

Informed/Heuristic Search

Algorithms	2
Best-First Search	2
General Best-First Search Algorithm	3
A* Search Algorithm	4
Iterative Deepening A* Search Algorithm	5
IDA*'s Contour Procedure	6
Analysis	7
Properties of A* Search	7
Optimality Proof	8
Efficiency of A*	9
Performance of Heuristic Functions	10
Book Experiment Avoiding Reverse Moves	11
Local Search	12
Local Search	12
Local Optima Example	13
Local Search Algorithm	14
Examples of Local Search Algorithms	15

Algorithms

Best-First Search

- Simple search algorithms such as IDS do not consider the goodness of states.
- *Informed/heuristic search* prefers to visit states that appear to be better.
- *Best-first search* visits states according to an *evaluation function*.
- An *evaluation function* f gives lower numbers to (seemingly) better states.
- The evaluation function for *A* search* is the cost g from the initial state plus a *heuristic function* h .
- A *heuristic function* estimates the cost from a given state to the closest goal state.

General Best-First Search Algorithm

```

function BEST-FS( $s_{initial}$ , EXPAND, GOAL,  $f$ )
 $q \leftarrow$  NEW-PRIORITY-QUEUE()
INSERT( $s_{initial}$ ,  $q$ ,  $f(s_{initial})$ )
while  $q$  is not empty
do  $s_{current} \leftarrow$  EXTRACT-MIN( $q$ )
   if GOAL( $s_{current}$ ) then return solution
   for each  $s_{successor}$  in EXPAND( $s_{current}$ )
     do INSERT( $s_{successor}$ ,  $q$ ,  $f(s_{successor})$ )
return failure
    
```

A* Search Algorithm

```
function A*( $s_{\text{initial}}$ , EXPAND, GOAL,  $g$ ,  $h$ )
 $q \leftarrow$  NEW-PRIORITY-QUEUE()
INSERT( $s_{\text{initial}}$ ,  $q$ ,  $h(s_{\text{initial}})$ )
while  $q$  is not empty
  do  $s_{\text{current}} \leftarrow$  EXTRACT-MIN( $q$ )
     if GOAL( $s_{\text{current}}$ ) then return solution
     for each  $s_{\text{successor}}$  in EXPAND( $s_{\text{current}}$ )
       do INSERT( $s_{\text{successor}}$ ,  $q$ ,
                  $g(s_{\text{successor}}) + h(s_{\text{successor}})$ )
return failure
```

CS 3793 Artificial Intelligence

Heuristic Search – 4

IDA*'s Contour Procedure

```
function CONTOUR( $s_{\text{current}}$ , limit)
 $eval \leftarrow g(s_{\text{current}}) + h(s_{\text{current}})$ 
if limit <  $eval$  then return null,  $eval$ 
if GOAL( $s_{\text{current}}$ ) then return solution,  $eval$ 
 $new-limit \leftarrow \infty$ 
for each  $s_{\text{successor}}$  in EXPAND( $s_{\text{current}}$ )
  do solution,  $eval \leftarrow$  CONTOUR( $s_{\text{successor}}$ , limit)
     if solution  $\neq$  null then return solution,  $eval$ 
      $new-limit \leftarrow \min(new-limit, eval)$ 
return failure,  $new-limit$ 
```

CS 3793 Artificial Intelligence

Heuristic Search – 6

Iterative Deepening A* Search Algorithm

```
function IDA*( $s_{\text{initial}}$ , EXPAND, GOAL,  $g$ ,  $h$ )
limit  $\leftarrow$  HEURISTIC( $s_{\text{initial}}$ )
loop
  do solution, limit  $\leftarrow$  CONTOUR( $s_{\text{initial}}$ , limit)
     if solution  $\neq$  null then return solution
     if limit =  $\infty$  then return failure
```

CS 3793 Artificial Intelligence

Heuristic Search – 5

Analysis

7

Properties of A* Search

- n : variable standing for a state
- $g(n)$: the cost from the initial state to n .
- $h(n)$: the estimate from n to a goal state.
- $f(n) = g(n) + h(n)$.
- Each action costs at least 1 unit.
- Number of actions are finite.
- h is *admissible* (h is never an overestimate).

- Under above conditions, A* finds optimal path.
- If above conditions, h has at most ϵ error, and the search space is a uniform tree with one goal state, then A* searches at most $\epsilon/2$ from solution path.

CS 3793 Artificial Intelligence

Heuristic Search – 7

Optimality Proof

- Let f^* be optimal path cost.
- Because h never overestimates, then all states n on optimal path have $f(n) \leq f^*$.
- Any nonoptimal goal state n' has $f(n') > f^*$.
- Because of priority queue, A^* will visit states on optimal path before any nonoptimal goal state.
- Other conditions prevent infinite search in a flat region of the state space.

CS 3793 Artificial Intelligence

Heuristic Search – 8

Efficiency of A^*

- Assume *tree-structured* state space (b = branching factor, d = goal depth), single goal state, each edge costs 1, and maximum error of ϵ .
- Any state n more than $\epsilon/2$ off of solution path has $f(n) = g(n) + h(n) > f^*$.
- All states n on solution path have $f(n) = g(n) + h(n) \leq f^*$.
- A^* and IDA^* visit $O(db^{\epsilon/2})$ states.
- A^* uses $O(db^{\epsilon/2})$ memory. IDA^* uses $O(db)$.

CS 3793 Artificial Intelligence

Heuristic Search – 9

Performance of Heuristic Functions

Consider these 8-puzzle heuristic functions:

- h_1 : number of tiles in goal position.
- h_2 : Manhattan distance from tiles to goals.
- Both never overestimate and $h_1 \leq h_2$

Characterize by *effective branching factor*

- Let N states be visited and solution depth be d .
- Solve for x in $N = \sum_{i=0}^d x^i$

CS 3793 Artificial Intelligence

Heuristic Search – 10

Book Experiment Avoiding Reverse Moves

d	States Visited (Effective BF)		
	IDS	$IDA^*(h_1)$	$IDA^*(h_2)$
4	52 (2.35)	10 (1.35)	7 (1.17)
8	569 (2.03)	42 (1.36)	14 (1.11)
12	5357 (1.92)	315 (1.47)	45 (1.19)
16	47271 (1.87)	2410 (1.52)	226 (1.28)
20		17646 (1.55)	764 (1.29)

CS 3793 Artificial Intelligence

Heuristic Search – 11

Local Search

12

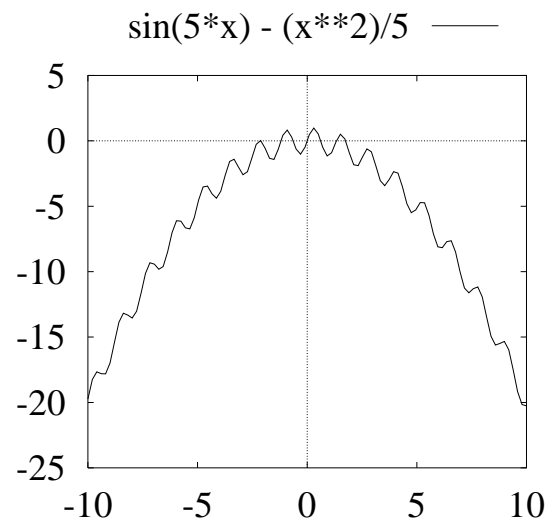
Local Search

- A *local search* algorithm keeps track of one state (or a few states) at a time.
- An evaluation function and a selection procedure is used to decide what state to visit next.
- Local search gives up optimality guarantees in hopes of finding good solutions more efficiently.
- The main difficulty is local minima/maxima.

CS 3793 Artificial Intelligence

Heuristic Search – 12

Local Optima Example



CS 3793 Artificial Intelligence

Heuristic Search – 13

Examples of Local Search Algorithms

- Hill-Climbing, Gradient Descent:
Select state improving an evaluation function.
- Random-restart hill-climbing:
Repeat hill climbing from random initial states.
- Stochastic Hill Climbing/Simulated Annealing:
Hill-climbing with randomized selection.
- Beam Search/Genetic Algorithms:
Maintain a set of “current states.” GA crossover generates new states from pairs of states.
- Tabu Search: Like hill-climbing, but avoid recently visited states or recently used operators.

CS 3793 Artificial Intelligence

Heuristic Search – 15

Local Search Algorithm

```
function LOCAL-SEARCH( $s_{\text{initial}}$ , EXPAND,  
                     GOAL, SELECT)  
  
   $s_{\text{current}} \leftarrow s_{\text{initial}}$   
  loop  
    do if GOAL( $s_{\text{current}}$ ) then return solution  
       $s_{\text{current}} \leftarrow \text{SELECT}(\text{EXPAND}(s_{\text{current}}))$ 
```

CS 3793 Artificial Intelligence

Heuristic Search – 14