Dynamic Symmetry Breaking in Constraint Programming and Linear Programming Hybrids: Still Life as a Case Study

Karen E. Petrie, Barbara M. Smith
University of Huddersfield, Huddersfield HD1 3DH, U.K.
{k.e.petrie,b.m.smith}@hud.ac.uk

Neil Yorke-Smith
IC-Parc, Imperial College London, London SW7 2AZ, U.K.
nys@icparc.ic.ac.uk

Abstract. The efficient solving of many CSPs has been achieved by hybrid methods that combine Constraint Programming (CP) and Linear Programming (LP). One such case is the 'Maximum Density Still Life' problem. An obstacle to overcome in order to solve this problem efficiently is the inherent symmetry.

Symmetry in Constraint Satisfaction Problems (CSPs) can lead to redundant search, since subtrees may be explored which are symmetric to subtrees previously explored. In CP symmetry can be excluded by Symmetry Breaking During Search (SBDS), a dynamic method that adds constraints during search so that partial assignments symmetric to those already considered will not be visited.

We outline a novel integration of SBDS with CP-LP hybrids which allows the 'Maximum Density Still Life' problem to be solved efficiently. We show that CP modelling techniques which improve the performance of the CP-LP hybrid can also be integrated with SBDS, possibly increasing their efficiency still further.

1 INTRODUCTION

Constraint Satisfaction Problems (CSPs) are often highly symmetric. Given any solution, there are others which are equivalent in terms of the underlying problem being solved. Symmetries may be inherent in the problem, as in placing queens on a chess board that may be rotated and reflected. Additionally, the modelling of a real problem as a CSP can introduce extra symmetry: problem entities which are indistinguishable may in the CSP be represented by separate variables leading to $n!$ symmetries between n variables.

Proceedings of the 2nd European Starting AI Researcher Symposium (STAIRS'04), held jointly with ECAI'04, Valencia, Spain, August 2004. ©2004 IOS Press.

Definition of Symmetry *Given a CSP P, with a set of constraints C, a symmetry of P is a bijective function f such that:*

1. *Given A, any partial or full assignment of P, then if A satisfies the constraints C, then so does $f(A)$.*
2. *Similarly, if A is a nogood, then so too is $f(A)$.* [14]

Symmetries can give rise to redundant search, since subtrees may be explored which are symmetric to subtrees already explored. To avoid this waste, constraint programmers try to exclude all but one in each equivalence class of solutions. In recent years this has been an active research topic, with practitioners experimenting with the addition of constraints and remodelling the problem to remove symmetry. These static methods operate before search commences, but it is also possible to eliminate symmetries dynamically during search as introduced by Brown *et. al.* [3].

Symmetry Breaking During Search (SBDS) is such a dynamic method. This approach was introduced by Backofen and Will [1] and further developed by Gent and Smith [11]. The search tree is built from decision points. When a decision point is first reached during search, a value is assigned to a variable ($var = val$); if at a later stage in search the decision point is revisited, then a constraint is imposed that the variable should not have the previously assigned value ($var \neq val$). SBDS takes a list of symmetry functions, provided by the user, and places related constraints. If A is the current partial assignment and it has been established that $var \neq val$, then a check is performed to ensure that $f(A)$ still holds, hence the symmetry is so far unbroken. Then the constraint $f(var \neq val)$ is placed to ensure that the symmetrically equivalent subtree to the current subtree will not be explored. SBDS has been implemented for constraint systems such as ECL^{i}PS^e [4].

The symmetries of a problem form a group, and SBDS requires a function for each group element other than the identity. Although SBDS has been successfully used with a few thousand symmetry functions, many problems have too many symmetries to permit a separate function for each. To allow SBDS to be used in such situations, Gent *et. al.* [9] have linked SBDS in ECL^{i}PS^e with GAP (Groups, Algorithms and Programming) [7], a system for computational algebra and in particular computational group theory. GAP-SBDS allows the symmetry group rather than its individual elements to be described. GAP is used when a value is assigned to a variable at a decision point to find the *stabiliser* of the current partial assignment, i.e. the subgroup which leaves it unchanged. Then if the decision point is revisited on backtracking, the constraints are dynamically calculated from the stabiliser and placed accordingly. GAP-SBDS allows the symmetry to be handled more efficiently than in SBDS; the elements of the group are not explicitly created. However, there is an overhead in communication necessitated between GAP and ECL^{i}PS^e.

Linear Programming (LP) optimises a linear objective function subject to linear inequality/equality constraints. When used in a branch-and-bound framework to solve integer linear programs, the LP relaxation at each search node provides bounds on the objective value which can be used to prune the search. Constraint Programming (CP) can model problems which cannot easily be expressed using linear constraints, but it has no equivalent of the bounding information provided by LP. Hybrids of CP and LP aim to provide those bounds, while not restricting the scope of CP.

Symmetry can cause problems in LP as in CP: *"Given any solution to a traditional model for such a problem, several equivalent symmetric solutions can be obtained by simply rein-*

dexing these indistinguishable objects. As a result, the branch-and-bound/cut procedure can get hopelessly mired by being forced to explore and fathom symmetric reflections of various solutions during the search process." [18].

To overcome this problem, LP practitioners have explored the static symmetry breaking method of adding constraints to the model [18]. This technique has also been integrated with CP-LP hybrids [2]. However, symmetry breaking constraints can be difficult to derive or can be too complicated to be useful. Simple constraints may break only a proportion of the symmetry, as Bosch and Trick found when studying the 'Maximum Density Still Life' problem [2]. Partial symmetry breaking leaves scope for wasted search and will not yield just the non-isomorphic solutions.

In this paper we show how SBDS and related dynamic methods can be successfully integrated into a full CP-LP hybrid, and a hybrid where LP solves the constraints while CP controls search. The latter combination shows that CP symmetry breaking methods can be used with stand-alone linear programs. Dynamic symmetry breaking in the form of Symmetry Breaking via Dominance Detection (SBDD) has been used with a CP-LP hybrid in the past [17], but in this case SBDD only directed search within the CP mechanism, having very little effect on the LP side. SBDS by its very nature of placing constraints to break symmetry provides more of an integration into the complete CP-LP hybrid. We perform a case study of our techniques on the 'Maximum Density Still Life' problem. We show that CP modelling techniques often used to improve the performance of a pure CP approach, such as introducing auxiliary variables and changing the search heuristic, can also be integrated with the CP-LP-SBDS hybrid possibly increasing its efficiency still further.

2 INTEGRATING CP-LP HYBRID WITH SBDS AND GAP-SBDS

Optimisation problems are commonly solved in CP using *branch-and-bound*. The problem is solved using a standard CP solving technique; then a constraint is added requiring the objective value to be better than the incumbent. Search then recommences, which can be done by restarting or by continuing from the current location. This process continues until no better objective value can be found, at which point the last value found is proven optimal.

Integrating linear programming into this process helps increase the efficiency of search. Before the constraint solver commences search, it invokes the LP solver. This call provides a bound for the optimal objective value. This bound is set as a constraint by the CP solver. Then the constraint program initiates search as normal, with the linear solver providing further constraints. During search, the LP solver is called following constraint propagation to recalculate the bound on the objective given the current partial assignment. If this new objective value is worse than the current optimal objective value then the latest assignment fails and search backtracks. The linear solver can be triggered due to any aspect of the constraint program. Common triggers are a variable becoming ground (instantiated by search or by propagation) or the bounds of a variable's domain changing. Which calling mechanism is used depends on the nature of the problem; the extra information gained must be balanced with the overhead of invoking the LP solver. Only the linear constraints in the model are passed to the LP solver: not all of the problem constraints need be easily expressible in this way, and indeed it may be more efficient not to translate some of the problem constraints into linear form.

It is conceptually not difficult to integrate SBDS into a CP-LP hybrid to deal with problems with symmetry. In CP, SBDS places constraints to eliminate subtrees symmetrical to

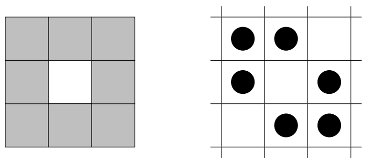

Figure 1: A cell with 8 neighbours and a 3×3 Still Life

those already unsuccessfully explored on backtracking. In a CP-LP method backtracking may happen earlier in the search tree than with a pure CP method, due to the better bounds on the optimum provided by the LP. This means SBDS may be triggered earlier in the search tree than with a pure CP implementation.

When SBDS places constraints in our CP-LP-SBDS implementation they are only added directly to the CP solver, but if propagating these constraints affects the problem variables in a way related to the LP calling mechanism, it will be triggered. The LP solver will acknowledge the updated domains of the variables, but it does not need to be aware of which factor has caused this change. Thus, the LP solver and the associated model along with the CP triggering mechanism do not need to be modified in any way by the inclusion of SBDS. This means *any* CP dynamic symmetry breaking method can in principle be used with a CP-LP hybrid, without necessitating any changes to the LP. The changes to the constraint program are also minimal: no changes are needed to the model since only the search part of the program is affected by SBDS.

3 STILL LIFE

The 'Maximum Density Still Life' problem or 'Still Life' for short is a variant of the game of 'Life', invented by John Horton Conway, and popularised by Martin Gardner in his *Scientific American* columns (e.g. [8]). It has been studied in both a CP-LP context where different hybrid formulations were tested [2] and a CP context with the use of SBDS [19]: these past results give a good test bed for our CP-LP-SBDS method. Life is played on a nominally infinite board, constructed of square cells where every cell has 8 neighbours, as shown in Figure 1. The configuration at time t leads to a new configuration at time $t + 1$ according to the rules of the game:

- If a cell has exactly 3 living neighbours at time t, it is alive at time $t + 1$. This is the 'birth condition'.
- If a cell has exactly 2 living neighbours at time t, it remains in the same state at time $t + 1$.
- 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 *still life* is a pattern that is not changed from time t to time $t+1$. The question addressed in this paper is: On a $N \times N$ section of the board, with all the rest of the board dead, what is the densest possible still life pattern?

The best results to date on this problem were achieved by Larrosa and Morancho using soft constraints and bucket elimination [12].[1] We use Still Life as a case study of symmetry breaking in hybrid models; our purpose here is not to provide new results for the problem.

[1]Known results are listed at: `www.icparc.ic.ac.uk/~nys/life/`.

Table 1: Results for the CP-LP-SBDS hybrid. Shown are the time (ms) and the number of backtracks (BT) needed to find the maximum density still life on a $N \times N$ board using CP and CP-LP, either with no symmetry breaking, with SBDS, or with GAP-SBDS. *Opt* is the optimal value.

		No Sym				SBDS				GAP-SBDS	
		CP		CP-LP		CP		CP-LP		CP-LP	
N	*Opt*	time	BT	time	BT	time	BT	time	BT	time	BT
4	8	50	26	50	6	30	6	40	2	341	2
5	16	120	66	30	1	80	19	30	1	311	1
6	18	5020	1758	2250	214	1410	291	800	305	1256	61
7	28	36350	11313	350	21	10540	2148	360	18	669	18
8	36	850510	246021	2970	203	211230	43289	1651	100	2376	104
9	43	-	-	196930	7407	-	-	70600	2630	83057	2647
10	54	-	-	810260	27612	-	-	328150	10835	385005	10866

It is worth noting though, that remodelling a problem for more efficient solving may take a domain expert, whereas SBDS can be integrated by the non-expert CP practitioner. Another study of symmetry in Still Life is [16], which looks at pseudo-symmetry in the problem.

A simple CP 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 alive. The constraints are then: If the sum of the eight neighbouring squares of cell (i, j) equals 3, then $x_{ij} = 1$; if it is less than 2 or greater than 3 then $x_{ij} = 0$. This formulation is modelled in ECL^{i}PS^e; it is the same as that modelled in OPL by Bosch and Trick in [2].

Bosch and Trick also give a hybrid formulation. They found that adding just the 'death by overcrowding condition' to the model as a linear constraint gave an efficient hybrid. If $N(i, j)$ is the set of neighbours of x_{ij}, then

$$3x_{ij} + \sum_{(i'j') \in N(i,j)} x_{i'j'} \leq 6 \tag{1}$$

enforces this condition.

The Still Life problem has the seven rotation and reflection symmetries of a chess board as described in [11]. These are to rotate the board through $90°$, $180°$ or $270°$ and to reflect the pattern across the horizontal, vertical or diagonals. Bosch and Trick [2] successfully eliminated some of the symmetry by the introduction of static constraints, but were unable to eliminate it all. To use our CP-LP-SBDS method on this CP-LP hybrid a symmetry function is written for each of the symmetries. These functions take a variable and a value then perform a transformation on them to output the symmetrically equivalent variable and value [11]. To interface with GAP-SBDS rather than SBDS, the group for the seven symmetries plus the identity is given instead.

All experiments in this paper use ECL^{i}PS^e 5.7 as the constraint solver; the inbuilt branch-and-bound library is used to control search along with the SBDS library [4]. The LP solver is XPRESS-MP version 14.2 [5]. The empirical tests were undertaken on a Pentium 4 1.6MHz processor. A timeout of 1 hour (3,600,000ms) was imposed.

The result of these first experiments are shown in Table 1. CP does not find the solutions to $N = 9$ and $N = 10$ in under an hour: although the CP-LP hybrid can, and including SBDS reduces the search effort further. Comparing the CP-LP method with no symmetry breaking to our CP-LP-SBDS method shows a decrease in both search effort and time of a factor of more than 2 for $N > 7$. As expected, this is not as high as the factor of 4 reduction in time shown by the similar CP comparison; this is because the search tree is reduced by the CP-LP hybrid, giving less backtracks and hence less scope for SBDS to operate.

Comparing the GAP-SBDS implementation we see that, although it provides a reduction in the amount of search and time needed (as opposed to no symmetry breaking), it does not provide as much of a reduction as SBDS. Due to the increased overhead, GAP-SBDS is less effective than SBDS on problems with a small symmetry group [15] such as in Still Life. We would expect it to outperform SBDS in a case where the symmetry group was larger, but it is unclear exactly where this cut-off lies. However, since SBDS and GAP-SBDS increase solving efficiency over no symmetry breaking in both the CP and CP-LP cases, the integration of either could be vital in solving a problem within reasonable time.

4 LP WITH SBDS

Integrating LP with SBDS, with CP controlling search, is very similar to integrating CP-LP with SBDS. The main difference is that there are no CP constraints apart from the symmetry breaking constraints added by SBDS. The full linear program is implemented and its constraints control all propagation, but CP is used to build the search tree. Variable to value assignments ($var = val$) are made at decision points in the search tree. After an assignment is made, the LP is triggered to see whether the new assignment is consistent with the current objective value and the linear constraints. If it is consistent then search continues; otherwise backtracking occurs and a constraint of the form $var \neq val$ is placed by the constraint solver. SBDS is initiated to add the symmetrical equivalents of the $var \neq val$ constraint to the constraint solver. These constraints are used to avoid searching branches symmetrical to those already explored. It would be more difficult to integrate dynamic symmetry breaking into LP outwith the CP framework, since it would necessitate a rewrite of the LP solver as shown by Margot [13].

Still Life can be implemented as a pure LP [2]. The board is represented by a binary matrix (x_{ij}) as before. The 'death by overcrowding' condition is constraint (1) shown earlier. The 'death by isolation' condition is:

$$2x_{ij} + \sum_{(i'j') \in N(i,j)} x_{i'j'} \leq 0 \tag{2}$$

Let S be a 3 element subset of $N(i,j)$. Then the birth constraint states that either $x_{i,j}$ is alive or one of the elements of $N(i,j) - S$ is alive:

$$-x_{ij} + \sum_{(i'j') \in S} x_{i'j'} - \sum_{(i'j') \in N(i,j) - S} x_{i'j'} \leq 2 \tag{3}$$

One of these constraints is defined for every subset S, which is approximately N^3 constraints.

Table 2 compares this LP formulation with no symmetry, and the same formulation with the integration of SBDS and GAP-SBDS, results are only shown to $N = 8$ due to a time allowance of an hour being imposed on each instance.

Integrating SBDS into search for this LP model gives a more than 5 fold improvement on time and a greater than 4 fold improvement on search effort (where $N = 8$), over the stand-alone LP formulation. Again, this comparison is for LP propagation where search is

Table 2: Dynamic symmetry breaking for LP with CP search control

	No Sym		SBDS		GAP-SBDS	
N	time	BT	time	BT	time	BT
4	230	4	121	1	419	1
5	260	1	240	1	515	1
6	26850	142	7009	33	7882	34
7	5929	14	4639	12	5617	13
8	151489	360	25560	75	32001	91

controlled by CP. It would be interesting to see how this compares to a standard LP formulation with no symmetry breaking and no CP (i.e. in our case, just using XPRESS-MP). In this case Bosch and Trick found that partial symmetry breaking caused a greater than 2 fold saving in search, so CP-LP-SBDS with full symmetry breaking may prove more effective. GAP-SBDS with its extra overhead does not provide as large a search efficiency improvement as SBDS, the reasons for which are the same as those outlined in Section 3.

5 MODELLING — SUPERCELL VARIABLES

Re-modelling the problem is one of the methods most used by CP practitioners to increase the efficency 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. In this section we study whether the extra effort required to formulate these models is still profitable when using a CP-LP hybrid. Since our CP-LP-SBDS method also integrates symmetry breaking, we look at how this affects the remodelling strategy.

5.1 Auxiliary Variables

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 no direct constraints connecting them in the basic model; or not enough of a connection for propagation to be forthcoming.

Smith [19] has formulated a model for the Still Life problem which is more efficient than the basic model introduced in Section 3. We will base some of our models in this section on it. In the basic model 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 suggests 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 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 *supercell*. Each supercell variable y_{ij} corresponds to an x_{ij} cell variable and its eight neighbours $N(i, j)$ (see Figure 1). The value of a supercell variable can be represented as a 9-bit number: $y_{ij} = x_{i,j} + 2x_{i,j+1} + 4x_{i,j+2} + 8x_{i+1,j} + 16x_{i+1,j+1} + 32x_{i+1,j+1} + 64x_{i+2,j} + 128x_{i+2,j+1} + 256x_{i+2,j+2}$. The domain of y_{ij} is the set of all the allowable values under the

Table 3: Reformulated model with supercell auxiliary variables

	No Sym				SBDS			
	CP		CP-LP		CP		CP-LP	
N	time	BT	time	BT	time	BT	time	BT
4	50	25	50	6	30	6	40	2
5	150	64	30	1	90	19	30	1
6	6340	1689	2510	210	1690	283	870	59
7	48350	10939	390	21	13220	2090	360	18
8	1418880	238513	3210	203	259790	42189	1710	100

original constraints for that supercell. This domain is different for each of the four corners, the four sides, and the middle of the $N \times N$ board.

We add these variables as auxiliary variables to increase the propagation between the underlying x_{ij} cell variables, rather than using them to replace the cell variables. Since the cell variables are still the search variables, no changes need to be made to the CP-LP-SBDS method. The results of these experiments are shown in Table 3.

Comparing the basic model results (in Table 1) for the CP implementation with those in Table 3, it can be seen that the number of backtracks has slightly decreased. This indicates that less search is undertaken because extra propagation occurs. However, the time column shows an increase, indicating the overhead of the extra variables is not compensated for by the increase in propagation. Looking at the hybrid implementation we see very little difference in the number of backtracks. This lack of extra propagation is because the LP is not affected by the auxiliary variables. In guiding search, the LP solver provides more information to the CP solver than the auxiliary variables provide by increased propagation. Including SBDS does not affect this reformulation in any way: there is still the same factor of speed-up between no symmetry breaking and SBDS. If a model to a problem could be found where adding auxiliary variables was effective, the inclusion of SBDS would remain an orthogonal and potentially beneficial factor.

5.2 Changing the Search Variables

In the previous section we introduced the supercell variables (y_{ij}), the duals of the cell variables (x_{ij}). A full instantiation of these variables represents a full instantiation of the board; hence we can make these supercell variables into the search variables. This model is simillar to one implemented by Smith [19]. The constraints remain the same as in the previous model.

This new model necessitates rewriting the SBDS symmetry functions, because the effect of the symmetries on the search variables must be described. These symmetry functions are more complicated than before as they incorporate a change in value. Combining any of the symmetries with the supercell variable gives a new configuration. This new configuration must be summed from top left to bottom right, e.g. for the vertical symmetry this becomes: $f(y_{ij}) = x_{i,j+2} + 2x_{i,j+1} + 4x_{i,j} + 8x_{i+1,j+2} + 16x_{i+1,j+1} + 32x_{i+1,j} + 64x_{i+2,j+2} + 128x_{i+2,j+1} + 256x_{i+2,j}$. These arithmetic expressions are pre-calculated and stored in a look-up table for use at run time.

No changes are made to the LP model. The results of using this model (Table 4) indicate how CP-LP-SBDS operates when the LP formulation is not based on the same variables as the symmetry breaking constraints.

Table 4: Search on supercell variables (with default lex variable ordering)

	No Sym				SBDS			
	CP		CP-LP		CP		CP-LP	
N	time	BT	time	BT	time	BT	time	BT
4	50	18	70	1	130	23	140	3
5	900	185	1060	100	1309	172	1170	89
6	9210	1531	3210	191	10691	1445	3260	174
7	104660	16317	2460	98	115290	15843	3029	116
8	-	-	18230	756	-	-	18020	717

Comparing this model with the model in the previous section (Table 3), it can be seen that the new model requires more backtracks and takes longer. This contrasts to Smith's results in [19]. However, Smith uses the table constraints of ILOG Solver to enforce arc consistency between supercell variables. At the time of writing, table constraints are not yet available in ECL^{i}PS^e [4].

Contrasting the results for CP and CP-LP, it can be noted that the hybrid approach still improves search dramatically. Overall, with the integration of SBDS into CP there is a slight reduction in search, but not by as large a factor as previous experiments. The reason for this is the symmetry functions which have a larger overhead and are placing more complicated constraints which do not provide as large an aid to propagation. This effect is mirrored in CP-LP; coupled with the fact that LP and SBDS are no longer operating on the same variables. This means they do not operate in parallel to prune the search space. If this formulation was used on a problem with a greater number of symmetries, where SBDS had more scope to place symmetry breaking constraints, a greater reduction in search might be expected.

5.3 Variable Ordering

Finally, variable ordering heuristics are often used to increase the efficiency of solving CSPs. The experiments reported so far have been run using the static heuristic of lexicographic ordering. Changing the variables from binary cell variables to non-binary supercell variables with larger domains, gives scope to try a dynamic variable ordering heuristic. A commonly effective heuristic is Smallest Domain First (SDF), which chooses next the variable with the smallest remaining domain. Smith has successfully applied this heuristic to the Still Life problem [19]. If symmetry is broken by the use of constraints, then care has to be taken over the use of variable ordering heuristics; but no such concerns apply with SBDS. The LP formulation is not affected by a change of search order. The results of applying SDF to our reformulated model are in Table 5.

SDF can increase the search effort, as shown by comparing the results in Tables 4 and 5, but sometimes can significantly reduce the number of backtracks. However, even in the latter case the additional overhead of SDF means that there is no time saving. There is a vast improvement again between the CP and CP-LP implementations, showing that the LP solver is not confused by this change in search order. SBDS provides a slight improvement over no symmetry breaking, but not as much of an improvement as earlier experiments.

Table 5: Search on supercell variables with SDF variable ordering

	No Sym				SBDS			
	CP		CP-LP		CP		CP-LP	
N	time	BT	time	BT	time	BT	time	BT
4	50	18	80	1	160	23	150	3
5	900	143	1130	79	1049	127	1180	69
6	118700	877	4010	126	12680	767	3700	109
7	246659	25060	7730	104	203529	23808	6410	107
8	-	-	64860	1402	-	-	44500	943

6 CONCLUSION AND FUTURE WORK

We have presented a method of combining dynamic symmetry breaking with CP-LP hybrids. The hybridization can be a standard CP-LP integration or one in which CP is only used to control search. This latter hybrid provides a method of easily introducing dynamic symmetry breaking into LP. We have shown how a successful implementation operates over both SBDS and GAP-SBDS. For the benchmark ‘Still Life’ problem, our CP-LP-SBDS method has produced a factor of two improvement in both time and search effort over CP-LP with no symmetry breaking. GAP-SBDS and SBDS both improve the efficency of the hybrid method whilst solving this problem, but due to the limited number of symmetries SBDS outperforms GAP-SBDS. Dynamic symmetry breaking can be added without changes to either the CP or the LP formulation.

We then studied how common CP techniques used to increase solving efficiency can be integrated into CP-LP and CP-LP-SBDS. Methods such as introducing auxiliary variables and changing the search heuristic do not give any adverse effects when we move from a CP formulation to a CP-LP hybrid. When SBDS provides a large improvement over no symmetry breaking on the reformulated CP model, it can provide an orthogonal and potentially benefical improvement when integrated with CP-LP. In our experiments the combination of remodelling the CP with CP-LP-SBDS never provided a less efficient program then the reformulated problem and CP-LP alone.

In the future we will apply CP-LP-SBDS to a problem that contains more symmetry, such as the SONET problem [18] where we conjecture we will see a bigger percentage gain in efficency and GAP-SBDS may outperform SBDS. We will also combine CP-LP with other dynamic symmetry breaking methods such as GAP-SBDD [6, 10]. On the LP side we will study what happens if we reformulate the LP model while keeping the CP part of the hybrid constant.

References

[1] R. Backofen and S. Will. Excluding symmetries in constraint-based search. In *Proc. of CP’99*, LNCS 1713, pages 73–87. Springer, 1999.

[2] R. Bosch and M. Trick. Constraint programming and hybrid formulations for three Life designs. In *Proc. of CP-AI-OR’02*, pages 1396–1407, 2002.

[3] C. A. Brown, L. Finkelstein, and P. W. Purdom, Jr. Backtrack searching in the presence of symmetry. In *Proc. of AAECC-6*, LNCS 357, pages 99–110. Springer, 1988.

[4] A. M. Cheadle, W. Harvey, A. J. Sadler, J. Schimpf, K. Shen, and M. G. Wallace. ECLiPSe: An introduction. Technical Report IC-Parc-03-1, IC-Parc, 2003. `www.icparc.ic.ac.uk/eclipse/`.

[5] Dash Optimization. *Xpress-MP 14 Reference Manual*, 2003. `www.dash.co.uk`.

[6] T. Fahle, S. Schamberger, and M. Sellmann. Symmetry breaking. In *Proc of CP'01*, LNCS 2239, pages 93–107. Springer, 2001.

[7] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.2*, 2000. `www.gap-system.org`.

[8] M. Gardner. The fantastic combinations of John Conway's new solitare game. *Scientific American*, 223:120–123, 1970.

[9] I. P. Gent, W. Harvey, and T. Kelsey. Groups and constraints: Symmetry breaking during search. In *Proc. of CP'02*, LNCS 2470, pages 415–430. Springer, 2002.

[10] I. P. Gent, W. Harvey, T. Kelsey, and S. Linton. Generic SBDD using computational group theory. In *Proc. of CP'03*, LNCS 2833, pages 333–347. Springer, 2003.

[11] I. P. Gent and B. M. Smith. Symmetry breaking in constraint programming. In *Proc. of ECAI-2002*, pages 599–603. IOS Press, 2000.

[12] J. Larrosa and E. Morancho. Solving 'Still Life' with soft constraints and bucket elimination. In *Proc. of CP'03*, LNCS 2833, pages 466–479. Springer, 2003.

[13] F. Margot. Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, 94:71–90, 2002.

[14] I. McDonald and B. M. Smith. Partial symmetry breaking. In *Proc. of CP'02*, LNCS 2470, pages 431–445. Springer, 2002.

[15] K. E. Petrie and B. M. Smith. Symmetry breaking in graceful graphs. In *Proc. of CP'03*, LNCS 2833, pages 930–934. Springer, 2003.

[16] S. Prestwich and J. C. Beck. Using pseudosymmetry to reduce search effort. Technical Report TR-01-2004, Cork Constraint Computation Centre, University College Cork, 2004.

[17] M. Sellmann. *Reduction Techniques in Constraint Programming and combinatorial Optimization*. PhD thesis, University of Paderborn, 2002.

[18] H. D. Sherali and J. C. Smith. Improving discrete model representations via symmetry considerations. *Management Science*, 47(10):1396–1407, 2001.

[19] B. M. Smith. A dual graph translation of a problem in 'Life'. In *Proc. of CP'02*, LNCS 2470, pages 402–414. Springer, 2002.