Constraint Modelling Challenge 2005

Ian Gent

Barbara Smith

July 31, 2005

Thanks

- Patrick Prosser
- Workshop Organizing Committee
  - especially Zeynep Kiziltan
- CSPLib & Toby Walsh
- Ian Miguel, Sylvain Soliman & CP Pod
- Judith Underwood
- And the Challenge Entrants

13 Entries and 24 Entrants

- Philippe Baptiste
- Nicolas Beldiceanu
- Tierry Benoist
- Mats Carlsson
- Maria Garcia de la Banda
- Peter Stuckey
- Emmanuel Hebrard
- Brahim Hnich
- Toby Walsh
- Alice Miller
- Patrick Prosser
- Chris Unsworth
- Gilles Pesant
- Steven Prestwich
- Paul Shaw
- Philippe Laborie
- Helmut Simonis
- Radoslaw Szymanek
- Mark Hennessy
- Charlotte Truchet
- Jérémie Bourdon
- Philippe Codognet
- Nic Wilson
- Karen Petrie

Advice to Future Organisers

- We hope there is a Challenge 2006 and beyond so here are some tips…
- Somehow, magically, get entrants to read the rules
- Be clear on allowing (or not) non constraints approaches
  - we view them as a plus for the challenge
  - but some participants found the name confusing
- Find someone young and energetic to check instances for validity and difficulty
  - some teething troubles with instances
  - possible 'ceiling effect' made judging hard
- Well defined results format a **must**
- Write a report
  - we hope ours gives a useful summary of approaches

The Challenge, 2005

- Announced on May 11, Closing date June 29
- Four page entry + appendices with results etc
- Small prize for best paper (not necessarily best results)
- The problem can be seen in a number of ways
  - pathwidth
  - an order processing optimisation problem
  - a mailbag sorting problem
  - …
- We presented it as the second choice
  - but notice the equivalence with pathwidth
  - why isn't this completely understood?
    - because there has never been a Challenge?

The problem

- Manufacturer has stacks of partially completed orders
- Wants to minimise the max number of stacks needed
  - given the set of orders
- Each order consists of a number of products
- Each product is made only once
- Solution is by choice of when to make each product

The problem

- Manufacturer has stacks of partially completed orders
- Wants to minimise the max number of stacks needed
  - given the set of orders
- Each order consists of a number of products
- Each product is made only once
- Solution is by choice of when to make each product

- For mailbag sorting…
  - orders are the bags of mail
  - products are the cities the letters are going to
  - stacks are the pigeonholes the mail goes into
  - want to minimise the number of holes needed
- For pathwidth …
  - products are the nodes of the graph
  - orders are the adjacency lists for each node

Example

	P_1	P_2	P_3	P_4	P_5
O_1	1	1	0	1	0
O_2	0	1	0	1	1
O_3	0	0	1	1	0
O_4	0	0	1	0	0
O_5	0	0	1	0	0

from [Simonis]

- 5 Products
- 4 Orders
- How many stacks?
  - 1 2 3 4 5
    - All 5 stacks needed!
  - 1 2 4 5 3
    - Only 3 stacks needed
    - obviously optimal

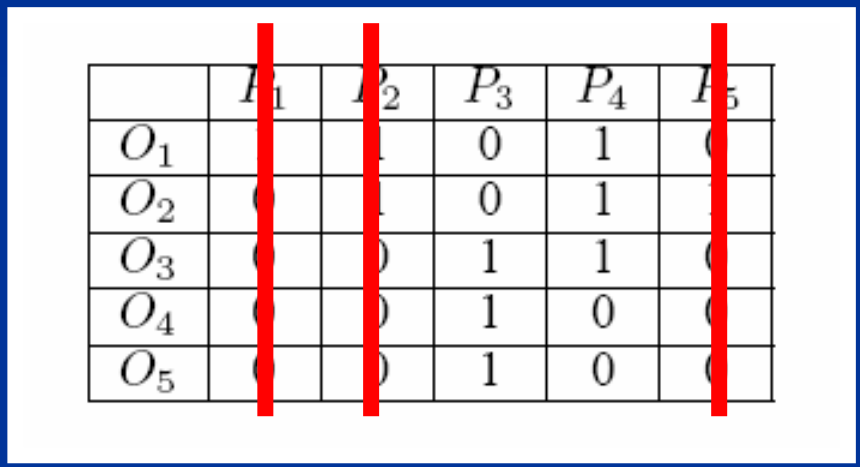
Preprocessing

	P_1	P_2	P_3	P_4	P_5
O_1	1	1	0	1	0
O_2	0	1	0	1	1
O_3	0	0	1	1	0
O_4	0	0	1	0	0
O_5	0	0	1	0	0

from [Simonis]

- orders(P1) $\subseteq$ orders(P2)
  - Put P1 next to P2
  - all the stacks necessary for P1 are needed for P2
  - P1 incurs no cost
- orders(P2) $\subseteq$ orders(P4)
- orders(P5) $\subseteq$ orders(P4)

Preprocessing



	P_1	P_2	P_3	P_4	P_5
O_1			0	1	
O_2			0	1	
O_3			1	1	
O_4			1	0	
O_5			1	0	

from [Simonis]

- Just sequence P3 / P4
- Useful for many of the Challenge instances
- Used by many entrants
- Other preprocessing steps possible but not as effective in Challenge

Lower Bounds

- Lower bounds very helpful for proving optimality
- Trivial one: max number of orders for a product
- More complicated ones abound in entries
  - [Baptiste]
  - [Garcia de la Banda, Stuckey]
  - [Miller]
  - [Pesant]
  - [Shaw, Laborie]
  - [Simonis]

Symmetry Breaking

- Order of products can be reversed without cost
  - easy to break in most models
- Some models introduce symmetry in modelling
  - greater or lesser problem depending on the model

Modelling the Open Stacks Problem

- Entrants used the following techniques…
  - Constraint Programming
    - considerable variety within this
  - Mixed Integer Programming
  - Local Search
  - Model Checking
  - Dynamic Programming
- We sketch the main techniques next

Constraint Programming (1)

- **Basic Model:**
  - a variable for each product
  - values are positions in the sequence
  - all-different constraint
  - secondary variables to count open stacks
  - objective is to minimise max number of open stacks
- Nobody used a model alone

Constraint Programming (2)

- **Dual Model**:
  - a variable for each position in the sequence
  - values are products
  - can link to basic model by channelling constraints
- [Miller, Prosser, Unsworth]
  - search from 1st to last position
  - use dynamic bounds (but expensive)
  - schedule a product next if it can be done so for free
- [Shaw, Laborie]
  - partition products into two subsets P1, P2
    - P1 will be sequenced before P2
    - each subset solved independently
    - search decisions are whether to put products into P1 or P2
- [Hebrard, Hnich, Walsh]
  - *only* the dual variables
  - special purpose global constraints for propagation

Constraint Programming (3)

- **Permuting the customers**
  - optimal permutation of the orders
    - proposed by Yanasse, EJOR 1997
    - consider the first order to be completed at time T
      - every product in that order must have been made by T
      - there is no need to schedule any other product before T
      - the max number of open stacks during the first order occurs at exactly time T
    - generalise this idea to work for subsequent customers
      - now search on **customer elimination ordering**
- [Wilson, Petrie]
  - encode this idea into CP
  - variables are positions in customer elimination ordering
  - value is customer order to be eliminated
  - initial solutions are very good (often optimal)
    - similar heuristic used by [Miller]

Constraint Programming (4)

- **Multiple viewpoints**
  - number of different models linked by channelling constraints
- [Szymanek, Hennessy]
  - main variables are for pairs of customer orders
    - 0 if the stack for one is closed before the stack for the other is opened
      - so they could share a stack potentially
    - 1 if they are both open at the same time
  - also uses a permutation of orders
  - again dominance rules
- [Shaw, Laborie] use many viewpoints

Constraint Programming (5)

- **Scheduling**
  - we are scheduling, so it's not surprising it is useful
  - view each order as a **task** requiring a **resource** (stack)
  - use **start** and **end** of each task
    - these can give useful derived constraints
- [Beldiceanu, Carlsson]
  - started from the basic model
  - use **cumulative** constraint in SICSTUS Prolog
  - order variables by decreasing number of customers requiring it
- [Shaw, Laborie]
  - some derived constraints
- [Simonis]
  - again some derived constraints

Constraint Programming (6)

- **Graph Colouring**
  - use the "co-demand" graph
    - node for each order
    - edge for orders needing the same product
  - need additional constraints for legality
  - can get derived constraints
- [Pesant]
  - based entirely on constrained graph colouring problem
  - break symmetries by finding a large clique quickly
  - from colourings, try to construct a legal ordering
- [Shaw, Laborie]
  - turn up again with some more derived constraints

Constraint Programming (7)

- **Putting products in order**
  - partial solution indicates order of products sequenced
    - but **not** their positions in the sequence
- [Simonis]
  - real valued variables used for position
    - so that any number of others can be inserted between any two
    - search tree is narrow at the root, broad at leaves
      - should help prove optimality quickly
  - choose products early needed by lots of customers
  - partial search used to find good solutions quickly

Mixed Integer Programming

- [Baptiste]
- MIP Formulation similar to Basic Model
  - with 0/1 variables instead of n-valued
  - cuts act analagously to implied constraints
- Weakness is inability to break symmetries
  - e.g. permutations not affecting number of stacks
    - "almost symmetries"

Local Search

- [Prestwich]
  - Similar model to [Baptiste] MIP
  - Increase solution density to help local search
    - each solution to original problem transformed to many in new version
    - each solution in new problem can be transformed back to original solution
- [Truchet, Bourdon, Codognet]
  - get orders with no products in common to share a stack
  - objective relaxed to be this potential instead of true value
    - in fact maximum of this can be used for true maximum
  - local search in this framework
- [Shaw, Laborie]
  - put this into the mix as well, using Large Neighbourhood Search

Model Checking

- [Miller]
- sequence with M open stacks violates a safety property
  - model checking gives a counterexample which can be translated to a solution of the stacks problem
- uses this with lower bounds to prove optimality
- some caching of visited states in Model Checker

Dynamic Programming

- [Garcia de la Banda, Stuckey]
- Consider state at time T, after some products ordered
  - open stacks are for orders involving
    - either product made at time T
    - or any product made before T & a product made after T
  - sequence of orders before/after T does not affect this
  - reduces to search of subsets (before/after T)
    - smaller search space
  - suitable for dynamic programming
- lower bounds used
- Do not use CP
  - but equivalent to CP with memoization

Conclusions on Problem

- Most successful entries were complex
- Preprocessing is vital
  - irrelevant products/customers and lower bounds
- Sequencing customers better than products
- Can divide and conquer
  - product sequence before time p does not affect optimal sequence after time p
- Key is re-using stacks
  - can only reuse if two orders have no products in common
- Local search can perform very well
- Harder benchmarks needed for this problem
  - to avoid overfitting to benchmark set

Conclusions on Challenge

- Far more successful than we expected
  - number & spread of entrants
  - variety of approaches
  - the challenge draws people in
    - thanks to Patrick again for proposing it
- More and deeper analysis than most problems
  - not dominated by the first model suggested
  - many entries of research paper quality
  - and all from May 11 to June 29, 2005
- Entrants don't know how others are doing!
  - fastest ones keep working on improvements
  - slowest ones still write good reports
- There should be another Challenge in 2006

And the runners up are …

… in alphabetical order

- Paul Shaw & Philippe Laborie
- Steven Prestwich
- Nic Wilson & Karen Petrie

And the winner is …

- Maria Garcia de la Banda & Peter Stuckey