

# Over-subscription planning with Boolean Optimization: An assessment of state-of-the-art solutions

Marco Maratea<sup>1</sup> and Luca Pulina<sup>2</sup>

<sup>1</sup> DIST, University of Genova, Viale F. Causa 15, Genova, Italy.  
marco@dist.unige.it

<sup>2</sup> DEIS, University of Sassari, Piazza Università 11, Sassari, Italy.  
lpulina@uniss.it

**Abstract.** In this work we present an assessment of state-of-the-art Boolean optimization solvers from different AI communities on over-subscription planning problems. The goal of the empirical analysis here presented is to assess the current respective performance of a wide variety of Boolean optimization solvers for solving such problems.

## 1 Introduction

Over-Subscription Planning Problems (OSPPs) [1, 2] are planning problems containing quantitative preferences expressed on goals in case not all the goals can be satisfied. In particular, a cost is associated to the violation of goals, and the aim is to find a plan whose metric maximizes the rewards of satisfied goals. OSPPs are thus suitable to model a wide set of practical applications – see, e.g., [2]. The increasing interest of the AI planning community on OSPPs is also witnessed by recent editions of the International Planning Competition (IPC), where all the “optimization” tracks of IPC’08 consider these problems. Furthermore, in IPC’08 also quantitative preferences expressed on action’s preconditions are taken into account for plan metrics.

Considering a fixed plan horizon, i.e. a *makespan*, a recently adopted approach used to deal with such problems is to reduce them to Boolean propositional problems with linear optimization functions [3, 4] – e.g., Max-SAT and Pseudo-Boolean (PB) problems. Both Max-SAT and PB are extensions of the well-known propositional satisfiability (SAT) problem. These formalisms allow the user to naturally reason with integers, which is one of the main limitation of SAT, instead of relying on complicated and/or space consuming encodings – see, e.g., [5].

In this paper we present an assessment of state-of-the-art systems coming from different scientific AI communities. We show the result of an experimental analysis involving all the best performing Max-SAT and PB solvers and other systems that, even if designed to mainly deal with other formalisms, can solve Boolean optimization problems. In particular, we also consider Answer Set Programming (ASP), Integer Programming (IP), Constraint Integer Programming (CIP), and Interval Constraints Propagation (ICP) systems. All the solvers are tested on domains comprised in both IPC’06 and IPC’08. Our results reveal that the ASP solver CLASP and the PB solver MINISAT+ are currently the overall best systems on these instances.

## 2 Instances and Solvers

For our analysis, we considered the domains OPENSTACKS, OPENSTACKS-IPC08, PATHWAYS, PEGSOL, STORAGE, TPP, and TRUCKS. They were used in past IPCs, in particular in the “SimplePreferences” track of IPC’06 and in the “netben-opt” track of IPC’08. In these domains, the plan metrics, in terms of quantitative preferences, are expressed on goals and/or on actions preconditions. In the following, we provide a short overview about the modeling of the problems of interest – details can be found in [3]. Considering a fixed makespan, benchmarks are reduced to Boolean optimization problems with different formalisms. This is done by using a modified version of the SATPLAN planner [6] on the STRIPS problems formulation at fixed makespan. Given that SATPLAN can only handle STRIPS domains, while IPC’06 domains are non-STRIPS, and some ADL [7] constructs are used, we have first adapted the non-STRIPS problems in the following way: (i) Preferences<sup>3</sup> on actions preconditions are expressed with two actions that do not contain preferences. For both actions, the related preference formula is treated as hard, further negated in the second. The second action also achieves a new dummy literal; and (ii) the goal preferences are imposed as preconditions of dummy actions, which achieve new dummy literals defining the new problem goals. The treatment of actions preferences is inspired by the ones used in [8]. The metrics of the planning problems are expressed with (linear) optimization function.

Concerning the instances generation, we have modified SATPLAN at each makespan of the SATPLAN’s approach, until the optimal. Thus, our compilation allows to find plans with optimal metrics at fixed makespan. Further, note that while literals related to goal preferences can be implicitly considered to hold only “at the end” modality [9] – i.e., at the final makespan – this is not the case for the ones related to preconditions that can, in general, hold at any makespan, unless we know that, instead, STRIPS actions can be only executed once (e.g., this is the case for well known real-world planning domain like blocks-world and logistics). The changes in SATPLAN were mainly related to the creation of formulas in the formats accepted by the various solvers employed instead of the DIMACS format for SAT formulas in CNF. To mention the more widely used input formats, our approach generates Weighted Partial Max-SAT problem – a further extension of Max-SAT. In particular, in the Weighted Partial Max-SAT problem, a positive integer weight is associated to each soft clause, and the goal is to satisfy all hard clauses and maximize the sum of weights associated to satisfied soft clauses.

In our experimental evaluation, we focus on the results obtained by the various solvers on the first satisfiable formula following the SATPLAN approach, augmented with optimization issues defined by the metric of the problem. Further, we consider the case where actions can be executed at most once. Globally, we will focus on the 71 instances that we could compile with the ADL2STRIPS tool, and then can be solved by at least one of the considered solvers. In particular, some PATHWAYS, TPP from #11 to #16, TRUCKS from #3 to #7, and OPENSTACKS #1 (as numbered in IPC’06) instances could

---

<sup>3</sup> We consider that at most one preference formula is expressed on the preconditions of an action: This is the case for all domains we consider in this paper. If this would not be the case, we should consider their power set.

Solver	Version	Formalism	Solver	Version	Formalism
AKMAXSAT	2010	Max-SAT	MINIMAXSAT	1.0	Max-SAT, PB
BSOLO	3.0.17	PB	MINISAT+	1.14	PB
CLASP	1.3.6	ASP	MSUNCORE	1.2	Max-SAT
CPLEX	12.0	IP	SAT4J	2.1.0	Max-SAT, PB
GLPPB	0.2	PB	SCIP	1.2.0	CIP
HYSAT	0.8.6	ICP	WBO	1.4	Max-SAT, PB
INCWMAXSATZ	1.2	Max-SAT	WMAXSATZ	2.5	Max-SAT

**Table 1.** Solvers involved in the evaluation. The table is structured as follows. Column “Solver” reports the name of the solver, while column “Version” indicates the version used in the experiments. For AKMAXSAT indicates the version submitted to the 2010 Max-SAT Competition. Finally, column “Formalism” reports the input formalisms accepted by the solver.

be compiled but not solved by any system. They thus provide challenging benchmarks for state-of-the-art solvers.

Table 1 summarizes the solvers involved in the evaluation. Looking at the table, we can see that the selected systems come from different scientific AI communities, namely ASP, CIP, ICP, IP, Max-SAT, and PB. Moreover, we can see that, in some cases, a solver is able to deal with problems expressed with different formalisms, e.g. MINIMAXSAT. Concerning PB and Max-SAT solvers, we selected the best solvers that have participated to Max-SAT and PB evaluations along the years [10, 11], with emphasis on the “Weighted Partial” and “OPT-SMALL-INT” categories, the last being part of PB evaluations, and where (i) there is no constraint with a sum of coefficients greater than  $2^{20}$  (20 bits), and (ii) the objective function is linear. CLASP [12] is the overall winner of the 2009 ASP Competition [13], CPLEX is a well-known linear arithmetic solver that can solve IP problems, while HYSAT [14] is the best solver based on ICP.

### 3 Experimental Analysis

The experiments reported in this section ran on 10 identical PCs equipped with a processor Intel Pentium IV running at 3.2GHz with 1GB of RAM, and running GNU Linux Debian 2.4.27-2. For each run, the CPU time limit was set to 900 CPU seconds and, in order to prevent memory swapping, we set a memory limit at 900MB.

Table 2 shows the results of our analysis for each single domain. Notice that we drop the domain OPENSTACKS, for which we report that no solver solved the single instance contained. Looking at Table 2 (top-left), domain OPENSTACKS-IPC08, we can see that only CLASP was able to deal with all domain instances, while MINIMAXSAT, MINISAT+, and WBO top to 1. The bad performance related to the remaining 10 solvers is mainly to ascribe to the instances size. Considering now domain PATHWAYS (Table 2, top-right), we report that no solver was able to solve all instances. The best solver is MINISAT+ that tops to 15 out of 20 instances, followed by CLASP. Also BSOLO and SAT4J solved the same amount of instances, but they spent one order of magnitude more of CPU time. Overall, 8 solvers were able to solve at least 50% of the instances, and we also report that all solvers solved at least 30% of the domain dataset. Looking yet at Table 2 (middle-left), domain PEGSOL, the reported results highlight how these instances seem to be easier for the solvers. All considered systems but 3 were able to

Domain	Solver	Solved		Domain	Solver	Solved	
		#	Time			#	Time
OPENSTACKS-IPC08 (2)	CLASP	2	623.81	PATHWAYS (20)	MINISAT+	15	21.00
	MINIMAXSAT	1	54.20		CLASP	15	24.79
	MINISAT+	1	62.58		BSOLO	15	127.24
	WBO	1	645.73		SAT4J	15	721.99
	AKMAXSAT	–	–		MINIMAXSAT	14	69.20
	BSOLO	–	–		CPLEX	14	943.86
	GLPPB	–	–		INCWMAXSATZ	13	499.80
	HYSAT	–	–		WMAXSATZ	12	1343.86
	INCWMAXSATZ	–	–		MSUNCORE	8	14.54
	MSUNCORE	–	–		SCIP	8	138.48
	SAT4J	–	–		AKMAXSAT	8	387.17
	SCIP	–	–		HYSAT	8	1537.65
WMAXSATZ	–	–	WBO	7	1.59		
CPLEX	–	–	GLPPB	6	389.46		
PEGSOL (28)	INCWMAXSATZ	28	0.35	STORAGE (7)	CPLEX	7	20.18
	MINISAT+	28	1.19		WMAXSATZ	7	36.62
	WMAXSATZ	28	1.23		MINISAT+	7	43.86
	MINIMAXSAT	28	1.35		INCWMAXSATZ	7	58.18
	BSOLO	28	1.53		CLASP	7	91.96
	CLASP	28	1.55		SCIP	7	197.90
	CPLEX	28	3.4		SAT4J	7	457.71
	AKMAXSAT	28	15.04		MINIMAXSAT	6	10.84
	SAT4J	28	68.64		AKMAXSAT	5	5.39
	SCIP	28	70.40		BSOLO	5	64.54
	HYSAT	28	207.70		WBO	5	101.66
	WBO	24	92.43		MSUNCORE	4	0.32
	MSUNCORE	21	694.43		HYSAT	4	56.86
	GLPPB	15	1817.76		GLPPB	2	1.17
TPP (10)	CLASP	10	106.2	TRUCKS (3)	MSUNCORE	2	18.67
	MINISAT+	10	124.97		WBO	2	23.04
	BSOLO	10	422.92		CLASP	2	48.46
	MINIMAXSAT	8	48.95		MINIMAXSAT	2	57.95
	WBO	8	168.15		MINISAT+	2	390.36
	SAT4J	8	424.54		BSOLO	1	119.69
	CPLEX	8	2042.1		SAT4J	1	359.17
	WMAXSATZ	7	687.19		AKMAXSAT	–	–
	MSUNCORE	4	0.34		CPLEX	–	–
	INCWMAXSATZ	4	1.32		GLPPB	–	–
	SCIP	4	15.21		HYSAT	–	–
	AKMAXSAT	4	51.56		INCWMAXSATZ	–	–
	HYSAT	4	63.45		SCIP	–	–
	GLPPB	–	–		WMAXSATZ	–	–

**Table 2.** Evaluation results by domain. We report the number of instances solved within the time limit (“#”) and the total CPU time (“Time”) spent on solved instances. Solvers are sorted according to the number of solved instances, and, in case of a tie, according to CPU time. A dash means that a solver did not solve any instance in the related domain.

solve all instances, and we can conjecture that these results are mainly due to the fact that such instances are composed of a small number of variables and constraints. Considering now domain STORAGE (Table 2, middle-right), we can see that 50% of solvers are able to solve all instances in the domain. Looking at the results, CPLEX is the solver having best performance, and it is faster than both MINISAT+ and CLASP, by a 2x and a 4x factor, respectively. Looking now at the results related to the domains at the bottom of Table 2, we can see that, concerning TPP domain (bottom-left), CLASP and MINISAT+ confirm their good performance. We also notice the performance of BSOLO in

Domain	Overall		Time	Hardness			Domain	Overall		Time	Hardness		
	N	#		EA	ME	MH		N	#		EA	ME	MH
OPENSTACKS-IPC08	2	2	623.81	–	1	1	PATHWAYS	20	15	17.77	6	9	–
PEGSOL	28	28	0.32	11	17	–	STORAGE	7	7	14.25	2	5	–
TPP	10	10	105.68	–	10	–	TRUCKS	3	2	18.67	–	2	–

**Table 3.** Classification of instances by domain. For each domain we report the name (column “Domain”), the total amount of instances in the domain, and the number of solved instances (group “Overall”, columns “N” and “#”, respectively). It follows column “Time”, which report the CPU time taken to solve the instances. Finally, group “Hardness” reports the total amount of easy, medium, and medium-hard instances (columns “EA”, “ME”, and “MH”, respectively).

this domain, that is the only other solver that solves all comprised instances. Six solvers are not able to solve 50% of the total amount of instances. Concluding the analysis of Table 2, about TRUCKS domain (bottom-right) we can first report that no solver solved all instances. We also can see that the best two solvers, MSUNCORE and WBO did not perform very well in the other domains. If we consider statistics related to the 2 solved instances, we report that their structure in terms of relationship between variables, soft, and hard constraints is quite different from instances in the other domains. We can conjecture that heuristics in MSUNCORE and WBO are effective in such cases.

Globally, we report that CLASP is the best solver, able to solve more than 90% of the dataset, and that MINISAT+ performance are very close. On the other hand, one solver only – namely GLPPB– was not able to solve 50% of the whole dataset, and 7 solvers were able to deal with at least 75% (53 instances) of the whole dataset. Notice that these 7 solvers were designed to solve 4 (out of 6 taken into account) different problem formalisms, namely ASP (CLASP), PB (MINISAT+ and BSOLO), Max-SAT (MINIMAXSAT, SAT4J, and WMAXSATZ), and IP (CPLEX). We also notice that SCIP, a CIP solver, tops to about 66% of the dataset, while the ICP solver (HYSAT) tops to 62% of the dataset.

In Table 3 we report a “domain-centered” classification. In the table, the number of instances solved and the cumulative time taken for each domain is computed considering the “State Of The Art” (SOTA) solver, i.e., the ideal solver that always fares the best time among all the solvers. Thus, an instance is solved if at least one of the solvers solves it, and the considered time is the best among the times of the solvers that solved the instances. The instances are also classified according to their empirical hardness as follows: an instance is called “easy” if it has been solved by all the considered solvers; “medium” are those non-easy that can be solved by at least two solvers; “medium-hard” are those solved by only one solver; “hard” are the ones remained unsolved. Looking at Table 3, we can see that the SOTA solver is able to solve 64 instances, resulting in 19 easy, 44 medium, 1 medium-hard, and 7 hard instances. Considering OPENSTACKS-IPC08, performances of SOTA solver are to ascribe to CLASP, that also solves uniquely 1 instance. Looking now at PATHWAYS, we notice that 5 (out of 20) instances are very challenging for the whole pool of solvers. We also notice that the SOTA solver major contributors are MINIMAXSAT and INCWMAXSATZ, with 6 and 4 instances, respectively. Finally, considering the whole dataset, we report that the major contributors to the SOTA solver are INCWMAXSATZ and MINIMAXSAT (33%), followed by MIN-

ISAT+ and CLASP (11%). We also report that the remaining 7 solvers (out of 14) do not contribute to the SOTA solver.

## References

1. M. van den Briel, R.S. Nigenda, M.B. Do, and S. Kambhampati. Effective approaches for partial satisfaction (over-subscription) planning. In *Proc. of AAAI 2004*, pages 562–569, 2004.
2. D.E. Smith. Choosing objectives in over-subscription planning. In *Proc. of ICAPS 2004*, pages 393–401. AAAI, 2004.
3. M. Maratea. Planning as satisfiability with IPC simple preferences and action costs. Technical report, Available at <http://www.star.dist.unige.it/~marco/Data/TR-planning.pdf>, 2011.
4. N. Robinson, C. Gretton, D. Nghia Pham, and A. Sattar. Partial weighted maxsat for optimal planning. In *Proc. of PRICAI 2010*, volume 6230 of *LNCS*, pages 231–243. Springer, 2010.
5. O. Bailleux and Y. Boufkhad. Efficient CNF encoding of boolean cardinality constraints. In *Proc. of CP 2003*, volume 2833 of *LNCS*, pages 108–122. Springer, 2003.
6. H. Kautz and B. Selman. Unifying SAT-based and graph-based planning. In Thomas Dean, editor, *Proc. of IJCAI 1999*, pages 318–325. Morgan-Kaufmann, 1999.
7. E. Pednault. ADL and the state-transition model of action. *Journal of Logic and Computation*, 4:467–512, 1994.
8. B. Cenk Gazen and Craig A. Knoblock. Combining the expressivity of UCPOP with the efficiency of Graphplan. In *Proc. of ECP 1997*, pages 221–233, 1997.
9. A. Gerevini, P. Haslum, D. Long, A. Saetti, and Y. Dimopoulos. Deterministic planning in the 5th IPC: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5-6):619–668, 2009.
10. J. Argelich, C.M. Li, F. Manyà, and J. Planes. The first and second Max-SAT evaluations. *Journal of Satisfiability, Boolean Modeling and Computation*, 4(2-4):251–278, 2008.
11. V.M. Manquinho and O. Roussel. The first evaluation of pseudo-Boolean solvers (PB’05). *Journal on Satisfiability, Boolean Modeling and Computation*, 2:103–143, 2006.
12. Martin Gebser, Benjamin Kaufmann, André Neumann, and Torsten Schaub. Conflict-driven answer set solving. In *In Proc. of IJCAI 2007*, pages 386–392, 2007.
13. M. Denecker, J. Vennekens, S. Bond, M. Gebser, and Mi. Truszczynski. The second Answer Set Programming Competition. In *Proc. of LPNMR 2009*, volume 5753 of *LNCS*, pages 637–654. Springer, 2009.
14. M. Franzle, C. Herde, T. Teige, S. Ratschan, and T. Schubert. Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. *Journal of Satisfiability, Boolean Modeling and Computation*, 1:209–236, 2007.