Why Channel?
Multiple viewpoints for branching heuristics

Brahim Hnich and Toby Walsh

Cork Constraint Computation Center, University College Cork, Ireland.
{brahim,tw}@4c.ucc.ie

Abstract. When modelling a problem, there are often alternative viewpoints that can be taken. It can even be advantageous to use multiple viewpoints, and to have constraints which channel between them to maintain consistency. Multiple viewpoints often make it much easier to post the different problem constraints, as well as improve the amount of constraint propagation. In this paper, we demonstrate another reason for using multiple viewpoints: branching heuristics can be more effective when they look at multiple viewpoints.

1 Introduction

Constraint programming is a highly successful technology for solving a wide variety of combinatorial problems like resource allocation, transportation, and scheduling. However, its uptake is hindered by the difficulty of modelling problems successfully as constraint programs. One key modelling decision is which viewpoint or viewpoints to use. For example, in modelling a sports tournament scheduling problem, do we take the viewpoint in which the games are the variables, and the times are their values? Or do we take the dual viewpoint in which the times are the variables, and the games are their values? Or do we take both viewpoints, and have channelling constraints to maintain consistency between the two viewpoints?

There are a number of reasons we might consider multiple viewpoints, even though using more than one viewpoint introduces additional overheads. First, different constraints may be easier to post in the different viewpoints. Second, propagation may be improved. For example, one viewpoint may be linear and so can be solved using ILP. A third possibility, proposed by Geelen in [1] and explored in detail here, is that branching heuristics may profit from looking at more than one viewpoint. The results given in [1] are promising but are limited to a small number of experiments on n-queens problems. It is therefore timely to perform a more extensive experimental study on a range of challenging problems.

2 Permutation problems

Our analysis will focus on permutation problems. A *permutation problem* is a constraint satisfaction problem in which each decision variable takes an unique value, and there is the same number of values as variables. In a permutation problem, we can easily transpose the roles of the variables and the values to give a *dual* model which is also a permutation problem. Each variable in the primal becomes a value in the dual, and vice versa. We shall also consider *multiple permutation problems* in which the variables divide into a number of (possibly overlapping) sets, each of which is a permutation problem.

It is possible to combine multiple viewpoints by using *channelling constraints* to maintain consistency between the different viewpoints. This approach is called "redundant modelling" by Cheng *et al.* [2] and was specifically suggested for permutation problems in [1]. In a permutation problem, the channelling constraints are of the form: $X_i = j$ iff $D_j = i$. Many constraint toolkits support channelling of this kind with efficient global constraints. For example, ILOG Solver has a constraint, `IlcInverse`, and the Sicstus finite domain constraint library has an `assignment` predicate which can be used to channel efficiently between the primal and dual viewpoints of a permutation.

To ensure that we have a permutation, we can post a global all-different constraint on the primal variables. Alternatively, we can post binary not-equals constraints between any two primal variables. However, if we have both primal and dual variables, the channelling constraints are on their own sufficient to ensure we have a permutation. Indeed, the channelling constraints provide an intermediate level of pruning between what GAC achieves on a primal all-different constraint and AC on binary not-equals constraints on the primal [3, 4]. AC on the binary not-equals constraints identifies singleton variables (those variables with a single value left in their domain). AC on the channelling constraints identifies both singleton variables and singleton values (those values which are left in the domain of a single variable). GAC on an all-different constraint identifies both singleton variables and singleton values plus even more complex situation (e.g. three variables with just two values left between them).

There are thus a large number of different ways to model and solve a permutation problem. We can, for example, post an all-different constraint posted on the primal and use Regin's efficient algorithm [5] to maintain GAC on this constraint (in the tables of results, we will write "$\forall$" for this model and solution method). Alternatively, we can maintain AC on channelling constraints between primal and dual (we write "c" for this model and solution method). A third viewpoint is to maintain AC on binary not-equals constraints between any two primal variables (we write "$\neq$" for this model and solution method). Finally, we can take any combination of these viewpoints. For example, we can maintain GAC on an all-different constraint on the primal and AC on channelling constraints between primal and dual (we write "$\forall c$" for this model and solution method).

3 Variable and value ordering

The aim of this paper is to study how multiple viewpoints may benefit variable and value ordering heuristics. A variable ordering heuristic like smallest domain is usually justified in terms of a "fail-first" principle. We have to pick eventually all the variables, so it is wise to choose one that is hard to assign, giving us hopefully much constraint propagation and a small search tree. On the other hand, a value ordering heuristics like most promise [1] is usually justified in terms of a "succeed-first" principle [6]. We pick a value likely to lead to a solution, so reducing the risk of backtracking and trying one of the alternative values. In a permutation problem, we can branch on the primal or the dual variables or on both. We therefore consider the following heuristics.

Smallest domain, SD(p+d) : choose the primal or the dual variable with the smallest domain, and choose the values in numeric order.

Primal smallest domain, SD(p) : choose the primal variable with the smallest domain, and choose the values in numeric order.

Dual smallest domain, SD(d) : choose the dual variable with the smallest domain, and choose the values in numeric order.

Double smallest domain, $\mathbf{SD^2(p+d)}$: choose the primal/dual variable with the smallest domain, and choose the value whose dual/primal variable has the smallest domain.

Primal double smallest domain, $\mathbf{SD^2(p)}$: choose the primal variable with the smallest domain, and choose the value whose dual variable has the smallest domain.

Dual double smallest domain, $\mathbf{SD^2(d)}$: choose the dual variable with the smallest domain, and choose the value whose primal variable has the smallest domain.

The idea of using the smallest domain heuristic on the dual as a value ordering heuristic can be traced at least as far back as [7]. It was also used in [2, 3]. We shall now argue that the variable and value ordering provided by the double smallest domain heuristics is consistent with the fail first principle for variable ordering and the succeed first for value ordering. Barbara Smith in a personal communication to the authors made a similar argument. Suppose we assign the primal value k to the primal variable X (an analogous argument can be given if we branch on a dual variable). Constraint propagation will prune the primal value k from the other primal variables, and the dual value X from the other dual variables. Of course, constraint propagation may do more than this if we have an all-different constraint or channelling constraints. However, to a first approximation, this is a reasonable starting point. Geelen's succeed first value ordering heuristic computes the "promise" of the different values by multiplying together the domain sizes of the uninstantiated variables [1]. Each term in this product is constant if k and X do not occur in the domain and is reduced by 1 if k or X occurs in the domain. This is likely to be maximized by ensuring we reduce as few terms as possible. That is, by ensuring k and X occur in as few domains as possible. That is X and D_k have the smallest domains possible. Hence double smallest domain will tend to branch on the variable with smallest domain and assign it the value with most promise.

4 Problem domains

We will compare these different models and heuristics on the following collection of permutation problems. All the models are implemented in Solver 5.300, and are available at CSPLib.

Langford's problem: Given two integers n and m, Langford's problem is to permute n sets of numbers 1 to m, so that each appearance of the number i is i on from the last. This is **prob024** in CSPLib.

Quasigroup existence problem: An order n quasigroup is a Latin square of size n. That is, an $n \times n$ multiplication table in which each row and column is a permutation of the numbers 1 to n. Quasigroups existence problem determines the existence or non-existence of quasigroups of a given size with additional properties. QG3(n) denotes quasigroups of order n for which $(a * b) * (b * a) = a$. QG4(n) denotes quasigroups of order n for which $(b * a) * (a * b) = a$. Furthermore, we may additionally demand that the quasigroup is idempotent, i.e., $a * a = a$ for every element a. This is **prob003** in CSPLib.

Golomb rulers problem: A Golomb ruler has n marks arranged on the ticks of a ruler of length m such that the distances between any pair of marks are all distinct. This is **prob006** in CSPLib.

Sport scheduling problem: We want to schedule games between n teams over $n - 1$ weeks when n is even (n weeks when n is odd). Each week is divided into $n/2$ periods when n is even ($(n-1)/2$ when n is odd). Each game is composed of two slots, "home" and "away", where one team plays home and the other team plays away. The objective is to schedule a game for each period of every week such that: every team plays against every other team; a team plays exactly once a week when we have an even number of teams, and at most once a week when we have an odd number of weeks; and a team plays at most twice in the same period over the course of the season. This is **prob026** in CSPLib.

Magic squares problem: An order n magic square is an n by n matrix containing the number 1 to n^2, with each row, column, and diagonal equal the same sum. This is **prob019** in CSPLib.

5 Experimental results

We now compare the different models and branching heuristics in an extensive set of experiments. The hypothesis we wish to test is that branching heuristics can profit from multiple viewpoints.

5.1 Langford's problem

The results are given in Table 1. We make a number of observations. The primal not-equals viewpoint ("$\neq$") gives the worst results (as it does in almost all the subsequent problem domains). We will not therefore discuss it further. The best

		L(3,12)		L(3,13)		L(3,14)		L(3,15)	
model	heuristic	fails	sec.	fails	sec.	fails	sec.	fails	sec.
$\neq$	SD(p)	62016	10.27	300800	53.72	1368322	272.03	7515260	1601.00
$\forall$	SD(p)	20795	3.59	93076	16.95	405519	78.18	2072534	414.71
c	SD(p+d)	11683	**2.16**	45271	**8.66**	184745	**36.46**	846851	**171.97**
c	SD(p)	21148	3.68	94795	16.84	412882	74.99	2112477	389.69
c	SD(d)	15214	2.64	59954	10.73	249852	46.39	1144168	221.01
c	SD^2(p+d)	11683	2.2	45271	9.04	184745	38.32	846851	180.00
c	SD^2(p)	20855	3.89	93237	17.07	406546	75.38	2077692	393.21
c	SD^2(d)	14314	2.62	56413	10.61	234770	45.68	1076352	213.51
$\forall c$	SD(p+d)	**11449**	2.84	**44253**	11.47	**180611**	48.71	**827564**	231.80
$\forall c$	SD(p)	20795	4.93	93076	22.61	405519	102.45	2072534	537.14
$\forall c$	SD(d)	14459	3.44	56701	13.94	234790	60.13	1069249	282.42
$\forall c$	SD^2(p+d)	11451	2.91	44254	11.72	180631	49.71	827605	235.56
$\forall c$	SD^2(p)	20488	4.98	91513	22.86	399092	103.09	2037159	540.04
$\forall c$	SD^2(d)	13639	3.38	53483	13.78	221307	59.33	1009250	278.32

Table 1. No. of backtracks (fails) and running time to find all solutions, or prove that there are no solutions, to four instances of Langford problem. Runtimes are for ILOG Solver 5.300 on 1200MHz, Pentium III processor, and 512 MB of RAM.

		QG3(6)		QG(7)		QG3(8)		QG3(9)	
model	heuristic	fails	sec.	fails	sec.	fails	sec.	fails	sec.
$\neq$	SD(p)	8	**0.01**	100	0.22	1895	8.46	83630	600.61
$\forall$	SD(p)	7	**0.01**	59	0.17	955	5.76	**35198**	385.57
c	SD(p+d)	7	0.02	63	**0.16**	1117	5.81	53766	463.40
c	SD(p)	7	0.02	59	0.17	1039	5.70	38196	373.38
c	SD(d)	6	**0.01**	54	0.19	888	**5.40**	46539	418.96
c	SD^2(p+d)	7	0.02	63	0.17	1117	5.83	53785	461.05
c	SD^2(p)	7	**0.01**	58	0.17	1043	5.68	38198	372.41
c	SD^2(d)	6	**0.01**	54	0.18	887	5.42	46741	419.94
$\forall c$	SD(p+d)	7	0.02	54	**0.16**	999	6.00	49678	474.82
$\forall c$	SD(p)	7	0.02	59	0.18	955	5.85	**35198**	376.06
$\forall c$	SD(d)	**5**	0.02	**52**	0.2	824	5.73	43278	438.81
$\forall c$	SD^2(p+d)	7	0.03	54	0.17	999	6.05	49702	477.04
$\forall c$	SD^2(p)	7	0.02	58	0.18	959	5.84	35201	**368.87**
$\forall c$	SD^2(d)	**5**	0.02	**52**	0.19	**823**	5.80	43452	432.89

Table 2. No. of backtracks (fails) and running time to find all solutions, or prove that there are no solutions, to four instances of QG3 problem. Runtimes are for ILOG Solver 5.300 on 1200MHz, Pentium III processor, and 512 MB of RAM.

runtimes are obtained with the channelling constraints and branching on the primal or dual variable with smallest domain. Being forced to branch on just the primal or dual tends to increase runtimes. The branching heuristic therefore profits from the multiple viewpoints. Note that maintaining GAC on the all-different constraint is neither the best startegy in terms of failures or runtimes. This is despite the fact that it has the strongest propagator. This model has only one viewpoint and this hinders the branching heuristic. Note also that the smallest search trees (but not runtimes) are obtained with models that combines the all-different constraint on the primal with the channelling constraints between the primal and dual. In such models, we have the benefits of the strongest propagator and a dual viewpoint for the branching heuristic. Finnally, note that the model with just an all-different constraint only has primal variables, and gives the same search tree as the model with the all-different constraint and channelling when it is forced to branch on just the primal variables. The pruning performed by the all-different constraint subsumes that performed by the channelling [3, 4].

5.2 Quasigroups

The quasigroups existence problem can be modelled as a multiple permutation problem with $2n$ intersecting permutation constraints. We introduce a variable for each entry in the multiplication table of the quasigroup. We then post permutation constraints on each row and column of variables. In Table 2, we give results for the QG3 family of problems. All the models and branching heuristics except the primal not-equals models are competitive. A dual viewpoint doesn't appear to offer much advantage, but it also does not hurt. In Table 3, we give results for the QG4 family of problems. All the models and branching heuristics are again competitive except the primal not-equals models and those models which force the branching heuristic to branch on a dual variable. Note also that the best runtime on the largest problem is with the double smallest domain heuristic.

5.3 Golomb rulers

To model the Golomb rulers problem as a permutation problem, we introduce a variable for each pairwise distance between marks. Since we may have more values than variables, we introduce additional variables to ensure that there are as many variables as values. Geelen advocates such a construction in [1] as we can then post a permutation constraint on the enlarged set of variables. In Table 4, we give results for finding four optimal length rulers. Despite the fact that it has the strongest propagator, the primal all-different model is not competitive on the larger problems. The best runtimes are obtained with the channelling constraints and branching on the primal or dual variable with smallest domain. Being forced to branch on just the primal variables hurts the branching heuristic.

model	heuristic	QG4(6)		QG4(7)		QG4(8)		QG4(9)	
		fails	sec.	fails	sec.	fails	sec.	fails	sec.
$\neq$	SD(p)	6	**0.01**	82	0.23	1779	8.29	116298	843.26
$\forall$	SD(p)	**4**	**0.01**	**57**	**0.19**	**892**	5.12	52419	496.24
c	SD(p+d)	6	0.02	59	0.20	935	4.99	55232	489.89
c	SD(p)	6	**0.01**	59	0.20	931	4.92	55397	485.72
c	SD(d)	6	0.02	74	0.21	1266	7.59	83316	772.17
c	SD^2(p+d)	6	0.02	59	**0.19**	940	**4.81**	55264	**476.66**
c	SD^2(p)	6	**0.01**	59	**0.19**	936	4.87	55442	478.48
c	SD^2(d)	6	**0.01**	73	0.22	1267	7.37	82916	766.33
$\forall c$	SD(p+d)	**4**	0.02	57	**0.19**	900	5.19	**52045**	486.72
$\forall c$	SD(p)	**4**	0.02	57	0.20	**892**	5.29	52419	491.54
$\forall c$	SD(d)	**4**	0.02	67	0.21	1102	7.04	73997	745.09
$\forall c$	SD^2(p+d)	**4**	**0.01**	57	**0.19**	905	5.24	52077	491.45
$\forall c$	SD^2(p)	**4**	**0.01**	57	0.20	897	5.23	52463	493.70
$\forall c$	SD^2(d)	**4**	**0.01**	66	0.23	1104	7.02	73714	745.86

Table 3. No. of backtracks (fails) and running time to find all solutions, or prove that there are no solutions, to four instances of QG4 problem. Runtimes are for ILOG Solver 5.300 on 1200MHz, Pentium III processor, and 512 MB of RAM.

model	heuristic	Golomb(7,25)		Golomb(8,34)		Golomb(9,44)		Golomb(10,55)	
		fails	sec.	fails	sec.	fails	sec.	fails	sec.
$\neq$	SD(p)	912	0.15	5543	1.12	–	–	–	–
$\forall$	SD(p)	500	**0.11**	2949	**0.81**	–	–	–	–
c	SD(p+d)	606	0.12	3330	1.01	17002	**7.54**	72751	**49.14**
c	SD(p)	890	0.15	5343	1.25	–	–	–	–
c	SD(d)	626	0.12	3390	1.02	17151	7.55	73539	49.25
c	SD^2(p+d)	608	0.12	3333	1.03	17022	7.63	72853	49.37
c	SD^2(p)	928	0.17	5648	1.27	–	–	–	–
c	SD^2(d)	626	0.12	3390	1.03	17179	7.59	73628	49.59
$\forall c$	SD(p+d)	**493**	0.12	**2771**	1.10	**14313**	8.29	**61572**	54.63
$\forall c$	SD(p)	500	0.13	2949	1.08	–	–	–	–
$\forall c$	SD(d)	495	0.13	2782	1.10	14325	8.28	61616	54.46
$\forall c$	SD^2(p+d)	504	0.14	2787	1.1	14392	8.38	61898	54.94
$\forall c$	SD^2(p)	542	0.14	3258	1.12	–	–	–	–
$\forall c$	SD^2(d)	495	0.13	2794	1.11	14400	8.39	61893	54.97

Table 4. No. of backtracks (fails) and running time to find all solutions, or prove that there are no solutions, to four instances of Golomb rulers problem. Runtimes are for ILOG Solver 5.300 on 1200MHz, Pentium III processor, and 512 MB of RAM. A dash means that no results were returned after 1 hour.

		Sport(6)		Sport(8)		Sport(10)		Sport(12)	
model	heuristic	fails	sec.	fails	sec.	fails	sec.	fails	sec.
$\neq$	SD(p)	**0**	**0.00**	1248	0.22	1863275	397.70	5777382	1971.92
$\forall$	SD(p)	**0**	0.01	566	0.15	1361686	350.92	3522705	1444.44
c	SD(p+d)	624	0.09	4	**0.01**	**7**	**0.03**	5232	**1.78**
c	SD(p)	**0**	**0.00**	566	0.14	1376143	355.99	3537447	1368.84
c	SD(d)	589	0.07	**3**	**0.01**	336	0.07	6368	1.9
c	SD^2(p+d)	7	**0.00**	9	**0.01**	1112	0.30	46122	18.4
c	SD^2(p)	113	0.02	6601	0.94	820693	168.91	–	–
c	SD^2(d)	514	0.06	43	**0.01**	7028	1.58	6252	2.29
$\forall c$	SD(p+d)	624	0.10	4	**0.01**	**7**	**0.03**	**5190**	1.98
$\forall c$	SD(p)	**0**	0.01	566	0.16	1361686	372.10	3522705	1495.41
$\forall c$	SD(d)	589	0.09	**3**	**0.01**	329	0.08	6262	2.18
$\forall c$	SD^2(p+d)	7	**0.00**	9	**0.01**	1102	0.35	45125	20.98
$\forall c$	SD^2(p)	113	0.02	6563	1.09	812696	186.23	–	–
$\forall c$	SD^2(d)	514	0.07	43	0.02	6920	1.76	6129	2.55

Table 5. No. of backtracks (fails) and running time to find first solution to four instances of sport scheduling problem. Runtimes are for ILOG Solver 5.300 on 1200MHz, Pentium III processor, and 512 MB of RAM.

		Magic(3)		Magic(4)		Magic(5)		Magic(6)	
model	heuristic	fails	sec.	fails	sec.	fails	sec.	fails	sec.
$\neq$	SD(p)	6	**0.00**	20	**0.00**	1576	0.11	–	–
$\forall$	SD(p)	**4**	**0.00**	19	**0.00**	1355	0.11	2748609	196.45
c	SD(p+d)	5	**0.00**	18	**0.00**	4637	0.37	–	–
c	SD(p)	**4**	**0.00**	20	**0.00**	1457	0.14	3448162	249.84
c	SD(d)	5	**0.00**	37	0.01	49312	4.61	–	–
c	SD^2(p+d)	5	**0.00**	10	**0.00**	555	0.06	**463865**	**37.41**
c	SD^2(p)	**4**	**0.00**	11	**0.00**	495	**0.05**	1648408	132.35
c	SD^2(d)	5	**0.00**	18	**0.00**	928217	86.07	–	–
$\forall c$	SD(p+d)	5	0.01	18	**0.00**	4436	0.48	–	–
$\forall c$	SD(p)	**4**	**0.00**	19	**0.00**	1355	0.17	–	–
$\forall c$	SD(d)	5	**0.00**	**5**	**0.00**	42426	5.33	–	–
$\forall c$	SD^2(p+d)	5	0.02	10	0.01	435	0.07	290103	39.01
$\forall c$	SD^2(p)	**4**	**0.00**	11	**0.00**	**355**	**0.05**	1083993	148.73
$\forall c$	SD^2(d)	5	**0.00**	16	**0.00**	919057	106.55	–	–

Table 6. No. of backtracks (fails) and running time to find the first solution to four instances of magic square problem. Runtimes are for ILOG Solver 5.300 on 1200MHz, Pentium III processor, and 512 MB of RAM. A dash means that no results were returned after 1 hour.

5.4 Sport scheduling

The sport scheduling problem is modelled as follows. The set of teams is $T = \{1, \ldots, n\}$ (we assume n is even), the set of weeks is $W = \{1, \ldots, n-1\}$, the set of periods is $P = \{1, \ldots, n/2\}$, and the set of slots ("home" and "away") are $S = \{1, 2\}$. A schedule is then a *bijection* from $P \times S$ into T for each week such that all the other constraints of the problem are satisfied. We report results in Table 5. Unlike the previous tables which report results to find all solutions, here we report results to find just the first solution. Despite this significant change in the experimental setup, we observe similar trends in our results. Even though it has the strongest propagator, the primal all-different model is again not competitive on the larger problems. The best runtimes are obtained with the channelling constraints and branching on the primal or dual variable with smallest domain. As with the Golomb ruler problem, being forced to branch on just the primal variables hurts the branching heuristic. Multiple viewpoints appear to offer the branching heuristic very significant advantages on this problem.

5.5 Magic squares

We model the order n magic square problem as a matrix model in which there is an n by n matrix of variables which take values from 1 to n^2. We then post permutation constraint on all the variables in such a matrix, and sum constraints on the rows, columns and diagonals. Results are given in Table 6, again to find the first solution. The best strategy is the double smallest domain heuristic on either the model with just channelling constraints, or on the model with channelling constraints and a primal all-different constraint. The former explores a larger search tree, but does so very slightly quicker than the later. We conjecture that the large domain sizes in this problem favour a branching heuristic like double smallest domain which chooses its values with care.

6 Conclusion

On permuation problems, branching heuristics can be significantly more effective when they look at both the primal and dual viewpoint. Indeed, branching on primal or dual variables was often more important to our results than using a stronger propagator. For example, the model that enforced GAC on an all-different constraint often gave worse performance both in runtime and search tree size compared to the model that enforced AC on the channelling constraints. With the later model, the branching heuristic was able to use the multiple viewpoints to make better branching decisions. We also studied the double smallest domain heuristic [7]. This branches on the primal/dual variable with smallest domain and assigns it the value whose dual/primal variable has the smallest domain. This makes decisions which are consistent with the fail-first principle for variable ordering and the succeed-first principle for value ordering. On some of our problem domains, it offered the best performance of all the heuristics studied.

What general lessons can be learnt from these experiments? First, we have strong support for the hypothesis that branching heuristics can profit from multiple viewpoints. Second, our experimental results suggest that you should not necessarily aim for more propagation. For instance, we usually saw better performance when we threw out the all-different constraint. Third, when we model, we need to think about both the heuristics and the propagation. It would be interesting in the future to study the benefits of multiple viewpoints for branching heuristics on problems other than permutations where there might not be such a natural dual viewpoint.

Acknowledgments

The authors are supported by the Science Foundation Ireland. They wish to thank the members of the APES research group.

References

1. Geelen, P.: Dual viewpoint heuristics for binary constraint satisfaction problems. In: Proceedings of the 10th ECAI, European Conference on Artificial Intelligence (1992) 31–35
2. Cheng, B., Choi, K., Lee, J., Wu, J.: Increasing constraint propagation by redundant modeling: an experience report. Constraints **4** (1999) 167–192
3. Smith, B.: Modelling a Permutation Problem. In: Proceedings of ECAI'2000 Workshop on Modelling and Solving Problems with Constraints. (2000) Also available as Research Report from http://scom.hud.ac.uk/staff/scombms/papers.html.
4. Walsh, T.: Permtuation problems and channelling constraints. In: Proceedings of 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2001). (2001)
5. Régin, J.C.: A filtering algorithm for constraints of difference in CSPs. In: Proceedings of the 12th National Conference on AI, American Association for Artificial Intelligence (1994) 362–367
6. Smith, B.: Succeed-first or Fail-first: a case study in variable and value ordering heuristics. In: Proceedings of the Third International Conference on Practical Applications of Constraint Programming (PACT-97). (1997) Available as Research Report 96.26, School of Computer Studies, University of Leeds.
7. Jourdan, J.: Concurrent constraint multiple models in CLP and CC languages: toward a programming methodology by modelling. In: Proceedings of the INFORMS Conference. (1995)