Methods for Interactive Constraint Satisfaction

M.Sc. Thesis by

Jeppe Nejsum Madsen

`nejsum@diku.dk`

Oral Defense

March 12th, 2003, 13:15 - 15:00

All material available at

www.diku.dk/forskning/performance-engineering/jeppe

Agenda

13:15 - 14:00 Thesis Presentation

- Constraint Networks
- Interactive Constraint Satisfaction
- Fundamental Operations
- Uniform Acyclic Networks
- Conclusion

14:00 - 14:15 Break

14:15 - 15:00 Questions

Constraint Satisfaction Problems

A constraint satisfaction problem (CSP) involves the assignment of values to variables subject to a set of constraints.

- Each variable can be given a value chosen from a set of possible values called its domain.
- The constraints impose limitations on the values the variables may be assigned simultaneously.

Together, variables, domains, and constraints form a constraint network.

Constraint Networks - 1

A constraint network is a triple $\mathcal{R} = (X, D, C)$ where

1. X is a set of variables.
2. D is a map that maps each variable $x \in X$ to a finite set of values which it is allowed to take. The set $D(x)$ is called the domain of x.
3. C is a set of constraints. Let $S = \{x_k, \ldots, x_\ell\} \subseteq X$. Each constraint $C_S \in C$ is a relation with scheme S and instance C_S. The set S is the scope of the constraint, and $|S|$ denotes the arity of the constraint. Each tuple in the instance $C_S \subseteq D(x_k) \times \cdots \times D(x_\ell)$ specifies a combination of values which the constraint allows.

Constraint Networks - 2

The following notation is used:

- $n = |X|$
- $d = \max_{x \in X} |D(x)|$
- $e = |C|$

A solution to a constraint network is a n-tuple $t \in D(x_1) \times \cdots \times D(x_n)$ such that $t_{|S} \in C_S$ for all $C_S \in C$.

CSP Categories

Given a constraint network $\mathcal{R}$, a CSP can be classified into the following categories:

1. Determine whether the network $\mathcal{R}$ has a solution. This is the $\mathcal{NP}$-complete CONSTRAINT SATISFIABILITY problem.
2. Find a solution to the network $\mathcal{R}$, with no preference as to which one.
3. Find the set of all solutions, denoted $Sol(\mathcal{R})$.
4. Find an optimal solution to the network $\mathcal{R}$, where optimality is defined by a function on the variables in $\mathcal{R}$.

Solving CSPs

A solution to a constraint network is usually found using backtracking.

- Many different methods exists
- Many different instantiation orders and pruning methods have been proposed.
- All known methods have $\mathcal{O}(d^n)$ worst-case running time

Interactive Constraint Satisfaction

However, many real life applications require interactive decision support rather than automatic problem solving.

- The initial constraint network admits many possible solutions.
- Human guidance is needed to select a solution based on some additional criteria.
- These criteria cannot be modeled as constraints in the original network since they are not yet known — the user can only identify these criteria when consequences of the initial constraints are revealed.

This is interactive constraint satisfaction.

Motivating Example: Product Configuration

- Goal: From a product family with many possible variants, select a specific product that matches the user's needs.
- Involves selecting combinations of predefined components subject to a number of problem constraints.
- Not restricted to physical entities, can also be paragraphs of a legal document, financial services, actions in a plan, etc.

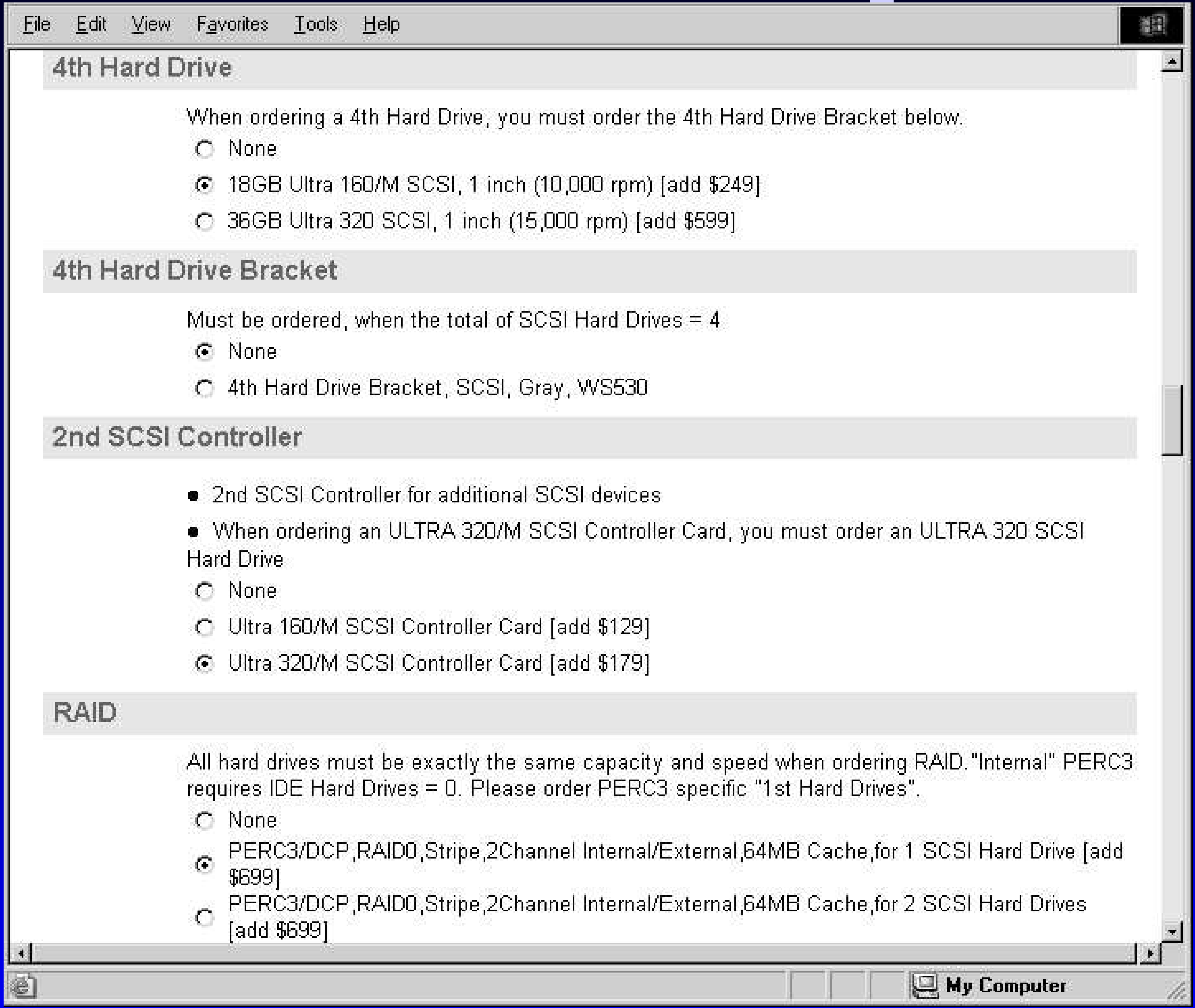
A configuration model is used to describe the components that are available as well as the relations between them.

Configuration Model

In constraint-based configuration, a configuration model is a constraint network $\mathcal{R} = (X, D, C)$.

- The variables in X list the components available for selection.
- The domains in D contain the possible choices for each component.
- The constraints in C contain the constraints between the components.

A Real-life Example



4th Hard Drive

When ordering a 4th Hard Drive, you must order the 4th Hard Drive Bracket below.

- None
- [✓ 18GB Ultra 160/M SCSI, 1 inch (10,000 rpm) [add $249]]
- 36GB Ultra 320 SCSI, 1 inch (15,000 rpm) [add $599]

4th Hard Drive Bracket

Must be ordered, when the total of SCSI Hard Drives = 4

- [✓ None]
- 4th Hard Drive Bracket, SCSI, Gray, WS530

2nd SCSI Controller

- 2nd SCSI Controller for additional SCSI devices
- When ordering an ULTRA 320/M SCSI Controller Card, you must order an ULTRA 320 SCSI Hard Drive

- None
- Ultra 160/M SCSI Controller Card [add $129]
- [✓ Ultra 320/M SCSI Controller Card [add $179]]

RAID

All hard drives must be exactly the same capacity and speed when ordering RAID. "Internal" PERC3 requires IDE Hard Drives = 0. Please order PERC3 specific "1st Hard Drives".

- None
- [✓ PERC3/DCP,RAID0,Stripe,2Channel Internal/External,64MB Cache,for 1 SCSI Hard Drive [add $699]]
- PERC3/DCP,RAID0,Stripe,2Channel Internal/External,64MB Cache,for 2 SCSI Hard Drives [add $699]

Usability Requirements

- The response time for all operations should be short.

Usability Requirements

- The response time for all operations should be short.
- The user should not be able to make selections that lead to a dead-end.

Usability Requirements

- The response time for all operations should be short.
- The user should not be able to make selections that lead to a dead-end.
- The user should be able to make a selection and later retract the selection.

Usability Requirements

- The response time for all operations should be short.
- The user should not be able to make selections that lead to a dead-end.
- The user should be able to make a selection and later retract the selection.
- The user should be able to make a “deselection”.

Usability Requirements

- The response time for all operations should be short.
- The user should not be able to make selections that lead to a dead-end.
- The user should be able to make a selection and later retract the selection.
- The user should be able to make a "deselection".
- The user should be able to make selections in arbitrary order.

Usability Requirements

- The response time for all operations should be short.
- The user should not be able to make selections that lead to a dead-end.
- The user should be able to make a selection and later retract the selection.
- The user should be able to make a “deselection”.
- The user should be able to make selections in arbitrary order.
- If a selection is made, which is inconsistent with previous selections, the user should see a list of previous selections that need to be retracted to make the new selection consistent.

Fundamental Operations

ADD-CONSTRAINT Add unary user constraint to $\mathcal{R}$ and remove domain values that can lead to a dead end.

REMOVE-CONSTRAINT Remove unary user constraint from $\mathcal{R}$ and update domain values accordingly.

RESTORATION Compute set of user constraints that must be retracted for a new user constraint C_x to be valid.

If these operations can be implemented efficiently, all usability requirements are fulfilled.

Logic Puzzle: n-queen problem

A classical problem from artificial intelligence. The goal is to place n queens $(n \geq 4)$ on a chess board of size $n \times n$ such that no queen can attack another.
A constraint network formulation:

$$\begin{aligned} X &= \{q_1, \ldots, q_n\}, & \\ D(x) &= \{1, \ldots, n\}, & \text{for } x \in X \\ |q_i - q_j| &\neq i - j, & \text{for } 1 \leq i \leq n, j > i \text{ and} \\ q_i &\neq q_j, & \text{for } 1 \leq i \leq n, j > i. \end{aligned}$$

For $n = 8$, the initial constraint network has 92 possible solutions.

Demo

Queen demo

Efficient Fundamental Operations

Divide processing in two parts:

1. Offline compilation step. Compile the initial constraint network into a form that allows efficient processing.
2. Online execution of fundamental operations on the compiled network.

For many applications, including product configuration, the online part is executed more frequently than the offline part.

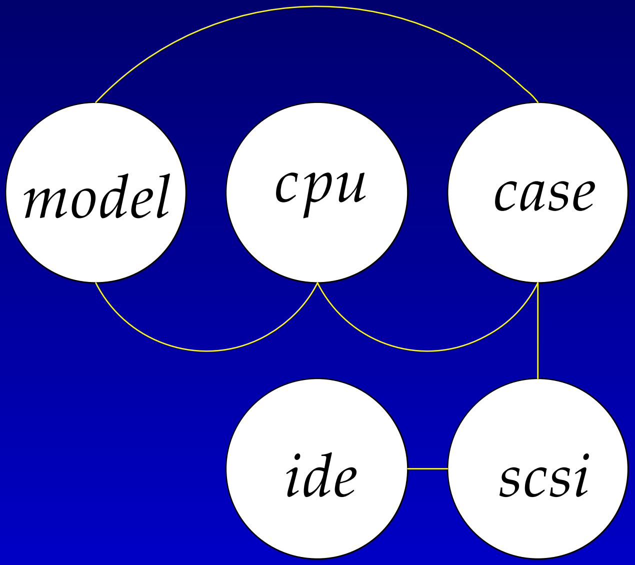
Graphical Representation

Constraint Graph Undirected graph in which each node represents a variable and there is an arc between any two variables that are related by a constraint.

Example

$X = \{model, case, ide, scsi, cpu\}$

$C = \{C_{\{model,cpu,case\}}, C_{\{ide,scsi\}}, C_{\{case,scsi\}}\}$



Arc Consistency in Binary Networks

- A variable x is arc consistent relative to y if, and only if, for every value $a \in D(x)$ there exists a value $b \in D(y)$ such that $(a, b) \in C_{xy}$.
- An arc $\{x, y\}$ in the constraint graph of $\mathcal{R}$ is arc consistent if and only if x is arc consistent relative to y and y is arc consistent relative to x.
- A constraint network is arc consistent if, and only if, all arcs are arc consistent.

Arc consistency can be achieved in time $\mathcal{O}(ed^2)$ and involves removing inconsistent domain values.

Tractable Problems

Theorem ([Freuder, 1982]). Let $\mathcal{R}$ be a binary constraint network and let (V, E) be the associated constraint graph. If (V, E) is a tree and $\mathcal{R}$ is arc consistent, then a solution to $\mathcal{R}$ can be obtained without backtracking by using a breadth first search in (V, E).

Unfortunately, in general

- networks are not binary, and
- the constraint graph is not a tree!

Transformation to Binary Network

Given a constraint network $\mathcal{R} = (X, D, C)$, the dual network is a binary constraint network $\mathcal{R}^d = (X^d, D^d, C^d)$ where

1. X^d is a set of dual variables. $X^d = \{S \mid C_S \in C\}$.
2. D^d is a function that maps each dual variable $S \in X^d$ to the dual domain $D^d(S) = \{t \mid t \in C_S\}$.
3. The constraints in C^d are

   $$C^d = \{C^d_{S,T} \mid S, T \in X^d, S \cap T \neq \emptyset\}$$

   where $C^d_{S,T}$ is the binary constraint

$$C^d_{S,T} = \{(u, v) \mid u \in C_S, v \in C_T, u_{|S \cap T} = v_{|S \cap T}\}.$$

Dual Network Properties

- There is a dual variable for each original constraint.
- The domain of a dual variable is the set of tuples from the corresponding original constraint.
- A solution to the dual problem can be mapped back to a solution to the original problem.

If we can make the constraint graph of the dual problem into a tree, the original problem becomes tractable !

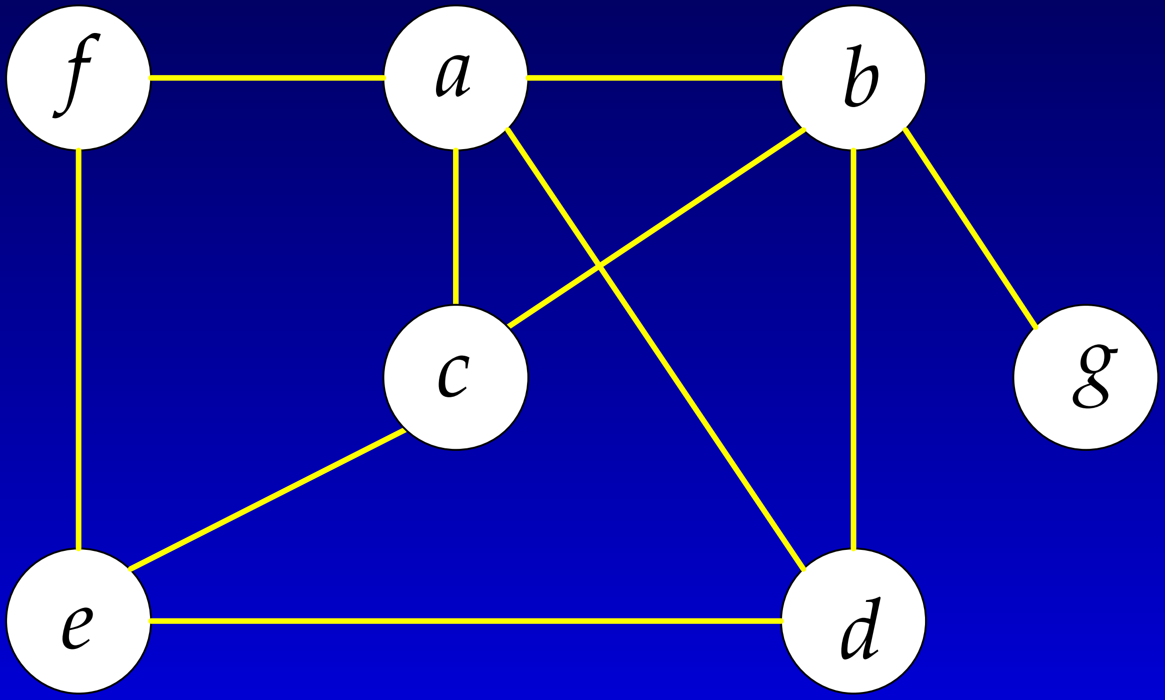
Tree Clustering

A systematic decomposition method for transforming the dual network into a tree [Dechter and Pearl, 1989]. Based on results for acyclic hypergraphs originating in relational database theory.

- Triangulate constraint graph by adding redundant constraints. Finding a minimal triangulation is $\mathcal{NP}$-complete, so a heuristic is used.
- Identify maximal cliques.
- For each clique, synthesize a new constraint for the clique's variables. Exponential in the size of the largest clique.
- Find a join tree as the maximum spanning tree of the resulting dual graph.

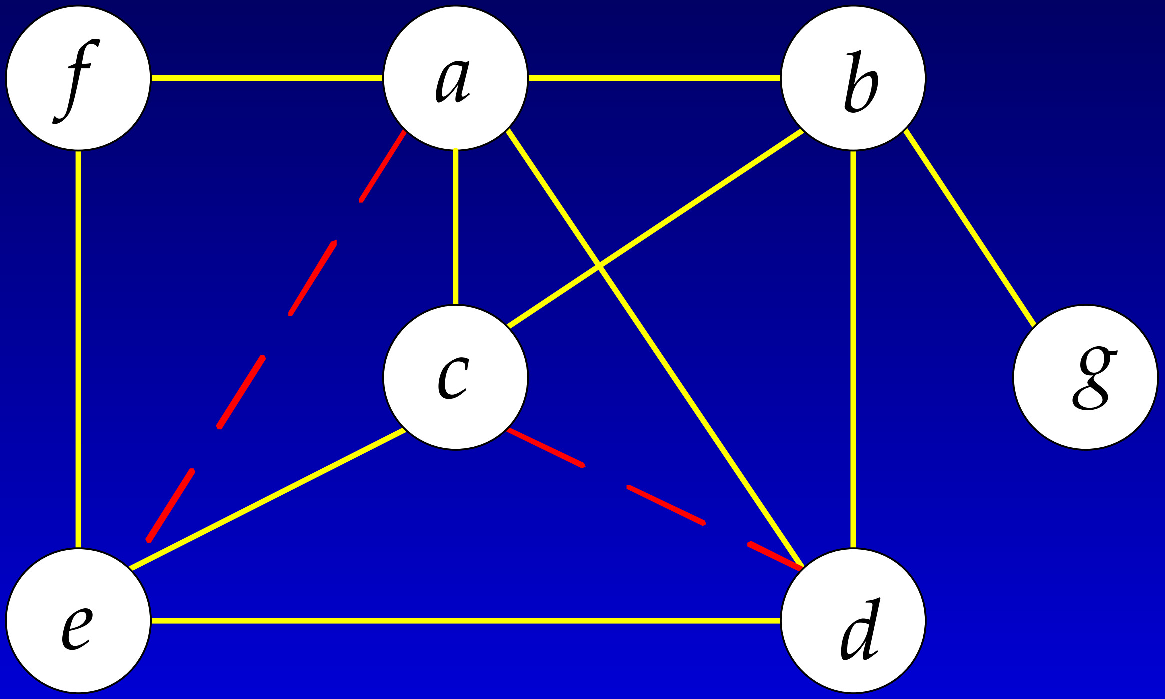
Tree Clustering Example

Input: Initial constraint graph.



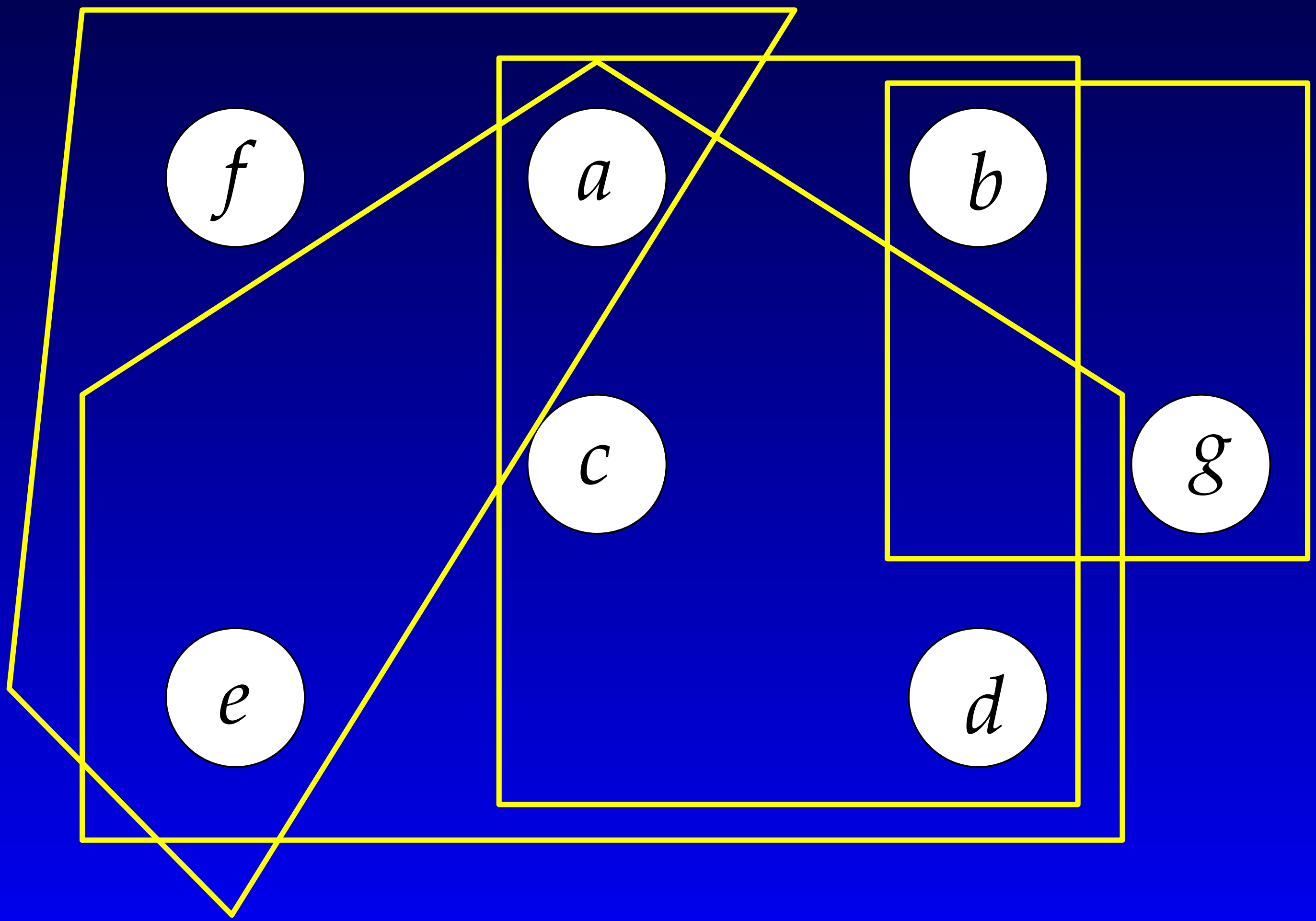
Tree Clustering Example

Step 1: Triangulate constraint graph.



Tree Clustering Example

Step 2: Identify maximal cliques and solve subproblems.

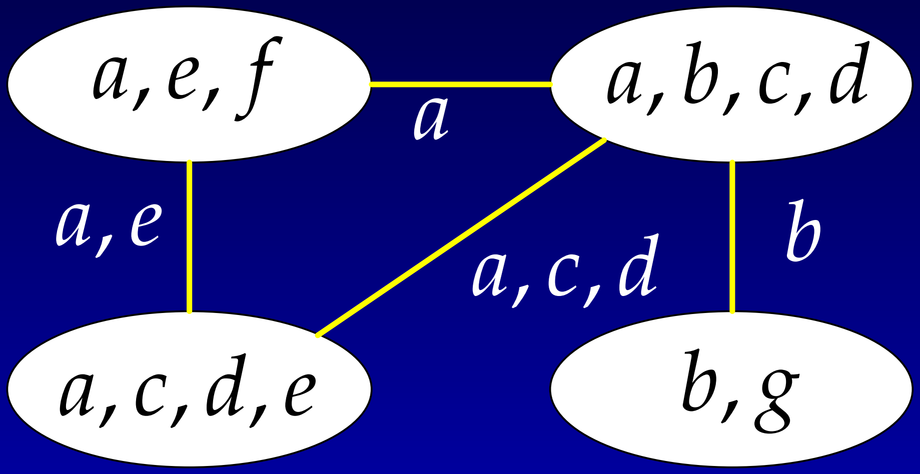


Tree Clustering Example

Step 3: Create constraint for each maximal clique.

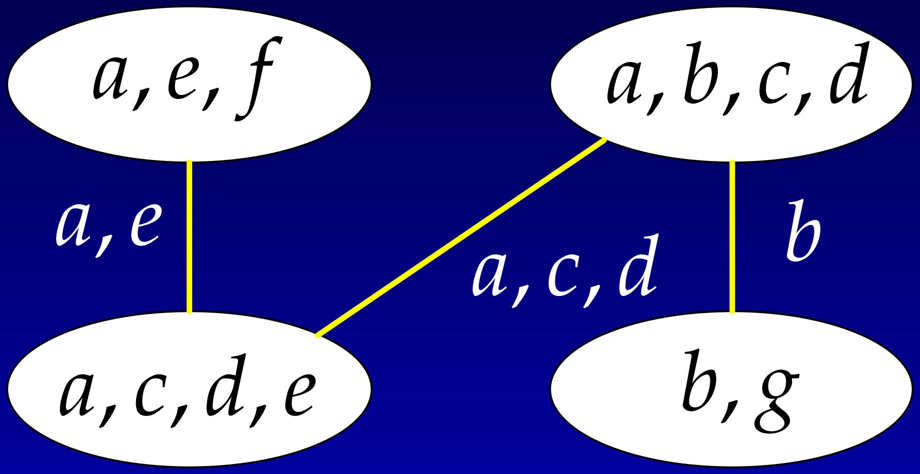
Tree Clustering Example

Step 3: Resulting dual network.



Tree Clustering Example

Step 4: Find join tree as maximum spanning tree.



Cartesian Product Representation

The maximum number of tuples in a constraint is exponential in the number of variables in the clique. For some test instances, more than 10^{12} tuples are generated.

Idea: Use Cartesian products to generate the set of tuples.

Example: Given a relation with tuples

$$\{(0,0,0),(0,0,1),(0,1,1),(1,0,1),(1,1,1)\}$$

we can generate the tuples using Cartesian products:

$$\{(0,0,0)\} \cup (\{0,1\} \times \{0,1\} \times \{1\})$$

Properties of the CPR

- Saves space. In all test instances, the number of tuples was reduced significantly. The 10^{12} tuples was reduced to 478.
- Preserves the relational operations select, project and join

A compression heuristic was presented in [Katajainen and Madsen, 2002]. Join algorithms for compressed relations can be found in [Madsen, 2002].

Experimental Results

Algorithms for tree clustering and the fundamental operations have been implemented and evaluated on a large number of constraint networks.

- Tree clustering succeeds on all real-life instances.
- The fundamental operations are too slow for all but the smallest networks.

The inefficiency of the fundamental operations is due to the fact that the running time of arc consistency is quadratic in the number of uncompressed tuples.

Improving the Performance

A uniform acyclic network is an acyclic network where

$$|S \cap T| \leq 1, \text{ for all } C_S, C_T \in C$$

In a uniform acyclic network, arc consistency can be maintained in time quadratic in the number of compressed tuples.

A uniform acyclic network can be constructed from an acyclic network by intelligently applying the split operator.

The Split Operator

Original constraint.

a	b	c	d
0	0	0	1
0	0	1	$\{0,1\}$
1	1	0	1
1	1	1	$\{0,1\}$

The Split Operator

After split. Original constraint replaced by two new constraints and a new meta variable λ is added.

a	b	λ
0	0	0
0	0	1
1	1	2
1	1	3

c	d	λ
0	1	0
1	$\{0,1\}$	1
0	1	2
1	$\{0,1\}$	3

The Split Operator

After compression.

a	b	λ
0	0	$\{0,1\}$
1	1	$\{2,3\}$

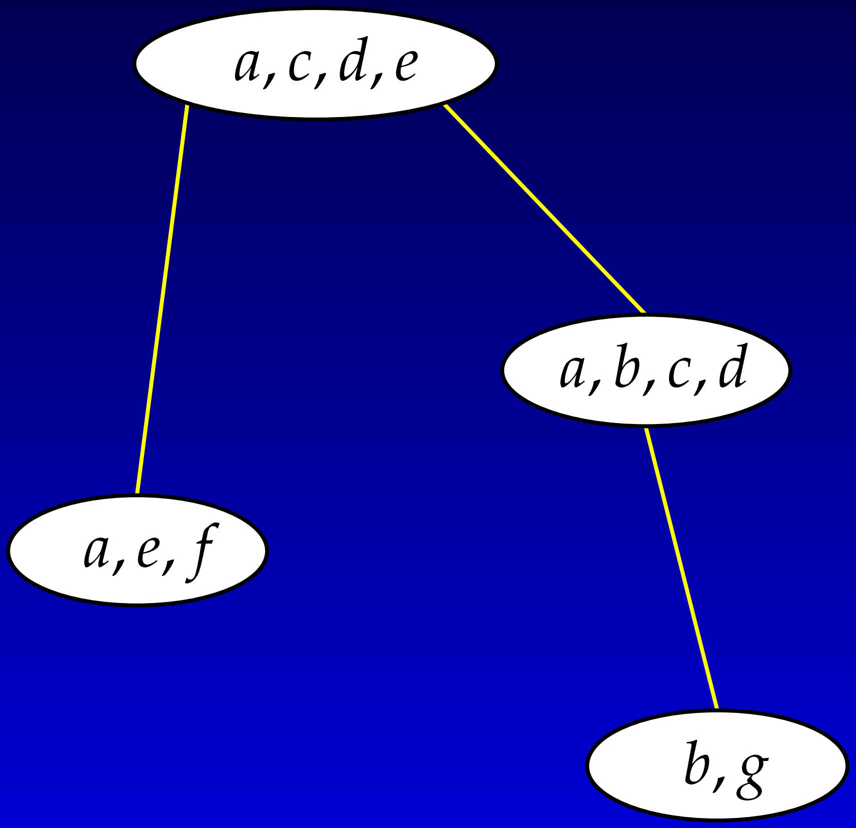
c	d	λ
0	1	$\{0,2\}$
1	$\{0,1\}$	$\{1,3\}$

Uniform Acyclic Network Construction

- Process each edge of the join tree in a bottom up fashion.
- For each edge $\{C_S, C_T\}$ split the constraints C_S and C_T into $C_{S'}, C_{S''}, C_{T'}, C_{T''}$.
- Join $C_{S''}$ and $C_{T''}$.
- Replace C_S and C_T with the three new constraints which are now linked with a single variable.

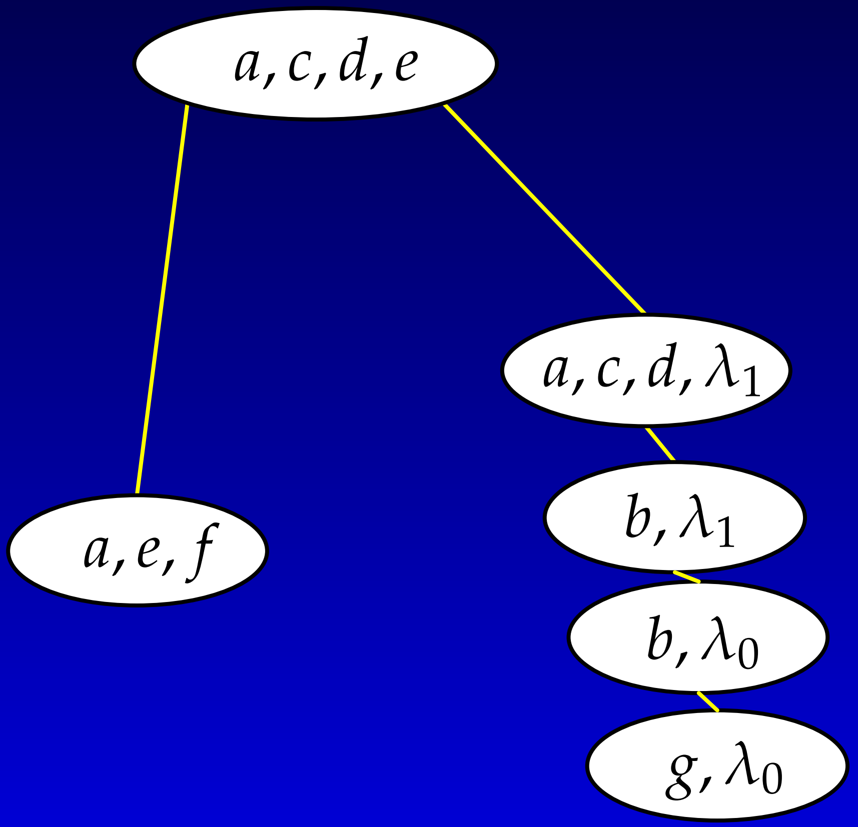
Example

Dual Network generated by tree clustering.



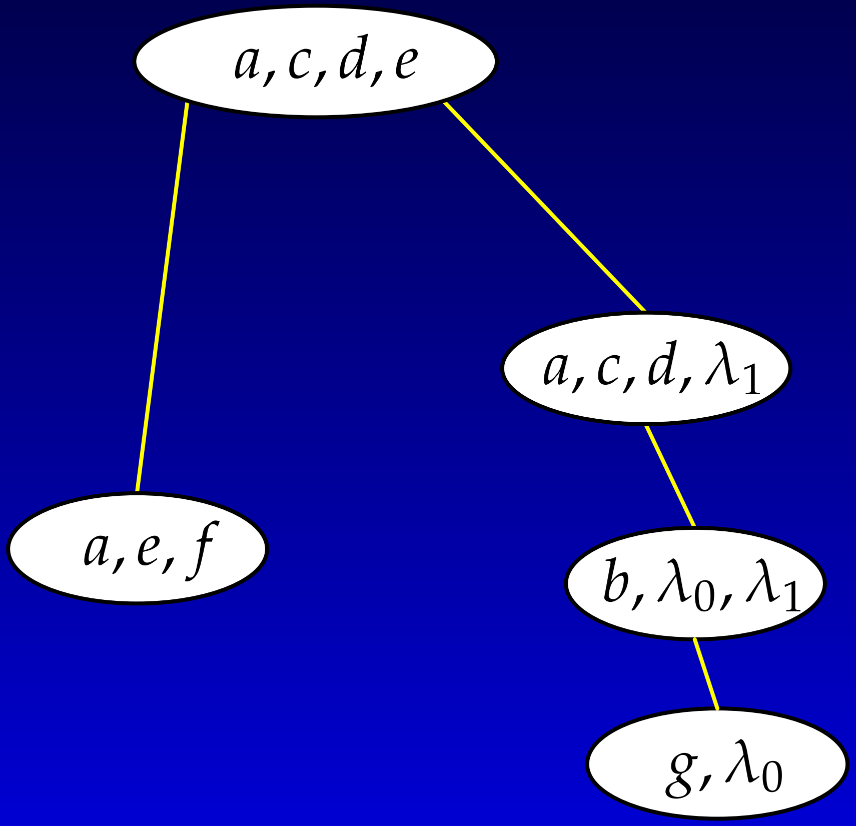
Example

Process edge. Split the two constraints.



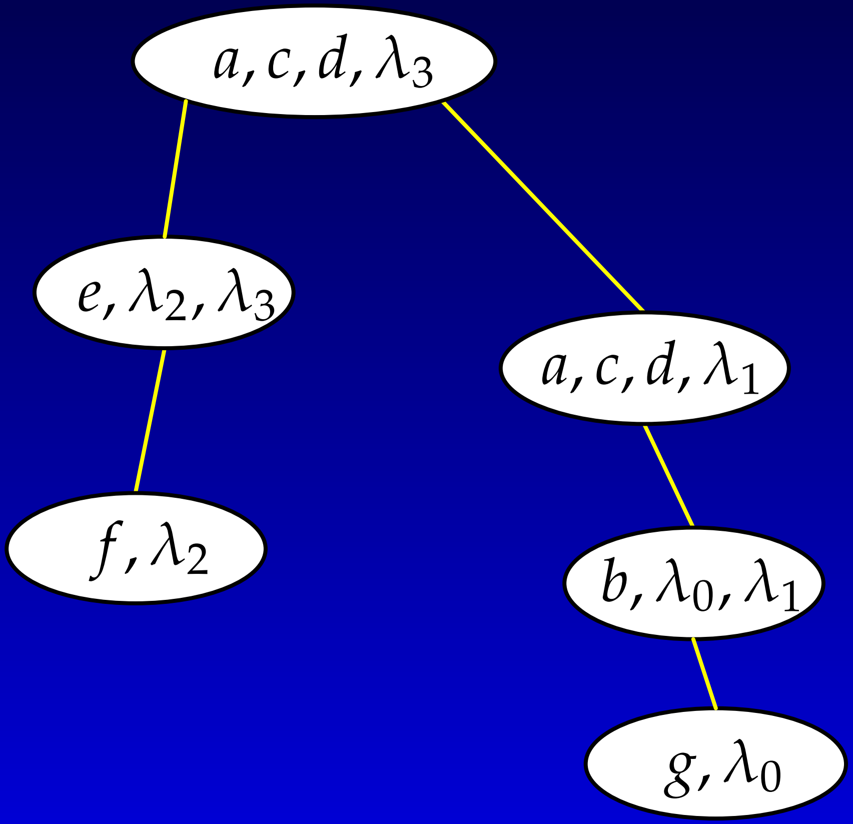
Example

Combine two of the new constraints.



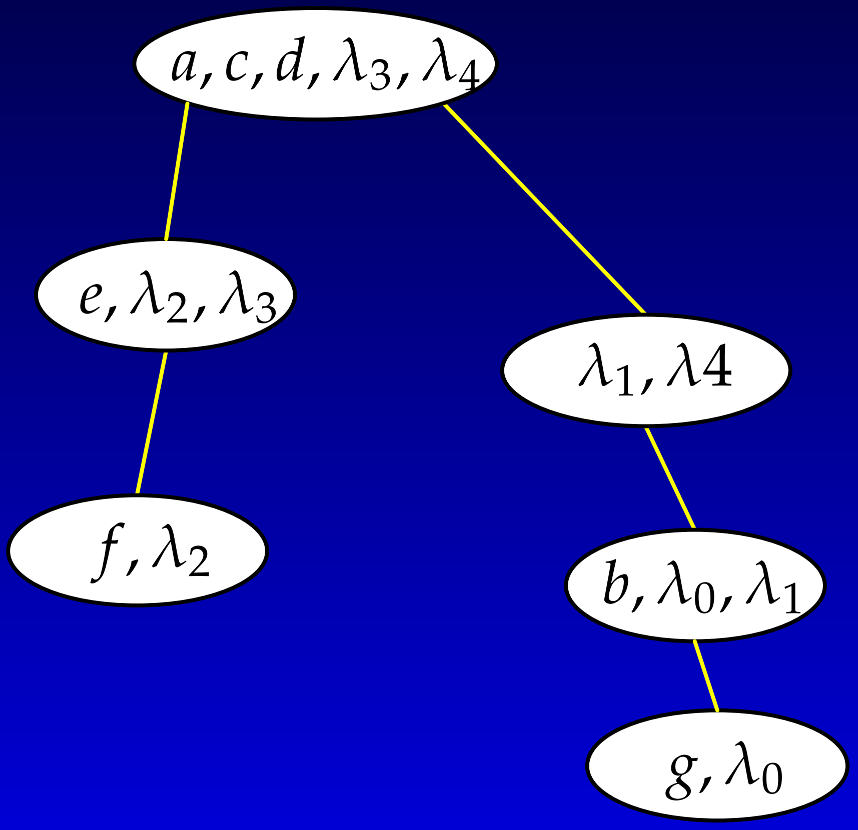
Example

Process next edge.



Example

Process final edge.


$a, c, d, \lambda_3, \lambda_4$
e, λ_2, λ_3
$\lambda_1, \lambda 4$
f, λ_2
b, λ_0, λ_1
g, λ_0

Experimental Results

- For some networks, a uniform acyclic network could not be constructed due to memory usage.
- For all the uniform acyclic networks constructed, the worst- case response time for the fundamental operations was less than 2 seconds and the average less than 0.1 seconds.

Summary of Results

- Fundamental operations for an interactive constraint solver have been identified and described formally.
- Tree clustering is viewed as one of the standard solutions for constraint network compilation. Experimental results show that, while being polynomial, it is not fast enough for interactive use.
- A new method based on uniform acyclic networks have been proposed.
- Uniform acyclic networks enables response times suitable for interactive use.
- It is not always feasible to construct a uniform acyclic network.

Break

References

R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence* **38**(3): 353–366, 1989.

E. C. Freuder. A sufficient condition for backtrack-free search. *Journal of the ACM* **29**(1):24–32, 1982.

J. Katajainen and J. N. Madsen. Performance tuning an algorithm for compressing relational tables. In *Proceedings of the 8th Scandinavian Workshop on Algorithm Theory*, volume 2368 of *Lecture Notes in Computer Science*, pages 398–407. Springer-Verlag, 2002.

J. N. Madsen. Algorithms for compressing and joining