Modelling for Constraint Programming

Barbara Smith

2. Implied Constraints, Optimization, Dominance Rules

Implied Constraints

- Implied constraints are logical consequences of the set of existing constraints
  - So are logically redundant (sometimes called redundant constraints)
- They do not change the set of solutions
- Adding implied constraints can reduce the search effort and run-time

Example: Car Sequencing

- Existing constraints only say that the option capacities cannot be exceeded
- Suppose there are 30 cars and 12 require option 1 (capacity 1/2)
- At least one car in slots 1 to 8 of the production sequence must require option 1; otherwise 12 of cars 9 to 30 will require option 1, i.e. too many
- Cars 1 to 10 must include at least two option 1 cars, … , and cars 1 to 28 must include at least 11 option 1 cars
- These are implied constraints

Useful Implied Constraints

- An implied constraint reduces search if:
  - at some point during search, a partial assignment will fail because of the implied constraint
  - without the implied constraint, the search would continue
  - the partial assignment cannot lead to a solution
    - the implied constraint forbids it, but does not change the set of solutions
- In car sequencing, partial assignments with option 1 under-used could be explored during search, without the implied constraints

Useless Implied Constraints

- The assignments forbidden by an implied constraint may never actually arise
  - depends on the search order
- e.g. in car sequencing,
  - at least one of cars 1 to 8 must require option 1
  - *any* 8 consecutive cars must have one option 1 car
  - but if the sequence is built up from slot 1, only the implied constraints on slots 1 to *k* can cause the search to backtrack
- If we find a *class* of implied constraints, maybe only some are useful
  - adding a lot of constraints that don't reduce search will increase the run-time

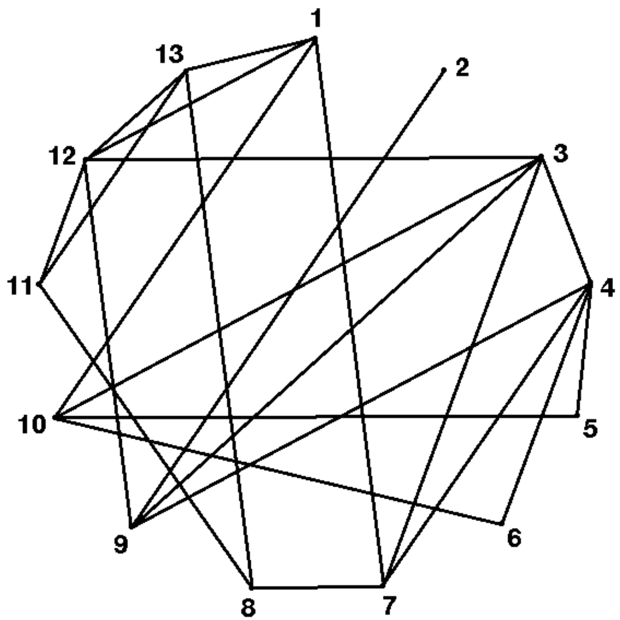
Implied Constraints v. Global Constraints

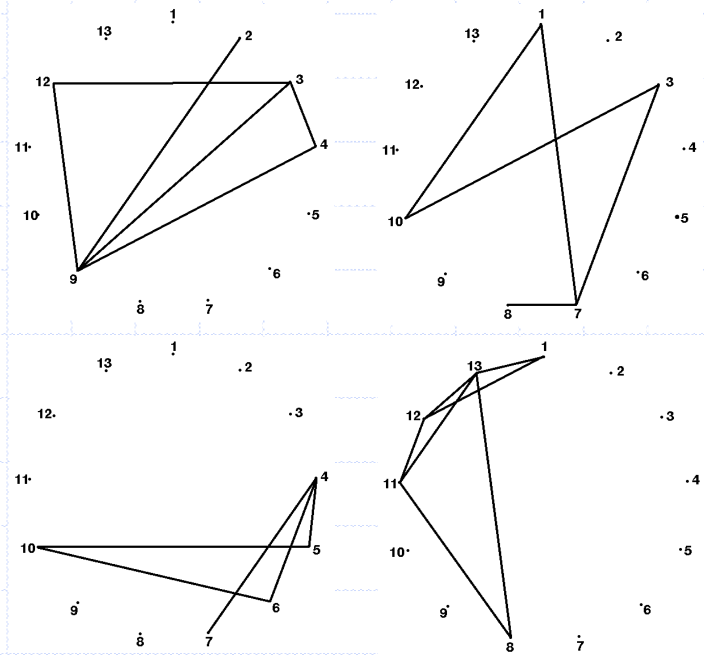
- Régin and Puget (CP97) developed a global constraint for sequence problems, including the car sequencing problem
  - “our filtering algorithm subsumes all the implied constraints” used by Dincbas et al.
- Implied constraints may only be useful because a suitable global constraint does not (yet) exist
- But many implied constraints are simple and quick to propagate
- Use a global constraint if there is one available and it is cost-effective
  - but look for useful implied constraints as well

Implied Constraints Example-Optimizing SONET Rings

- Transmission over optical fibre networks
- Known traffic demands between pairs of client nodes
- A node is installed on a SONET ring using an ADM (add-drop multiplexer)
- If there is traffic demand between 2 nodes, there must be a ring that they are both on
- Rings have capacity limits (number of ADMs, i.e. nodes, & traffic)
- Satisfy demands using the minimum number of ADMs

Simplified SONET Problem





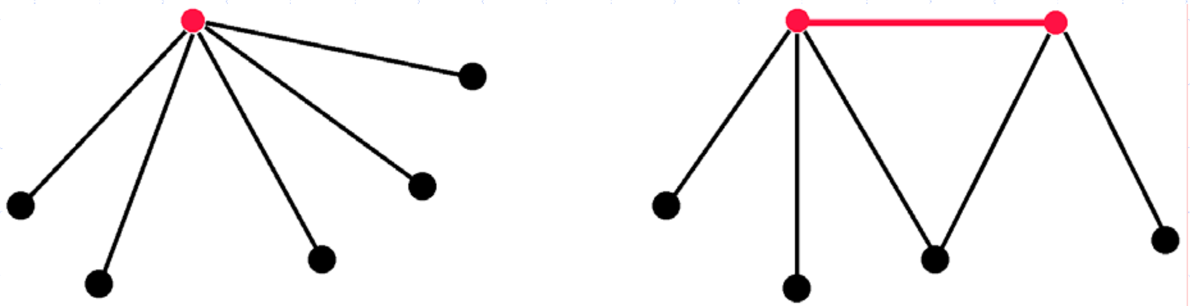
- Split the demand graph into subgraphs (SONET rings):
  - every edge is in at least one subgraph
  - a subgraph has at most 5 nodes
  - minimize total number of nodes in the subgraphs

Implied Constraints on Auxiliary Variables

- The viewpoint variables are Boolean variables, x_{ij} , such that x_{ij} = 1 if node i is on ring j
- Introduce an auxiliary variable for each node: n_i = number of rings that node i is on
- We can derive implied constraints on these variables from subproblems
  - a node and its neighbours
  - a pair of nodes and their neighbours

Implied Constraints: SONET

- A node with degree in the demand graph > 4 must be on more than 1 ring ($n_i > 1$)
- If a pair of connected nodes have more than 3 neighbours in total, at least one of the pair must be on more than 1 ring ($n_k + n_l > 2$)



Implied Constraints & Consistency

- Implied constraints can often be seen as partially enforcing some higher level of consistency
  - during search, consistency is maintained on *single* constraints
  - some forms of consistency checking take all the constraints on a subset of the variables and remove inconsistent tuples
- Enforcing consistency on subsets of the constraints is computationally expensive, even if only done before search
  - often no inconsistent tuples would be found
  - any that are found may not reduce search
  - forbidden tuples are hard to handle in constraint solvers

Implied Constraints & Nogoods

- A way to find implied constraints is to see that incorrect compound assignments are being explored
  - e.g. by examining the search in detail
- Implied constraints express & generalize what is incorrect about these assignments
- So implied constraints are like nogoods (inconsistent compound assignments)
  - whenever the search backtracks, a new nogood has been found
  - but the same compound assignment will not occur again
  - if we could learn implied constraints in this way, they do take account of the search heuristics

Finding Useful Implied Constraints

- Identify obviously wrong partial assignments that may/do occur during search
  - Try to predict them by contemplation/intuition
  - Observe the search in progress
  - Having auxiliary variables in the model enables observing/thinking about many possible aspects of the search
- Check empirically that new constraints do reduce both search and running time

Optimization

- A Constraint Satisfaction Optimization Problem (CSOP) is:
  - a CSP $\langle X,D,C \rangle$
  - and an optimization function *f* mapping every solution to a numerical value
- find the solution *T* such that the value of *f(T)* is maximized (or minimized, depending on the requirements)

Optimization: Branch and Bound

- Include a variable, say *t*, for the objective *f(T)*
- Include constraints (and maybe new variables) linking the existing variables and *t*
- Find a solution with value (say) t_0
  - Add a constraint $t < t_0$ (if minimizing)
  - Find a new solution
- Repeat last 2 steps
- When there is no solution, the last solution found has been proved optimal
  - (Or if you know a good bound on the optimal value, maybe you can recognise an optimal solution when you find it)

Optimization as a Sequence of CSPs

- Sometimes, optimization problems are solved as a sequence of decision problems
  - e.g find the matrix with the smallest number of rows that satisfies certain constraints
  - model with variables x_{ij} to represent each entry in the matrix
  - the objective is a parameter of the model, not a variable
  - so solve a sequence of CSPs with increasing matrix size until a solution is found
    - the solution is optimal

1	2	3	4	5
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Objective as a Search Variable

- If the objective is a variable, it can be a search variable
  - e.g. in the SONET problem:
    - x_{ij} = 1 if node i is on ring j
    - n_i = number of rings that node i is on
    - t (objective) = sum of n_i variables = total number of ADMs used
  - search strategy
    - assign the smallest available value to t
    - assign values to n_i variables
    - assign values to x_{ij} variables
    - backtrack to choose a larger value of t if search fails
  - the first solution found is optimal

Cork Constraint Computation Centre

Optimization: Dominance Rules

- A compound assignment that satisfies the constraints can be forbidden if it is *dominated*:
  - for any solution that this assignment would lead to, there must be another solution that is equally good or better
- Dominance rules are similar to implied constraints but
  - are not logical consequences of the constraints
  - do not necessarily preserve the set of optimal solutions

Finding Dominance Rules

- Useful dominance rules are often very simple and obvious
  - in satisfaction problems, search heuristics should guide the search away from obviously wrong compound assignments
  - in optimization problems, to prove optimality we have to prove that there is no better solution
  - every possibility allowed by the constraints has to be explored
- Examples from the SONET problem
  - no ring should have just one node on it
  - any two rings must have more than 5 nodes in total (otherwise we could merge them)

Summary

- Implied constraints can be very useful in allowing infeasible subproblems to be detected earlier
- Make sure they are useful
  - they do reduce search
  - they do reduce the run-time
  - there is no global constraint that could do the same job
- Optimization requires new solving strategies
  - usually need to find a sequence of solutions
  - to prove optimality, we often have to prove a problem unsatisfiable
  - dominance rules can help