Time-Reversal in Conway's *Life* as SAT

Stuart Bain

Institute of Transport and Logistics Studies
Faculty of Economics and Business
University of Sydney, Sydney, NSW 2006, Australia
stuartb@itls.usyd.edu.au

Abstract. This paper describes a translation of the time-reversal problem in *Life* to propositional satisfiability. Two useful features of this translation are: that the encoding is linear (in both variables and clauses) with respect to the number of cells in the original problem; and, it can be used to generate problem instances that are known *a priori* to be satisfiable. The problem is shown to have statistically defined hard regions where instances are on average more difficult than in other regions.

1 Introduction

Conway's *Life* [1] is perhaps the most widely known of all cellular automata. Whilst determining the successor of a *Life* pattern is a polynomial-time procedure, determining the precursor of a pattern is significantly more challenging.

The development of SAT solvers suitable for 'real-world' problems benefits from ready access to structured problem instances. As genuine instances are often in short supply, may be cumbersome to distribute, or subject to commercial priviledge, structured problem generators therefore serve an important function in solver development and testing. A useful property of a candidate problem is whether it can be used to generate instances that are known to be satisfiable in advance. This will be shown to be easily enforceable in *Life* time-reversal.

The focus of this paper is that the *Life* time-reversal problem can be used to generate challenging, structured, propositional satisfiability instances for use in the development and testing of SAT solvers.

2 Problem Description

Life operates using an infinite square grid of cells, where at each time-step, each cell is either alive or dead. The state of a cell depends exclusively on the state of itself and its neighbours in the immediately preceding time-step. Each cell has exactly 8 neighbours, being the surrounding cells in the horizontal, vertical and diagonal directions. The rules in *Life* may be described succintly as follows: A cell will be dead in the following time-step unless it currently has either exactly 3 live neighbours or exactly 2 live neighbours and is presently alive itself. Time-reversal in *Life* can be therefore be specified as a decision problem as follows:

Does the specified pattern of cells possess at least one precursor?

M.A. Orgun and J. Thornton (Eds.): AI 2007, LNAI 4830, pp. 614–618, 2007.
© Springer-Verlag Berlin Heidelberg 2007

3 Problem Encoding

Although *Life* is generally considered to operate using an infinitely large grid of cells, it is possible to restrict the domain of the problem to a finite grid. Given the specification of the state of the grid cells $(1,1)$-(x,y) (the inner cells) at time t_1, it is possible to consider their time-reversal by considering not an infinite grid, but only the cells $(0,0)$-$(x+1,y+1)$ (the outer cells). As cells in *Life* can exist in only two possible states, SAT is a natural method for encoding such a problem, since the state of each cell can be represented by a single SAT variable.

The following notation is used to specify the clauses of the SAT encoding. $\bigwedge$ and $\bigvee$ are non-binary, set versions of the conjunction and disjunction operators respectively. Sets are written in bold face. Each rule defines a propositional formula (in conjunctive normal form), for a specified cell x and its corresponding neighbour set $\mathbf{n}$. $\mathcal{P}_i(\mathbf{A})$ is defined to be a variation of the power set axiom, which returns all subsets of $\mathbf{A}$ having a cardinality of i. Overlining denotes a literal or set of literals that is negated in the formula. All literals refer to the state of a cell at time t_0. The overall SAT formula is formed by the conjunction of clauses from rules 1-3 for cells live at t_1, and rules 4-5 for cells dead at t_1.

(1) Loneliness: A cell with fewer than 2 live neighbours (at least 7 dead neighbours) at time t_0 is dead at time t_1, irrespective of its own state at t_0. *(2) Stagnation*: A dead cell with exactly two live neighbours at time t_0 will still be dead at time t_1. *(3) Overcrowding*: A cell with four or more live neighbours at time t_0 will be dead at time t_1 irrespective of its own state at t_0. *(4) Preservation*: A cell that is alive at time t_0 with exactly two live neighbours will remain alive at time t_1. *(5) Life*: A cell with exactly 3 live neighbours at time t_0 will be alive at time t_1, irrespective of its prior state. Formally:

$$\mathit{Loneliness}\,(x,\mathbf{n}) = \bigwedge_{\mathbf{c}\in\mathcal{P}_7(\mathbf{n})} \left(\bigvee \mathbf{c}\right) \tag{1}$$

$$\mathit{Stagnation}\,(x,\mathbf{n}) = \bigwedge_{\mathbf{c}\in\mathcal{P}_2(\mathbf{n})} \left(x \vee \bigvee \overline{\mathbf{c}} \vee \bigvee (\mathbf{n}-\mathbf{c})\right) \tag{2}$$

$$\mathit{Overcrowding}\,(x,\mathbf{n}) = \bigwedge_{\mathbf{c}\in\mathcal{P}_4(\mathbf{n})} \left(\bigvee \overline{\mathbf{c}}\right) \tag{3}$$

$$\mathit{Preservation}\,(x,\mathbf{n}) = \bigwedge_{\mathbf{c}\in\mathcal{P}_2(\mathbf{n})} \left(\overline{x} \vee \bigvee \overline{\mathbf{c}} \vee \bigvee (\mathbf{n}-\mathbf{c})\right) \tag{4}$$

$$\mathit{Life}\,(x,\mathbf{n}) = \bigwedge_{\mathbf{c}\in\mathcal{P}_3(\mathbf{n})} \left(\bigvee \overline{\mathbf{c}} \vee \bigvee (\mathbf{n}-\mathbf{c})\right) \tag{5}$$

4 Instance Generation and Evaluation

Two methods of generating random instances of the $Life$ time-reversal problem were examined, both of which generate a mix of satisfiable and unsatisfiable instances.

The first method involves choosing which cells are live at time t_1 entirely at random. The second method distributes live cells in a balanced way, by making cells live in a regular order and then randomly permuting the rows and columns of the grid to create different instances. This method has also been used to create balanced instances of the quasigroup with holes problem [2].

Problem sizes (inner grid dimensions) ranged from 15 through 25, using a square grid so that $x = y$. The percentage of live cells was varied from 4% to 96% in steps of 4%. 100 different instances were created for each parameter combination. Run-times are for a Sun UltraSPARC-III 900MHz computer.

4.1 Basic Results

The difficulty of the SAT encoding of both the random and balanced instances was empirically examined using two different complete solvers: MiniSAT version 1.14.1 [3] and zChaff version 2004.5.13 [4]. The run-time profiles for each solver were similar (so only the results for the more efficient MiniSAT are presented).

Plots of the mean run-time on instances of both types are shown in Fig. 1. The median run-times do not differ substantially from the mean times so are omitted here, but will be presented for the balanced case in the following section.

These figures show that there are two distinct 'hard' regions, located respectively at 48% and 88% live cells. Whilst obvious in the larger instances, smaller instances do not tend to exhibit these peaks; their peak occurs in the valley between the two peaks instead. Also of interest is that the location of these regions does not depend greatly on the method of assignment of live cells (balanced or random).

What explanation can be given for the observed problem difficulty? In other SAT problems, hardness peaks are often associated with a phase transition

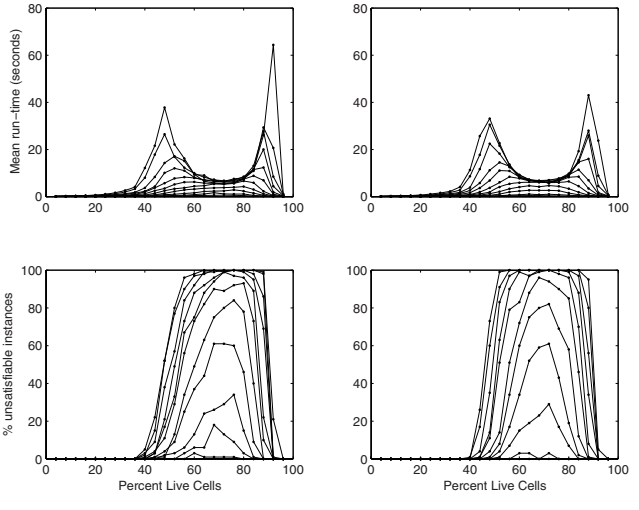

Fig. 1. Average time required to determine the satisfiability of problem instances and percent unsatisfiable instances, by number of live cells

between satisfiable and unsatisfiable instances [5]. The percentage of unsatisfiable instances by percent live cells is also shown in Fig. 1. These figure shows that the hardness peaks in the problem do coincide with the transition from mostly-SAT to mostly-UNSAT instances and *vice versa*. The observed hardness peaks occur approximately at the points where only 50% of generated instances are satisfiable, analogous to results for other SAT-encoded problems [2].

4.2 Known Satisfiable Instances

It is possible to generate instances of this problem that are guaranteed to be satisfiable in advance. This is achieved by generating an $(x-2)$-by-$(y-2)$ pattern (as above) and applying the rules of *Life* in the forward-time direction to determine the state of the x-by-y grid at time t_1. Unlike the instances generated previously, in this case outer cells and any cells external to these are explicitly declared to be off at time t_0 (achieved by adding one literal to represent all external cells and a single unit clause with this literal present negatively).

Instances were generated using the balanced method. It can be seen from Fig. 2 that the previous right-most hardness peak coincident with the UNSAT-SAT transition has been eliminated. But the peak on the left remains, albeit shifted to now exhibit greatest difficulty when between 36% and 40% of cells are originally live, depending on whether mean or median times are considered.

Of particular interest is that the difficulty of the known SAT instances is comparable to that of the mixed SAT/UNSAT instances (at each respective peak). It is reasonable to conclude then, that enforcing satisfiability in this problem does not lead to trivially solved instances as occurs with some problems [6].

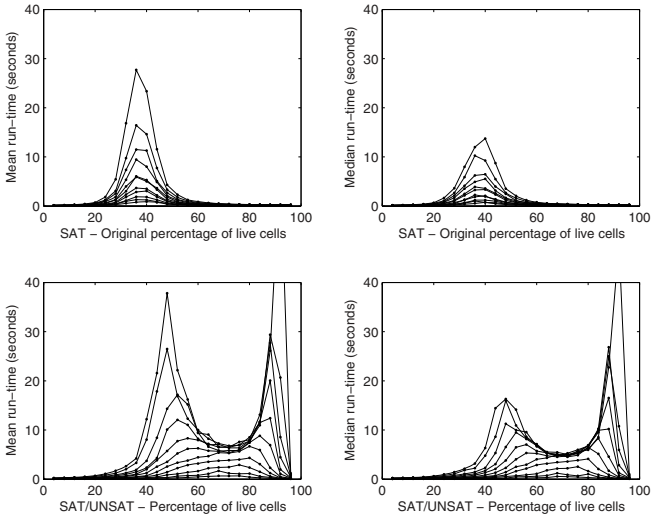

Fig. 2. Comparison of mean/median run-times on SAT vs SAT/UNSAT instances

The run-time distributions for the hardest two parameter settings (those with 36% and 40% of cells originally live) were compared against those of all other settings using the Wilcoxon Rank-Sum test. With the exception of their comparison to each other, the Wilcoxon test confirmed that these two run-time distributions were harder than all other distributions: with strong significance ($\alpha < 0.01$) in all cases for the 36% distribution; and similarly for the 40% distribution but with the single exception of the comparison to the 32% distribution (which was simply significant, $\alpha < 0.05$).

5 Conclusions and Future Work

This paper has presented a method for translating the time-reversal problem of *Life* into SAT. This is a useful structured problem to consider for satisfiability testing, particularly as it can be used to generate known satisfiable instances.

Distributions with both SAT/UNSAT and only SAT instances were examined. In the former case, the most difficult instances were found to be coincident with the phase-transitions from mostly-SAT to mostly-UNSAT instances.

In the latter case however, a hard region still occurs, unrelated to any phase transition phenomena. Perhaps the most pressing questions then are the underlying reasons for the greater difficulty of instances in the hard region of the known SAT instances. Having discounted a phase-transition, a study of backbone size and number of solutions may offer some insight as to their greater difficulty.

The non-parametric Wilicoxon Rank-Sum test was used to confirm that the empirically identified hard distributions were on average more difficult than competing distributions with strong statistical certainty. Since this problem has been shown to have defined hard regions, and the locations of these regions identified, it presents an ideal problem for the testing of SAT solvers.

An extended version of this paper and the instance generator are available from the author's homepage at http://stuart.multics.org

References

1. Gardner, M.: Mathematical Games: The fantastic combinations of John Conway's new solitaire game Life. Scientific American 223, 120–123 (1970)
2. Kautz, H., Ruan, Y., Achlioptas, D., Gomes, C., Selman, B., Stickel, M.: Balance and filtering in structured satisfiable problems. In: IJCAI 2001, pp. 351–358 (2001)
3. Eén, N., Sörensen, N.: An extensible SAT solver. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 502–518. Springer, Heidelberg (2004)
4. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: Engineering an efficient SAT solver. In: Proc. of the 38th Design Automation Conference (2001)
5. Achlioptas, D., Naor, A., Peres, Y.: Rigorous location of phase transitions in hard optimization problems. Nature 435, 759–764 (2005)
6. Achlioptas, D., Gomes, C.P., Kautz, H.A., Selman, B.: Generating satisfiable problem instances. In: AAAI 2000, pp. 256–261 (2000)