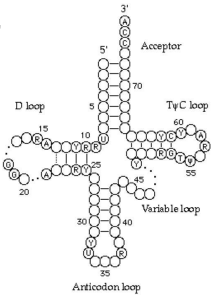
Soft Constraint Processing

16.412J/6.834J Cognitive Robotics

Martin Sachenbacher
(Using material from Thomas Schiex)

Example: Bioinformatics

- **RNA** is single-strand molecule, composed of A,U,G,C
- Function of RNA depends on **structure** (3-D folding)
- Structure induced by **base pairing**: Watson-Crick (A-U, G-C) and Wobble (G-U).
- Problem: Find RNA structure that **maximizes** base pairings.
- Cumbersome to frame as Optimal CSP!

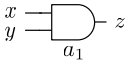


Overview

- Soft Constraints Framework
- Algorithms: Search (Branch-and-Bound)
- Algorithms: Inference (Dynamic Programming)
- Applications: Frequency Assignment Problems

From Optimal CSP to Soft CSP

- Optimal CSP: Minimize **function** f(y), s.t. **constraints** C(x) are **satisfiable**.



Utility Function

$f : a_1 \rightarrow [0,1]$

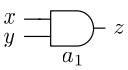
$f(\mathrm{G}) = .99$
$f(\mathrm{U}) = .01$

Constraint

$c:$ a_1	x	y	z
G	0	0	0
G	0	1	0
G	1	0	0
G	1	1	1
U	0	0	0
...			

From Optimal CSP to Soft CSP

- Soft CSP: **Extend** the notion of **constraints** to **include preferences**.



Soft Constraint

$c:$ a_1	x	y	z	
G	0	0	0	.99
G	0	1	0	.99
G	1	0	0	.99
G	1	1	1	.99
U	0	0	0	.01
...				...

Notation

- A k-**tuple** is a sequence of $k$objects $\langle v_1, \ldots, v_k \rangle$
- The i-th component of a tuple t is denoted $t[i]$.
- The **projection** of a tuple t on a subset $S$of its components is denoted $t[S]$.
- The **cartesian product** of sets $A_1, \ldots, A_k$, denoted $\Pi_{i=1}^{k} A_i$, is the set of all k-tuples such that $t[i] \in A_i$.

Classical CSP

A constraint network $\langle X, D, C \rangle$
- set of **variables** $X = \{x_1, \ldots, x_n\}$
- set of **domains** $D = \{d_1, \ldots, d_n\}$
- set of **constraints** $C = \{c_1, \ldots, c_m\}$

A constraint $c \in C$ is a **relation** $c \subseteq \Pi_{x_j \in \mathrm{var}(c)} d_j$ on variables $\mathrm{var}(c)$ with arity $|\mathrm{var}(c)|$.

A complete assignment t is **allowed** if $\forall c \in C, t[\mathrm{var}(c)] \in c$.

Valued CSP

- For each **constraint/tuple**: a **valuation** that reflects preference (e.g. cost, weight, priority, probability, ...).
- The **valuation of an assignment** is the combination of the valuations expressed by each constraint using a **binary operator** (with special axioms).
- Assignments can be compared using a **total order** on valuations.
- The problem is to produce an **assignment of minimum valuation**.

Formally: Valuation Structure

$S = \langle E, \oplus, \preceq, \bot, \top \rangle$

- E = **set of valuations**, used to assess assignments
- $\bot$ = **minimum element** of E, corresponds to totally consistent assignments
- $\top$ = **maximum element** of E, corresponds to totally inconsistent assignments
- $\preceq$ = **total order** on E, used to compare two valuations
- $\oplus$ = **operator** used to **combine** two valuations

Valued CSP

A constraint network $\langle X, D, C, S \rangle$
- set of **variables** $X = \{x_1, \ldots, x_n\}$
- set of **domains** $D = \{d_1, \ldots, d_n\}$
- set of **constraints** $C = \{c_1, \ldots, c_m\}$
- **valuation structure** $S = \langle E, \oplus, \preceq, \bot, \top \rangle$

A constraint $c \in C$ is a **function** $c : \Pi_{x_j \in \mathrm{var}(c)} d_j \rightarrow E$ mapping tuples over $\mathrm{var}(c)$ to valuations.

The **valuation** of a complete assignment t is $\bigoplus_{c \in C} c(t[\mathrm{var}(c)])$.

Required Properties

- $\forall \alpha, \beta \in E, (\alpha \oplus \beta) = (\beta \oplus \alpha)$. (Commutativity)
- $\forall \alpha, \beta, \gamma \in E, (\alpha \oplus (\beta \oplus \gamma)) = ((\alpha \oplus \beta) \oplus \gamma)$. (Associativity)
- $\forall \alpha, \beta, \gamma \in E, (\alpha \preceq \beta) \Rightarrow ((\alpha \oplus \gamma) \preceq (\beta \oplus \gamma))$. (Monotonicity)
- $\forall \alpha \in E, (\alpha \oplus \bot) = \alpha$. (Neutral element)
- $\forall \alpha \in E, (\alpha \oplus \top) = \top$. (Annihilator)

Exercise: Justify properties.

Instances of the Framework

	E	$\preceq$	$\bot$	$\top$	$\oplus$
Classical	$\{\mathrm{t,f}\}$	$\mathrm{t} \prec \mathrm{f}$	t	f	$\wedge$
Weighted	$N_0^+ \cup \infty$	$\leq$	0	∞	$+$
Probabilistic	$[0,1]$	$\geq$	1	0	$*$
Fuzzy	$[0,1]$	$\geq$	1	0	min

Many others in the literature.

From Valued CSP to Optimal CSP

- Introduce **decision variable** for each constraint
- Its **values** correspond to **different valuations**

c :

x	y	z	
a	a	a	0
a	b	a	0
b	a	a	1
b	b	b	1

$S = \langle N_0^+ \cup \infty, +, \leq, 0, \infty \rangle$

From Valued CSP to Optimal CSP

- Introduce **decision variable** for each constraint
- Its **values** correspond to **different valuations**

c :

d	x	y	z	
v_1	a	a	a	0
v_1	a	b	a	0
v_2	b	a	a	1
v_2	b	b	b	1

$S = \langle N_0^+ \cup \infty, +, \leq, 0, \infty \rangle$

From Valued CSP to Optimal CSP

- Introduce **decision variable** for each constraint
- Its **values** correspond to **different valuations**
- **Utility function** maps values to valuations
- Constraints become **relations**

c :

d	x	y	z
v_1	a	a	a
v_1	a	b	a
v_1	b	a	a
v_2	b	b	b

$f : d \rightarrow N_0^+ \cup \infty$

$f(v_1) = 0$

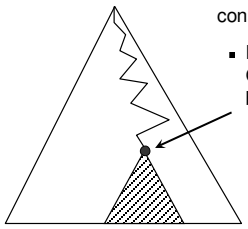
$f(v_2) = 1$

Multiattribute utility function $= +$

Overview

- Soft Constraints Framework
- **Algorithms: Search (Branch-and-Bound)**
- Algorithms: Inference (Dynamic Programming)
- Applications: Frequency Assignment Problems

Branch-and-Bound Search



- Each search node is a soft constraint subproblem
  - **Lower Bound (lb):** Optimistic estimate of best solution in subtree
    - **Upper Bound (ub):** Best solution found so far
      - Prune, if **lb ≥ ub**.

Branch-and-Bound Algorithm

- **Function** DFBB (t : assignment, ub: value): value
  - $v \leftarrow lb(t)$
  - **if** $v \prec ub$ **then**
    - **if** $|t| = n$ **then return** v
    - **let** x_i be an unassigned variable
    - **for each** $a \in d_i$ **do**
      - $ub \leftarrow \min(ub, \text{DFBB}(t \cup \{(i, a)\}, ub))$
    - **return** ub
  - **return** |

Time: O(exp(n))

Space: O(n)

Lower Bound Procedure

Must be:

- **Strong**: the closest to the real value, the better.
- **Efficient**: as easy to compute as possible.

Creates a **trade-off**. Choice is often a matter of compromises and experimental evaluation.

Distance Lower Bound

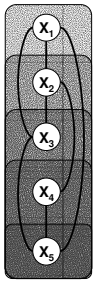
- At each node, let $AC \subseteq C$ be the set of constraints all of whose variables have been assigned.
- Use the bound

$$lb(t) = \bigoplus_{c \in AC} c(t[\mathrm{var}(c)])$$

- Problem: often **weak**, as it takes into account only **past variables**.

Improvement: Russian Doll Search

- Idea: we can add the value of the **optimal solution** to the **subproblem** over **future variables** to distance lower bound, and get a stronger lower bound.
- Must solve subproblem over future variables **beforehand**.
- Yields **recursive** procedure that solves increasingly large subproblems.



Russian Doll Search

- [Lemaitre Verfaillie Schiex 96]: Experiments with Earth Observation Satellite Scheduling Problems (maximization problem).
- Example: 105 variables, 403 constraints.
- Branch-and-Bound with distance lower bound: Aborted after **30 min**, best solution so far = **8095**.
- Russian Doll Search: Optimal solution = **9096** found in **2.5 sec**.

Overview

- Soft Constraints Framework
- Algorithms: Search (Branch-and-Bound)
- **Algorithms: Inference (Dynamic Programming)**
- Applications: Frequency Assignment Problems

Inference

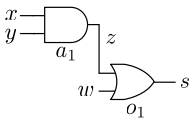
- Inference produces new constraints that are **implied** by the problem.
- Makes problem more **explicit**, easier to solve.
- Operations on constraints: **combination** and **projection**.

VCSP ⟶ VCSP'

Equivalent, simpler to solve

Combination

- $c_1 \bowtie c_2$ is constraint on $\mathrm{var}(c_1) \cup \mathrm{var}(c_2)$ s.t.
  $(c_1 \bowtie c_2)(t) = c_1(t[\mathrm{var}(c_1)]) \oplus c_2(t[\mathrm{var}(c_2)])$



c_1 :

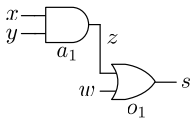
a_1	x	y	z	
G	0	0	0	.99
G	0	1	0	.99
G	1	0	0	.99
G	1	1	1	.99
U	0	0	0	.01
...				...

c_2 :

o_1	z	w	s	
G	0	0	0	.95
G	0	1	1	.95
G	1	0	1	.95
G	1	1	1	.95
U	0	0	0	.05
...				...

Combination

- $c_1 \bowtie c_2$ is constraint on $\mathrm{var}(c_1) \cup \mathrm{var}(c_2)$ s.t.
  $(c_1 \bowtie c_2)(t) = c_1(t[\mathrm{var}(c_1)]) \oplus c_2(t[\mathrm{var}(c_2)])$

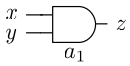


$c_1 \bowtie c_2$:

a_1	o_1	x	y	z	w	s	
G	G	0	0	0	0	0	.9405
G	G	0	0	0	1	1	.9405
G	U	0	0	0	0	1	.0495
U	G	1	1	0	0	0	.0092
U	U	0	0	0	0	0	.0005
...							...

Projection

- $c \Downarrow_Y, Y \subseteq X$ is a constraint on $\mathrm{var}(c_1) \cap Y$ s.t.
  $(c \Downarrow_Y)(t) = \min_{t'[Y]=t} c(t')$



c :

a_1	x	y	z	
G	0	0	0	.99
G	0	1	0	.99
G	1	0	0	.99
G	1	1	1	.99
U	0	0	0	.01
...				...

$c \Downarrow_{\{x,z\}}$:

x	z	
0	0	.99
1	0	.99
1	1	.99
0	1	.01

Inferring Solutions

- Constraint network $\langle X, D, C, S \rangle$
- Value of optimal solution obtained by **combining all constraints** C and **eliminating all variables** X:

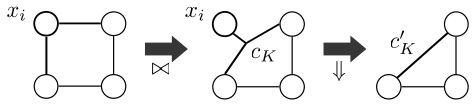
$$(c_1 \bowtie c_2 \bowtie \ldots \bowtie c_m) \Downarrow_\emptyset$$

- Problem: **Very costly**: Time O(exp(n)), Space O(exp(n).

Improvement: Bucket Elimination

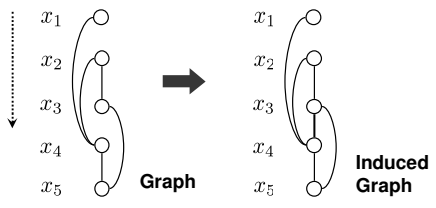
Idea: Eliminate variable as **soon as it no longer occurs** in **remaining** set of constraints.

- For variable $x_i \in X$, let $K_{x_i} = \{c \in C : x_i \in \mathrm{var}(c)\}$
- Compute **combination** c_K of all constraints in K_{x_i}
- Now **eliminate** x_i from c_K: $c'_K = c_K \Downarrow_{X \setminus \{x_i\}}$
- **Remove** K_{x_i} from C and **add** c'_K to C.



Induced Graph

- When processing a node (variable), **connect all neighboring nodes** not yet processed.

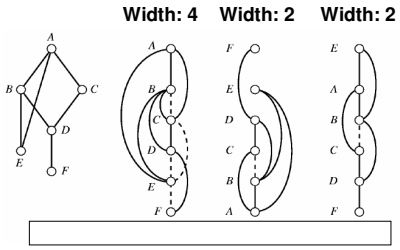


Bucket Elimination: Complexity

Let **width** be the maximum number of successors in the induced graph. Then:

- **Time** dominated by computation of largest c_K: **O(exp(width+1))**
- **Space** dominated by storage of largest c'_K: **O(exp(width))**

Impact of Variable Ordering



Finding ordering with minimal width is NP-hard.

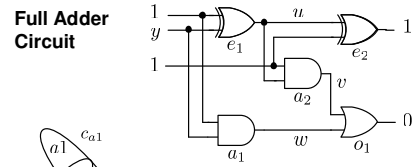
Min-Fill Ordering Heuristic

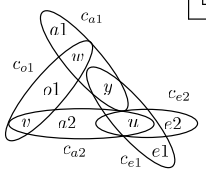
- Function MF (G: Graph with edges E and nodes $V = \{v_1, \ldots, v_n\}$): Order
  - **for** $j = 1$ **to** n
    - **let** v be a node in V with minimal number of edges required to connect its neighbors
    - put v in position j of order
    - $E \leftarrow E \cup \{(v_i, v_j) : (v_i, v) \in E, (v_j, v) \in E\}$
    - $V \leftarrow V \setminus \{v\}$

Often finds good orderings in practice.

Example

Full Adder Circuit





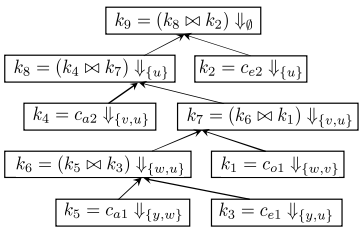
Full Adder Circuit Hyper-Graph

Example

MF Order

$1: o_1$
$2: e_2$
$3: e_1$
$4: a_2$
$5: a_1$
$6: y$
$7: w$
$8: v$
$9: u$

Computational Scheme ("Bucket Tree")

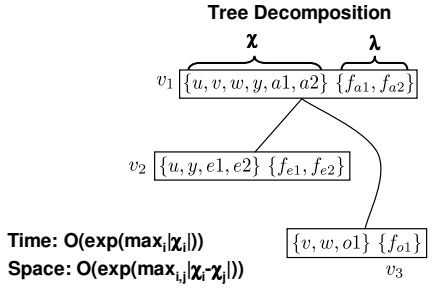


Tree Decomposition

A **tree decomposition** for a problem $\langle X, D, C, S \rangle$ is a triple (T, χ, λ), where $T = (V, E)$ is a rooted tree, and χ, λ are labeling functions associating with each node $v_i \in V$ two sets $\chi(v_i) \subseteq X$, $\lambda(v_i) \subseteq C$ such that:

- For each $c \in C$, there is exactly one v_i such that $c \in \lambda(v_i)$. For this $v_i, \mathrm{var}(c) \subseteq \lambda(v_i)$ (**covering condition**);
- For each $x \in X$, the set $\{v_j \in V : x \in \chi(v_j)\}$ of vertices labeled with x induces a connected subtree of T (**connectedness condition**).

Example

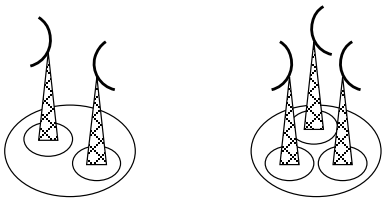


Time: O(exp(max$_i$|χ_i|))
Space: O(exp(max$_{i,j}$|χ_i-χ_j|))

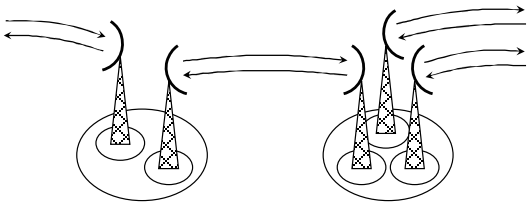
Overview

- Soft Constraints Framework
- Algorithms: Search (Branch-and-Bound)
- Algorithms: Inference (Dynamic Programming)
- **Applications: Frequency Assignment Problems**

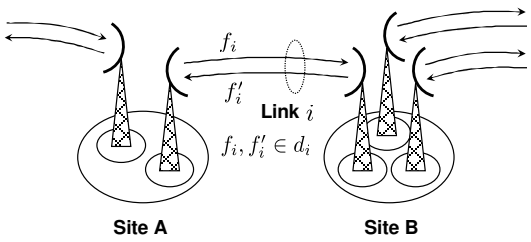
Frequency Assignment



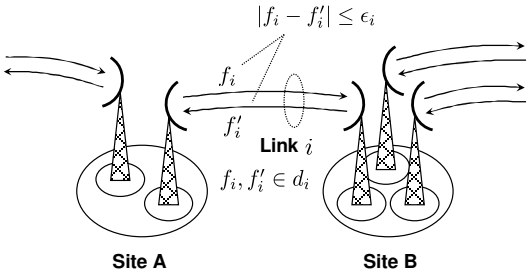
Frequency Assignment



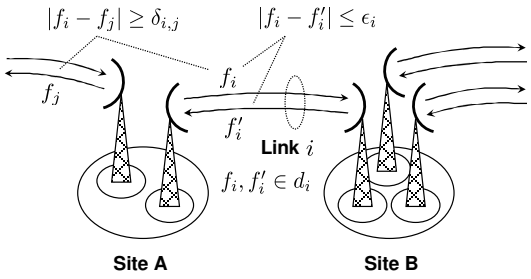
Frequency Assignment



Frequency Assignment



Frequency Assignment



Frequency Assignment

- Several instances available from CELAR (200 to 916 variables, 1200 to 5000 constraints, domain size >30)
- Very hard instances for branch-and-bound search.
- Good results reported for using tree decomposition.