Search 2

Tanel Tammet, Juhan Ernits
Institute of Computer Science
Tallinn University of Technology
tanel.tammet@ttu.ee juhan.ernits@ttu.ee
2016

Outline

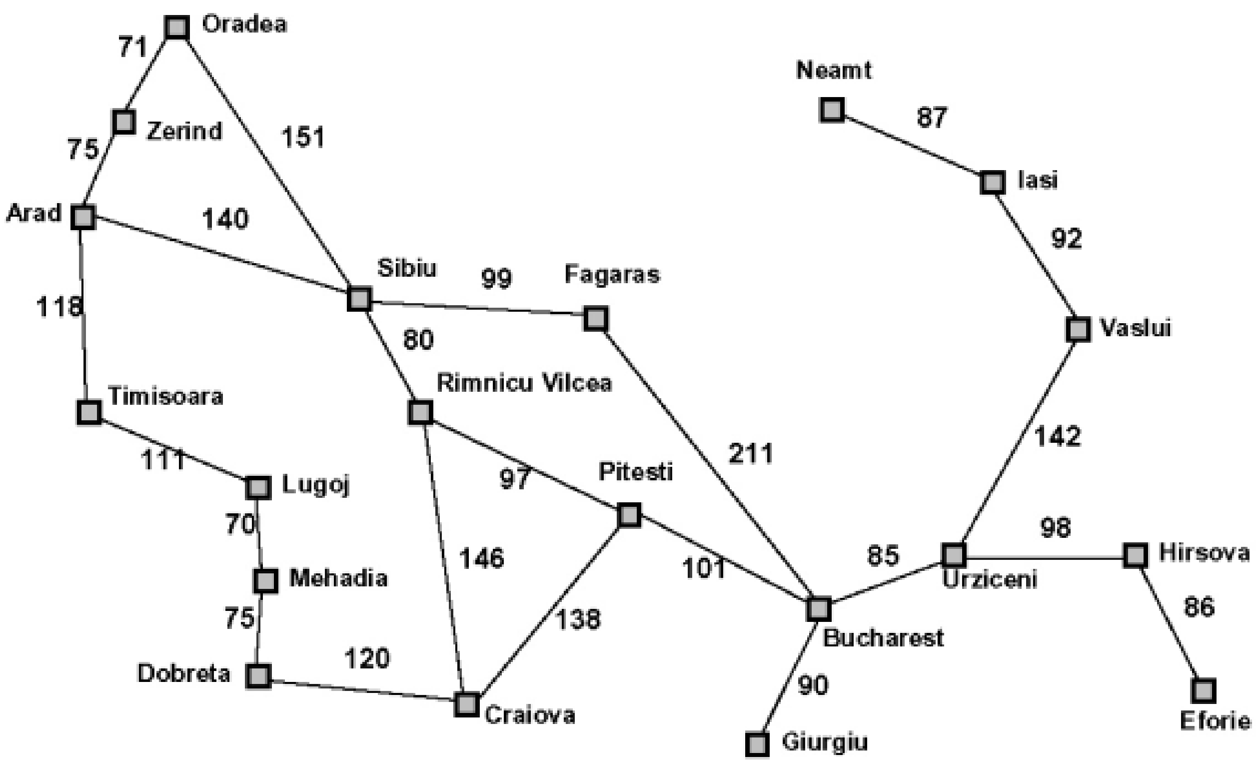
- Informed (Heuristic) search strategies
  - (Greedy) Best-first search
  - A* search
- (Admissible) Heuristic Functions
  - Relaxed problem
  - Subproblem
- Local search algorithms
  - Hill-climbing search
  - Simulated anneal search
  - Local beam search
  - Genetic algorithms
- Online search *
  - Online local search
  - learning in online search

Informed search strategies

- Informed search
  - uses problem-specific knowledge beyond the problem definition
  - finds solution more efficiently than the uninformed search
- Best-first search
  - uses an *evaluation function* $f(n)$ for each node
    - e.g., Measures distance to the goal – lowest evaluation
  - Implementation:
    - Fringe is a queue sorted in increasing order of f-values.
  - Can we really expand the best node first?
    - No! only the one that appears to be best based on $f(n)$.
  - *heuristic function* $h(n)$
    - estimated cost of the cheapest path from node n to a goal node
  - Specific algorithms
    - greedy best-first search
    - A* search

Greedy best-first search

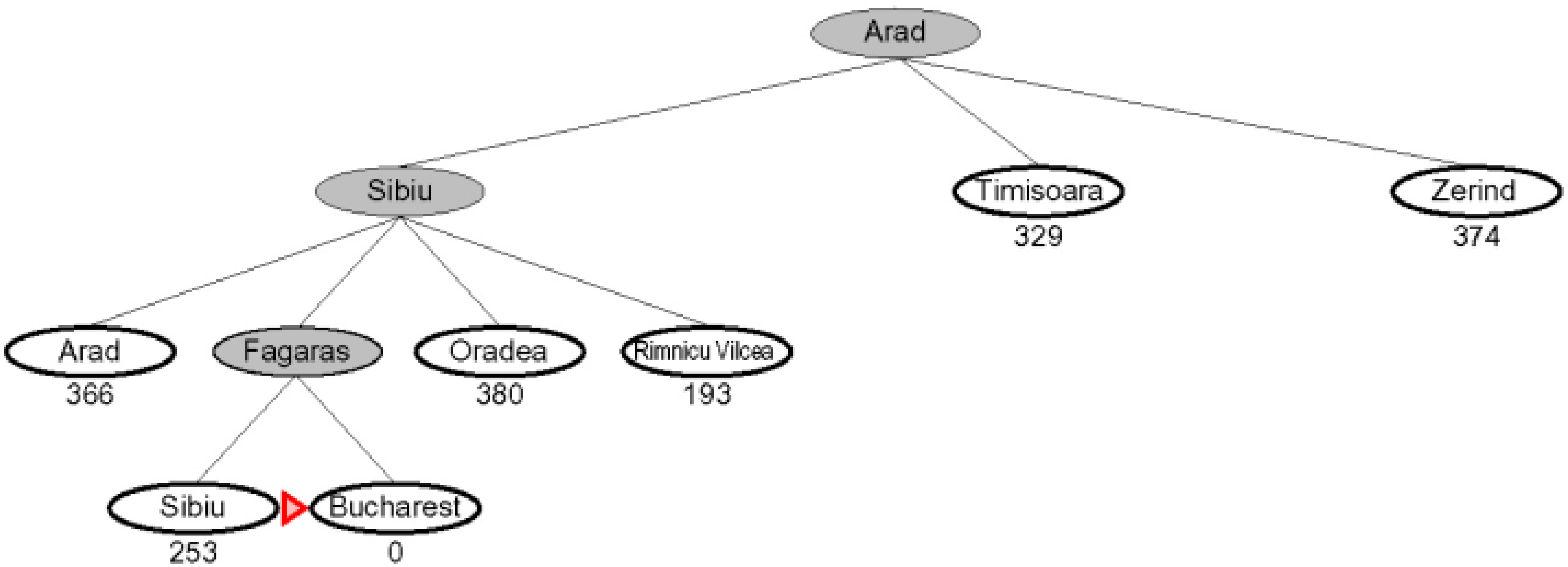
- expand the node that is closest to the goal
- $f(n) = h_{SLD}(n)$ *Straight line distance* heuristic



Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy best-first search example



Properties of Greedy best-first search

- Complete?

  No – can get stuck in loops, e.g., **Iasi** –> **Neamt** –> **Iasi** –> **Neamt**

  Yes – complete in finite states with repeated-state checking

- Optimal?

  No

- Time?

  $O(b^m)$, but a good heuristic function can give dramatic improvement

- Space?

  $O(b^m)$ – keeps all nodes in memory

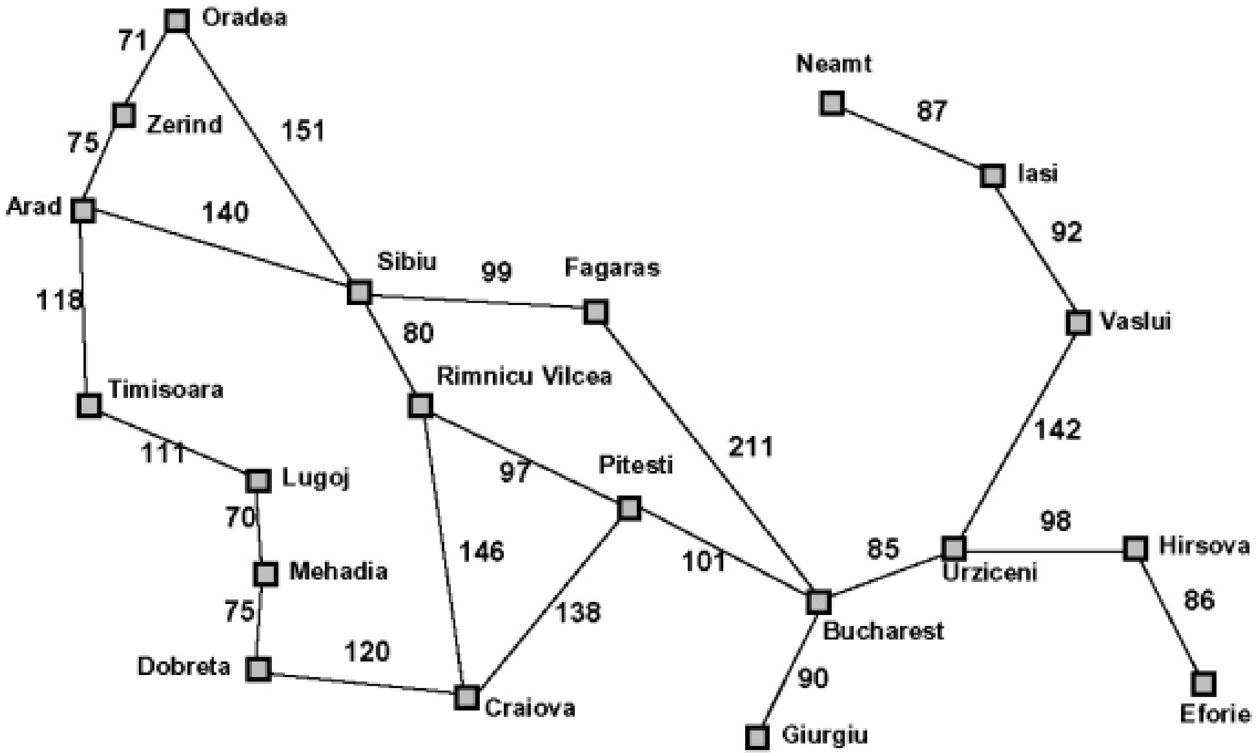
A* search

- evaluation function $f(n) = g(n) + h(n)$
  - $g(n)$ = cost to reach the node
  - $h(n)$ = estimated cost to the goal from n
  - $f(n)$ = estimated total cost of path through n to the goal

- an admissible (optimistic) heuristic
  - never overestimates the cost to reach the goal
  - estimates the cost of solving the problem is less than it actually is
  - e.g., $h_{SLD}(n)$ never overestimates the actual road distances

- A* using Tree-Search is optimal if $h(n)$ is admissible
- could get suboptimal solutions using Graph-Search
  - might discard the optimal path to a repeated state if it is not the first one generated
  - a simple solution is to discard the more expensive of any two paths found to the same node (extra memory)

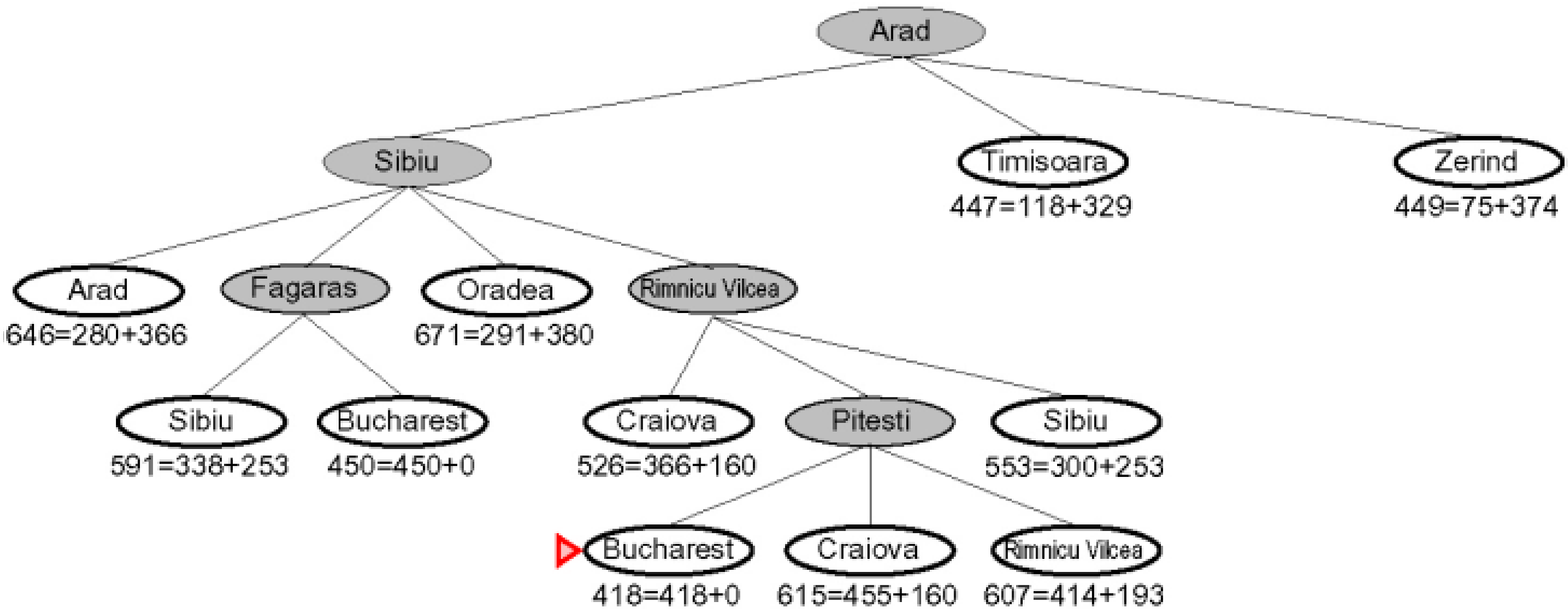
$h_{SLD}(n)$: *Straight line distance* heuristic



Straight−line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

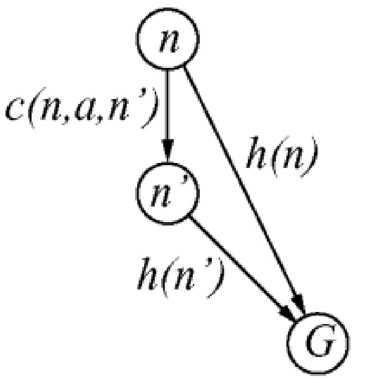
A* search example



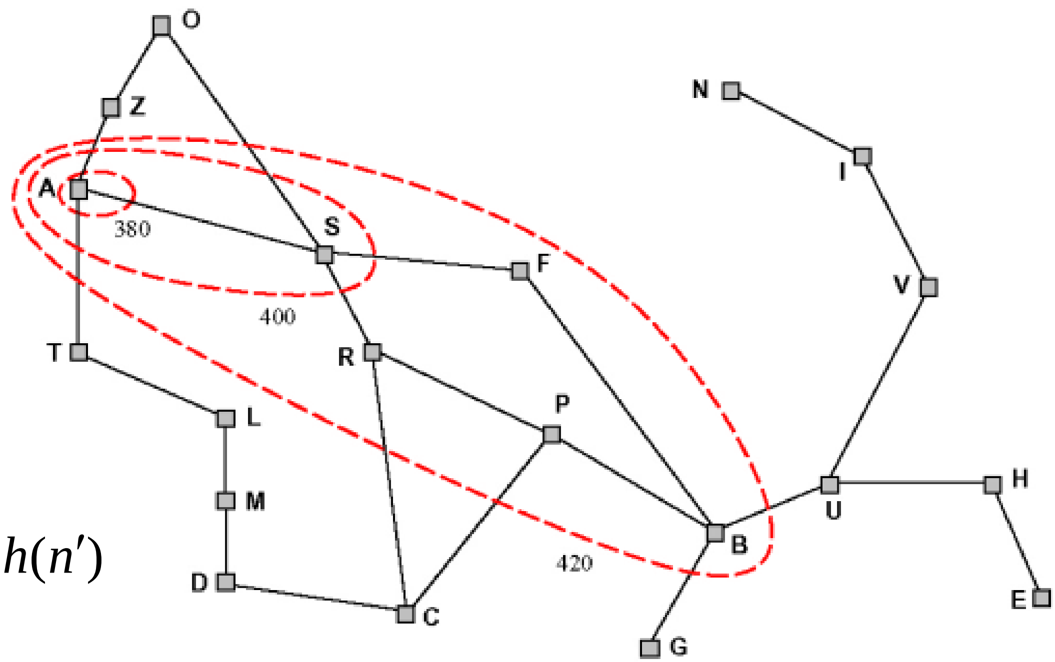
Optimality of A*

- Consistency (monotonicity) $h(n) \le c(n,a,n') + h(n')$
  - *n'* is any successor of *n,* general triangle inequality (*n*, *n'*, and the goal)
  - consistent heuristic is also admissible

- A* using Graph-Search is optimal if *h*(*n*) is consistent
  - the values of *f*(*n*) along any path are nondecreasing



$$\begin{aligned} f(n') &= g(n') + h(n') \\ &= g(n) + c(n,a,n') + h(n') \\ &\ge g(n) + h(n) = f(n) \end{aligned}$$



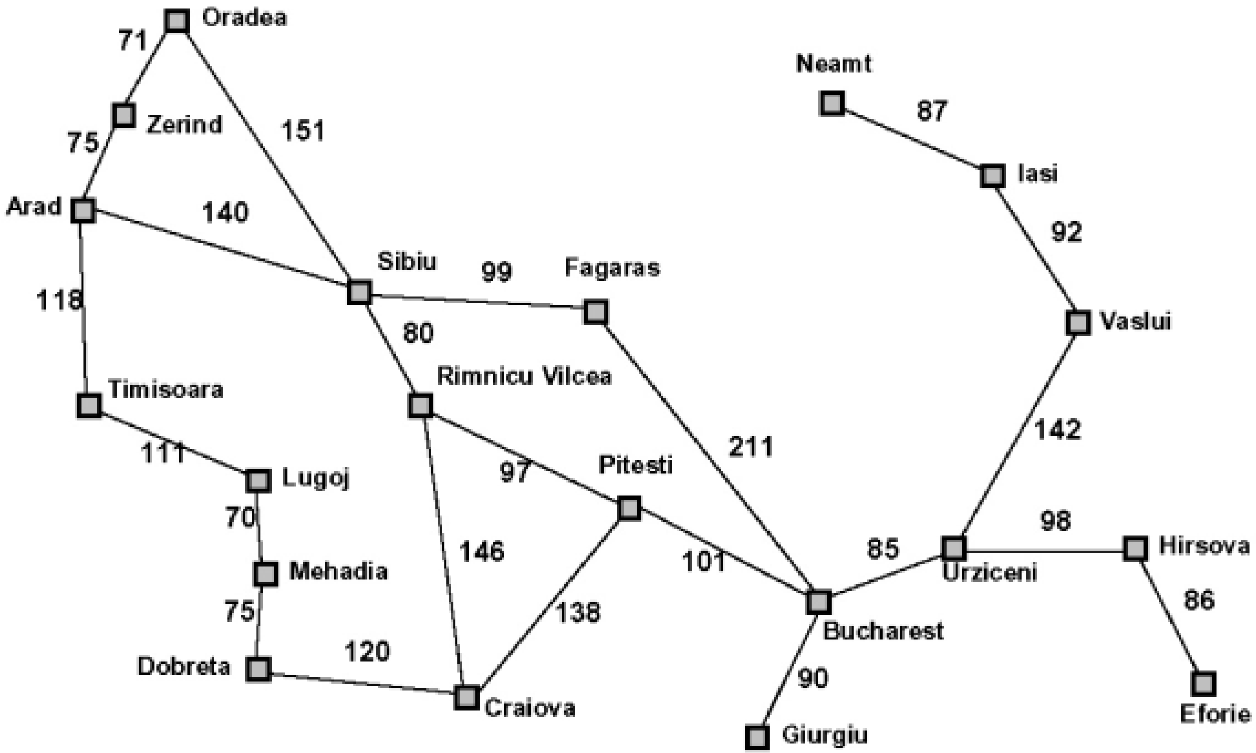
Properties of A*

- Suppose C^* is the cost of the optimal solution path
  - A* expands all nodes with $f(n) < C^*$
  - A* might expand some of nodes with $f(n) = C^*$ on the "goal contour"
  - A* will expand no nodes with $f(n) > C^*$, which are pruned!
  - Pruning: eliminating possibilities from consideration without examination
- A* is optimally efficient for any given heuristic function
  - no other optimal algorithm is guaranteed to expand fewer nodes than A*
  - an algorithm might miss the optimal solution if it does not expand all nodes with $f(n) < C^*$
- A* is complete
- Time complexity
  - exponential number of nodes within the goal contour
- Space complexity
  - keeps all generated nodes in memory
  - runs out of space long before runs out of time

Memory-bounded heuristic search

- Iterative-deepening A* (IDA*)
  - uses *f*-value ($g + h$) as the cutoff
- Recursive best-first search (RBFS)
  - replaces the *f*-value of each node along the path with the best *f*-value of its children
  - remembers the *f*-value of the best leaf in the "forgotten" subtree so that it can reexpand it later if necessary
  - is efficient than IDA* but generates excessive nodes
  - changes mind: go back to pick up the second-best path due to the extension (*f*-value increased) of current best path
  - optimal if $h(n)$ is admissible
  - space complexity is $O(bd)$
  - time complexity depends on the accuracy of $h(n)$ and how often the current best path is changed
- Exponential time complexity of Both IDA* and RBFS
  - cannot check repeated states other than those on the current path when search on Graphs – Should have used more memory (to store the nodes visited)!

$h_{SLD}(n)$: *Straight line distance* heuristic

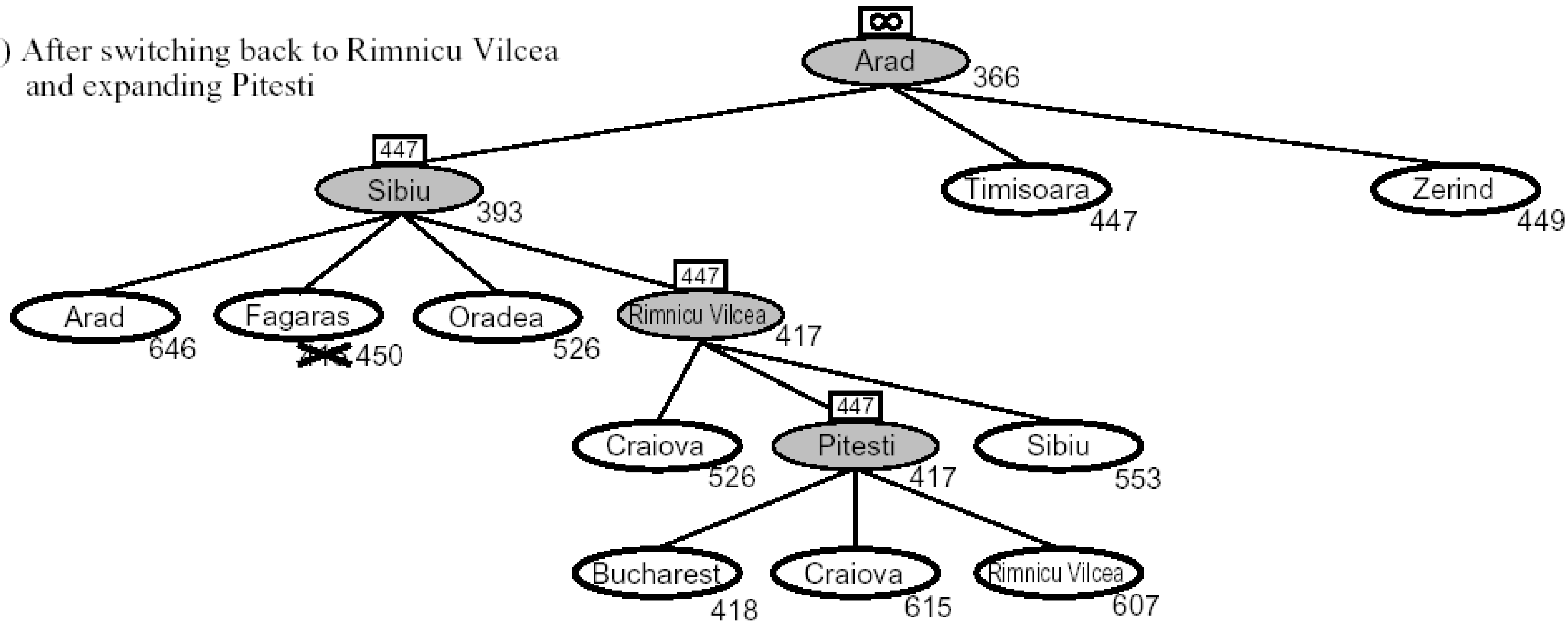


Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

RBFS example

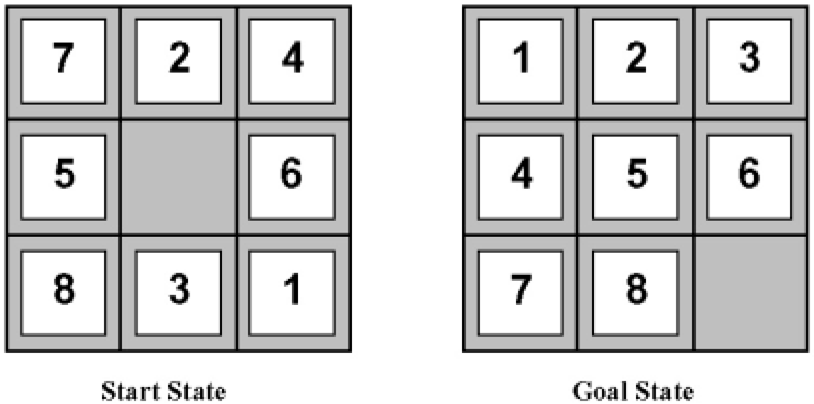
(c) After switching back to Rimnicu Vilcea
and expanding Pitesti



Memory-bounded heuristic search (cont'd)

- SMA* – Simplified MA* (Memory-bounded A*)
  - expands the best leaf node until memory is full
  - then drops the worst leaf node – the one has the highest *f*-value
  - regenerates the subtree only when all other paths have been shown to look worse than the path it has forgotten
  - complete and optimal if there is a solution reachable
  - might be the best general-purpose algorithm for finding optimal solutions
- If there is no way to balance the trade off between time an memory, drop the optimality requirement!

(Admissible) Heuristic Functions



$h_1(n)$ = the number of misplaced tiles

$h_2(n)$ = total **Manhattan** (city block) distance

h_1? = 7 tiles are out of position

h_2? = 4+0+3+3+1+0+2+1 = 14

Effect of heuristic accuracy

- Effective branching factor b^*
  - total # of nodes generated by A* is N, the solution depth is d
  - b^* is b that a uniform tree of depth d containing N+1 nodes would have

$$N + 1 = 1 + b^* + (b^*)^2 + \ldots + (b^*)^d$$

  - well-designed heuristic would have a value close to 1
  - h_2 is better than h_1 based on the b^*

- Domination
  - h_2 dominates h_1 if $h_2(n) \geq h_1(n)$ for any node n
  - A* using h_2 will never expand more nodes than A* using h_1

    every node n with $f(n) < C^*$ will be expanded

$$f(n) = g(n) + h(n) < C^* \quad \Rightarrow \quad h(n) < C^* - g(n)$$

$$\Rightarrow \quad h_1(n) \leq h_2(n) < C^* - g(n)$$

  - the larger the better, as long as it does not overestimate!

Inventing admissible heuristic functions

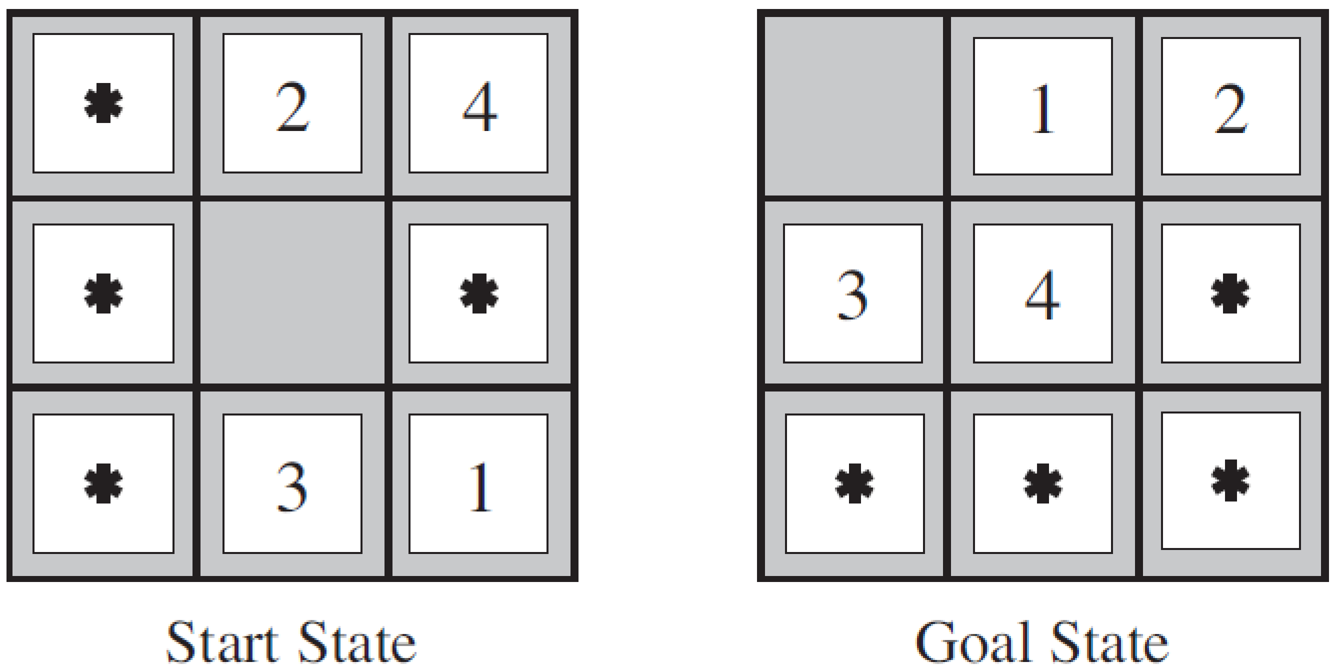
- h_1 and h_2 are solutions to relaxed (simplified) version of the puzzle.
  - If the rules of the 8-puzzle are relaxed so that a tile can move anywhere, then h_1 gives the shortest solution
  - If the rules are relaxed so that a tile can move to any adjacent square, then h_2 gives the shortest solution
- Relaxed problem: A problem with fewer restrictions on the actions
  - Admissible heuristics for the original problem can be derived from the optimal (exact) solution to a relaxed problem
  - Key point: the optimal solution cost of a relaxed problem is no greater than the optimal solution cost of the original problem
  - Which should we choose if none of the h_1 … h_m dominates any of the others?

    We can have the best of all worlds, i.e., use whichever function is most accurate on the current node

$$h(n) = \max\{h_1(n), \ldots, h_m(n)\}$$

- Subproblem *
  - Admissible heuristics for the original problem can also be derived from the solution cost of the subproblem.
- Learning from experience *

Example of subproblems in 8-puzzle



- Acknowledgements

- This set of slides contains several prepared by Hwee Tou Ng and Stuart Russell, available from the AIMA pages.