Counting Knot Mosaics with ALLSAT (Student Abstract)

Hannah Miller

Golisano College of Computing and Information Sciences
Rochester Institute of Technology, Rochester, NY 14623
hm@mail.rit.edu

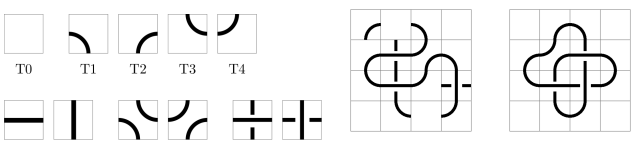
Abstract

Knot mosaics are a model of a quantum knot system. A knot mosaic is a m-by-n grid where each location on the grid may contain any of 11 possible tiles such that the final layout has closed loops. Oh et al. proved a recurrence relation of state matrices to count the number of m-by-n knot mosaics. Our contribution is to compare different ALLSAT solvers as tools for counting knot mosaics and to experimentally evaluate different ways to encode the AT MOST ONE constraint in SAT. We plan to use our SAT method as a tool to list knot mosaics of interest for specific classes of knots.

Introduction and Related Work

A mathematical *knot* is an embedding of a circle in 3D space, and a *link* is two or more knots. Motivated by defining a quantum knot system, Lomonaco and Kauffman (2008) designed the 11 possible tiles required to draw knot mosaics (Figure 1A). Oh et al. (2015) proved a recurrence relation of state matrices to count all knot mosaics of general size $m \times n$. With the aim to develop a toolkit for studying knot mosaics, our work uses ALLSAT solvers to count knot mosaics up to 8×7 boards (inclusive), and we plan to use our ALLSAT encoding as a tool to count and to enumerate specific classes of knot mosaics.

An *n-mosaic* is an $n \times n$ array (or *board*) of mosaic tiles; $m \times n$ arrays are also allowed. A tile's *connection point* is where the tile's curve intersects the midpoint of an edge. A tile is *suitably connected* if each of its connection points touches a connection point of a contiguous neighbor tile. A *knot n-mosaic* is an n-mosaic where all tiles are suitably connected. See Figure 1.



(A) The 11 mosaic tiles (B) Not suitably conn'd (C) Suitably connected

Figure 1: The 11 tiles and two examples of knot mosaics.

Boolean satisfiability (SAT) is the classic NP-complete problem. A SAT formula φ consists of Boolean variables v; logical operations AND ($\wedge$), OR ($\vee$), and NOT ($\neg$); and parentheses. SAT solvers find a *satisfying assignment* to φ by setting every variable to either true or false to make φ true. ALLSAT solvers find all satisfying assignments to φ.

Our Contribution

In contrast to the recurrence relation of Oh et al. (2015), we use ALLSAT solvers to count all of the knots and links (i.e., all possible suitably connected mosaics) up to 8×7 boards. Our contributions are the following: encoding knot mosaics as a SAT formula, trying two ALLSAT solvers (Toda and Soh 2016) on the formula, and comparing different ways to encode the AT MOST ONE constraint for the formula.

SAT Encoding: General

Formula size Let $t = 11$ be the number of tiles. The number of clauses in φ is polynomial in m and n. For example, Equation (1) gives $O(mn)$ clauses of length t. For the general encoding, the number of variables in φ is $O(tmn) = O(mn)$.

Mapping to a variable For an $m \times n$ board with $t = 11$ tiles, we map a board location (i, j) containing tile k to a single variable $v^k_{i,j}$.

Allowed corner tiles and edge tiles To keep the knot suitably connected, we constrain allowed corner tile and allowed edge tiles. Tile 0 is always allowed in a corner; the only other tiles allowed in corners are the arc tiles: T2 in the upper left, T1 in the upper right, T3 in the lower left, and T4 in the lower right; e.g., for tile 2, the clause is $(v^0_{0,0} \vee v^2_{0,0})$.

Allowed neighbors Let A^k_{dir} be the list of allowed neighbors of tile k in each direction (left, right, above, and below); e.g., for location (i, j) with tile k in the left direction, the clause is $\left[\neg v^k_{i,j} \vee \bigvee_{a \in A^k_{\text{left}}} v^a_{i,j-1}\right]$.

EXACTLY ONE Tile Per Board Location We encode AT LEAST ONE and AT MOST ONE; together, these constraints enforce EXACTLY ONE tile per board location.

m	n	# solutions	AT MOST ONE: Pairwise				AT MOST ONE: Sinz SEQ			
			# vars	# clauses	BC time	BDD time	# vars	# clauses	BC time	BDD time
4	6	1,144,526	264	1890	6,400	10	504	1266	8,930	OoR
5	4	54,226	220	1566	8	< 1	420	1046	26	1.61
5	5	4,183,954	275	1974	90,400	< 1	525	1324	–	OoR
5	6	331,745,962	330	2382	OoR	30	630	1602	–	–
6	4	1,144,526	264	1890	8,900	< 1	504	1266	5,440	–
6	5	331,745,962	330	2382	OoT	2	630	1602	–	–
6	6	$> 101 \times 10^9$	396	2874	OoR	85	756	1938	–	–
8	7	$> 47 \times 10^{18}$	616	4518	–	8,580	1176	3062	–	–

Table 1: Solve time results with the AT MOST ONE methods of pairwise and Sinz SEQ as solved with the blocking clause-based solver and the BDD-based solver (*BC* and *BDD*, respectively). The solve time is in seconds and is rounded to the nearest 1, 10, or 100 seconds. *OoR* and *OoT* mean that the solver ran out of RAM (limit was 16 GB) and time (limit was 24 hours), respectively. A dash (–) means that we did not run this combination due to clear performance trends.

SAT Encoding: AT LEAST ONE

For all (i, j) locations, the complete encoding for AT MOST ONE tile per board location is

$$\bigwedge_{\substack{0 \le i < m \\ 0 \le j < n}} \left(\bigvee_{0 \le k \le t} v_{i,j}^k \right) \tag{1}$$

where the inner OR encodes that at least one variable (i.e., tile) is true at a given (i, j) location, and the outer AND encodes across all (i, j) locations in the board.

SAT Encoding: AT MOST ONE

Pairwise method For all (i, j) locations, the complete encoding for AT MOST ONE tile per board location is

$$\bigwedge_{\substack{0 \le i < m \\ 0 \le j < n}} \left(\bigwedge_{\substack{0 \le k < t \\ k < \ell \le t}} \left[\neg v_{i,j}^k \vee \neg v_{i,j}^\ell \right] \right) \tag{2}$$

where the pairwise negated terms encode that at most one variable is true. The pairwise method (sometimes called the naive method) is folklore. The pairwise method has no additional variables and $O(t^2 mn) = O(mn)$ clauses, and a SAT solver can decide the pairwise method immediately.

SEQ method Sinz (2005) gives a *sequential unary counter* (SEQ) for encoding general Boolean cardinality constraints. The specific SEQ for $k = 1$ is

$$(\neg v_1 \vee s_{1,1}) \wedge (\neg v_n \vee \neg s_{n-1,1}) \wedge \tag{3}$$
$$\bigwedge_{1 < i < n} \left((\neg v_i \vee s_{i,1}) \wedge (\neg s_{i-1,1} \vee s_{i,1}) \wedge (\neg v_i \vee \neg s_{i-1,i}) \right)$$

where $s_{i,1}$ is a helper variable; $s_{i,1}$ is a partial sum of the first v_i values. The partial sum $s_{i,1}$ is computed sequentially (hence the SEQ name) on unary numbers. For $k = 1$ (the AT MOST ONE case), the SEQ method has $O(n)$ additional variables and $O(n)$ clauses, and a SAT solver can decide the SEQ method by unit propagation.

Results

We used ALLSAT solvers by Toda and Soh (2016): a blocking clause-based solver and a BDD-based solver.[1] Table 1 shows our results. Our counts agree with Oh et al. (2015).

Formulas encoded with pairwise method typically finished for both the BC and BDD solvers, but the BC solver was orders of magnitude slower than the BDD solver. Formulas encoded with the Sinz SEQ method suffered from out-of-RAM issues with the BDD solver.

With only the AT MOST ONE pairwise method, we ran only the BDD solver on boards from 5×5 up to 8×7. The largest board that the BDD solver finished was 8×7, which finished in 8,580 seconds and counted 47,696,523,856,560,453,790 knots (over 47 quintillion knots). The 7×8 formula reached the 24-hour timeout limit without finishing.

Future Work

Our future work includes studying the interesting ALLSAT behavior presented here and extending our encoding to produce specific classes of knots.

Acknowledgments

We thank S. P. Radziszowski for his helpful comments. Partially supported by NSF grant #1819546.

References

Hong, K.; Lee, H.; Lee, H. J.; and Oh, S. 2014. Small knot mosaics and partition matrices. *J. Phys. A* 47(43).

Lomonaco, S. J.; and Kauffman, L. H. 2008. Quantum knots and mosaics. *Quantum Inf. Process.* 7(2): 85–115.

Oh, S.; Hong, K.; Lee, H.; and Lee, H. J. 2015. Quantum knots and the number of knot mosaics. *Quantum Inf. Process.* 14(3): 801–811.

Sinz, C. 2005. Towards an optimal CNF encoding of Boolean cardinality constraints. In *CP*, 827–831. Springer.

Toda, T.; and Soh, T. 2016. Implementing efficient all solutions SAT solvers. *ACM J. Exp. Algorithmics* 21: 1–44.

[1] Solvers at https://www.disc.lab.uec.ac.jp/toda/code/allsat.html