Soft constraints: Algorithms (2)

T. Schiex

INRA - Toulouse, France

Inference

In classical CSP, inference produces new constraints which are implied by the problem. Makes implicit c explicit.

$(X, D, C) \to c$ s.t. c satisfied by all solutions.

Then $(X, D, C \cup \{c\})$ is equivalent to (X, D, C) (same solutions). More explicit. Simpler to solve.

Complete inference: transform (X, D, C) in to an equivalent problem where all forbidden combinations are explicit (or all solutions explicit, or (in)satisfiability proven).

Soft constraints

$P = \langle X, D, C, S \rangle$ describes a distribution of valuations on the search space (combination of all constraints).

We say that c_s is implied by P iff

$$\forall t_X, c_S(t_X[S]) \preccurlyeq_v \bigoplus_{c_S \in C} c_S(t[S])$$

Makes explicit the fact that the violation level of some tuples is, at lest, equal to $c_S(t)$. Explicit lb.

Adding c_s to P may change the distribution of valuations unless... $\oplus$ idempotent.

$\Rightarrow$ replace constraints by new maximally explicit constraints, preserving optimal cost.

Combination and projection in VCSP

- Combination: $c_S \bowtie c'_{S'}$ is a constraint on $S \cup S'$ s.t.:

$$c_S \bowtie c'_{S'}(t) = c(t[S]) \oplus c'(t[S'])$$

- Projection: $V \subset X, c_S[V]$ is a constraint on $V \cap S$ s.t.:

$$c_S[V](t) = \min_{t'[V]=t} c_S[t']$$

total order: best extension of t_V to S.

$K \subset C, L = \cup c_S \in KS$. Then $\bowtie_{c \in K} c[V]$ produces maximally informative induced constraints on $V \cap L$: there is a t on L such that $(\bowtie_{c \in K} c)[t] = (\bowtie_{c \in K} c[V])[t]$.

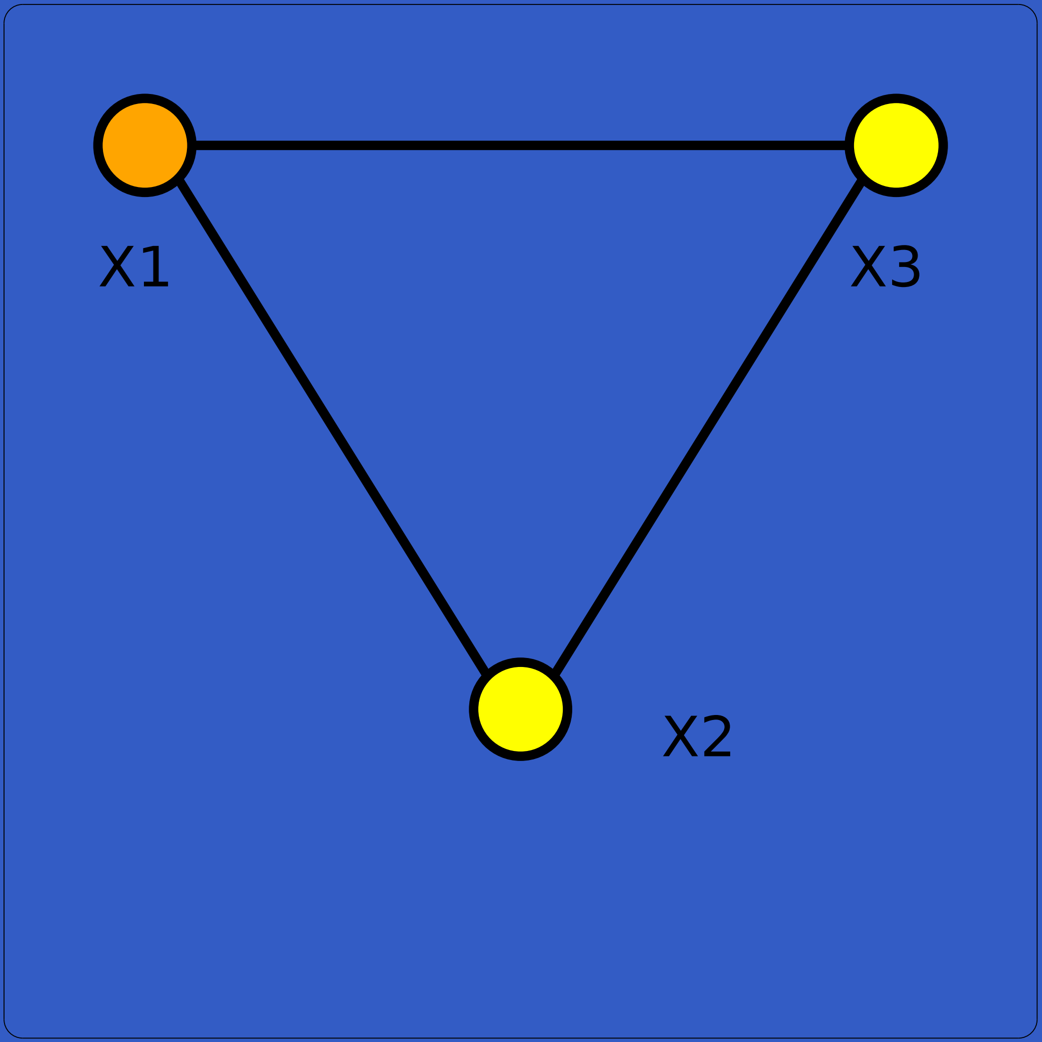
Variable elimination (Bertele & Brioshi 1972

No tree search. Directly synthetizes all optimal solutions.

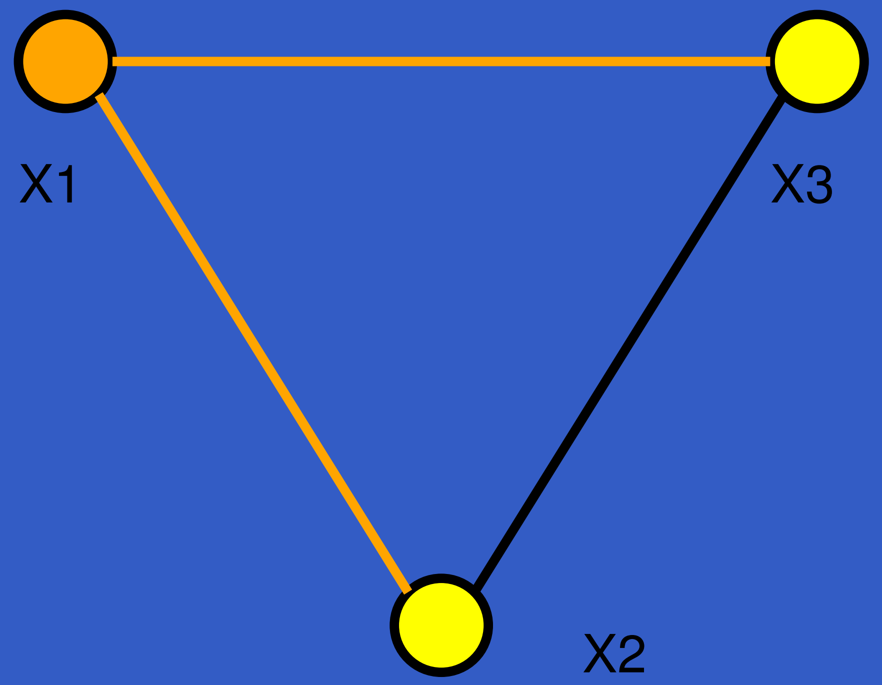
- Consider variable $x \in X$, $K_x = \{c_S \in C, x \in S\}$, $L = (\cup_{c_S \in K_x}(S)) - \{x\}$.
- Compute the combination $\bowtie K_x$ of all constraints in K_x
- Compute the optimal possible costs $(\bowtie K_x)[L]$ induced on L by K_x.
- Remove K_x from C and replace it by $(\bowtie K_x)[L]$.

Same optimal cost.

3x3 queens



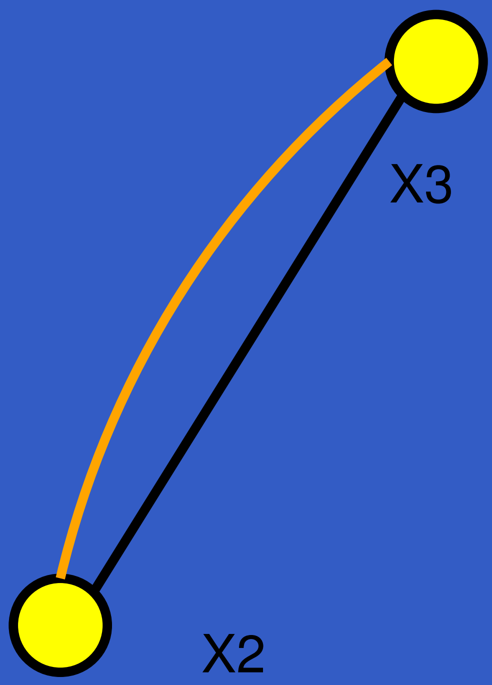
3x3 queens



$K_x = \{C12, C13\}$

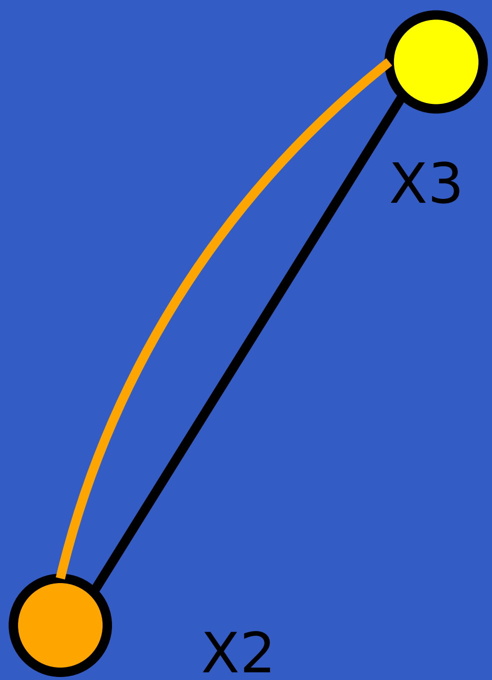
$L = \{X2, X3\}$

3x3 queens



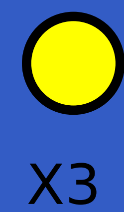
K_x={C12,C13}
L={X2,X3}

3x3 queens



$K_x=\{C23\}$
$L=\{X3\}$

3x3 queens



$K_x = \{C23\}$

$L = \{X3\}$

Constraints management

X1	X2	V
1	1	1
2	1	1
3	1	0
1	2	1
2	2	1
3	2	1
1	3	0
2	3	1
3	3	1

X1	X3	V
1	1	1
2	1	0
3	1	1
1	2	0
2	2	1
3	2	0
1	3	1
2	3	0
3	3	1

X1	X2	X3	V
1	1	1	2
2	1	1	1
3	1	1	1
1	2	1	2
2	2	1	1
3	2	1	2
1	3	1	1
2	3	1	1
3	3	1	2

X1	X2	X3	V
1	1	2	1
2	1	2	2
3	1	2	0
1	2	2	1
2	2	2	2
3	2	2	1
1	3	2	0
2	3	2	2
3	3	2	1

X1	X2	X3	V
1	1	3	2
2	1	3	1
3	1	3	1
1	2	3	2
2	2	3	1
3	2	3	2
1	3	3	1
2	3	3	1
3	3	3	2

Constraints management

X1	X2	V
1	1	1
2	1	1
3	1	0
1	2	1
2	2	1
3	2	1
1	3	0
2	3	1
3	3	1

X1	X3	V
1	1	1
2	1	0
3	1	1
1	2	0
2	2	1
3	2	0
1	3	1
2	3	0
3	3	1

X1	X2	X3	V
1	1	1	2
2	1	1	1
3	1	1	1
1	2	1	2
2	2	1	1
3	2	1	2
1	3	1	1
2	3	1	1
3	3	1	2

X1	X2	X3	V
1	1	2	1
2	1	2	2
3	1	2	0
1	2	2	1
2	2	2	2
3	2	2	1
1	3	2	0
2	3	2	2
3	3	2	1

X1	X2	X3	V
1	1	3	2
2	1	3	1
3	1	3	1
1	2	3	2
2	2	3	1
3	2	3	2
1	3	3	1
2	3	3	1
3	3	3	2

Constraints management

X1	X2	V
1	1	1
2	1	1
3	1	0
1	2	1
2	2	1
3	2	1
1	3	0
2	3	1
3	3	1

X1	X3	V
1	1	1
2	1	0
3	1	1
1	2	0
2	2	1
3	2	0
1	3	1
2	3	0
3	3	1

X1	X2	X3	V
1	1	1	2
2	1	1	1
3	1	1	1
1	2	1	2
2	2	1	1
3	2	1	2
1	3	1	1
2	3	1	1
3	3	1	2

X1	X2	X3	V
1	1	2	1
2	1	2	2
3	1	2	0
1	2	2	1
2	2	2	2
3	2	2	1
1	3	2	0
2	3	2	2
3	3	2	1

X1	X2	X3	V
1	1	3	2
2	1	3	1
3	1	3	1
1	2	3	2
2	2	3	1
3	2	3	2
1	3	3	1
2	3	3	1
3	3	3	2

X2	X3	V
1	1	1
2	1	1
3	1	1
1	2	0
2	2	1
3	2	0
1	3	1
2	3	1
3	3	1

Constraints management

X1	X2	V
1	1	1
2	1	1
3	1	0
1	2	1
2	2	1
3	2	1
1	3	0
2	3	1
3	3	1

X1	X3	V
1	1	1
2	1	0
3	1	1
1	2	0
2	2	1
3	2	0
1	3	1
2	3	0
3	3	1

X1	X2	X3	V
1	1	1	2
2	1	1	1
3	1	1	1
1	2	1	2
2	2	1	1
3	2	1	2
1	3	1	1
2	3	1	1
3	3	1	2

X1	X2	X3	V
1	1	2	1
2	1	2	2
3	1	2	0
1	2	2	1
2	2	2	2
3	2	2	1
1	3	2	0
2	3	2	2
3	3	2	1

X1	X2	X3	V
1	1	3	2
2	1	3	1
3	1	3	1
1	2	3	2
2	2	3	1
3	2	3	2
1	3	3	1
2	3	3	1
3	3	3	2

X2	X3	V
1	1	1
2	1	1
3	1	1
1	2	0
2	2	1
3	2	0
1	3	1
2	3	1
3	3	1

X2	X3	V
1	1	1
2	1	1
3	1	0
1	2	1
2	2	1
3	2	1
1	3	0
2	3	1
3	3	1

X2	X3	V
1	1	2
2	1	2
3	1	1
1	2	1
2	2	2
3	2	1
1	3	1
2	3	2
3	3	2

Constraints management

X1	X2	V
1	1	1
2	1	1
3	1	0
1	2	1
2	2	1
3	2	1
1	3	0
2	3	1
3	3	1

X1	X3	V
1	1	1
2	1	0
3	1	1
1	2	0
2	2	1
3	2	0
1	3	1
2	3	0
3	3	1

X1	X2	X3	V
1	1	1	2
2	1	1	1
3	1	1	1
1	2	1	2
2	2	1	1
3	2	1	2
1	3	1	1
2	3	1	1
3	3	1	2

X1	X2	X3	V
1	1	2	1
2	1	2	2
3	1	2	0
1	2	2	1
2	2	2	2
3	2	2	1
1	3	2	0
2	3	2	2
3	3	2	1

X1	X2	X3	V
1	1	3	2
2	1	3	1
3	1	3	1
1	2	3	2
2	2	3	1
3	2	3	2
1	3	3	1
2	3	3	1
3	3	3	2

X2	X3	V
1	1	1
2	1	1
3	1	1
1	2	0
2	2	1
3	2	0
1	3	1
2	3	1
3	3	1

X2	X3	V
1	1	1
2	1	1
3	1	0
1	2	1
2	2	1
3	2	1
1	3	0
2	3	1
3	3	1

X2	X3	V
1	1	2
2	1	2
3	1	1
1	2	1
2	2	2
3	2	1
1	3	1
2	3	2
3	3	2

Constraints management

X1	X2	V
1	1	1
2	1	1
3	1	0
1	2	1
2	2	1
3	2	1
1	3	0
2	3	1
3	3	1

X1	X3	V
1	1	1
2	1	0
3	1	1
1	2	0
2	2	1
3	2	0
1	3	1
2	3	0
3	3	1

X1	X2	X3	V
1	1	1	2
2	1	1	1
3	1	1	1
1	2	1	2
2	2	1	1
3	2	1	2
1	3	1	1
2	3	1	1
3	3	1	2

X1	X2	X3	V
1	1	2	1
2	1	2	2
3	1	2	0
1	2	2	1
2	2	2	2
3	2	2	1
1	3	2	0
2	3	2	2
3	3	2	1

X1	X2	X3	V
1	1	3	2
2	1	3	1
3	1	3	1
1	2	3	2
2	2	3	1
3	2	3	2
1	3	3	1
2	3	3	1
3	3	3	2

X2	X3	V
1	1	1
2	1	1
3	1	1
1	2	0
2	2	1
3	2	0
1	3	1
2	3	1
3	3	1

X2	X3	V
1	1	1
2	1	1
3	1	0
1	2	1
2	2	1
3	2	1
1	3	0
2	3	1
3	3	1

X2	X3	V
1	1	2
2	1	2
3	1	1
1	2	1
2	2	2
3	2	1
1	3	1
2	3	2
3	3	2

Bucket elimination (Dechter 1997)

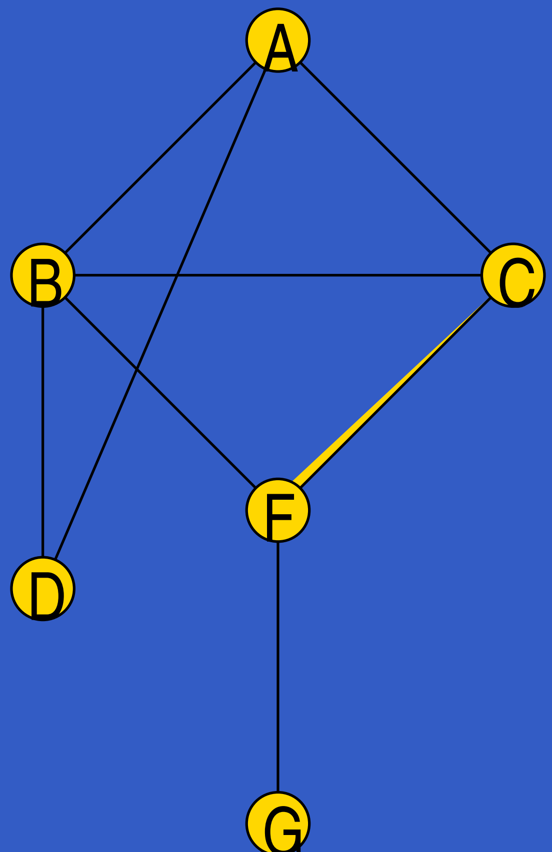
- fix a variable ordering $x_1, \ldots, x_n$
- one bucket per variable, from last to first: contains all constraints involving the variable (not already in a bucket).
- process from last to first:
  1. join all constraints K in the bucket
  2. eliminate the current variable by projecting on L
  3. put the projection in the first bucket that contain one variable of L.

Complexity

- Time complexity: dominated by the time to compute the largest $\bowtie K$. Exponential in $|L| + 1$ for the largest L.
- Space complexity: dominated by the space to store the largest projection $(\bowtie K)[L]$. Exponential in $|L|$ for the same L.

Can we influence this maximum $|L|$? Order of elimination...

Let see on a more complex graph…



Ex: Order A, C, B, F, D, G. Maximum $|L| = ?$.
Ex: Order A, F, D, C, B, G. Maximum $|L| = ?$.

Width

For a graph $G = (V, E)$, an order of vertices d:

- width of a vertex: number of connected predecessors (parents)
- width of ordered graph: maximum width of a vertex
- width of graph: min. width over all ordering

Ex: on previous graph using order A, C, B, F, D, G and A, F, D, C, B, G.

Induced graph

Mimic elimination: when we eliminate we induce a new constraint $\Rightarrow$ new edges.

Processing vertices from the last to the first, connect all the parents of a vertex together (set L)..

The width on an induced graph is equal to the set of the largest L we will deal during elimination (k-tree number, max-clique size-1, tree width. . .)..

Finding a min-induced width ordering is NP-hard.

A structural pol. time class

The induced-width of a tree is 1 (all vertex have one parent under a topological ordering of vertices)..

All tree-structured problems can be solved in polynomial time.

Understandable caracterisation of graph with induced width k:

They are partial k-trees. A k-tree being inductively defined as a k-clique, or by the addition of a new vertex to a k-tree, connecting it to all vertices of a k-clique in it.

Block by block elimination (Bertelé Brioshi,

- when we compute $\bowtie K$, we actually solve a subproblem defined on $L \cup \{x\}$ almost completely, ignoring other constraints.
- constraints connecting variables in L will still be handled later (they are now in a clique).

Could we do all this in one step ?

Block by block elimination

Consider a set of variables $V \subset X$.

- $S(V)$ the set of all variables not in V and connected to V (the separator).
- $K(V)$ the set of constraints with scope included in $V \cup S(V)$.

1. compute $\bowtie K(V)$ and project on $S(V)$.
2. replace $K(V)$ by $(\bowtie K(V))[S(V)]$
3. forget (eliminate) V.

Block by block elimination

Better than variable elimination (compare on k-trees).

- Time: exponential in the largest $V \cup S(V)$ used.
- Space: exponential in $S(V)$. We can join and project at the same time.

The $V \cup S(V)$ can be the same as the maximal $L \cup \{x\}$ of VE.

In fact, this is the best way to proceed if we want to minimize $|V \cup S(V)$ (induced width+1). But we could prefer to minize just $S(V)$...

BBE and VE

Reexploited many times. . .
(Beeri et al 1983) Databases, (Lauritzen and Spiegelhalter 1988) Bayesian Nets,(Dechter and Pearl 1989) Bayesian nets and CSP, (Shenoy and Shafer 1990) Generic,(Bistarelli Rossi 1995) SCSP. . .

Cycle cutset (Dechter 90)

We know that tree-structured problem are easy to solve by VE.

From the graph of (X, C, D, S) identify a set of variables whose removal makes the graph a tree: cycle-cutset.

For all possible combination of values of all the variables in the cutset:

- assign the variable in the cutset (makes elim. easy).
- solve by VE (linear time)

Overall exponential in the cutset-size. NP-hard to minimize.

More general: boosting search by VE

Usually using VE or BBE is to expensive in time/space. Can it be used to perform some inference that can help Branch and Bound search?

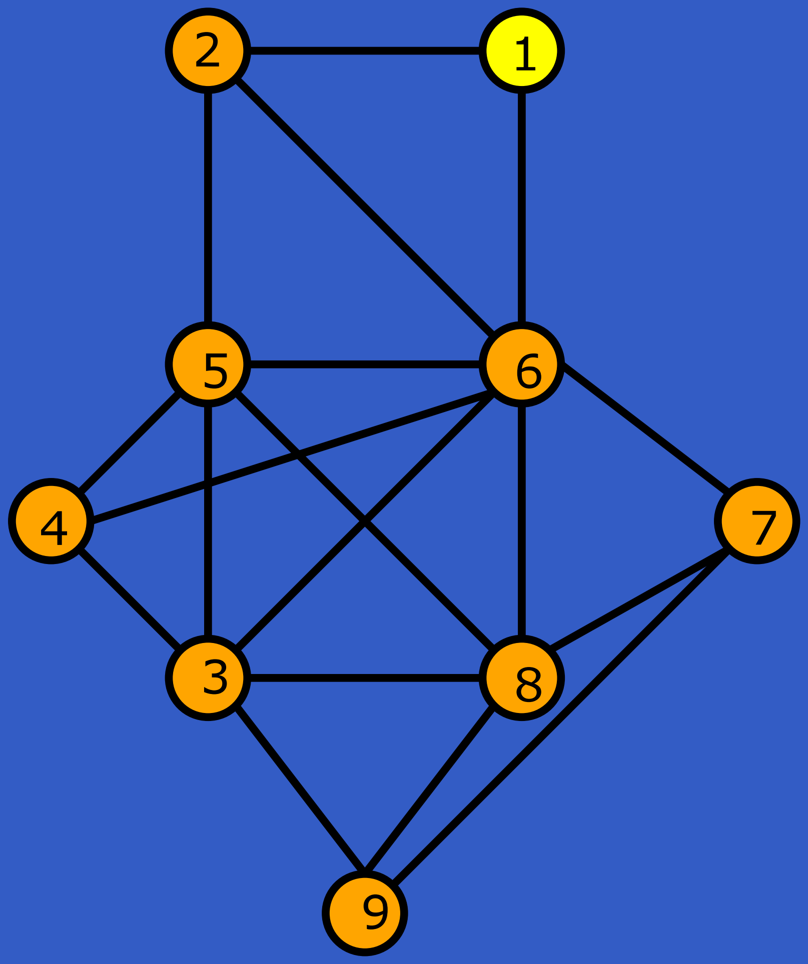
We know that eliminating variables with low degree is easy.

Boosting search by VE

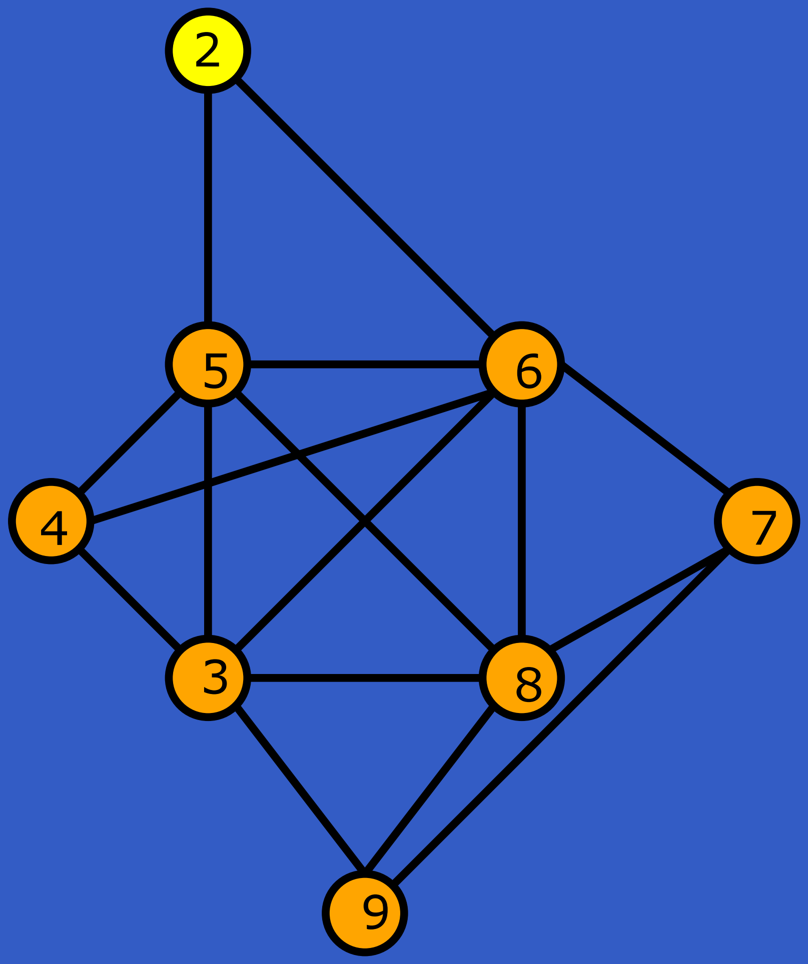
Consider (X, D, C, S)

- if all var. are eliminated, return optimal cost.
- while there are easy to eliminate variables, eliminate.
- choose a variable to branch on
- for all possible values
  - fix the variable value (easy to elim.)
  - call recursively and update the best cost

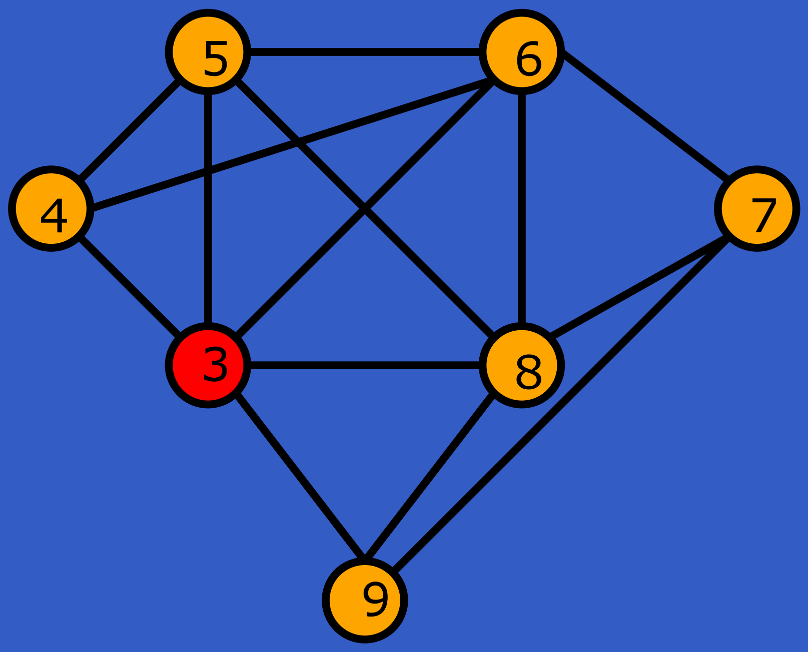
Example



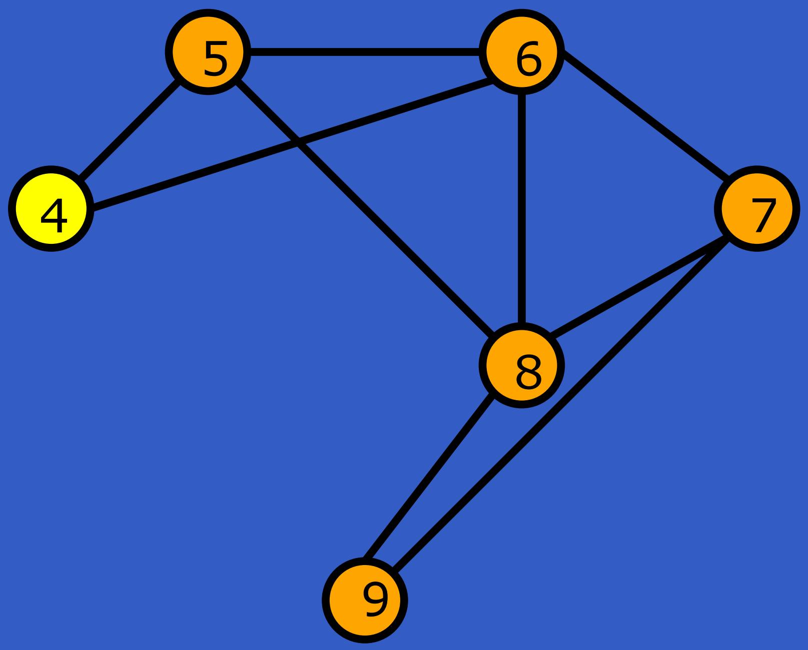
Example



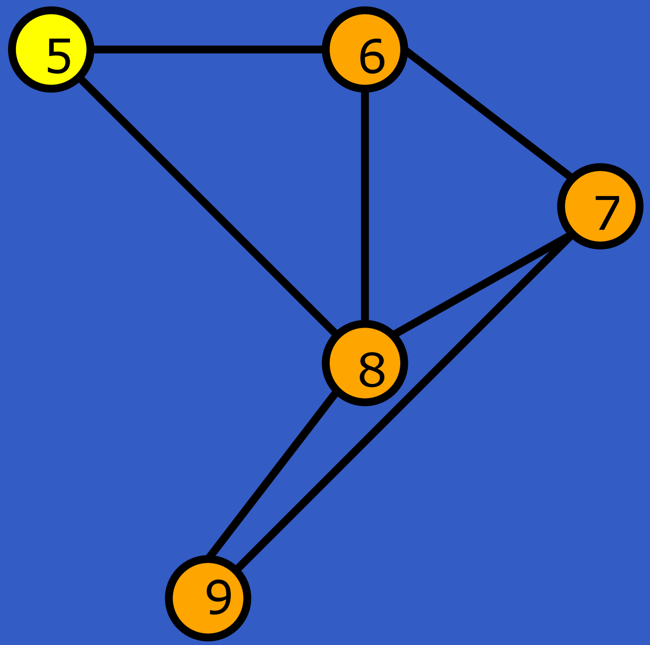
Example



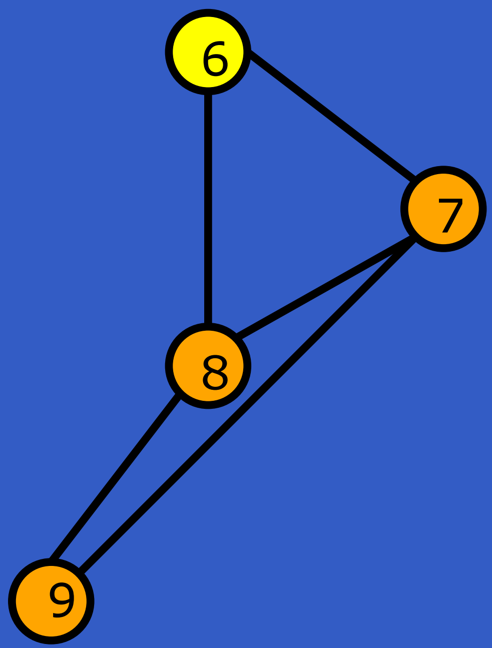
Example



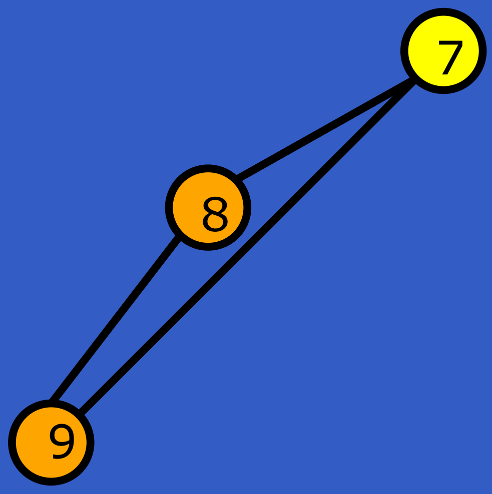
Example



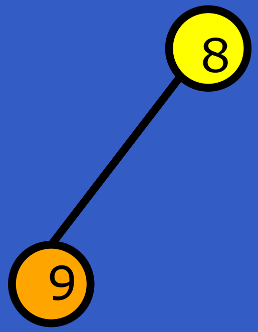
Example



Example

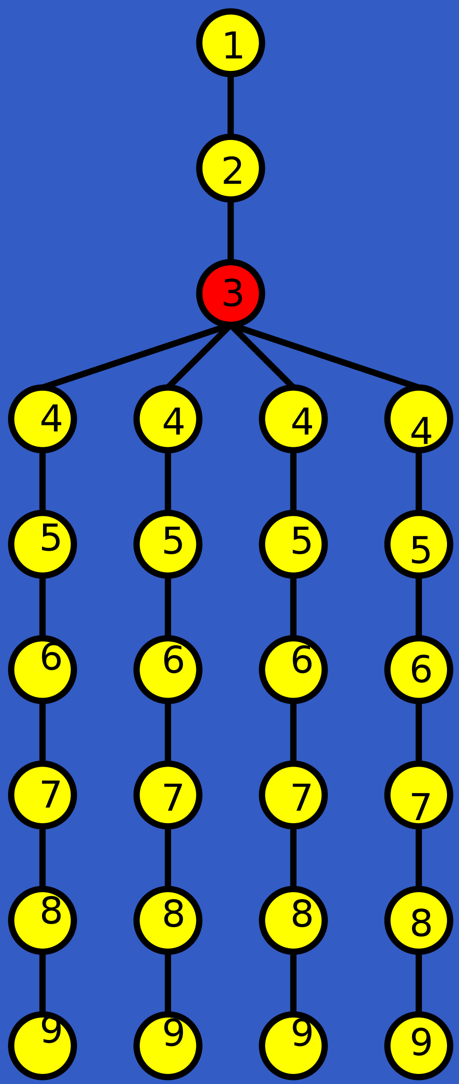


Example



Example

Example



Computing a lb by VE: mini buckets

When we eliminate a variable x, $\bowtie K(x)$ can be expensive.

Instead we can:

- partition $K(x)$ in sets of bounded size $K(x) = \cup K_i(x) \; |K_i(x)| \leq b$.
- Compute each $\bowtie K_i(x)$ and project: yields $\kappa_i(x)$
- eliminate x (and $K(x)$), replace by all $\kappa_i(x)$

Computing a lb by VE: mini buckets

The problem may be not equivalent:

We ignore interactions.

It is less constrained. Its optimal cost will be a lb on the optimal cost of the original problem.

Repeat recursively: polynomial in $O(d^b)$. The larger b the better the lb. Ultimately: will be bucket elim.

Using mini-buckets inside Branch and Bour

- use a static variable ordering
- compute all mini-buckets (process var. in reverse order)
- Perform B&B using the static variable ordering

For a given node, with future variables F, consider all the $\kappa_i(x)$ for $x \in F$. Only some are assigned at the node. Let κ be the set of these.
$lb_{mb}(t) = lb_d(t) + \sum_{\kappa_i \in \kappa} \kappa_i[t]$.
PFC-MRDAC is still quite competitive.

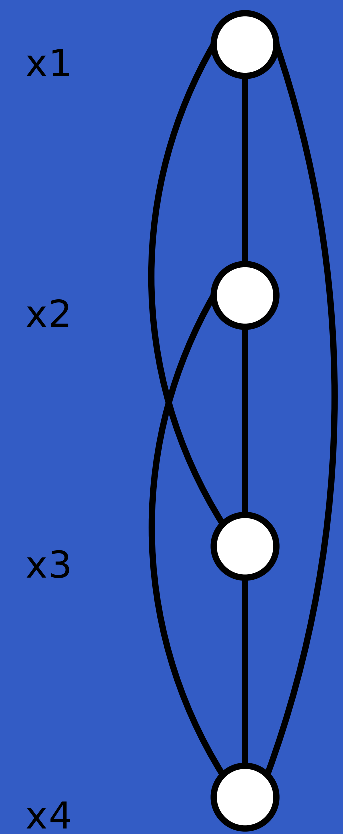
Russian Doll Search (Lemaitre et al. 1996)

A general way to boost a lower bound that ignores constraints linking future variables (like PFC).

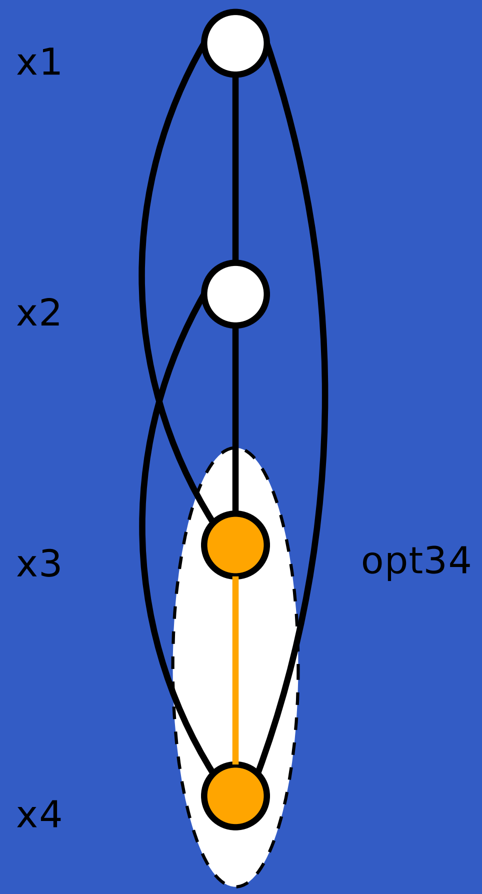
At a given node, we have a set F of unassigned variables and constraints linking them.

Main idea: the cost of an optimal solution of the problem P_F defined by F and relevant constraints can be added to $lb_d \oplus lb_{fc}$ and is still a lb. We must solve P_F beforehand.

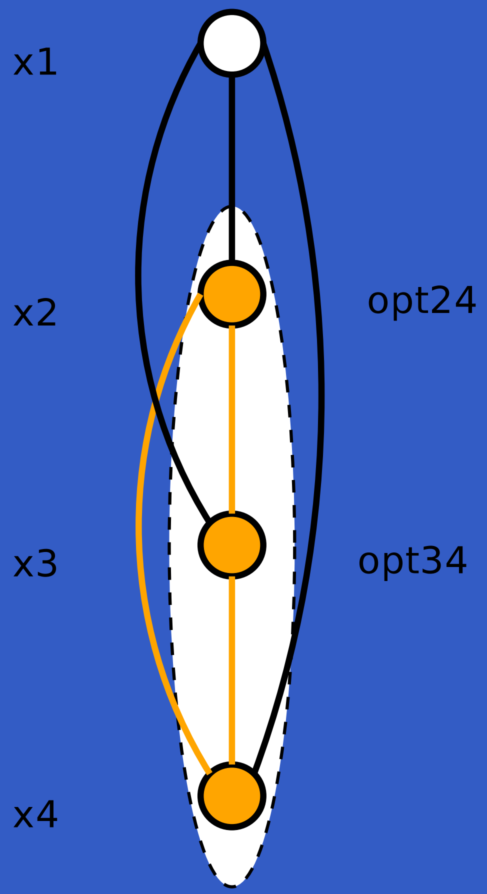
RDS in image



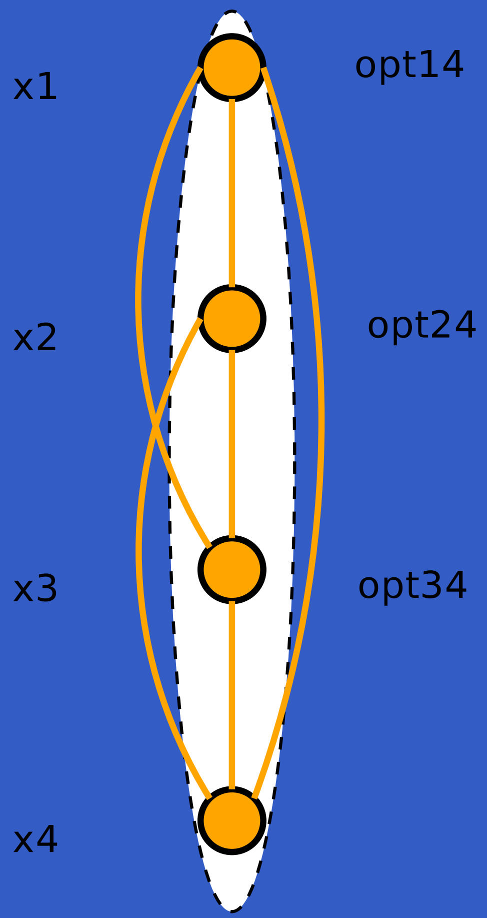
RDS in image



RDS in image



RDS in image



RDS

- use a static variable ordering $x_1 \dots x_n$
- call P_k the problem defined by variables x_k to x_n and relevant constraints.
- Solve P_n to P_1 using the following lb (where O_k denotes the optimal cost of P_k:

$$lb(t) = lb_{fc}(t) \oplus O_k$$

where k is the first unassigned variable in t.
Solving P_k gives also value ordering heuristics and upper bounds (using the solution found when solving P_k).

Specialized RDS

When we solve P_k, we do not only compute the optimal solution O_k but for each value a of x_k, the first variable of P_k:
$O_k^a =$ best solution of P_k that uses $x_k = a$.

$$lb(t) = lb_d(t) \oplus \min_{a \in D_k} (fc_{ka} \oplus O_k^a) \oplus \bigoplus_{x_i \in F, j \neq k} \min_{a \in Di} fc_{ja}$$

where k is the first unassigned variable in t.
A flavor of VE: solves path-structured efficiently.