Using auxiliary variables and implied constraints to model non-binary problems

Barbara Smith
School of Computer Studies
University of Leeds
Leeds LS2 9JT
England
bms@scs.leeds.ac.uk

Kostas Stergiou
Department of Computer Science
University of Strathclyde
Glasgow G1 1XL
Scotland
ks@cs.strath.ac.uk

Toby Walsh
Department of Computer Science
University of York
York YO10 5DD
England
tw@cs.york.ac.uk

Abstract

We perform an extensive theoretical and empirical analysis of the use of auxiliary variables and implied constraints in modelling a class of non-binary constraint satisfaction problems called *problems of distance*. This class of problems include 1-d, 2-d and circular Golomb rulers. We identify a large number of different models, both binary and non-binary, and compare theoretically the level of consistency achieved by generalized arc consistency on them. Our experiments show that the introduction of auxiliary variables and implied constraints can significantly reduce the size of the search space. For instance, our final models reduce the time to find an optimal 10-mark Golomb ruler 50-fold.

Introduction

In an invited talk at AAAI-98, Gene Freuder identified modelling as a major hurdle preventing the uptake of constraint satisfaction technology. Modelling is especially challenging when using non-binary constraints as the number of possible models is very large. In this paper, we model problems of distance, a challenging set of problems based on Golomb rulers. We identify a large number of different models, and compare them theoretically and empirically. Our results demonstrate the considerable benefits of including additional auxiliary variables and implied constraints in the models. We believe that many more studies like this are needed to help turn the art of modelling into a science.

Problems of distance

Peter van Beek has proposed Golomb rulers as a challenging constraint satisfaction problem for the CSPLib benchmark library (available as `prob006` at `http://csplib.cs.strath.ac.uk`). The specification given there is: "A Golomb ruler may be defined as a set of m integers $0 = x_1 < x_2 < ... < x_m$, such that the $m(m-1)/2$ differences $x_j - x_i$, $1 \leq i < j \leq m$, are distinct. Such a ruler is said to contain m marks and is of length x_m. The objective is to find optimal (minimum length) or near optimal rulers."

The longest known optimal ruler has 21 marks and is of length 333. Such rulers have practical applications in radio astronomy. Peter van Beek reports that even quite small problems (with fewer than 15 marks) are very difficult for complete methods, and that their difficulty lies both in proving optimality and in finding a solution since problems have very few solutions.

Golomb rulers are instances of a more general class of problem which we call *problems of distance*. Such problems are defined by a graph in which m nodes are labelled with integers, the edges are labelled by the difference between the node labels at either end of each edge, and there are constraints that all edge labels are different. A Golomb ruler is a problem of distance in which this underlying graph is complete. We may, however, have a problem of distance in which the underlying graph is not complete. For example, in a 2-d Golomb ruler we have (2 or more) layers of cliques, with edges between node i in clique j and node i in clique $j+1$.

A further generalization is to *modular* problems of distance in which the underlying graph is directed, the edge from node i to node j is labelled with the label of j less that of i, and there are constraints that all edge labels mod n are different. For instance, in a circular Golomb ruler, the underlying directed graph is complete, and n is the length of the circular rule.

Non-binary models

To model problems of distance, we use m variables, $x_1, \ldots, x_m$, each with a (finite) integer domain. There are three obvious non-binary representations:

quaternary model: for each pair of edges, we post quaternary constraints of the form $|x_j - x_i| \neq |x_l - x_k|$

ternary and not-equals model: for each edge, we introduce an auxiliary variable, d_{ij}; we post ternary constraints of the form $d_{ij} = |x_j - x_i|$, and binary not-equals constraints between the auxiliary variables

ternary and all-different model: we again introduce auxiliary variables, d_{ij} and ternary constraints of the form $d_{ij} = |x_j - x_i|$; however, we now post a single all-different constraint on the auxiliary variables

Additional constraints can also be added to eliminate various symmetries. For example, with a Golomb ruler,

we post the monotonicity constraints, $x_i < x_{i+1}$ for $1 \leq i < m$. A reflection symmetry can also be broken by adding the constraint $|x_2 - x_1| < |x_m - x_{m-1}|$ (or equivalently, $d_{12} < d_{m-1,m}$).

Theoretical comparison

As in previous studies (Stergiou & Walsh 1999a; 1999b)), we will compare theoretically the effects of constraint propagation on the different models; in particular, we focus on enforcing generalized arc-consistent (GAC) on the different non-binary constraints. A problem is GAC iff for any assignment of a value to a variable in a (non-binary) constraint, there exist compatible values for all the other variables in the constraint (Mohr & Masini 1988).

The introduction into a model of auxiliary variables might be expected to reduce the amount of constraint propagation achieved. However, the large all-different constraint can more than compensate for this reduction.

Theorem 1 *On a problem of distance, GAC on the ternary and all-different model is incomparable to GAC on the quaternary model.*

Proof: Consider a Golomb ruler with $x_1 = \{0\}$, $x_2 = \{1, 2\}$, and $x_3 = \{4\}$. The ternary and all-different model is GAC, but the quaternary model is not GAC (the value 2 must be removed from the domain of x_2).

Consider a Golomb ruler with $x_1 = \{0\}$, $x_2 = \{1, 2\}$, $x_3 = \{3\}$, and $x_4 = \{4, 5\}$. The quaternary model is GAC. However, enforcing GAC on the ternary and all-different model shows that the problem is insoluble since the auxiliary variables d_{12}, d_{23} and d_{34} have domains $\{1, 2\}$ and thus cannot be all different. QED.

More surprisingly, whilst replacing the large all-different constraint with binary not-equals constraints reduces the amount of constraint propagation, GAC on the ternary and not-equals model can still sometimes be stronger than GAC on the quaternary model.

Theorem 2 *On a problem of distance, GAC on the ternary and not-equals model is incomparable to GAC on the quaternary model.*

Proof: Consider the first Golomb ruler in the previous proof. The ternary and not-equals model is GAC. However, enforcing GAC on the quaternary model prunes the value 2 from the domain of x_2.

Consider a Golomb ruler with $x_1 = \{0\}$, $x_2 = \{1\}$, $x_3 = \{3\}$, $x_4 = \{7, 8\}$, and $x_5 = \{8, 9\}$. The quaternary model is GAC. However, enforcing GAC on the ternary and not-equals model shows that the problem is insoluble since the auxiliary variable d_{45} has all its possible values (1, 2 or 3) removed from its domain by the constraints with the auxiliary variables d_{12}, d_{23} and d_{13}. QED.

It is less surprising that replacing the single all-different constraint with binary not-equals constraints reduces the amount of constraint propagation.

Theorem 3 *On a problem of distance, GAC on the ternary and all-different model is strictly stronger than GAC on the ternary and not-equals model.*

Proof: It is trivially stronger as GAC on an all-different constraint is stronger than GAC on binary not-equals constraints. To show strictness, consider again the second Golomb ruler in the first proof. The ternary and not-equals model is GAC. However, enforcing GAC on the ternary and all-different model shows that is insoluble. QED.

By including additional implied constraints, the ternary and not-equals model can be made strictly stronger than the quaternary model. By transitivity, the ternary and all-different model can also be made strictly stronger than the quaternary model.

Theorem 4 *On a problem of distance, there exist some implied ternary constraints with which GAC on the ternary and not-equals model is strictly stronger than GAC on the quaternary model.*

Proof: The implied ternary constraints are of the form $|x_j - x_i| \neq |x_k - x_i|$ for all pairs of nodes j and k connected to a third node i. Consider one of the quaternary constraints, $|x_j - x_i| \neq |x_l - x_k|$. Assume that enforcing GAC on this quaternary constraint prunes a value but this value is not removed in the ternary and not-equals model. Due to the inclusion of the implied ternary constraints, i, j, k, l must all be different. The quaternary constraint is GAC unless at least three of the variables have a singleton domain, and one of the values of the remaining variable, say x_i, violates the constraint. But in that case, d_{kl} also has a singleton domain, and GAC on $d_{ij} \neq d_{kl}$ would delete the only possible value for d_{kl} from the domain of d_{ij}. Enforcing GAC on the ternary constraint $d_{ij} = x_j - x_i$ will then delete the same value of x_i. This contradicts the assumption that we prune a value in the quaternary model but not in the ternary and binary model. To show strictness, consider again the second Golomb ruler in the first proof. QED.

Our results show that a model with auxiliary variables and ternary constraints can in theory improve upon one with quaternary constraints. The next section shows that the differences can be very large in practice.

Golomb rulers

Table 1 shows the number of branches explored and the CPU time used on a Silicon Graphics O2 to find optimal Golomb rulers using ILOG Solver's inbuilt minimization functions. For efficiency, we use Jean-Charles Régin's specialized GAC algorithm on the all-different constraint (Régin 1994), simple bounds consistency on the ternary and quaternary constraints, and arc consistency on the binary constraints (which is equivalent to GAC on binary constraints). We show later on that GAC on the ternary constraints is not competitive in terms of CPU time with bounds consistency. The same is likely to be true of GAC on the quaternary constraints, especially as GAC is more expensive as we increase the constraint arity and as it will tend to prune fewer values.

In all three models, we used a fixed lexicographical variable ordering. Variables are assigned in order start-

Marks (m)	quaternary br.	quaternary CPU	ternary + $\neq$ br.	ternary + $\neq$ CPU	ternary + alldiff br.	ternary + alldiff CPU
6 - F	15	0.020	6	0.007	6	0.012
- P	63	0.042	39	0.015	10	0.006
7 - F	116	0.170	28	0.023	26	0.033
- P	594	0.801	327	0.198	84	0.125
8 - F	756	2.03	130	0.104	98	0.124
- P	4852	14.0	2605	2.27	599	1.23
9 - F	7271	31.7	1622	1.70	816	1.56
- P	33679	168	17823	22.1	2924	9.69
10 - F	78503	657	21507	27.9	9757	24.3
- P	-	-	-	-	13707	68.3
11 - F	-	-	-	-	31666	94.5
- P	-	-	-	-	-	-

Table 1: Branches explored and CPU time (seconds) used to find a minimal length ruler (F) or prove that none shorter exists (P). A - means that the run was cut off after 10^5 branches.

ing from x_2 (since x_1 can be unconditionally assigned to 0). Constraint propagation on the ternary constraints will assign the auxiliary variables. We tried the smallest domain dynamic variable ordering heuristic on a variety of different sets of search variables, but were unable to beat a simple lexicographic heuristic that builds up the ruler from one end.

Table 1 shows that the quaternary model is much less efficient than the ternary models, both in terms of branches explored and CPU time. Introducing auxiliary variables is in practice very worthwhile for these problems. With the ternary models, making the alldiff constraint GAC gives the smallest search tree. On the smaller problems, runtimes are not always shorter than if binary $\neq$ constraints are used, but on the larger problems the savings are considerable, particularly when proving optimality.

Binary encodings

An alternative strategy for solving a non-binary model is to encode it into a binary model using one of the standard encodings such as the hidden variable or the dual encoding (Bacchus & van Beek 1998; Stergiou & Walsh 1999b). In the case of Golomb rulers, the double encoding introduced in (Stergiou & Walsh 1999b) is more practical than the dual encoding. The double encoding combines together all the constraints from the dual and the hidden variable encodings. In a dual encoding, the dual variables associated with either the all-different constraint or the binary not-equals constraints have such large domains that we cannot afford to enforce arc consistency. In a double encoding, whilst we use dual variables associated with the ternary constraints, we can ignore the dual variables associated with the all-different constraint or the binary not-equals constraints as they are redundant. This makes it computationally feasible to use the double encoding. Finally, whilst encodings of models with ternary constraints are practical, encodings of the quaternary constraints have domains which are prohibitively large. We therefore looked at four new models.

hidden variable + all-different model: each ternary constraint is replaced by a hidden variable with domain of size $O(l^2)$; we also post a single all-different constraint between the auxiliary variables;

hidden variable + not-equals model: again each ternary constraint is replaced by a hidden variable; we also post binary not-equals constraints between the auxiliary variables; this model contains purely binary constraints;

double encoding + all-different model: again each ternary constraint is replaced by a hidden variable; we also post compatibility constraints between hidden variables that share variables, and a single all-different constraint between the auxiliary variables;

double encoding + not-equals model: again each ternary constraint is replaced by a hidden variable; we also post compatibility constraints between hidden variables that share variables, and binary not-equals constraints between auxiliary variables; this model contains purely binary constraints.

It was not feasible to find optimal rulers using the binary encodings as this requires setting l to some large initial value, and the domain size of the hidden variables is then prohibitively large. Instead, we have used the known optimal rulers to compare the encodings. We first find a ruler with length equal to the optimal length, and then show that there is no shorter ruler. Table 2 compares the models in which all the constraints are binary with the ternary and not-equals model.

m	l	hidden + $\neq$ br.	hidden + $\neq$ CPU	double + $\neq$ br.	double + $\neq$ CPU	ternary + $\neq$ br.	ternary + $\neq$ CPU
6	17	5	0.150	5	0.473	5	0.052
6	16*	36	0.304	32	0.912	36	0.109
7	25	18	0.645	17	3.70	18	0.251
7	24*	286	3.13	237	11.9	286	1.18
8	34	45	2.34	45	21.2	45	0.964
8	33*	2015	31.4	1461	117	2012	12.1
9	44	708	22.0	506	147	705	8.51
9	43*	12822	302	8846	1180	12815	115

Table 2: Branches explored and CPU time (seconds) used to find a ruler of length no more than l or prove that none exists, using lexicographic ordering. * indicates that there is no ruler of this length.

Fewer branches are explored in the double than in the hidden encoding. The hidden encoding gives very similar results, in terms of the size of the search tree, to the ternary model with GAC. However, the CPU times tell a different story. Both the binary encodings, and especially the double, are far worse than the ternary model. This is not surprising as arc consistency takes longer to enforce in the binary encodings due to the large domain sizes. Results are similar (though slightly better) with the models using all-different constraints: the double encoding gives the best results in terms of the number of branches explored, but the worst CPU times.

Implied constraints

As Theorem 4 suggests, one route to improved performance is to add implied constraints to the model. Although implied constraints can often significantly re-

duce runtimes (e.g. (Proll & Smith 1998; Regin 1998)), choosing useful implied constraints remains an art. We can, however, state two basic criteria. First, we require either implied constraints for which specialized and efficient constraint propagation algorithms are available, or constraints of small arity. Second, circumstances in which an implied constraint leads to pruning of additional values should be obvious and frequent. However, whilst these criteria are desirable, they are not sufficient as implied constraints must offer enough pruning to offset their overheads. This is hard to predict without experimentation. For instance, the implied constraints of Theorem 4 satisfy the two criteria, but do not justify their overhead. In the rest of this section, we consider other implied constraints, and their effect when added to the current best model, i.e. the ternary and all-different model with lexicographic variable ordering.

Ordering of Auxiliary Variables

For all $i < j < k$, we can post the implied constraints $d_{ij} < d_{ik}$ and $d_{jk} < d_{ik}$. These implied constraints are binary, and hence cheap to propagate. It is also easy to see that they can lead to domain reductions not achievable otherwise. However, experiments show that they only reduce the size of the search tree modestly, and do not reduce runtimes on larger problems.

Tighter Bounds on Auxiliary Variables

Since $d_{ij} = d_{i,i+1} + d_{i+1,i+2} + \ldots + d_{j-1,j}$, and each of term in this sum is different, d_{ij} must at least equal the sum of the first $j-i$ integers, i.e. $d_{ij} \geq (j-i)(j-i+1)/2$. We can tighten this bound further as a subsection of the ruler, i.e. from mark i to mark j, must itself form a (not necessarily optimal) Golomb ruler of $j - i + 1$ marks. Therefore, $d_{ij} \geq l_{j-i+1}$ where l_k is the optimal length of a k mark ruler. Both these implied constraints are cheap to implement as they are unary.

Another bound on the auxiliary variables comes from the constraint $x_n = d_{12} + d_{23} + \ldots + d_{n-1,n}$. That is, $d_{ij} = x_n - (d_{12} + d_{23} + \ldots + d_{i-1,i} + d_{j,j+1} + \ldots + d_{n-1,n})$. The RHS is x_n less the sum of $n - 1 - j + 1$ different integers. So we can maximize the RHS by minimizing this sum. This gives the implied constraint $d_{ij} \leq x_n - (n - 1 - j + i)(n - j + i)/2$. This is again efficient to implement as it is binary. As x_n is reduced, the constraint gets tighter, and so becomes more effective as we approach the minimal length.

Table 3 shows the significant benefits of adding these bounds to the model, with and without the ordering constraints described in the last section. On the larger problems, search space and runtimes are significantly reduced. Because of the smaller domains, CPU time reduces on the smaller problems even when the number of branches remains constant. The ordering constraints on the auxiliary variables continue to offer little benefit.

Dynamic Bounds on Auxiliary Variables

By adding implied constraints *dynamically* during search, we can post even tighter bounds on the auxiliary variables. For example, if we have assigned 1 and 3 to x_2 and x_3, then any auxiliary variable d_{ij} for $i \geq 2$ must be at least the sum of $j - i$ different integers *excluding* 1, 2 and 3, the values already assigned. We have implemented a limited form of propagation for this type of implied constraint. We record the largest mark so far assigned (x_{max}) and the values assigned to the auxiliary variables. For all $m > max$, we calculate the sum of the $m - max$ smallest integers which have not yet been assigned to auxiliary variables, and post a constraint on this branch of the search tree that the value of $d_{max,m}$ must be at least this sum.

m	ternary + alldiff		+ tighter bounds		+ tighter bounds + ordering constraints	
	br.	CPU	br.	CPU	br.	CPU
7 - F	26	0.033	26	0.032	26	0.038
- P	84	0.118	54	0.051	54	0.063
8 - F	98	0.124	98	0.128	98	0.160
- P	599	1.23	385	0.503	385	0.635
9 - F	816	1.56	718	1.17	718	1.47
- P	2924	9.69	1751	3.61	1751	4.48
10 - F	9757	24.3	7971	17.1	7948	21.3
- P	13707	68.3	7812	24.0	7807	29.5
11 - F	31666	94.5	29251	80.8	29190	108
- P	343220	2000	252985	1020	252577	1300

Table 3: Branches explored and CPU time to find a minimal length ruler (F) or prove that none shorter exists (P), with and without tighter bounds on the auxiliary variables.

m	ternary + alldiff + + tighter bounds		+ dynamic bounds	
	br.	CPU	br.	CPU
7 - F	26	0.032	25	0.032
- P	57	0.051	50	0.049
8 - F	98	0.128	80	0.111
- P	385	0.503	355	0.480
9 - F	718	1.17	509	0.822
- P	1751	3.61	1564	3.41
10 - F	7971	17.1	5428	12.0
- P	7812	24.0	6810	22.1
11 - F	29251	80.8	19211	53.6
- P	252985	1020	224636	967

Table 4: Branches explored and CPU time to find a ruler of minimal length (F) or prove that none shorter exists (P), with and without the dynamic bounds on the auxiliary variables.

Table 4 shows that adding this implied constraint dynamically to the model during search gives the best results so far. Except for the smallest problems, there are significant reductions in both the number of branches explored and CPU time.

2-d Golomb rulers

In the next two sections, we see how these results map onto related problems. We first consider 2-d Golomb rulers with two layers, each of which is a 1-d ruler, with marks at $x_1, x_2, \ldots x_m$ and $y_1, y_2, \ldots y_m$ respectively. As before we introduce auxiliary variables to represent the pairwise distances between the marks in each layer, as well as the distances $|x_i - y_i|$. We add a symmetry constraint $x_2 - x_1 < x_m - x_{m-1}$ on the first layer, as before, but we cannot add a similar constraint on the second. However, there is a symmetry between the two layers, and we can break this by adding $x_2 - x_1 < y_2 - y_1$.

As in the 1-d case, the quaternary model is very inefficient compared to the ternary and all-different one, which again is the best model. The hidden variable representation reduces the search space substantially but again the CPU times are much worse. Since each layer is itself a (not necessarily optimal) 1-d Golomb ruler, we can use all the implied constraints introduced in the last section. However, now they have very little effect on the number of fails and are more expensive in terms of cpu time. The minimum length is much greater than in the 1-d case for the same value of m, and hence the lower bounds provided by the implied constraints are much less effective.

Variable ordering again has a significant effect on performance, and the smallest domain heuristic once more performs poorly. Table 5 gives the number of branches explored and the CPU time used (on a 166MHz Pentium) to find and prove optimal 2-d Golomb rulers with two layers, each with m marks, with different variable orderings. In all cases, the search variables are those representing the marks. We compare the dynamic smallest domain ordering with two static orderings; in one, the marks from the first layer are assigned and then those from the second; in the other, marks from two layers are assigned alternately. As in the 1-d case, building up each layer successively is much quicker than the dynamic ordering, but it is better to build up both layers in parallel than to complete one layer first. This allows the distances between the layers to be assigned at the same time.

m	smallest dom.		$x_2, .., x_m, y_2, ..., y_m$		$x_2, y_2, ..., x_m, y_m$	
	br.	CPU	br.	CPU	br.	CPU
3 - F	3	0.049	1	0.038	1	0.036
- P	9	0.058	9	0.057	5	0.047
4 - F	89	0.44	78	0.15	32	0.070
- P	419	0.45	395	0.48	199	0.23
5 - F	2929	4.19	2217	3.7	1479	1.84
- P	21069	36	19510	35	7719	11

Table 5: Branches explored and CPU time to find a minimal length 2-d Golomb ruler with two layers (F) or prove that none shorter exists (P), with different variable orderings.

Note that these problems are much harder to solve than 1-d Golomb rulers. Our fastest model takes more than 15 minutes to find and prove an optimal solution for $m = 6$, and we have not been able to prove optimality for $m = 7$.

Circular Golomb rulers

Circular or modular Golomb rulers can be thought of as a number of marks arranged on the circumference of a circle in such a way that the distances between any pair of marks, in either direction along the circumference, are different. The problem of finding a minimal length ruler for a given number of marks is different in several respects from the linear problem. Solutions can be constructed (see `http://www.research.ibm.com/people/s/shearer/ mgrule.html` for an account) and if $m - 1$ is a prime power, there is a perfect ruler, i.e. every intermediate length occurs. This is very different from the linear case, where perfect rulers are extremely rare. Although search will not be a good approach to finding solutions to these problems, it is instructive to consider the issues that arise when modelling them as CSPs.

With a circular Golomb ruler, we can ignore the variables associated with the marks, and assign instead search variables $d_0, \ldots, d_{m-1}$ to represent the distance between adjacent marks. In this sense, the ternary models of circular Golomb rulers are more fundamental than the quaternary models. We break the rotational and reflective symmetry by insisting that $d_0 = 1$ and $d_1 < d_{m-1}$.

We also define a set of composite distances between every pair of non-adjacent marks: $d_{i,j} = d_i + d_{(i+1) \bmod m} + \ldots + d_{(i+k) \bmod m}$, where $(i+k) \bmod m = j$, for all $i \neq j$ between 0 and $m - 1$. To express the constraint that the circumference of the ruler is minimal, we introduce an objective variable l constrained to equal $d_0 + d_1 + \ldots + d_{m-1}$. This is the basic model whose results are shown below in Table 6, using lexicographic ordering of the search variables.

Because of the large arity of the constraint defining l and of some of the constraints defining the composite distances, we introduced new ternary constraints expressing the composite distances and l in terms of one of the basic distances and a shorter composite distance. This gives in general two constraints for $d_{i,j}$: $d_{i,j} = d_i + d_{(i+1) \bmod m, j}$ and $d_{i,j} = d_{i,(j-1) \bmod m} + d_j$, and m constraints for l, one for every basic distance. The effect of these implied constraints is shown in the second column of Table 6: there is a significant saving in both the number of branches and CPU time, for larger problems.

We can also use the lower bounds derived earlier for linear rulers, based on the fact that each composite distance is at least the sum of a number of different integers. However, adding these bounds makes virtually no difference to the number of branches.

The third column of Table 6 shows the effect of making the all-different constraint GAC. The arity of the all-different constraint, for the same number of marks, is much higher than with a linear ruler, so making the constraint GAC is more expensive. The Table shows that it is not worthwhile, in terms of CPU time, for finding the optimal solution. However, it does allow optimality to be proved very quickly, except for $m = 7$. For the other values of m, the optimal ruler is perfect. Since there are $m * (m - 1)$ different distances between the marks, the length of a perfect circular ruler is $m * (m - 1) + 1$. This could be included in the model as a lower bound on l, and the proof of optimality would be immediate, except for $m = 7$. The benefit of making the all-different constraint GAC would then disappear.

Another difference between modelling the circular and linear problems is that in this case the lexicographic ordering $d_0, d_1, \ldots, d_{m-1}$ does not reflect the geometry of the problem so well as in the linear case. Perhaps for this reason, the smallest domain ordering performs

relatively well, solving some instances more quickly.

m	basic model		+ implied constraints		+ alldiff	
	br.	CPU	br.	CPU	br.	CPU
5 - F	22	0.067	19	0.044	16	0.161
- P	33	0.079	29	0.055	18	0.168
6 - F	10	0.082	9	0.049	8	0.215
- P	166	0.198	117	0.151	11	0.226
7 - F	724	0.941	333	0.516	297	1.07
- P	5523	6.32	2819	3.91	1889	5.61
8 - F	4042	8.08	1350	3.18	1146	4.66
- P	23109	43.3	8564	19.0	1150	4.68
9 - F	948	2.31	346	1.26	331	2.17
- P	-		-		336	2.23

Table 6: Effect of implied constraints and making the all-different constraint GAC on branches explored and CPU time to find a circular Golomb ruler of minimal circumference (F) or prove that none shorter exists (P).

Related work

Nadel performed a study on various different ways of modelling the n-Queens problem (Nadel 1990). In CSPLib, Jean-Francois Puget reports finding a 12-mark Golomb ruler using ILOG Solver in 2,042,001 branches, and proving it optimal in a further 1,141,316 branches. Our best model offers a significant improvement over this result. With the ternary and all-different model and implied constraints, we found an optimal 12-mark solution in 1,398,327 branches, and needed just a further 513,109 branches to prove it optimal. This is a significant reduction in search effort. Auxiliary variables are important in modelling cryptarithmetic problems like SEND + MORE = MONEY. Large arithmetic constraints such as these can be decomposed into more useful ternary and quaternary constraints via auxiliary variables that represent the "carries" (Brailsford, Potts, & Smith 1999).

Conclusions

We have performed an extensive theoretical and empirical analysis of different models for problems of distance like 1-d, 2-d and circular Golomb rulers. We introduced a number of different models with quaternary, ternary, binary and all-different constraints. We proved that, whilst the level of consistency achieved by GAC on the quaternary and ternary models is incomparable in general, the ternary models are strictly stronger than the quaternary if we introduce some simple implied ternary constraints. We also considered purely binary models using hidden variables for the non-binary constraints. Whilst these reduced the number of branches explored as predicted by theory, they did not always give a reduction in runtime because of the cost of enforcing arc consistency on hidden variables with large domains. We also identified several sets of implied constraints, some of which reduced both the number of branches explored and the CPU time significantly. Our final models offer dramatic improvement reducing, for instance, the time to find an optimal 10–mark Golomb ruler 50-fold.

What general lessons can be learned from this study? First, even simple problems can be modelled in many different ways. Counting all possible encodings, this study alone has identified fifteen possible models for the Golomb ruler problem, although we have omitted some of these from our experiments (in particular the binary encodings of the quaternary models). If we include all possible ways of adding the implied constraints, there would be many more. Secondly, finding the best model involves a trade-off between the arity of the constraints and the efficiency with which we can reason about them. The most effective model in this case study was not the quaternary model but the ternary model with a single, large, all-different constraint. Thirdly, the addition of auxiliary variables and implied constraints can allow us to achieve higher levels of consistency. However, identifying those implied constraints which will reduce runtimes is not easy. Although we have suggested some guidelines, our experience indicates that experiment is often needed to tell whether a set of implied constraints is worthwhile.

Acknowledgements

The third author is an EPSRC advanced research fellow. The authors are members of the APES research group (http://www.cs.strath.ac.uk/~apes) and wish to thank the other members.

References

Bacchus, F., and van Beek, P. 1998. On the conversion between non-binary and binary constraint satisfaction problems. In *Proceedings of AAAI-98*, 311–318.

Brailsford, S. C.; Potts, C. N.; and Smith, B. M. 1999. Constraint Satisfaction Problems: Algorithms and Applications. *European Journal of O.R.* 119:557–581.

Mohr, R., and Masini, G. 1988. Good old discrete relaxation. In *Proceedings ECAI-88*, 651–656.

Nadel, B. 1990. Representation Selection for Constraint Satisfaction: A Case Study using n-Queens. *IEEE Expert* 5(3):16–24.

Proll, L., and Smith, B. 1998. ILP and constraint programming approaches to a template design problem. *INFORMS Journal on Computing* 10:265–277.

Régin, J.-C. 1994. A filtering algorithm for constraints of difference in CSPs. In *Proceedings of AAAI-94*, 362–367.

Regin, J.-C. 1998. Minimization of the number of breaks in sports scheduling problems using constraint programming. In *Proceedings of the DIMACS Workshop on Constraint Programming and Large Scale Discrete Optimization.*

Stergiou, K., and Walsh, T. 1999a. The difference all-difference makes. In *Proceedings of IJCAI-99.*

Stergiou, K., and Walsh, T. 1999b. Encodings of non-binary constraint satisfaction problems. In *Proceedings of AAAI-99.*