No One SATPlan Encoding To Rule Them All

Tomáš Balyo
Karlsruhe Institute of Technology
Karlsruhe, Germany
tomas.balyo@kit.edu

Roman Barták
Charles University in Prague, Faculty of Math. and Physics
Prague, Czech Republic
bartak@ktiml.mff.cuni.cz

Abstract

Solving planning problems via translation to propositional satisfiability (SAT) is one of the most successful approaches to automated planning. An important aspect of this approach is the encoding, i.e., the construction of a propositional formula from a given planning problem instance. Numerous encoding schemes have been proposed in the recent years each aiming to outperform the previous encodings on the majority of the benchmark problems. In this paper we take a different approach. Instead of trying to develop a new encoding that is better for all kinds of benchmarks we take recently developed specialized encoding schemes and design a method to automatically select the proper encoding for a given planning problem instance. In the paper we also examine ranking heuristics for the Relaxed Relaxed Exists-Step encoding, which plays an important role in our algorithm. Experiments show that our new approach significantly outperforms the state-of-the-art encoding schemes when compared on the benchmarks of the 2011 International Planning Competition.

Introduction

One of the most successful approaches to automated planning is encoding the planning problem into a series of satisfiability (SAT) formulas and then using a SAT solver to solve them. The method was first introduced by Kautz and Selman (Kautz and Selman 1992) and is still very popular and competitive. This is partly due to the power of SAT solvers, which are getting more efficient year by year. Since then many new improvements have been made to the method, such as new compact and efficient encodings (Huang et al. 2010; Rintanen et al. 2006; Robinson et al. 2009; Balyo 2013), better ways of scheduling the SAT solvers (Rintanen et al. 2006), and modifying the SAT solver's heuristics to be more suitable for solving planning problems (Rintanen 2012).

In this paper we focus on the problem of encoding, i.e., the construction of propositional formulas from planning problem instances. The traditional approach is to design a universal encoding that works well for the largest subset of the benchmark problems and then use that encoding for each problem. In contrast, our approach is to select a set of diverse encoding schemes and design a rule that can choose the best encoding from this set for a given problem instance. The idea is closely related to the approach of sequential planning portfolios (Seipp et al. 2012). We also revisit action ranking, which is a crucial aspect of the Relaxed Relaxed Exists-Step encoding used in our set of encodings.

In the experimental section of the paper we compare our new method to state-of-the-art encodings on the complete set of benchmark problems from the 2011 International Planning Competition (IPC) (Coles et al. 2012). The experimental results show that our new approach is a significant improvement over the state-of-the-art encodings with respect to both the number of solved problems and the time required to solve them.

Preliminaries

In this section we give the formal definitions of automated planning. We will use the multivalued SAS+ formalism (Bäckström and Nebel 1995) instead of the classical STRIPS formalism (Fikes and Nilsson 1971) based on propositional logic. A planning task Π in the SAS+ formalism is defined as a tuple $\Pi = (X, O, s_I, s_G)$ where

- $X = \{x_1, \ldots, x_n\}$ is a set of multivalued variables with finite domains $\mathrm{dom}(x_i)$.
- O is a set of actions (or operators). Each action $a \in O$ is a tuple $(\mathrm{pre}(a), \mathrm{eff}(a))$ where $\mathrm{pre}(a)$ is the set of preconditions of a and $\mathrm{eff}(a)$ is the set of effects of a. Both preconditions and effects are of the form $x_i = v$ where $v \in \mathrm{dom}(x_i)$.
- A state is a set of assignments to the state variables. Each state variable has exactly one value assigned from its respective domain. We denote by S the set of all states. $s_I \in S$ is the initial state. s_G is a partial assignment of the state variables (not all variables have assigned values) and a state $s \in S$ is a goal state if $s_G \subseteq s$.

An action a is *applicable* in the given state s if $\mathrm{pre}(a) \subseteq s$. By $s' = \mathrm{apply}(a, s)$ we denote the state after executing the action a in the state s, where a is applicable in s. All the assignments in s' are the same as in s except for the assignments in $\mathrm{eff}(a)$ which replace the corresponding (same variable) assignments in s. If $P =$

$[a_1, \ldots, a_k]$ is a sequence of actions, then $\text{apply}(P, s) = \text{apply}(a_k, \text{apply}(a_{k-1} \ldots \text{apply}(a_2, \text{apply}(a_1, s)) \ldots))$. A *sequential plan* P of *length* k for a given planning task Π is a sequence of k actions P such that $s_G \subseteq \text{apply}(P, s_I)$.

Parallel Plans

A *parallel plan* P with *makespan* k for a given planning task Π is a sequence of sets of actions (called parallel steps) $P = [A_1, \ldots, A_k]$ such that $\trianglelefteq(A_1) \oplus \cdots \oplus \trianglelefteq(A_k)$ is a sequential plan for Π, where $\trianglelefteq$ is an ordering function, which transforms a set of actions A_i into a sequence of actions $\trianglelefteq(A_i)$ and $\oplus$ denotes the concatenation of sequences.

Let us denote by s_j the world state in between the parallel steps A_j and A_{j+1}, which is obtained by applying the sequence $\trianglelefteq(A_j)$ on s_{j-1}, i.e., $s_j = \text{apply}(\trianglelefteq(A_j), s_{j-1})$ (except for $s_0 = s_I$).

Which actions can be together in a parallel step is defined by four parallel planning semantics: "for all"($\forall$)(Kautz and Selman 1996), "exists"($\exists$)(Rintanen et al. 2006), "relaxed exists"($R\exists$)(Wehrle and Rintanen 2007), and "relaxed relaxed exists"($R^2\exists$)(Balyo 2013). The requirements of the semantics are displayed in the following table.

	$\forall$	$\exists$	$R\exists$	$R^2\exists$
All possible orderings $\trianglelefteq$ of the sets A_j make valid plans	$\surd$			
Each action $a \in A_j$ is applicable in the state s_j	$\surd$	$\surd$		
The effects of all actions $a \in A_j$ are applied in s_{j+1}	$\surd$	$\surd$	$\surd$	
There exists an ordering $\trianglelefteq$ of the sets A_j to make a valid plan	$\surd$	$\surd$	$\surd$	$\surd$

The following example demonstrates the differences between the four semantics and how the more relaxed semantics can allow shorter makespan plans.

Example 1 *We will model a simple package delivery scenario with a truck that needs to deliver two packages to the location C from the locations A and B. In the beginning the truck is located in A. The locations A, B, and C are all connected by direct roads. We will model the planning task using the following variables:*

- *Truck location T,* $dom(T) = \{A, B, C\}$
- *Package locations* P_1 *and* P_2, $dom(P_1) = dom(P_2) = \{A, B, C, L\}$ *(*$P_i = L$ *represents that the package* i *is on the truck).*

Now, having defined the variables $X = \{T, P_1, P_2\}$ *and their respective domains, we can define the initial state* s_I *and the goal conditions* s_G*.*

$s_I = \{T = A, P_1 = A, P_2 = B\}$ $s_G = \{P_1 = C, P_2 = C\}$

The action templates (operators) are described in the following table. To get the actual actions we need to substitute $l, l1, l2$ *with* A, B *and* C *and* i *with* 1 *and* 2*.*

Action	*Preconditions*	*Effects*
$move(l1, l2)$	$T = l1$	$T = l2$
$loadP_i(l)$	$T = l, P_i = l$	$P_i = l$
$unloadP_i(l)$	$T = l, P_i = l$	$P_i = l$

An optimal sequential plan for this planning problem is the following: $P^* = [loadP_1(A), move(A, B), loadP_2(B), move(B, C), unloadP_1(C), unloadP_2(C)]$*. The four semantics allow four different parallel plans:*

- *The* $R^2\exists$*-Step semantics allows the single step plan* $[\{loadP_1(A), move(A, B), loadP_2(B), move(B, C), unloadP_1(C), unloadP_2(C)\}]$.
- *The Relaxed* $\exists$*-Step semantics forbids having both* $move$ *operations inside one step, since the effect of the first* $move$ *action is not applied after the step. Therefore the shortest possible Relaxed* $\exists$*-Step plan is the following two step parallel plan* $[\{loadP_1(A), move(A, B), loadP_2(B)\}, \{move(B, C), unloadP_1(C), unloadP_2(C)\}]$.
- *The* $\exists$*-Step semantics does not allow the second* $load$ *action to be inside the first step, since its precondition is not satisfied in the initial state. Similarly, the* $unload$ *actions cannot be in the same step as the second* $move$ *action. Therefore the plan needs to have three parallel steps* $[\{loadP_1(A), move(A, B)\}, \{loadP_2(B), move(B, C)\}, \{unloadP_1(C), unloadP_2(C)\}]$.
- *To satisfy the* $\forall$*-Step semantics all actions, except for the* $unload$ *actions, must be in separate parallel steps. The following five step plan is possible* $[\{loadP_1(A)\}, \{move(A, B)\}, \{loadP_2(B)\}, \{move(B, C)\}, \{unloadP_1(C), unloadP_2(C)\}]$.

Finding Plans using SAT

The basic idea of solving planning as SAT is the following (Kautz and Selman 1992). We construct (by encoding the planning task) a series of SAT formulas $F_1, F_2, \ldots$ such that if F_i is satisfiable then there is a parallel plan of makespan $\leq i$. Then we solve them one by one starting from F_1 until we reach the first satisfiable formula F_k. From the satisfying assignment of F_k we can extract a plan of makespan k.

There exist more advanced algorithms for scheduling the solving of the F_i formulas (Rintanen et al. 2006) but since our goal is only to compare different encoding methods we will use the basic sequential scheduling algorithm.

Action Ranking for the $R^2\exists$-Step Encoding

The recently introduced (Balyo 2013) Relaxed Relaxed Exists-Step Encoding which approximates the $R^2\exists$-Step parallel planning semantics has a very important aspect that was not addressed adequately enough in the original publication. This aspect is the ranking of actions and since it is an important part of our proposed method we will take a more detailed look in this paper.

The Relaxed Relaxed Exists Step encoding only approximates the $R^2\exists$-Step semantics due to working with a fixed ordering of actions. This ordering is also used when the sequential plan is constructed from the action sets of the parallel plan obtained from the satisfying assignment of the formula F_k. To truly represent the $R^2\exists$-Step parallel planning semantics the encoding would have to consider all the possible rankings (orderings) of actions. Since we work with just one particular ordering a great care must be given to its selection.

Table 1: Problems solved (#P) and their total makespan.

Domain	TSort #P/Mks	TSort^{-1} #P/Mks	Input #P/Mks	Input^{-1} #P/Mks	Random #P/Mks
barman	**4**/36	2/29	**4**/60	2/28	1/11
elevators	**20**/85	**20**/99	**20**/106	**20**/79	**20**/75
floortile	17/158	**18**/185	16/149	**18**/178	**18**/167
nomystery	3/14	4/20	3/13	**6**/33	3/14
openstacks	12/75	13/66	**20**/59	5/43	10/57
parcprinter	**20**/30	**20**/249	**20**/88	**20**/186	**20**/140
parking	0/0	0/0	0/0	0/0	0/0
pegsol	**19**/158	18/155	12/147	16/142	18/152
scanalyzer	6/11	**9**/16	7/12	6/13	6/12
sokoban	**1**/17	**1**/19	**1**/18	**1**/17	**1**/19
tidybot	**1**/1	**1**/1	**1**/1	**1**/1	**1**/1
transport	5/20	6/40	8/44	**9**/57	4/19
visitall	**20**/34	12/113	9/55	9/49	12/80
woodwork	**20**/33	**20**/57	**20**/58	**20**/30	**20**/40
Total	**148**	144	141	133	134

The ordering of actions will be defined using action ranks. The *ranking of actions* is the assignment of a unique integer rank $r(a)$ to each action $a \in O$. For the sake of correctness of the encoding the ranking can be arbitrary, however the selection of the ranking affects the performance of the planning process dramatically. Intuitively, the ranking should be such that the actions are ranked according to their order in a valid plan for the given planning task. The problem is, of course, that we do not know the plan in advance. Therefore, we should try to guess the order in which the given actions could appear in a plan.

In the original $R^2\exists$-Step paper (Balyo 2013) only one ranking heuristic was described and used – the *Topological Ranking*. In this paper we describe new ranking methods and compare them experimentally. Altogether we examine the following ranking heuristics.

- *Random Ranking*. The ranks are assigned randomly to actions, each action has a unique rank. The purpose of this method is to serve as a baseline for comparison with the other methods.
- *Input Ranking*. The ranks are assigned from 1 to $n = |O|$ to the actions in the order in which they are listed in the input, i.e., in the definition of the planning task. The intuition behind this method is, that the author of the problem may have defined the actions in the order they are expected to appear in the plans.
- *Inverted Input ranking*. First we take the Input Ranking, then we invert the ranks: $r(a) := n - r(a)$, where n is the number of actions. If the Input Ranking is a 'good' ranking, then this ranking is supposed to be bad.
- *Topological Ranking*. The original ranking method of (Balyo 2013) is based on topologically ordering the action enabling graph while ignoring cycles.
- *Inverted Topological Ranking*. We invert the Topological Ranking in the same way we inverted the Input Ranking.

The weakness of all these ranking methods is that only the actions of a planning task are considered. For example the initial state and the goal conditions are not used at all. Designing better ranking heuristics is a matter of future work.

Ranking Comparison

We have run the basic SAT planning algorithm with the $R^2\exists$-Step encoding using the five different ranking methods described above with a time limit of five minutes for SAT solving (time for ranking and encoding is not considered). The details of the experimental environment are given in the Experiments section below.

The number of solved problems (out of 20 in each domain) and the total makespan of the found plans is presented in Table 1. Note, that the makespan represents the number of propositional formulas that were solved to obtain the parallel plan and not the length of the final sequential plans.

The Topological Ranking (TSort) method solved the highest number of problems while the Inverted Input and Random methods were the least successful. Nevertheless, there are five domains, where TSort was outperformed. Most notable among these is the openstacks domain, where the Input ranking solved almost twice as many problems while its total makespan remained low. Focusing on the makespans, we can observe, that in several cases TSort maintains the smallest total makespan while solving as many or more problems than the other methods. This is most apparent for the parcprinter and visitall domains.

Overall, we can conclude, that the action ranking affects the performance of planning very significantly and different ranking methods work well for different domains. Nevertheless, TSort appears to be the best choice in general while Input provides great performance on some of the domains and therefore we will use these two in our algorithm.

Encoding Selection

As we described in the introduction, our algorithm consists of a set of encodings E and a selection rule. The task of the selection rule is to select an encoding from E that should be used to solve a given planning task based on the planning task itself and other available information (for example the makespan parameter of the formula that we want to construct). For our algorithm we will use two encodings, the Reinforced encoding (Balyo et al. 2015), which represents the $\forall$-Step parallel planning step semantics, and the $R^2\exists$-Step encoding (Balyo 2013) with two ranking methods – Topological and Input ranking.

The selection rule is based on our experimental observations, which suggest, that the $R^2\exists$-Step encoding performs best if the number of transitions is low. A transition (Huang et al. 2010) is a change of a state variable and the set of transitions of a planning problem instance is defined by its actions. The selection rule decides based on the average number of transitions, i.e., the total number of transitions ($|\Delta|$) divided by the number of state variables ($|X|$) in the given planning task. Based on our experiments we decided to use the threshold value 10. Therefore the first part of the rule is: *If* $|\Delta|/|X| > 10$ *then use the Reinforced encoding otherwise use the* $R^2\exists$*-Step encoding*. Furthermore, when the

Table 2: Problems solved (out of 20) in 30 mins.

Domain	Reinf	$R^2\exists$	Sel	R$\forall$	R$\exists$	R*
barman	4	8	**9**	8	4	8
elevators	**20**	**20**	**20**	**20**	**20**	**20**
floortile	18	18	18	16	**20**	**20**
nomystery	**20**	6	**20**	**20**	**20**	**20**
openstacks	0	15	**20**	0	0	0
parcprinter	**20**	**20**	**20**	**20**	**20**	**20**
parking	0	0	0	0	0	0
pegsol	10	**19**	**19**	11	12	12
scanalyzer	15	9	15	17	**18**	**18**
sokoban	2	2	2	**6**	**6**	**6**
tidybot	2	2	2	13	**15**	**15**
transport	18	13	**19**	18	18	18
visitall	10	**20**	**20**	11	11	11
woodworking	**20**	**20**	**20**	**20**	**20**	**20**
Total	159	172	**204**	180	184	188

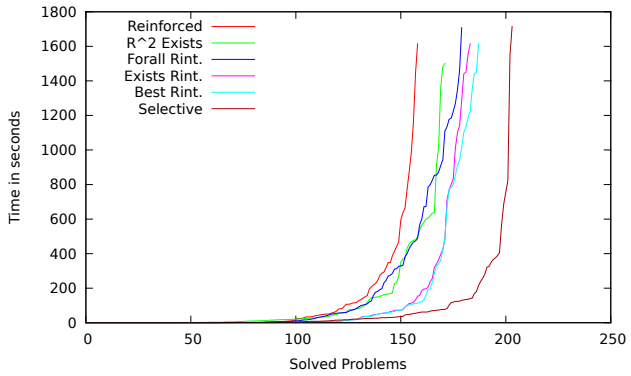

Figure 1: Coverage per time limit.

$R^2\exists$-Step Encoding is used, we alternate between two action ranking heuristics. For even makespans we use the Topological Ranking method and for the odd makespans we use the Input Ranking method (rank the actions in the order they are defined in the input problem).

Experiments

The experiments were run on a computer with Intel i7 920 CPU @ 2.67 GHz processor and 6 GB of memory. The benchmark problems are from the optimal track of the 2011 International Planning Competition (IPC) (Coles et al. 2012) translated from PDDL to SAS+ using Fast Downward (Helmert 2006). To solve the instances we used the SAT solver Lingeling (Biere 2014) (version ats).

We did experiments with a 30 minutes time limit for SAT solving using the following six encodings. We used the Reinforced Encoding (Reinf) (Balyo et al. 2015) and $R^2\exists$-Step Encoding ($R^2\exists$), which are the two components of the Selective Encoding (Sel). We also used Rintanen's $\forall$-Step (R$\forall$) and $\exists$-Step Encodings (Rintanen et al. 2006) from his popular state-of-the-art SAT based planning system Madagascar (Rintanen 2013). Best of Rintanen (R*) is a virtual best encoding obtained by selecting the better result for each instance from the results of R$\forall$ and R$\exists$.

The number of solved instances is presented in Table 2. We can observe, that the openstacks domain is very difficult for all but the $R^2\exists$-Step encoding (and Selective, which also uses it). The Selective encoding solves more than the $R^2\exists$-Step encoding thanks to using two kinds of action ranking instead of just one. The sokoban and tidybot domains are very hard for all of our encodings, while the encodings of Rintanen can handle them much better.

If we compare the encodings, we can observe that the best results are obtained by the Selective encoding followed by the Rintanen encodings. The Selective encoding very successfully combines the Reinforced and $R^2\exists$-Step encodings and is never worse for any domain than any of its components. The strength of the Selective encoding is that it can properly select the Reinforced encoding for the domains and problems, where the $R^2\exists$-Step is not efficient.

Although R$\forall$ and R$\exists$ are the best individual encodings, their combination (the R*) does not bring any significant improvement. This suggest that they are not as diverse in the sense of problem coverage.

In Figure 1 we provide a so called cactus plot, that visually compares the six tested encodings. The plot represents how many problems can be solved withing a given time limit. It is interesting, that the lines for the $R^2\exists$-Step and the 'Rintanen $\forall$' encoding cross each other several times, which means, that for certain time limits $R^2\exists$-Step would solve more instances than the 'Rintanen $\forall$' encoding. The cactus plot confirms, that the Selective method significantly outperforms all the other methods, furthermore, this holds for any reasonable time limit.

Conclusion

In this paper we demonstrated that the idea of combining encodings by designing a simple rule that selects the more suitable encoding for a given planning problem instance is a perspective research direction to improve the performance of SAT based planning algorithms. Our experiments also revealed that simply combining the best state-of-the-art encodings (of Rintanen) does not lead to the best results.

The presented rule is very simple and the set of encodings used for the selection is relatively small. Therefore we believe further improvements could be achieved by extending the set of encodings by novel or existing encodings and finding more sophisticated selection rules.

Another promising topic of future work is finding better ranking heuristics to be used for the Relaxed Relaxed Exists-Step encoding. The current ones are very simplistic and do not use all the available information about the problem instance (such as initial state or goal conditions).

Acknowledgments This research was partially supported by DFG project SA 933/11-1. Roman Barták is supported by the Czech Science Foundation under the project P103-15-19877S.

References

Bäckström, C., and Nebel, B. 1995. Complexity results for sas+ planning. *Computational Intelligence* 11:625–656.

Balyo, T.; Barták, R.; and Trunda, O. 2015. Reinforced encoding for planning as sat. In *Acta Polytechnica CTU Proceedings*.

Balyo, T. 2013. Relaxing the relaxed exist-step parallel planning semantics. In *ICTAI*, 865–871. IEEE.

Biere, A. 2014. Lingeling and plingeling home page. http://fmv.jku.at/lingeling/.

Coles, A. J.; Coles, A.; Olaya, A. G.; Celorrio, S. J.; López, C. L.; Sanner, S.; and Yoon, S. 2012. A survey of the seventh international planning competition. *AI Magazine* 33(1).

Fikes, R., and Nilsson, N. J. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artif. Intell.* 2(3/4):189–208.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research (JAIR)* 26:191–246.

Huang, R.; Chen, Y.; and Zhang, W. 2010. A novel transition based encoding scheme for planning as satisfiability. In Fox, M., and Poole, D., eds., *AAAI*. AAAI Press.

Kautz, H. A., and Selman, B. 1992. Planning as satisfiability. In *ECAI*, 359–363.

Kautz, H., and Selman, B. 1996. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, 1194–1201. AAAI Press.

Rintanen, J.; Heljanko, K.; and Niemelä, I. 2006. Planning as satisfiability: parallel plans and algorithms for plan search. *Artif. Intell.* 170(12-13):1031–1080.

Rintanen, J. 2012. Planning as satisfiability: Heuristics. *Artif. Intell.* 193:45–86.

Rintanen, J. 2013. Planning as satisfiability: state of the art. http://users.cecs.anu.edu.au/ jussi/satplan.html.

Robinson, N.; Gretton, C.; Pham, D. N.; and Sattar, A. 2009. Sat-based parallel planning using a split representation of actions. In Gerevini, A.; Howe, A. E.; Cesta, A.; and Refanidis, I., eds., *ICAPS*. AAAI.

Seipp, J.; Braun, M.; Garimort, J.; and Helmert, M. 2012. Learning portfolios of automatically tuned planners. In McCluskey, L.; Williams, B.; Silva, J. R.; and Bonet, B., eds., *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012*. AAAI.

Wehrle, M., and Rintanen, J. 2007. Planning as satisfiability with relaxed exist-step plans. In Orgun, M. A., and Thornton, J., eds., *Australian Conference on Artificial Intelligence*, volume 4830 of *Lecture Notes in Computer Science*, 244–253. Springer.