Exploiting Dominance in Three Symmetric Problems

Steven Prestwich and J. Christopher Beck
Cork Constraint Computation Centre
Department of Computer Science, University College, Cork, Ireland
{s.prestwich,c.beck}@4c.ucc.ie

Abstract

Symmetry breaking has led to huge improvements in search performance, and has recently been the subject of considerable research. The related concept of dominance is even more powerful than symmetry, yet it has been relatively unused in Constraint Programming. This paper describes previously unexploited dominances for three well-studied symmetric problems. Experiments show the benefit of adding constraints to exclude both symmetric and dominated solutions.

1 Introduction

A great deal of work has been done by the Constraint Programming (CP) community in the area of detection and exploitation of symmetric search states. Informally, a pair of search states s_i, s_j is symmetric if there is a validity preserving mapping between them. For each extension of s_i (including s_i itself) there is a corresponding state that is an extension of s_j with the same validity. If the extension to s_i is a (non-)solution then so is the corresponding extension to s_j. Obviously this relationship is bi-directional: if s_i is symmetric to s_j then s_j is symmetric to s_i.

Dominance relations are a standard tool in the search for optimal solutions to combinatorial optimization problems. Informally, a dominance relation is a relation on a pair of search states s_i, s_j stating that the best solution that is an extension of s_j is no better than the best extension of s_i. Therefore only s_i needs to be extended. There are two primary differences between dominance relations and symmetry: the cost function of the former and the bi-directionality of the latter. While symmetry has traditionally been applied in satisfaction problems, we can easily model satisfaction problems as having a cost function that is zero for any solution and infinite for a non-solution. The bi-directionality can be dealt with by adding constraints to enforce the anti-symmetric condition of the dominance relation.

Even well-studied problems in the symmetry literature may also contain unexploited dominances that can be used to significantly improve search performance. Proll & Smith [16] used the term *pseudo-symmetry* to denote the same weakening of bi-directionality. They added pseudo-symmetry breaking constraints to improve search in a template design problem. Getoor et al. [12] added domain-specific redundant constraints to remove sub-optimal solutions from online scheduling problems. Gent et al. [10] recently added constraints to the Graceful Graphs problem to exclude dominated solutions. But despite these examples, dominance has been far less exploited than symmetry in CP.

With the aim of stimulating further research in this area, we present three case studies using symmetric problems from the CP literature. Section 2 provides some background. Section 3 studies the Maximum-Density Still Life problem, Section 4 Steel Mill Slab Design, and Section 5 Peaceable Armies of Queens. Section 6 concludes the paper. All our experiments are performed on a 733 MHz Pentium II.

2 Dominance Relations and Symmetry

Ibaraki [14] provides a formal definition of dominance relations to find a single optimal solution.[1] Assuming a standard constructive tree search as in common in CP, let S be the set of all search states, and let $f(s)$ be the minimum cost feasible solution that is an extension of the search state, $s \in S$. If s is infeasible then $f(s) = \infty$. A dominance relation is a binary relation $\preceq$ on search states that satisfies the following conditions:

- $s_i \preceq s_j$ implies $f(s_i) \leq f(s_j)$
- $\preceq$ is a partial ordering: transitive, reflexive, and anti-symmetric
- $s_i \preceq s_j \wedge s_i \neq s_j$ implies that there exists some extension $s_{i'}$ of s_i such that for all extensions $s_{j'}$ of s_j, $s_{i'} \preceq s_{j'} \wedge s_{i'} \neq s_{j'}$

The anti-symmetric requirement means that if $f(s_i) = f(s_j)$ and $s_i \neq s_j$ then only one of $s_i \preceq s_j$ or $s_j \preceq s_i$ can hold. In that case, a simple way to tie-break is to allow $s_i \preceq s_j$ if s_i was found before s_j in the tree search.

As an example of dominance relations, Ibaraki uses the (now) familiar 8-queens where the dominance relation $s_i \preceq s_j$ holds if and only if the patterns represented by the search states are isomorphic and s_i was found before s_j.

[1] Ibaraki presents alternative conditions for finding all optimal solutions, while another condition that we do not discuss here arises from technical aspects of branch-and-bound search.

Because an explicit relation on all pairs of search states is impractical, dominance relations are often based on properties of the search states. That is, a property X is identified and it is proved that search states with X dominate search states without X. For example, in job shop scheduling, solutions which are "semi-active" have been shown to dominate non-semi-active solutions and a common heuristic searches only for semi-active solutions [7]. We will follow this property-based approach.

Suppose that we are able to identify a property P of a search state with the following attribute: if there exists a solution satisfying P then there also exists at least one solution satisfying $\neg P$. This is a one-way relationship because the existence of a solution satisfying $\neg P$ need not imply the existence of a solution satisfying P. However, by only considering solutions satisfying $\neg P$ we can reduce the search space without affecting validity. One way to do this is to add constraints to enforce $\neg P$. We shall call this technique *dominance enforcement*. It is directly analogous to the addition of symmetry breaking constraints to a model [17] and is justified by the following simple theorem:

Theorem. *Dominance enforcement preserves validity.*

Proof. Consider two cases. (i) There is no solution satisfying P. Then adding the constraint $\neg P$ excludes no solutions. (ii) There is at least one solution satisfying P. Then there also exists at least one solution satisfying $\neg P$, which the constraint $\neg P$ does not exclude. QED.

However, this does not prove that combining two dominance constraints is guaranteed to preserve at least one solution. (The same applies to symmetry breaking, for example when breaking both row and column symmetries in a matrix model [8] one must be careful to combine the two sets of constraints in the correct way.) In this paper we shall ignore this important point, as this is a work in progress, but a more formal treatment of the subject would be a useful direction for future work. This might proceed along similar lines to the many recent applications of group theory to symmetry breaking. Other approaches such as Symmetry Breaking During Search (SBDS) [2; 11] might also be generalized to dominance enforcement.

Note that, unlike symmetry breaking, by enforcing dominance we may lose access to the full set of solutions. But this is unimportant for problems in which we are interested in finding any [optimal] solution, or in proving insolubility [optimality].

3 Maximum-Density Still Life

Our first case study is the game of Life, invented by Conway in the 1960s. In an infinite 2-dimensional array, each cell is either alive or dead and has 8 neighbours. The game is initialized by setting each cell alive or dead. Subsequently the array is transformed into a new pattern for as many iterations as desired using a few simple rules: (1) a cell with 2 living neighbours is unchanged in the new pattern; (2) a cell with 3 neighbours is alive in the new pattern; and (3) any other cell is dead in the new pattern. A *still-life* is a pattern that does not change between iterations. A *maximum density* still-life is one with the greatest number of live cells, in a finite square region R of $N \times N$ cells (all cells outside R are dead).

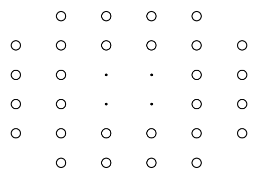
Figure 1: A still-life pattern for dominance enforcement.

3.1 A Basic Model

Two integer program (IP) formulations of this problem were given in [3]. We use the second, better model which has a 0/1 variable x_e for each cell $e \in R$, 1 denoting a live cell and 0 a dead one. The constraints are as follows:

- Death by isolation: $2x_e - \sum_{f \in N(e)} x_f \leq 0$, where $N(e)$ denotes the variables corresponding to the neighbours of e's cell.
- Death by overcrowding: $3x_e + \sum_{f \in N(e)} x_f \leq 6$.
- Birth: $\forall S \subset N(e), |S| = 3 : -x_e + \sum_{f \in S} x_f - \sum_{f \in N(e)-S} x_f \leq 2$.
- Cells outside R cannot become alive: $x_f + x_g + x_h \leq 2$, where f, g and h are 3 cells lying in a line along the boundary of R.
- The density must be at least d: $\sum_{e \in R} x_e \geq d$.

To solve the problem we solve a series of CSPs with increasing d.

3.2 Dominance

Consider the pattern of cells in Figure 1 with $\circ$ denoting dead cells. In any still-life the cells marked $\cdot$ can be all live or all dead. Moreover, if there is a solution of density at least d in which they are all dead then there is also a solution of density greater than d in which they are not all dead. This is the property P we need to apply dominance enforcement constraints: we add a constraint $\neg P$ forcing one of them to be live. However, if one of the four is live they they must all be live, in order to form a still-life, so we can add a stronger constraint: $\sum_{i \in O} 4x_i + \sum_{j \in X} x_j \geq 4$. X denotes the cells marked $\cdot$ while O denotes those marked $\circ$. The pattern may be partly off the edge of the finite region R, as long as the four central cells are inside R. We call these constraints D.

A second dominance is based on the following observation: if there is a solution whose top row contains only dead cells then there is also a solution in which this is not true. We can simply translate the pattern one or more rows upward until a live cell appears in the top row. Therefore we can add a translational dominance enforcement constraint to exclude patterns in which the top row is all dead. Similarly, we can exclude patterns in which the left column is all dead: $\sum_{i=1}^{N} x_{i1} \geq 1$ and $\sum_{j=1}^{N} x_{1j} \geq 1$. We call these constraints T. Note that T are not symmetry breaking constraints: not every pattern with live top row cells can be translated to one whose top cells are all dead.

N	M	M+T	M+D	M+T+D
5 opt	999	877	999	877
proof	709	639	709	639
6 opt	7501	3009	7484	2993
proof	25703	22292	25700	22289
7 opt	161626	145563	161542	145487
proof	159718	149054	159686	149023
8 opt	4893631	4682075	4882021	4672235
proof	3289248	3143639	3282801	3137996

N	M	M+T	M+D	M+T+D
5 opt	<0.1s	<0.1s	<0.1s	<0.1s
proof	<0.1s	<0.1s	<0.1s	<0.1s
6 opt	0.5s	0.2s	0.5s	0.2s
proof	1.4s	1.2s	1.5s	1.3s
7 opt	11s	10s	12s	10s
proof	11s	11s	13s	11s
8 opt	6m2s	5m48s	6m22s	6m34s
proof	4m23s	4m18s	4m40s	4m16s

Figure 2: Still-life results (backtracks and CPU time)

3.3 Results and Discussion

We transform the basic model M, with and without the T and D constraints, to linear pseudo-Boolean form and then apply a simple backtracker to solve the problem. Pseudo-Boolean form is a special form of 0/1 integer program that can be solved by SAT-based algorithms (see for example [1]). The backtracker uses a lexicographical variable ordering (except that a variable whose domain size becomes 1 is immediately assigned) and a value ordering that tries 0 before 1. The results are shown in Figure 2: "opt" is the CPU time or backtracks needed to find an optimal solution and "proof" is the time or backtracks to prove that no denser solution exists (restarting the algorithm with a lower bound equal to the best known value plus 1).

In all cases the dominance enforcement constraints reduce or leave unchanged the required number of backtracks. The D constraints result in little reduction in backtracks while incurring a CPU time overhead. The T constraints, on the other hand, reduce both the backtracks and the CPU time. The results are comparable with the basic CP and IP results of [6] but not as good as their hybrid approach nor other state-of-the-art approaches [15; 19]. Though the improvement due to dominance is small we feel that it is worth reporting, because the geometrically-inspired dominances may inspire more effective versions for this or other problems. (In fact we reuse the idea of translational dominance in Section 5.)

4 Steel Mill Slab Design

Our second case study is a simplified industrial problem. In a steel mill, slabs are produced from molten iron in a finite number of sizes which are later cut to fulfil individual orders. Given a set of orders of certain sizes, we must pack the orders onto the slabs while minimizing the total size of the slabs. In addition, each order is assigned a colour, corresponding to a route through the mill. There is a limit, p, on how many different colours may be assigned to a slab. We denote the S slab sizes by σ_i, the K colours by κ_i, and the O order weights by ω_i.

4.1 A Basic Model

We start from a basic model similar to the IP model of [13], but explicitly model wastage as suggested in [9]. The number of slabs is not known in advance so we assume that there may be as many slabs as orders. A *size variable* s_{ij} is 1 if and only if slab i takes size σ_j. Each slab has up to one size, $\sum_j s_{ij} \leq 1$, and a slab with no size implicitly has size 0, denoting that it is unused. The optimization problem of minimizing the total slab size can be reduced to a series of CSPs with decreasing upper bound U on the size. Each CSP has a total capacity constraint: $\sum_i \sum_j s_{ij}\sigma_j \leq U$. An *order variable* o_{ij} is 1 if and only if order j is assigned to slab i. Each slab capacity must not be exceeded and each order is assigned to exactly one slab:

$$\sum_j o_{ij}\omega_j \leq \sum_k s_{ik}\sigma_k \qquad \sum_i o_{ij} = 1$$

A *colour variable* c_{ij} is 1 if and only if colour j is assigned to slab i. (In some instances not all colours are used, so we relabel them to consecutive numbers.) No more than p colours may be assigned to a slab, and if an order is assigned to a slab then so is its colour: $\sum_j c_{ij} \leq p$ and $o_{ij} \leq c_{i\kappa_j}$. A *wastage variable* e_{il} for $l = 1 \ldots B$ where $B = \lceil \log_2(U - \sum_j \omega_j + 1) \rceil$ is such that the wastage for slab i is bounded by $\sum_{l=1}^{B} 2^{l-1} e_{il}$:

$$\sum_j s_{ij}\sigma_j - \sum_k o_{ik}\omega_k \leq \sum_l 2^{l-1} e_{il}$$

The total wastage is bounded by:

$$\sum_i \sum_l 2^{l-1} e_{il} \leq U - \sum_k \omega_k$$

4.2 Symmetry

When variable sets form matrices, as they do here, a powerful way of breaking symmetry is to impose lexicographical ordering on rows and/or columns [8]. In principle this can be expressed in linear constraints by comparing weighted sums of the form $\sum_i 2^{i-1} x_i$ (assuming binary variables), but the finite word length of a computer makes this impractical for large vectors of variables. Instead we approximately break symmetry by using sums of the form $\sum_i i^2 x_i$ but retain the symmetry breaking ideas of Frisch et al. This technique (which we have not seen used before but do not claim to be original) potentially breaks some symmetry but leaves at least one solution, because at least one weighted sum will be greatest whatever coefficients we choose. But it may not break all symmetries because more than one weighted sum may be equal. In future work we hope to repeat the experiments below using lexicographic ordering.

Orders of the same weight and colour are interchangeable so we approximately order those columns of the order variable matrix: $\sum_i i^2(o_{ik} - o_{ij}) \geq 0$, where $j < k$, $\kappa_j = \kappa_k$

and $\omega_j = \omega_k$. The constraints

$$\left(\sum_l l^2\right) \sum_k k^2 (s_{ik} - s_{jk}) + \sum_l l^2 (o_{il} - o_{jl}) \geq 0$$

order slabs by decreasing size, and where two slabs take the same size force their order vectors to be approximately ordered. The o variables are prevented from interfering with the slab size ordering by the coefficient $\sum_l l^2$.

4.3 Implied Constraints

Consider an implied constraint of Frisch et al. If symmetry breaking constraints are used then the first slab must be large enough to be assigned the largest order μ (though it is not necessarily assigned to the first slab). Furthermore, because of the slab size ordering, order μ must be assigned to a slab $1 \ldots M$ where $M = \min(O, \lfloor U/\omega_\mu \rfloor)$: $o_{i\mu} = 0$ and $i > M$. The upper bound U on the total capacity can be used in implied constraints:

$$\sum_{k \geq i} \sum_j s_{kj}\sigma_j + (i-1) \sum_j s_{ij}\sigma_j \leq U$$

where $2 \leq i \leq O$. Similar constraints impose a lower bound on the total capacity:

$$\sum_{k \leq i} \sum_j s_{kj}\sigma_j + (O-i) \sum_j s_{ij}\sigma_j \geq \sum_k \omega_k$$

4.4 Dominance

We have found four dominances in the steel mill slab design problem. To the best of our knowledge the fourth is new, but the other three could have been incorporated immediately into the model without considerations of dominance. However, they were not used in previous work on this problem, and we aim to show that thinking in dominance terms can lead to them in a natural way.[2]

We use a small example of Frisch et al. as an illustration: the available slab sizes are $\{1, 3, 4\}$, the available colours are $\{$red, green, blue, orange, brown$\}$ and the input orders are shown in Figure 3. It also shows three optimal solutions, the first taken from Frisch et al.

Colour Dominance. Consider a hypothetical solution (not in Figure 3) in which only two orders are assigned to slab 1: 5,6 which are both orange. Assume that $p = 2$ so that each slab can be assigned up to 2 colours. Then we are free to assign another colour such as blue to slab 1 without violating a colour constraint, even though no blue orders are assigned to it. This is a form of dominance: if a solution exists with orders 5,6 assigned to slab 1 which is assigned the colour orange, then there exists a solution with slab 1 *also* assigned the colour blue. To enforce the dominance we add constraints to exclude states in which a colour is assigned to a slab but

[2]These dominances may have been made unnecessary by not branching on the related variables, for example by not branching on colour variables we do not encounter colour-dominated solutions. It is an open question which strategy works best, but by leaving the choice of branching variables free we are able to apply a generic solver that does not provide branching control.

order	1	2	3	4	5	6	7	8	9
weight	2	3	1	1	1	1	1	2	1
colour	R	G	G	B	O	O	O	B	B

Solution 1			Solution 2			Solution 3		
slab	size	orders	slab	size	orders	slab	size	orders
1	4	7,8,9	1	3	8,9	1	3	8,9
2	3	1,3	2	3	1,3	2	3	1,3
3	3	2	3	3	2	3	3	2
4	3	4,5,6	4	3	4,5,6	4	1	4
			5	1	7	5	1	7
						6	1	5
						7	1	6

Figure 3: A small example and three optimal solutions

no order of that colour is: $c_{ik} \leq \sum_{j \in S_k} o_{ij}$, where $S_k = \{j \mid \kappa_j = k\}$. The definition of the colour constraint therefore has been changed from an "if" to an "if-and-only-if".

Wastage Dominance. The $\leq$-definition of wastage can be transformed to an $=$-definition by adding inequalities:

$$\sum_j s_{ij}\sigma_j - \sum_k o_{ik}\omega_k \geq \sum_l 2^{l-1} e_{il}$$

Now the e_{il} give the exact wastage for slab i instead of an upper bound. This is a dominance: if a solution exists in which the wastage is greater than necessary, then there also exists one in which it is exactly the total slab size minus the total order weight. This is not a symmetry as we may not be able to increase a slab wastage without violating the upper bound on total wastage.

Capacity Dominance. Consider any solution in which the size of a slab is larger than necessary: that is, it could be reduced to the next smaller size (or even smaller) while still exceeding or equalling the sum of the weights of its assigned orders. The existence of the wasteful solution implies the existence of the better solution (but not vice-versa) so we can add constraints to exclude the former:

$$\sum_{k=2}^{O} s_{ik}(1 + \sigma_{k-1}) \leq \sum_j o_{ij}\omega_j \qquad s_{i1} \leq \sum_j o_{ij}$$

A slab's size is now a function of its orders: it is the smallest size that is large enough to contain the slab's orders. If no orders are assigned then a slab has size 0.

Cutting Dominance. The second solution of Figure 3 is derived from the first by cutting slab 1 of size 4 into slab 1 of size 3 and slab 5 of size 1. The third solution is derived from the second by cutting slab 4 of size 3 into slabs 4,6,7 each of size 1. These transformations are reversible and could be treated as a symmetry, giving an opportunity for further symmetry breaking by adding constraints such as: $s_{13} + o_{17} + o_{18} + o_{19} \leq 3$. But we may need to enumerate a large number of assignment-size combinations. This situation can be improved by relaxing the symmetry requirement to a dominance as follows. Consider any solution in which slab 1 has size 4 and is assigned orders $\{7\} \cup S$ for some orders S. Then it can be decomposed into two slabs, one of size

			PBS			
O	U_{opt}	Solver	(a)	(b)	(c)	(d)
12	77	0.3	1.89	0.44	0.4	1.59
16	99	18.9	13.2	4.8	3.81	5.43
18	110	226	948	57.9	48.5	45.4
19	115		1493	36.9	141	7.00
20	122		2683	394	10.5	54.1
21	135		246	801	0.74	1085
25	166		882	18.5	964	31.3
30	195		—	—	37.6	198

Figure 4: Search times in seconds for optimum solutions.

1 assigned order 7, the other of size 3 assigned orders S. This is always possible: no colour constraint can become violated by cutting a slab into two, nor can the total capacity constraint be violated as the capacity is unchanged. The reverse transformation (merging two slabs into one) might violate a colour constraint so this is not a symmetry. This dominance can be enforced by adding a constraint to exclude any solution in which slab 1 has size 4 (size number 3) and is assigned order 7: $s_{13} + o_{17} \leq 1$. This binary constraint subsumes several symmetry breaking constraints of higher arity.

These binary constraints do not enforce all cutting dominances. In the second solution of Figure 3, orders 4,5,6 are assigned to slab 4, which therefore has size $1 + 1 + 1 = 3$. We cannot cut this into two slabs of sizes 1 and 2 because 2 is not a valid slab size. We can cut it into 3 slabs of size 1 as in the third solution. All the cutting dominance constraints can be described by $s_{ij} + \sum_{k \in \Omega} o_{ik} \leq |\Omega|$ where:

- Ω contains orders of no more than p different colours;
- either (i) $\sum_{k \in \Omega} \omega_k = \sigma_j$ and there is a proper partitioning of Ω into subsets each of whose size is a slab weight; or (ii) $\sigma_j - \sum_{k \in \Omega} \omega_k = \sigma_{j'}$ for some j' and there is a partitioning of Ω (possibly the trivial partitioning $\{\Omega\}$) into subsets each of whose size is a slab weight;
- Ω has no proper subset that can be so partitioned (to avoid generating subsumed constraints).

4.5 Results and Discussion

We transform the linear constraint models to pseudo-Boolean form and apply PBS [1] (with VSIDS variable ordering and G=50 as recommended). Using the instances from [9; 13], Figure 4 shows the time in seconds taken to find an optimum solution for ILOG Solver 5.0 on a 750 MHz Pentium III [**?**]; this has similar performance to our machine so the times are roughly comparable. Case (a) and the Solver results use symmetry breaking and implied constraints, case (b) adds capacity dominance enforcement constraints (implied when U is set to the optimum value), case (c) adds to (b) colour and wastage dominance enforcement constraints, and case (d) adds to (c) cutting dominance enforcement constraints up to arity 3. Times longer than one hour are denoted by "—". The upper bound U was set to the known optimum value for the problem.

PBS with symmetry breaking and implied constraints is (perhaps surprisingly) not much worse than Solver. Adding capacity dominance enforcement constraints significantly improves performance. Adding colour and wastage dominance enforcement constraints has a slightly erratic effect but is positive overall, and enables PBS to solve the largest problem. Adding cutting dominance enforcement constraints also has an erratic effect, with improvements on some instances.

Figure 5(i) compares PBS on model (c) with the results of [13] who used a faster 1.17 GHz Pentium III. (The CPLEX results are the best of several models; CPLEX was unable to solve all instances within the time limit with any single model.) They experimented with: IP models implemented in OPL and solved with CPLEX; CP models with two different branching strategies, solved with ILOG Solver 5.2; and hybrid CP/IP models solved by Solver and CPLEX. Both Solver and CPLEX were called via OPL. *Solver model 1* denotes an IP model similar to our model (a), and *model 2* denotes a hybrid model with channelling constraints.

We start PBS with $U = 1000$ instead of ∞ because our model needs a finite upper bound in order to express the wastage constraints (a better method would be to compute an upper bound for U). Each time we find a solution we restart the search with U set to the total slab size of that solution minus 1. There is no other communication between iterations so some search effort is wasted. When starting from a high value of U the best PBS variable ordering heuristic turn out to be a fixed ordering (o, s, c, e) instead of the recommended VSIDS ordering. We now fail to solve the largest instance within the time limit but obtain better results overall. All times include proof of optimality, which is trivial on these instances because each has a "perfect" solution with zero wastage. We use a threshold of 5746 seconds instead of one hour, to allow for our slower machine. Numbers in brackets indicate the best result found within the time limit, while times longer than the threshold are denoted by "—".

PBS with model (c) now has better overall performance than Solver with either constraint model, though our results are not as good as their CPLEX or hybrid results. Note that dominance enforcement constraints could be added to the hybrid model. (In retrospect it may have been more informative to add dominance in the same models as used by previous researchers.) As noted, for these problem instances the proof of optimality is trivial as they all have perfect solutions. We expect the dominance enforcement constraints to have an impact on the time required to prove optimality as they prune alternative (but not better) solutions. We therefore designed a set of benchmarks without perfect optimum solutions. For a problem of size N, we define a set of N orders with size 2 and colour 1, N orders with size 3 and colour 2, and 1 order of size 4 and colour 3. The available slab sizes are 3 and 6.

Figure 5(ii) shows the optimum total slab size U_{opt} and total order weight $\Sigma\omega$ for various values of N, with execution times for proofs of optimality ($U = U_{opt} - 1$) under various models. Values of N having perfect solutions are omitted. Cases (a), (c) and (d) are as in Figure 4. The colour, wastage and capacity dominance enforcement constraints make a very large difference to the proofs of optimality.

O	Solver model 1	Solver model 2	CPLEX model 1	Solver+ CPLEX	PBS (c)
12	(79)	0.28	9.90	0.52	1.05
16	(112)	6.51	10.7	0.53	11.9
18	(121)	145	1.98	0.67	118
19	(121)	303	2.6	1.91	78.7
20	(152)	2014	5.91	2.97	509
21	—	21.8	12.0	4.84	46.4
25	—	1216	65.2	7.6	5.1
30	—	(197)	1180	15.9	(196)

N	$\Sigma\omega$	U_{opt}	(a)	(c)	(d)
3	19	21	0.01	0.02	0.02
5	29	30	0.05	0.03	0.03
6	34	36	0.07	0.07	0.05
8	44	45	2.16	0.54	0.2
9	49	51	0.58	0.68	0.14
11	59	60	45.5	6.32	0.69
12	64	66	5.33	5.79	0.39
14	74	75	1777	131	6.7
15	79	81	51.2	55.5	0.99
17	89	90	—	1320	27.1

Figure 5: Slab design results.

5 Peaceable Armies of Queens

For our final case study we consider the problem of placing equally sized armies of black and white queens on an $N \times N$ chess board so that no white queen can attack a black queen (or vice-versa), and to maximize the size of the armies [4]. An IP model was defined by Plastria [5]. Smith et al. [18] defined and tested three constraint models for the problem, breaking symmetry by using SBDS [2; 11].

5.1 A Basic Model

We define a new IP for the problem, drawing on ideas from the constraint models of [18]. Associate a pair of 0/1 variables b_{ij} and w_{ij} with each square. The b_{ij} (resp. w_{ij}) take value 1 if there is a black (resp. white) queen on square (i, j) and 0 otherwise. In addition to these *square variables*, define 0/1 *line variables* b_ℓ and w_ℓ for each line ℓ (row, column or diagonal; one diagonal at each corner contains a single square). The optimization problem can be expressed as a series of CSPs with increasing lower bound Q on the number of black and white queens:

$$\sum_i \sum_j b_{ij} \geq Q \qquad \sum_i \sum_j w_{ij} \geq Q$$

Any surplus queens may be removed to obtain a pattern with exactly Q black and Q white queens. The other constraints are as follows. No square or line may be both black and white: $b_{ij} + w_{ij} \leq 1$ and $b_\ell + w_\ell \leq 1$. If a square is black [white] then its four associated line variables are black [white]:

$$4\, b_{ij} \leq \sum_{\ell \in L_{ij}} b_\ell \qquad 4\, w_{ij} \leq \sum_{\ell \in L_{ij}} w_\ell$$

where L_{ij} denotes the lines passing through square (i, j). If a line is black [white] then at least one of its associated squares is black [white]:

$$b_\ell \leq \sum_{(i,j) \in S_\ell} b_{ij} \qquad w_\ell \leq \sum_{(i,j) \in S_\ell} w_{ij}$$

where S_ℓ denotes the squares on line ℓ.

5.2 Symmetry

Smith et al. use SBDS to break four rotational symmetries, two reflectional symmetries and one colour symmetry (flip all colours). We add constraints to the model before search, which is weaker than SBDS but has the advantage that it can be used with any search algorithm. First the colour symmetry is broken by insisting that no white queens are placed on the top row: $w_1 = 0$. For the rotational and reflectional symmetries, we consider the eight half-rows and half-columns at the edges of the board as binary representations of integers. We then post constraints to make the black left half of the top row represent the greatest number:

$$\begin{array}{l}
\sum_{i=1}^{\lfloor N/2 \rfloor} 2^{i-1}(b_{i1} - b_{Ni} - w_{Ni}) \geq 0 \\
\sum_{i=1}^{\lfloor N/2 \rfloor} 2^{i-1}(b_{i1} - b_{iN} - w_{iN}) \geq 0 \\
\sum_{i=1}^{\lfloor N/2 \rfloor} 2^{i-1}(b_{i1} - b_{1i} - w_{1i}) \geq 0 \\
\sum_{i=1}^{\lfloor N/2 \rfloor} 2^{i-1}(b_{i1} - b_{N-i-1\,N} - w_{N-i-1\,N}) \geq 0 \\
\sum_{i=1}^{\lfloor N/2 \rfloor} 2^{i-1}(b_{i1} - b_{1\,N-i-1} - w_{1\,N-i-1}) \geq 0 \\
\sum_{i=1}^{\lfloor N/2 \rfloor} 2^{i-1}(b_{i1} - b_{N-i-1\,1} - w_{N-i-1\,1}) \geq 0 \\
\sum_{i=1}^{\lfloor N/2 \rfloor} 2^{i-1}(b_{i1} - b_{N\,N-i-1} - w_{N\,N-i-1}) \geq 0
\end{array}$$

5.3 Dominance

Suppose that in a solution, queens are *only* placed in a smaller rectangle than $N \times N$, for example rows and columns $2 \ldots N$. The solution can still be rotated and reflected within the smaller square, and also translated upward until a queen appears in the top row, and leftwards until a queen appears in the left column. The reverse is not true: a solution with queens in the top row and left column cannot necessarily be translated, because other queens might fall off the bottom row and right column. Thus we have two translational dominances, which can be enforced by insisting that at least one queen be placed in the top row and at least one queen be placed in the left column: $b_1 + w_1 \geq 1$ and $b_{N+1} + w_{N+1} \geq 1$, where line $N + 1$ denotes the left column. Combining these constraints with the colour symmetry breaking constraint above, results in the constraint $b_1 = 1$, requiring a black queen in the top row.

Another dominance occurs when placing a queen on a square cannot violate any constraint. There are three cases. Firstly, if a solution exists with an empty square whose lines are uncoloured then a solution also exists with a queen of either colour in that square (recall that in our formulation only lower bounds are placed on the sizes of the armies). We call this the *uncoloured dominance* and it can be enforced by placing a queen in that square; more specifically, a black queen. Secondly, if a solution exists in which an empty square has no white lines and at least one black line passing through it, then a black queen may be placed there. We call this the *black dominance*. Thirdly, if a solution exists in which an empty square has no black lines and at least one white line,

		pseudo-Boolean					ILOG Solver		
N	Q	basic	symm	dom	both	+VO	basic	symm	U+VO
4	2	<0.1	<0.01	<0.1	<0.1	0.01	0.03	0.02	0.02
5	4	0.12	0.02	0.08	0.01	0.02	0.11	0.04	0.03
6	5	1.6	0.15	0.6	0.07	0.08	2.9	0.41	0.13
7	7	29	2.2	6	0.49	0.66	56	7.8	1.12
8	9	236	41	93	5.4	5.25	2100	240	11.7
9	12	26091	1704	1262	93	31			116
10	14				879	298			2460
11	17				12379	2874			37100
12	21					17868			
13	24					247331			

Figure 6: Results for the Peaceable Armies of Queens

then a white queen may be placed there. This is the *white dominance*. Two families of constraints can be used to enforce these dominances. One enforces both the black and uncoloured and the other enforces the white:

$$b_{ij} + \sum_{\ell \in L_{ij}} w_\ell \geq 1 \qquad b_{ij} + w_{ij} + \sum_{\ell \in L_{ij}} b_\ell \geq 1$$

5.4 Results and Discussion

As above, these models are transformed to pseudo-Boolean form. We apply a naive backtracker using a lexicographical variable ordering: first the square variables row by row, then the line variables (experiments with the much more sophisticated PBS solver [1] surprisingly gave inferior results). Figure 6 shows the CPU times for the algorithms to find and prove optimal solutions for various values of N. It shows results for the basic model (*basic*), the basic model with symmetry breaking (*symm*), with dominance enforcement (*dom*), with both (*both*), and both with a modified (but still static) variable ordering heuristic (*+VO*). The modified variable ordering chooses variables with the highest score $pn + p + n$ where p and n are the number of positive and negative occurrences of the variable, breaking ties lexicographically. The figure also shows results for ILOG Solver with and without SBDS on a basic squares-only model (*basic* and *symm*), and ILOG Solver results with SBDS on the best (*unattacked squares*) model and a *most-unattacked-square* variable ordering heuristic (*U+VO*). The results for Solver are taken from Smith et al. [18] who use a 600 MHz Celeron PC. Q denotes the number of queens.

Both symmetry breaking and dominance enforcement make a significant difference to search time. Though our symmetry breaking approach is simple it achieves speedups comparable to those of SBDS on this problem. Dominance enforcement achieves similar speedups, that increase with N until it gives better results than symmetry breaking (we do not know whether this can be extrapolated to larger instances). Applying both symmetry breaking and dominance enforcement gives even better results than ILOG Solver, and adding the modified variable ordering further reduces execution times so that we are able to increase N by 2. Figure 7 shows the first known optimal solutions, as far as we are aware, for $N = 12$ and $N = 13$. The solution for $N = 13$ actually contains 25 black queens and 24 white ones; to obtain a solution with equal-sized armies we can simply remove any black queen.

```
  . . . b b b . . . . . b
  . . . b b b . . . . b .
  . . . b b b . . . b . b
  . . . . b . . . . . b b
  . . . . . . . . . b b b
  . . . . . . . . . b b .
  . . w . . . w . . . . .
  . w w . . . . . . . . .
  w w w . . . . . w . . .
  w w . . . . . w w . . .
  w . w . . . w w w . . .
  . w . . . . w w w . . .

b . . . b . b . . . b . b
. . w . . . . . w . . . .
. w . w . w . w . w . w .
. . w . . . . . w . . . .
b . . . b . b . . . b . b
. . w . . . . . w . . . .
b . . . b . b . . . b . b
. . w . . . . . w . . . .
. w . w . w . w . w . w .
. . w . . . . . w . . . .
b . . . b . b . . . b . b
. . w . . . . . w . . . .
b . . . b . b . . . b . b
```

Figure 7: Optimal 12×12 and 13×13 solutions.

6 Conclusion

Problem symmetry has recently received a great deal of attention in the literature, and exploiting it has led to huge improvements in search performance. Symmetry can be seen as a special (bi-directional) form of dominance, which is a weaker, more general property. However, dominance has been far less exploited than symmetry in the CP literature. We believe that it should rank alongside symmetry breaking as a generic CP technique, and that it can be profitable to treat both in a uniform way.

To support this thesis we added new dominance enforcement constraints to three well-studied symmetric problems. In the Still-Life and Peaceable Armies of Queens problems we found geometrically-inspired dominances that are analogous to some common symmetries: whereas *rotation* and *reflection* are common symmetries in geometric problems, we found *translational* and pattern-based dominances, leading to improved results on the latter problem. In the Steel Mill Slab Design problem we found a new form of dominance (*cutting*) that improves proofs of optimality. We hope that these examples will serve to stimulate further research on dominance in CP.

It may be that dominance has been relatively unexploited because it is more natural to think in terms of symmetry. However, we found that after a little practice it became quite natural to think in dominance terms, and that it helped to guide the modeling process. In the Steel Mill Slab Design problem we found several dominance enforcement constraints that transform *if* and *less-than-or-equal-to* definitions to tighter *if-and-only-if* and *equals* definitions. Though these could have been exploited without thinking in dominance terms, they were not used by previous researchers and we were led to them by considerations of dominance.

Dominance also led us to an insight on another problem. The best constraint model for the Peaceable Queens problem found by Smith et al. [18] was the *unattacked queens* model. That model represents only the white queens, ensuring that there are at least as many unattacked squares as white queens; black queens are implicitly placed in these squares. The stated advantages of this model are that the search variables have smaller domains and that there are fewer constraints. But another perspective on this model is that by merging the cases of a square containing no queen and a black queen, it implicitly enforces the black and uncoloured (but not the white or translational) dominances. Perhaps part of the advantage of this model can be traced to the absence of these dominances. In retrospect, the unattacked queens model *could have been* inspired by the identification and elimination of dominated solutions. Given that constraint modeling is a poorly understood and difficult process, heuristics that lead to good models are an important research direction.

Acknowledgments

This work was supported in part by the Boole Centre for Research in Informatics, University College, Cork, Ireland, and also by Science Foundation Ireland under Grant 00/PI.1/C075. Thanks to Ken Brown, Ian Gent, Brahim Hnich, Ian Miguel and Barbara Smith for helpful discussions.

References

[1] F. Aloul, A. Ramani, I. Markov, and K. Sakallah. PBS: A Backtrack Search Pseudo-Boolean Solver. *Symposium on the Theory and Applications of Satisfiability Testing*, 2002.

[2] R. Backofen, S. Will. Excluding Symmetries in Constraint-Based Search. *Fifth International Conference on Principles and Practice of Constraint Programming, Lecture Notes in Computer Science* vol. 1713, Springer-Verlag 1999, pp. 73–87.

[3] R. Bosch. Integer Programming and Conway's Game of Life. *SIAM Review* 41(3), 1999, pp. 594–604.

[4] R. Bosch. Peaceably Coexisting Armies of Queens. *Optima (Newsletter of the Mathematical Programming Society)* vol. 62 pp. 6–9, 1999.

[5] R. Bosch. Armies of Queens, Revisited. *Optima (Newsletter of the Mathematical Programming Society)* vol. 64, 2000, p. 15.

[6] R. Bosch and M. Trick. Constraint Programming and Hybrid Formulations for Life. *Workshop on Modelling and Problem Formulation,* Cyprus, 2001.

[7] H.-L. Fang. *Genetic Algorithms in Timetabling and Scheduling*. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 1994.

[8] P. Flener, A. Frisch, B. Hnich, Z. Kızıltan, I. Miguel, J. Pearson, T. Walsh. Symmetry in Matrix Models. *Workshop on Symmetry in Constraints*, Cyprus, 2001.

[9] A. M. Frisch, I. Miguel, T. Walsh. Symmetry and Implied Constraints in the Steel Mill Slab Design Problem. *Workshop on Modelling and Problem Formulation,* Cyprus, 2001.

[10] I. P. Gent, I. McDonald, I. Miguel, B. M. Smith. Approaches to Conditional Symmetry Breaking. *4th International Workshop on Symmetry and Constraint Satisfaction Problems*, Toronto, Canada, 2004.

[11] I. P. Gent, B. M. Smith. Symmetry Breaking During Search in Constraint Programming. *Fourteenth European Conference on Artificial Intelligence*, Berlin, Germany, 2000, pp. 599–603.

[12] L. Getoor, G. Ottosson, M. P. J. Fromherz, B. Carlson. Effective Redundant Constraints for Online Scheduling. *Fourteenth National Conference on Artificial Intelligence*, Providence, Rhode Island, 1997, pp. 302–307.

[13] B. Hnich, Z. Kızıltan, I. Miguel, T. Walsh. Hybrid Modelling for Robust Solving. *Annals of Operations Research* (to appear).

[14] T. Ibaraki The Power of Dominance Relations in Branch-and-Bond Algorithms *Journal of the Association for Computing Machinery*, vol. 24, 1977, ACM Press, pp. 264–279

[15] J. Larrosa, E. Morancho. Solving 'Still Life' with Soft Constraints and Bucket Elimination. *Ninth International Conference on Principles and Practice of Constraint Programming*, Kinsale, County Cork, Ireland, 2003, pp. 466–479.

[16] L. Proll, B. M. Smith. Integer Linear Programming and Constraint Programming Approaches to a Template Design Problem. *INFORMS Journal of Computing* vol. 10, 1998, pp. 265–275.

[17] J.-F. Puget. On the Satisfiability of Symmetrical Constrained Satisfaction Problems. J. Komorowski, Z. W. Ras (eds.), Methodologies for Intelligent Systems, *International Symposium on Methodologies for Intelligent Systems, Lecture Notes in Computer Science* vol. 689, Springer-Verlag, 1993, pp. 350–361.

[18] B. M. Smith, K. E. Petrie, I. P. Gent. Models and Symmetry Breaking for Peaceable Armies of Queens. *ECAI Workshop on Modelling and Solving Problems with Constraints*, 2002.

[19] B. M. Smith. A Dual Graph Translation of a Problem in 'Life'. *Principles and Practice of Constraint Programming, Lecture Notes in Computer Science* vol. 2470, Springer-Verlag, 2002, pp. 402-414.