Informed Search

AI Class 5 (Ch. 3.5-3.7)

"An informed search strategy—one that uses problem specific knowledge… can find solutions more efficiently then an uninformed strategy." – R&N pg. 92

Based on slides by Dr. Marie desJardin. Some material also adapted from slides by Dr. Matuszek @ Villanova University, which are based on Hwee Tou Ng at Berkeley, which are based on Russell at Berkeley. Some diagrams are based on AIMA.

1

Blind Search (Redux)

- Last time:
  - Breadth-first
  - Depth-first
  - Uniform-cost
  - Iterative deepening
- From the book:
  - Bidirectional
  - Holy Grail Search

2

Today's Class

- Heuristic search
- Heuristic functions
- Admissibility
- Best-first search
  - Greedy search, beam search, A, A*
  - Examples
- Memory-conserving variations of A*

Questions?

"An informed search strategy—one that uses problem specific knowledge… can find solutions more efficiently then an uninformed strategy."

– R&N pg. 92

3

Definition: Heuristic

Free On-line Dictionary of Computing*

1. **A rule of thumb, simplification, or educated guess**
2. Reduces, limits, or guides search in particular domains
3. Does not guarantee feasible solutions; often with no theoretical guarantee

Playing chess: try to take the opponent's queen

Getting someplace: head in that compass direction when possible

WordNet (r) 1.6*

1. Commonsense rule (or set of rules) intended to increase the probability of solving some problem

*Heavily edited for clarity

4

Heuristic Search

- Uninformed search is **generic**
  - Node selection depends only on shape of tree and node expansion strategy
- **Domain knowledge** → better decisions (sometimes)
  - Knowledge about the specific problem
  - Often calculated based on state

5

Is It A Heuristic?

- A **heuristic function** is:
  - An **estimate** of how close we are to a goal
    - We don't assume perfect knowledge
      - That would be holy grail search
    - The estimate can be wrong
  - Based on domain-specific information
  - Computable from the current state description

6

Heuristic Search

- Romania: Arad→ Bucharest (for example)

7

7

Heuristic Search

- Romania:
  - Eyeballing it → certain cities first
  - They "look closer" to where we are going
- Can domain knowledge be captured in a **heuristic?**

8

8

Heuristics Examples

- 8-puzzle:
  - # of tiles in wrong place
- 8-puzzle (better):
  - Sum of distances from goal
  - Captures distance *and* number of nodes
- Romania:
  - Straight-line distance from start node to Bucharest
  - Captures "closer to Bucharest"

5	4	
6	1	8
7	3	2

9

9

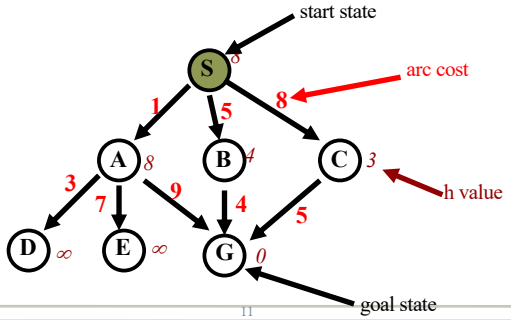
Heuristic Function

- **All** domain-specific knowledge is encoded in heuristic function *h*
- *h* is some **estimate** of how desirable a move is
  - How "close" (we think, maybe) it gets us to our goal
- Usually:
  - $h(n) \geq 0$: for all nodes *n*
  - $h(n) = 0$: *n* is a goal node
  - $h(n) = \infty$: *n* is a dead end (no goal can be reached from *n*)

10

10

Example Search Space Revisited



11

11

Weak vs. Strong Methods

- **Weak methods**:
  - Extremely general, not tailored to a specific situation
- Examples
  - **Subgoaling**: split a large problem into several smaller ones that can be solved one at a time.
  - **Space splitting:** try to list possible solutions to a problem, then try to rule out *classes* of these possibilities
  - **Means-ends analysis:** consider current situation and goal, then look for ways to shrink the differences between the two
- Called "weak" methods because they do not take advantage of more powerful domain-specific heuristics

12

12

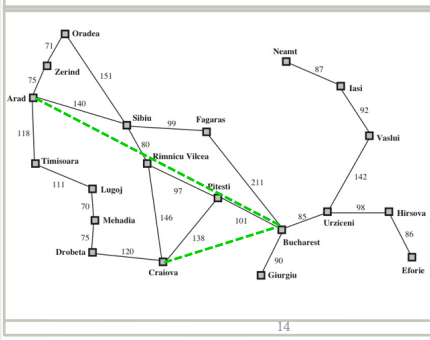
Domain Information

- Informed methods add domain-specific information!
- Goal: **select** the best path to continue searching
- Define ***h(n)*** to estimate the "goodness" of node *n*
  - *h*(*n*) = **estimated cost** (or distance) of minimal cost path from *n* **to a goal state**

13

13

Straight Lines to Budapest (km)



R&N pg. 68, 93

14

14

Admissible Heuristics

- Admissible heuristics never overestimate cost
  - They are *optimistic* – think goal is closer than it is
  - $h(n) \leq h^*(n)$
    - where $h^*(n)$ is **true** cost to reach goal from *n*
  - h_{SLD}(Lugoj) = 244
    - Can there be a shorter path?
- Using admissible heuristics guarantees that the first solution found will be optimal

15

15

Admissibility

- Admissibility is a property of **heuristics**
  - They are *optimistic* – think goal is closer than it is
  - (Or, exactly right)
- Is *h*(*n*): "1 kilometer" admissible?
- Admissible algorithms can be pretty bad!

- Using admissible heuristics guarantees that the first solution found will be optimal, **for some algorithms** (A*).

16

16

Admissibility and Optimality

- Intuitively:
  - When A* finds a path of length *k*, it has already tried **every other path which can have length ≤ k**
  - Because all frontier nodes have been sorted in ascending order of $f(n)=g(n)+h(n)$
- Does an admissible heuristic guarantee optimality for greedy search?
  - Reminder: $f(n) = h(n)$, always choose node "nearest" goal
  - No sorting beyond that

17

17

Best-First Search

- A generic way of referring to informed methods
- Use an **evaluation function** $f(n)$ for each node
  → estimate of "desirability"
  - $f(n)$ incorporates domain-specific information
  - Different $f(n)$ → Different searches
  - *h*(*n*) instead ranks alternative paths at a node

18

18

Best-First Search (more)

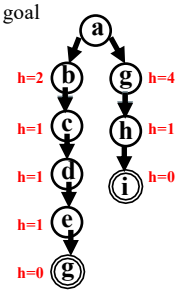
- Order nodes on the list by
  - Increasing value of $f(n)$
- Expand **most desirable** unexpanded node
  - Implementation:
  - Order nodes in frontier in decreasing order of desirability
- Special cases:
  - Greedy best-first search
  - A* search

19

19

Greedy Best-First Search

- Idea: always choose "closest node" to goal
  - Most likely to lead to a solution quickly
- So, evaluate nodes based only on heuristic function
- Sort nodes by increasing values of f
- Select node believed to be **closest** to a goal node (hence "greedy")
  - That is, select node with smallest f value

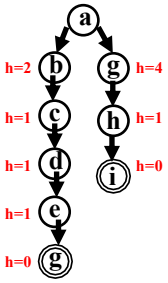


20

20

Greedy Best-First Search

- Admissible?
  - Why not?
- Example:
  - Greedy search will find:
    a→b→c→d→e→g ; cost = 5
  - Optimal solution:
    a→g→h→i ; cost = 3
- Not complete (why?)

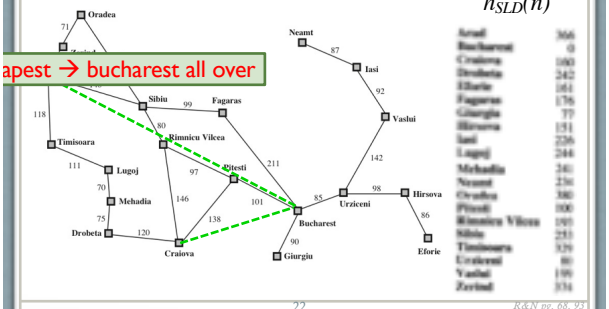


21

21

Straight Lines to Budapest (km)

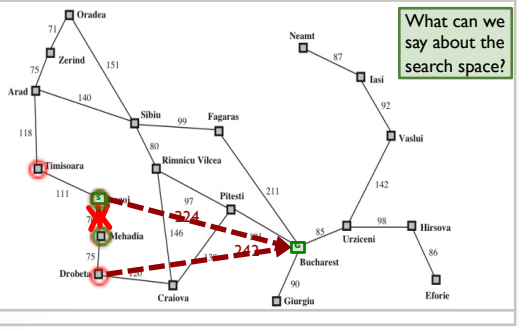
$h_{SLD}(n)$



22

22

Greedy Best-First Search: Ex. 1



23

23

Greedy Best-First Search: Ex. 2

24

24

Greedy Best-First Search: Ex. 2

25

25

Greedy Best-First Search: Ex. 2

26

26

Greedy Best-First Search: Ex. 2

27

27

Beam Search

- Use an evaluation function $f(n) = h(n)$, but the maximum size of the nodes list is k, a fixed constant
- Only keeps k best nodes as candidates for expansion, and throws the rest away
- More space-efficient than greedy search, but may throw away a node that is on a solution path
- Not complete
- Not admissible

28

28

Quick Terminology Reminders

- What is $f(n)$?
  - An **evaluation function** that gives…
  - …
  - The distance from n to G
- What is $h(n)$?
  - A **heuristic function** that…
  - Encodes domain knowledge about…
  - The search space
- What is $h^*(n)$?
  - A **heuristic function** that gives the…
  - **True** cost to reach goal from n
  - Why don't we just use that?
- What is $g(n)$?
  - The **path cost** of getting from S to n
  - describes the "already spent" costs of the current search

love this slide WAY up

36

36

Algorithm A*

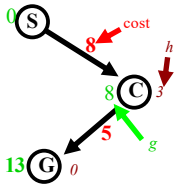
- Use evaluation function $\boldsymbol{f(n) = g(n) + h(n)}$
- $g(n)$ = minimal-cost path from any S to state n
  - That is, the cost of getting to the node **so far**
- Ranks nodes on frontier by *estimated* cost of solution
  - From start node, through given node, to goal
- Not complete if $h(n)$ can $= \infty$

37

37

A* Search

- **Idea:** Evaluate nodes by combining g(n), the cost of reaching the node, with h(n), the cost of getting from the node to the goal.
- Evaluation function:
  ***f*(*n*) = *g*(*n*) + *h*(*n*)**
  - *g(n)* = cost so far to reach *n*
  - *h(n)* = estimated cost from *n* to goal
  - *f(n)* = estimated total cost of path through *n* to goal

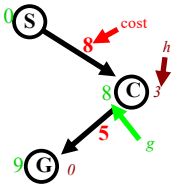


38

38

A* Search

- Avoid expanding paths that are already expensive
  - Combines costs-so-far with expected-costs
- A* is **complete** iff
  - Branching factor is finite
  - Every operator has a fixed positive cost
- A* is **admissible** iff
  - *h*(*n*) is admissible



39

39

A* Example 1

40

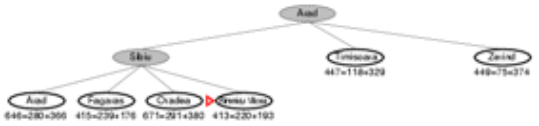
40

A* Example 1

41

41

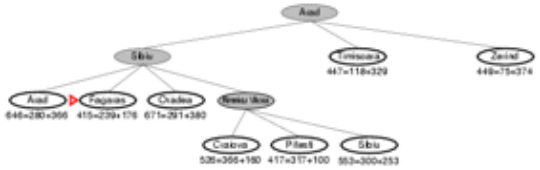
A* Example 1



42

42

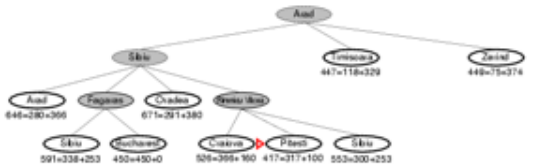
A* Example 1



43

43

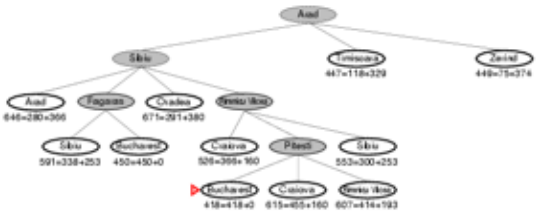
A* Example 1



44

44

A* Example 1



45

45

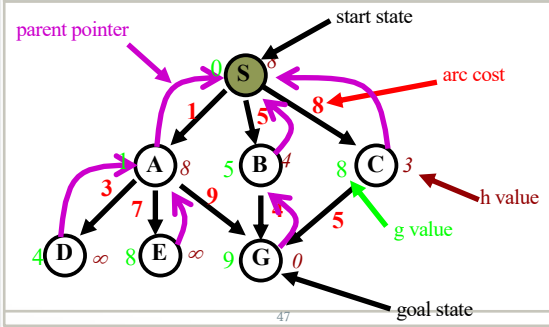
Algorithm A*

- Algorithm A with constraint that $h(n) \leq h^*(n)$
  - $h^*(n)$ = true cost of the minimal cost path from n to a goal.
- Therefore, $h(n)$ is an **underestimate** of the distance to the goal
- $h()$ is **admissible** when $h(n) \leq h^*(n)$
  - Guarantees optimality
- A* is **complete** whenever the branching factor is finite, and every operator has a fixed positive cost
- A* is **admissible**

46

46

Example Search Space Revisited

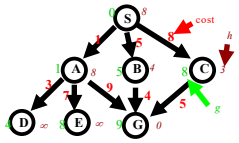


47

47

Example

n	*g*(*n*)	*h*(*n*)	*f*(*n*)	*h**(*n*)
S	0	8	8	9
A	1	8	9	9
B	5	4	9	4
C	8	3	11	5
D	4	∞	∞	∞
E	8	∞	∞	∞
G	9	0	9	0



- $h^*(n)$ is the (hypothetical) perfect heuristic.
- Since $h(n) \leq h^*(n)$ for all n, h is admissible
- Optimal path = S B G with cost 9.

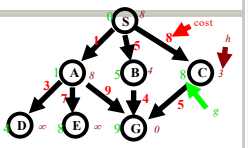
48

48

Greedy Search

$f(n) = h(n)$

Node expanded	Node list
	{ S(8) }
S	{ C(3) B(4) A(8) }
C	{ G(0) B(4) A(8) }
G	{ B(4) A(8) }



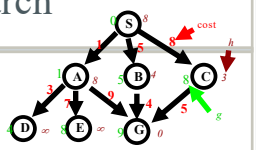
- Solution path found is S C G, 3 nodes expanded.
- Fast!! But NOT optimal.

49

49

A* Search

$f(n) = g(n) + h(n)$



node exp.	nodes list
	{ S(8) }
S	{ A(9) B(9) C(11) }
A	{ B(9) G(10) C(11) D(∞) E(∞) }
B	{ G(9) G(10) C(11) D(inf) E(∞) }
G	{ C(11) D(∞) E(∞) }

- Solution path found is S B G, 4 nodes expanded..
- Still pretty fast, *and* optimal

50

50

Proof of the Optimality of A*

- Assume that A* has selected G_2, a goal state with a suboptimal solution ($g(G_2) > f^*$).
- We show that this is impossible.
  - Choose a node n on the optimal path to G.
  - Because $h(n)$ is admissible, $f(n) \leq f^*$.
  - If we choose G_2 instead of n for expansion, $f(G_2) \leq f(n)$.
  - This implies $f(G_2) \leq f^*$.
  - G_2 is a goal state: $h(G_2) = 0, f(G_2) = g(G_2)$.
  - Therefore $g(G_2) \leq f^*$
  - Contradiction.

51

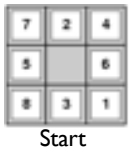
51

Admissible heuristics

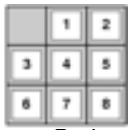
E.g., for the 8-puzzle:

$h_1(n)$ = number of misplaced tiles

$h_2(n)$ = total Manhattan distance
(i.e., # of squares each tile is from desired location)



Start



Goal

- $h_1(S) = ?$
- $h_2(S) = ?$

52

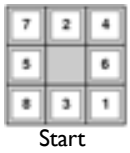
52

Admissible heuristics

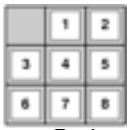
E.g., for the 8-puzzle:

$h_1(n)$ = number of misplaced tiles

$h_2(n)$ = total Manhattan distance
(i.e., # of squares each tile is from desired location)



Start



Goal

- $h_1(S) = 8$
- $h_2(S) = 3+1+2+2+2+3+3+2 = 18$

53

53

Dealing with Hard Problems

- For large problems, A* often requires too much space.
- Two variations conserve memory: IDA* and SMA*
- IDA* – iterative deepening A*
  - uses successive iteration with growing limits on f. For example,
    - A* but don't consider any node n where $f(n) > 10$
    - A* but don't consider any node n where $f(n) > 20$
    - A* but don't consider any node n where $f(n) > 30$, ...
- SMA* – Simplified Memory-Bounded A*
  - uses a queue of restricted size to limit memory use.
  - throws away the "oldest" worst solution.

54

54

What's a Good Heuristic?

- If $h_1(n) < h_2(n) \leq h^*(n)$ for all n, then:
  - Both are admissible
  - h_2 is strictly better than (**dominates**) h_1.
- How do we find one?

1. **Relaxing the problem:**
   - Remove constraints to create a (much) easier problem
   - Use the solution cost for this problem as the heuristic function
2. **Combining heuristics:**
   - Take the max of several admissible heuristics
   - Still have an admissible heuristic, and it's better!

55

55

What's a Good Heuristic? (2)

3. Use statistical estimates to compute *h*
   - May lose admissibility
4. Identify good features, then use a learning algorithm to find a heuristic function
   - Also may lose admissibility

- Why are these a good idea, then?
  - Machine learning can give you answers you don't "think of"
  - Can be applied to new puzzles without human intervention
  - Often work

56

56

Some Examples of Heuristics?

- 8-puzzle?
  - Manhattan distance
- Driving directions?
  - Straight line distance
- Crossword puzzle?
- Making a medical diagnosis?

57

57

Summary: Informed Search

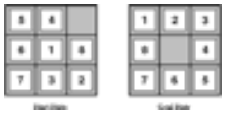
- **Best-first search:** general search where the *minimum-cost nodes* (according to some measure) are expanded first.
- **Greedy search:** uses *minimal estimated cost h*(*n*) to the goal state as measure. Reduces search time but, is neither complete nor optimal.
- **A* search:** combines UCS and greedy search
  - $f(n) = g(n) + h(n)$
  - A* is complete and optimal, but space complexity is high.
  - Time complexity depends on the quality of the heuristic function.
- IDA* and SMA* reduce the memory requirements of A*.

58

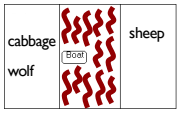
58

In-class Exercise: Creating Heuristics

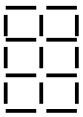
8-Puzzle



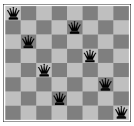
Boat Problems



Remove 5 Sticks



N-Queens



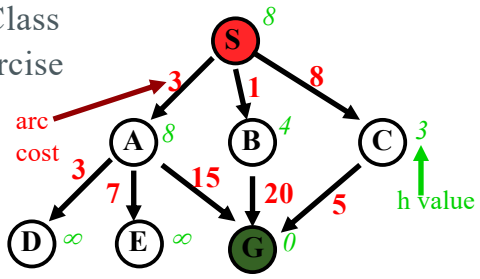
Water Jug Problem

Route Planning

59

59

In-Class Exercise



Apply the following to search this space. At each search step, show: the current node being expanded, $g(n)$ (path cost so far), $h(n)$ (heuristic estimate), $f(n)$ (evaluation function), and $h^*(n)$ (true goal distance).

Depth-first search
Uniform-cost search
Breadth-first search
Greedy search
A* search

60