Part 2: Algorithms
Overview

• Problem formulation:
  – Optimization task

• Exact algorithms:
  – Branch and Bound
  – Partial Forward Checking
  – Reversible DACs
  – Russian Doll Search
  – Bucket Elimination

• Approximate algorithms:
  – Local search approaches
  – Interval approximation
Terminology

- Variables:  $i, j, k, ...$
- Values:  $a, b, c, ...$
- Constraints:  $f, g, h, ...$  \( \text{var}(f) \): variables involved in $f$
- Domains:  \( D_0(i) \): initial domain of variable $i$
  \( D(i) \): current domain of variable $i$
- $P$: set of assigned or past variables
- $F$: set of unassigned or future variables
- $C_P$: set of constraints involving past variables
- $C_{PF}$: set of constraints involving past and future variables
- $C_F$: set of constraints involving future variables
- $\tau$: current assignment
- $\tau[i]$: current assignment projected over variable $i$
Problem Formulation

• Soft CSP: \((X, D, C)\)
  - \(X\) is a set of \(n\) variables
  - \(D = \{ D_0(1), D_0(2), \ldots, D_0(n) \}\) is a collection of variable domains
  - \(C\) is a set of \(r\) soft constraints

\[ f \in C \quad f : \prod_{i \in \text{var}(f)} D_0(i) \rightarrow [0, +\infty] \]

\[ f(t) = \begin{cases} 0 & \text{t satisfies completely } f \\ (0, +\infty) & \text{t satisfies / violates partially } f \\ +\infty & \text{t violates completely } f \end{cases} \]

\( f(t) \): cost associated with violation of \(f\) by \(t\)

• Goal: \(\text{minimize } \sum_{t \in C} f(t)\) weighted CSP
  (NP-hard)
Branch and Bound

- Depth-first tree search:
  - *internal node*: partial assignment
  - *leaf*: total assignment

- At each node:
  
  *Distance*:  \( \text{dist}(\tau) = \sum_{f \in C_P} f(\tau) \)

  *Upper bound (UB)*: minimum distance of visited leaves: distance of the current best solution

  *Lower bound (LB)*: underestimation of minimum distance among leaves below current node

  *Pruning*:  \( UB \leq LB \)

- Simplest LB:  \( \text{dist}(\tau) \)
Lower Bound

LB quality: very important for branch and bound efficiency
Partial Forward Checking
[Freuder & Wallace 92]

• Branch and Bound + Lookahead

• Cost of value $a$ of future variable $i$ because constraint $f$:

$$\text{cost} \ (i, a, f) = \min_{t[i]=a} f(t) \quad t \in \prod_{i \in \text{var}(f)} D(i) \quad \text{valid } t$$

$$\text{cost}_0 \ (i, a, f) = \min_{t[i]=a} f(t) \quad t \in \prod_{i \in \text{var}(f)} D_0(i) \quad \text{initial } t$$

• Lookahead: after assigning a value to a variable
  – inconsistency counts on future values are computed
  – inconsistency count of value $a$ of future variable $i$:

$$ic \ (i, a) = \sum_{f \in C_{PF}} \text{cost} \ (i, a, f)$$

$C_{PF}$: constraints involving one past and one future variables

$binary$ constraints
Inconsistency Counts

[Freuder & Wallace 92]

\[ \sum_{i \in F} \min_a (ic(i,a)) \]
PFC Lower Bound
[Freuder & Wallace 92]

• New lower bound:

\[
LB(\tau, F) = \text{dist}(\tau) + \sum_{i \in F} \min_a \{ic(i,a)\}
\]

\[
\text{cost from } C_P \quad \text{cost from } C_{PF}
\]

\[
LB_{jb}(\tau, F) = \text{dist}(\tau) + ic(j,b) + \sum_{i \in F - \{j\}} \min_a \{ic(i,a)\}
\]

• Future value pruning: \( LB_{jb}(\tau, F) \geq UB \)
  – the cost of extending \( \tau \) with \((j,b)\) is greater than or equal to the cost of the current best solution
  – \((j,b)\) can be pruned: it will never improve the current best solution
  – when backtracking to \(i\), \((j,b)\) must be restored
Static DAC

[Wallace 95]

- DAC: (directed arc-inconsistency counts) on future values
  - static variable ordering: 1, 2, ..., i, ..., j, ..., n
  - \( C_F \): constraints involving two future variables
  \[
dac(i,a) = \sum_{f \in C_F} cost_0(i,a,f) \quad var(f) = (i, j), \ i < j
\]

Static order: 1, 2, ..., i, ..., j, ..., n

Contributions to the lower bound

\[
\sum_{i \in F} \min_a (dac(i,a))
\]
Combining IC + DAC

[Wallace 95] [Larrosa & Meseguer 96]

Static order: $1, 2, \ldots, i, \ldots, j, \ldots, n$

IC and DAC of value $a$ of variable $i$:
- refer to different constraints
- can be added

- New lower bound:

$$LB (\tau, F) = \text{dist} (\tau) + \sum_{i \in F} \min_a \{\text{ic} (i, a) + \text{dac} (i, a)\}$$

$$\text{cost from } C_P$$

$$LB_{jb} (\tau, F) = \text{dist} (\tau) + \text{ic} (j, b) + \text{dac} (j, b) +$$

$$\sum_{i \in F - \{j\}} \min_a \{\text{ic} (i, a) + \text{dac} (i, a)\}$$

$$\text{cost from } C_{PFUC_F}$$
DAC: Directed Constraints

\[ \min \text{dac}(i) + \min \text{dac}(j) = 1+1=2 \text{ but } f(a,a) = 1! \]

- For DAC usage, constraints must be directed
- DAC stored in the variable pointed by the constraint
- Implicit in static DAC by static variable order
Graph-based DAC

[Larrosa, Meseguer, Schiex 99]

- Directed constraint graph $G^F$
  - $\text{Nodes}(G^F) = F$
  - $\text{Edges}(G^F) = \{(j, i) \mid C_F, \text{directed from } j \text{ to } i\}$

- DAC based on $G^F$:
  $$\text{dac}(i, a, G^F) = \sum_{f \in C_F} \text{cost}_0(i, a, f) \quad \text{var}(f) = (i, j), \quad (j, i) \in \text{Edges}(G^F)$$

- New lower bound:
  $$\text{LB}(\tau, F, G^F) = \text{dist}(\tau) + \sum_{i \in F} \min_a \{ic(i, a) + \text{dac}(i, a, G^F)\}$$
  $$\text{LB}_{jb}(\tau, F, G^F) = \text{dist}(\tau) + ic(j, b) + \text{dac}(j, b, G^F) + \sum_{i \in F - \{j\}} \min_a \{ic(i, a) + \text{dac}(i, a, G^F)\}$$
Reversible DAC

[Larrosa, Meseguer, Schiex 99]

- Reversible RDAC:
  - Any $G^F$ is suitable for LB computation
  - Dynamically selects a good $G^F$ (optimal NP-hard, greedy search)
  - Single operation: reversing edge direction

- Maintaining DAC: redefinition $cost_0 \rightarrow cost$

$$
dac (i, a, G^F) = \sum_{f \in C_F} cost (i, a, f) \quad \text{var}(f) = (i, j), (j, i) \in Edges(G^F)
$$

- Previous DAC: initially precomputed
- DAC can be maintained at run time
  - removed values generate further DAC
  - AC adapted algorithm
Russian Doll Search
[Verfaillie, Lemaitre, Schiex 96]

- To replace one search by \( n \) successive searches on nested subproblems

- Sequence subpr: 1, 2, \ldots, \( i \), \ldots, \( n \)

- Each subproblem is optimally solved:
  
  \[
  rds \ (\text{subpr}) = \text{optimal cost of subpr}
  \]

- When solving subproblem \( i+1 \) costs of solutions of subproblem \( i \) to 1 are used
Russian Doll Search
[Verfaillie, Lemaitre, Schiex 96]

• New lower bound:

\[ LB(\tau, F) = \text{dist}(\tau) + \sum_{i \in F} \min_a \{ic(i,a)\} + rds(F) \]

\[ \text{cost from } C_P \quad \text{cost from } C_{PF} \quad \text{cost from } C_F \]

• Solving subproblem \( i+1 \):

\[
\begin{array}{cccc}
\text{n-i, n-i+1, n-i+2, \ldots, n} & F = \text{subpr } i \\
\text{P} & \text{F} & \text{F} = \text{subpr } i-1 \\
\text{n-i, n-i+1, n-i+2, \ldots, n} & \text{F} = \text{subpr } 1 \\
\text{P} & \text{F} & \text{F} = \text{subpr } 1 \\
\end{array}
\]
Bucket Elimination
[Dechter 99]

• Dynamic programming/ Variable elimination:
  – no tree search
  – synthesize the best solution
  – static variable ordering: 1, 2, . . . ., n

• Bucket $k$: set of constraints involving variable $k$ and $i, j, ...$
  such that $i, j,... < k$.

• Operations on constraints:
  – Addition: $f(t) + g(t) = f(t[\text{var}(f)]) + g(t[\text{var}(g)])$
  
  – Projecting out variable $k$: $(f \downarrow k)(t) = \min_{a \in D(k)} f(t) \quad t[k] = a$
## Bucket Elimination

[Dechter 99]

### Process from bucket $n$ to 1:

1. Add all constraints, getting a new one
   \[ g = f_1 + f_2 + \ldots + f_p \]

2. Project out of $g$ variable $k$
   \[ h = g \downarrow k \]

3. Add $h$ to the corresponding bucket

### Comments:

- Constraint $g$ summarizes all information of constraints $f_1, f_2, \ldots, f_p$
- Each iteration transforms a problem $P$ into an equivalent $P'$ with one less var
- The process iterates until no variables

<table>
<thead>
<tr>
<th>Var</th>
<th>Bucket</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>bucket $n$</td>
</tr>
<tr>
<td>$n-1$</td>
<td>bucket $n-1$</td>
</tr>
<tr>
<td>$n-2$</td>
<td>bucket $n-2$</td>
</tr>
<tr>
<td>$1$</td>
<td>bucket 1</td>
</tr>
</tbody>
</table>
Bucket Elimination

[Dechter 99]

Example:

Solution: from var 1 to $n$

Assign to variable $i$ the best value in $g$ according with previous assignements

Complexity: Space and time exponential in the induced width of the graph: \textit{max arity} of new constraints
Search + Variable Elimination

[Larrosa 00]

- Branching:

```
if eliminating variable i causes a new constraint with arity <= k
then eliminate variable i
else do branching on the most connected variable
```

- Combining strategy: $k$ parameter

```
lookahead
changes graph topology
```

$k=2$

```
var elim
branch
var elim
var elim
```

backtrack to branching points
Local Search

• Optimization problem: minimize $\Sigma_{f \in C} f(t)$

• Local search approaches: produce an upper bound
  – hill climbing [Hao & Dorne 96]
  – simulated annealing [Wah & Chen 00]
  – tabu search [Galinier & Hao 97]
  – genetic algorithms: special operators
    [Lau & Tsang 01] [Wiese & Goodwin 01]

• Elements:
  – evaluation function: function to optimize
  – neighborhood: one move, two moves, ...
  – selection criteria: new state, escaping local optima,
    randomization, cooling schedule, ...

Interval Approximation

- in many cases, looking for the optimum is too complex
- instead, look for an interval $[LB, UB]$ containing the optimum
- when $[LB, UB]$ is narrow enough, stop search

Idea:

- $LB_1, LB_2, \ldots, LB_k$: problem simplification
- $UB_1, UB_2, \ldots, UB_k$: objective simplification
- $LB$: three strategies
- $UB$: local search approaches

Anytime bounds:

- $UB$: local search approaches
- $LB$: three strategies

- problem simplification:
- objective simplification:
- sum of local costs
- russian doll + iterative deepening