# 472 Recitation

Week 4

Problem set 1 due on this Friday at 5:00 PM

# Weighted A* Search

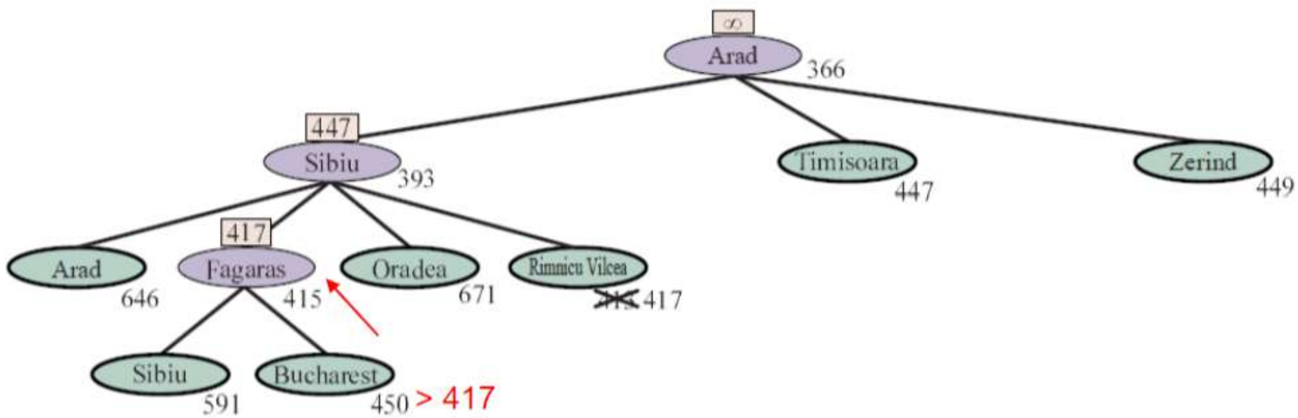
| | | | |
|---|---|---|---|
| A* search | $g(n) + h(n)$ | $(W = 1)$ | |
| Uniform-cost search | $g(n)$ | $(W = 0)$ | |
| Best-cost search | $h(n)$ | $(W = \infty)$ | Greedy Search |
| Weighted A* search | $g(n) + W \times h(n)$ | $(1 \leq W < \infty)$ | |

# Memory Bounded Search

- Beam search keeps the $k$ nodes with the best $f$ scores.
  - ♦ Less memory and faster execution.
  - ♣ Incomplete and suboptimal.
- Iterative-deepening A* search (IDA*)
  - The cutoff at each iteration is the $f$-cost.

# Recursive Best First Search (RBFS)

- Best-first search at the currently visited node $v$, if $f(v) \leq f_{limit}(v)$
- Otherwise, back to the alterative path and update $f$ for nodes on current path




| Criteria | Performance |
|---|---|
| Completeness | Complete with positive cost (I guess so) |
| Optimality | Yes, if the heuristic function is admissible |
| Time Complexity | Depending on $h(n)$ and how often the best path changes |
| Space Complexity | $O(bd)$ |

Both IDA* and RBFS suffering from using too little memory and may explore the same state multiple times.

# Heuristic Functions

- If $h_2$ dominates $h_1$ ($h_2 \geq h_1$for every node), A* using $h_2$ will not expand more nodes than using $h_2$ .

- An admissible heuristic can be derived from optimal solution cost of a relaxed problem (has fewer restrictions).

- Multiple Heuristics

$$h(n) = \max\{h_1(n), h_2(n), \cdots, h_k(n)\}$$

# Local Search- Hill Climbing

**function** HILL-CLIMBING(*problem*) **returns** a state that is a local maximum
    *current* ← *problem*.INITIAL
    **while** *true* **do**
        *neighbor* ← a highest-valued successor state of *current*
        **if** VALUE(*neighbor*) $\leq$ VALUE(*current*) **then return** *current*
        *current* ← *neighbor*

Hill climbing can get stuck on the local optimal state.

- Stochastic hill climbing
- First-choice hill climbing
- Random restart hill climbing

# Local Search- Simulated Annealing

**function** SIMULATED-ANNEALING( *problem*, *schedule*) **returns** a solution state
  $current \leftarrow problem.$INITIAL
  **for** $t$ = 1 **to** $\infty$ **do**
temperature → $T \leftarrow schedule(t)$ *Minimization*
    **if** $T$ = 0 **then return** *current* // solution
    $next \leftarrow$ a randomly selected successor of *current*
badness → $\Delta E \leftarrow$ VALUE(*current*) – VALUE(*next*)
    **if** $\Delta E > 0$ **then** $current \leftarrow next$
    **else** $current \leftarrow next$ only with probability $\cancel{e^{-\Delta E/T}}$ $e^{\Delta E/T}$ $// \Delta E < 0$

- $e^{\Delta E/T} \in (0\ \ 1]$
- T is decreasing
- $e^{\Delta E/T}$ decrease, possibility to switch to new state decrease

# Local Beam Search

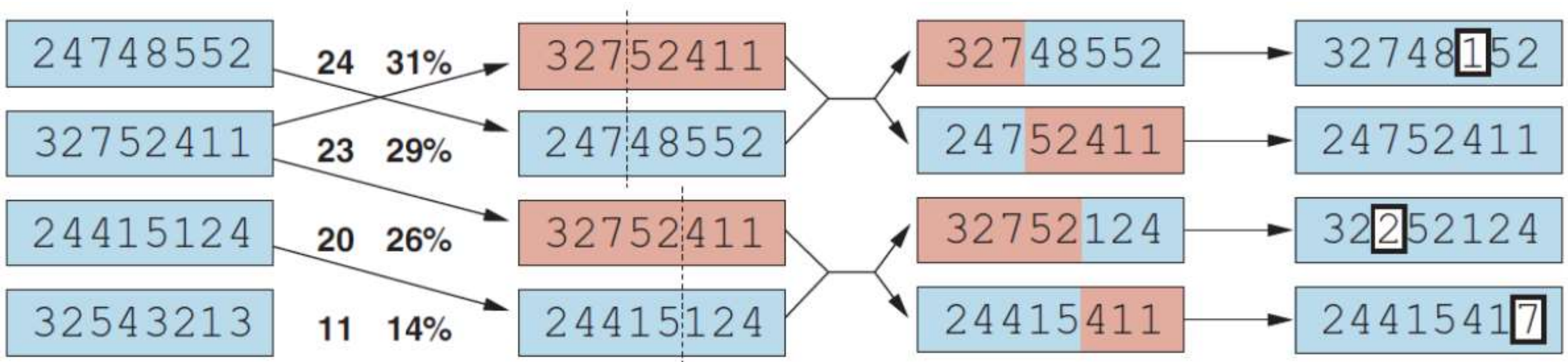
- Main idea: keep track of k states rather than 1

1. Start with $k$ randomly generated states.
2. Generate all their successors.
3. Stop if any successor is a goal.
4. Otherwise, keep the $k$ best successors and go back to step 2.

➢ stochastic beam search which chooses successors with probabilities proportional to their values.

# Genetic Algorithms

1. Start with a population of $k$ randomly generated states (individuals).
2. Select *most fit* individuals to become parents of the next generation
3. Combine every $\rho$ parents to form an offspring (typically $\rho = 2$).
4. Go back to step 2 and repeat until *sufficiently fit* states are discovered (in which case the best one is chosen as a solution).



# Local Search in Continuous Space

## Gradient Descent

1. Start at an initial state $\boldsymbol{x}$=$\boldsymbol{x}$
2. Calculate $\nabla f(\boldsymbol{x})$
3. Update: $\boldsymbol{x} \leftarrow \boldsymbol{x} - \alpha \nabla f(\boldsymbol{x})$ (Gradient ascent: $\boldsymbol{x} \leftarrow \boldsymbol{x} + \alpha \nabla f(\boldsymbol{x})$ )
4. Go back to step 2, until reach the optimal point

## Newton-Raphson Method

$$\boldsymbol{x} \leftarrow \boldsymbol{x} - H_f^{-1}(\boldsymbol{x})(\nabla f(\boldsymbol{x}))^T$$

Hessian of $f$: matrix $\left(\frac{\partial^2 f}{\partial x_i \partial x_j}\right)$

# Nondeterministic Actions

- The agent may not know the current state in a partially observable environment.
- The agent may not know the next state after taking an action in a nondeterministic environment.

Solution: Conditional Plan (depending on what percepts the agent receives while executing the plan)