First-Solution Search with Symmetry Breaking and Implied Constraints

Steve Prestwich
Cork Constraint Computation Centre
University College, Cork, Ireland
s.prestwich@cs.ucc.ie

introduction...

2 important issues in constraint modeling:

implied constraints (IC)

- part of manual modeling
- can be generated automatically

symmetry breaking (SB)

- add constraints to model
- add constraints during search
- guide search to non-symmetric subspaces

...introduction...

good past results:

- IC shown to speed up backtrack and local search

- SB is often used with backtracking, especially to show unsatisfiability (eg pigeon hole problem)

- SB can speed up 1-solution search

...introduction...

a few negative results:

- SB can interfere with dynamic variable ordering

- SB may guide the search away from easily-found solutions

- SB and IC may add overheads

...introduction...

other anomalies: backtracking can interact badly with

- domain pruning
- arc consistency preprocessing
- removal of inconsistent or redundant domain values or subproblems

effects of "good" techniques can be surprising — what about SB and IC?

...introduction

some unanswered questions:

- are SB and IC always good when searching for one solution (modulo overheads)?
- does the answer depend on whether we use backtracking or local search?
- does it depend on the problem?
- does it ever make sense to add SB or remove IC?

some experiments...

experiments...

3 combinatorial problems: *clique*, *set cover*, *BIBD*

all can be naturally modeled as CSPs using Boolean variables

but the models contain complicated constraints: expensive, hard to SAT-encode

can reformulate the CSP using integer variables, eliminating the difficult constraints

this introduces extra symmetries (permutations on the assignments): add SB constraints, which imply other constraints

...experiments

all performed on direct SAT encodings of reformulated CSPs

compare 3 distinct algorithms:

- SATZ backtracker (actually SATZ-RAND without use of restarts)

- WSAT local search (SKC version)

- CLS randomised backtracker

search parameters roughly optimised to each problem class

cliques...

a graph $G = (V, E)$ (vertices, edges); two vertices connected by an edge are *adjacent*; a *clique* is a subset of V whose vertices are pairwise adjacent

natural model:

- Boolean variables $v_1, \ldots, v_n$, one for each vertex in G, $v_i = T$ denotes vertex i is in the clique

- constraint: $\neg v_i \vee \neg v_j$ ($i < j$, i and j are non-adjacent vertices)

- size constraint:

$$|\{v_i \in G : \ v_i = \text{true}\}| = k$$

last constraint hard to use

…cliques…

reformulation:

- integer variables $v_1, \ldots, v_k \in \{1, \ldots, n\}$; $v_i = j$ denotes vertex j is in the clique

- constraint: no two non-adjacent vertices may be in the clique:

  $v_i \neq j \vee v_{i'} \neq j'$
  ($i < i'$, j and j' are non-adjacent vertices)

...cliques...

highly symmetrical model: $k!$ representations of each clique

can be removed by adding SB constraints to order the values

3 ways:

$$\begin{array}{lr} 1. & v_i < v_{i+1} \\ 2. & v_i < v_j \quad (i < j) \\ 3. & v_i + j - i - 1 < v_j \quad (i < j) \end{array}$$

call these models 1–3, and the problem without SB model 0

model 1 constraints $\Rightarrow$ model 2 constraints $\Rightarrow$ model 3 constraints

...cliques...

problems: generate 5 random (IID) graphs with 50 vertices, density 0.5 and random seeds 1–5

apply a clique algorithm to each graph to find small clique size

seed	k	variables	clauses_0	clauses_1	clauses_2	clauses_3
1	7	350	27475	34075	53200	54880
2	8	400	36632	44157	70932	73606
3	8	400	36128	43653	70428	73102
4	7	350	27643	34243	53368	55048
5	8	400	35568	43093	69868	72542

...cliques...

results:

problem	SATZ back.	sec.	WSAT flips	sec.	CLS back.	sec.
clique1_0	360	1.48	236	0.046	268	0.338
clique2_0	11719	24.5	1032	0.13	2678	1.92
clique3_0	4332	9.25	384	0.063	626	0.656
clique4_0	641	1.89	294	0.047	637	0.525
clique5_0	5570	11.2	303	0.055	661	0.656
clique1_1	22	1.55	2146	0.21	9130	7.13
clique2_1	139	3.80	129127	12.3	652375	575
clique3_1	1.4	1.12	13428	1.30	118309	103
clique4_1	3.25	1.10	7317	0.66	21466	16.5
clique5_1	27	2.05	13150	1.26	43704	37.6
clique1_2	—	—	701	0.12	1349	1.88
clique2_2	—	—	14450	2.50	47620	64.5
clique3_2	—	—	1865	0.37	9459	13.1
clique4_2	—	—	1533	0.22	3263	4.02
clique5_2	—	—	2783	0.52	7694	10.8
clique1_3	—	—	667	0.12	1036	1.56
clique2_3	—	—	11911	2.16	50973	71.2
clique3_3	—	—	2149	0.44	14044	19.8
clique4_3	—	—	1078	0.18	3119	4.00
clique5_3	—	—	1593	0.35	7107	20.2

...cliques...

summary of results:

- SB slows WSAT but IC helps slightly
- SB slows CLS but IC helps slightly
- SB help SATZ but IC drastically slows it

set covers...

given ground set of objects V and a set E of subsets of V; a *cover* for V is a subset C of E such that every member of V also occurs in some member of C; find a cover of size k

natural model:

- Boolean variable v_i for each subset $e_i \in E$, denotes membership of e_i in cover C

- constraint (each member of V is covered by some member of C): for each $x \in V$ where $x \in e_{i_1} \cap \ldots \cap e_{i_m}$, $v_{i_1} \vee \ldots \vee v_{i_m}$

- size constraint:

$$|\{v_i \in C : \; v_i = \text{true}\}| = k$$

again a difficult constraint

...set covers...

reformulation:

- k integer variables v_i with domains $\{1, \ldots, |E|\}$
- $v_i = j$ denotes subset e_j is in the cover
- constraint: $v_i \neq v_j$ $(1 \leq i < j \leq k)$
- constraint (each element of V is covered by at least one element of E): for each $x \in V$ where $x \in e_{i_1} \cap \ldots \cap e_{i_m}$:

$$(v_1 = i_1 \ \vee \ \ldots \ \vee \ v_k = i_1) \ \vee \ \ldots \ \vee$$
$$(v_1 = i_m \ \vee \ \ldots \ \vee \ v_k = i_m)$$

- again 4 SB models

...set covers...

problems: ground set $|V| = 20$, $|E| = 40$, $e \in E$ contains 3 random members of V, each has (near?-)minimal cover size $k = 7$

apply transformation to break up clauses with > 30 literals

seed	variables	clauses_0	clauses_1	clauses_2	clauses_3
1	324	6371	11051	22751	24116
2	332	6379	11059	22759	24124
3	322	6369	11049	22749	24114
4	332	6379	11059	22759	24124
5	328	6375	11055	22755	24120

...set covers...

results:

	SATZ		WSAT		CLS	
problem	back.	sec.	flips	sec.	back.	sec.
cover1_0	—	—	59949	1.29	41296	3.55
cover2_0	—	—	131499	2.81	29475	2.49
cover3_0	—	—	29244	0.624	18254	1.63
cover4_0	—	—	64869	1.38	35438	2.92
cover5_0	—	—	33100	0.708	20071	1.72
cover1_1	—	—	77493	2.33	33703	4.64
cover2_1	—	—	142498	4.13	26957	3.57
cover3_1	—	—	27581	0.822	15570	2.26
cover4_1	—	—	48828	1.48	31084	4.09
cover5_1	—	—	35977	1.16	16001	2.17
cover1_2	895	6.73	59939	3.28	188328	46.0
cover2_2	495	5.26	87025	5.32	107205	23.7
cover3_2	223	2.93	50991	2.82	69562	17.5
cover4_2	579	5.44	29598	1.72	98698	21.2
cover5_2	908	7.22	27615	1.51	20981	4.99
cover1_3	822	6.43	57771	3.45	120490	32.3
cover2_3	449	4.83	105213	6.54	60257	14.3
cover3_3	201	2.77	19861	1.17	117322	31.8
cover4_3	366	4.38	43942	2.66	84472	19.2
cover5_3	531	5.38	19630	1.15	23838	5.98

...set covers

summary of results:

- IC helps SATZ, SB helps but only with IC
- SB slows WSAT, IC neutral
- SB slows CLS, IC slows CLS

very different to clique results

BIBDs...

specified by 5 parameters (v, b, r, k, λ): find a $v \times b$ binary matrix with exactly r ones per row, k ones per column, and scalar product λ between any 2 rows

natural model:

- Boolean variable for each matrix entry
- v b-ary constraints for r ones per row
- b v-ary constraints for the k ones per column
- $v(v-1)/2$ $2b$-ary constraints for scalar products

again difficult constraints

...BIBDs...

reformulation:

- integer variables to denote (i) positions in each row of the r ones, (ii) those of the $b-r$ zeroes, (iii) positions in each column of the k ones, (iv) those of the $v-k$ zeroes, (v) shared positions of the λ ones in each pair of rows

- binary constraints only: no 0 in same place as a 1 in each row and column, 0s/1s must agree in rows/columns, no shared 1 coincides with a 0

- again 4 SB models

...BIBDs...

problem	variables	$clauses_0$	$clauses_1$	$clauses_2$	$clauses_3$
1	833	7028	8106	9674	10080
2	1260	13650	16245	20625	21745
3	2646	40418	48013	72366	79093
4	2700	36756	41940	64728	71865
5	3640	58520	70280	101444	109284

results:

	SATZ		WSAT		CLS	
problem	back.	sec.	flips	sec.	back.	sec.
$bibd1_0$	—	—	51412	0.88	43961	6.63
$bibd2_0$	—	—	121259	2.53	69704	15.1
$bibd3_0$	—	—	2028247	65.0	620790	264
$bibd4_0$	—	—	6382697	163	313356	140
$bibd1_1$	57.1	0.90	589010	6.44	4569058	745
$bibd2_1$	14523	32.6	—	—	—	—
$bibd3_1$	—	—	—	—	—	—
$bibd4_1$	—	—	—	—	—	—
$bibd1_2$	16.2	0.82	107279	1.89	347079	61.0
$bibd2_2$	313	1.60	633589	16.0	1967046	575
$bibd3_2$	—	—	11138495	499	—	—
$bibd4_2$	—	—	18387011	644	—	—
$bibd1_3$	459	1.57	113879	1.34	248697	42.3
$bibd2_3$	553	2.22	400273	8.48	1531026	392
$bibd3_3$	—	—	3319906	129	—	—
$bibd4_3$	—	—	3681702	115	—	—

...BIBDs

summary of results:

- problem 5 hard for all algorithms: results are surprisingly bad
- SB drastically slows WSAT but IC helps
- SB drastically slows CLS but IC helps
- SB helps SATZ and IC helps too

again the results are different

discussion...

SB and IC may have good, bad or neutral effects

results consistent within each problem type: effects are not random but depend on properties of the SAT problems

adding SB and IC to a model can have an unpredictable effect on first-solution search

add to collection of anomalous results

guess: IC anomalies caused by interaction with dynamic variable ordering (CLS and SATZ use different orderings, and were anomalous on different problem classes)

...discussion...

main result: *SB consistently slows down local search but helps backtrack search*

so best choice of model depends on type of search algorithm

explanation: could invoke variable ordering but...

it's been pointed out [Clark et al.] [Yokoo] that a problem with fewer solutions should be harder to solve for local search – so SB *should* slow down local search

but they found that local search took longer on random k-SAT, 3-colouring and binary CSPs

the cause was removal of local minima by SB

other results: backtracking slightly helped when there are more solutions

...discussion...

the new results contradict both: removing solutions helped backtrack search but slowed local search

complicates the picture of what makes a good model

begs the question why should removing solutions by SB speed up backtrack search for first solution?

...discussion...

guess: an extension of Minton et al.'s *non-systematic search hypothesis*, which is:

- if solutions occur in clusters then the average separation between solutions is greater than if they were uniformly distributed

- first-solution time for backtrack search increases with the average separation between solutions, so solution clustering slows down backtrack search

- first-solution time for local search depends more on solution density: insensitive to clustering

...discussion...

clustering hypothesis for SB:

- suppose SB reduces cluster size but not number

- then by eliminating empty regions of the search space SB reduces separation between clusters, speeding up backtrack search

- local search still slowed because of reduced solution density

...discussion...

symmetries in the experiments were all permutations, so expect similar clustering

on other problems SB may reduce the number of clusters instead of size, increasing separation between clusters and therefore backtrack search time

more work needed

...discussion

advice to modelers when using local search:

- for highly symmetric problems, try models that preserve symmetries

- for combinatorial problems in general, try models that introduce new symmetries

relevant to large problems where local search may be necessary

we don't always need complex and costly SB constraints, and can select models based on low number of variables and constraints, low constraint arity, usefulness for propagation and cost reasoning...

(previous CLS results used such models, which probably helped)