Iterative Deepening A* Search

Alfons Juan
Jorge Civera

Departament de Sistemes
Informàtics i Computació

Objectives

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

Contents

1 Introduction 3

2 The IDA* algorithm 4

3 IDA* search space 6

4 Optimality and complexity 7

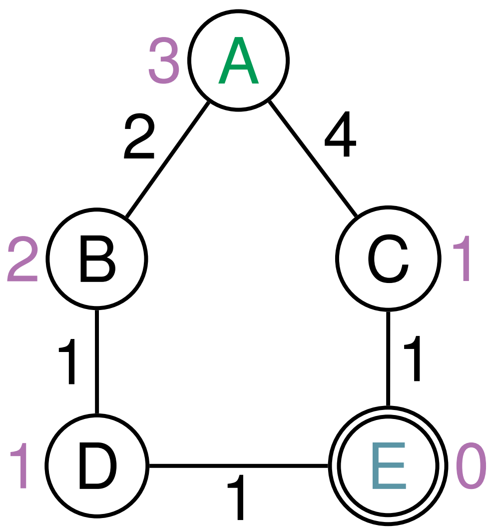
5 Conclusions 8

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

1 Introduction

IDA* search is based on IDS with backtracking using an f value to bound the search at each iteration, instead of a maximum depth:

State graph



Search

A 0+3 bound = 3

AB 2+2
AC 4+1

A 0+3 bound = 4

AB 2+2
ABD 3+1

ABDE 4+0 sol.!

IDA* computes the next bound as the minimum f value of those nodes exceeding the current bound.

2 The IDA* algorithm (main) [1]

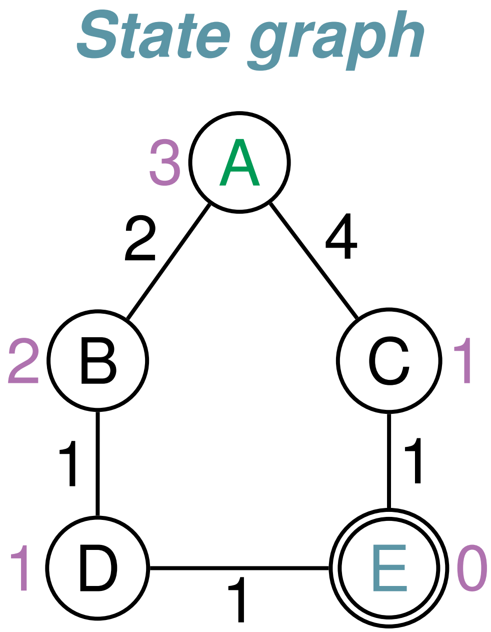
```
IDA(G, s', h)                                   // G weighed graph, s' start, h heuristic
  P = InitStack(s')                             // Init Path with source node
  b = h(s')                                     // Init bound with f_s' = h(s')
  while True:
    (nextb, r) = BT(G, P, h, b)                 //BT returns next bound and goal state
    if r ≠ NULL: return P                       // if solution, return Path to goal
    if nextb = ∞: return NULL                   // no children to compute next bound
    b = nextb                                   // bound updated for next iteration
```

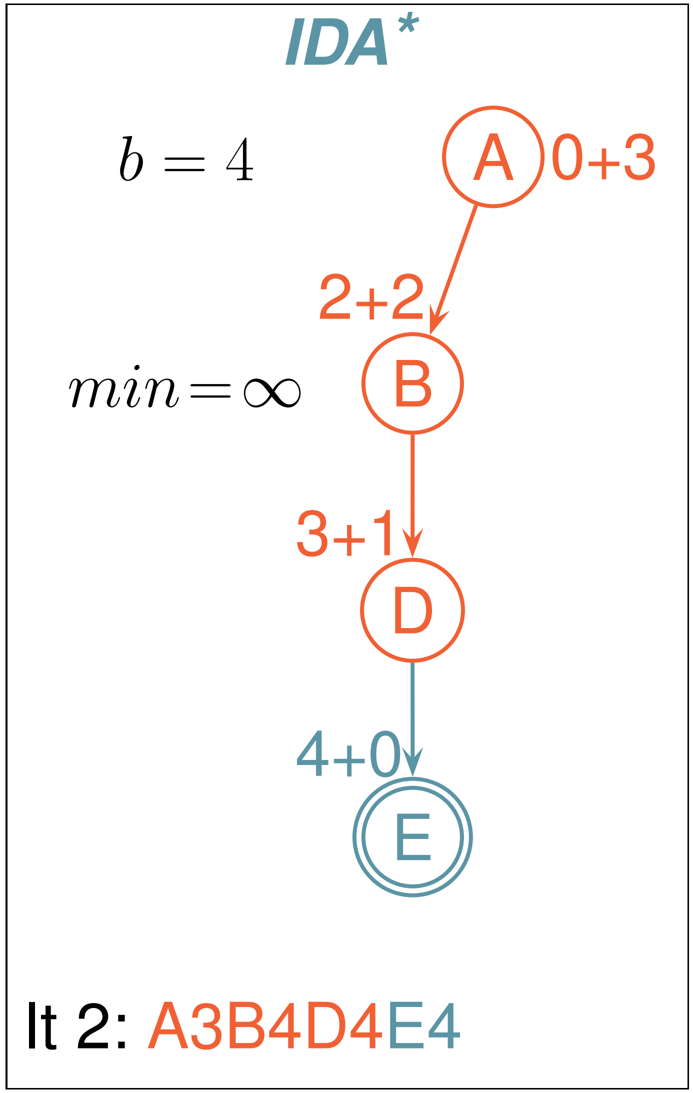

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

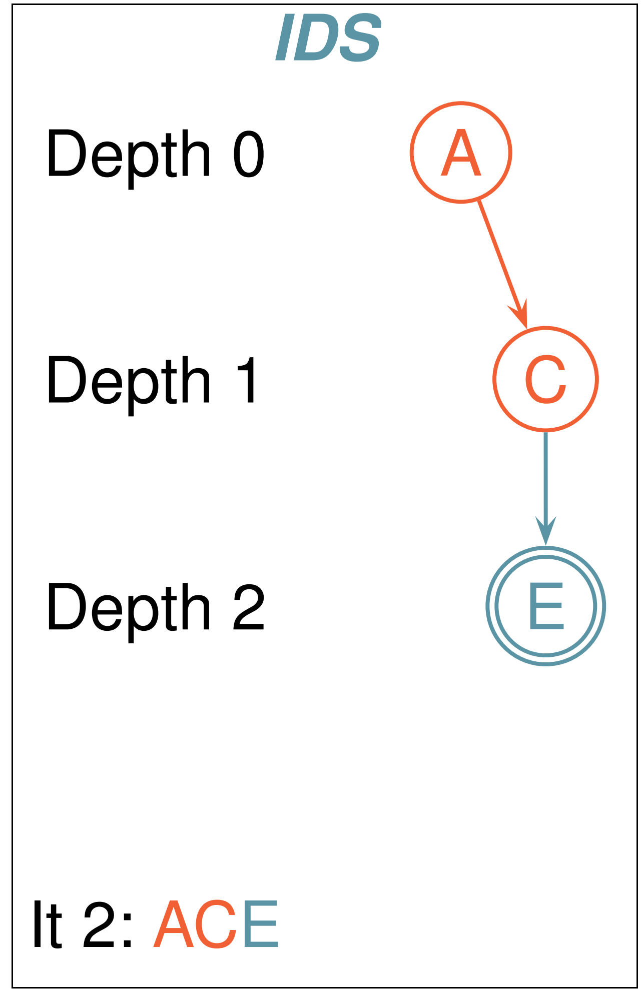
The IDA* algorithm (backtracking) [1]

```
BT(G, P, h, b)                            // G weighed graph, Path P, h, bound b
  s = Top(P)                              // Path: extract node from stack
  f_s = g_s + h(s)                        // f value of node being explored
  if f_s > b: return (f_s, NULL)          // b exceeded return to compute nextb
  if Goal(s): return (f_s, s)             // solution found!
  min = ∞                                 // children's minimum f value
  n = FirstAdjacent(G, s)                 // generation: n first child of s
  while n ≠ NULL:                         // while there are children left to explore
    if n ∉ P:                             // n not in Path to avoid loops
      Push(P, n)                          // add child to the Path being explored
      (nextb, r) = BT(G, P, h, b)         // child returns min f and goal state
      if r ≠ NULL: return (nextb, r)      // if r solution, get out recursion
      if nextb < min: min = nextb         // update minimum f value
      Pop(P)                              // Discard last child from Path
    n = NextAdjacent(G, s, n)             // generation: n next child of s
  return (min, NULL)                      // sol. not found, return minimum f
```

3 IDA* search space







4 Optimality and complexity

- ***Completeness:*** As A*, it always finishes in finite graphs.
- ***Optimality:*** If h is admissible, IDA* returns the optimal solution. IDA* expands nodes in increasing f value.
- ***Space complexity:*** As IDS with backtracking, $O(d)$
- ***Temporal complexity:*** As A*, $O(b^d)$, in practice:
  - A subset of nodes are re-expanded at each iteration
  - Iterations depends on number of nodes with different f value
  - No need of ***Open*** priority queue or ***Closed*** set

5 Conclusions

We have studied:

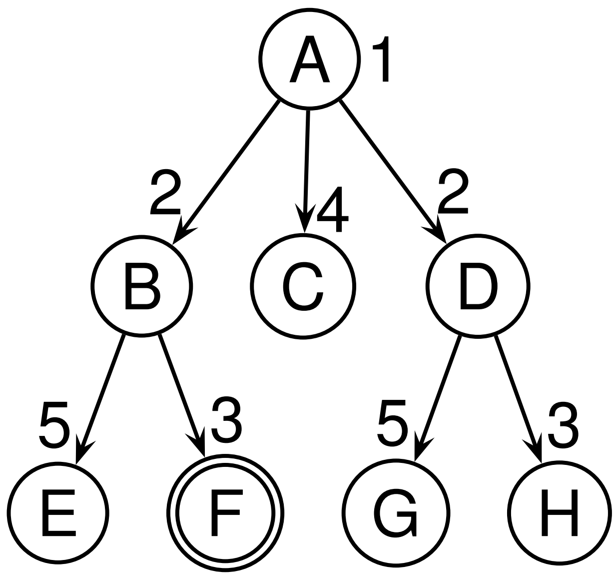
- The IDA* algorithm.
- The IDA* search space.
- Optimality and complexity in IDA* search.

Some aspects to highlight on IDA*:

- Complete and optimum with positive-cost edges and h admissible.
- Reduced spatial cost thanks to backtracking.
- Temporal cost depends on evaluation function ***f***

IDA* exercise

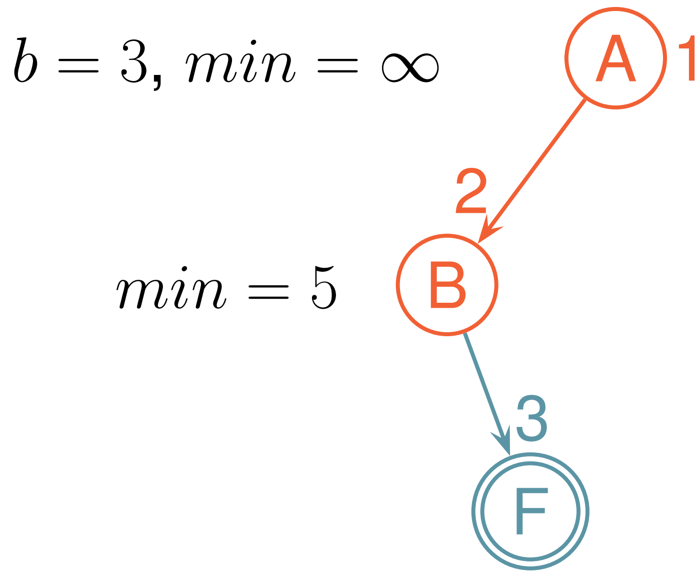
f-value next
to each node



Run a trace of IDA* on the state space on the left and answer the following questions:

- Number of iterations to find solution?
- Maximum number of nodes in memory?
- Total number of nodes generated?

IDA* solution



It 3: nodes generated = 14 sol.!

References

[1] R. E. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 27:97–109, 1985.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

IDS with backtracking

IDS(G, s) // ***Iterative deepening search***
  for $m = 0, 1, 2, \ldots$: **if** $(r = \text{BT}(G, s, m)) \neq$ NULL: **return** r

BT(G, s, m) // ***Backtracking*** with maximum depth of m
  if $Goal(s)$ **return** s // solution found!
  if $m = 0$ **return** NULL // maximum depth reached
  $n = FirstAdjacent(G, s)$ // generation: n first child of s
  while $n \neq$ NULL:
    $r = \text{BT}(G, n, m - 1)$ // current child result
    if $r \neq$ NULL: **return** r // if r is solution, stop
    $n = NextAdjacent(G, s, n)$ // generation: n next child of s
  return NULL // no solution found