Constraint Satisfaction

UMaine COS 470/570 – Introduction to AI

Spring 2019

Search

Search so far...

- Uninformed search – nothing known about state space
- Heuristic search – *something* known, at least – defeasible
- Both: searching for a state with little internal structure
- Many problems: state has internal structure
- Important class of problems: state is assignment of *values* to *variables*

Artificial Intelligence

Examples

- Cryptarithmetic: Assign 0–9 uniquely to letters so that a symbolic expression is valid

```
 SEND
+MORE
-----
MONEY
```

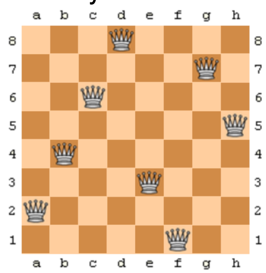

Artificial Intelligence

Examples

- Cryptarithmetic: Assign 0–9 uniquely to letters so that a symbolic expression is valid

```
 SEND
+MORE
-----
MONEY
```

- N-queens problem: Place n queens on an $n \times n$ chessboard so that they don't attack one another

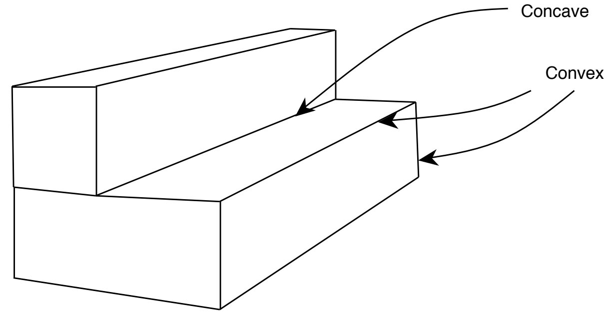


(From okpanico.files.wordpress.com)

Examples

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- Computer vision: Classify edges in an image as convex or concave



Copyright © 2017 UMaine School of Computing and Information Science

Examples

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- Solving simultaneous equations

$$\begin{aligned} 3x + 4y + 6z &= 3 \\ 4x + 6y - 3z &= 4 \\ 7x - 3y - 4z &= 10 \end{aligned}$$

Copyright © 2017 UMaine School of Computing and Information Science

Examples

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- Solving simultaneous equations

$$\begin{aligned} 3x + 4y + 6z &= 3 \\ 4x + 6y - 3z &= 4 \\ 7x - 3y - 4z &= 10 \end{aligned}$$

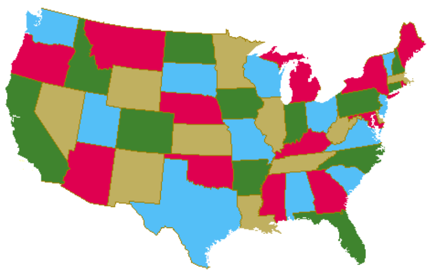
- BSAT: Is a sum-of-products binary expression satisfiable, and if so, with what T/F assignments?

$$ABC + \overline{A}\,\overline{B}\,C + \cdots + ABC\overline{D}$$

Copyright © 2017 UMaine School of Computing and Information Science

Examples

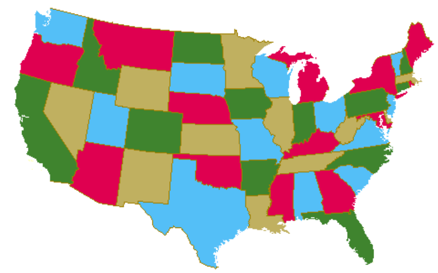
- Map coloring: Can we color a map of connected regions with n colors without two adjacent regions having the same color?



(From people.math.gatech.edu/~thomas)

Examples

- Map coloring: Can we color a map of connected regions with n colors without two adjacent regions having the same color?



(From people.math.gatech.edu/~thomas)

- Scheduling: Scheduling a meeting with n people

Constraints

- To solve: could use blind or heuristic search

Constraints

- To solve: could use blind or heuristic search
- But:
  - Often very large search spaces:

Constraints

- To solve: could use blind or heuristic search
- But:
  - Often very large search spaces:
    - v variables, d values $\Rightarrow \mathcal{O}(d^v)$

Constraints

- To solve: could use blind or heuristic search
- But:
  - Often very large search spaces:
    - v variables, d values $\Rightarrow \mathcal{O}(d^v)$
    - E.g., BSAT with 40 variables $\Rightarrow \mathcal{O}(2^{40})$

Constraints

- To solve: could use blind or heuristic search
- But:
  - Often very large search spaces:
    - v variables, d values $\Rightarrow \mathcal{O}(d^v)$
    - E.g., BSAT with 40 variables $\Rightarrow \mathcal{O}(2^{40})$
    - E.g., Map coloring continental US w/ 4 colors $\Rightarrow$ $\mathcal{O}(4^{48}) = \mathcal{O}(2^{96}) = \mathcal{O}(10^{28})$

Constraints

- To solve: could use blind or heuristic search
- But:
  - Often very large search spaces:
    - v variables, d values $\Rightarrow \mathcal{O}(d^v)$
    - E.g., BSAT with 40 variables $\Rightarrow \mathcal{O}(2^{40})$
    - E.g., Map coloring continental US w/ 4 colors $\Rightarrow$ $\mathcal{O}(4^{48}) = \mathcal{O}(2^{96}) = \mathcal{O}(10^{28})$
  - Often selecting a value for one variable *constrains* the values another can have

Constraints

- To solve: could use blind or heuristic search
- But:
  - Often very large search spaces:
    - v variables, d values $\Rightarrow \mathcal{O}(d^v)$
    - E.g., BSAT with 40 variables $\Rightarrow \mathcal{O}(2^{40})$
    - E.g., Map coloring continental US w/ 4 colors $\Rightarrow \mathcal{O}(4^{48}) = \mathcal{O}(2^{96}) = \mathcal{O}(10^{28})$
  - Often selecting a value for one variable *constrains* the values another can have

Artificial Intelligence

Constraints

Constraint Satisfaction

Search

Constraint satisfaction problems

CSP formalism

Constraint Propagation

Special cases

Miscellaneous

Application: Task assignment in CoDA

- To solve: could use blind or heuristic search
- But:
  - Often very large search spaces:
    - v variables, d values $\Rightarrow \mathcal{O}(d^v)$
    - E.g., BSAT with 40 variables $\Rightarrow \mathcal{O}(2^{40})$
    - E.g., Map coloring continental US w/ 4 colors $\Rightarrow \mathcal{O}(4^{48}) = \mathcal{O}(2^{96}) = \mathcal{O}(10^{28})$
  - Often selecting a value for one variable *constrains* the values another can have
- Better approach:

Copyright © 2017 UMaine School of Computing and Information Science

Artificial Intelligence

Constraints

Constraint Satisfaction

Search

Constraint satisfaction problems

CSP formalism

Constraint Propagation

Special cases

Miscellaneous

Application: Task assignment in CoDA

- To solve: could use blind or heuristic search
- But:
  - Often very large search spaces:
    - v variables, d values $\Rightarrow \mathcal{O}(d^v)$
    - E.g., BSAT with 40 variables $\Rightarrow \mathcal{O}(2^{40})$
    - E.g., Map coloring continental US w/ 4 colors $\Rightarrow \mathcal{O}(4^{48}) = \mathcal{O}(2^{96}) = \mathcal{O}(10^{28})$
  - Often selecting a value for one variable *constrains* the values another can have
- Better approach:
  - Explicitly recognize *constraints* between variables
  - Make use of constraints to guide search

Copyright © 2017 UMaine School of Computing and Information Science

Artificial Intelligence

Constraints

Constraint Satisfaction

Search

Constraint satisfaction problems

CSP formalism

Constraint Propagation

Special cases

Miscellaneous

Application: Task assignment in CoDA

- To solve: could use blind or heuristic search
- But:
  - Often very large search spaces:
    - v variables, d values $\Rightarrow \mathcal{O}(d^v)$
    - E.g., BSAT with 40 variables $\Rightarrow \mathcal{O}(2^{40})$
    - E.g., Map coloring continental US w/ 4 colors $\Rightarrow \mathcal{O}(4^{48}) = \mathcal{O}(2^{96}) = \mathcal{O}(10^{28})$
  - Often selecting a value for one variable *constrains* the values another can have
- Better approach:
  - Explicitly recognize *constraints* between variables
  - Make use of constraints to guide search
- Constraints can focus search: concentrate where variables constrain each other (e.g.)

Copyright © 2017 UMaine School of Computing and Information Science

Artificial Intelligence

Constraints

- To solve: could use blind or heuristic search
- But:
  - Often very large search spaces:
    - v variables, d values $\Rightarrow \mathcal{O}(d^v)$
    - E.g., BSAT with 40 variables $\Rightarrow \mathcal{O}(2^{40})$
    - E.g., Map coloring continental US w/ 4 colors $\Rightarrow \mathcal{O}(4^{48}) = \mathcal{O}(2^{96}) = \mathcal{O}(10^{28})$
  - Often selecting a value for one variable *constrains* the values another can have
- Better approach:
  - Explicitly recognize *constraints* between variables
  - Make use of constraints to guide search
- Constraints can focus search: concentrate where variables constrain each other (e.g.)
- Sometimes: radically reduce search effort

Constraint satisfaction problems

Constraint-satisfaction problems

- *Constraint satisfaction problems* (CSPs):
  - Require set of variables to be bound to values $\in$ domain
  - Require constraints to be *satisfied*
- Instead of trying all possible variable/value assignments via search...
- *Propagate* constraints and values $\Rightarrow$ reduce domains of variables
- $\mathcal{O}(v^d)$ in w.c.: try $\Rightarrow \mathcal{O}(v^{d'})$, $d' \ll d$ in average case
- Fox, others: All problems can be reformulated $\Rightarrow$ CSPs

Constraints

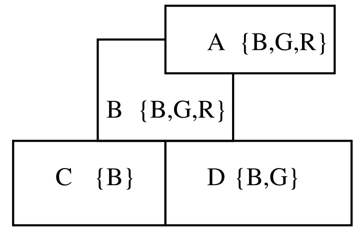
- Types by arity:
  - *Unary constraints:* constraint on single value
  - *Binary, ternary, n-ary constraints:* restrict value of variable depending on value of other variable(s)
  - All $n - ary$ constraints can be $\Rightarrow$ binary constraints
- Types by whether absolute or preference constraints

Constraint propagation

- Suppose we reduce domain of v_1 that constrains v_2
- Some values in v_2 might now be eliminated
- Thus decision at v_1 *propagates* via the constraint to v_2
- Propagation continues from v_2, etc.
- May ultimately change v_1 again
- Stop when no more changes occur
- More constrained the values $\rightarrow$ faster to a solution

An Example of constraint propagation

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA



Map-coloring Problem

- Cannot color adjacent areas with the same color
- Some areas may have unary constraints which limit their domains
- Eliminate impossible assignments by propagating constraints

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

CSP formalism

Constraint satisfaction problem

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- *Constraint graph*

Constraint satisfaction problem

- *Constraint graph*
  - Nodes = variables

Constraint satisfaction problem

- *Constraint graph*
  - Nodes = variables
  - Arcs = constraints

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Constraint satisfaction problem

- *Constraint graph*
  - Nodes = variables
  - Arcs = constraints
- *Domain* for each variable

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Constraint satisfaction problem

- *Constraint graph*
  - Nodes = variables
  - Arcs = constraints
- *Domain* for each variable
- One possible Constraint representation: *intensionally* – e.g., $v_1 \neq v_2$

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Constraint satisfaction problem

- *Constraint graph*
  - Nodes = variables
  - Arcs = constraints
- *Domain* for each variable
- One possible Constraint representation: *intensionally* – e.g., $v_1 \neq v_2$
- Easier (for finite domains):

Constraint satisfaction problem

Constraint Satisfaction

Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- *Constraint graph*
  - Nodes = variables
  - Arcs = constraints
- *Domain* for each variable
- One possible Constraint representation: *intensionally* – e.g., $v_1 \neq v_2$
- Easier (for finite domains):

  - *Extensionally* – list values that satisfy constraint

Copyright © 2017 UMaine School of Computing and Information Science

Constraint satisfaction problem

Constraint Satisfaction

Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- *Constraint graph*
  - Nodes = variables
  - Arcs = constraints
- *Domain* for each variable
- One possible Constraint representation: *intensionally* – e.g., $v_1 \neq v_2$
- Easier (for finite domains):
  - *Extensionally* – list values that satisfy constraint

  - I.e., *positive constraints*

Copyright © 2017 UMaine School of Computing and Information Science

Constraint satisfaction problem

Constraint Satisfaction

Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- *Constraint graph*
  - Nodes = variables
  - Arcs = constraints
- *Domain* for each variable
- One possible Constraint representation: *intensionally* – e.g., $v_1 \neq v_2$
- Easier (for finite domains):
  - *Extensionally* – list values that satisfy constraint
  - I.e., *positive constraints*

- Constraint $C = \{(d_1, d_2) \mid d_1 \in \mathrm{dom}(v_1)\ \&\ d_2 \in \mathrm{dom}(v_2)\}$

Copyright © 2017 UMaine School of Computing and Information Science

Constraint satisfaction problem

- *Constraint graph*
  - Nodes = variables
  - Arcs = constraints
- *Domain* for each variable
- One possible Constraint representation: *intensionally* – e.g., $v_1 \neq v_2$
- Easier (for finite domains):
  - *Extensionally* – list values that satisfy constraint
  - I.e., *positive constraints*
- Constraint $C = \{(d_1, d_2) \mid d_1 \in \text{dom}(v_1) \,\&\, d_2 \in \text{dom}(v_2)\}$
- Goal: All variables instantiated, no violated constraints

Artificial Intelligence

What is a state?

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- State representation 1: *complete assignments*
  - Start: Graph + random assignments
  - Operator: Change variable's value
  - Goal: All constraints satisfied
  - *Generate and test* search:
    - Set variable, check for goal
    - No guidance on which variable, value to choose
    - Quickly intractable: $n!d^n$ leaves

Artificial Intelligence

What is a state?

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- State representation 1: *complete assignments*
  - Start: Graph + random assignments
  - Operator: Change variable's value
  - Goal: All constraints satisfied
  - *Generate and test* search:
    - Set variable, check for goal
    - No guidance on which variable, value to choose
    - Quickly intractable: $n!d^n$ leaves
    - E.g.: For 4-coloring of 48 states: ~10^{90}
- State representation 2: *partial assignments*
  - State: Graph + domains – singleton = assignment
  - Operator: Make assignment
  - After each assignment: propagate constraints
  - Goal: all singleton domains
  - Encounter empty domain: backtrack
  - Systematically explore space by choosing how vars instantiated

Artificial Intelligence

Constraint Propagation

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Constraint propagation

- After value selected, propagate effects using constraints
- Propagation $\Rightarrow$ narrowing of domains to be consistent
- Two types of consistency:
  - Node consistency
  - Arc consistency

Node consistency

Constraint Satisfaction

- Check unary constraints
- Pre-processing, $\mathcal{O}(n)$ step
- E.g.:
  - Map-coloring problem: $\forall\, v\ \mathrm{dom}(v) = \{red, green, blue, yellow\}$
  - Texans object to blue, $\mathrm{dom}(\mathrm{Texas}) = \{red, green, yellow\}$
- All unary constraints satisfied: graph is *node-consistent*
- Unary constraints reduce |domain| $\Rightarrow$ prunes search tree, $\downarrow$ branching factor

Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Arc Consistency

Constraint Satisfaction

- Eliminate any constraint violations

Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Arc Consistency

Constraint Satisfaction

- Eliminate any constraint violations
- Pairwise checking of constraints, propagation of changes

Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Arc Consistency

- Eliminate any constraint violations
- Pairwise checking of constraints, propagation of changes
- Delete values from domain of variable if they are not consistent with all constraints on the variable

Arc Consistency

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- Eliminate any constraint violations
- Pairwise checking of constraints, propagation of changes
- Delete values from domain of variable if they are not consistent with all constraints on the variable
- What does "consistent" mean?

Copyright © 2017 UMaine School of Computing and Information Science

Arc Consistency

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- Eliminate any constraint violations
- Pairwise checking of constraints, propagation of changes
- Delete values from domain of variable if they are not consistent with all constraints on the variable
- What does "consistent" mean?
  - Let v_1, v_2 be variables connected by constraint c

Copyright © 2017 UMaine School of Computing and Information Science

Arc Consistency

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- Eliminate any constraint violations
- Pairwise checking of constraints, propagation of changes
- Delete values from domain of variable if they are not consistent with all constraints on the variable
- What does "consistent" mean?
  - Let v_1, v_2 be variables connected by constraint c
  - Value $y \in \text{dom}(v_2)$ is consistent with c iff $\exists x \in \text{dom}(v_1)\ \&\ (x, y) \in c$

Copyright © 2017 UMaine School of Computing and Information Science

Arc Consistency

- Eliminate any constraint violations
- Pairwise checking of constraints, propagation of changes
- Delete values from domain of variable if they are not consistent with all constraints on the variable
- What does "consistent" mean?
  - Let v_1, v_2 be variables connected by constraint c
  - Value $y \in \text{dom}(v_2)$ is consistent with c iff $\exists x \in \text{dom}(v_1)$ & $(x, y) \in c$
- *Forward checking:*

Arc Consistency

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- Eliminate any constraint violations
- Pairwise checking of constraints, propagation of changes
- Delete values from domain of variable if they are not consistent with all constraints on the variable
- What does "consistent" mean?
  - Let v_1, v_2 be variables connected by constraint c
  - Value $y \in \text{dom}(v_2)$ is consistent with c iff $\exists x \in \text{dom}(v_1)$ & $(x, y) \in c$
- *Forward checking:*
  - Special case of arc consistency

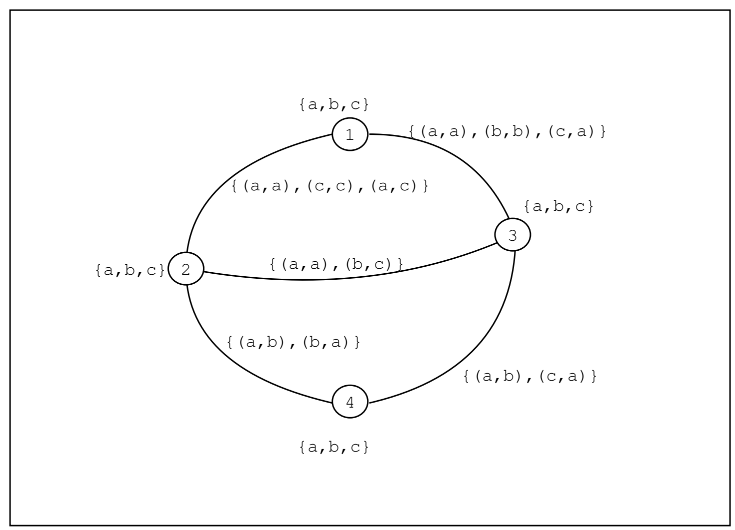
Arc Consistency

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- Eliminate any constraint violations
- Pairwise checking of constraints, propagation of changes
- Delete values from domain of variable if they are not consistent with all constraints on the variable
- What does "consistent" mean?
  - Let v_1, v_2 be variables connected by constraint c
  - Value $y \in \text{dom}(v_2)$ is consistent with c iff $\exists x \in \text{dom}(v_1)$ & $(x, y) \in c$
- *Forward checking:*
  - Special case of arc consistency
  - Initiated when variable assigned value

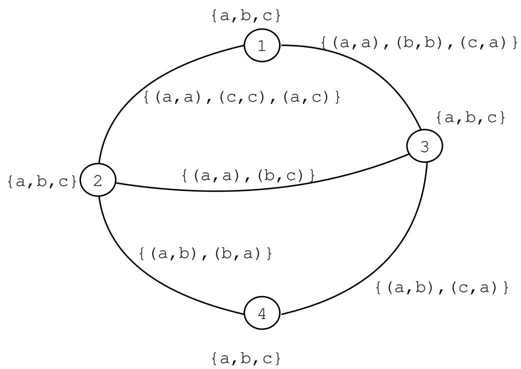
Example

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA



Example

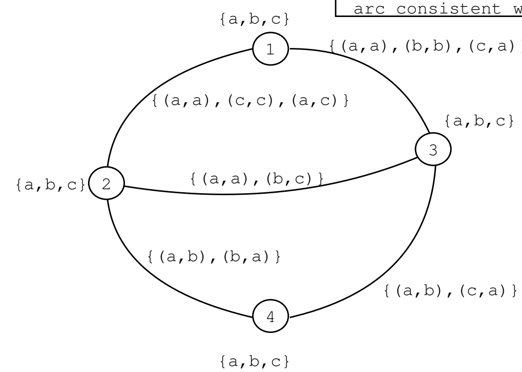

```
Tuples listed so that lower-numbered
variable is to left.
```



Artificial Intelligence

Example

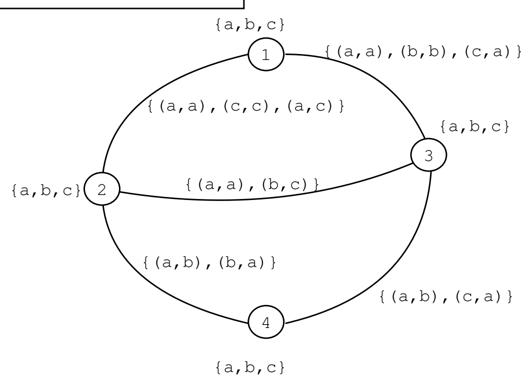

```
Look at 1 & 3: 1 is
arc consistent with 3
```



Artificial Intelligence

Example

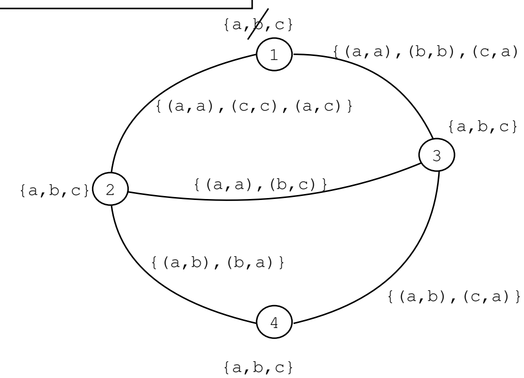

```
Look at 1 & 2:
   No value in 2's domain
   consistent with 1=b.
```



Artificial Intelligence

Example

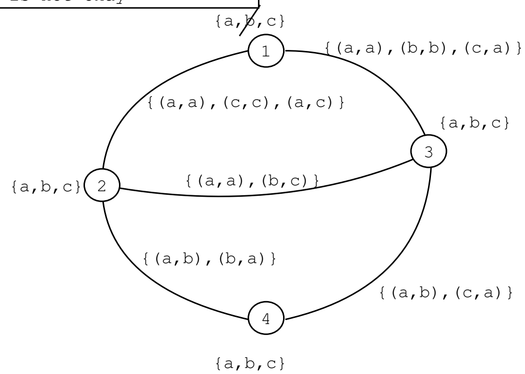

```
Look at 1 & 2:
   No value in 2's domain
   consistent with 1=b.
```



Artificial Intelligence

Example

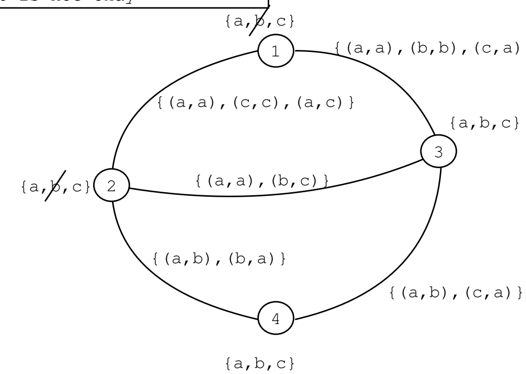

```
Lookat 2 & 1:
  2=a is okay:  (a,a)
  2=c is okay: (c,c) and (a,c)
  2=b is not okay
```



Example

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

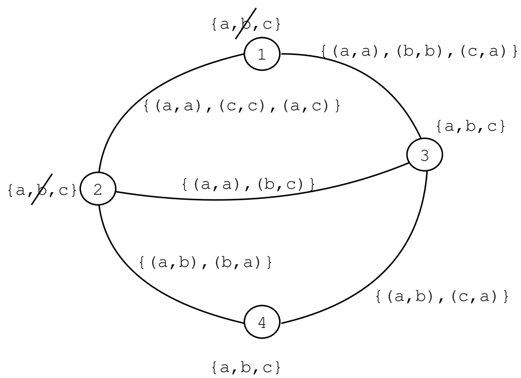
```
Lookat 2 & 1:
  2=a is okay:  (a,a)
  2=c is okay: (c,c) and (a,c)
  2=b is not okay
```



Example

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

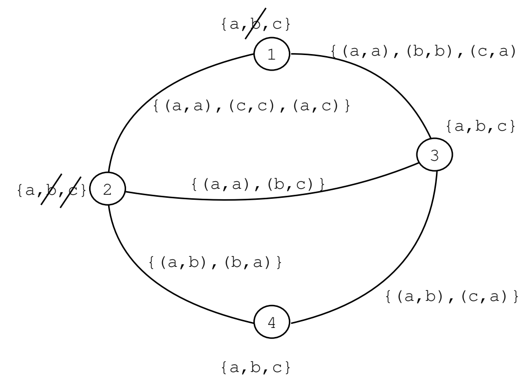
```
Look at 2 & 3:
  2=a is okay
  2=c is not
```



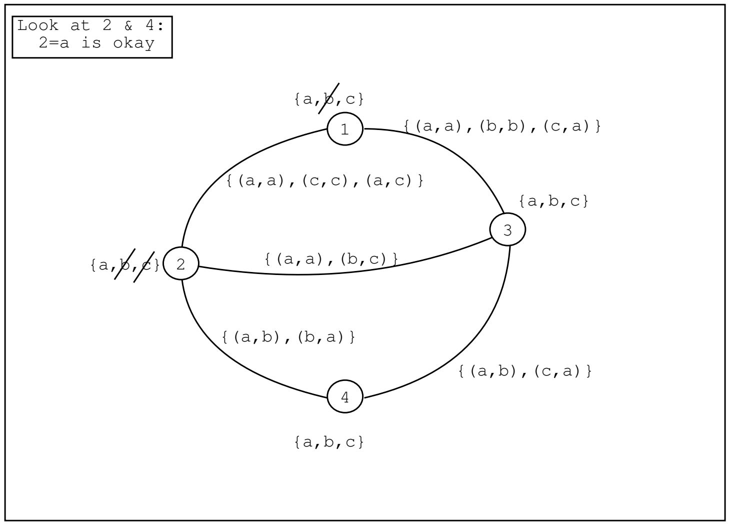
Example

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

```
Look at 2 & 3:
  2=a is okay
  2=c is not
```



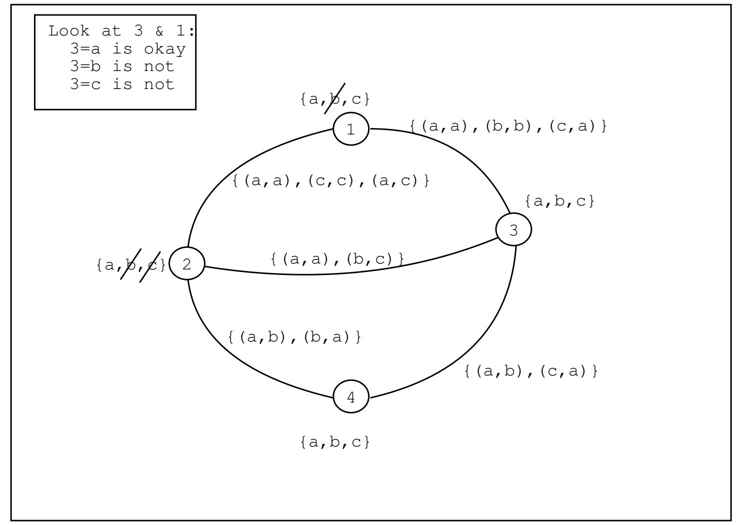
Example



Artificial Intelligence

Example

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

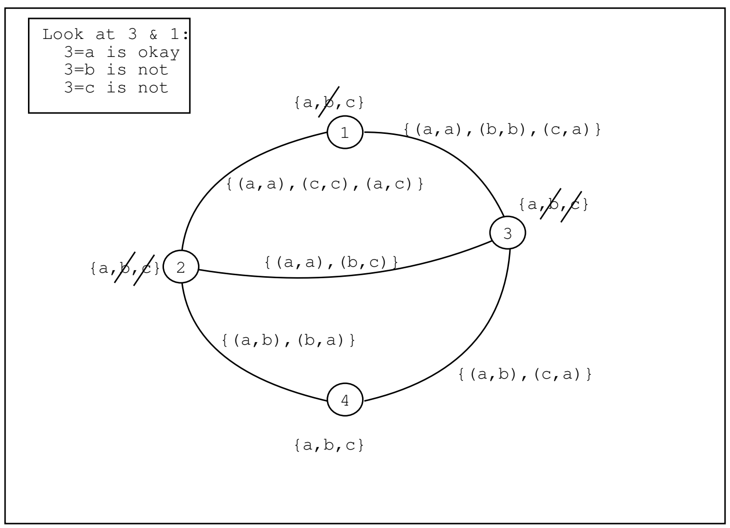


Copyright © 2017 UMaine School of Computing and Information Science

Artificial Intelligence

Example

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

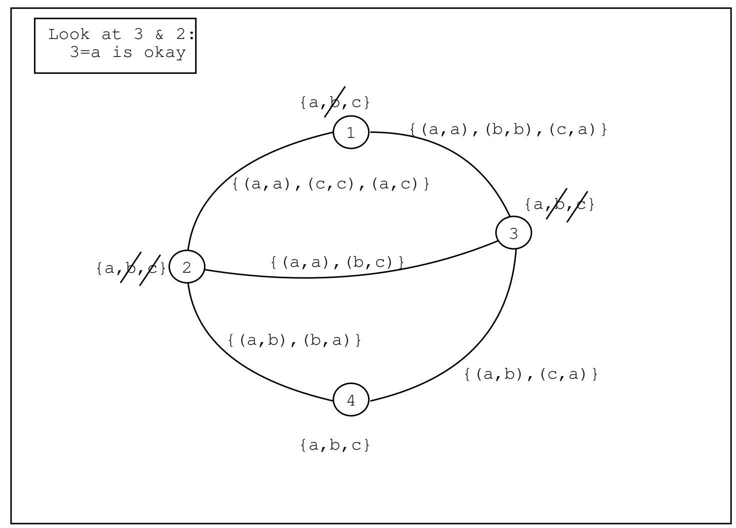


Copyright © 2017 UMaine School of Computing and Information Science

Artificial Intelligence

Example

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

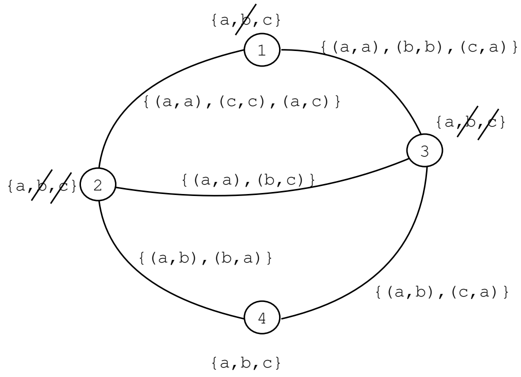


Copyright © 2017 UMaine School of Computing and Information Science

Artificial Intelligence

Example

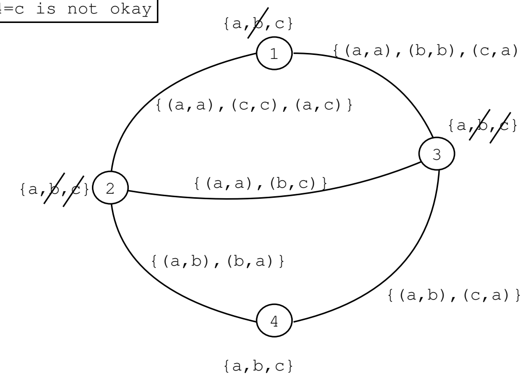
Look at 3 & 4:
3=a is okay



Artificial Intelligence

Example

Look at 4 & 2:
4=a is not okay
4=b is okay
4=c is not okay



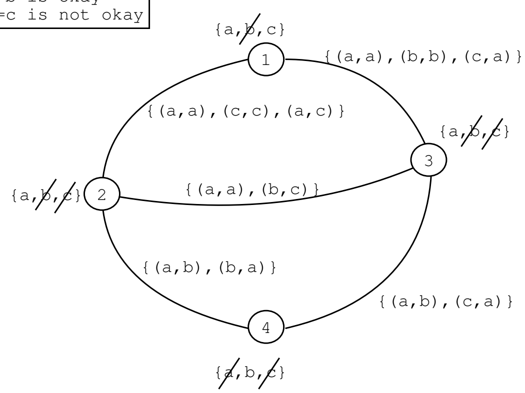
Constraint Satisfaction

Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Artificial Intelligence

Example

Look at 4 & 2:
4=a is not okay
4=b is okay
4=c is not okay



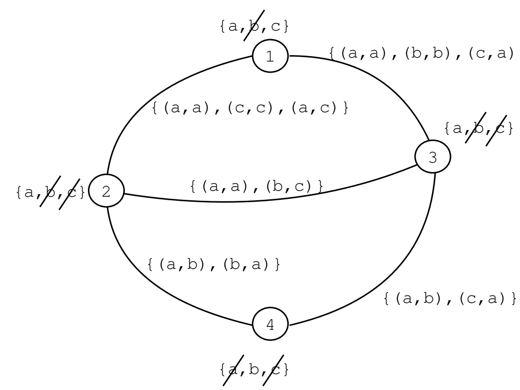
Constraint Satisfaction

Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Artificial Intelligence

Example

Look at 4 & 3:
4=b is okay



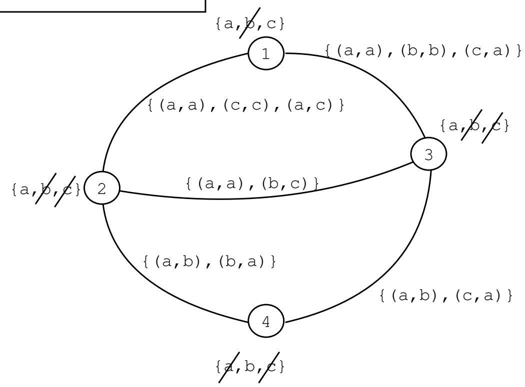
Constraint Satisfaction

Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Artificial Intelligence

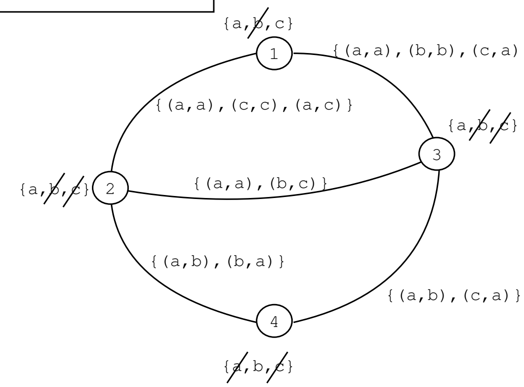
Example

```
At this point: Is CSP
arc-consistent?
```



Example

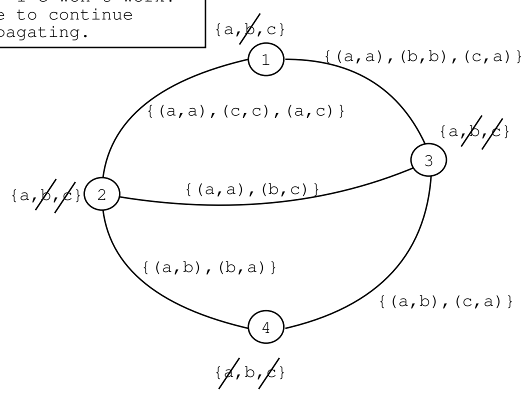
```
At this point: Is CSP
arc-consistent?
No!  1=c won't work!
```



Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Example

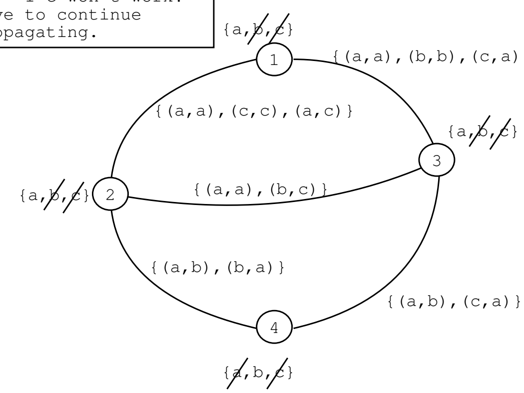
```
At this point: Is CSP
arc-consistent?
No!  1=c won't work!
Have to continue
propagating.
```



Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

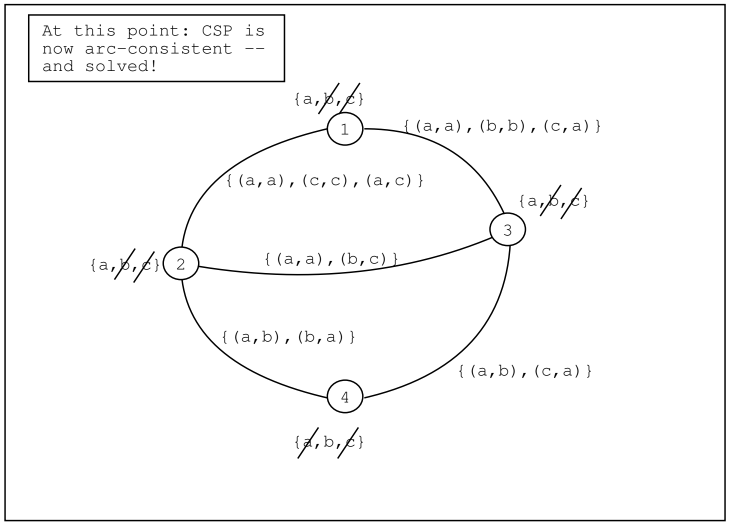
Example

```
At this point: Is CSP
arc-consistent?
No!  1=c won't work!
Have to continue
propagating.
```



Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

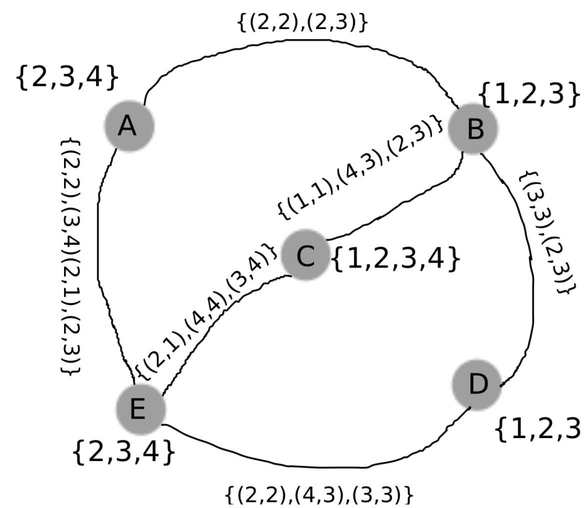
Example



Artificial Intelligence

Your turn

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

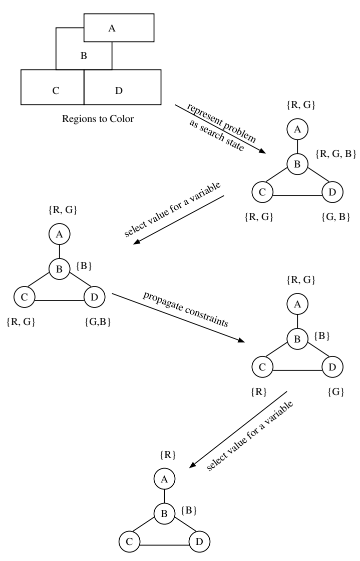


Copyright © 2017 UMaine School of Computing and Information Science

Artificial Intelligence

Example of CSP

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA



Copyright © 2017 UMaine School of Computing and Information Science

Artificial Intelligence

CSP as search

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- Best case: value selection + propagation →→ solution

Copyright © 2017 UMaine School of Computing and Information Science

Artificial Intelligence

CSP as search

- Best case: value selection + propagation →→ solution
- But it's a search process

CSP as search

- Best case: value selection + propagation →→ solution
- But it's a search process :
  - But what if dead-end?

CSP as search

- Best case: value selection + propagation →→ solution
- But it's a search process :
  - But what if dead-end?\ Backtrack

CSP as search

- Best case: value selection + propagation →→ solution
- But it's a search process :
  - But what if dead-end?\ Backtrack
  - And *which* variable, *which* value to pick each choice point?

Heuristics for CSP search

- Which variable to set?

Heuristics for CSP search

Constraint Satisfaction

- Which variable to set?
  - *Most-constrained variable heuristic:*

Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Copyright © 2017 UMaine School of Computing and Information Science

Heuristics for CSP search

Constraint Satisfaction

- Which variable to set?
  - *Most-constrained variable heuristic:*
    - Pick variable with smallest remaining domain

Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Copyright © 2017 UMaine School of Computing and Information Science

Heuristics for CSP search

Constraint Satisfaction

- Which variable to set?
  - *Most-constrained variable heuristic:*
    - Pick variable with smallest remaining domain
    - Reduces branching factor: fewest alternatives to backtrack to

Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Copyright © 2017 UMaine School of Computing and Information Science

Heuristics for CSP search

- Which variable to set?
  - *Most-constrained variable heuristic:*
    - Pick variable with smallest remaining domain
    - Reduces branching factor: fewest alternatives to backtrack to
  - *Most-constraining variable heuristic:*

Heuristics for CSP search

- Which variable to set?
  - *Most-constrained variable heuristic:*
    - Pick variable with smallest remaining domain
    - Reduces branching factor: fewest alternatives to backtrack to
  - *Most-constraining variable heuristic:*
    - Assign variable with most constraints

Heuristics for CSP search

- Which variable to set?
  - *Most-constrained variable heuristic:*
    - Pick variable with smallest remaining domain
    - Reduces branching factor: fewest alternatives to backtrack to
  - *Most-constraining variable heuristic:*
    - Assign variable with most constraints
    - Reduces branching factor by pruning other variables' domains

Heuristics for CSP search

- Which variable to set?
  - *Most-constrained variable heuristic:*
    - Pick variable with smallest remaining domain
    - Reduces branching factor: fewest alternatives to backtrack to
  - *Most-constraining variable heuristic:*
    - Assign variable with most constraints
    - Reduces branching factor by pruning other variables' domains
- Which value to use?

Heuristics for CSP search

- Which variable to set?
  - *Most-constrained variable heuristic:*
    - Pick variable with smallest remaining domain
    - Reduces branching factor: fewest alternatives to backtrack to
  - *Most-constraining variable heuristic:*
    - Assign variable with most constraints
    - Reduces branching factor by pruning other variables' domains
- Which value to use?
  - *Least-constraining value heuristic:*

Heuristics for CSP search

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- Which variable to set?
  - *Most-constrained variable heuristic:*
    - Pick variable with smallest remaining domain
    - Reduces branching factor: fewest alternatives to backtrack to
  - *Most-constraining variable heuristic:*
    - Assign variable with most constraints
    - Reduces branching factor by pruning other variables' domains
- Which value to use?
  - *Least-constraining value heuristic:*
    - Choose value that rules out fewest values from connected variables

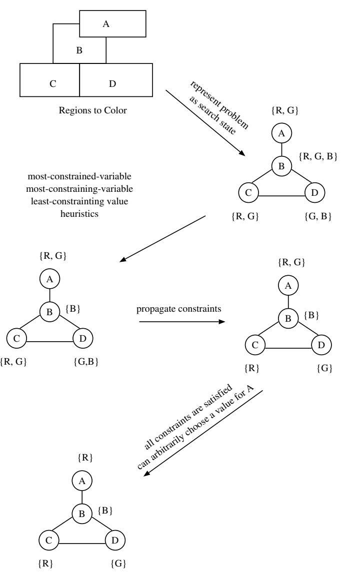
Heuristics for CSP search

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- Which variable to set?
  - *Most-constrained variable heuristic:*
    - Pick variable with smallest remaining domain
    - Reduces branching factor: fewest alternatives to backtrack to
  - *Most-constraining variable heuristic:*
    - Assign variable with most constraints
    - Reduces branching factor by pruning other variables' domains
- Which value to use?
  - *Least-constraining value heuristic:*
    - Choose value that rules out fewest values from connected variables
    - increases likelihood of success

Example

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA



Special cases

Special cases

- *Independent subproblems:*

Special cases

- *Independent subproblems:*
  - Identify connected components of graph, solve separately

Special cases

- *Independent subproblems:*
  - Identify connected components of graph, solve separately
  - Suppose each subproblem has c variables of total n

Special cases

- *Independent subproblems:*
  - Identify connected components of graph, solve separately
  - Suppose each subproblem has c variables of total n
  - Becomes *linear* in n: $\mathcal{O}(n/c \times d^c)$

Special cases

- *Independent subproblems:*
  - Identify connected components of graph, solve separately
  - Suppose each subproblem has c variables of total n
  - Becomes *linear* in n: $\mathcal{O}(n/c \times d^c)$
  - $n = 80, \ d = 2, \ c = 20, 10^7$ nodes/sec: 4 billion years without, 0.4 s with

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Copyright © 2017 UMaine School of Computing and Information Science

Special cases

- *Independent subproblems:*
  - Identify connected components of graph, solve separately
  - Suppose each subproblem has c variables of total n
  - Becomes *linear* in n: $\mathcal{O}(n/c \times d^c)$
  - $n = 80, \ d = 2, \ c = 20, 10^7$ nodes/sec: 4 billion years without, 0.4 s with
- *Acyclic constraint graph:*

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Copyright © 2017 UMaine School of Computing and Information Science

Special cases

- *Independent subproblems:*
  - Identify connected components of graph, solve separately
  - Suppose each subproblem has c variables of total n
  - Becomes *linear* in n: $\mathcal{O}(n/c \times d^c)$
  - $n = 80, \ d = 2, \ c = 20, 10^7$ nodes/sec: 4 billion years without, 0.4 s with
- *Acyclic constraint graph:*
  - Pick root, order nodes parent $\rightarrow$ child

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Copyright © 2017 UMaine School of Computing and Information Science

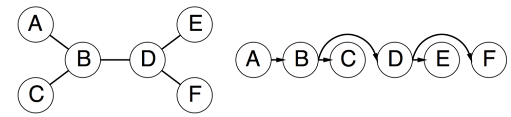
Special cases

- *Independent subproblems:*
  - Identify connected components of graph, solve separately
  - Suppose each subproblem has c variables of total n
  - Becomes *linear* in n: $\mathcal{O}(n/c \times d^c)$
  - $n = 80,\ d = 2,\ c = 20, 10^7$ nodes/sec: 4 billion years without, 0.4 s with
- *Acyclic constraint graph:*
  - Pick root, order nodes parent → child

Artificial Intelligence

Special cases

- *Independent subproblems:*
  - Identify connected components of graph, solve separately
  - Suppose each subproblem has c variables of total n
  - Becomes *linear* in n: $\mathcal{O}(n/c \times d^c)$
  - $n = 80,\ d = 2,\ c = 20, 10^7$ nodes/sec: 4 billion years without, 0.4 s with
- *Acyclic constraint graph:*
  - Pick root, order nodes parent → child



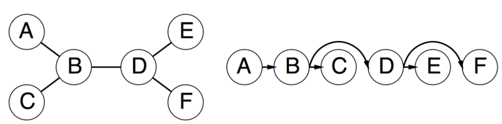
  - From leaves → root, remove inconsistencies between child, parent

(From S. Russell's slides)

Artificial Intelligence

Special cases

- *Independent subproblems:*
  - Identify connected components of graph, solve separately
  - Suppose each subproblem has c variables of total n
  - Becomes *linear* in n: $\mathcal{O}(n/c \times d^c)$
  - $n = 80,\ d = 2,\ c = 20, 10^7$ nodes/sec: 4 billion years without, 0.4 s with
- *Acyclic constraint graph:*
  - Pick root, order nodes parent → child



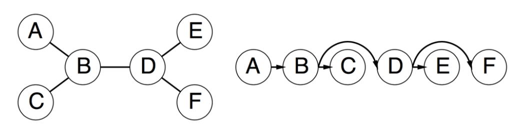
  - From leaves → root, remove inconsistencies between child, parent
  - From root → leaves: pick value consistent w/ parent

(From S. Russell's slides)

Artificial Intelligence

Special cases

- *Independent subproblems:*
  - Identify connected components of graph, solve separately
  - Suppose each subproblem has c variables of total n
  - Becomes *linear* in n: $\mathcal{O}(n/c \times d^c)$
  - $n = 80,\ d = 2,\ c = 20, 10^7$ nodes/sec: 4 billion years without, 0.4 s with
- *Acyclic constraint graph:*
  - Pick root, order nodes parent → child



  - From leaves → root, remove inconsistencies between child, parent
  - From root → leaves: pick value consistent w/ parent
  - $\mathcal{O}(nd^2)$

(From S. Russell's slides)

Artificial Intelligence

Special cases

- *Almost tree-structured:*

(*From S. Russell's slides*)

Special cases

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- *Almost tree-structured:*
  - Instantiate set of variables in all possible ways s.t. remainder is tree-structured

(*From S. Russell's slides*)

Copyright © 2017 UMaine School of Computing and Information Science

Artificial Intelligence

Special cases

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- *Almost tree-structured:*
  - Instantiate set of variables in all possible ways s.t. remainder is tree-structured
  - Take out the *cutset*

(*From S. Russell's slides*)

Copyright © 2017 UMaine School of Computing and Information Science

Artificial Intelligence

Special cases

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- *Almost tree-structured:*
  - Instantiate set of variables in all possible ways s.t. remainder is tree-structured
  - Take out the *cutset*
  - If cutset size c, $\mathcal{O}(d^c \cdot (n-c)d^2)$

(*From S. Russell's slides*)

Copyright © 2017 UMaine School of Computing and Information Science

Artificial Intelligence

Iterative algorithms

- Use hill-climbing, simulated annealing

(From S. Russell's slides)

Iterative algorithms

Constraint Satisfaction

- Use hill-climbing, simulated annealing
- Complete assignment, allow violated constraints

(From S. Russell's slides)

Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Copyright © 2017 UMaine School of Computing and Information Science

Iterative algorithms

Constraint Satisfaction

- Use hill-climbing, simulated annealing
- Complete assignment, allow violated constraints
- Operators: reassign variables

(From S. Russell's slides)

Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Copyright © 2017 UMaine School of Computing and Information Science

Iterative algorithms

Constraint Satisfaction

- Use hill-climbing, simulated annealing
- Complete assignment, allow violated constraints
- Operators: reassign variables
- Select any variable

(From S. Russell's slides)

Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Copyright © 2017 UMaine School of Computing and Information Science

Iterative algorithms

- Use hill-climbing, simulated annealing
- Complete assignment, allow violated constraints
- Operators: reassign variables
- Select any variable
- Value: use *min-conflicts* heuristic – choose state w/ fewest constraints violated

(From S. Russell's slides)

Iterative algorithms

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- Use hill-climbing, simulated annealing
- Complete assignment, allow violated constraints
- Operators: reassign variables
- Select any variable
- Value: use *min-conflicts* heuristic – choose state w/ fewest constraints violated
- How good?

(From S. Russell's slides)

Copyright © 2017 UMaine School of Computing and Information Science

Iterative algorithms

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- Use hill-climbing, simulated annealing
- Complete assignment, allow violated constraints
- Operators: reassign variables
- Select any variable
- Value: use *min-conflicts* heuristic – choose state w/ fewest constraints violated
- How good?
  - Result for (e.g.) n-queens

(From S. Russell's slides)

Copyright © 2017 UMaine School of Computing and Information Science

Iterative algorithms

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- Use hill-climbing, simulated annealing
- Complete assignment, allow violated constraints
- Operators: reassign variables
- Select any variable
- Value: use *min-conflicts* heuristic – choose state w/ fewest constraints violated
- How good?
  - Result for (e.g.) n-queens
  - Can solve in almost $\mathcal{O}(n)$ time with high probability

(From S. Russell's slides)

Copyright © 2017 UMaine School of Computing and Information Science

Iterative algorithms

- Use hill-climbing, simulated annealing
- Complete assignment, allow violated constraints
- Operators: reassign variables
- Select any variable
- Value: use *min-conflicts* heuristic – choose state w/ fewest constraints violated
- How good?
  - Result for (e.g.) *n*-queens
  - Can solve in almost $\mathcal{O}(n)$ time with high probability
  - For almost any number of queens

(From S. Russell's slides)

Miscellaneous

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Continuous variables

- Many real-world problems – e.g., scheduling times for space applications, etc.
- If linear constraints: solvable by *linear programming* in polynomial time

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Copyright © 2017 UMaine School of Computing and Information Science

Constrained Heuristic Search (CHS)

- Can we use even more heuristic information?
- CHS (Fox et al., 1989): Constraint graphs become states in state space search graph
- Operators: assign value, add/delete constraint, constrain domain of variable
- Heuristics: look for *textures* in graph $\Rightarrow$ operator to apply

Constraint Satisfaction
Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

Copyright © 2017 UMaine School of Computing and Information Science

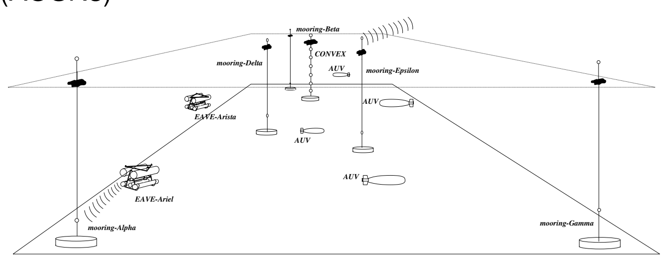
Application: Task assignment in CoDA

CoDA

- Autonomous oceanographic sampling networks (AOSNs)

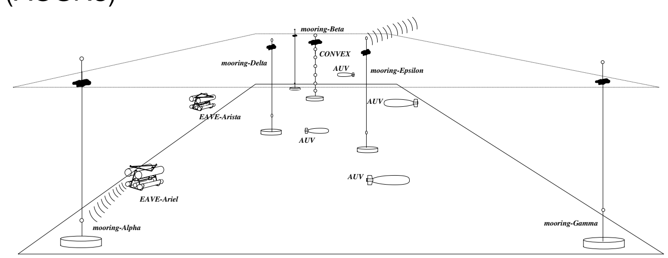
CoDA

- Autonomous oceanographic sampling networks (AOSNs)



CoDA

- Autonomous oceanographic sampling networks (AOSNs)



- Treat as multiagent systems (MAS): CoDA (Turner & Turner)
- Need task assignment: Constraint satisfaction problem
- Use CHS

Overview

- Identify *capabilities*: of AUVs, needed for problem
- Create *task-decomposition tree*

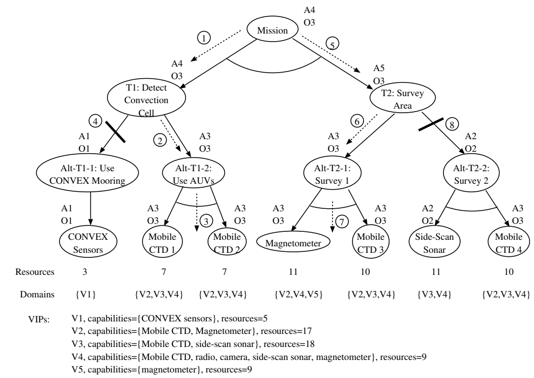
Artificial Intelligence

Overview

Constraint Satisfaction

Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- Identify *capabilities*: of AUVs, needed for problem
- Create *task-decomposition tree*



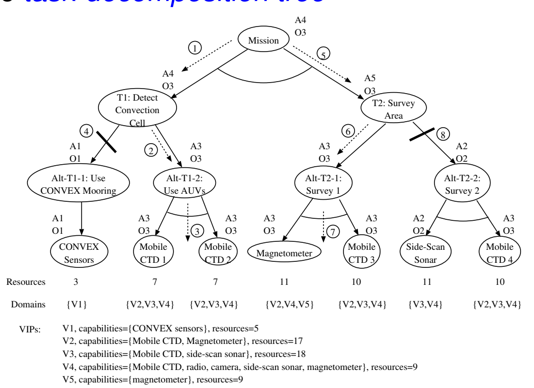
Artificial Intelligence

Overview

Constraint Satisfaction

Search
Constraint satisfaction problems
CSP formalism
Constraint Propagation
Special cases
Miscellaneous
Application: Task assignment in CoDA

- Identify *capabilities*: of AUVs, needed for problem
- Create *task-decomposition tree*



- State: TDT + constraint graph (initially empty)
- Operators: add to constraint graph, set value
- Perform CHS algorithm on constructed constraint graph

Artificial Intelligence