Constraint satisfaction search.  
Combinatorial optimization search.

Milos Hauskrecht  
milos@cs.pitt.edu  
5329 Sennott Square

Constraint satisfaction problem (CSP)

Objective:
• Find a configuration satisfying goal conditions

• Constraint satisfaction problem (CSP) is a configuration search problem where:
  – A state is defined by a set of variables and their values
  – Goal condition is represented by a set constraints on possible variable values
CSP example: N-queens

Goal: n queens placed in non-attacking positions on the board

Variables:
• Represent queens, one for each column:
  – $Q_1, Q_2, Q_3, Q_4$
• Values:
  – Row placement of each queen on the board
    \{1, 2, 3, 4\}

Constraints:
$Q_i \neq Q_j$ Two queens not in the same row
$|Q_i - Q_j| \neq |i - j|$ Two queens not on the same diagonal

Map coloring

Color a map using k different colors such that no adjacent countries have the same color

Variables:
• Represent countries
  – $A, B, C, D, E$
• Values:
  – K -different colors
    \{Red, Blue, Green,..\}

Constraints: $A \neq B, A \neq C, C \neq E$, etc
An example of a problem with binary constraints
Constraint satisfaction as a search problem

Formulation of a CSP as a search problem:

• **States.** Assignment (partial, complete) of values to variables.
• **Initial state.** No variable is assigned a value.
• **Operators.** Assign a value to one of the unassigned variables.
• **Goal condition.** All variables are assigned, no constraints are violated.

• **Constraints** can be represented:
  – **Explicitly** by a set of allowable values
  – **Implicitly** by a function that tests for the satisfaction of constraints

Solving a CSP through standard search

• **Maximum depth of the tree (m):** ?
• **Depth of the solution (d) :** ?
• **Branching factor (b) :** ?
Solving a CSP through standard search

- **Maximum depth of the tree**: Number of variables in the CSP
- **Depth of the solution**: Number of variables in the CSP
- **Branching factor**: If we fix the order of variable assignments, the branch factor depends on the number of their values

```
Unassigned: Q1, Q2, Q3, Q4
Assigned:

Unassigned: Q2, Q3, Q4
Assigned: Q1 = 1

Unassigned: Q2, Q3, Q4
Assigned: Q1 = 2

Unassigned: Q1, Q4
Assigned: Q1 = 2, Q2 = 4
```

Solving a CSP through standard search

- **What search algorithm to use**: ?

  Depth of the tree = Depth of the solution = number of vars

```
Unassigned: Q1, Q2, Q3, Q4
Assigned:

Unassigned: Q2, Q3, Q4
Assigned: Q1 = 1

Unassigned: Q2, Q3, Q4
Assigned: Q1 = 2

Unassigned: Q1, Q4
Assigned: Q1 = 2, Q2 = 4

... ...
```
Solving a CSP through standard search

- What search algorithm to use: **Depth first search !!!**
  - Since we know the depth of the solution
  - We do not have to keep large number of nodes in queues

Constraint consistency

**Assuring consistency of constraints:**
- Current variable assignments together with constraints restrict remaining legal values of unassigned variables
  - The remaining legal and illegal values of variables may be inferred (effect of constraints propagates)
  - To prevent “blind” exploration we can keep track of the remaining legal values, so we know when the constraints are violated and when to terminate the search
**Constraint propagation**

- Assign $A=\text{Red}$

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Blue</th>
<th>Green</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$B$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$E$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$F$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Assignments (equations)
- Invalid assignments (disequations)
Constraint propagation

- Assign E=Blue

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Blue</th>
<th>Green</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>✗</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td>✗</td>
<td></td>
</tr>
</tbody>
</table>

Constraint propagation

- Assign E=Blue

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Blue</th>
<th>Green</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>✓</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>✗</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td>✗</td>
<td></td>
</tr>
</tbody>
</table>
Constraint propagation

- Assign F = Green

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Blue</th>
<th>Green</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>✓</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>F</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Three techniques for propagating the effects of past assignments and constraints:

- **Node propagation**
- **Arc consistency**
- **Forward checking**

**Difference:**
- Completeness of inferences
- Time complexity of inferences.
Constraint propagation

1. Node consistency. Infers:
   - equations (valid assignments) or disequations (invalid assignments) for an individual variable by applying a unary constraint

2. Arc consistency. Infers:
   - disequations from the set of equations and disequations defining the partial assignment, and a constraint
   - equations through the exhaustion of alternatives

3. Forward checking. Infers:
   - disequations from a set of equations defining the partial assignment, and a constraint
   - Equations through the exhaustion of alternatives

Restricted forward checking:
   - uses only active constraints (active constraint – only one variable unassigned in the constraint)

Example

Map coloring of Australia territories

Note: problem with binary constraints
Example: node consistency

Map coloring

Assume a constraint:
WA ≠ Green

Infer: invalid assignments from WA ≠ Green constraint

vars | WA | NT | Q | NSW | V | SA | T
--- | --- | --- | --- | --- | --- | --- | ---
domain | R G B | R G B | R G B | R G B | R G B | R G B | R G B
Example: forward checking

Map coloring

Set: WA=Red

<table>
<thead>
<tr>
<th>vars</th>
<th>WA</th>
<th>NT</th>
<th>Q</th>
<th>NSW</th>
<th>V</th>
<th>SA</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
</tr>
</tbody>
</table>

Infer: invalid assignments from WA=Red + constraints
### Example: forward checking

**Map coloring**

Set: Q=Green

<table>
<thead>
<tr>
<th>vars</th>
<th>WA</th>
<th>NT</th>
<th>Q</th>
<th>NSW</th>
<th>V</th>
<th>SA</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>domain</strong></td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
</tr>
<tr>
<td>WA=Red</td>
<td>R</td>
<td>G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>G B</td>
<td>R G B</td>
</tr>
</tbody>
</table>

---

### Example: forward checking

**Map coloring**

Infer: invalid assignments from Q=Green + constraints

<table>
<thead>
<tr>
<th>vars</th>
<th>WA</th>
<th>NT</th>
<th>Q</th>
<th>NSW</th>
<th>V</th>
<th>SA</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>domain</strong></td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
</tr>
<tr>
<td>WA=Red</td>
<td>R</td>
<td>G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>G B</td>
<td>R G B</td>
</tr>
<tr>
<td>Q=Green</td>
<td>R</td>
<td>B</td>
<td>G</td>
<td>R B</td>
<td>R G B</td>
<td>B</td>
<td>R G B</td>
</tr>
</tbody>
</table>
Example: forward checking

Map coloring

Infer: NT=B
Exhaustions of alternatives

<table>
<thead>
<tr>
<th>vars</th>
<th>WA</th>
<th>NT</th>
<th>Q</th>
<th>NSW</th>
<th>V</th>
<th>SA</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
</tr>
<tr>
<td>WA=Red</td>
<td>R</td>
<td>G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>G B</td>
<td>R G B</td>
</tr>
<tr>
<td>Q=Green</td>
<td>R</td>
<td>B</td>
<td>G</td>
<td>R B</td>
<td>R G B</td>
<td>B</td>
<td>R G B</td>
</tr>
<tr>
<td>Infer NT</td>
<td>R</td>
<td>B</td>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Infer: invalid assignments from NT=B

<table>
<thead>
<tr>
<th>vars</th>
<th>WA</th>
<th>NT</th>
<th>Q</th>
<th>NSW</th>
<th>V</th>
<th>SA</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
</tr>
<tr>
<td>WA=Red</td>
<td>R</td>
<td>G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>G B</td>
<td>R G B</td>
</tr>
<tr>
<td>Q=Green</td>
<td>R</td>
<td>B</td>
<td>G</td>
<td>R B</td>
<td>R G B</td>
<td>B</td>
<td>R G B</td>
</tr>
<tr>
<td>Infer NT</td>
<td>R</td>
<td>B</td>
<td>G</td>
<td>R B</td>
<td>R G B</td>
<td></td>
<td>R G B</td>
</tr>
</tbody>
</table>
Example: arc consistency

Map coloring

Set: WA=Red
Set: Q=Green

<table>
<thead>
<tr>
<th>vars</th>
<th>WA</th>
<th>NT</th>
<th>Q</th>
<th>NSW</th>
<th>V</th>
<th>SA</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
</tr>
<tr>
<td>WA=Red</td>
<td>R</td>
<td>G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>G B</td>
<td>R G B</td>
</tr>
<tr>
<td>Q=Green</td>
<td>R</td>
<td>B</td>
<td>G</td>
<td>R B</td>
<td>R G B</td>
<td>B</td>
<td>R G B</td>
</tr>
</tbody>
</table>

Infer: invalid assignments from valid and invalid assignments

<table>
<thead>
<tr>
<th>vars</th>
<th>WA</th>
<th>NT</th>
<th>Q</th>
<th>NSW</th>
<th>V</th>
<th>SA</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
</tr>
<tr>
<td>WA=Red</td>
<td>R</td>
<td>G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>G B</td>
<td>R G B</td>
</tr>
<tr>
<td>Q=Green</td>
<td>R</td>
<td>B</td>
<td>G</td>
<td>R B</td>
<td>R G B</td>
<td>B</td>
<td>R G B</td>
</tr>
</tbody>
</table>
Example: arc consistency

Map coloring

Infer: invalid assignments from valid and invalid assignments

<table>
<thead>
<tr>
<th>vars</th>
<th>WA</th>
<th>NT</th>
<th>Q</th>
<th>NSW</th>
<th>V</th>
<th>SA</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
</tr>
<tr>
<td>WA=Red</td>
<td>R</td>
<td>G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>G B</td>
<td>R G B</td>
</tr>
<tr>
<td>Q=Green</td>
<td>R</td>
<td>B</td>
<td>G</td>
<td>R B</td>
<td>R G B</td>
<td>B</td>
<td>R G B</td>
</tr>
</tbody>
</table>

Infer: invalid assignments from valid and invalid assignments

<table>
<thead>
<tr>
<th>vars</th>
<th>WA</th>
<th>NT</th>
<th>Q</th>
<th>NSW</th>
<th>V</th>
<th>SA</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
</tr>
<tr>
<td>NSW=R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SA=B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Consistent assignment
### Example: arc consistency

#### Map coloring

![Map coloring diagram]

**Infer:** invalid assignments from valid and invalid assignments

<table>
<thead>
<tr>
<th>vars</th>
<th>WA</th>
<th>NT</th>
<th>Q</th>
<th>NSW</th>
<th>V</th>
<th>SA</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
</tr>
<tr>
<td>WA=Red</td>
<td>R</td>
<td>G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
</tr>
<tr>
<td>Q=Green</td>
<td>R</td>
<td>B</td>
<td>G</td>
<td>R B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
</tr>
</tbody>
</table>

---

### Example: arc consistency

#### Map coloring

![Map coloring diagram]

**Infer:** invalid assignments from valid and invalid assignments

<table>
<thead>
<tr>
<th>vars</th>
<th>WA</th>
<th>NT</th>
<th>Q</th>
<th>NSW</th>
<th>V</th>
<th>SA</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
</tr>
<tr>
<td>WA=Red</td>
<td>R</td>
<td>G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
</tr>
<tr>
<td>Q=Green</td>
<td>R</td>
<td>B</td>
<td>G</td>
<td>R B</td>
<td>R G B</td>
<td>R G B</td>
<td>R G B</td>
</tr>
</tbody>
</table>
Heuristics for CSPs

CSP searches the space in the depth-first manner. But we still can choose:
• Which variable to assign next?
• Which value to choose first?

Heuristics
• Most constrained variable
  – Which variable is likely to become a bottleneck?
• Least constraining value
  – Which value gives us more flexibility later?

Example: map coloring

Heuristics
• Most constrained variable
  – ?
• Least constraining value
  – ?
Heuristics for CSP

Examples: map coloring

Heuristics

• Most constrained variable
  – Country E is the most constrained one (cannot use Red, Green)

• Least constraining value
  – Assume we have chosen variable C
  – What color is the least constraining color?
Heuristics for CSP

Examples: **map coloring**

**Heuristics**

- **Most constrained variable**
  - Country E is the most constrained one (cannot use Red, Green)

- **Least constraining value**
  - Assume we have chosen variable C
  - Red is the least constraining valid color for the future

---

**Finding optimal configurations**
Search for the optimal configuration

Constrain satisfaction problem:
Objective: find a configuration that satisfies all constraints

Optimal configuration problem:
Objective: find the best configuration
The quality of a configuration: is defined by some quality measure that reflects our preference towards each configuration (or state)

Our goal: optimize the configuration according to the quality measure also referred to as objective function

Search for the optimal configuration

Optimal configuration search:
• Configurations are described in terms of variables and their values
• Each configuration has a quality measure
• Goal: find the configuration with the best value

If the space of configurations we search among is
• Discrete or finite
  – then it is a combinatorial optimization problem
• Continuous
  – then it is a parametric optimization problem
Example: Traveling salesman problem

Problem:
- A graph with distances
- A tour – a path that visits every city once and returns to the start e.g. ABCDEF
- **Goal**: find the shortest tour

Example: N queens

- A CSP problem
- Is it possible to formulate the problem as an optimal configuration search problem?
Example: N queens

- A CSP problem
- Is it possible to formulate the problem as an optimal configuration search problem? Yes.
- The quality of a configuration in a CSP can be measured by the number of violated constraints
- Solving: minimize the number of constraint violations

![Images of chessboards with different numbers of violations]

# of violations = 3  # of violations = 1  # of violations = 0