not build the plan sequentially, in contrast to state-space search planners.

Acknowledgments

This work was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation and by TAILOR, a project funded by the EU Horizon 2020 research and innovation programme under grant agreement no. 952215. The computations were enabled by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreements no. 2022-06725.

References

1. Alcázar, V., Torralba, Á.: A reminder about the importance of computing and exploiting invariants in planning. In: Proc. ICAPS 2015. pp. 2–6 (2015)
2. Babaki, B., Pesant, G., Quimper, C.: Solving classical AI planning problems using planning-independent CP modeling and search. In: Proc. SoCS 2020. pp. 2–10 (2020)
3. Bäckström, C., Nebel, B.: Complexity results for SAS^+ planning. Computational Intelligence **11**(4), 625–655 (1995)
4. Barták, R., Toropila, D.: Reformulating constraint models for classical planning. In: Proc. FLAIRS 2008. pp. 525–530 (2008)
5. Do, M.B., Kambhampati, S.: Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP. AIJ **132**(2), 151–182 (2001)
6. Franco, S., Torralba, Á., Lelis, L.H.S., Barley, M.: On creating complementary pattern databases. In: Proc. IJCAI 2017. pp. 4302–4309 (2017)
7. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: Theory and Practice. Morgan Kaufmann (2004)
8. Ghanbari Ghooshchi, N., Namazi, M., Newton, M., Sattar, A.: Encoding domain transitions for constraint-based planning. JAIR **58**, 905–966 (2017)
9. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics **4**(2), 100–107 (1968)
10. Helmert, M.: The Fast Downward planning system. JAIR **26**, 191–246 (2006)
11. Kautz, H., Selman, B.: Planning as satisfiability. In: Proc. ECAI 1992. pp. 359–363 (1992)
12. McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., Wilkins, D.: PDDL – The Planning Domain Definition Language – Version 1.2. Tech. Rep. CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, Yale University (1998)
13. Pesant, G.: The MiniCPBP Solver, `https://github.com/PesantGilles/MiniCPBP`
14. Pesant, G.: From support propagation to belief propagation in constraint programming. JAIR **66**, 123–150 (2019)
15. Pesant, G., Quimper, C.G., Zanarini, A.: Counting-based search: Branching heuristics for constraint satisfaction problems. JAIR **43**, 173–210 (2012)
16. Seipp, J., Keller, T., Helmert, M.: Saturated cost partitioning for optimal classical planning. JAIR **67**, 129–167 (2020)
17. van Beek, P., Chen, X.: CPlan: A constraint programming approach to planning. In: Proc. AAAI 1999. pp. 585–590 (1999)
18. Zanarini, A., Pesant, G., Milano, M.: Planning with soft regular constraints. In: ICAPS 2006 Workshop on Preferences and Soft Constraints in Planning. pp. 73–78 (2006)