

Features

Features	2
Basics	2
Satisficing Search	3
Constraint Satisfaction Problems	3
Generalized Arc Consistency	4
Example CSP	5
Domain Splitting	6
Optimization Search	7
Local Search	7
Iterative Improvement	8
Local Optima	9
Randomization	10
Genetic Algorithms	11

Features

Basics

- Consider problems where a solution is a single state (Sudoku, schedule).
- Instead of treating each state like a black box, one can reason about the *features* of a state.
- Each feature is mapped to a *variable*, which has a set of possible values, its *domain*.
- A *possible world* is an assignment of values to the variables. A valid possible world is a *model*.
- A *hard constraint* specifies valid combinations of values for, typically, a small subset of the variables.
- A *soft constraint* specifies the costs of different combinations of values.

Satisficing Search

Constraint Satisfaction Problems

- A Constraint Satisfaction Problems (CSP) consist of a set of variables, a domain for each variable, and a set of (hard) constraints.
- One way to solve CSPs is to use *arc consistency*.
 - Each variable X starts with all of its possible values.
 - A possible value x can be eliminated if arc inconsistent with a constraint c .
 - This means no combination of possible values allowed by c includes $X = x$.
- To solve CSPs, combine arc consistency with domain splitting, trying all values of a variable.

Generalized Arc Consistency

Procedure $GAC(V, C, D)$

Inputs: V : a set of variables
 C : a set of constraints
 D_X : a set of values for each variable X

Output: an arc-consistent D

repeat until no more changes are made

 for each constraint $c \in C$

 for each variable X used by c

 for each value $x \in D_X$

 if $X = x$ is not valid for c and D

 remove x from D_X

 return D

CS 3793/5233 Artificial Intelligence

Features – 4

Domain Splitting

Procedure $DS(V, C, D)$

Inputs: V : a set of variables
 C : a set of constraints
 D_X : a set of values for each variable X

Output: null or a solution D'

$D \leftarrow GAC(V, C, D)$

if some $D_X = \emptyset$ return null

if every D_X has one value return D

$X \leftarrow$ some variable where $|D_X| > 1$

for each value $x \in X$

$D' \leftarrow DS(V, C, \text{copy of } D \text{ with } D'_X = \{x\})$

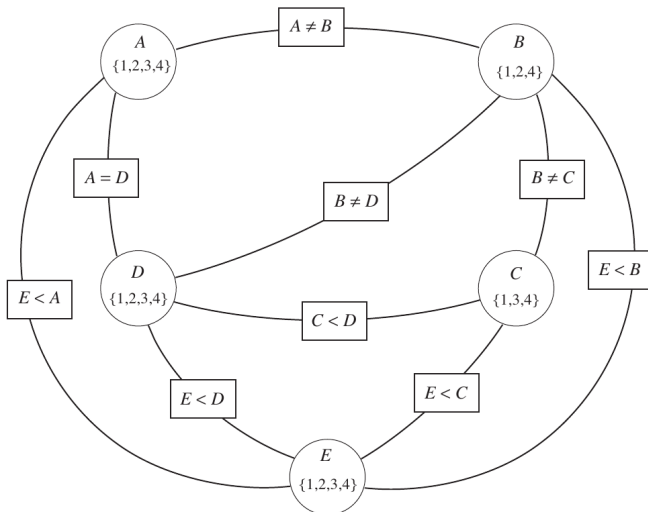
 if $D' \neq \text{null}$ return D'

return null

CS 3793/5233 Artificial Intelligence

Features – 6

Example CSP



CS 3793/5233 Artificial Intelligence

Features – 5

Optimization Search

7

Local Search

Local search methods try to improve an assignment to the variables by taking a series of small steps. The goal is to to efficiently find good solutions.

Procedure $Local-Search(V, C)$

Inputs: V : a set of variables
 C : a set of constraints

Output: a solution A

$A \leftarrow$ an assignment of variables to values

while A does not satisfy C

$A \leftarrow$ choose a “neighbor” of A

 return A

CS 3793/5233 Artificial Intelligence

Features – 7

Iterative Improvement

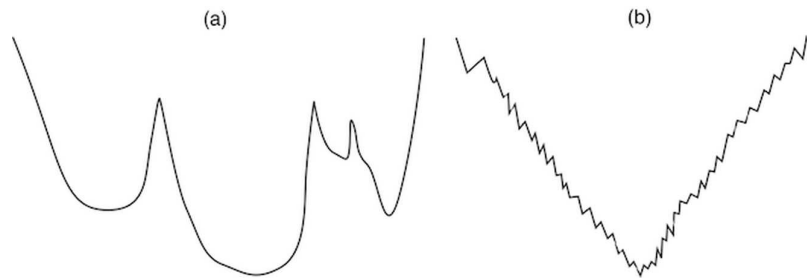
- An *evaluation function* provides the value of each state.
 - For CSPs, one could use the number of constraints that are not satisfied.
- *Hill climbing* selects the best neighbor.
- For continuous variables, *gradient descent* changes all the variables based on partial derivatives.

CS 3793/5233 Artificial Intelligence

Features – 8

Local Optima

The problem with iterative improvement is finding states that are locally optimal, but far from globally optimal.



CS 3793/5233 Artificial Intelligence

Features – 9

Randomization

Use *randomization* to avoid local optima.

- Random restart: repeatedly start local search with different assignments.
- Random walk: interleave random steps with improvement steps.
- Tabu list: avoid recently used variable-value assignments to try to avoid cycles.
- Simulated annealing: Start local search with mostly random steps and gradually increase the proportion of improvement steps.

CS 3793/5233 Artificial Intelligence

Features – 10

Genetic Algorithms

Genetic algorithms is a popular local search method, inspired by natural selection.

- Population: Instead of storing one assignment (individual) at a time, store many individuals.
- Mutation: A new individual can be generated by random small steps from an old individual.
- Crossover: A new individual can be generated by combining the variable assignments of two old individuals.
- Random Selection: Select the new population using the evaluation function combined with some randomness.

CS 3793/5233 Artificial Intelligence

Features – 11