Symmetry Breaking

Toby Walsh
NICTA and UNSW

Outline

- What
  - What is symmetry?
- Why
  - Why is symmetry a problem?
- How
  - How do we deal with symmetry?

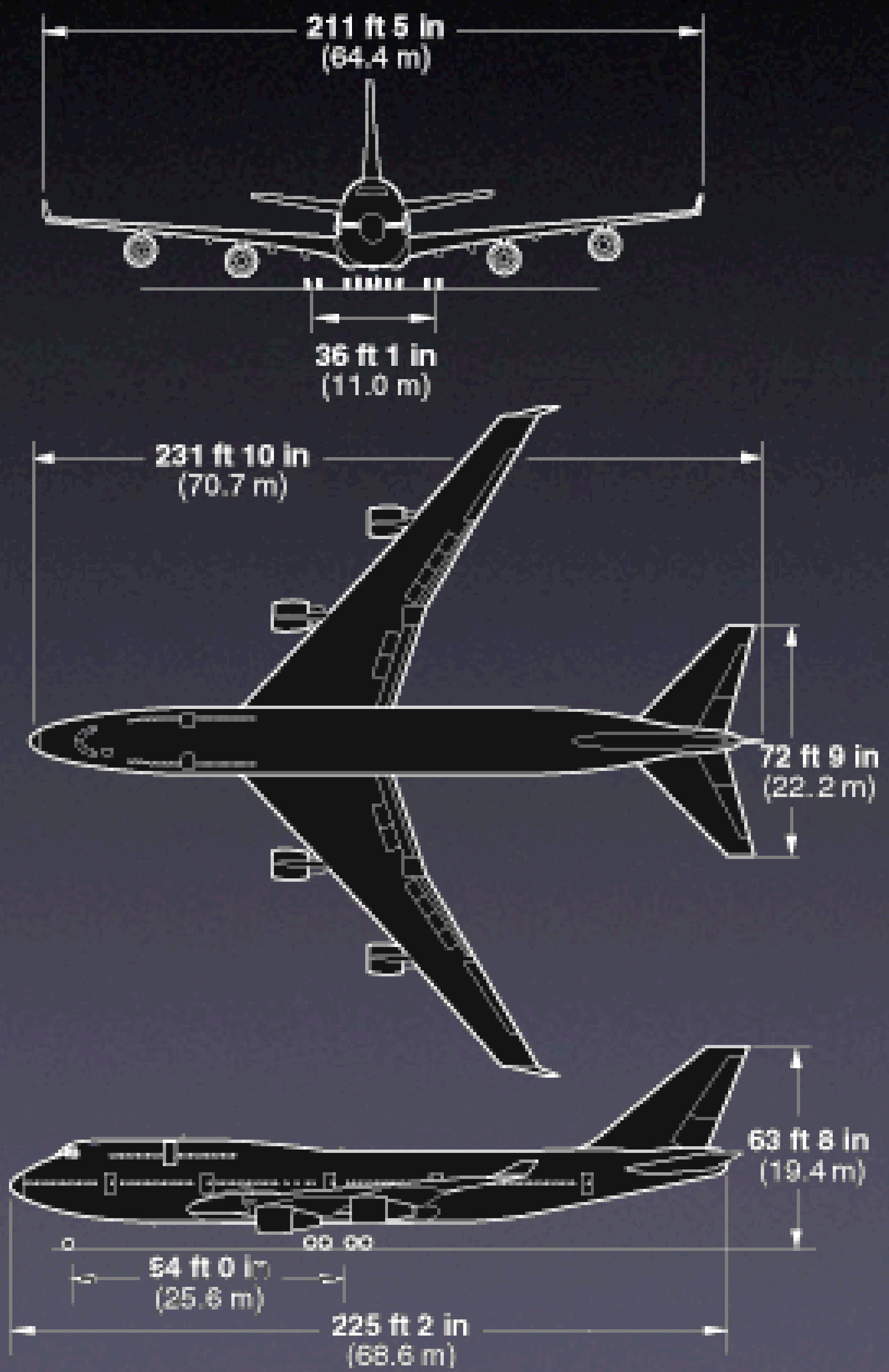
Apology

- Symmetry in constraint programming
- But similar ideas will apply to other domains:
  - Combinatorial optimization
  - Planning
  - Search
  - ...

Active research area

- SymCon'01 workshop, Cyprus 2001
- SymCon'02 workshop, Ithaca 2002
- SymCon'03 workshop, Kinsale 2003
- SymCon'04 workshop, Toronto 2004
- SymCon'05 workshop, Sitges 2005
- SymCon'06 workshop, Nantes 2006
- 1st International Symmetry Conference, Edinburgh 2007

Symmetry



- Within objects
  - Design an airplane
  - Boeing 747 has reflection symmetry

Symmetry

- Between objects
  - Scheduling problem
  - Fleet of identical 747's

Graph colouring

- Variable for each county
  - Italy, France, ...
  - Values are colours
- Constraints
  - Italy≠Switzerland, Italy≠France, ..

Graph colouring

- Proper colouring
  - Italy=green
  - France=blue
  - Spain=red

Graph colouring

- Symmetric colouring
  - Italy=blue
  - France=red
  - Spain=green

Graph colouring

- Symmetric colouring
  - If there are m colours
  - m! symmetric solutions

Peaceable armies of coexisting queens

- Place 9 queens and 1 king of each colour on chessboard
- No piece to attack another of the opposite colour

Armies of queens

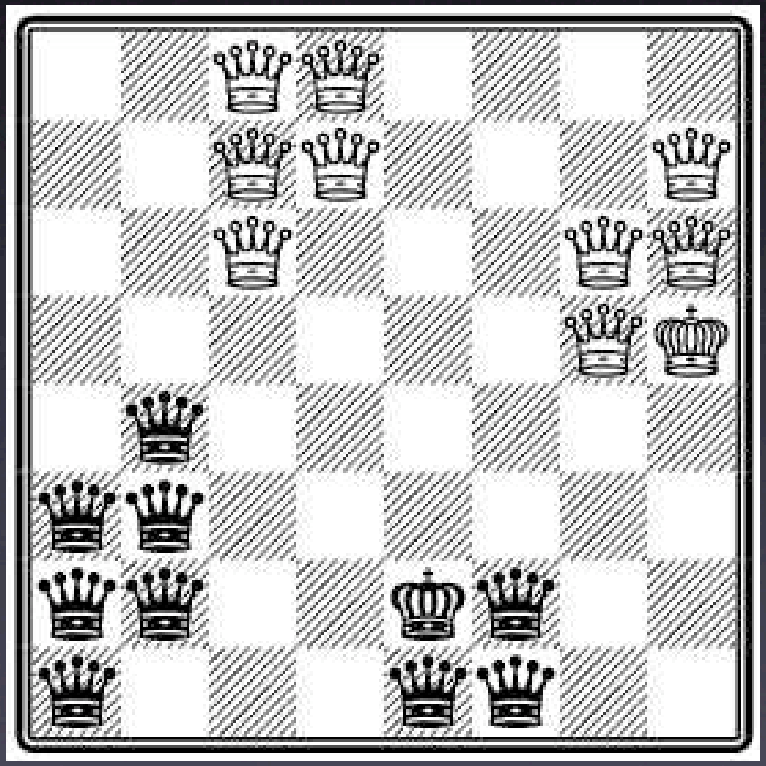
- Set of variables
  - X[i,j] for the square on ith row, jth col
- Set of values
  - {white queen, black queen, empty}

Armies of queens

- Set of constraints
  - X[i,j]=white queen => X[i,k]≠black queen
  - X[i,j]=white queen => X[k,j]≠black queen
  - X[i,j]=white queen => X[i+1,j+1]≠black queen
  - ...

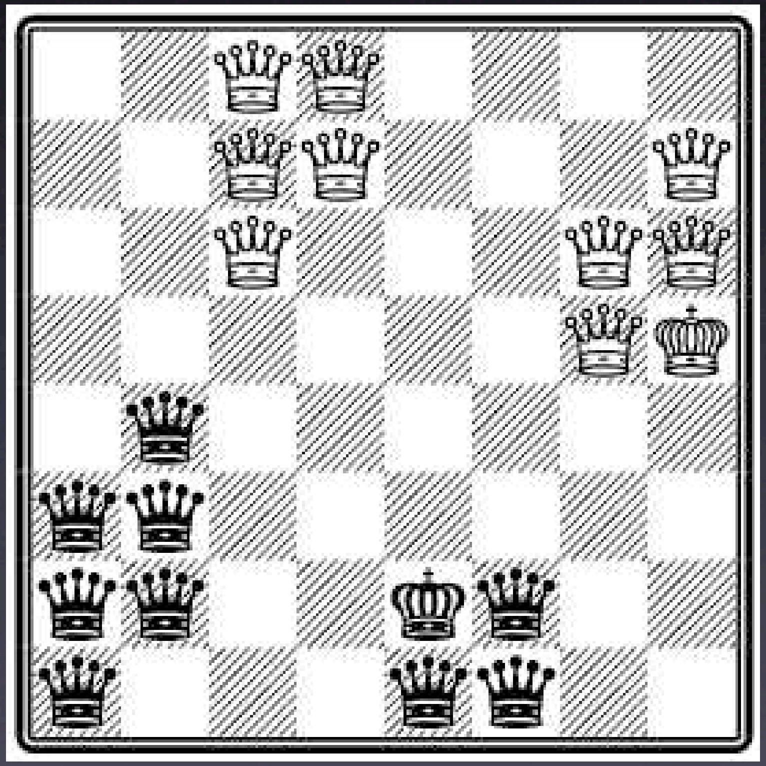
Peaceable armies

- Gives this to a constraint solver
  - Here's one solution!



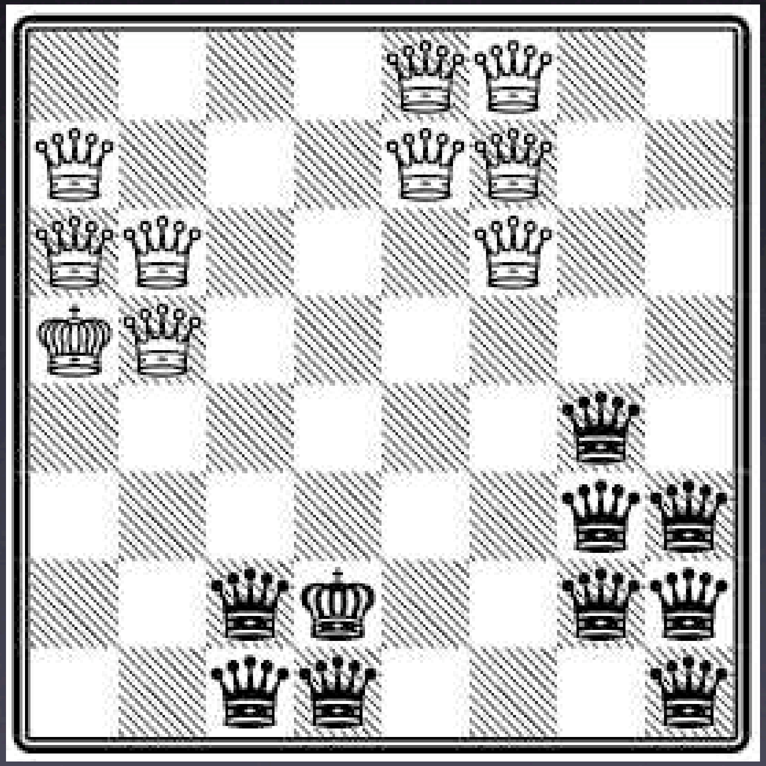
Peaceable armies

- Symmetries of chessboard give other solutions
  - horizontal reflection



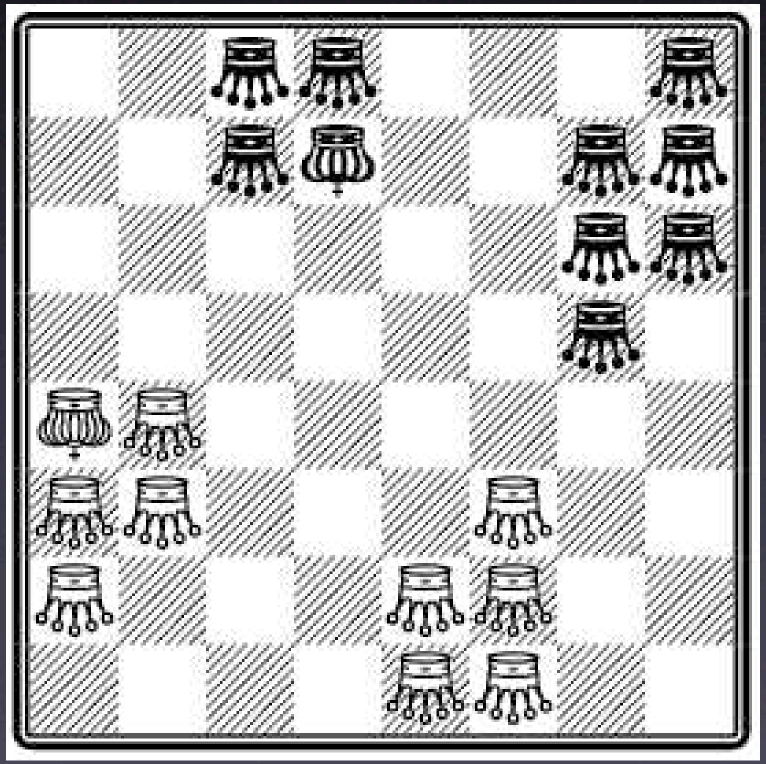
Peaceable armies

- Symmetries of chessboard give other solutions
  - horizontal reflection



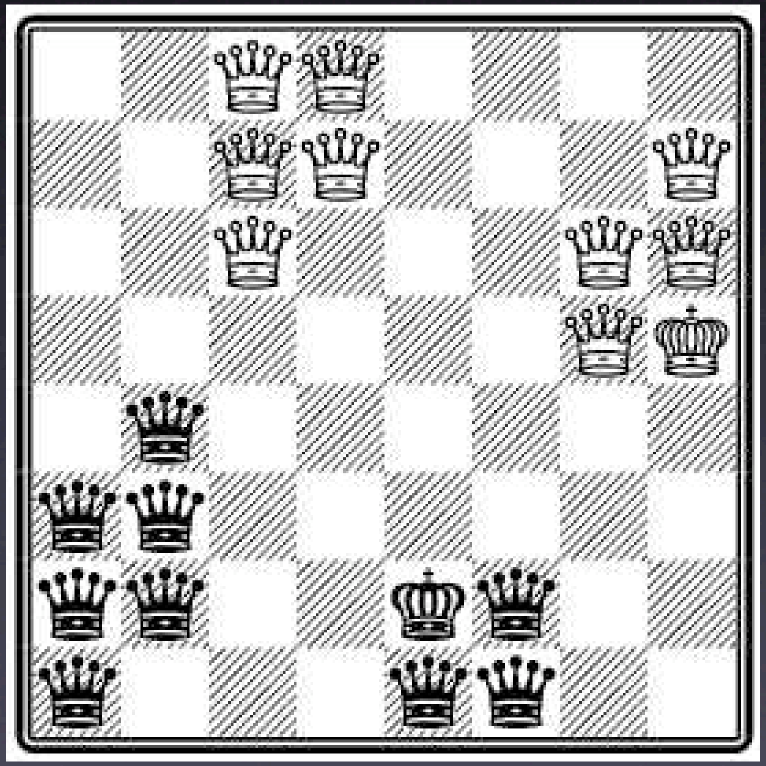
Peaceable armies

- Symmetries of chessboard give other solutions
  - vertical reflection



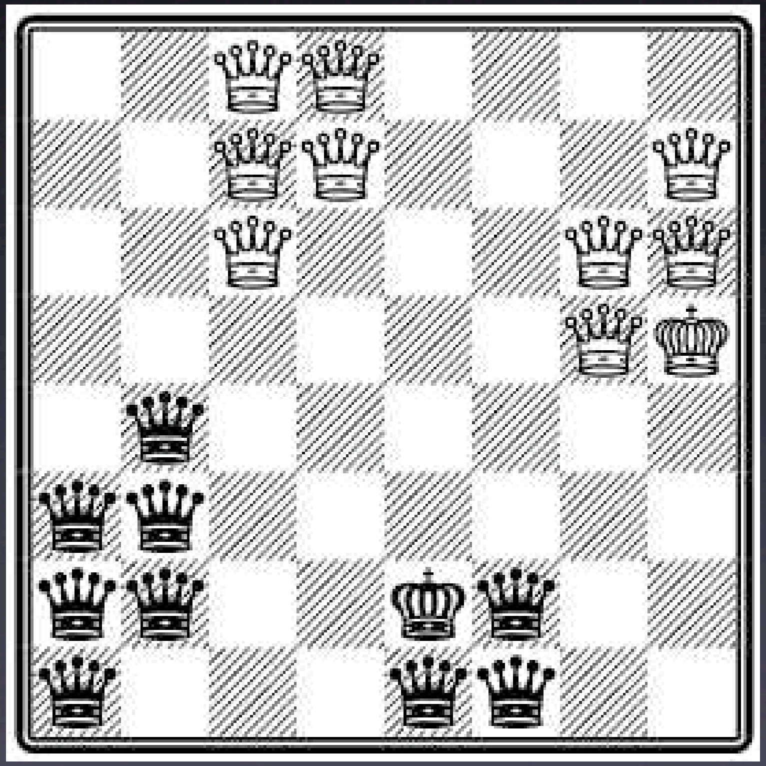
Peaceable armies

- Symmetries of chessboard give other solutions
  - diagonal reflections



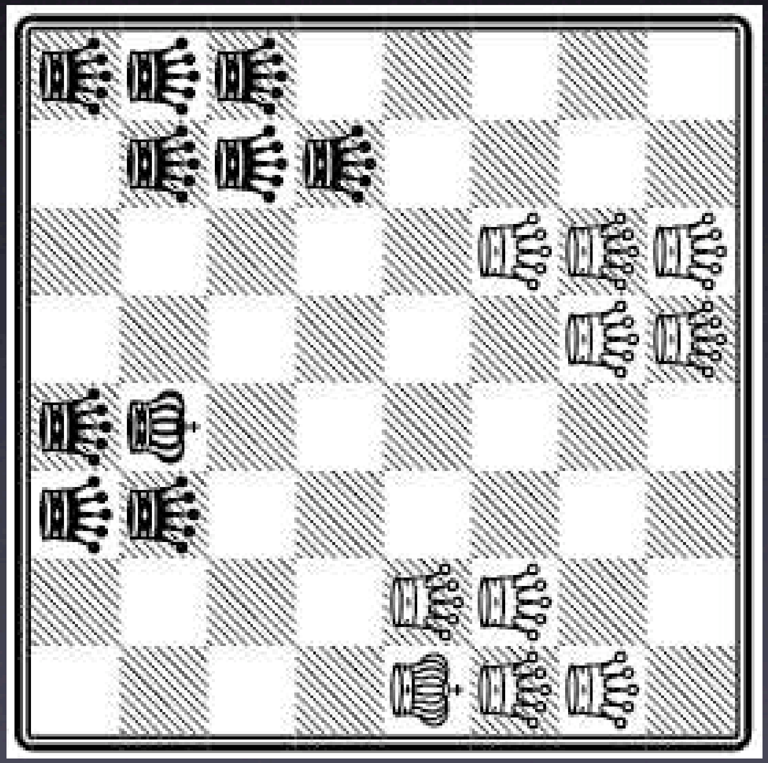
Peaceable armies

- Symmetries of chessboard give other solutions
  - rotation 90 degrees



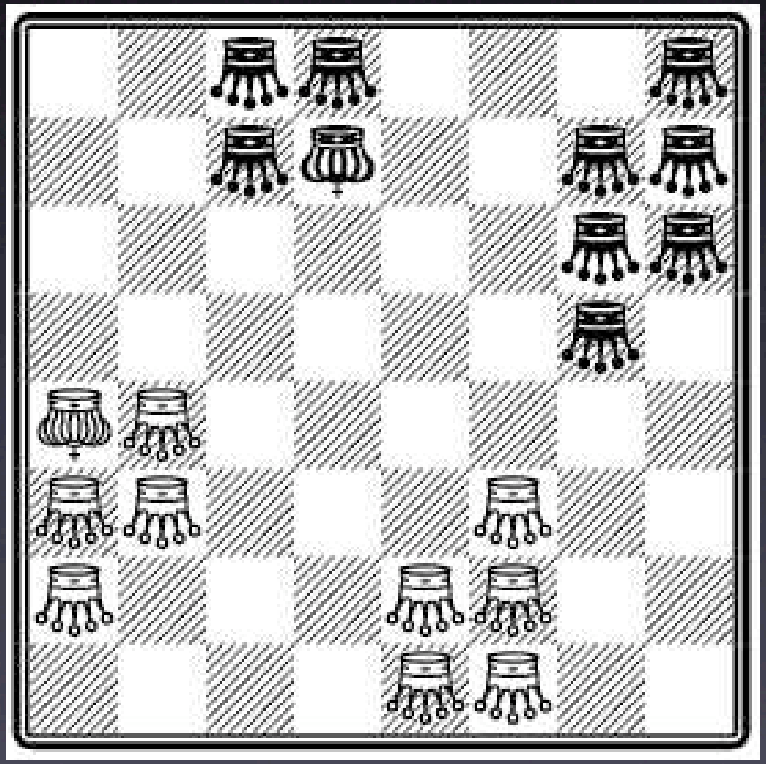
Peaceable armies

- Symmetries of chessboard give other solutions
    - rotation 90 degrees



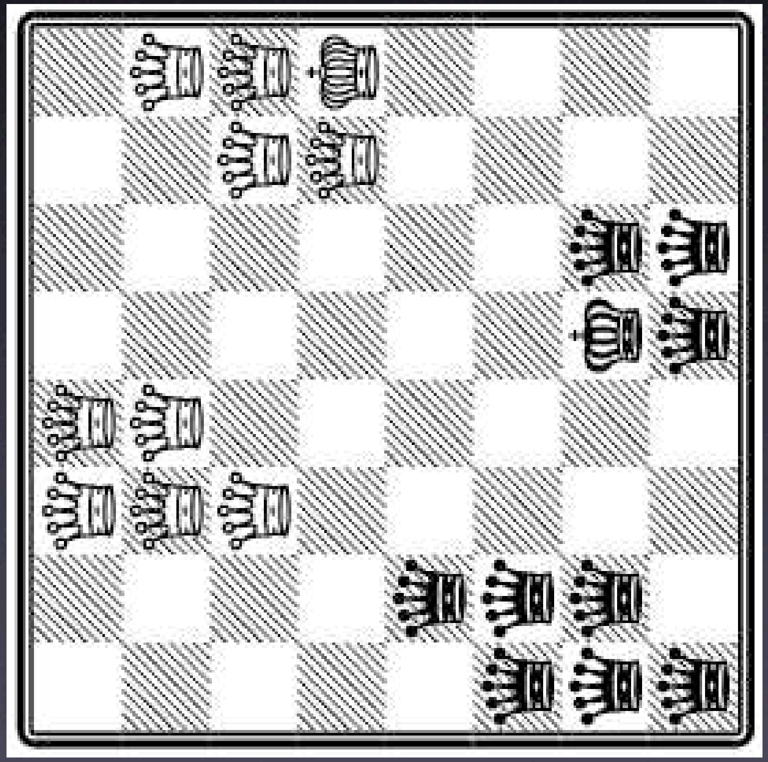
Peaceable armies

- Symmetries of chessboard give other solutions
  - rotation 180 degrees



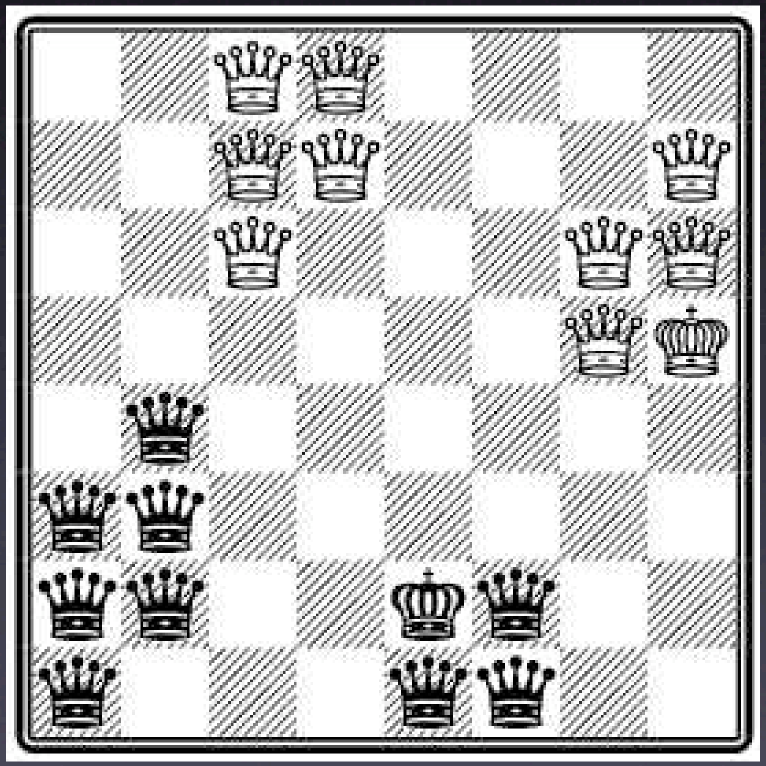
Peaceable armies

- Symmetries of chessboard give other solutions
  - rotation 270 degrees



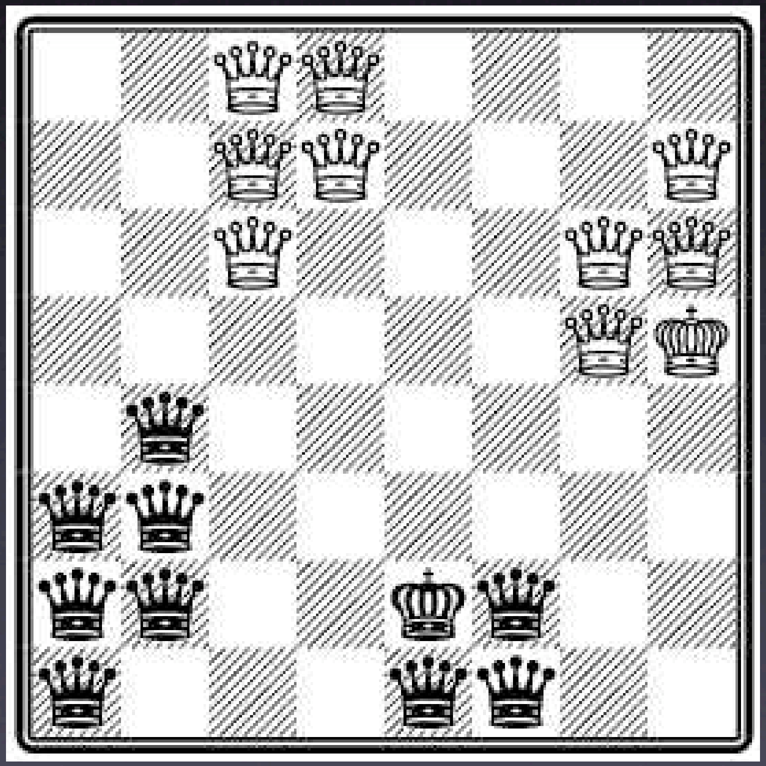
Peaceable armies

- Symmetries of pieces
  - permute any pair of white (or black) queens
  - permute all white pieces with black



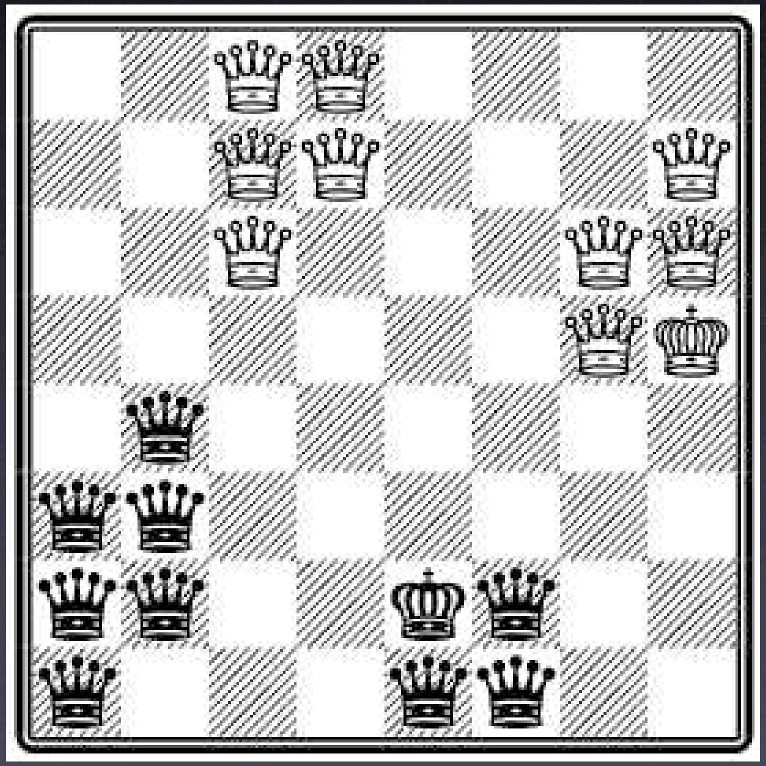
Peaceable armies

- Difficult optimization problem
  - Unique solution up to symmetry!
  - 2,106,910,310,400 symmetric solutions
  - 1/4 US national debt in $



Peaceable armies

- Difficult optimization problem
  - Unique solution up to symmetry!
  - 2,106,910,310,400 symmetric solutions
  - Don't want to visit symmetric search states



Social golfers

"Johnson here is the 4th for our team. He's not accurate, but he hits the ball a mile."

- 32 golfers play once a week in a foursome
- Each week they want to meet 3 different people
- How many weeks can they play?

Social golfers

- 11 weeks is infeasible
  - You meet 3 new players each week
  - There are only 31 other players
- 10 weeks is possible

Social golfers

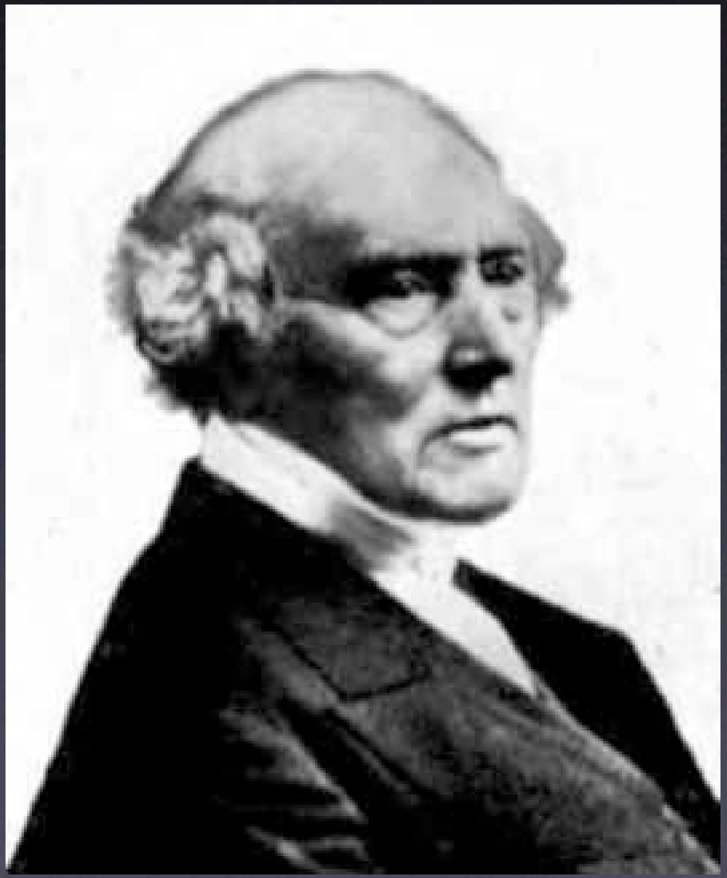
- Difficult optimization problem
  - Golfers symmetric
  - Weeks symmetric
  - Order of groups and of foursome irrelevant
  - 32! x10! x 8! x 4! symmetries = 923988455532966699771808443174160957 440000000000 = (mass of Universe in kg)

Social golfers

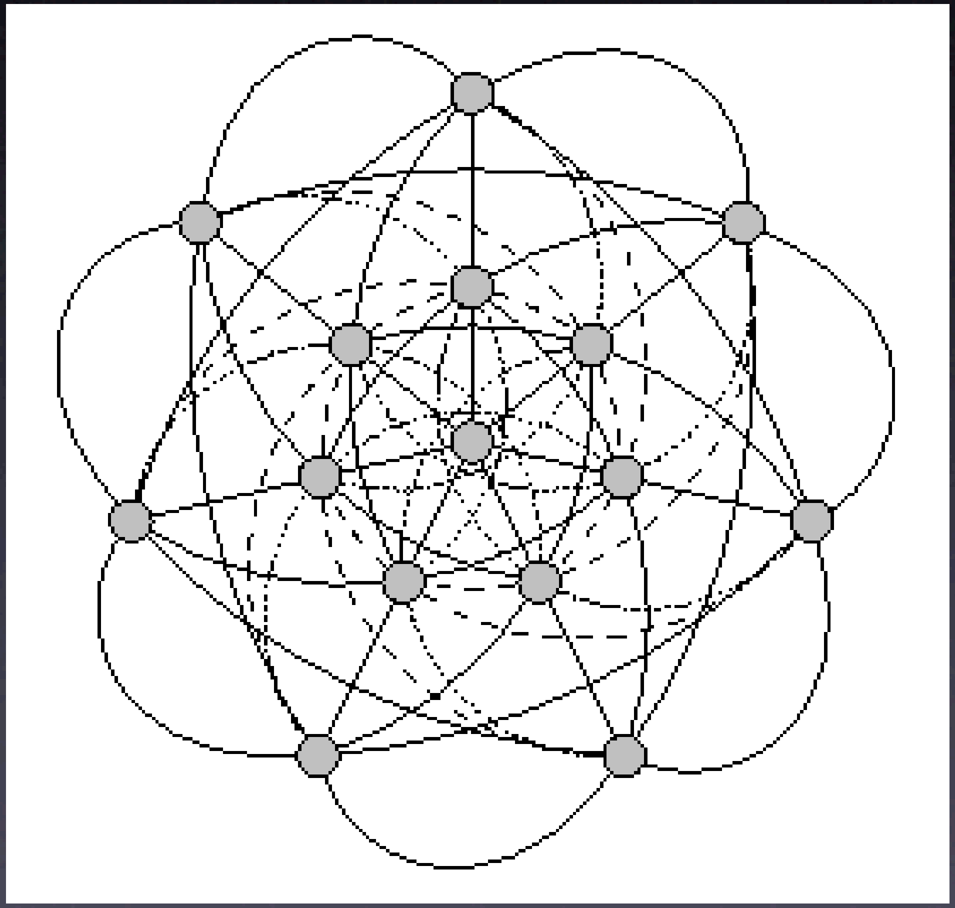
- Simple generalization: (g,s,w) problem
  - g groups
  - groups of size s
  - w weeks

Schoolgirl problem

- Proposed by Rev. Thomas Penyngton Kirkman in the "Ladies and Gentleman's diary" in 1850
    - 15 girls walk in 5 groups of 3 each day for a week. How can the girls be arranged so they walk together with different girls?

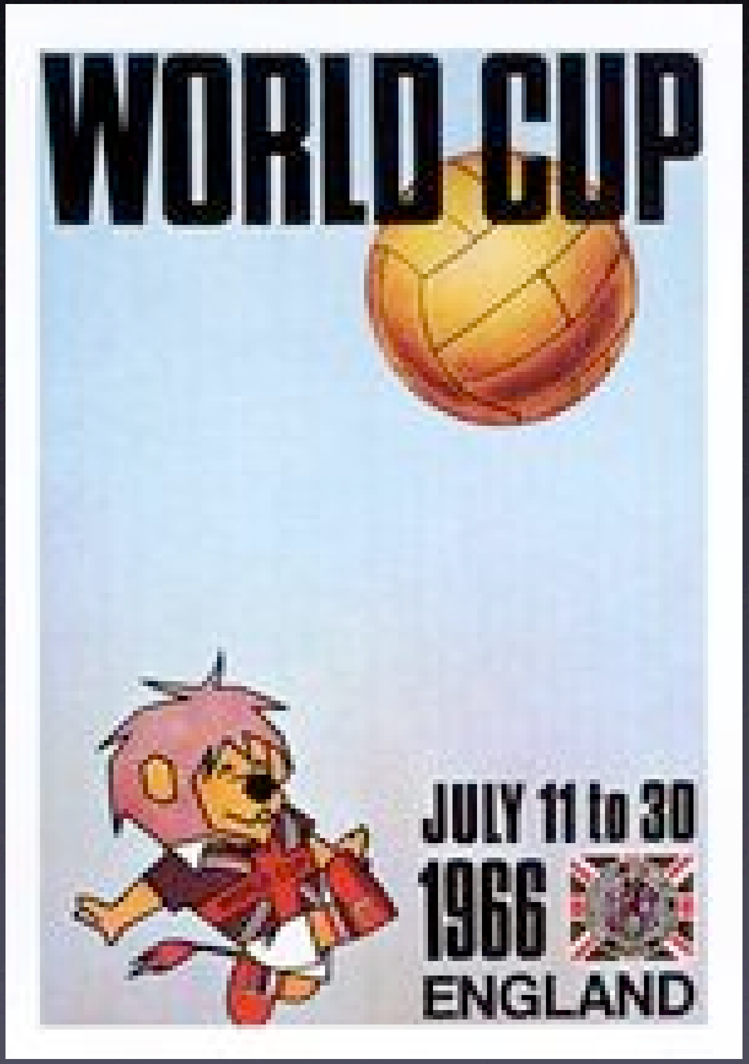


Schoolgirl problem



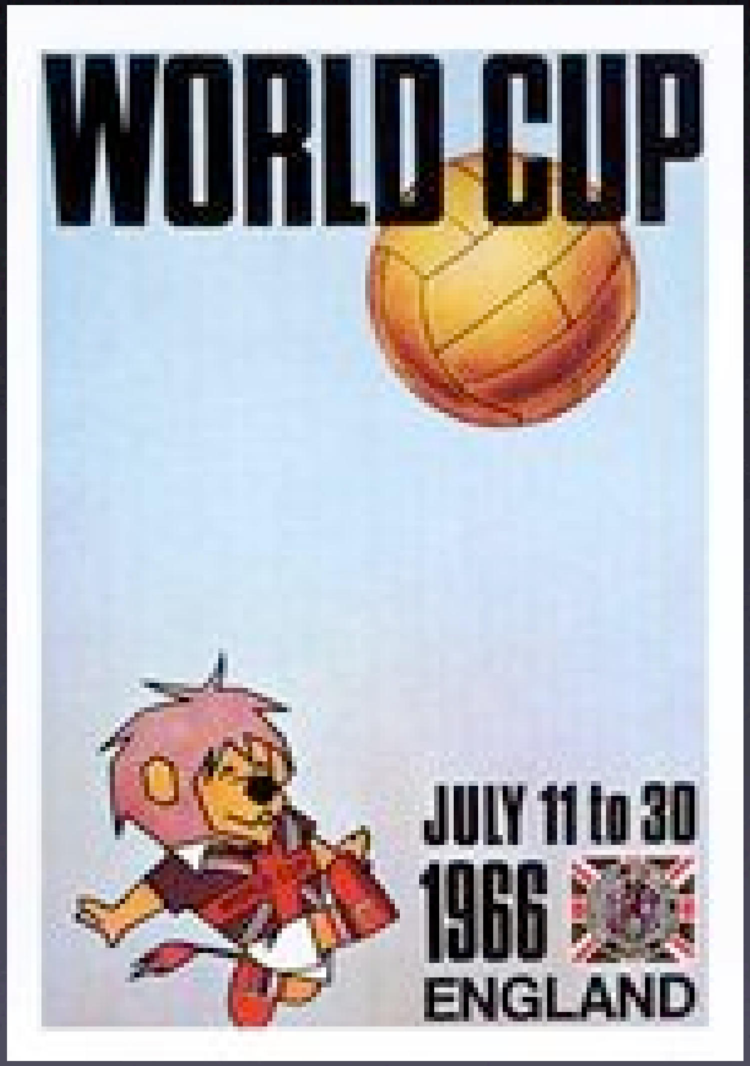
- (5,3,7) problem
  - Special type of balanced incomplete block design
- Again lots of symmetry
  - girls, groups, days, ...

Tournament scheduling



- Imagine scheduling an event like 1st round of World Cup
  - Suppose 4 venues
  - 8 teams
  - 7 matches (or rounds)

Tournament scheduling



- Often lots of other constraints
  - “Home” v “Away” matches
  - TV rights
  - ..

Tournament scheduling

- Set of variables
  - Match[i,j] is match played in venue i on round j
- Set of values
  - {AvB, AvC, AvD, ..}

Tournament scheduling

- Set of constraints
  - Each team plays once in each round
  - Each team plays every other team
  - ...

Tournament scheduling

- Again lots of symmetry
  - Venues
  - Teams
  - Rounds
- 4! x 8! x 7! = 4,877,107,200

Symmetry

- Scheduling
  - Identical machines, orders
- Rostering
  - Equally skilled workers
- Vehicle routing
  - Identical trucks

Symmetry

- Define in terms of *bijection* on assignments
- Bijection is mapping $\sigma: A \rightarrow B$ that is:
  - Injective: $\sigma(x)=\sigma(y) \Rightarrow x=y$
  - Surjective (onto): $\forall b \in B \ \exists a \in A \ . \ \sigma(a)=b$
  - Also known as *permutation* when A=B

Symmetry

- Bijection σ:A→A
  - A={ <Italy,red>, <Italy,blue>, <France,red>, <France,blue>, ...}
  - σ(<Italy,red>) = <Italy,blue>
  - σ(<Italy,blue>) = <Italy,red>
  - σ(<France,red>) = <France,blue>
  - ..

Symmetry in CP

- Solution symmetry
  - Bijection on assignments that preserves solutions (and non-solutions)
- Constraint symmetry
  - Bijection on assignments that preserves constraints

Symmetry in CP

- Solution symmetry
  - even(X1+X2), even(X2+X3)
  - consider $\sigma(\langle X2,*\rangle) = \langle X3,*\rangle$
- Constraint symmetry
  - even(X1+X2), even(X2+X3), even(X1+X3)

Symmetry in CP

- Solution symmetry
  - constraint symmetries $\subset$ solution symmetries
- Constraint symmetry
  - Often the type of symmetries found automatically (using graph isomorphism)

Rotation symmetry

- Symmetry is bijection, σ on assignments that preserves solutions
  - 90 degree rotation
  - X[1,1]=white queen, X[2,3]=black queen .. => X[1,8]=white queen, X[3,7]=black queen ..

Permutation symmetry

- Symmetry is bijection, σ on assignments that preserves solutions
  - Permute venues
  - Match[1,1]=AvB, Match[2,1]=CvD ..=> Match[2,1]=AvB, Match[1,1]=CvD ..

Permutation symmetry

- Symmetry is bijection, σ on assignments that preserves solutions
  - Permute teams
  - Match[1,1]=AvB, Match[2,1]=CvD ..=> Match[1,1]=AvC, Match[2,1]=BvD ..

Types of symmetry

- Variable symmetry
- Value symmetry
- Variable/value symmetry

Types of symmetry

- Variable symmetry
  - Only variables are changed
    - E.g. rotations or reflections of chessboard
    - X[1,1]=>X[1,8], X[2,3]=>X[3,7]
  - Often represent this by permutation of variable indices
    - (Z[1],Z[2],..) => (Z[σ(1)],Z[σ(2)],..)

Types of symmetry

- Value symmetry
  - Only values are changed
    - E.g. white queen => black queen
    - E.g. AvB => AvC, CvD => BvD
  - In general, (Z[1],Z[2],..) => (σ(Z[1]),σ(Z[2]),..)

Types of symmetry

- Symmetry can act on both variables and values simultaneously
  - E.g. 90 degree rotation of 8-Queens problem
  - Row[1]=col2 => Row[2]=col8, ..

Set of symmetries

- Set of symmetries forms a group
- Symmetry breaking exploits group theory
  - generators
  - stabilizers
  - ...

Groups

- Group is set of objects S, and a binary operation •
  - closure: $\forall a,b \in S \,.\, a \bullet b \in S$
  - associativity: $\forall a,b,c \in S \,.\, (a \bullet b) \bullet c = a \bullet (b \bullet c)$
  - identity: $\exists e \in S \; \forall a \in S .\; e \bullet a = a \bullet e = a$
  - inverse: $\forall a \in S \; \exists b \in S \,.\; a \bullet b = b \bullet a = e$

Examples of groups

- C2:
  - {e,s} where s•s=e
- C4:
  - {e,s,s^2,s^3} where s•s=s^2, s^2•s=s^3, s^3•s=e

Examples of groups

- C2:
  - {id, reflect} where reflect•reflect=id
- C4:
  - {id,r90,r180,r270} where r90•r90=r180, r180•r90=r270, r270•r90=id

Example of groups

- Group is set of symmetries S, and a binary operation • which is composition
  - closure: since solution/constraints preserved
  - associativity: composition is associative
  - identity: leave assignments unchanged
  - inverse: invert bijection

Permutation group

- Consider permutations of the set {1,2,3}
  - e = identity, so e(1)=1, e(2)=2, e(3)=3
  - a = (12), so a(1)=2, a(2)=1, a(3)=3
  - b = (23), so b(1)=1,b(2)=3, b(3)=2
- S3 = {e,a,b,ab,ba,aba} forms a group under composition of permutations

Permutation group

- Consider value symmetry in 3 colouring from the set {r,g,b}
  - e = identity
  - a = (r g)
  - b = (g b)
- S3 = {e,a,b,ab,ba,aba} gives the 6 possible permutations of the 3 colours

Group theory

- Generators
  - {e,a,b} generates S3 = {e,a,b,ab,ba,aba}
    - a=(1 2), b=(2 3)
- Not necessarily unique
  - {e,a,aba} also generates S3
    - a=(1 2), aba=(1 3)

Dealing with symmetry

- Don’t want to visit symmetric search states
  - “Identical” solutions
  - “Identical” failing states
- How do we eliminate these from search?

Reformulation

- Change representation
  - WhiteQueen[1]=(1,1), WhiteQueen[2]=(1,2), .., BlackQueen[1]=(5,7), ..
  - X[1,1]=white queen, X[1,2]=white queen, .., X[5,7]=black queen

All interval series

- Order numbers 0 to n-1 so that
  - Each difference between neighbouring numbers occurs once
  - E.g. 0 8 1 7 2 6 3 5 4
  - Diff: 8 7 6 5 4 3 2 1
  - What symmetries does this problem have?

All interval series

- Order numbers 0 to n-1 so that
  - Each difference between neighbouring numbers occurs once
  - E.g. 0 8 1 7 2 6 3 5 4
  - Diff: 8 7 6 5 4 3 2 1
  - Reversal symmetry: 4 5 3 6 2 7 1 8 0

All interval series

- Order numbers 0 to n-1 so that
  - Each difference between neighbouring numbers occurs once
  - E.g. 0 8 1 7 2 6 3 5 4
  - Diff: 8 7 6 5 4 3 2 1
  - Complementation: 8 0 7 1 6 2 5 3 4

Reformulation of AIS

- Cyclic view
  - Order numbers 0 to n-1 in a cycle
  - Each difference 1 to n-1 occurs
  - E.g. 0 8 1 7 2 6 3 5 4
  - Diffs: 8 7 6 5 **4** 3 2 1 **4**
  - What symmetries does this now have?

Reformulation of AIS

- Cyclic view
  - Order numbers 0 to n-1 in a cycle
  - Each difference 1 to n-1 occurs
  - E.g. 0 8 1 7 2 6 3 5 4
  - Diffs: 8 7 6 5 **4** 3 2 1 **4**
  - Reversal symmetry

Reformulation of AIS

- Cyclic view
  - Order numbers 0 to n-1 in a cycle
  - Each difference 1 to n-1 occurs
  - E.g. 0 8 1 7 2 6 3 5 4
  - Diffs: 8 7 6 5 **4** 3 2 1 **4**
  - Complementation symmetry

Reformulation of AIS

- Cyclic view
  - Order numbers 0 to n-1 in a cycle
  - Each difference 1 to n-1 occurs
  - E.g. 0 8 1 7 2 6 3 5 4
  - Diffs: 8 7 6 5 **4** 3 2 1 **4**
  - Rotation symmetry

Reformulation of AIS

- Cyclic view
  - Order numbers 0 to n-1 in a cycle
  - Each difference 1 to n-1 occurs
  - E.g. 0 8 1 7 2 6 3 5 4
  - Diffs: 8 7 6 5 **4** 3 2 1 **4**
  - Symmetry easily broken: sequence starts 0 n-1 1

Reformulation of AIS

- E.g. 0 8 1 7 2 6 3 5 4
- Diffs: 8 7 6 5 **4** 3 2 1 **4**
  - Given solution to cyclic view
    - reverse: 4 5 3 6 2 7 1 8 0
    - complement: 8 0 7 1 6 2 5 3 4
    - both: 4 3 5 2 6 1 7 0 8
    - common diff: 6 3 5 4 **0** 8 1 7 2 (and its symmetries)

Breaking symmetry

- Add symmetry breaking constraints
  - Match[1,1]=AvB
  - Match[2,1]=CvD
  - ...

Rehearsal problem

- X[i] = scene rehearsed in ith time slot
  - Actors must arrive before their first scene and stay till their last scene
- Reflection symmetry
  - Can reverse any rehearsal sequence
  - Prevent this with constraint: X[1] < X[n]

LEX LEADER

- For variable symmetries, [Crawford et al. KR96] give general method:
  - Pick order on vars: X[1] to X[n]
  - For each variable symmetry σ, post LEX LEADER constraint:
  - $(X[1],..X[n]) \leq_{lex} (X[\sigma(1)],..X[\sigma(n)])$

Lexicographical order

- (Y1,Y2,...) ≤lex (Z1,Z2,..) iff
  - Y1<Z1 or
  - Y1=Z1 & (Y2,...) ≤lex (Z2,...)
- Order used in dictionaries, etc
  - (1,1,2,1,2,3,1..) ≤lex (1,1,3,1,3,2,1,..)
- Linear time propagator [Frisch, Hnich, Kiziltan, Miguel, Walsh CP02]

Rehearsal problem

- X[i] = scene rehearsed in ith time slot
  - Actors must arrive before their first scene and stay till their last scene
- Reflection symmetry
  - Can reverse any rehearsal sequence
  - $(X[1],..X[n]) \leq_{lex} (X[n],..X[1])$
  - Simplifies to $X[1] < X[n]$

Non-attacking queens

- X[i,j] ∈ {white queen, black queen, empty}
- 90 rotation symmetry
    - (X[1,1],X[1,2],..,X[1,8],X[2,1],..,X[2,8],..)$\leq_{lex}$ (X[8,1],X[7,1],..,X[1,1],X[8,2],..,X[1,2],..)

Non-attacking queens

- X[i,j] ∈ {white queen, black queen, empty}
- 180 rotation symmetry
  - $(X[1,1],X[1,2],..,X[1,8],X[2,1],..,X[2,8],..) \leq_{lex} (X[8,8],X[8,7],..,X[8,1],X[7,8],..,X[7,1],..)$

Non-attacking queens

- X[i,j] $\in$ {white queen, black queen, empty}
- 270 rotation symmetry
  - (X[1,1],X[1,2],..,X[1,8],X[2,1],..,X[2,8],..)$\leq_{lex}$ (X[1,8],X[2,8],..,X[8,8],X[1,7],..,X[8,7],..)

Non-attacking queens

- $X[i,j] \in$ {white queen, black queen, empty}
- horizontal reflection
  - $(X[1,1],X[1,2],..,X[1,8],X[2,1],..,X[2,8],..) \leq_{lex} (X[8,1],X[8,2],..,X[8,8],X[7,1],..,X[7,8],..)$

Non-attacking queens

- X[i,j] ∈ {white queen, black queen, empty}
- vertical reflection
  - (X[1,1],X[1,2],..,X[1,8],X[2,1],..,X[2,8],..) $\leq_{lex}$ (X[1,8],X[1,7],..,X[1,1],X[2,8],..,X[2,1],..)

LEX LEADER method

- Three challenges
  - Extend method to work with other types of symmetry (e.g. value symmetries)
  - Deal with exponential number of symmetries
  - Conflict between branching heuristic and symmetry breaking constraints

Variable symmetry

- Bijection σ on vars which maps solutions onto solutions
  - E.g. reflection symmetry: X[1]→X[n], X[2]→X[n-1], ...
- LEX LEADER method
  - E.g. (X[1],..X[n]) ≤lex (X[n],..X[1])

Value symmetry

- Bijection ϑ on values which maps solutions onto solutions
  - E.g. suppose two scenes have same actors, then can permute these two scenes (=values) in any rehearsal
- LEX LEADER method
  - $(X[1],..X[n]) \leq_{lex} (\vartheta(X[1]),..\vartheta(X[n]))$

Value symmetry

- Puget's propagator
  - Construct symmetric assignment:
    - E.g. Element(X[i], [ϑ(1),..ϑ(m)], Y[i])
  - Lex ordering result
    - (X[1],..X[n]) ≤lex (Y[1],..Y[n])
  - But does not acheive GAC!

Value symmetry

- Linear time GAC propagator
  - X[1] X[2] .. X[n] ≤lex
    ϑ(X[1]) ϑ(X[2]) .. ϑ(X[n])
    B[1]=0 B[2] .. B[n] B[n+1]
  - Post C(X[i],B[i],B[i+1]) where
    - B[i]=B[i+1]=0 and X[i]=ϑ(X[i]), or
    - B[i]=0, B[i]=1 and X[i]<ϑ(X[i]), or
    - B[i]=B[i+1]=1
- Example: X[1]∈{1,2}, X[2]=2, σ(1)=2,σ(2)=1

Var and value symmetry

- Bijection σ on vars, and bijection ϑ on values that maps solutions to solutions
  - E.g. reversal of rehearsal (var symmetry) and permutation of scenes (val symmetry)
- LEX LEADER method
  - $(X[1],..X[n]) \leq_{lex} (\vartheta(X[\sigma(1)]),..\vartheta(X[\sigma(n)]))$

Var/value symmetrey

- Symmetries may act simultaneously on vars and values
  - Cannot be decomposed into bijection on vars, and bijection on values
  - E.g. in n queens problem, rotate 90˚: $X[i]=j \rightarrow X[j]=n-i+1$
- Bijection on (vars,values)
  - E.g. $\sigma(i,j)= j,n-i+1$

Var/value symmetrey

- Not all (partial) assignments map onto proper (partial) assignments
  - E.g. X[1]=1, X[2]=1 .. → X[1]=n, X[1]=n-1 ..
- LEX leader method
  - Admissible([X[1],..X[n]]) & (X[1],..X[n]) ≤lex σ(X[1],..X[n])

Lots of symmetries

- LEX LEADER method posts one constraint per symmetry
  - Can be exponential number of symmetries
  - E.g. m indistinguishable values gives m! value symmetries
- How can we deal efficiently and effectively with such situations?

Modifying search

- Avoid visiting symmetric states
  - SBDS (symmetry breaking during search)
  - SBDD (symmetry breaking by dominance detection)
  - GE-trees (group equivalence trees)

Modifying search

- Symmetry Breaking During Search
  - add a constraint at each node to rule out symmetric equivalents in the future
- Symmetry Breaking by Dominance Detection
  - check each node before entering it, to make sure you have not been to an equivalent in the past

SBDS

- Branch
  - Given assignments so far X[1]=a[1], .. , X[k-1]=a[k-1]
  - Try X[k]=b

SBDS

- Branch
  - Given assignments so far X[1]=a[1], .. , X[k-1]=a[k-1]
  - Try X[k]=b, if this fails
  - Post X[k]≠b

SBDS

- Branch
  - Given assignments so far X[1]=a[1], .. , X[k-1]=a[k-1]
  - Try X[k]=b, if this fails
  - Post X[k]≠b, and don't visit a symmetric state to the last branch

SBDS

- Branch
  - Given assignments so far X[1]=a[1], .. , X[k-1]=a[k-1]
  - Try X[k]=b, if this fails
  - Post $X[k] \neq b$, if $\sigma(X[1]=a[1])$ & ... $\sigma(X[k-1]=a[k-1])$ then $\neg\sigma(X[k]=b)$

SBDS

- E.g. reflection symmetry
- Given assignments so far X[1]=a[1], .. , X[k-1]=a[k-1]
- Try X[k]=b, if this fails
- Post X[k]≠b, if X[n]=a[1] & ... X[n-k+2]=a[k] then X[n-k+1]≠b

SBDS

- +ve
  - Does not conflict with branching heuristics
- -ve
  - Need to post symmetry breaking constraint for each symmetry
  - In general, may be exponential number of symmetries

SBDD

- Fahle, Schamberger, Sellmann, 2001
- Foccaci, Milano, 2001
  - prefigured by Brown, Finkelstein, Purdom, 1988
  - do not search a node if you have searched its equivalent before
  - check before entering a node

SBDD

- +ve
  - Does not conflict with branching heuristic
- -ve
  - Need to code dominance detection
  - Only "forward checking"
  - Can take exponential time on problems static methods solve without search

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 solution, we can uniformly swap values

Interchangeable values

- Often we have some (sub)set of values which can be freely interchanged
  - {golfer1, golfer2,...}
  - {white queen, black queen}
- If there are m values, m! symmetries
  - But we can deal with them efficiently and effectively!

Interchangeable variables

- Often we have some (sub)set of variables which can be freely interchanged
    - Queen[1]=(1,2), Queen[2]=(4,3), ..
- Easy to break this symmetry!
    - Order variables, Queen[1] < Queen[2] < ..

Interchangeable vars and values

- Sometimes we can have both interchangeable variables and values
  - Consider graph colouring
  - Node1 = red, Node2 = blue, ..
  - Suppose Node1 and Node2 have the same neighbours

Interchangeable vars and values

- Sometimes we can have both interchangeable variables and values
  - Consider pigeonhole problem
  - Hole1 = pigeon1, Hole2 = pigeon3, ..
  - Holes and pigeons all interchangeable

Row and col symmetries

- Many problems can be modelled with matrix of decision variables
  - Combinatorial problems like BIBD
  - Rows and cols can be freely permuted

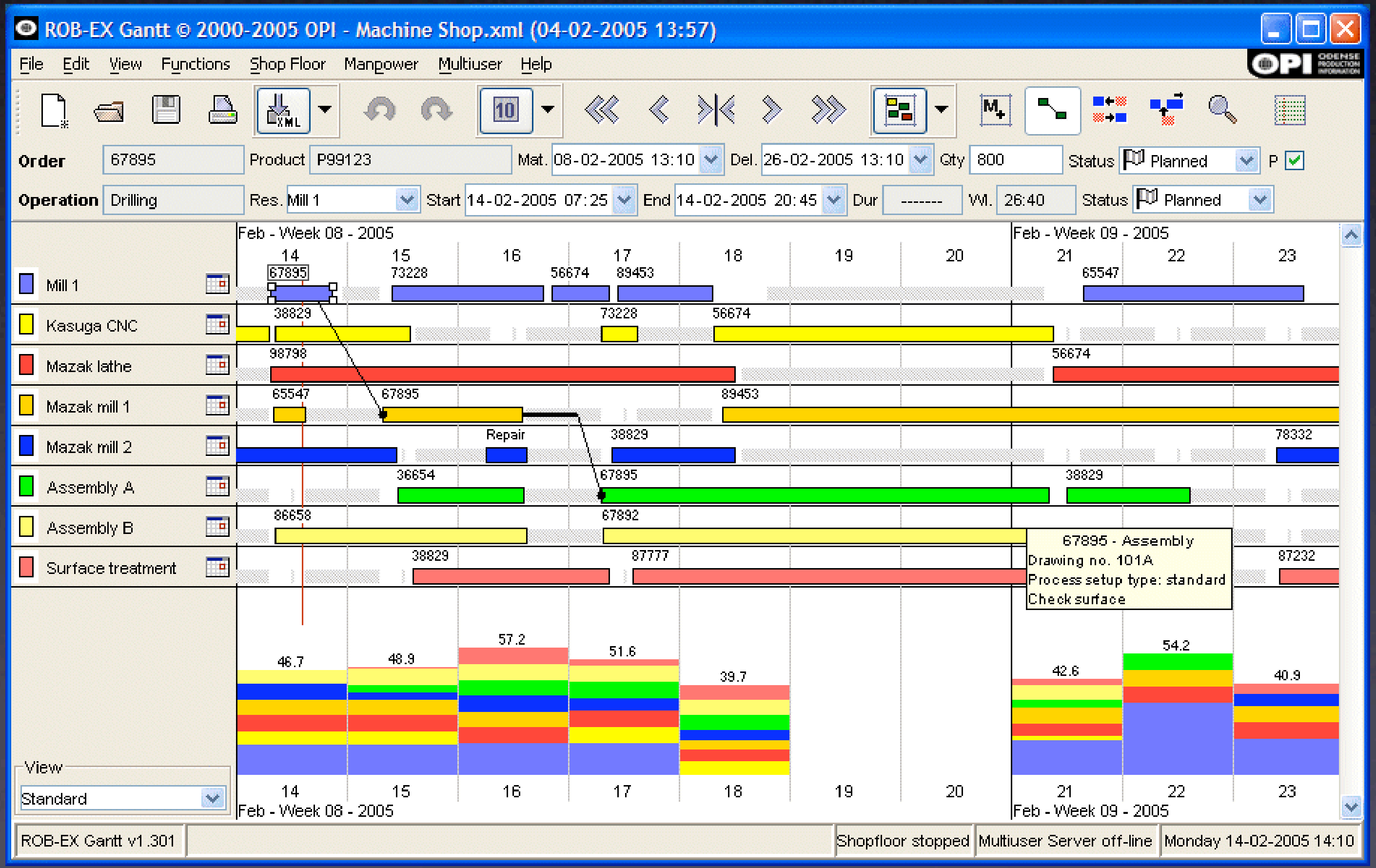
0	1	1	0	0	1	0
1	0	1	0	1	0	0
0	0	1	1	0	0	1
1	1	0	0	0	0	1
0	0	0	0	1	1	1
1	0	0	1	0	1	0
0	1	0	1	1	0	0

Row and col symmetries

- Many problems can be modelled with matrix of decision variables
  - Scheduling problems like social golfer
  - Group[i,j] are golfers playing in ith group on week j
  - Rows and cols can be freely permuted

Row and col symmetries

- Many problems can be modelled with matrix of decision variables
  - Production planning problems
  - Order[i,j,k]=1 iff order i goes on machine j in shift k
  - Rows and cols can be (partially) permuted



Row and col symmetries

- If we have a n by m matrix of decision variables
  - m!n! row and col symmetries
- However, as we shall see later, efficient and effective means to deal with this large number of symmetries
  - Again uses the LEX constraint!

Outline

- What is symmetry?
  - Bijection on assignments preserving solutions/ constraints
  - Variable and value symmetry
  - Two important special cases
    - Interchangeable values
    - Row and col symmetry

Outline

- Why is symmetry a problem?
  - Increases size of search space!
- How do we deal with symmetry?
  - Reformulate problem
  - Add constraints
    - LEX LEADER method
  - Modify search
    - SBDS, SBDD, GE-tree

Conclusions

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

Questions?