Constraint Satisfaction Problems (CSP)

Part B

R&N: Chap. 5

Slides from Jean-Claude Latombe at Stanford University
(used with permission)

Backtracking Algorithm

CSP-BACKTRACKING(A)

1. If assignment A is complete then return A
2. X ← select a variable not in A
3. D ← select an ordering on the domain of X
4. For each value v in D do
   a. Add (X←v) to A
   b. If A is valid then
      i. result ← CSP-BACKTRACKING(A)
      ii. If result ≠ failure then return result
   c. Remove (X←v) from A
5. Return failure

Call CSP-BACKTRACKING({})

[This recursive algorithm keeps too much data in memory. An iterative version could save memory]

Critical Questions for the Efficiency of CSP-Backtracking

CSP-BACKTRACKING(A)

1. If assignment A is complete then return A
2. X ← **select** a variable not in A
3. D ← **select** an ordering on the domain of X
4. For each value v in D do
   a. Add (X←v) to A
   b. If a is valid then
      i. result ← CSP-BACKTRACKING(A)
      ii. If result ≠ failure then return result
   c. Remove (X←v) from A
5. Return failure

Critical Questions for the Efficiency of CSP-Backtracking

1) Which variable X should be assigned a value next?

2) In which order should X's values be assigned?

Critical Questions for the Efficiency of CSP-Backtracking

1) Which variable X should be assigned a value next?
   The current assignment may not lead to any solution, but the algorithm does not know it yet. Selecting the right variable X may help discover the contradiction more quickly

2) In which order should X's values be assigned?

Critical Questions for the Efficiency of CSP-Backtracking

1) Which variable X should be assigned a value next?
   The current assignment may not lead to any solution, but the algorithm does not know it yet. Selecting the right variable X may help discover the contradiction more quickly

2) In which order should X's values be assigned?
   The current assignment may be part of a solution. Selecting the right value to assign to X may help discover this solution more quickly

Critical Questions for the Efficiency of CSP-Backtracking

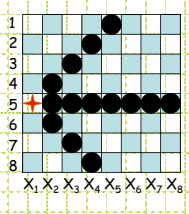
1) Which variable X should be assigned a value next?
   The current assignment may not lead to any solution, but the algorithm does not know it yet. Selecting the right variable X may help discover the contradiction more quickly

2) In which order should X's values be assigned?
   The current assignment may be part of a solution. Selecting the right value to assign to X may help discover this solution more quickly

More on these questions very soon ...

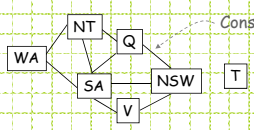
Forward Checking

A simple constraint-propagation technique:



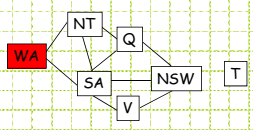
Assigning the value 5 to X_1 leads to removing values from the domains of X_2, X_3, ..., X_8

Forward Checking in Map Coloring



WA	NT	Q	NSW	V	SA	T
RGB	RGB	RGB	RGB	RGB	RGB	RGB

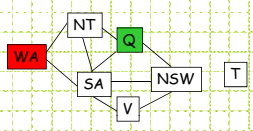
Forward Checking in Map Coloring



WA	NT	Q	NSW	V	SA	T
RGB	RGB	RGB	RGB	RGB	RGB	RGB
R	~~G~~B	RGB	RGB	RGB	~~G~~B	RGB

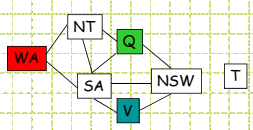
Forward checking removes the value Red of NT and of SA

Forward Checking in Map Coloring



WA	NT	Q	NSW	V	SA	T
RGB	RGB	RGB	RGB	RGB	RGB	RGB
R	GB	RGB	RGB	RGB	GB	RGB
R	~~G~~B	G	R~~G~~B	RGB	~~G~~B	RGB

Forward Checking in Map Coloring



WA	NT	Q	NSW	V	SA	T
RGB	RGB	RGB	RGB	RGB	RGB	RGB
R	GB	RGB	RGB	RGB	GB	RGB
R	B	G	RB	RGB	B	RGB
R	B	G	R~~B~~	B	~~B~~	RGB

Forward Checking in Map Coloring

Empty set: the current assignment {(WA ← R), (Q ← G), (V ← B)} does not lead to a solution

WA	NT	Q	NSW	V	SA	T
RGB	RGB	RGB	RGB	RGB	RGB	RGB
R	GB	RGB	RGB	RGB	GB	RGB
R	B	G	RB	RGB	B	RGB
R	B	G	~~R~~B	B	~~B~~	RGB

Forward Checking (General Form)

Whenever a pair (X←v) is added to assignment A do:

For each variable Y not in A do:

For every constraint C relating Y to the variables in A do:

Remove all values from Y's domain that do not satisfy C

Modified Backtracking Algorithm

CSP-BACKTRACKING(A, var-domains)

1. If assignment A is complete then return A
2. X ← select a variable not in A
3. D ← select an ordering on the domain of X
4. For each value v in D do
   a. Add (X←v) to A
   b. var-domains ← forward checking(var-domains, X, v, A)
   c. If a variable has an empty domain then return failure
   d. result ← CSP-BACKTRACKING(A, var-domains)
   e. If result ≠ failure then return result
   f. Remove (X←v) from A
5. Return failure

Modified Backtracking Algorithm

CSP-BACKTRACKING(A, var-domains)

1. If assignment A is complete then return A
2. X ← select a variable not in A
3. D ← select an ordering on the domain of X
4. For each value v in D do
   a. Add (X←v) to A
   b. var-domains ← forward checking(var-domains, X, v, A)
   c. If a variable has an empty domain then return failure
   d. result ← CSP-BACKTRACKING(A, var-domains)
   e. If result ≠ failure then return result
   f. Remove (X←v) from A
5. Return failure

No need any more to verify that A is valid

Modified Backtracking Algorithm

CSP-BACKTRACKING(A, var-domains)

1. If assignment A is complete then return A
2. X ← select a variable not in A
3. D ← select an ordering on the domain of X
4. For each value v in D do
   a. Add (X←v) to A
   b. var-domains ← forward checking(var-domains, X, v, A)
   c. If a variable has an empty domain then return failure
   d. result ← CSP-BACKTRACKING(A, var-domains)
   e. If result ≠ failure then return result
   f. Remove (X←v) from A
5. Return failure

Need to pass down the updated variable domains

Modified Backtracking Algorithm

CSP-BACKTRACKING(A, var-domains)

1. If assignment A is complete then return A
2. X ← select a variable not in A
3. D ← select an ordering on the domain of X
4. For each value v in D do
   a. Add (X←v) to A
   b. var-domains ← forward checking(var-domains, X, v, A)
   c. If a variable has an empty domain then return failure
   d. result ← CSP-BACKTRACKING(A, var-domains)
   e. If result ≠ failure then return result
   f. Remove (X←v) from A
5. Return failure

1) Which variable X_i should be assigned a value next?
   → Most-constrained-variable heuristic
   → Most-constraining-variable heuristic
2) In which order should its values be assigned?
   → Least-constraining-value heuristic

These heuristics can be quite confusing

Keep in mind that **all** variables must eventually get a value, while only **one** value from a domain must be assigned to each variable

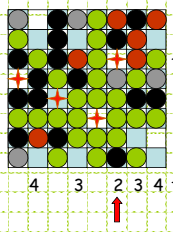
Most-Constrained-Variable Heuristic

1) Which variable X_i should be assigned a value next?

Select the variable with the smallest remaining domain

[Rationale: Minimize the branching factor]

8-Queens

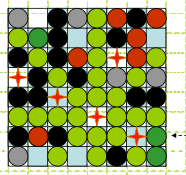


Forward checking

New assignment

Numbers of values for each un-assigned variable

8-Queens



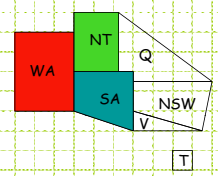
Forward checking

New assignment

3 2 1 3

New numbers of values for each un-assigned variable

Map Coloring



- SA's remaining domain has size 1 (value Blue remaining)
- Q's remaining domain has size 2
- NSW's, V's, and T's remaining domains have size 3

→ Select SA

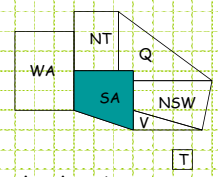
Most-Constraining-Variable Heuristic

1) Which variable X_i should be assigned a value next?

Among the variables with the smallest remaining domains (ties with respect to the most-constrained-variable heuristic), select the one that appears in the largest number of constraints on variables not in the current assignment

[Rationale: Increase future elimination of values, to reduce future branching factors]

Map Coloring



- Before any value has been assigned, all variables have a domain of size 3, but SA is involved in more constraints (5) than any other variable

→ Select SA and assign a value to it (e.g., Blue)

Least-Constraining-Value Heuristic

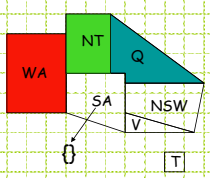
2) In which order should X's values be assigned?

Select the value of X that removes the smallest number of values from the domains of those variables which are not in the current assignment

[Rationale: Since only one value will eventually be assigned to X, pick the least-constraining value first, since it is the most likely not to lead to an invalid assignment]

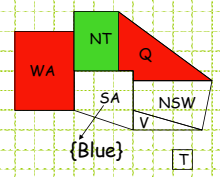
[Note: Using this heuristic requires performing a forward-checking step for every value, not just for the selected value]

Map Coloring



- Q's domain has two remaining values: Blue and Red
- Assigning Blue to Q would leave 0 value for SA, while assigning Red would leave 1 value

Map Coloring



- Q's domain has two remaining values: Blue and Red
- Assigning Blue to Q would leave 0 value for SA, while assigning Red would leave 1 value

→ So, assign Red to Q

Modified Backtracking Algorithm

CSP-BACKTRACKING(A, var-domains)

1. If assignment A is complete then return A
2. X ← select a variable not in A
3. D ← select an ordering on the domain of X
4. For each value v in D do
   a. Add (X←v) to A
   b. var-domains ← forward checking(var-domains, X, v, A)
   c. If a variable has an empty domain then return failure
   d. result ← CSP-BACKTRACKING(A, var-domains)
   e. If result ≠ failure then return result
   f. Remove (X←v) from A
5. Return failure

1) Most-constrained-variable heuristic
2) Most-constraining-variable heuristic

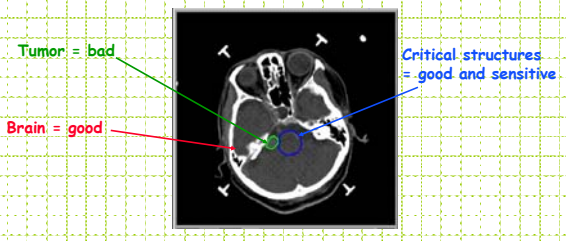
3) Least-constraining-value heuristic

Applications of CSP

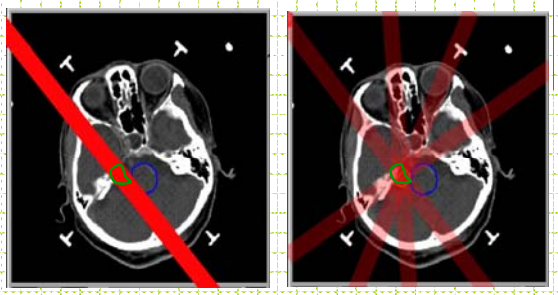
- CSP techniques are widely used
- Applications include:
  - Crew assignments to flights
  - Management of transportation fleet
  - Flight/rail schedules
  - Job shop scheduling
  - Task scheduling in port operations
  - Design, including spatial layout design
  - Radiosurgical procedures

Radiosurgery

Minimally invasive procedure that uses a beam of radiation as an ablative surgical instrument to destroy tumors

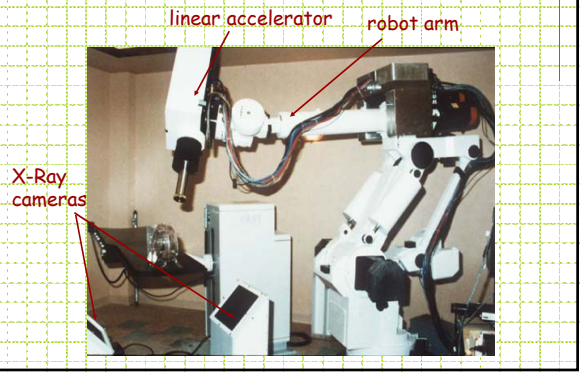


Problem



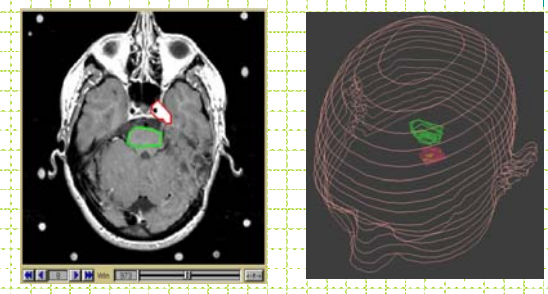
Burn tumor without damaging healthy tissue

The CyberKnife



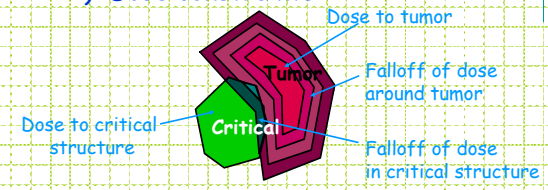
Inputs

1) Regions of interest

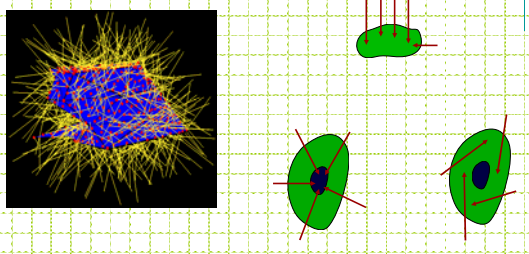


Inputs

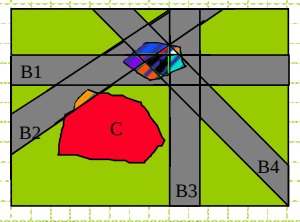
2) Dose constraints



Beam Sampling



Constraints



- 2000 ≤ Tumor ≤ 2200
  - 2000 ≤ B2 + B4 ≤ 2200
  - 2000 ≤ B4 ≤ 2200
  - 2000 ≤ B3 + B4 ≤ 2200
  - 2000 ≤ B1 + B3 + B4 ≤ 2200
  - 2000 ≤ B1 + B4 ≤ 2200
  - 2000 ≤ B1 + B2 + B4 ≤ 2200
  - 2000 ≤ B1 ≤ 2200
  - 2000 ≤ B1 + B2 ≤ 2200
- 0 ≤ Critical ≤ 500
  - 0 ≤ B2 ≤ 500

2000 < Tumor < 2200

2000 < B2 + B4 < 2200
2000 < B4 < 2200
2000 < B3 + B4 < 2200
2000 < B1 + B3 + B4 < 2200
2000 < B1 + B4 < 2200
2000 < B1 + B2 + B4 < 2200
2000 < B1 < 2200
2000 < B1 + B2 < 2200

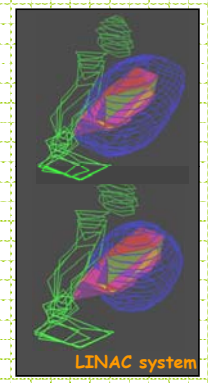
2000 < Tumor < 2200

2000 < B4

B1 + B3 + B4 < 2200

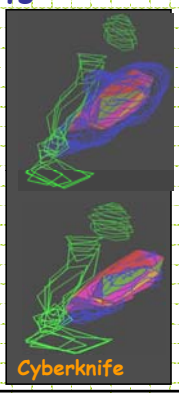
B1 + B2 + B4 < 2200
2000 < B1

Case Results



50% Isodose Surface

80% Isodose Surface



LINAC system

Cyberknife

THE POWER OF T⁴ TECHNOLOGY

Cyberknife® Tight-to-the-Tumor (T⁴) Radiosurgery with Ultimate Conformality