

Solving Constraint Satisfaction Problems: Search and Forward Checking

Brian C. Williams
16.410-13
Session 11

Slides draw from material from:
6.034 notes, by Tomas Lozano Perez
AIMA, by Stuart Russell & Peter Norvig
Constraint Processing, by Rina Dechter

1

Reading Assignments: Constraint Satisfaction

Readings:

- Lecture Slides (most material in slides only, READ ALL).
- AIMA Ch. 5 – Constraint Satisfaction Problems (CSPs)
- For more detail: Ch. 5 of Constraint Processing, - Dechter.

Problem Set #5:

- Covers search methods for solving constraints.
- Online.
- Out today, October 19th.
- Due next Wednesday, October 26th.
- Get started early!

Mid-Term:

- Monday, October 31st.
- No problem Set that week.
- Sample Mid-term, distributed and posted.

2

To Solve CSPs we combine arc consistency and search

1. Arc consistency (Constraint propagation),
 - Eliminates values that are shown locally to not be a part of any solution.
2. Search
 - Explores consequences of committing to particular assignments.

Methods That Incorporate Search:

- Standard Search
- Back Track Search (BT)
- BT with Forward Checking (FC)
- Dynamic Variable Ordering (DV) (Appendix)
- Iterative Repair (IR) (Appendix)
- Back Jumping (BJ) (Appendix)

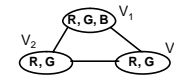
3

Solving CSPs with Standard Search

- State
 - Variable assignments thus far
- Initial State
 - No assignments
- Operator
 - New assignment =
 - Select any unassigned variable
 - Select any one of its domain values
 - Child extend assignments with new
- Goal Test
 - All variables are assigned
 - All constraints are satisfied

Branching factor?

→ Sum of domain size of all variables $O(v \cdot d)$

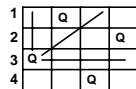


Performance?

→ Exponential of branching factor $O([v \cdot d]^n)$

4

Search Performance on N Queens



- Standard Search
- A handful of queens
- Backtracking

5

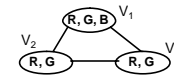
Solving CSPs with Standard Search

Standard Search:

- Children select any value to any variable $[O(v \cdot d)]$
- Test complete assignments against CSP

Observations:

1. The order in which variables are assigned does not change the solution.
 - Many paths denote the same solution (n!).
 - so expand only one path (i.e., one variable ordering).
2. We can identify a dead end before assigning all variables
 - Extensions to inconsistent partial assignments are always inconsistent
 - So check after each assignment.



6

BackTrack Search (BT)

1. Expand the assignments of only one variable at each step.
2. Pursue depth first.
3. Check consistency after each expansion, and backup.

V₁ assignments

V₂ assignments

V₃ assignments

Preselect order of variables to assign

Expand designated variable

7

BackTrack Search (BT)

1. Expand the assignments of only one variable at each step.
2. Pursue depth first.
3. Check consistency after each expansion, and backup.

V₁ assignments

V₂ assignments

V₃ assignments

Preselect order of variables to assign

Assign designated variable

Backup at inconsistent assignment

8

Procedure Backtracking($\langle x, D, C \rangle$)

Input: A constraint network $R = \langle X, D, C \rangle$
Output: A solution, or notification the network is inconsistent.

```

i ? 1; ai = {}           (initialize variable counter, assignments)
Di ? Di                (copy domain of first variable)
while 1 = i = n
  instantiate xi ? Select-Value(); (add to assignments, making ai)
  if xi is null                (no value was returned)
    i ? i - 1;                  (backtrack)
  else
    i ? i + 1;                  (step forward)
    Di ? Di
  end while
if i = 0
  return "inconsistent"
else
  return ai, the instantiated values of {x1, ..., xn}
end procedure
  
```

9

Procedure Select-Value()

Output: A value in D_i consistent with \bar{a}_{i-1} , or null, if none.

```

while Di is not empty
  select an arbitrary element a ∈ Di and remove a from Di;
  if consistent(āi-1, xi = a)
    return a;
end while
return null
end procedure
  
```

10

Search Performance on N Queens

| | | | |
|---|---|---|---|
| 1 | | q | |
| 2 | | | q |
| 3 | q | | |
| 4 | | | q |

- Standard Search
- A handful of queens
- Backtracking
- About 15 queens
- BT with Forward Checking

11

Combine Backtracking and Limited Constraint Propagation

Initially: Prune domains using constraint propagation (optional)

Loop:

- If complete consistent assignment, then return it, Else...
- Choose unassigned variable
- Choose assignment from its pruned domain
- Prune (some) domains using revise (arc-consistency)
- if a domain has no remaining elements, then backtrack.

Question: Full propagation is $O(ed^3)$,
 How much propagation should we do?

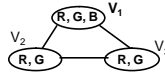
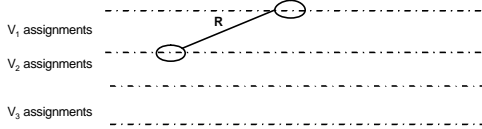
Very little: (except big problems)

- Just check arc consistency for those arcs that terminate on the new assignment [$O(ed)$].
- called **forward checking (FC)**.

12

Backtracking with Forward Checking (BT-FC)

2. After selecting each assignment, remove any values of neighboring domains that are inconsistent with the new assignment.

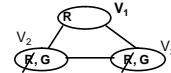
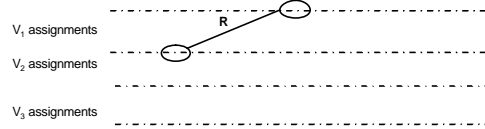


1. Perform initial pruning.

13

Backtracking with Forward Checking (BT-FC)

2. After selecting each assignment, remove any values of neighboring domains that are inconsistent with the new assignment.

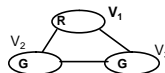
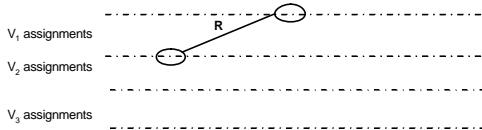


1. Perform initial pruning.

14

Backtracking with Forward Checking (BT-FC)

2. After selecting each assignment, remove any values of neighboring domains that are inconsistent with the new assignment.

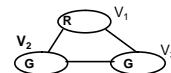
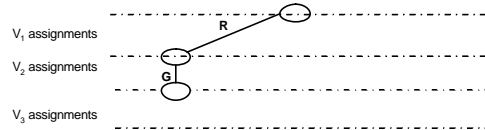


1. Perform initial pruning.

15

Backtracking with Forward Checking (BT-FC)

1. After selecting each assignment, remove any values of neighboring domains that are inconsistent with the new assignment.



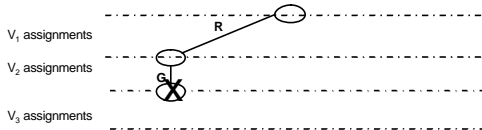
Note: No need to check new assignment against previous assignments

1. Perform initial pruning.

16

Backtracking with Forward Checking (BT-FC)

2. After selecting each assignment, remove any values of neighboring domains that are inconsistent with the new assignment.



3. We have a conflict whenever a domain becomes empty.

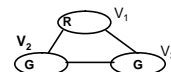
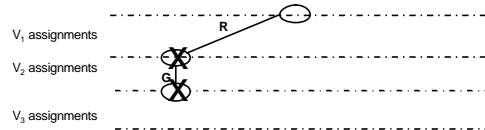
- Back track

1. Perform initial pruning.

17

Backtracking with Forward Checking (BT-FC)

2. After selecting each assignment, remove any values of neighboring domains that are inconsistent with the new assignment.



3. We have a conflict whenever a domain becomes empty.

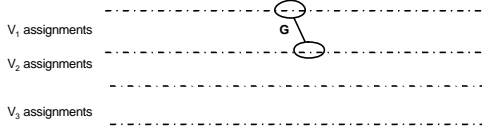
- Back track

1. Perform initial pruning.

18

Backtracking with Forward Checking (BT-FC)

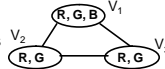
2. After selecting each assignment, remove any values of neighboring domains that are inconsistent with the new assignment.



3. We have a conflict whenever a domain becomes empty.

• Back track

• Restore domain values

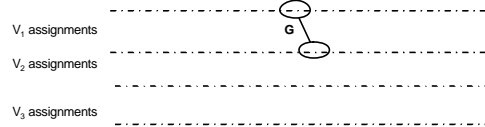


1. Perform initial pruning.

19

Backtracking with Forward Checking (BT-FC)

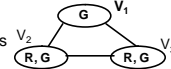
2. After selecting each assignment, remove any values of neighboring domains that are inconsistent with the new assignment.



3. We have a conflict whenever a domain becomes empty.

• Back track

• Restore domain values

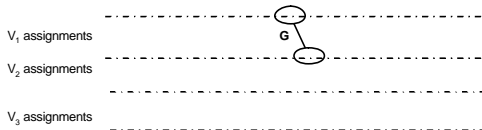


1. Perform initial pruning.

20

Backtracking with Forward Checking (BT-FC)

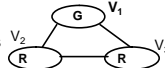
2. After selecting each assignment, remove any values of neighboring domains that are inconsistent with the new assignment.



3. We have a conflict whenever a domain becomes empty.

• Back track

• Restore domain values

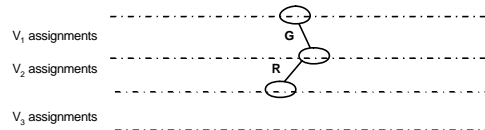


1. Perform initial pruning.

21

Backtracking with Forward Checking (BT-FC)

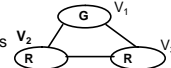
2. After selecting each assignment, remove any values of neighboring domains that are inconsistent with the new assignment.



3. We have a conflict whenever a domain becomes empty.

• Back track

• Restore domain values

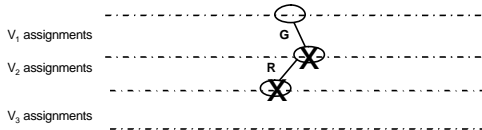


1. Perform initial pruning.

22

Backtracking with Forward Checking (BT-FC)

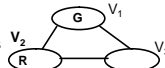
2. After selecting each assignment, remove any values of neighboring domains that are inconsistent with the new assignment.



3. We have a conflict whenever a domain becomes empty.

• Back track

• Restore domain values

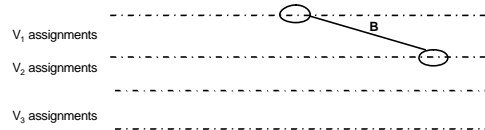


1. Perform initial pruning.

23

Backtracking with Forward Checking (BT-FC)

2. After selecting each assignment, remove any values of neighboring domains that are inconsistent with the new assignment.



3. We have a conflict whenever a domain becomes empty.

• Back track

• Restore domain values



1. Perform initial pruning.

24

Backtracking with Forward Checking (BT-FC)

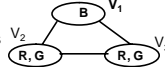
2. After selecting each assignment, remove any values of neighboring domains that are inconsistent with the new assignment.



3. We have a conflict whenever a domain becomes empty.

- Back track

- Restore domain values

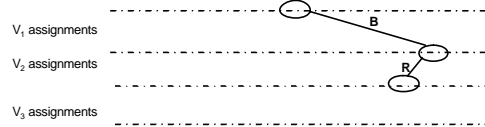


1. Perform initial pruning.

25

Backtracking with Forward Checking (BT-FC)

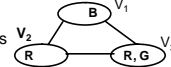
2. After selecting each assignment, remove any values of neighboring domains that are inconsistent with the new assignment.



3. We have a conflict whenever a domain becomes empty.

- Back track

- Restore domain values

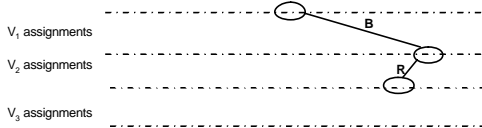


1. Perform initial pruning.

26

Backtracking with Forward Checking (BT-FC)

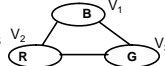
2. After selecting each assignment, remove any values of neighboring domains that are inconsistent with the new assignment.



3. We have a conflict whenever a domain becomes empty.

- Back track

- Restore domain values

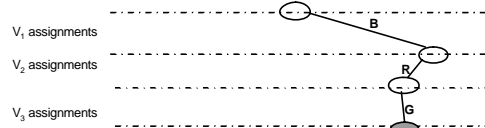


1. Perform initial pruning.

27

Backtracking with Forward Checking (BT-FC)

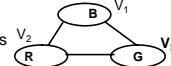
2. After selecting each assignment, remove any values of neighboring domains that are inconsistent with the new assignment.



3. We have a conflict whenever a domain becomes empty.

- Back track

- Restore domain values



1. Perform initial pruning.

Solution!

28

Backtracking with Forward Checking (BT-FC)

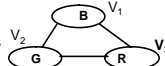
2. After selecting each assignment, remove any values of neighboring domains that are inconsistent with the new assignment.



3. We have a conflict whenever a domain becomes empty.

- Back track

- Restore domain values



1. Perform initial pruning.

BT-FC is generally faster than pure BT because it avoids rediscovering inconsistencies.

29

Procedure Backtrack-Forward-Checking($\langle X, D, C \rangle$)

Input: A constraint network $R = \langle X, D, C \rangle$

Output: A solution, or notification the network is inconsistent.

```

 $D_i \leftarrow D_i$  for  $1 = i = n$ ;           (copy all domains)
 $i \leftarrow 1$ ;  $a_i = \{ \}$            (init variable counter, assignments)
while  $1 = i = n$ 
  instantiate  $x_i$ ? Select-Value-FC(); (add to assignments, making  $\bar{a}_i$ )
  if  $x_i$  is null (no value was returned)
    reset each  $D_k$  for  $k > i$ , to its value before  $x_i$  was last instantiated;
     $i \leftarrow i - 1$ ; (backtrack)
  else
     $i \leftarrow i + 1$ ; (step forward)
end while
if  $i = 0$ 
  return "inconsistent"
else
  return  $\bar{a}_i$ , the instantiated values of  $\{x_1, \dots, x_n\}$ 
end procedure
    
```

30

Procedure Select-Value-FC()

Output: A value in D_i consistent with a_{i-1} , or null, if none.

```

while  $D_i$  is not empty
  select an arbitrary element  $a \in D_i$  and remove  $a$  from  $D_i$ ;
  for all  $k, i < k = n$ 
    for all values  $b$  in  $D_k$ 
      if not consistent( $a_{i-1}, x_i = a, x_k = b$ )
        remove  $b$  from  $D_k$ ;
    end for
  if  $D_k = \text{is empty}$  ( $x_i = a$  leads to a dead-end, don't select)
    reset each  $D_k, i < k = n$  to its value before  $a$  was selected;
  else
    return  $a$ ;
end while
return null
end procedure
    
```

31

Search Performance on N Queens

| | | | | |
|---|---|---|---|---|
| 1 | | Q | | |
| 2 | | | | Q |
| 3 | Q | | | |
| 4 | | | Q | |

- Standard Search
- Backtracking
- Backjumping
- BT with Forward Checking
- Iterative Repair
- Back Jumping
- A handful of queens
- About 15 queens
- ???
- About 30 queens
- About 1,000 queens
- (appendix)
- (appendix)

32

To Solve CSPs we combine arc consistency and search

1. Inference (Arc consistency via Constraint propagation),
 - Eliminates values that are shown locally to not be a part of any solution.
2. Search
 - Explores consequences of committing to particular assignments.

Methods That Incorporate Search:

- Standard Search
- Back Track search (BT)
- Backjumping (BJ)
- BT with Forward Checking (FC)
- Dynamic Variable Ordering (DV)
- Iterative Repair

33

Appendix

- Dynamic Variable Ordering
- Iterative Repair
- Back Jumping

34

Search Performance on N Queens

| | | | | |
|---|---|---|---|---|
| 1 | | Q | | |
| 2 | | | | Q |
| 3 | Q | | | |
| 4 | | | Q | |

- Standard Search
- Backtracking
- BT with Forward Checking
- Dynamic Variable Ordering
- A handful of queens
- About 15 queens
- About 30 queens

35

BT-FC with dynamic ordering

Traditional backtracking uses fixed ordering of variables & values

Typically better to choose ordering dynamically as search proceeds.

- Most constrained variable
 - when doing forward-checking, pick variable with fewest legal values in domain to assign next
 - ⇒ minimizes branching factor
- Least constraining value
 - choose value that rules out the smallest number of values in variables connected to the chosen variable by constraints.
 - ⇒ Leaves most options to find satisfying assignment.

36

Colors: R, G, B, Y

Which country should we color next — E most-constrained variable (smallest domain)

What color should we pick for it? — RED least-constraining value (eliminates fewest values from neighboring domains)

37

Procedure Dynamic-Var-Forward-Checking($\langle X, D, C \rangle$)

Input: A constraint network $R = \langle X, D, C \rangle$
Output: A solution, or notification the network is inconsistent.

```

D_i ? D_i for 1 = i = n;           (copy all domains)
i ? 1;  a_i = {}                  (init variable counter, assignments)
s = min_{x_i in D_i} |D_i|         (find future variable w smallest domain)
while 1 = i = n
  instantiate x_i? Select-Value-FC(); (add to assignments, making a_i)
  if x_i is null                    (no value was returned)
    reset each D_k for k > i, to its value before x_i was last instantiated;
    i ? i - 1;                       (backtrack)
  else
    if i < n
      i ? i + 1;                       (step forward to x_i)
      s = min_{x_{i+1} in D_{i+1}} |D_{i+1}| (find future variable w smallest domain)
      x_{i+1} ? x_i                    (rearrange variables so that x_i follows x_i)
    else
      i ? i + 1;                       (step forward to x_n)
  end while
if i = 0
  return "inconsistent"
else
  return a_i, the instantiated values of (x_1, ..., x_n)
end procedure

```

38

Search Performance on N Queens

- Standard Search
- Backtracking
- Backjumping
- BT with Forward Checking
- Dynamic Variable Ordering
- Iterative Repair
- Back Jumping

- A handful of queens
- About 15 queens
- ???
- About 30 queens
- About 1,000 queens

39

Incremental Repair (min-conflict heuristic)

- Initialize a candidate solution using "greedy" heuristic – get solution "near" correct one.
- Select a variable in conflict and assign it a value that minimizes the number of conflicts (break ties randomly).

Heuristic used in a local hill-climber (without or with backup).

| | | | | |
|---------|-----|-----|-----|-----|
| R R R:3 | BRR | GRR | RGR | RRG |
| | | | | |
| | | | | |
| | | | | |

40

Min-conflict heuristic

Pure hill climber (w/o backtracking) gets stuck in local minima:

- Add random moves to attempt to get out of minima – generally quite effective.
- Add weights on violated constraints & increase weight every cycle the constraint remains violated.

Performance on n-queens. (with good initial guesses)

GSAT: Randomized hill climber used to solve propositional logic SATisfiability problems.

41

Search Performance on N Queens

- Standard Search
- Backtracking
- BT with Forward Checking
- Dynamic Variable Ordering
- Iterative Repair
- Back Jumping

- A handful of queens
- About 15 queens
- About 30 queens
- About 1,000 queens
- About 10,000,000 queens (except truly hard problems)

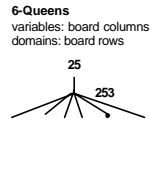
42

Back Jumping

Backtracking At dead end backup to most recent variable,

Backjumping At dead end backup to most recent variable that eliminated a value in a current domain that is empty.

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | | 1 | | | 3 | 2 |
| 2 | Q | 1 | 1 | 1 | 1 | 1 |
| 3 | | 1 | Q | 2 | 3 | 3 |
| 4 | | | 1 | 3 | | |
| 5 | Q | 2 | 1 | 2 | 2 | |
| 6 | | 2 | 2 | 1 | 1 | 3 |
| | 1 | 2 | 3 | 4 | 5 | 6 |



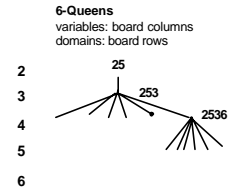
43

Back jumping

Backtracking At dead end backup to most recent variable,

Backjumping At dead end backup to most recent variable that eliminated a value in the current (empty) domain.

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | | 1 | | | 3 | 2 |
| 2 | Q | 1 | 1 | 1 | 1 | 1 |
| 3 | 4 | 1 | Q | 2 | 3 | 3 |
| 4 | | 4 | 1 | 3 | | 4 |
| 5 | Q | 2 | 1 | 2 | 2 | |
| 6 | | 2 | 2 | Q | 1 | 3 |
| | 1 | 2 | 3 | 4 | 5 | 6 |



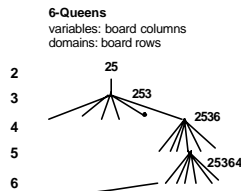
44

Back jumping

Backtracking At dead end backup to most recent variable,

Backjumping At dead end backup to most recent variable that eliminated a value in the current (empty) domain.

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | | 1 | | | 3 | 2 |
| 2 | Q | 1 | 1 | 1 | 1 | 1 |
| 3 | 4 | 1 | Q | 2 | 3 | 3 |
| 4 | | 4 | 1 | 3 | Q | 4 |
| 5 | Q | 2 | 1 | 2 | 2 | |
| 6 | | 2 | 2 | Q | 1 | 3 |
| | 1 | 2 | 3 | 4 | 5 | 6 |



Why does this work?

One instance of a powerful suite of methods that "diagnosis" the cause of conflicting assignments, and act to resolve the conflict. (Conflict-directed Search Methods).

Failures here should look to variable 4. Changing variable 5 won't help

45