EECS 391
Intro to AI

Constraint Satisfaction Problems

L8 Tue Sep 26, 2017

What we've covered so far

- Problem solving with search
  - many ways to define problem spaces
  - many different algorithms to find solutions
  - Uniformed search: BFS, DFS, UCS, DLS, ID DFS, BDS
  - Informed search: Heuristics, Greedy BFS, A*, RBFS, IDA*, SMA*
  - Local search: Hill Climbing, stochastic HC, local beam search (book: SA, GA)
- Main idea: problem solving is defined by searching in a space of states
- Game Playing
  - ways to define game trees, utility, stochastic games
  - Minimax, alpha-beta pruning, evaluation functions
- Today: Constraint Satisfaction Problems (CSPs)

Note: Much of this lecture was written out on the board.

Key concepts today

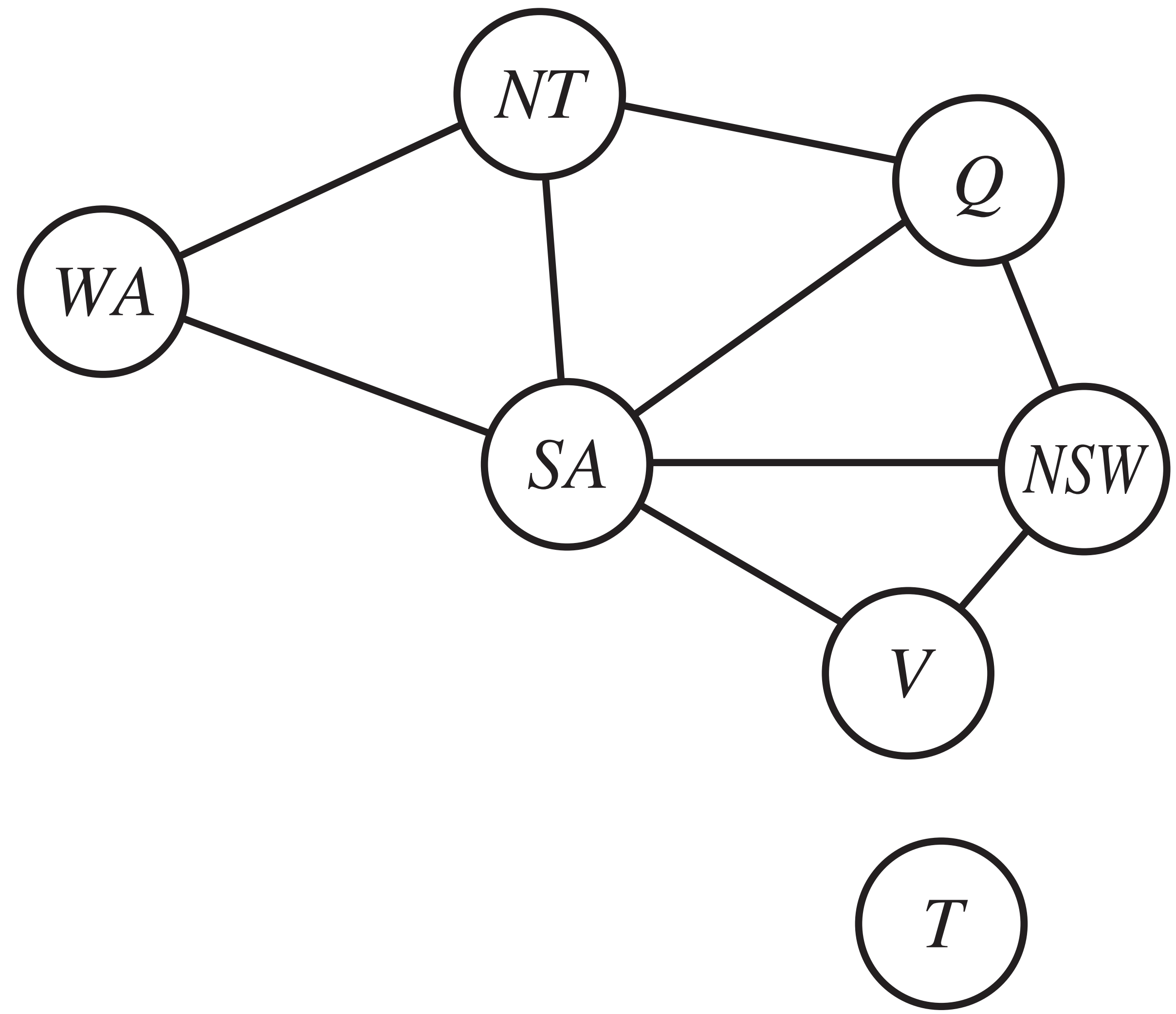
- introducing constraint satisfaction problems (CSPs)
- defining and representing CSPs
- backtracking search for CSPs
- heuristics for reducing size of the search space
  - minimum remaining values
  - degree heuristic
- types of constraint satisfaction problems

Constraint satisfaction example: Map coloring

How do we assign colors to each region so that no neighboring regions have the same color?

Using a graph to represent constraints

- no connected nodes can have same color
- all we need to do is find a valid (i.e. consistent) assignment



Backtracking search

Equiv. to DFS but only one successor at a time: *O(m)* vs *O(bm)* memory

```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
  return BACKTRACK({ }, csp)

function BACKTRACK(assignment, csp) returns a solution, or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(csp)
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
     if value is consistent with assignment then
        add {var = value} to assignment
        inferences ← INFERENCE(csp, var, value)
        if inferences ≠ failure then
           add inferences to assignment
           result ← BACKTRACK(assignment, csp)
           if result ≠ failure then
              return result
     remove {var = value} and inferences from assignment
  return failure
```

Backtracking search

function BACKTRACKING-SEARCH(*csp*) **returns** a solution, or failure
  return BACKTRACK({ }, *csp*)

function BACKTRACK(*assignment*, *csp*) **returns** a solution, or failure
  if *assignment* is complete **then return** *assignment*
  var ← SELECT-UNASSIGNED-VARIABLE(*csp*)
  for each *value* **in** ORDER-DOMAIN-VALUES(*var*, *assignment*, *csp*) **do**
    if *value* is consistent with *assignment* **then**
      add {*var* = *value*} to *assignment*
      inferences ← INFERENCE(*csp*, *var*, *value*)
      if *inferences* ≠ *failure* **then**
        add *inferences* to *assignment*
        result ← BACKTRACK(*assignment*, *csp*)
        if *result* ≠ *failure* **then**
          return *result*
    remove {*var* = *value*} and *inferences* from *assignment*
  return *failure*

only need one:
complete & valid

Backtracking search

function BACKTRACKING-SEARCH(*csp*) **returns** a solution, or failure
  return BACKTRACK({ }, *csp*)

function BACKTRACK(*assignment*, *csp*) **returns** a solution, or failure
  if *assignment* is complete **then return** *assignment*
  var ← SELECT-UNASSIGNED-VARIABLE(*csp*)
  for each *value* **in** ORDER-DOMAIN-VALUES(*var*, *assignment*, *csp*) **do**
    if *value* is consistent with *assignment* **then**
      add {*var* = *value*} to *assignment*
      inferences ← INFERENCE(*csp*, *var*, *value*)
      if *inferences* ≠ *failure* **then**
        add *inferences* to *assignment*
        result ← BACKTRACK(*assignment*, *csp*)
        if *result* ≠ *failure* **then**
          return *result*
    remove {*var* = *value*} and *inferences* from *assignment*
  return *failure*

If not, try assigning another variable. The *order of consideration* will be important.

Backtracking search

function BACKTRACKING-SEARCH(*csp*) **returns** a solution, or failure
  return BACKTRACK({ }, *csp*)

function BACKTRACK(*assignment*, *csp*) **returns** a solution, or failure
  if *assignment* is complete **then return** *assignment*
  var ← SELECT-UNASSIGNED-VARIABLE(*csp*)
  for each *value* **in** ORDER-DOMAIN-VALUES(*var*, *assignment*, *csp*) **do**
    if *value* is consistent with *assignment* **then**
      add {*var* = *value*} to *assignment*
      inferences ← INFERENCE(*csp*, *var*, *value*)
      if *inferences* ≠ *failure* **then**
        add *inferences* to *assignment*
        result ← BACKTRACK(*assignment*, *csp*)
        if *result* ≠ *failure* **then**
          return *result*
    remove {*var* = *value*} and *inferences* from *assignment*
  return *failure*

Loop to search for the next valid assignment. The *order of values considered* will also be important.

Backtracking search

function BACKTRACKING-SEARCH(csp) **returns** a solution, or failure
  return BACKTRACK({ }, csp)

function BACKTRACK($assignment$, csp) **returns** a solution, or failure
  if $assignment$ is complete **then return** $assignment$
  $var \leftarrow$ SELECT-UNASSIGNED-VARIABLE(csp)
  for each $value$ **in** ORDER-DOMAIN-VALUES(var, $assignment$, csp) **do**
    if $value$ is consistent with $assignment$ **then**
      add $\{var = value\}$ to $assignment$
      $inferences \leftarrow$ INFERENCE(csp, var, $value$)
      if $inferences \neq failure$ **then**
        add $inferences$ to $assignment$
        $result \leftarrow$ BACKTRACK($assignment$, csp)
        if $result \neq failure$ **then**
          return $result$
    remove $\{var = value\}$ and $inferences$ from $assignment$
  return $failure$

Add new assignment if consistent with constraints.

Backtracking search

function BACKTRACKING-SEARCH(csp) **returns** a solution, or failure
  return BACKTRACK({ }, csp)

function BACKTRACK($assignment$, csp) **returns** a solution, or failure
  if $assignment$ is complete **then return** $assignment$
  $var \leftarrow$ SELECT-UNASSIGNED-VARIABLE(csp)
  for each $value$ **in** ORDER-DOMAIN-VALUES(var, $assignment$, csp) **do**
    if $value$ is consistent with $assignment$ **then**
      add $\{var = value\}$ to $assignment$
      $inferences \leftarrow$ INFERENCE($csp, var, value$)
      if $inferences \neq failure$ **then**
        add $inferences$ to $assignment$
        $result \leftarrow$ BACKTRACK($assignment$, csp)
        if $result \neq failure$ **then**
          return $result$
    remove $\{var = value\}$ and $inferences$ from $assignment$
  return $failure$

This uses constraint propagation (next lecture) to reduce the size of the search space.

Backtracking search

function BACKTRACKING-SEARCH(*csp*) **returns** a solution, or failure
  return BACKTRACK({ }, *csp*)

function BACKTRACK(*assignment*, *csp*) **returns** a solution, or failure
  if *assignment* is complete **then return** *assignment*
  var ← SELECT-UNASSIGNED-VARIABLE(*csp*)
  for each *value* **in** ORDER-DOMAIN-VALUES(*var*, *assignment*, *csp*) **do**
    if *value* is consistent with *assignment* **then**
      add {*var* = *value*} to *assignment*
      inferences ← INFERENCE(*csp*, *var*, *value*)
      if *inferences* ≠ *failure* **then**
        add *inferences* to *assignment*
        result ← BACKTRACK(*assignment*, *csp*)
        if *result* ≠ *failure* **then**
          return *result*
    remove {*var* = *value*} and *inferences* from *assignment*
  return *failure*

Use recursion to continue down search tree and expand solution.

Backtracking search

```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
  return BACKTRACK({ }, csp)

function BACKTRACK(assignment, csp) returns a solution, or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(csp)
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
      if value is consistent with assignment then
          add {var = value} to assignment
          inferences ← INFERENCE(csp, var, value)
          if inferences ≠ failure then
             add inferences to assignment
             result ← BACKTRACK(assignment, csp)
             if result ≠ failure then
                return result
      remove {var = value} and inferences from assignment
  return failure
```

If we made it here, it means the assignment resulted in an inconsistency, so we have to remove it.

Backtracking search

function BACKTRACKING-SEARCH(*csp*) **returns** a solution, or failure
  return BACKTRACK({ }, *csp*)

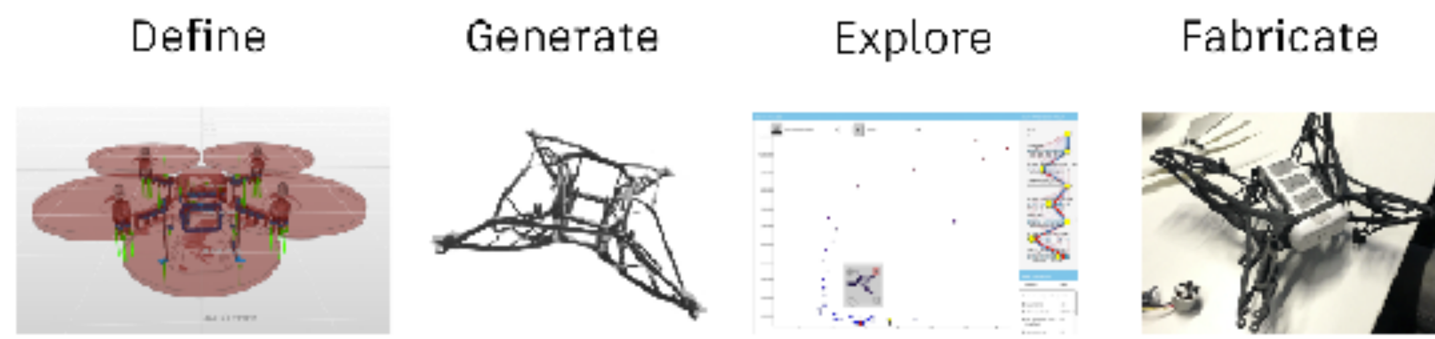
function BACKTRACK(*assignment*, *csp*) **returns** a solution, or failure
  if *assignment* is complete **then return** *assignment*
  var ← SELECT-UNASSIGNED-VARIABLE(*csp*)
  for each *value* **in** ORDER-DOMAIN-VALUES(*var*, *assignment*, *csp*) **do**
    if *value* is consistent with *assignment* **then**
      add {*var* = *value*} to *assignment*
      inferences ← INFERENCE(*csp*, *var*, *value*)
      if *inferences* ≠ *failure* **then**
        add *inferences* to *assignment*
        result ← BACKTRACK(*assignment*, *csp*)
        if *result* ≠ *failure* **then**
          return *result*
    remove {*var* = *value*} and *inferences* from *assignment*
  return *failure*

And finally, this means no solution could be found.

Algorithmic design

Berkley Mills
Lambda Chair

- What if algorithms could "design" chairs?
- Arthur Harsuvanakit and Brittany Presten of Autodesk's generative design lab
- Use Autodesk's Dreamcatcher



- Given constraints:
  - seat is 18" off floor; can hold 300 lbs; arms are clear of human body
- Algorithm:
  - shaved dead weight; adjusted joint placement to improve strength
  - could select interesting designs and iterate from there
- Result:
  - 18% less material; stronger

The “bone” chair

The “bone” chair