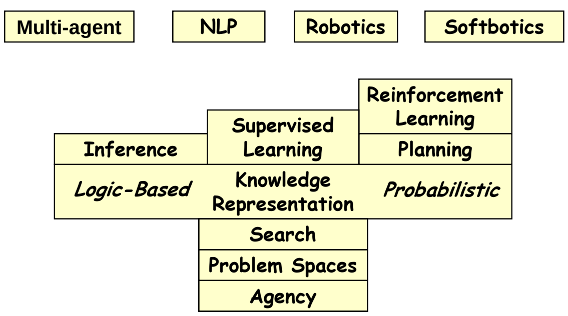
Heuristics
&
Constraint Satisfaction

CSE 573
University of Washington

Logistics

- Reading for Monday
  Ch 6 (Game playing)
- Guest Speaker
  Henry Kautz
- Mini Projects
  A. Game Playing
    - Choose your own game
    - Experiment with heursitics
  B. Compare two SAT solvers
    - DPLL *vs* WalkSAT
    - Experiment with heuristics, constraint propagation, …

573 Topics



Symbolic Integration

- E.g. $\int x^2 e^x \, dx =$
- $e^x(x^2-2x+2) + C$

Operators:

Integration by parts

Integration by substitution

…

Search

√ Problem spaces
√ Blind
  √ Depth-first, breadth-first, iterative-deepening,
  √ iterative broadening
√ Informed
  √ Best-first, Dijkstra's, A*, IDA*, ~~SMA*~~, ~~DFB&B~~, Beam,
Local search
  hill climb, limited discrep, RTDP
Heuristics
  Evaluation, construction via relaxation
Pattern databases
Constraint satisfaction
Adversary search

Symbolic Integration

- E.g. $\int x^2 e^x \, dx =$
- $e^x(x^2-2x+2) + C$

Operators:

Integration by parts

Integration by substitution

…

Depth-First Branch & Bound

- Single DF search
  → uses linear space
- Keep track of best solution so far
- If $f(n) = g(n)+h(n) \geq cost(best\text{-}soln)$
  Then prune n
- Requires
  Finite search tree, or
  Good upper bound on solution cost
- Generates duplicate nodes in cyclic graphs

Beam Search

- Idea
  Best first but only keep N best items on priority queue
- Evaluation
  Complete?

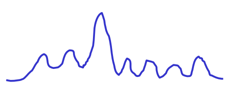
  Time Complexity?

  Space Complexity?

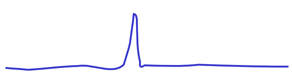
Hill Climbing

"Gradient ascent"

- Idea
  Always choose best child; no backtracking
  Beam search with |queue| = 1
- Problems?
  Local maxima



  Plateaus



  Diagonal ridges

Randomizing Hill Climbing

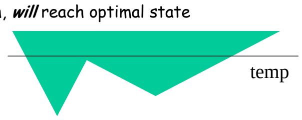
- Randomly disobeying heuristic
- Random restarts

(heavy tailed distributions)

Simulated Annealing

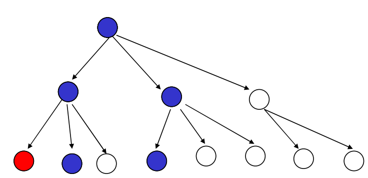
- Objective: avoid local minima
- Technique:
  For the most part use hill climbing
  When no improvement possible
  - Choose random neighbor
  - Let Δ be the decrease in quality
  - Move to neighbor with probability $e^{-\Delta/T}$

  Reduce "temperature" (T) over time
- Pros & cons
  Optimal?
  If T decreased slowly enough, ***will*** reach optimal state
- Widely used
  See also WalkSAT



Limited Discrepancy Search

- Discrepancy bound indicates how often to violate heuristic
- Iteratively increase...



Assume that heuristic says go ***left***

Genetic Algorithms

- Start with random population
  - Representation serialized
  - States are ranked with "fitness function"
- Produce new generation
  - Select random pair(s):
    - probability ~ fitness
  - Randomly choose "crossover point"
    - Offspring mix halves
  - Randomly mutate bits

	Crossover		Mutation	
174629844710	→	174611094281	→	164611094281
776511094281		776529844710		776029844210

Search

√ Problem spaces
√ Blind
- √ Depth-first, breadth-first, iterative-deepening,
- √ iterative broadening

√ Informed
- √ Best-first, Dijkstra's, A*, IDA*, SMA*, DFB&B,
- √ Beam, hill climb, limited discrep, RTDP

√ Local search
Heuristics
- Evaluation, construction via relaxation

Pattern databases
Constraint satisfaction
Adversary search

Admissable Heuristics

- f(x) = g(x) + h(x)
- g: cost so far
- h: underestimate of remaining costs

Where do heuristics come from?

Relaxed Problems

- Derive admissible heuristic from **exact** cost of a solution to a **relaxed** version of problem
  - For transportation planning, relax requirement that car has to stay on road → Euclidean dist
- Cost of optimal soln to relaxed problem ≤ cost of optimal soln for real problem

Simplifying Integrals

vertex = formula
goal = closed form formula without integrals
arcs = mathematical transformations

$$\int x^n dx \rightarrow \frac{x^{n+1}}{n+1}$$

heuristic = number of integrals still in formula

what is being relaxed?

Traveling Salesman Problem

- Problem Space
  - **States = partial path (not nec. connected)**
  - **Operator = add an edge**
  - **Start state = empty path**
  - **Goal = complete path**
- Heuristic?

What can be Relaxed?

Heuristics for eight puzzle

7	2	3
5	1	6
8	3	

→

1	2	3
4	5	6
7	8	

start **goal**

- What can we relax?

Importance of Heuristics

7	2	3
4	1	6
8	5	

- h1 = number of tiles in wrong place
- h2 = Σ distances of tiles from correct loc

D	IDS	A*(h1)	A*(h2)
2	10	6	6
4	112	13	12
6	680	20	18
8	6384	39	25
10	47127	93	39
12	364404	227	73
14	3473941	539	113
18		3056	363
24		39135	1641

Need More Power!

Performance of Manhattan Distance Heuristic

8 Puzzle	< 1 second
15 Puzzle	1 minute
24 Puzzle	65000 years

Need even better heuristics!

Subgoal Interactions

- Manhattan distance assumes
  - Each tile can be moved independently of others
- Underestimates because
  - Doesn't consider interactions between tiles

1	2	3
4	6	5
7	8	

Pattern Databases

[Culberson & Schaeffer 1996]

- Pick any subset of tiles
  - E.g., 3, 7, 11, 12, 13, 14, 15
- Precompute a table
  - Optimal cost of solving just these tiles
  - For all possible configurations
    - 57 Million in this case
  - Use breadth first search back from goal state
    - State = position of just these tiles (& blank)

Using a Pattern Database

- As each state is generated
  - Use position of chosen tiles as index into DB
  - Use lookup value as heuristic, h(n)

  Admissible?

Combining Multiple Databases

- Can choose another set of tiles
  - Precompute multiple tables
- How combine table values?

- E.g. Optimal solutions to Rubik's cube
  - First found w/ IDA* using pattern DB heuristics
  - Multiple DBs were used (dif subsets of cubies)
  - Most problems solved optimally in 1 day
  - Compare with ***574,000 years*** for IDDFS

Drawbacks of Standard Pattern DBs

- Since we can only take ***max***
  - Diminishing returns on additional DBs

- Would like to be able to ***add*** values

Disjoint Pattern DBs

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

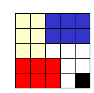
- Partition tiles into disjoint sets
  - For each set, precompute table
    - E.g. 8 tile DB has 519 million entries
    - And 7 tile DB has 58 million
- During search
  - Look up heuristic values for each set
  - ***Can add values without overestimating!***

  Manhattan distance is a special case of this idea where each set is a single tile

Performance

- 15 Puzzle: 2000x speedup *vs* Manhattan dist
  - IDA* with the two DBs shown previously solves 15 Puzzles optimally in 30 milliseconds

- 24 Puzzle: 12 million x speedup *vs* Manhattan
  - IDA* can solve random instances in 2 days.
  - Requires 4 DBs as shown
    - Each DB has 128 million entries
  - Without PDBs: 65000 years



Outline

√ Problem spaces
√ Search
- √ Blind
- √ Informed
- √ Local
- √ Heuristics & Pattern DBs for
- Constraint satisfaction
  - Definition
    - Factoring state spaces
  - Backtracking policies
  - Variable-ordering heuristics
  - Preprocessing algorithms
- Adversary search

Constraint Satisfaction

- Kind of ***search*** in which
  - States are ***factored*** into sets of variables
  - Search = assigning values to these variables
  - Structure of space is encoded with constraints
- Backtracking-style algorithms work
  - E.g. DFS for SAT (i.e. DPLL)
- But other techniques add speed
  - Propagation
  - Variable ordering
  - Preprocessing

Chinese Food as Search?

- **States?**
  - Partially specified meals
- **Operators?**
  - Add, remove, change dishes
- **Start state?**
  - Null meal
- **Goal states?**
  - Meal meeting certain conditions (rating?)

Factoring States

- Rather than state = meal
- Model state's (independent) parts, e.g.

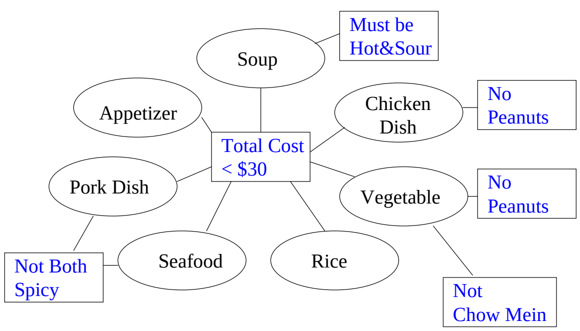
Suppose every meal for n people
Has n dishes plus soup

Soup =
Meal 1 =
Meal 2 =
…
Meal n =

- Or… physical state =

X coordinate =
Y coordinate =

Chinese Constraint Network



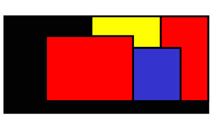
CSPs in the Real World

- Scheduling space shuttle repair
- Airport gate assignments
- Transportation Planning
- Supply-chain management
- Computer configuration
- Diagnosis
- UI optimization
- Etc...

Binary Constraint Network

- Set of n ***variables***: $x_1 \ldots x_n$
- Value ***domains*** for each variable: $D_1 \ldots D_n$
- Set of binary ***constraints*** (also "relations")

$R_{ij} \subseteq D_i \times D_j$
Specifies which value pairs (x_i x_j) are consistent



- V for each country
- Each domain = 4 colors
- R_{ij} enforces $\neq$

Binary Constraint Network

Partial ***assignment*** of values = tuple of pairs
{...(x, a)...} means variable x gets value a...
Tuple=***consistent*** if all constraints satisfied
Tuple=***full solution*** if consistent + has all vars

Tuple $\{(x_i, a_i) \ldots (x_j, a_j)\}$ = ***consistent w/ a set of vars*** $\{x_m \ldots x_n\}$
iff $\exists\, a_m \ldots a_n$ such that
$\{(x_i, a_i) \ldots (x_j, a_j), (x_m, a_m) \ldots (x_n, a_n)\}$ = consistent

Cryptarithmetic

- ~~State Space~~
  - ~~Set of states~~
  - ~~Operators [and costs]~~
  - ~~Start state~~
  - ~~Goal states~~

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

- Variables?
- Domains (variable values)?
- Constraints?

Classroom Scheduling

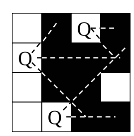
- Variables?
- Domains (possible values for variables)?
- Constraints?

N Queens

- As a CSP?

N Queens

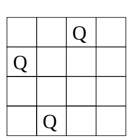
- Variables = board columns
- Domain values = rows
- $R_{ij} = \{(a_i, a_j) \;:\; (a_i \neq a_j) \wedge (|i-j| \neq |a_i-a_j|)\}$
  e.g. $R_{12} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$



- $\{(x_1, 2), (x_2, 4), (x_3, 1)\}$ consistent with (x_4)
- Shorthand: "$\{2, 4, 1\}$ consistent with x_4"

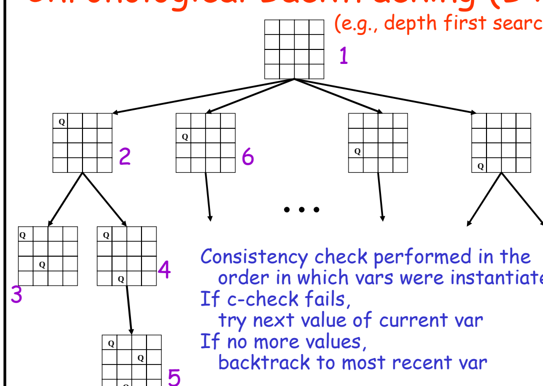
CSP as a search problem?

- What are states?
  (nodes in graph)
- What are the operators?
  (arcs between nodes)
- Initial state?
- Goal test?



Chronological Backtracking (BT)

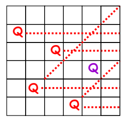
(e.g., depth first search)



Consistency check performed in the order in which vars were instantiated
If c-check fails,
try next value of current var
If no more values,
backtrack to most recent var

Backjumping (BJ)

- Similar to BT, but
  more efficient when no consistent instantiation can be found for the current var
- Instead of backtracking to most recent var…
  BJ reverts to deepest var which was c-checked against the current var



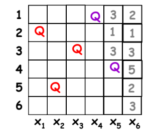
BJ Discovers
(2, 5, 3, 6) inconsistent with x_6
No sense trying other values of x_5

Conflict-Directed Backjumping (CBJ)

- More sophisticated backjumping behavior
- Each variable has ***conflict set CS***
  Set of vars that failed c-checks w/ current val
  Update this set on every failed c-check
- When no more values to try for x_i
  Backtrack to deepest var, x_d, in $CS(x_i)$
  And update $CS(x_d):=CS(x_d) \cup CS(x_i)-\{x_d\}$

CBJ Discovers (2, 5, 3) inconsistent with $\{x_5, x_6\}$

$CS(x_5)$	$CS(x_6)$
1,2,3	1,2,3,5



BT *vs.* BJ *vs.* CBJ

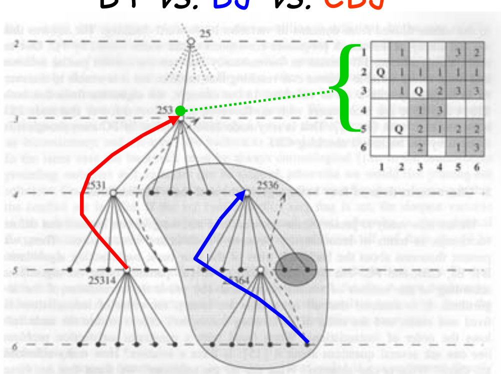
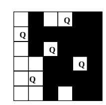


Fig. 2. A fragment of the BT backtrack tree for the 6-queens problem.

Forward Checking (FC)

- Perform Consistency Check ***Forward***
- Whenever a var is assigned a value
  Prune inconsistent values from
  As-yet unvisited variables
  Backtrack if domain of any var ever collapses

FC only visits consistent nodes
but not *all* such nodes
skips (2, 5, 3, 4) which CBJ visits
But FC can't detect that
(2, 5, 3) inconsistent with $\{x_5, x_6\}$



Number of Nodes Explored

More

BT=BM

BJ=BMJ=BMJ2

FC

CBJ=BM-CBJ
=BM-CBJ2

Fewer

FC-CBJ

Number of Consistency Checks

More

BT

BJ

BM

BMJ

CBJ

FC

FC-CBJ

BMJ2

BM-CBJ

Fewer

BM-CBJ2

