UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Recursive Best First Search

Alfons Juan
Jorge Civera

DSIC

Departament de Sistemes
Informàtics i Computació

Objectives

- To apply the RBFS algorithm.
- To build the RBFS search tree.
- To analyse the optimality and complexity of RBFS search.

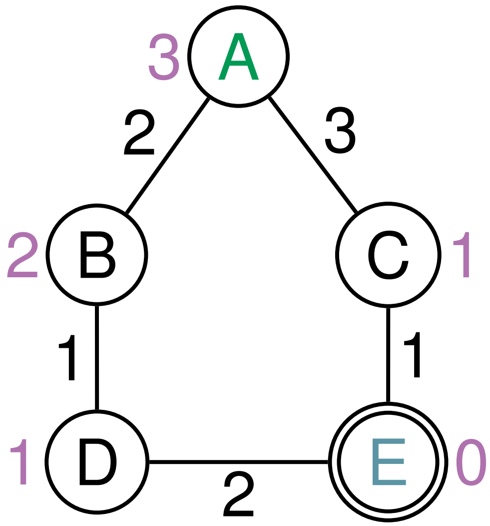
Contents

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

1 Introduction

RBFS search based on bounding BT search with an f value, but, unlike IDA*, it guarantees BF for non-consistent evaluation functions

State graph



Search

A 0+3	$F_A = 3$	bound = ∞
AB 2+2	$F_B = 4$	
AC 3+1	$F_C = 4$	bound = 4
ABD 3+1	$F_D = 4$	
ABDE 5+0	$F_E = 5$	
ABD	$F_D = 5$	
AB	$F_B = 5$	
AC	$F_C = 4$	bound = 5
ACE 4+0	$F_E = 4$	
ACE sol.!		

RBFS obtains the bound from the second-best f value of sibling nodes in the path being explored.

2 The RBFS algorithm (main) [1]

RBFS(G, s', f) // G weighed graph, s' start, ***evaluation function*** f

$P = InitStack(s')$ // Init ***Path*** with source node

$b = \infty$ // Init bound

$F_{s'} = f_{s'}$ // Stored value is initialised to f value

$(F_r, r) =$ **BT**$(G, P, F_{s'}, f, b)$ // Return goal state and its stored value

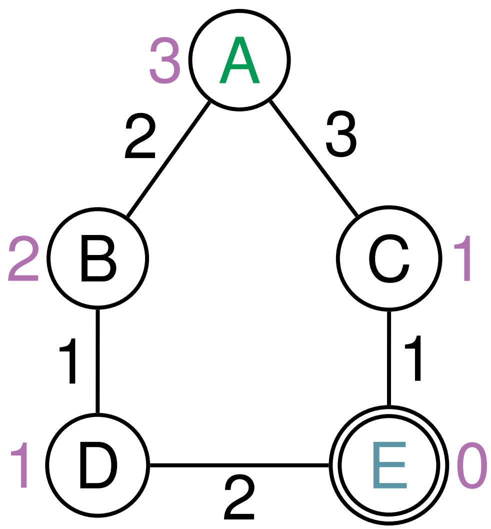
if $r \neq$ NULL: **return** P // If solution, return ***Path*** to goal

The RBFS algorithm (backtracking) [1]

```
BT(G, P, F_s, f, b)                              // G graph, Path P, stored value F_s', f, bound b
  s = Top(P)                                     // Path: extract node from stack
  if Goal(s): return (f_s, s)                    // Solution found!
  O = InitQueue()                                // Open: priority queue for child nodes
  for all (s, n) ∈ Adjacents(G, s) and n ∉ P:    // Generating child n not in the Path
     if f_s < F_s : F_n = max(f_n, F_s)          // If s visited, child may inherit stored value
     else: F_n = f_n                             // Otherwise, stored value is f value
     Push(O, n, F_n)                             // Sorting children by stored value in priority queue
  if EmptyQueue(O): return (∞, NULL)             // No children, bound = ∞
  while True:
     (n, F_n) = Top(O)                           // Best child according to stored value F
     if F_n > b: return (F_n, NULL)              // Exceeding bound, backtracking
     (n', F_n') = Top2(O)                        // 2nd best F or if it does not exist, then F_n' = ∞
     Push(P, n)                                  // Add child to the Path being explored
     (F_n, r) = BT(G, P, F_n, f, min(b, F_n'))   // Recursive call with possible new bound
     if r ≠ NULL: return (F_n, r)                // If sol. found, out of recursion without update
     Update(O, n, F_n)                           // Update node n in O
    Pop(P)                                       // Discard last child from Path
```

3 RBFS search space

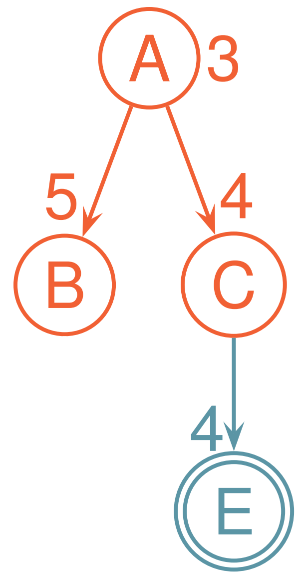
State graph



RBFS

$b = \infty$

$b = 5$

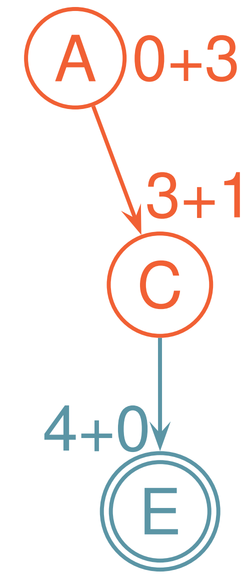


Path: A3C4B5E4

IDA*

$b = 4$

$min = 5$



It 2: A3C4E4

4 Properties

- A node is already visited when $f_s < F_s$, otherwise $f_s = F_s$
  - $f_s < F_s$: Child inherits parent's stored value if $f_n < F_s$
  - $f_s = F_s$: Child's stored value is f_n in first exploration
- Bound is updated when going into the recursion
  - Minimum between current bound and 2nd best child stored value
- F_n is the minimum ***f*** value of the expanded subtree below n
  - F_n is updated when going out the recursion
- New nodes explored in BF order, even for non-consistent ***f*** function
- Backtracking implementation only prevents cycles in the ***Path***

5 Optimality and complexity

- ***Completeness:*** As A* always finishes in finite graphs.
- ***Optimality:*** As BF, it depends on the evaluation function.
- ***Space complexity:*** $O(bd)$
- ***Temporal complexity:*** As IDA*, $O(b^d)$, in practice:
  - A subset of nodes are re-expanded at each iteration
  - Need of ***Open*** priority queue for children of each node
  - More time efficient than IDA*, re-expansion from 2nd best

6 Conclusions

We have studied:

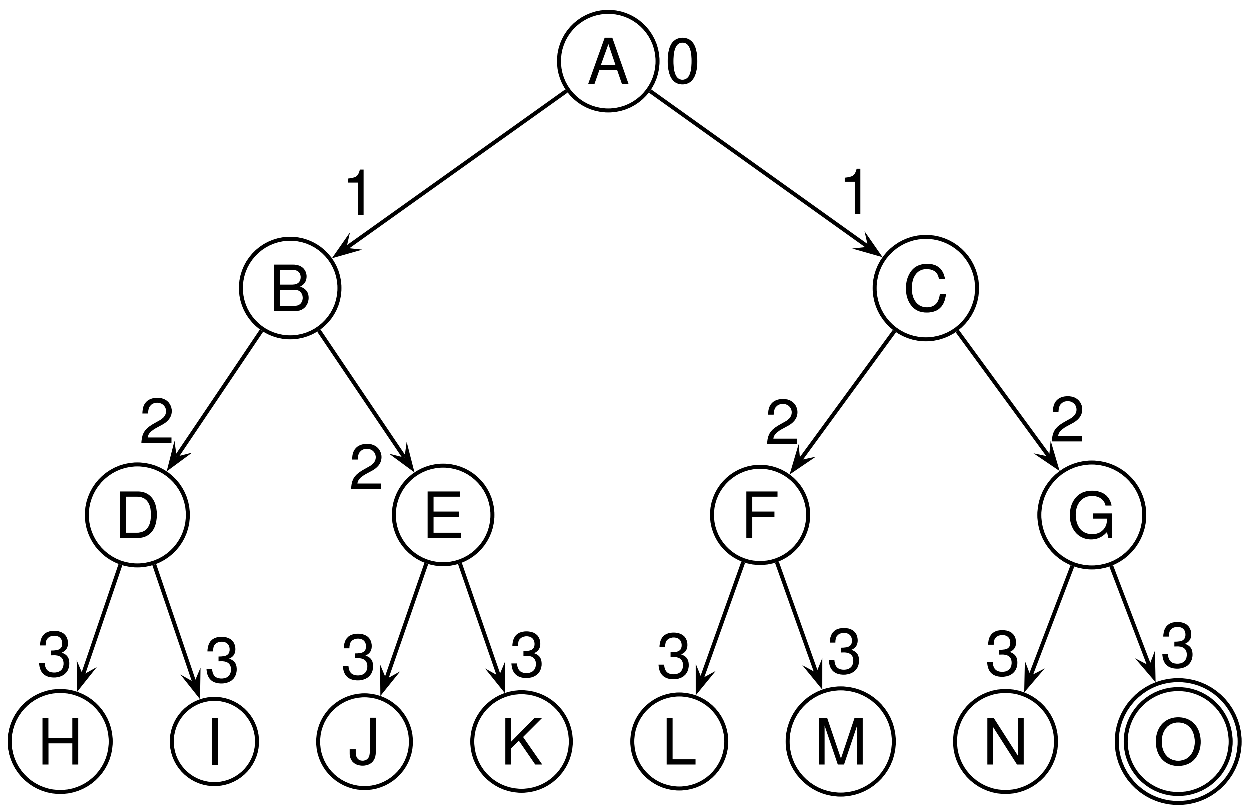
- The RBFS algorithm.
- The RBFS search space.
- Properties, optimality and complexity in RBFS search.

Some aspects to highlight on RBFS:

- Complete and optimum if $f = g + h$ where h is admissible.
- Reduced spatial cost but more than IDA*
- Temporal cost depends on evaluation function ***f***

RBFS exercise

Run a trace of RBFS on the state space below (f-value next to each node) and answer the questions on the bottom:



- Maximum number of nodes in memory?
- Total number of nodes generated?

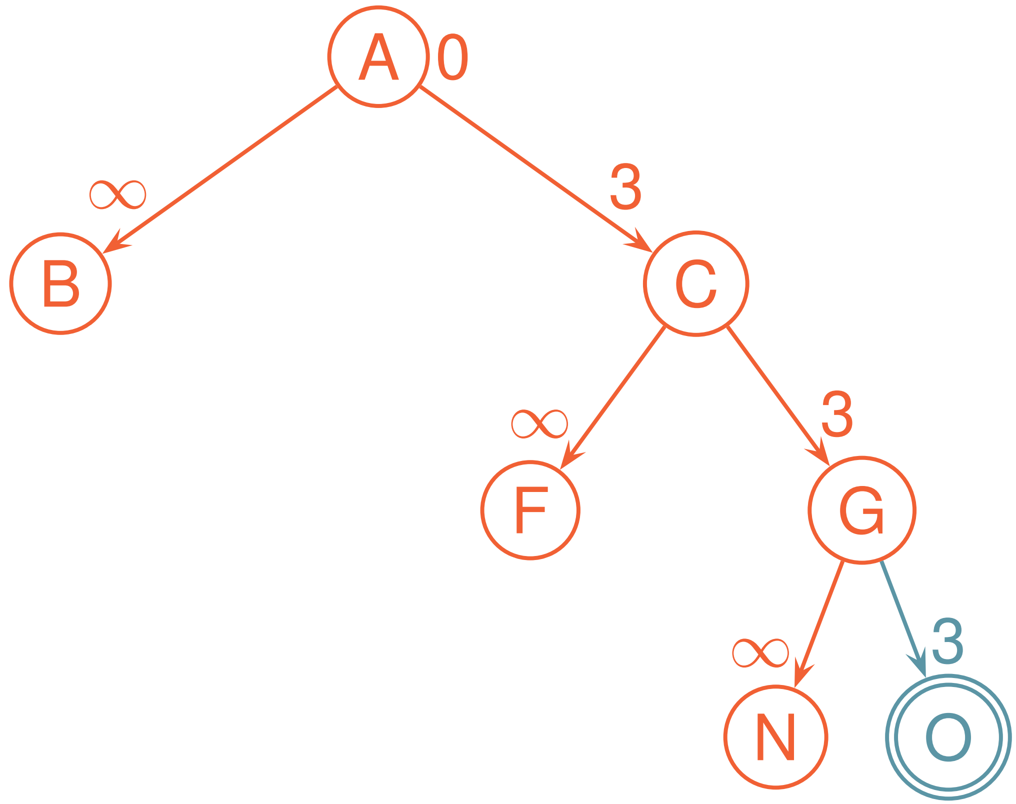
RBFS solution

$b = \infty$

$b = \infty$

$b = \infty$

$b = 3$



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

References

[1] Richard E. Korf. Linear-space best-first search. *Artificial Intelligence*, 62(1):41–78, 1993.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA