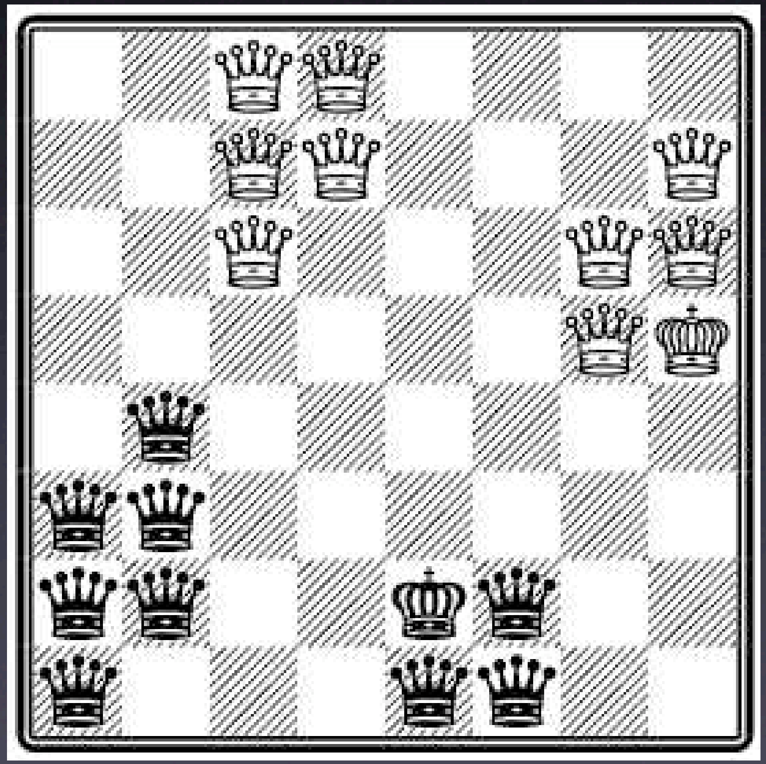
Value Symmetry Breaking

Toby Walsh
NICTA and UNSW

Symmetry

- Symmetry is bijection, σ on assignments that preserves solutions/constraints
  - In armies of queens problem, swap colours!
  - X[1,3]=white queen, X[6,2] =black queen ..
  - X[1,3]=black queen, X[6,2] =white queen ..



Types of symmetry

- Which part of the assignment does the bijection act upon?
  - Variable symmetry
  - Value symmetry
  - Variable/value symmetry

Value symmetry

- Only values are changed
  - E.g. white queen => black queen
  - E.g. blue => red, red => green, ..
  - E.g. AvB => AvC, CvD => BvD
- $(Z[1],Z[2],..) \Rightarrow (\sigma(Z[1]),\sigma(Z[2]),..)$

Symmetry breaking

- General method for variable symmetries
  [Crawford, Ginsberg, Luks and Roy KR96]
  - Look for lexicographically least assignment
    - (Z[1],Z[2],...) $\leq_{lex}$ (Z[σ(1)],Z[σ(2)],...)
    - reversal symmetry:
      (X[1],X[2],...,X[n-1],X[n]) $\leq_{lex}$
      (X[n],X[n-1],...,X[2],X[1])

Adding constraints

- Same method works with value symmetries [Walsh CP06]
  - Look for lex least assignment
  - $(Z[1],Z[2],\ldots) \leq_{lex} (\sigma(Z[1]),\sigma(Z[1]),\ldots)$
  - Simple propagator for this global constraint based on a ternary decomposition

Adding constraints

- Same method works with symmetries in general [Walsh CP06]
  - Including those that act both on variables and values simultaneously
  - Look for assignment that is lex smaller than all its symmetries
  - So, we're done? Symmetry solved problem?

Adding constraints

- No! Too many constraints in general
  - For instance, *m* interchangeable values gives *m!* symmetry breaking constraints
  - Look for special cases where we can do better

Special cases

- Value symmetry
  - Interchangeable values
- Variable symmetry
  - Row and column symmetry

Interchangeable values

- Often we have some (sub)set of values which can be freely interchanged
  - {golfer1, golfer2,...}
  - {white queen, black queen}
- Given m values, m! symmetries
  - Cannot post LEX LEADER constraints for every symmetry!

Generator symmetries

- Post just LEX LEADER for generator of symmetry group
  - Suppose 1 to m are interchangeable
  - One set of generators are permutations (1 i)
  - Posting just these LEX LEADER constraints leaves symmetry
    - Consider: X1=1, X2=2 and X1=1,X2=3

Generator symmetries

- Post just LEX LEADER for generator of symmetry group
  - Suppose 1 to m are interchangeable
  - Another set of generators are permutations (i i+1)
  - Think bubble sort!
  - Posting just these LEX LEADER constraints breaks all symmetry

Generator symmetries

- Post just LEX LEADER for generator of symmetry group
  - Suppose 1 to m are interchangeable
  - Another set of generators are permutations (i i+1)
  - Enforcing GAC on these LEX LEADER constraints does not prune all symmetric values
  - $X1=1, X2 \in \{1,2\}, X3 \in \{1,3\}, X4 \in \{1,4\}, X5=5$

Value precedence

- Order 1st time we use a value [Law & Lee CP04]
  - 1,1,2,1,3,2,1,2,4 .... satisfies value precedence
  - 1,1,2,1,4,2,1,2,3 .... does not
- Breaks all symmetry due to interchangeable values

Enforcing value precedence

- Puget's method
  - Introduce Zi for position at which i first used
    - If Xi=j then Zj≤i
    - If Zj=i then Xi=j
  - Order Zi
    - Zi < Zi+1

Enforcing value precedence

- Puget's method
  - Introduce Zi for position at which i first used
  - Order Zi
- Decomposes problem into binary constraints
  - Hinders propagation
  - Consider: $X1=1$, $X2 \in \{1,2\}$, $X3 \in \{1,3\}$, $X4 \in \{3,4\}$, $X5=2$, $X6=3$, $X7=4$

Value precedence

- Linear time method to ensure value precedence [Walsh ECAI06]
  - Introduce sequence of variables, Y[i] for largest value used so far by X[i]
  - X[i]: 1,1,2,1,3,2,1,..
  - Y[i]: 1,1,2,2,3,3,3,..

Value precedence

- Linear time method to ensure value precedence [Walsh ECAI06]
  - Introduce sequence of variables, Y[i] for largest value used so far by X[i]
  - $X[i+1] \leq Y[i+1]+1$
  - $Y[i+1] = \max(X[i],Y[i])$

Value precedence

- Linear time method to ensure value precedence [Walsh ECAI06]
  - $X[i+1] \leq Y[i+1]+1$
  - $Y[i+1] = \max(X[i],Y[i])$
  - Consider: $X1=1$, $X2 \in \{1,2\}$, $X3 \in \{1,3\}$, $X4 \in \{1,4\}$, $X5=5$

Value precedence

- Value precedence implies lex least assignment
  - Consider assignment: 1,1,2,1,3,2,..
  - Take any permutation, σ of 1 to n
  - Suppose σ(1)=1, σ(2)=2, σ(3)=5
  - $(1,1,2,1,3,2,..) \leq_{lex}$ $(1,1,2,1,5,2,..)$

Value precedence

- Lex least assignment implies value precedence
  - X[1]=1 otherwise suppose X[1]=2, & consider $\sigma(2)=1$ and (2,...) $\leq_{lex}$ (1,...)
  - X[2]=1 or 2 otherwise consider $\sigma(1)=1$, $\sigma(3)=2$ and (1,3,..) $\leq_{lex}$ (1,2,..)
  - ...

Value precedence

- Lex least assignment equivalent to value precedence
  - One value precedence constraint equivalent to exponential number of lex ordering constraints
  - Very effective means to break symmetry of interchangeable values

Value precedence

- Map value symmetry into variable symmetry
  - X[i]=j iff Z[i,j]=1
  - Value precedence iff cols lex ordered
  - Consider (1,1,2,1,3,2)
  - Why not use lex chain?
    - Rows also must sum to 1
    - Consider X[1]=1, X[2]∈{1,2,3,4}, X[3]∈{1,2,3,4},X[4]=4

Dynamic methods

- Relatively easy to expand tree so we don’t visit symmetric nodes
  - GE-tree, SBDD, ..
- Basic rule: only use one new value
  - X1=1
  - X2=1 or 2
  - X3=1 or 2 or 3 ..

Dynamic methods

- Dynamic methods can be exponentially slower than static methods
  - Consider pigeonhole problem:
  - $X1, .. Xn \in \{1,..,n+1\}$
  - $\forall i . 1 \leq i \leq n+1 \Rightarrow X1=i \vee .. \vee Xn=i$

Dynamic methods

- Dynamic methods can be exponentially slower than static methods
  - Dynamic methods essentially only do forward checking on next variable
  - Do not prune deeper variables
  - No interaction between problem constraints and symmetry breaking constraints

Extensions to value precedence

- Disjoint sets of interchangeable values
  - E.g. car assembly line sequencing
  - values 1,2,.. cars with sunroofs
  - values a,b,.. cars without
  - 1,1,a,2,a,b,1,a,c,3,.. satisfies value precedence as both 1,1,2,1,3,.. and a,a,b,a,c do

Extensions to value precedence

- Two sets of interchangeable values
  - O(nd^2) time method to ensure value precedence [Walsh ECAI06]
  - Introduce sequence of variables, Y[i] for largest pair of values used so far by X[i]
    - X[i]: 1, 1, a, 2, b, 1, ..
    - Y[i]: (1,_), (1,_), (1,a), (2,a), (2,b), (2,b) ..

Extensions to value precedence

- *k* sets of interchangeable values
  - O(nd^*k*) time method to ensure value precedence [Walsh ECAI06]
  - If *k*=O(n) this is not polynomial!
  - In fact, enforcing GAC in this case is NP-hard
  - Breaking value symmetry is intractable!

Breaking value symmetry is NP-hard

- Reduction of SAT to value precedence
  - values 4i-3, 4i-2 are interchangeable
    - represent xi=true
  - values 4i-1,4i are interchangeable
    - represent xi=false

Breaking value symmetry is NP-hard

- Truth assignment
  - $X_i \in \{4i-3, 4i-1\}$
    - representing $x_i \in$ {true, false}
    - for instance, $X_i=4i-1$ in CSP iff x_i=false in SAT problem

Breaking value symmetry is NP-hard

- CSP variables to represent clauses
  - Suppose n Boolean variables in SAT problem and ith clause is $x_j \vee \neg x_k$
  - Then $X_{n+i} \in \{4j-2, 4k\}$
  - Can only use 4j-2 if 4j-1 appears earlier
  - In other words only if xj=true in truth assignment

Breaking value symmetry is NP-hard

- Reduction of SAT to value precedence
  - truth assignment
    - $X_i \in \{4i-3, 4i-1\}$
  - clause variables, ith clauses is $x_j \vee \neg x_k$
    - $X_{n+i} \in \{4j-2, 4k\}$
  - Consider $\{x_1, \neg x_1 \vee x_2\}$

Breaking value symmetry is NP-hard

- Domains not symmetric!
  - $X_i \in \{4i-3, 4i-1\}$
  - $X_i \in \{4i-3, 4i-2, 4i-1, 4i\}$
  - Switch var: $X_{n+m+1} \in \{4n+1, 4n+2\}$
  - $\text{Even}(X_{n+m+1}) \Rightarrow \text{Odd}(X_i)$

Breaking value symmetry is NP-hard

- Add constraints to CSP so it has the right value symmetries
  - Even(X_{n+m+1}) ⇒ unsat
    - Unsatisfiable problem has every symmetry
  - Odd(X_{n+m+1}) ⇒ Φ
    - Φ can be anything with correct value symmetries (e.g. pigeonhole problem)

Dynamically breaking value symmetry

- Pruning all symmetric values statically is NP-hard
  - Dynamic methods can break all symmetry (ie not visit symmetric states) in polynomial time
  - Dynamic methods only forward check
  - Can take exponential time on problems that can be solved using static methods in polynomial time

Breaking value symmetries in general

- LEX LEADER constraints
  - May be exponential number of such constraints
- Puget’s method
  - Works on any value symmetry, not just interchangeable values

Breaking value symmetries in general

- Puget's method
  - Breaks *any* value symmetry using polynomial number of constraints
  - But may do worse than specialized methods that exploit structure of symmetry group
    - E.g. value precedence for symmetry of interchangeable values

Puget’s method

- Detour: CSP with variable symmetry in which variables are all different
  - Map value symmetry into such a CSP
- All different problems occur frequently
  - Rehearsal problem: each scene is rehearsed once and only once ..

Puget's method

- Need some more group theory
  - Given a group S
  - Stabilizer of i, stab(i) = $\{\sigma \in S \mid \sigma(i)=i\}$
  - For example, if S is all possible permutations of 1 to n then
    - (2 3) is in stab(4) ...

Puget's method

- If we have an all-different constraint, we can simplify the LEX LEADER constraints
  - Consider (2 3) (4 5)
  - $<X1,X2,X3,X4,X5> \leq_{lex}$ $<X1,X3,X2,X5,X4>$
  - Simplifies to X2 $<$ X3

Puget's method

- If we have an all-different constraint, we can simplify the LEX LEADER constraints
  - In general, let $j = \min\{i \mid \sigma(i) \neq i\}$
  - Then the LEX LEADER constraint for σ simplifies to:
    - $X[j] < X[\sigma(j)]$
  - We can have at most a quadratic number of such constraints!

Puget's method

- How to compute these ordering constraints efficiently?
  - Use the (famous) Schreier Sims algorithm for computing stabilizer chains and coset representatives
  - In fact, some of the ordering constraints are redundant and we need a linear number at most

Puget's method

- Use the (famous) Schreier Sims algorithm for computing coset representatives
  - $U1 = \{\sigma(1) \mid \sigma \in S\}$
  - $U2 = \{\sigma(2) \mid \sigma \in S, \sigma(1)=1\}$
  - $U3 = \{\sigma(3) \mid \sigma \in S, \sigma(1)=1, \sigma(2)=2\}$
  - ...

Puget’s method

- Use the (famous) Schreier Sims algorithm for computing coset representatives
  - $Ui = \{\sigma(i) \mid \sigma \in S, \forall j<i \,.\, \sigma(j)=j\}$
  - LEX LEADER constraints simplify to
    - $X[i] < X[j]$ for $j \in Ui \setminus \{i\}$

Puget's method

- Example: gracefully labelling K3 x P2
  - Graceful graph has unique label for each vertex, f(x)
    - Constraint that |f(x)-f(y)| is unique for each edge (x,y) in the graph

Puget's method

- Example: gracefully labelling K3 x P2
  - Variable for each vertex, symmetries:
  - (1,2,3,4,5,6), (1,3,2,4,6,5), (2,3,1,5,6,4), (2,1,3,5,4,6), (3,1,2,5,4,5), (3,2,1,6,5,4), (4,5,6,1,2,3), (4,6,5,1,3,2), ...

Puget's method

- Example: gracefully labelling K3 x P2
  - U1 = {σ(1) | σ∈S} = {1,2,3,4,5,6}
  - U2 = {σ(2) | σ∈S, σ(1)=1} = {2,3}
  - U3 = {σ(3) | σ∈S, σ(1)=1, σ(2)=2} = {3}
  - U4 = {4}
  - U5 = {5}

Puget’s method

- Example: gracefully labelling K3 x P2
  - U1 = {1,2,3,4,5,6}, U2 = {2,3}, U3 = {3}, U4 = {4}, U5 = {5}
- LEX LEADER simplifies to:
  - X1<X2, X1<X3, X1<X4, X1<X5, X1<X6
  - X2< X3
  - Note: X1<X3 is redundant

Puget’s method

- From quadratic to linear number of ordering constraints
  - Remove redundant constraints entailed by transitivity of <
  - For each j, if $\exists$ i<j. j $\in$ Ui then let
    - k = max{i | j $\in$ Ui, i<j}
    - Post Xk < Xj

Puget's method

- For each j, if ∃ i<j. j ∈ Ui then let
  - k = max{i | j ∈ Ui, i<j}, post Xk < Xj
- Example: gracefully labelling K3 x P2
  - U1 = {1,2,3,4,5,6}, U2 = {2,3}, U3 = {3}, U4 = {4}, U5 = {5}
  - j=2, k=1, X1<X2
  - j=3, k=2, X2<X3 (nb X1<X3 redundant)
  - j=4, k=1, X1<X4 ..

Puget's method

- So, we can break all variable symmetries with polynomial number of ordering constraints for an all-different problem
  - What's this got to do with breaking value symmetry?
  - Map value symmetry into variable symmetry on all-different problem

Puget's method

- Map value symmetry into variable symmetry on all-different problem
  - Introduce Z[j] for position at which j first used
    - If X[i]=j then Z[j]≤i
    - If Z[j]=i then X[i]=j
    - If some value un-used, introduce dummy indices (or add additional X[i] so all values are used)

Puget's method

- Map value symmetry into variable symmetry on all-different problem
  - Introduce Z[j] for position at which j first used
    - Z[j] are all-different (as only one value at each position!)
  - Value symmetry on X[i] becomes variable symmetry on Z[j]

Puget's method

- Map value symmetry into variable symmetry on all-different problem
  - Can break all such variable symmetry with linear number of binary ordering constraints
  - And quadratic number of channelling constraints between X[i] and Z[j]
  - Of course, no free lunch. This decomposition may hinder propagation!

Puget’s method

- Map value symmetry into variable symmetry on all-different problem
  - For completely interchangeable values
  - Gives Z[j] < Z[j+1]
  - Value precedence (values first appear in order)

Conclusions

- Symmetry occurs in many problems
    - We must deal with it or face a combinatorial explosion!
- We have a generic method (for small numbers of symmetries)
    - In special cases, we can break all symmetries

Questions?