Factor Graph and Constraint Satisfaction Problems (CSPs) 2

Hwanjo Yu

POSTECH

http://di.postech.ac.kr/hwanjoyu

Roadmap

Modeling

Definitions

Examples

Backtracking (exact) search

Dynamic ordering

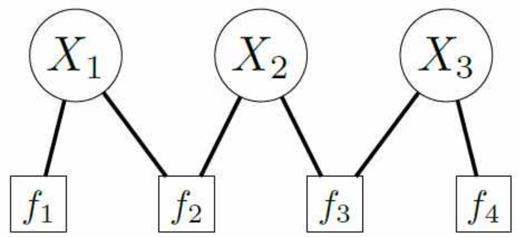
Arc consistency

Approximate search

Beam search

Local search

Review: factor graph and CSPs



Definition: factor graph

- Variables:

$X = (X_1, \dots, X_n)$, where $X_i \in \mathrm{Domain}_i$

- Factors:

$f_1, \dots, f_m$, with each $f_j(X) \geq 0$

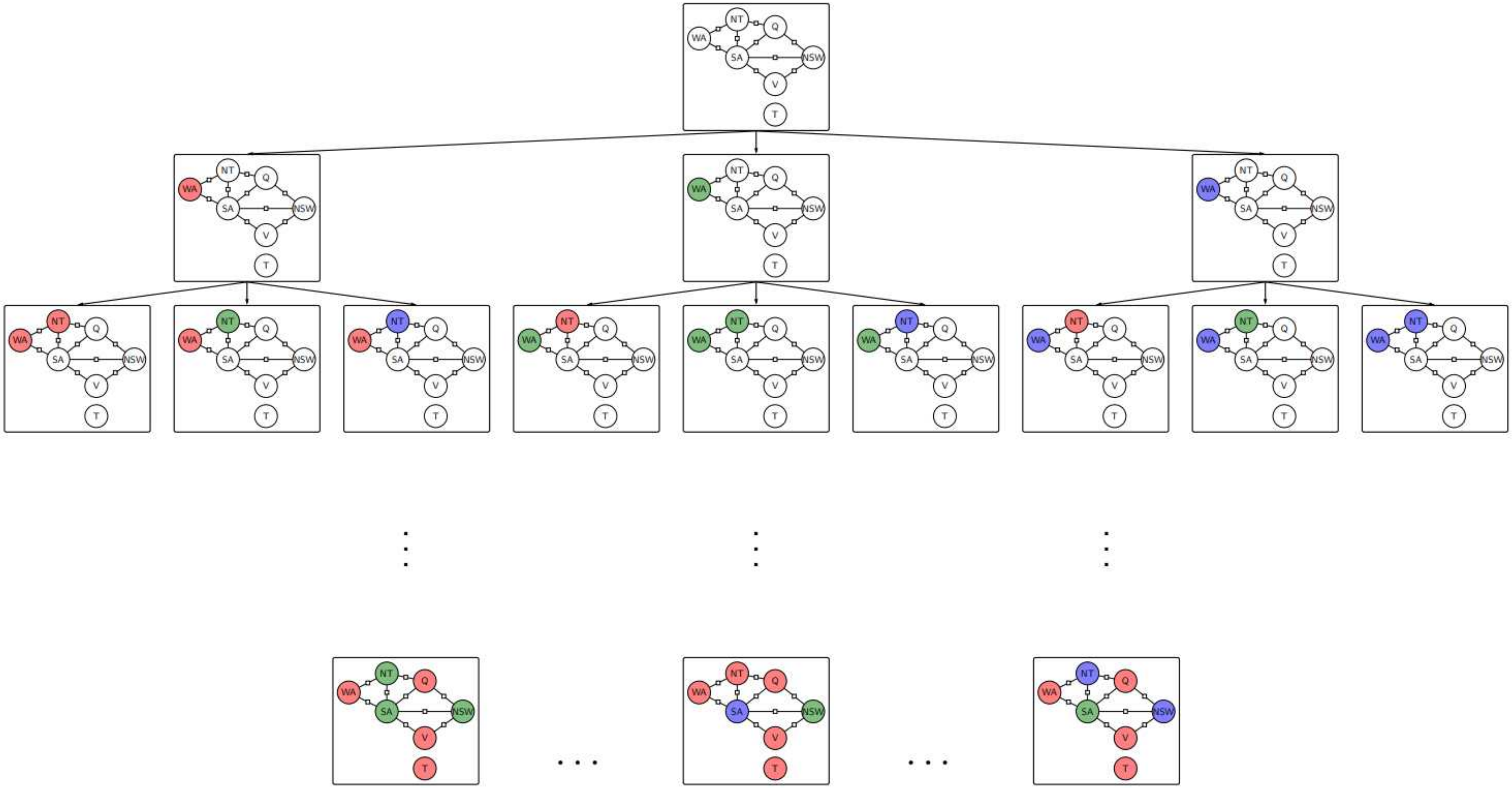
Definition: assignment weight

- Each **assignment** $x = (x_1, \dots, x_n)$ has a **weight**:

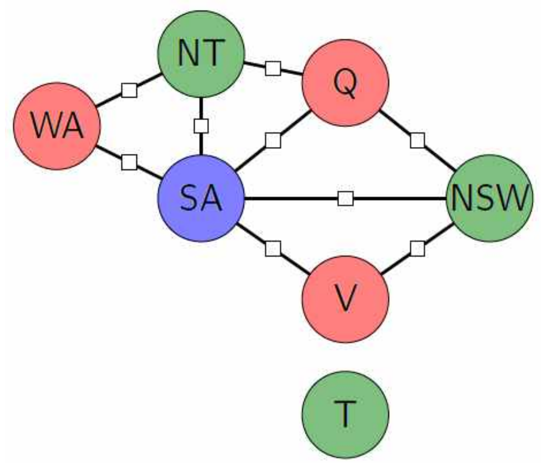
$$\mathrm{Weight}(x) = \prod_{j=1}^{m} f_j(x)$$

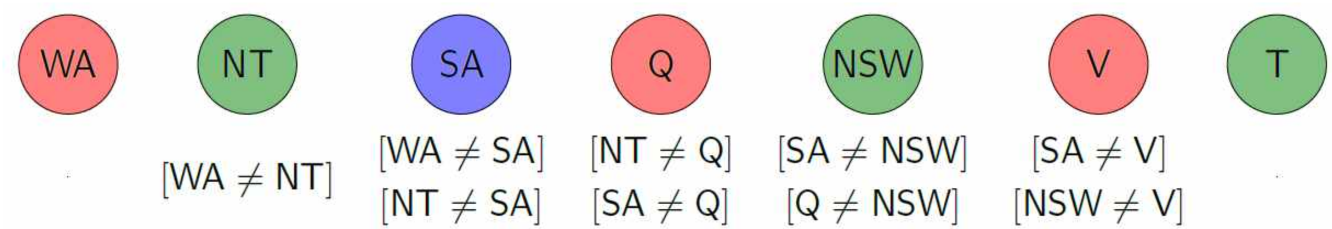
Objective: find the maximum weight assignment

$$arg \max_x \mathrm{Weight}(x)$$


WA
NT
Q
SA
NSW
V
T

Partial assignment weights

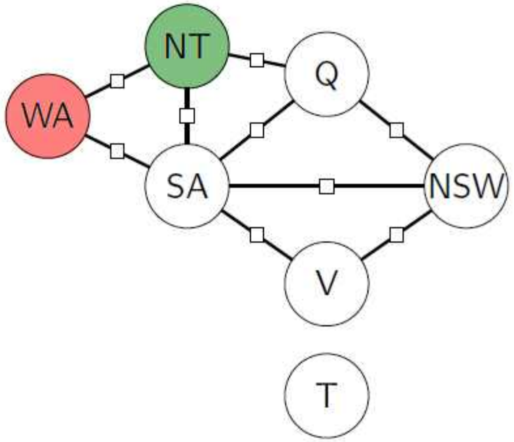




- Compute weight of partial assignments as we go. (Weight of partial assignment is product of all factors whose scope includes <u>only</u> assigned variables.)

Dependent factors

- Partial assignment (e.g., $x = \{\text{WA}: \text{R}, \text{NT}: \text{G}\}$)



Definition: dependent factors

- Let $D(x, X_i)$ be set of factors depending on X_i but not on unassigned variables.
- $D(\{\text{WA}: \text{R}, \text{NT}: \text{G}\}, \text{SA}) = \{[\text{WA} \neq \text{SA}], [\text{NT} \neq \text{SA}]\}$

Backtracking search

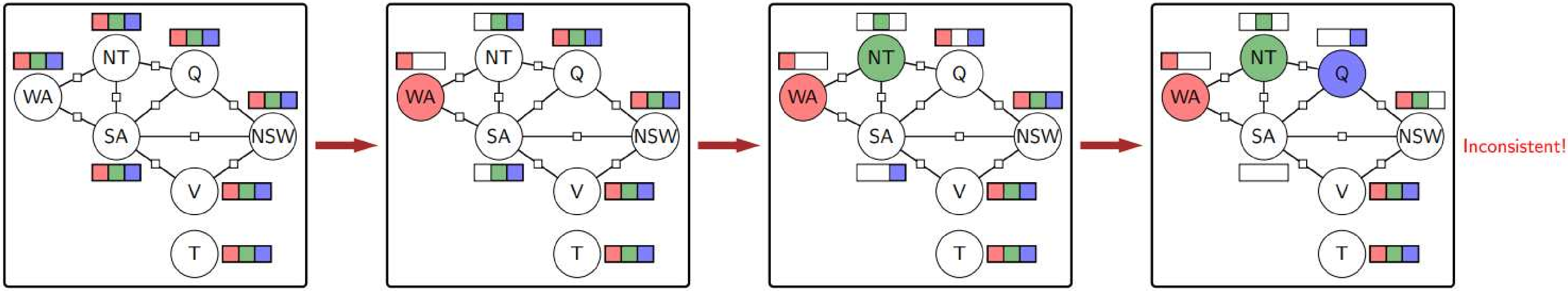
Algorithm: backtracking search

- Backtrack($x, w, \mathrm{Domains}$):
  - If x is complete assignment: update best and return
  - Choose unassigned VARIABLE X_i
  - Order VALUES Domain_i of chosen X_i
  - For each value v in that order:
    - $\delta \leftarrow \prod_{f_j \in D(x, X_i)} f_j(x \cup \{X_i : v\})$
    - If $\delta = 0$: continue
    - $\mathrm{Domains}' \leftarrow \mathrm{Domains}$ via LOOKAHEAD
    - If any $\mathrm{Domains}_i'$ is empty: continue
    - Backtrack($x \cup \{X_i : v\}, w\delta, \mathrm{Domains}'$)

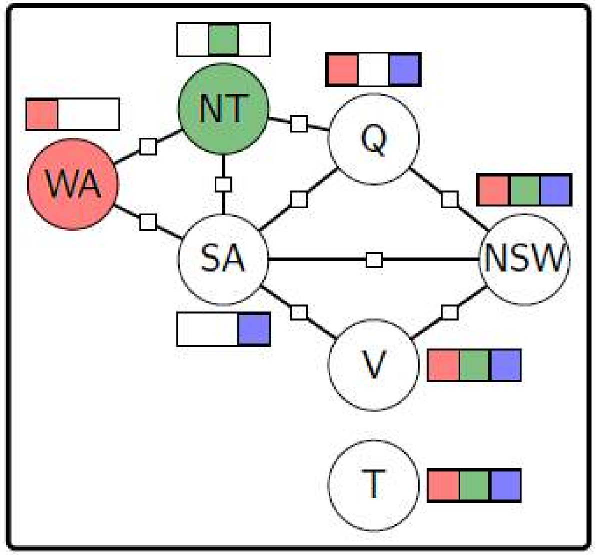
Backtracking search: Lookahead (forward checking)

Key idea: forward checking (one-step lookahead)

- After assigning a variable X_i, eliminate inconsistent values from the domains of X_i's neighbors.
- If any domain becomes empty, return.



Choosing an unassigned variable



Which variable to assign next?

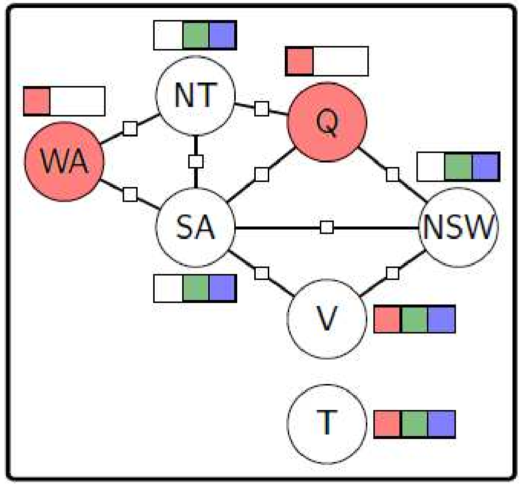
Key idea: most constrained variable

- Choose variable that has the smallest domain.

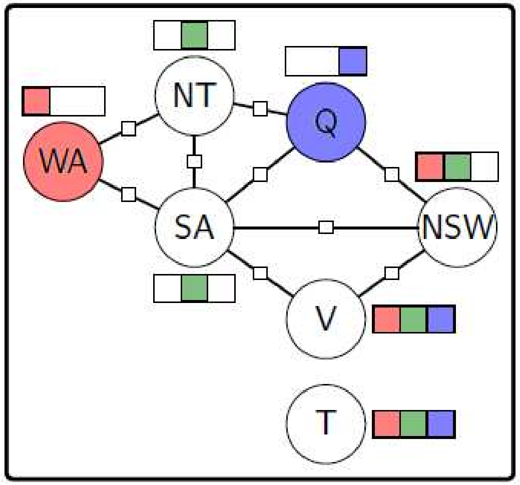
This example: SA (has only one value)

Order values of a selected variable

What values to try for Q?



$2+2+2=6$ consistent values



$1+1+2=4$ consistent values

Key idea: least constrained value

- Order values of selected X_i by decreasing number of consistent values of neighboring variables.

When to fail?

Most constrained **variable** (MCV):

- Must assign **every** variable
- If going to fail, fail early => more pruning

Least constrained **value** (LCV):

- Need to choose **some** value
- Choosing value most likely to lead to solution

When do these heuristics help?

- Most constrained variable: useful when **some** factors are constraints (can prune assignments with weight 0)

- Least constrained value: useful when **all** factors are constraints (all assignment weights are 1 or 0)

- Forward checking: needed to prune domains to make heuristics useful!

Review: backtracking search with heuristics

Algorithm: backtracking search

- Backtrack($x, w, \mathrm{Domains}$):
    - If x is complete assignment: update best and return
    - Choose unassigned **VARIABLE** X_i (MCV)
    - Order **VALUES** Domain_i of chosen X_i (LCV)
    - For each value v in that order:
        - $\delta \leftarrow \prod_{f_j \in D(x, X_i)} f_j(x \cup \{X_i : v\})$
        - If $\delta = 0$: continue
        - $\mathrm{Domains}' \leftarrow \mathrm{Domains}$ via **LOOKAHEAD** (forward checking)
        - If any $\mathrm{Domains}_i'$ is empty: continue
        - Backtrack($x \cup \{X_i : v\}, w\delta, \mathrm{Domains}'$)

Roadmap

Modeling

Definitions

Examples

Backtracking (exact) search

Dynamic ordering

Arc consistency

Approximate search

Beam search

Local search

Review: backtracking search with heuristics

Algorithm: backtracking search

- Backtrack($x, w, \mathrm{Domains}$):
  - If x is complete assignment: update best and return
  - Choose unassigned **VARIABLE** X_i (MCV)
  - Order **VALUES** Domain_i of chosen X_i (LCV)
  - For each value v in that order:
    - $\delta \leftarrow \prod_{f_j \in D(x, X_i)} f_j(x \cup \{X_i : v\})$
    - If $\delta = 0$: continue
    - $\mathrm{Domains}' \leftarrow \mathrm{Domains}$ via **LOOKAHEAD (AC-3)**
    - If any $\mathrm{Domains}_i'$ is empty: continue
    - Backtrack($x \cup \{X_i : v\}, w\delta, \mathrm{Domains}'$)

Arc consistency

Idea: eliminate values from domains => reduce branching

Example: numbers

- Before enforcing arc consistency on X_i:
  - $X_i \in \text{Domain}_i = \{1,2,3,4,5\}$
  - $X_j \in \text{Domain}_j = \{1,2\}$
  - Factor: $[X_i + X_j = 4]$
- After enforcing arc consistency on X_i:
  - $X_i \in \text{Domain}_i = \{2,3\}$

Arc consistency

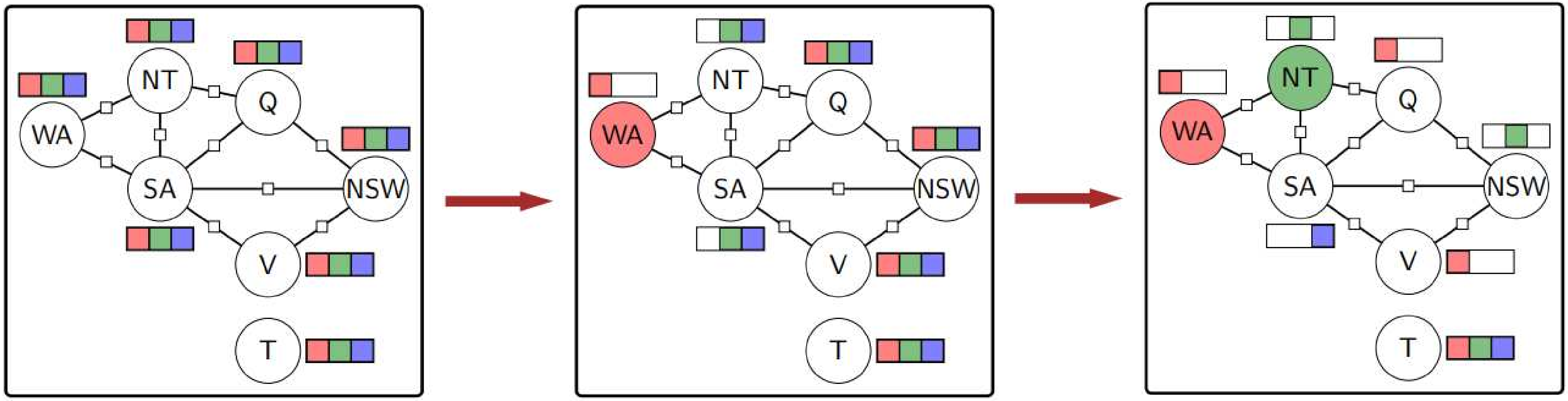
Definition: arc consistency

- A variable X_i is **arc consistent** with respect to X_j if for each $x_i \in \mathrm{Domain}_i$, there exists $x_j \in \mathrm{Domain}_j$ such that $f(\{X_i: x_i, X_j: x_j\}) \neq 0$ for all factors f whose scope contains X_i and X_j .

Algorithm: enforce arc consistency

- $\mathrm{EnforceArcConsistency}(X_i, X_j)$: Remove values from Domain_i to make X_i arc consistent with respect to X_j.

AC-3 (example)



AC-3

Forward checking: when assign $X_j : x_j$, set $\mathrm{Domain}_j = \{x_j\}$ and enforce arc consistency on all neighbors X_i with respect to X_j

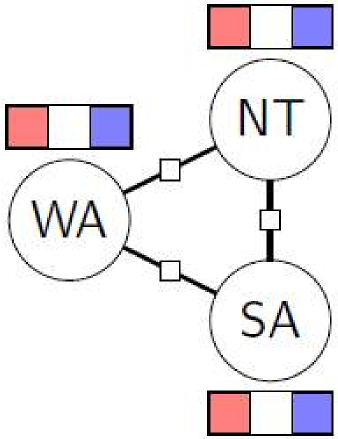
AC-3: repeatedly enforce arc consistency on all variables

Algorithm: AC-3

- $S \leftarrow \{X_j\}$.
- While S is non-empty:
  - Remove any X_j from S.
  - For all neighbors X_i of X_j:
    - Enforce arc consistency on X_i w.r.t. X_j.
    - If Domain_i changed, add X_i to S.

Limitations of AC-3

- AC-3 isn't always effective:



- No consistent assignments, but AC-3 doesn’t detect a problem!
- Intuition: if we look locally at the graph, nothing blatantly wrong…

Summary

- Enforcing arc consistency: make domains consistent with factors
- Forward checking: enforces arc consistency on neighbors
- AC-3: enforces arc consistency on neighbors and their neighbors, etc.
- Lookahead can speed up backtracking search!

Roadmap

Modeling

Definitions

Examples

Backtracking (exact) search

Dynamic ordering

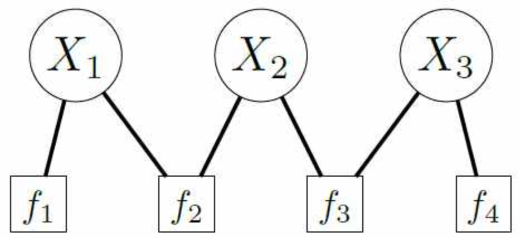
Arc consistency

Approximate search

Beam search

Local search

Review: factor graph and CSPs



Definition: factor graph

- Variables:

$X = (X_1, \dots, X_n)$, where $X_i \in \text{Domain}_i$

- Factors:

$f_1, \dots, f_m$, with each $f_j(X) \geq 0$

Definition: assignment weight

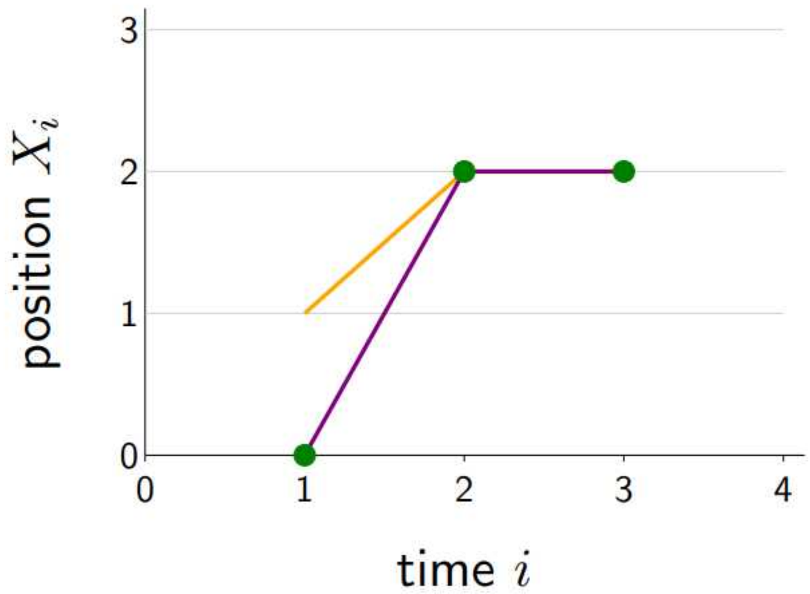
- Each **assignment** $x = (x_1, \dots, x_n)$ has a **weight**:

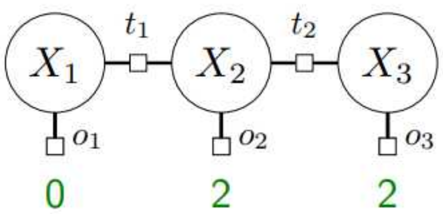
$$\text{Weight}(x) = \prod_{j=1}^{m} f_j(x)$$

Objective: find the maximum weight assignment

$$arg \ \max_x \text{Weight}(x)$$

Example: object tracking





x_1	$o_1(x_1)$
0	2
1	1
2	0

x_2	$o_2(x_2)$
0	0
1	1
2	2

x_3	$o_3(x_3)$
0	0
1	1
2	2

$\lvert x_i - x_{i+1} \rvert$	$t_i(x_i, x_{i+1})$
0	2
1	1
2	0

Review: backtracking search

Vanilla version:

$$O(|\mathrm{Domain}|^n) \text{ time}$$

Lookahead: forward checking, AC-3

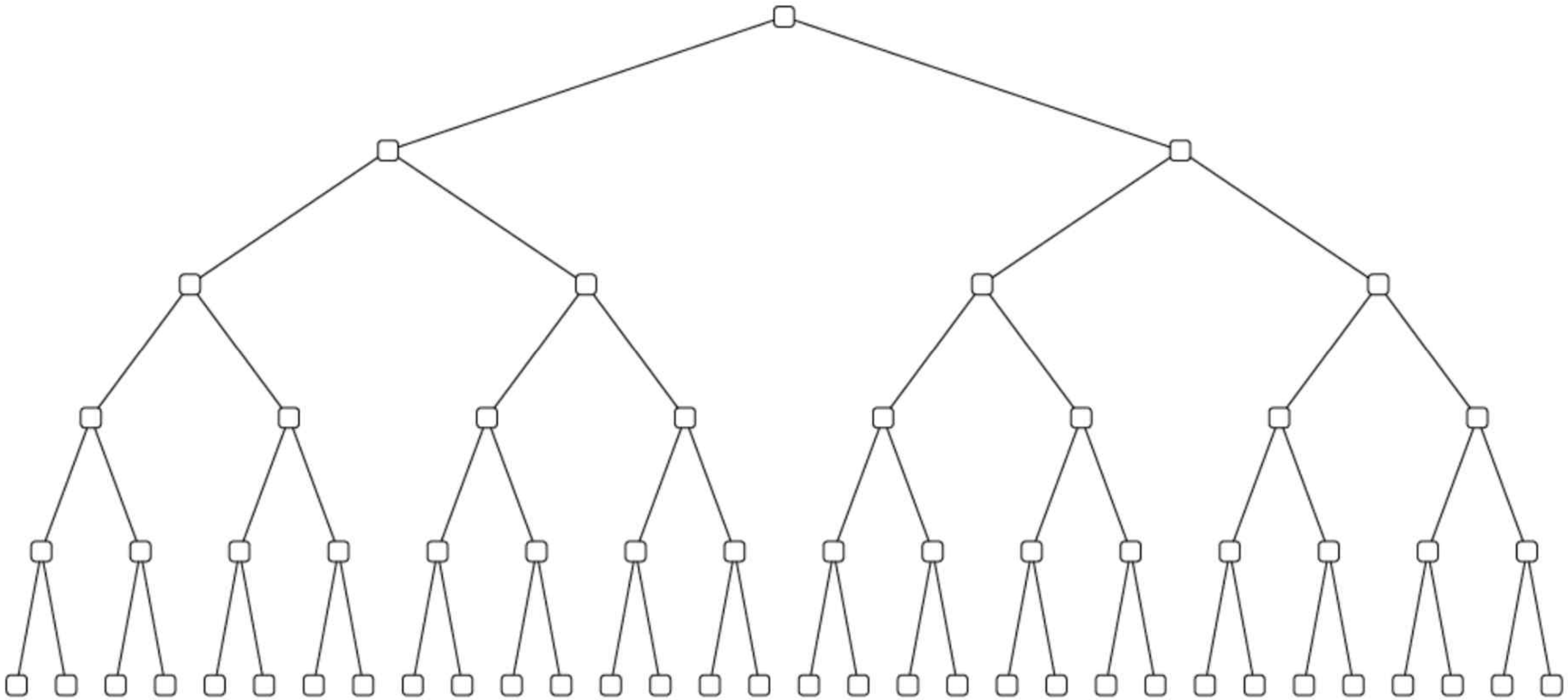
$$O(|\mathrm{Domain}|^n) \text{ time}$$

Dynamic ordering: most constrained variable, least constrained value

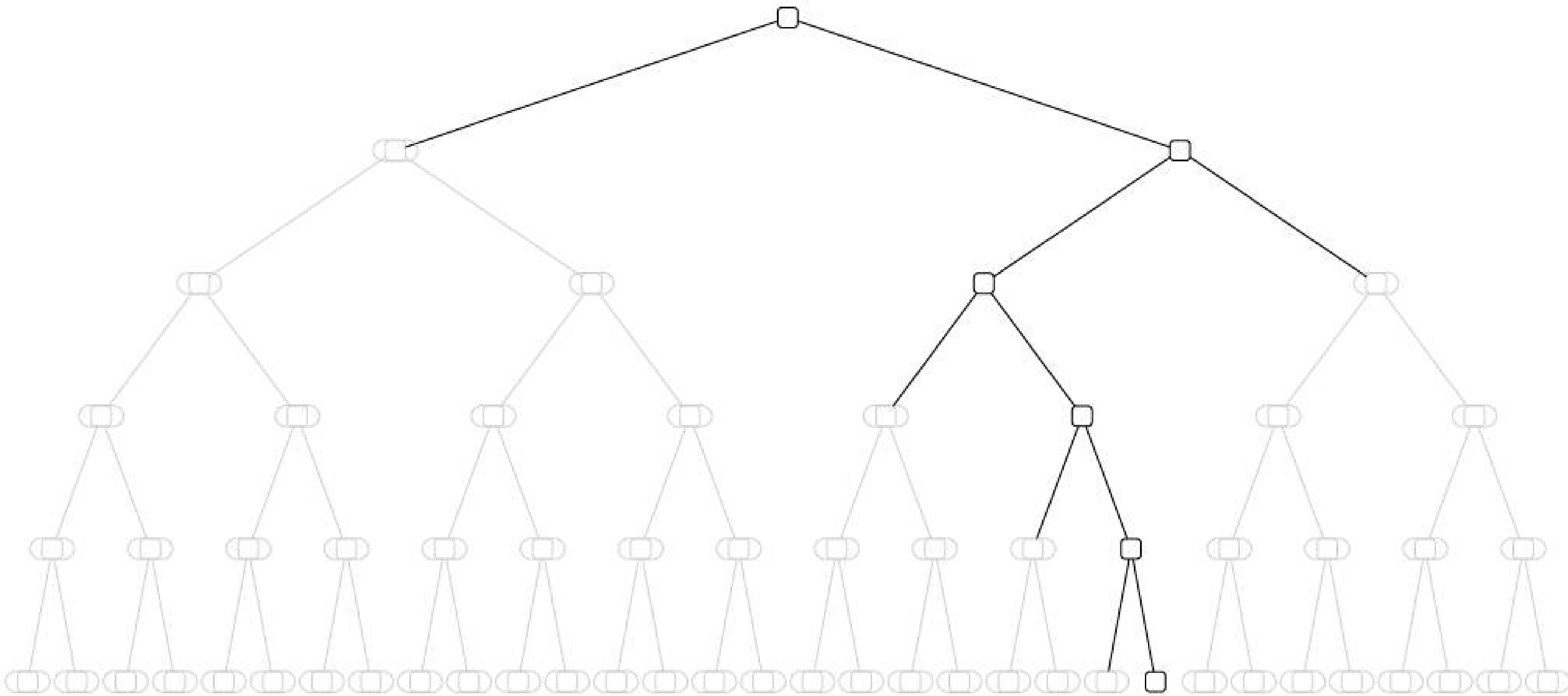
$$O(|\mathrm{Domain}|^n) \text{ time}$$

Note: these pruning techniques useful only for constraints. What if all the factors return strictly positive values?

Backtracking search



Greedy search



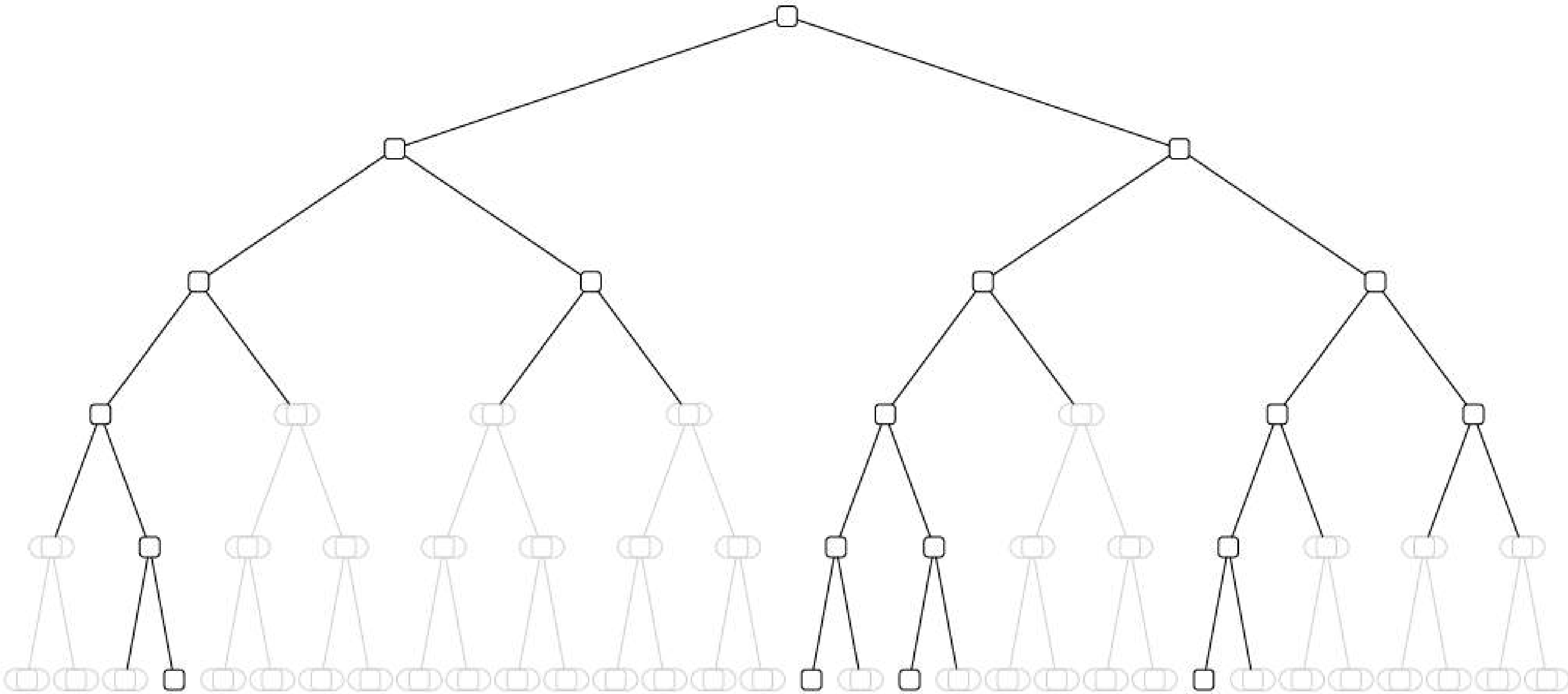
Greedy search

Algorithm: greedy search

- Partial assignment $x \leftarrow \{\}$
- For each $i = 1, \dots, n$:
  - Extend:
    - Compute weight of each $x_v = x \cup \{X_i : v\}$
  - Prune:
    - $x \leftarrow x_v$ with highest weight

Not guaranteed to find optimal assignment!

Beam search



Beam size $K = 4$

Beam search

Idea: keep $\leq K$ candidate list C of partial assignments

Algorithm: beam search

- Initialize $C \leftarrow [\{\}]$
- For each $i = 1, \ldots, n$:
  - Extend:
    - $C' \leftarrow \{x \cup \{X_i : v\} : x \in C, v \in \mathrm{Domain}_i\}$
  - Prune:
    - $C \leftarrow K$ elements of C' with highest weights

Not guaranteed to find optimal assignment!

Beam search

Time complexity:

- $K = 1$ is greedy ($O(nb)$ time) with branching factor $b = |\text{Domain}|$.
- $K = \infty$ is BFS tree search ($O(b^n)$ time)
- $O(n \cdot (Kb) \cdot \log(Kb))$ with beam size K (since need to sort Kb candidates)
- Beam size K controls tradeoff between efficiency and accuracy
- Backtracking search : DFS :: beam search : pruned BFS

Roadmap

Modeling

Definitions

Examples

Backtracking (exact) search

Dynamic ordering

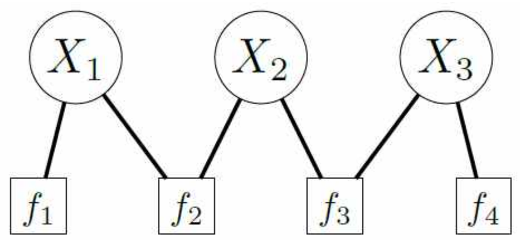
Arc consistency

Approximate search

Beam search

Local search

Review: factor graph and CSPs



Definition: factor graph

- Variables:

$X = (X_1, \dots, X_n)$, where $X_i \in \text{Domain}_i$

- Factors:

$f_1, \dots, f_m$, with each $f_j(X) \geq 0$

Definition: assignment weight

- Each **assignment** $x = (x_1, \dots, x_n)$ has a **weight**:

$$\text{Weight}(x) = \prod_{j=1}^{m} f_j(x)$$

Objective: find the maximum weight assignment

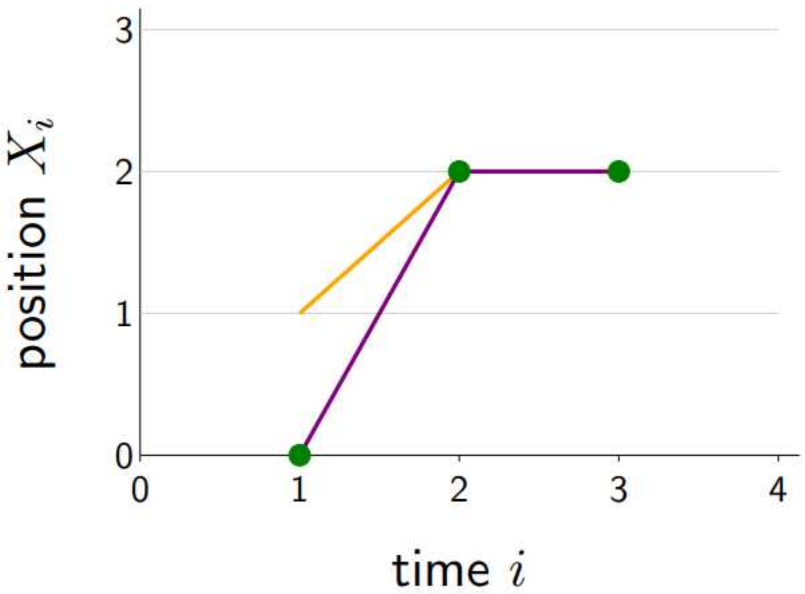
$$arg \max_x \text{Weight}(x)$$

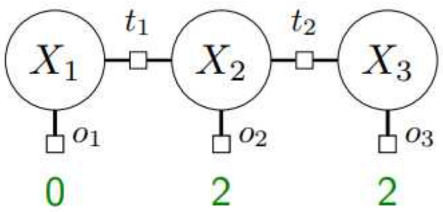
Local search

- Backtracking/beam search: extend partial assignments

- Local search: modify complete assignments

Example: object tracking





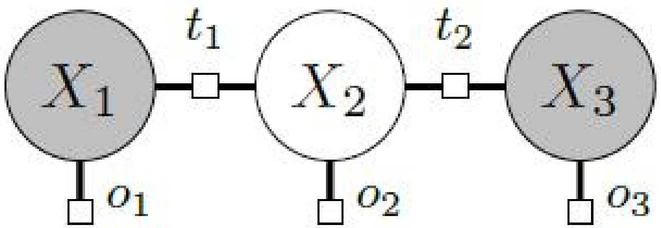
x_1	$o_1(x_1)$
0	2
1	1
2	0

x_2	$o_2(x_2)$
0	0
1	1
2	2

x_3	$o_3(x_3)$
0	0
1	1
2	2

$\lvert x_i - x_{i+1} \rvert$	$t_i(x_i, x_{i+1})$
0	2
1	1
2	0

One small step

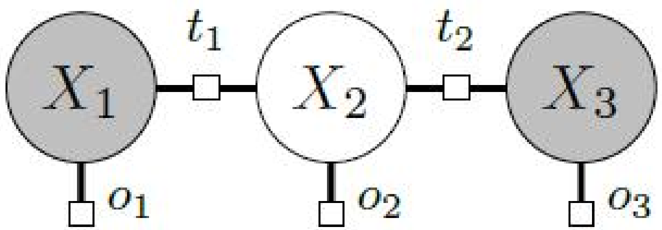


Current assignment: (0, 0, 1); how to improve?

(x_1, v, x_3)	weight
$(0,0,1)$	$2 \cdot 2 \cdot 0 \cdot 1 \cdot 1 = 0$
$(0,1,1)$	$2 \cdot 1 \cdot 1 \cdot 2 \cdot 1 = 4$
$(0,2,1)$	$2 \cdot 0 \cdot 1 \cdot 1 \cdot 1 = 0$

New assignment: (0, 1, 1)

Exploiting locality



Weight of new assignment (x_1, v, x_3)

$$o_1(x_1)t_1(x_1, v)o_2(v)t_2(v, x_3)o_3(x_3)$$

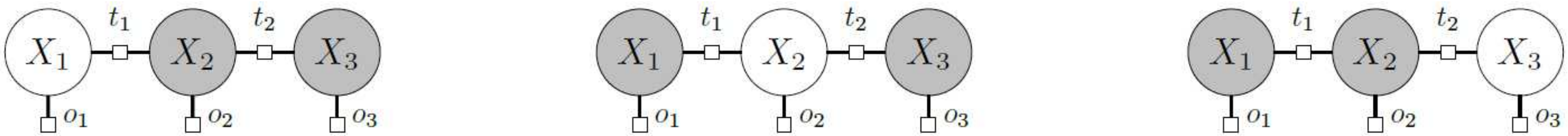
Key idea: locality

- When evaluating possible re-assignments to X_i, only need to consider the factors that depend on X_i.

Iterated conditional modes (ICM)

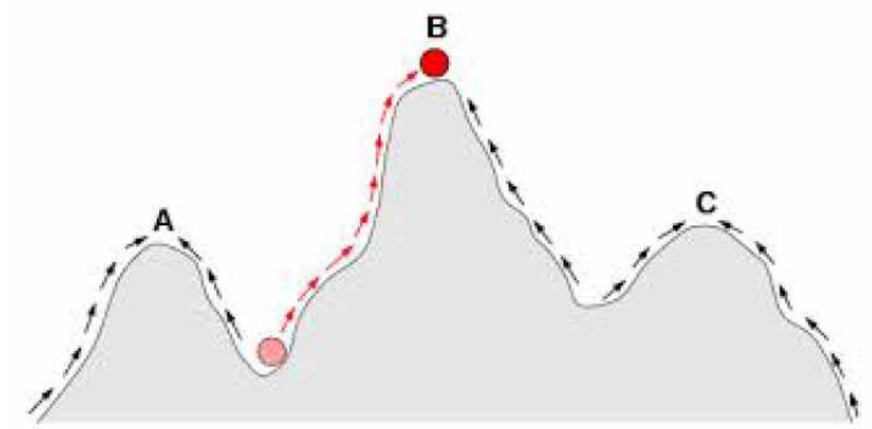
Algorithm: iterated conditional modes (ICM)

- Initialize x to a random complete assignment
- Loop through $i = 1, \ldots, n$ until convergence:
  - Compute weight of $x_v = x \cup \{X_i : v\}$ for each v
  - $x \leftarrow x_v$ with highest weight

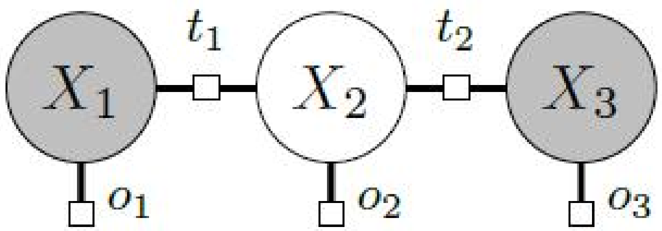


Convergence properties

- $\text{Weight}(x)$ increases or stays the same each iteration
- Converges in a finite number of iterations
- Can get stuck in **local optima**
- Not guaranteed to find optimal assignment!



Summary



Algorithms for max-weight assignments in factor graphs:

(1) Extend partial assignments:

- Backtracking search: exact, exponential time
- Beam search: approximate, linear time

(2) Modify complete assignments:

- Iterated conditional modes: approximate, linear time, deterministic
- (Markov networks) Gibbs sampling: approximate, linear time, randomized