Binary puzzle as a SAT problem

Technische Universiteit Eindhoven University of Technology

Binary puzzle as a SAT problem

Putranto Utomo, Ruud Pellikaan
p.h.utomo@tue.nl
g.r.pellikaan@tue.nl

Binary puzzle

The Binary puzzle is a one person board game where the player has to put a bit 1 or 0 in a partially filled square array such that

1. There are no three consecutive ones and also no three consecutive zeros in each row and each column,
2. Every row and column is balanced, that is the number of ones and zeros must be equal in each row and in each column,
3. No repeated rows and no repeated columns are allowed.

	0						
			1		1		0
		0					
	1						
				1			
	0				1		
	0			0			
				0		0	

Figure 1: Unsolved Puzzle

1	0	**1**	**0**	**1**	**0**	**1**	**0**
0	**1**	**0**	1	**0**	1	**1**	0
1	**0**	0	**1**	**1**	**0**	**0**	**1**
0	1	**1**	**0**	**0**	**1**	**1**	**0**
0	**1**	**0**	**1**	1	**0**	**0**	**1**
1	0	**1**	**0**	**1**	1	**0**	**0**
1	0	**0**	**1**	0	**0**	**1**	**1**
0	**1**	**1**	**0**	0	**1**	0	**1**

Figure 2: Solved Puzzle

Binary puzzle as SAT problem

The satisfiability (SAT) problem is the problem to find an assignment to all variables such that a logical formula is true. Our problem:

1. Converting the Binary puzzle into a SAT problem.
2. Converting the SAT problem into a CNF.
3. Solve using a SAT solver.

The no three consecutive constraint. Let x_1, x_2, x_3 be any three consecutive cells. The following should hold:

$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3).$$

The CNF has $8m(2m-2)$ clauses, and each clause has 3 literals.

The balancedness constraint. Let $\mathbf{x}$ be a vector of length $2m$, every $m+1$ cells must have at least one 1 and one 0:

$$\left(\bigvee_{k=1}^{m+1} x_k\right) \wedge \left(\bigvee_{k=1}^{m+1} \neg x_k\right).$$

Problem: Exponential blow up, since the CNF has $8m\binom{2m}{m+1}$ clauses, where each clause has $m+1$ literals.

Solution: (i) Iteratively sum the vector $\mathbf{x}$ from length 1 up to length $2m$.

Let $\mathbf{a}$ and $\mathbf{b}$ be the same length 2-ary representation of two integers m and n. Let $a_l = b_l = 0$. Then let $\mathbf{c}$ be the representation of $m+n$. Let $\mathbf{t}$ be the memory variable such that $t_1 = 0$ and $t_{i+1} = (a_i \wedge b_i) \oplus (a_i \wedge t_i) \oplus (b_i \wedge t_i)$. Then $c_i = a_i \oplus b_i \oplus t_i$. The result should be the same as the representation of m.

(ii) Use Tseytin transformation [1]. For example, the 2-variable x, y and OR operator gives with the Tseytin variable z the following CNF, which has 3 clauses.

$$z \equiv x \vee y: \quad (\neg z \vee x \vee y) \wedge (z \vee \neg x) \wedge (z \vee \neg y).$$

The no repeated vector constraint. Let $\mathbf{x} \neq \mathbf{y}$ and both have length $2m$. Enumerate all 2^{2m} combinations of $\bigvee_{0 \le i \le j \le 2m} [(\neg x_i \vee \neg y_i) \vee (x_i \vee y_i)]$. Then we have an exponential blow up with $2 \cdot 2^{2m}\binom{2m}{2}$ clauses where each clause has $4m$ literals.

Solution: Use the following formula and Tseytin transformation.

$$\neg \bigwedge_{i=1}^{2m} [(x_i \wedge y_i) \vee (\neg x_i \wedge \neg y_i)].$$

Experiment and conclusion

Intel(R) Core(TM) i7-4700MQ CPU @ 2.40GHz, 10GB RAM. 64 bit Debian 8, Sagemath 7.5.1, CryptoMiniSat 2.

Conclusion: The binary puzzle has a polynomial representation of CNF:

1. $8m(2m-1)$ clauses and each clause has 3 literals for the first constraint.
2. $14(2m)^2 \log_2(2m)$ clauses and each clause has $m+1$ literals for the second constraint.
3. $2\binom{2m}{2}(20m+1)$ clauses and each clause has $4m$ literals for the third constraint.

Table 1: Number of literals

m	Poly. formula	Exp. formula
2	4572	1824
3	17634	24768
4	52152	238912
5	117750	1894560
6	233268	13243680
7	419874	84840224
8	713328	509908608

Table 2: Execution time (sec.)

Exp. formula		Poly. formula	
Pre-com.	Solver	Pre-com.	Solver
0.02	0.00	0.29	0.001
0.14	0.02	0.89	0.002
1.45	0.10	2.28	0.005
10.80	0.69	4.21	0.007
81.87	4.67	7.01	0.011
-	-	10.81	0.016
-	-	16.79	0.027

Remark: We could also apply satisfiability modulo theory (SMT) to solve the puzzle [2]

References

[1] Grigorii Samuilovich Tseitin. *Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970*, chapter On the Complexity of Derivation in Propositional Calculus, pages 466–483. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983.

[2] Putranto Utomo. Satisfiability modulo theory and binary puzzle. In *The 2016 International Conference on Mathematics: Education, Theory & Application. December 14, 2016 âĂŞ December 15, 2016*, December 2016. To appear in Journal of Physics : Conference Series.