Effective CNF Encodings for the Towers of Hanoi

Ruben Martins and Inês Lynce
IST/INESC-ID, Technical University of Lisbon, Portugal
{ruben,ines}@sat.inesc-id.pt

Abstract

One of the most well-known CNF benchmark encodes the problem of the Towers of Hanoi. This benchmark is available from SATlib and has been part of the set of problem instances used in more than one edition of the SAT competition. The existing CNF instances build upon an encoding using the STRIPS language. Although the available instances (ranging from 4 to 6 disks) are hard to solve for most of the solvers, we introduce improvements over the existing encodings and present a new CNF encoding that makes these problem instances trivial to solve using only unit propagation. This new encoding is also based on the STRIPS language and makes it possible to solve the problem of 18 disks in a reasonable amount of time.

1 Introduction

Propositional satisfiability (SAT) solvers are currently known for being very efficient for solving different problems. Also, its use is quite simple as the conjunctive normal form (CNF) is traditionally and commonly accepted by SAT solvers. This contrasts with other technologies where different formats are accepted as each format carries its own advantages. Maybe for being restricted to the CNF format, not much importance has been given to modelling techniques in SAT.

The Towers of Hanoi (ToH) [4] have been been encoded in the past based on the STRIPS language [1] since it is easily translated into SAT (e.g. [3]). This paper introduces improvements over the existing encodings and presents a new effective CNF encoding for ToH that is also based on the STRIPS language. (An extended version of this work is available as a technical report [5].) Although SAT technology has never advocated being the best approach for solving the ToH, it is also true that the existing CNF problem instances of the ToH are very hard to solve. This comes somehow as a surprise: a problem with 4 disks may be easily solved by hand but looks intractable to SAT solvers.

The new CNF encoding builds on existing encodings [3, 7] and further incorporates a key number of properties of the ToH [8] which seem to be essential when solving the problem automatically. Although the use of these properties makes the ToH much easier to solve, we argue that such kind of properties should be used when modelling a problem to be solved using SAT or any other technology. It makes non sense to produce hard CNF encodings from problem instances that are known to be trivial to solve.

The produced CNF instances can now be solved using only unit propagation, but at the cost of requiring a significant number of variables and clauses for larger instances. Even though this represents a drawback, we are now able to generate and solve the problem of 18 disks in a reasonable amount of time.

2 Towers of Hanoi (ToH)

The Towers of Hanoi are a mathematical puzzle formalized by the French mathematician Édouard Lucas in 1883 [4]. Each problem consists of three towers (T_1, T_2, T_3), and n disks of different sizes which can slide onto any tower. At the initial state the disks are stacked in order of size on one tower (T_1), the smallest being at the top. The problem is solved by moving the entire initial stack to another tower (T_3), obeying to the following rules: (1) only one disk may be moved at a time, (2) no disk may be placed on the top of a smaller disk and (3) each move consists in taking the upper disk from one of the towers and sliding it onto the top of another tower. A solution is therefore a sequence of disk movements from the

G. Sutcliffe, A. Voronkov (eds.): easychair 1.0, 2008, volume 1, issue: 1, pp. 1-5

initial state where all disks are stacked in order of size on T_1 to the final state where all disks are stacked in the same order on T_3.

The simplest solution to the ToH is based on a divide-and-conquer strategy: a solution to the problem of n disks is expressed in terms of a solution to the ToH with $n-1$ disks. The ToH has several key properties [8] that are essential for an effective solving. Next we present those key properties which will be encoded in section 4 in order to improve the existing encodings.

Property 1. *Given a ToH of size n, one may easily find a solution taking into account the solution for a ToH of size $n-1$. The solution is built by first moving the smallest $n-1$ disks to T_2, then moving the n^{th} disk to T_3 and finally moving the smallest $n-1$ disks from T_2 to T_3. The sequence of disks to be moved when considering only the smallest $n-1$ disks may be obtained from the solution for a ToH of size $n-1$. More generally, the sequence of moves for a ToH of size n (S(n)) is recursively defined as follows:*

$$S(1) = \{1\}; S(n) = \{S(n-1), n, S(n-1)\}$$

Notice that the order of the disks to be moved after moving the largest disk is exactly the same as before moving it. Moreover, there is a relation between the towers involved in the same movement before and after the largest disk is moved. It consists in simply shifting the towers from/to where the disks have to be moved ($T_1 \rightarrow T_2, T_2 \rightarrow T_3, T_3 \rightarrow T_1$). For example, an initial move from T_1 to T_3 should later correspond to a move from T_2 to T_1.

Property 2. *Assuming that $n-1$ disks are moved using the minimum number of moves, then the recursive algorithm makes no more than the minimum number of moves. From property 1, we may infer by mathematical induction that for n disks the number of required moves ($M(n)$) is the following:*

$$M(1) = 1; M(n) = 2 \cdot M(n-1) + 1 = 2^n - 1$$

If we also consider the relationship between the movement of the disks before/after moving the largest disk we only need to determine the first $2^{n-1}-1$ steps (M(n-1)) since the remaining moves may be trivially obtained from those.

Property 3. *When moving disks, if we consider the disks numbered by their size, no two disks of the same parity can be in direct contact. In other words, no two odd/even disks can be moved next to each other.*

Property 4. *All disks cycle in a given order between the towers [2]. If n is even then the odd disks will cycle clockwise ($T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_1$) while the even disks will cycle counterclockwise ($T_1 \rightarrow T_3 \rightarrow T_2 \rightarrow T_1$). Otherwise, if n is odd then the odd disks will cycle counterclockwise while the even disks will cycle clockwise.*

3 Existing CNF Encodings for ToH

The first CNF STRIPS-based encoding [3] uses the set of variables $\{on(d,dt,t), clear(dt,t), move(d,dt,dt',t)\}$, where the additional argument t represents a time step with $0 \leq t \leq 2^n - 1$. In order to reduce the number of required variables, the variable $move(d,dt,dt',t)$ is replaced by conjunction ($obj(d,t) \wedge from(dt,t) \wedge to(dt',t)$). The clauses encode the following constraints: (1) exactly one disk is moved at each time step, (2) there is exactly one movement at each time step (which implies that there is exactly one pair *from/to*), (3) there are no movements moving a disk from/to exactly the same position (which would not imply a movement in practice), (4) for a movement to be performed the preconditions must be satisfied, (5) after performing a movement the postconditions are implied and (6) no disks can be moved to the top of smaller disks. In addition, the initial state holds at time step 0 and the goal state holds at time step $2^n - 1$. Finally, extra clauses are required for preserving what is unaffected by movements.

3.1 An improved CNF STRIPS-based encoding

A recent encoding proposed by Prestwich [7] produces a more compact formula as a result of using the structure of the ToH. ToH have originally been modeled similarly to the Blocks World problem and it has been observed that for that reason the encoding is significantly larger.

The action $move(d,tw,tw',t)$ where tw/tw' correspond to towers, now only considers movements of disks between towers and not between disks or towers like the previous encoding. With this change the variables *move* and *on* are kept but the variable *clear* can be eliminated. The preconditions are $(on(d,tw,t) \land \neg on(1,tw,t) \land \ldots \land \neg on(d-1,tw,t) \land \neg on(1,tw',t) \land \ldots \land \neg on(d-1,tw',t))$ and the post-conditions are $(on(d,tw',t+1) \land \neg on(d,tw,t+1))$. The initial state is represented by clause $(on(1,T_1,0)$ $\land \ldots \land on(n,T_1,0))$ and the goal state is represented by clause $(on(1,T_3,2^n-1) \land \ldots \land on(n,T_3,2^n-1))$.

This encoding uses the set of variables $\{obj(d,t),\ from(tw,t),\ to(tw,t),\ on(d,tw,t)\}$. Variables *from* and *to* on the first argument and variables *on* on the second argument only refer to towers where in the previous encoding they would also refer to disks. Therefore the number of variables is reduced and those have a major impact on the size of the formula since we will require significant less clauses to encode the constraints presented in the previous encoding. For example, for 5 towers the previous encoding requires 1,931 variables and 14,468 clauses, whereas this new encoding requires 821 variables and 6,457 clauses.

3.2 Other encodings

The two editions of the CSP competition (2005 and 2006) have included a set of problem instances of the ToH. These instances range from 3 to 7 disks. Detailed results show that a translation from CSP to SAT using the support encoding (rather than the direct encoding) which is performed by solver `sat4jcsp` (available from http://www.sat4j.org) produces CNF formulas that can be solved using only unit propagation. The CSP description for ToH includes properties 1 and 2 described earlier. However, the produced CNF instances are much larger than the ones produced by our encoding.

In addition, independent work [6] also reports a CNF encoding that produces formulas that contain only Horn clauses. This encoding uses predicate $p(td_1,\ldots,td_n)$ to encode each state. Each variable td_i has domain values $\{0,1,2\}$ and denotes the tower on which disk i is placed. Nonetheless, this encoding requires a significant number of variables and clauses in such a way that more time is required to generate the formula rather than to solve it.

4 Effective CNF Encodings for ToH

We took the encoding of Prestwich [7] and improved it by encoding the properties described previously.

The Disk Parity encoding

Property 2 is incorporated in the encoding by reducing the search space to the first $2^{n-1}-1$ steps and by modifying the goal state so that we have the larger disk on T_1 and the remaining disks on T_2. The goal state is now $(on(n,T_1,2^{n-1}-1) \land on(1,T_2,2^{n-1}-1) \land \ldots \land on(n-1,T_2,2^{n-1}-1))$. The remaining moves may be trivially obtained as explained earlier. Property 3 is encoded through additional clauses.

The Disk Cycle encoding

The Disk Parity encoding can be further improved by additionally incorporating property 4.

The Disk Sequence encoding

Property 1 recursively determines the disks to be moved at each step. We first generate the sequence of disks to be moved and then add unit clauses $(obj(D,t))$ where D corresponds to the disk to be moved

Table 1: Results for the encodings with a time limit of 10,000 seconds.

Size	Prestwich	Disk Parity	Disk Cycle	Disk Sequence
4	0.01	0	0	0
5	0.08	0.01	0.02	0
6	0.47	0.03	0.05	0
7	3.65	0.70	0.20	0.01
8	109.7	5.19	5.18	0.03
9	7126.57	79.11	7.65	0.09
10	-	1997.19	973.95	0.23
11	-	-	1206.37	0.56
12	-	-	-	1.32

at time step t. The introduction of additional clauses for encoding properties 1 together with 2 suffice to generate a formula that requires only unit propagation to be solved. However, we may further reduce the size of the encoding. Taking into consideration property 1 we can keep only the variables *on* and drop all the other variables. The resulting variables are therefore $on(d,tw,t)$ with $1 \leq d \leq n$, $1 \leq tw \leq 3$ and $0 \leq t \leq 2^{n-1}$. On the other hand, after the elimination of all variables but variable *on*, the formula is no longer solved using only unit propagation. Extra clauses encoding property 3 had to be added to guarantee the resulting formula to be solved with unit propagation. The clauses encode the following constraints: (1) at each time step each disk is placed exactly on one tower, (2) at time step t disk D is moved to another tower where no disks smaller than D may exist, (3) disks not moved at time step t will remain on the same tower at step $t+1$ and (4) odd/even disks will never be in touch with other odd/even disks. In addition, the initial state holds at time step 0 and the goal state holds at time step $2^{n-1}-1$.

5 Experimental Results

We have first generated problem instances ranging from 4 to 12 disks using Prestwich encoding [7] and the improved encodings presented in section 4.

The results of the different encodings are given in table 1 and were obtained on a Intel Xeon 5160 server (3.0GHz, 1333Mhz, 4MB) running Red Hat Enterprise Linux WS 4. For each instance is given the CPU time (s) for solving the instance using `picosat-535` (available from http://fmv.jku.at/picosat/) with a time limit of 10,000 seconds ('-' denotes that the given instance was not solved within the time limit).

Table 1 shows that even the Disk Parity encoding is substantially more effective than Prestwich encoding since we can now solve up to 10 disks in the given time limit. With the Disk Cycle encoding we are able to solve the problem instance with 11 disks and the smaller instances much faster.

These improvements can be first explained by the compactness of the encodings. The number of variables and clauses are reduced to about one half. Moreover, the additional properties help in solving the problem in a more efficient way. However, if we look at the results for the Disk Sequence encoding we can see that we are able to easily solve all instances. This is due to no search being required for solving any of the instances.

In table 2 we present more detailed results for problem instances ranging from 13 to 18 disks using the Disk Sequence encoding. Larger instances could possibly have been generated but would have required a prohibitively amount of memory. For each instance is given the number of variables and clauses of the CNF formula, the memory (MB) required for representing each instance and the CPU time (s) for

Table 2: Results for the Disk Sequence encoding for the ToH.

Size	#Vars	#Cls	Mem	GenTime	SolveTime
13	159,705	827,007	20.9	0.74	3.14
14	344,022	1,859,150	50.9	1.73	7.33
15	737,235	4,177,431	121.6	4.07	17.16
16	1,572,816	9,272,800	294.1	9.44	38.94
17	3,342,285	20,577,699	708.6	12.31	90.15
18	7,077,834	45,219,174	1,637.4	50.48	203.05

generating (GenTime) and solving (SolveTime) the instance using `picosat-535`.

The results reported represent a significant speedup with respect to previous encodings. Even our improvements over Prestwich encoding are not enough to compete with the Disk Sequence encoding. The improvements presented by this new encoding can be first explained by the use of properties 1 and 2: the encoding of both properties allows to solve a problem instance using only unit propagation but none of the properties by itself would allow to. Moreover, restricting variables to *on* allows to significantly reduce the size of the CNF formula. Other existing encodings have clear limitations: the CNF formula produced by `sat4jcsp` has 271,192 variables and 176,639 clauses for 7 disks and the encoding described in [6] requires 40 minutes for generating the problem of 14 disks and more than two hours for 15 disks.

6 Conclusions

This paper introduces improvements on the Prestwich encoding for the ToH [7]. Similarly to well-known existing encodings, our new encodings are based on the STRIPS language. Making use of the properties of the ToH we were able to improve the original encoding since we can significantly reduce the size of the CNF formula. Taking advantage of those properties we were also able to produce a new encoding that is more compact and can solve ToH by only using unit propagation.

References

[1] R. Fikes and N. J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2(3/4):189–208, 1971.

[2] M. M. Fokkinga. On the Iterative Solution of the Towers of Hanoi Problem. Unregistered Technical Note. University of Twente, Enschede, Netherlands, 2000.

[3] H. A. Kautz and B. Selman. Planning as Satisfiability. In *Tenth European Conference on Artificial Intelligence (ECAI'92)*, pages 359–363, 1992.

[4] E. Lucas. La Tour d'Hanoï (Véritable casse-tête annamite). Original printed by Paul Bousrez, Tours, 1883. Published under the acronym N. Claus. Available from http://www.cs.wm.edu/∼pkstoc/page_1f.html.

[5] R. Martins and I. Lynce. Effective CNF Encodings of the Towers of Hanoi. Technical Report 47, INESC-ID, September 2008. Available from http://www.inesc-id.pt/ficheiros/publicacoes/4919.pdf.

[6] J. A. Navarro-Pérez. *Encoding and Solving Problems in Effectively Propositional Logic*. PhD thesis, University of Manchester, November 2000.

[7] S. D. Prestwich. Variable Dependency in Local Search: Prevention Is Better Than Cure. In *Tenth International Conference on Theory and Applications of Satisfiability Testing (SAT'07)*, pages 107–120, 2007.

[8] D. Wood. The Towers of Brahma and Hanoi Revisited. *Journal of Recreational Mathematics*, 14(1):17–24, 1981-1982.