Chapter 10
Constraint Programming to Solve Maximal Density Still Life

Geoffrey Chu, Karen Elizabeth Petrie, and Neil Yorke-Smith

The Maximum Density Still Life problem fills a finite Game of Life board with a stable pattern of cells that has as many live cells as possible. Although simple to state, this problem is computationally challenging for any but the smallest sizes of board. Especially difficult is to prove that the maximum number of live cells has been found. Various approaches have been employed. The most successful are approaches based on Constraint Programming (CP). We describe the Maximum Density Still Life problem, introduce the concept of constraint programming, give an overview on how the problem can be modelled and solved with CP, and report on best-known results for the problem.

10.1 Introduction

The Game of Life was invented by John Horton Conway in the 1960s and popularized by Martin Gardner in his *Scientific American* columns (e.g., [8]). Many variants of the game and many problems arising from them have been studied. This chapter describes one beautiful and simple problem that has taxed academics for over a decade, and recounts how it has been solved using the technique of constraint programming.

Problems often consist of choices. Making an optimal choice that is compatible with all other choices made is difficult. *Constraint programming* (CP) [12] is a branch of Artificial Intelligence, where computers help people to make these choices. CP is a multidisciplinary technology combining computer science, operations research, and mathematics. Constraints arise in design and configuration, planning and scheduling, diagnosis and testing, and in many other contexts. This means that constraints are a powerful and natural means of knowledge representation and inference in many areas of industry and academia.

The problem studied in this chapter is the *Maximum Density Still Life* problem. The Maximum Density Still Life fills a finite Game of Life board with a *stable pattern* of cells — one that does not change from generation to generation — that

A. Adamatzky (ed.), *Game of Life Cellular Automata*,
DOI 10.1007/978-1-84996-217-9_10, © Springer-Verlag London Limited 2010

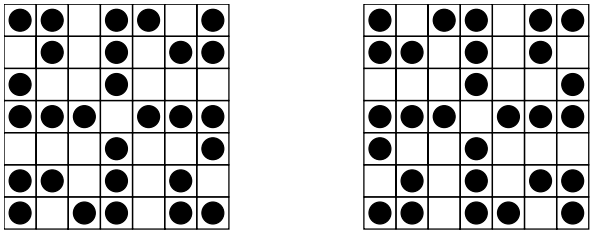

Fig. 10.1 The two optimal solutions for the 7×7 board

has as many live cells as possible. Figure 10.1 shows two optimal solutions for the 7×7 board.

We introduce more fully the Maximum Density Still Life (MDSL) problem (Sect. 10.2), give an introduction to the methodology of constraint programming (Sect. 10.3), and then describe how the MDSL problem can be modelled using constraint programming (Sect. 10.4). We report progress on the problem over a number of years, leading to the current best-known results which have essentially solved the problem (Sect. 10.5).

10.2 Maximum Density Still Life

The *Maximum Density Still Life* (MDSL) problem, or *Still Life* for short, is a variant of the Game of Life. The problem is to find a stable pattern of maximum density [1, 13, 16]. Still Life has been tackled through a variety of approaches, including constraint programming.

The Game of Life is played on a nominally infinite board, constructed of square cells where every cell has eight neighbours. The configuration of live and dead evolves over the generations. The configuration at time t leads to a new configuration at time $t + 1$ according to the three familiar rules of the game:

1. If a cell has exactly 3 living neighbours at time t, it is alive at time $t + 1$. This is the *birth condition*.
2. If a cell has exactly 2 living neighbours at time t, it remains in the same state at time $t + 1$. That is, if dead at time t, it remains dead at time $t + 1$; if live at time t, it remains live at time $t + 1$.
3. If a cell has less than 2 or more than 3 living neighbours at time t, then it is dead at time $t + 1$. These are the *death by isolation* and *death by overcrowding* conditions respectively.

A *stable pattern* (or a *still life*) is a pattern that is not changed from time t to time $t + 1$. The question addressed in this chapter is: On a $N \times N$ section of the board, with all the rest of the board dead (the finite board restriction), what is the densest possible still life pattern?

The *density* of a stable pattern is the ratio of the number of live cells in the board. For example, the optimal solutions for the 7×7 board (Fig. 10.1) have 28 live cells.

They have a density of $\frac{28}{49} = 0.571$. As a formula, the density $d(N)$ for a board of size $N \times N$ is defined by $d(N) = a(N)/N^2$, where $a(N)$ is the number of live cells.

The MDSL problem is distinguished by the finite board restriction and by the search for the maximum density stable pattern. A separate but related concept in the Game of Life is exploring stable patterns of various interesting forms [6, 10].

Especially difficult is to prove that the maximum number of live cells has been found. That is, to prove that the highest density solution found for a given board size N is in fact the *maximal density* $D(N)$ for that size. An *optimal solution* (or optimal stable pattern) of size N is one that attains the maximum density $D(N)$.

An even more difficult problem is to count the number of *non-symmetric optimal solutions* for a given board size. Two optimal solutions are *symmetric* if one can be transformed into the other by rotating the board (through 90, 180, or 270 degrees) or by reflecting it horizontally, vertically, or diagonally. For example, the two optimal solutions for size $N = 7$ (Fig. 10.1) are not unique, as the one on the right is the reflection of the one on the left in the vertical axis. Solving the MDSL problem to *uniqueness* means counting all the unique optimal solutions; it is a different and harder problem to just finding (and proving) the maximal density. For the case of $N = 7$, there is only one unique optimal solution.

10.3 Constraint Programming

Constraints are a powerful and natural means of knowledge representation and inference in many disparate fields, both in industry and academia [12, 15]. Simple illustrative examples include: adjacent countries on the map cannot be coloured the same; the orders assigned to a slab of steel must not exceed the slab's capacity; and a warehouse should only be opened if it serves at least one store. This generality underpins the success with which constraint programming has been applied to disciplines such as scheduling, industrial design, and combinatorics.

A *constraint satisfaction problem* (CSP) is a set of decision variables, each with an associated domain of potential values, and a set of constraints. For example, the problem might be to fit components (values) to circuit boards (decision variables), subject to the constraint that no two components can be overlapping. An assignment maps a variable to a value from its domain. Each constraint specifies allowed combinations of assignments of values to a subset of the variables. A *solution* to a CSP is an assignment to all the variables which satisfies all the constraints. Solutions can be found for CSPs through a process of backtrack search with an inference step at each node.

During the search for a solution of a CSP, constraint *propagation* algorithms are used. These propagators make inferences, recorded as domain reductions, based on the domains of the variables that are constrained. If at any point these inferences result in any variable having an empty domain then search backtracks and a new branch is considered. Through this process of systematic search many problems can be tackled, including Maximum Density Still Life.

10.4 Constraint Programming Models for Still Life

In constraint programming, *modelling* is the process of representing a problem as a CSP. It is straightforward to form a simple CP model based closely on the conditions for a still life. This model has a binary variable x_{ij} for the cell in row i and column j of the $N \times N$ board: $x_{ij} = 0$ if the cell is dead and $x_{ij} = 1$ if it is live.

The three types of constraints echo the three rules of the game:

1. If the sum of the eight neighbouring squares of cell (i, j) equals 3, then $x_{ij} = 1$.
2. If the sum of the eight neighbouring squares of cell (i, j) is less than 2, then $x_{ij} = 0$.
3. If the sum of the eight neighbouring squares of cell (i, j) is greater than 3, then $x_{ij} = 0$.

The above model was proposed by Bosch and Trick [3]. However, the model is inefficient as very little propagation occurs between the variables. This means that solving the problem essentially comes down to an exhaustive search. In order to counter these inefficiencies, Smith [14] re-modelled the problem.

Re-modelling is one of the methods most used by CP practitioners to increase the efficiency of solving a problem. Commonly used practices include introducing *auxiliary variables* which aid propagation, and introducing *implied constraints*. In addition, changing the order in which variables and values are assigned during search can increase search efficiency.

Auxiliary variables are variables which do not need to be in the model for it to be a full representation of a given problem. Their inclusion, along with the inclusion of appropriate constraints embracing these variables, may link search variables that had either no direct constraints connecting them in the basic model, or not enough of a connection for propagation to be forthcoming.

In the basic model for Still Life the constraints between a cell variable x_{ij} and its eight neighbouring variables are not helpful in guiding search. In order to aid this situation Smith [14] suggested looking at the *dual translation* of a CSP with non-binary constraints into a binary CSP. The constraints of the original problem become the variables of the dual CSP; the domain of a dual variable is the set of values that satisfy the original constraints.

In Still Life, the dual variables represent a 3×3 square of cells from the original board, called a *super-cell*. Each super-cell variable y_{ij} corresponds to an x_{ij} cell variable and its eight neighbours. The value of a super-cell variable can be represented as a 9-bit number: $y_{i,j} = x_{i,j} + 2x_{i,j+1} + 4x_{i,j-1} + 8x_{i+1,j} + 16x_{i+1,j+1} + 32x_{i+1,j-1} + 64x_{i-1,j} + 128x_{i-1,j+1} + 256x_{i-1,j-1}$. The domain of y_{ij} is the set of all the allowable values under the original constraints for that super-cell. This domain is different for each of the four corners, the four sides, and the middle of the $N \times N$ board. These variables are added as auxiliary variables to increase the propagation between the underlying x_{ij} cell variables, rather than using them to replace the cell variables. This is a far better model than the simple one previously expressed in this section. However, by using more efficient solving methods and models, as described in the next section, solutions for larger board sizes can be found.

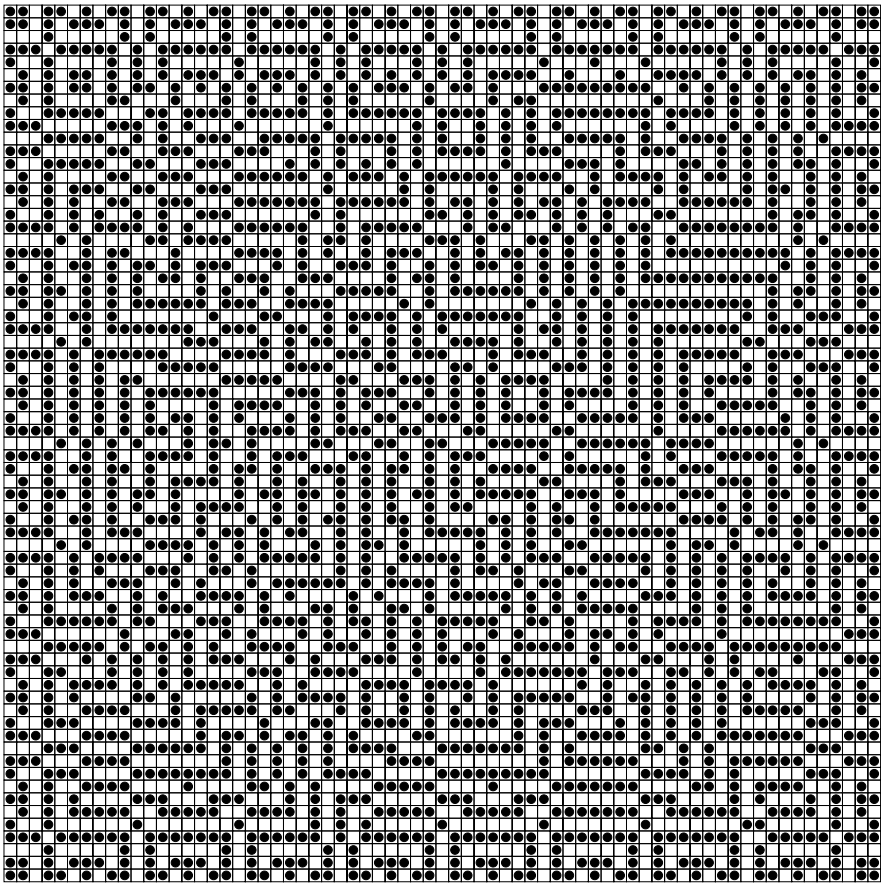

Fig. 10.2 The optimal solution for the 69×69 board (courtesy of [5])

10.5 Solving Still Life

Although simple to state, the MDSL problem is computationally challenging for any but the smallest sizes of board. Especially difficult is to prove that the maximum number of live cells has been found. The raw search space for a maximum density stable pattern on a board of $N * N$ is $\mathcal{O}(2^{N^2})$ in size.

Various approaches have been employed. Elkies [7] relaxed the finite board restriction, and applied graph-theoretic techniques for the case of the infinite board. It is known that the maximal density converges to $\frac{1}{2}$ as N tends to infinity.

The first systematic approach for the full, finite-board problem was by Bosch [1, 2], who employed *Integer Programming* (IP) techniques to the problem. The optimal solutions up to size $N = 7$ were already known within Game of Life folklore (Fig. 10.1) [2, 13, 16]. Bosch found and proved the maximal solution for board sizes up to $N = 10$.

Bosch and Trick extended this approach to a hybrid of IP and CP [3]. Their most successful approach found and proved the maximal density for board sizes up to $N = 15$.

Smith [14] developed a pure CP approach in which the straightforward CP model was reformulated as a *dual* representation of 3×3 'super-cells', as described earlier. Finding all unique, i.e., non-symmetric, solution patterns was pursued further by Petrie et al. [11]. They extended Smith's modelling approach with the IP–CP hybrid of Bosch and Trick, to solve to uniqueness the problem up to board size $N = 10$.

Larrosa et al. [9] employed a space-intensive method in order to reduce the time complexity to $\mathcal{O}(N^2 2^{3N})$. Their most successful approach, which combined *bucket elimination* and search, found and proved the maximal density for board sizes up to $N = 20$.

Chu et al. [5] returned to a pure CP approach. They reformulated the problem in terms of *wastage*. Instead of counting how many live cells there are in a partial solution, they count how much space has been 'wasted' by not packing in live cells tightly enough. This perspective enabled strong upper bounds to be obtained based on a theoretical analysis.

By combining powerful lookahead techniques based on static relaxations of the problem with selected incomplete search, Chu et al. [4, 5] solved or nearly solved[1] board sizes up to $N = 50$. These results have been extended, obtaining and proving the maximum density of boards for all N to within a window of 1 live cell. These best-known results for small N are given in Table 10.1.

10.5.1 Density as Board Size Increases

Although the maximal number of live cells $A(N)$ follows no obvious pattern as the board size N increases, the *wastage count* $W(N)$, as defined in [5], has been proven to grow linearly asymptotically with N. The two are related by $A(N) = \frac{N^2}{2} + N - \frac{W(N)}{4}$. Hence the maximal density $D(N)$ is of order $\frac{1}{2} + \frac{1}{N} - \frac{1}{4N}$. This formula implies that density will asymptotically become $\frac{1}{2}$ as N tends to infinity, just as predicted by theory [7], as Fig. 10.3 shows.

The optimal solutions for MDSL for large N fall into 54 modulus classes [4]. All board sizes for $N \leq 54$ are covered in the results given in Table 10.1. For $N \geq 55$, the asymptotic formula for $A(N)$ is proven to be

$$\left\lfloor \frac{N^2}{2} + \frac{17}{27}N - 2 \right\rfloor \leq A(N) \leq \left\lceil \frac{N^2}{2} + \frac{17}{27}N - 2 \right\rceil \tag{10.1}$$

thus either defining the maximal number of live cells exactly, or bounding it to within 1 cell.

[1] The found upper bound is either attained (i.e., instance is solved), or is attained by one fewer live cell (i.e., instance solved up to ± 1).

Table 10.1 Best known results for MDSL (from [4, 5]). *Bold* denotes cases which are not completely solved at the time of writing

N	Lower	Upper	*N*	Lower	Upper	*N*	Lower	Upper
1	0	0	21	232	232	41	864	864
2	4	4	22	253	253	42	907	907
3	6	6	23	276	276	43	**949**	**950**
4	8	8	24	301	301	44	993	993
5	16	16	25	326	326	45	1,039	1,039
6	18	18	26	352	352	46	**1,084**	**1,085**
7	28	28	27	379	379	47	**1,131**	**1,132**
8	36	36	28	**406**	**407**	48	1,181	1,181
9	43	43	29	437	437	49	1,229	1,229
10	54	54	30	466	466	50	1,280	1,280
11	64	64	31	497	497	51	1,331	1,331
12	76	76	32	530	530	52	**1,382**	**1,383**
13	90	90	33	563	563	53	1,436	1,436
14	104	104	34	598	598	54	**1,489**	**1,490**
15	119	119	35	632	632	55	1,545	1,545
16	136	136	36	668	668	56	1,602	1,602
17	152	152	37	706	706	57	**1,658**	**1,659**
18	171	171	38	**743**	**744**	58	1,717	1,717
19	190	190	39	782	782	59	1,776	1,776
20	210	210	40	824	824	60	**1,835**	**1,836**

10.6 Conclusion

Solving instances of the Maximum Density Still Life problem has taxed researchers in combinatorial optimization for more than a decade. New insights combined with powerful CP techniques have now almost entirely solved the MDSL problem, proving the maximum density to within 1 cell for all N and allowing arbitrarily large optimal solutions to be constructed (Fig. 10.2).

The Still Life problem has stimulated developments in hybrid techniques, symmetry breaking, and constraint programming. The advances in known solutions have come more from improved approaches than from increased computational power. Once more, the complexity and richness underlying the simplicity of Conway's Game of Life is seen.

Acknowledgements For helpful comments on earlier versions of parts of this chapter, we thank particularly Ian Gent, Mike Trick, Peter van Beek, and especially Barbara Smith. Peter J. Stuckey prepared the solution for the 69×69 board with help from Michael Wybrow. Parts of the work described were performed when some of the authors were at the University of Huddersfield and Imperial College London.

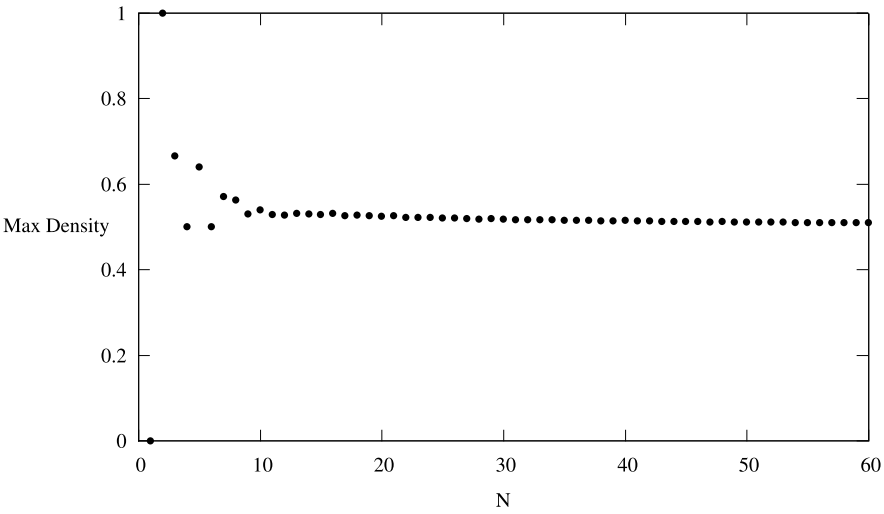

Fig. 10.3 Density converges to $\frac{1}{2}$ as the board size increases

References

1. Bosch, R.: Integer programming and Conway's Game of Life. SIAM Rev. **41**(3), 596–604 (1999)
2. Bosch, R.A.: Maximum density stable patterns in variants of Conway's Game of Life. Oper. Res. Lett. **27**, 7–11 (2000)
3. Bosch, R., Trick, M.: Constraint programming and hybrid formulations for three Life designs. Ann. Oper. Res. **130**, 41–56 (2004)
4. Chu, G., Stuckey, P., de la Banda, M.G.: A complete solution to the Maximum Density Still Life problem (in preparation)
5. Chu, G., Stuckey, P., de la Banda, M.G.: Using relaxations in Maximum Density Still Life. In: Proc. of Fifteenth Intl. Conf. on Principles and Practice of Constraint Programming (CP'09), pp. 258–273 (2009).
6. Cook, M.: Still life theory. http://www.paradise.caltech.edu/~cook/Warehouse/index.html. Accessed September 2005
7. Elkies, N.D.: The still-life density problem and its generalizations. In: Voronoi's Impact on Modern Science, Book I. Proc. of the Institute of Mathematics of the National Academy of Sciences of Ukraine, vol. 21, pp. 228–253 (1998)
8. Gardner, M.: The fantastic combinations of John Conway's new solitaire game. Sci. Am. **223**, 120–123 (1970).
9. Larrosa, J., Morancho, E., Niso, D.: On the practical use of variable elimination in constraint optimization problems: 'still-life' as a case study. J. Artif. Intell. Res. **23**, 421–440 (2005)
10. McIntosh, H.V.: Life's still lifes. http://delta.cs.cinvestav.mx/~mcintosh/newweb/marcostill.html (September 1988). Accessed September 2005
11. Petrie, K.E., Smith, B.M., Yorke-Smith, N.: Dynamic symmetry breaking in constraint programming and linear programming hybrids. In: Proc. of 2nd European Starting AI Researcher Symposium (STAIRS'04), pp. 96–106 (2004)
12. Rossi, F., P. van Beek, Walsh, T.: Handbook of Constraint Programming. Foundations of Artificial Intelligence, vol. 2. Elsevier, New York (2006)
13. Silver, S.: Dense stable patterns. http://www.argentum.freeserve.co.uk/dense.htm. Accessed September 2005

14. Smith, B.M.: A dual graph translation of a problem in 'Life'. In: Proc. of Eighth Intl. Conf. on Principles and Practice of Constraint Programming (CP'02), pp. 402–414 (2002)
15. Wallace, M.G.: Practical applications of constraint programming. Constraints **1**(1/2), 139–168 (1996).
16. Weisstein, E.: Density. In: Eric Weisstein's Treasure Trove of Life C.A. http://www.ericweisstein.com/encyclopedias/life/Density.html. Accessed September 2005