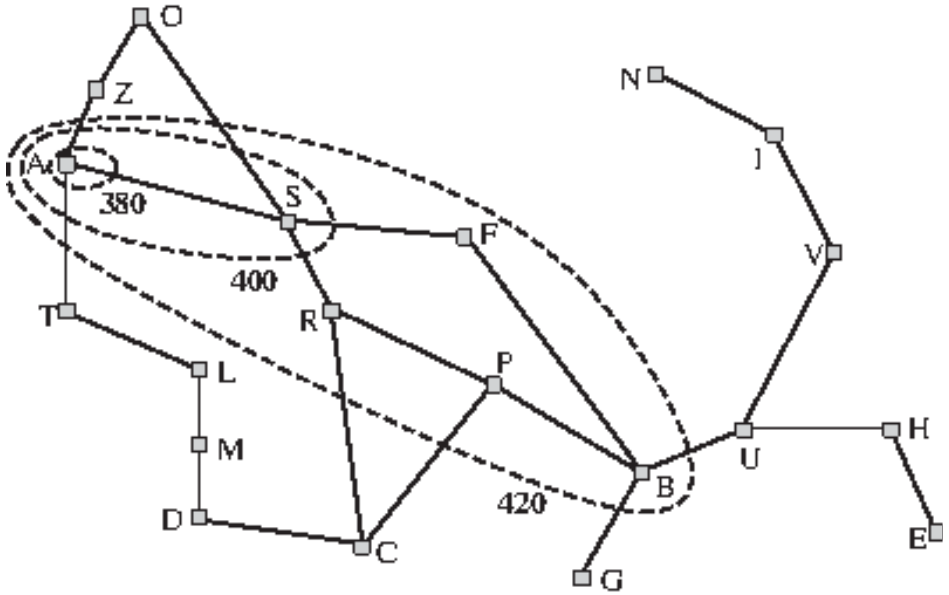
Busca com Memória Limitada

- IDA^*: busca A^* em profundidade iterativa.
- $RBFS$: Recursive-bounded best-first search.
- SMA^*: Simplified Memory-bounded A*.

Busca com Memória Limitada: IDA^*

- IDA^*: tenta encontrar soluções iterativamente variando o valor da função de custo.



- procura solução com custo f. Se não encontrar, retorna novo f (f1) e continua procurando solução com custo $f1$, e assim por diante.
- IDA^* é completa e ótima da mesma forma que A^*, mas como a busca é feita em profundidade utiliza menos espaço, que é proporcional ao tamanho do caminho mais longo explorado.

Busca com Memória Limitada: IDA^*

```
function IDA*(problem) return solution
      root <- MAKE_NODE(INITIAL_STATE[problem])
      f_limit <- fcost(root)
      loop
         solution,f_limit <- DFS_CONTOUR(root,f_limit)
         if solution is non-null then return solution
         if f_limit = infinito then return failure
      end
```

Busca com Memória Limitada: IDA^*

```
function DFS_CONTOUR(node,f_limit) return solution and a new cost
next_f ← infinito
if fcost(node) > f_limit then
    return null,fcost(node)
end if
if GOAL_TEST[problem](STATE(node)) then
    return node,f_limit
end if
for each node S in successors(node) do
    solution,new_f ← DFS_CONTOUR(S,f_limit)
    if solution is non-null then
        return solution,f_limit
    end if
    next_f ← MIN(next_f,new_f)
end for

return null,next_f
```

Busca com Memória Limitada: IDA^*, Análise

- complexidade espacial: na maioria dos casos no. de nós armazenados $b \times d$.
- no pior caso: $\approx \frac{bf^*}{\delta}$, onde f^* custo da solução ótima e δ custo da operação de valor mínimo.
- em geral: IDA^* passa por 2 ou 3 iterações
- eficiência similar a do A^*
- overhead pode ser menor porque nós não precisam ser inseridos na lista em ordem.

Busca com Memória Limitada: SMA^*

- Simplified Memory-Bounded A^*
- Propriedades:
  - utiliza somente a memória disponível
  - evita estados repetidos se memória permitir
  - é completa se memória suficiente para armazenar o caminho da solução menos profunda
  - é ótima se memória suficiente para armazenar caminho da solução ótima

Busca com Memória Limitada: SMA^*

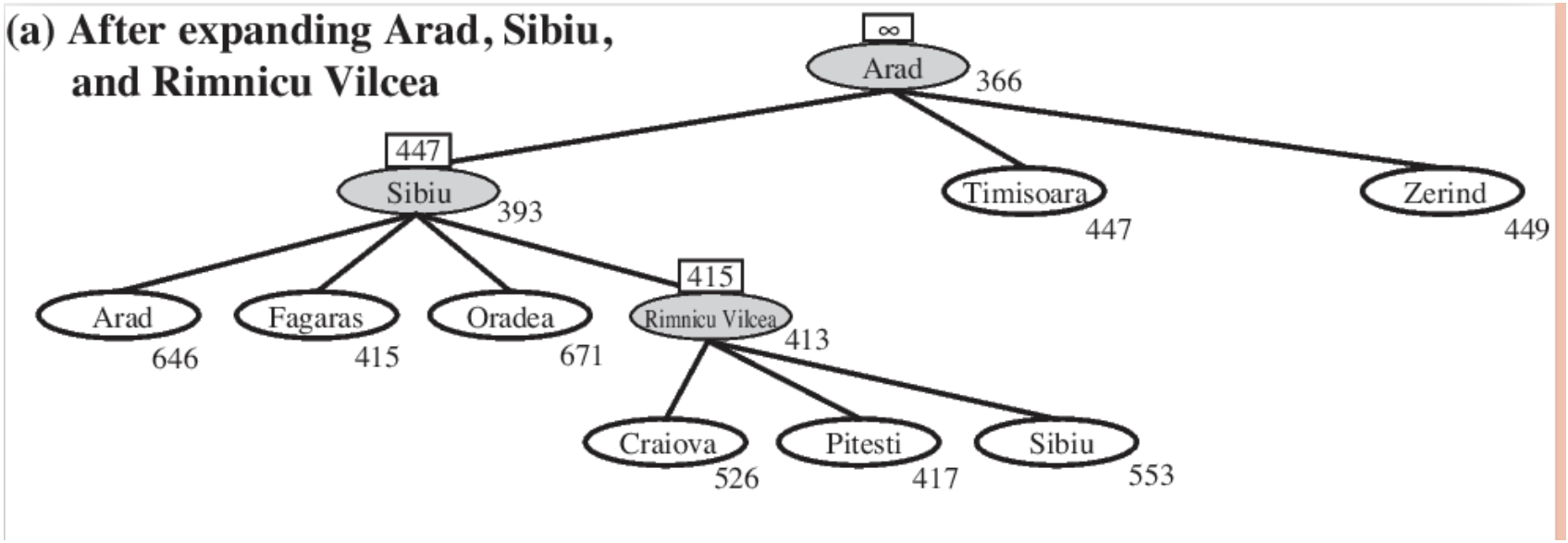
```
function SMA*(problem) return solucao
  Queue <- MAKE_QUEUE(MAKE_NODE(
                       INITIAL_STATE[problem]))
  loop
    if EMPTY?(Queue) then return failure
    n <- no mais profundo de menor custo de Queue
    if GOAL_TEST(n) then return solucao
    s <- NEXT_SUCCESSOR(n)
    if not GOAL_TEST(s) e s em nivel maximo then
       f(s) <- infinito
    else
       f(s) <- max(f(n),g(s)+h(s))
    if todos os sucessores de n foram gerados
       atualiza fcost de n e de todos os
       ancestrais, se necessario
    if SUCCESSORS(n) todos em memoria then
       remove n da Queue
    if memory is full then
       rem. no mais raso e de > custo de Queue
       remover este no da lista de suc. do pai
       inserir o pai em Queue, se necessario
    inserir s em Queue
  end
```

Busca com Memória Limitada: RBFS

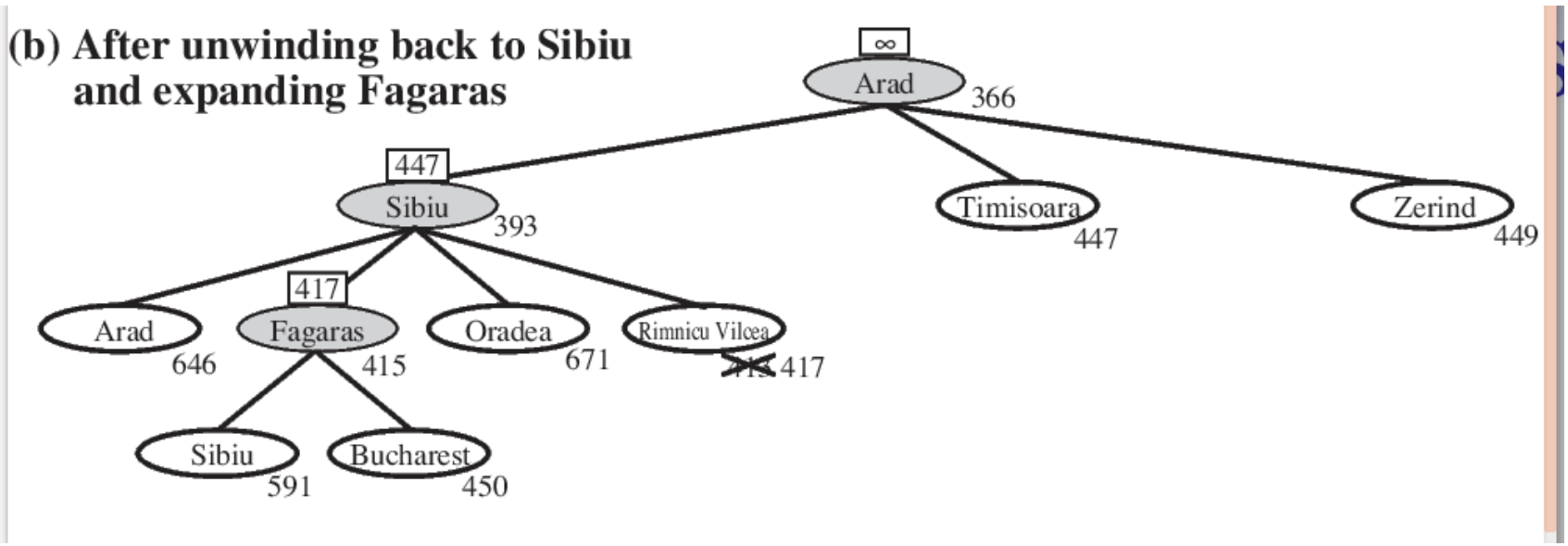
```
function RECURSIVE-BEST-FIRST-SEARCH(problem) returns a solution, or failure
    return RBFS(problem, MAKE-NODE(problem.INITIAL-STATE), ∞)

function RBFS(problem, node, f_limit) returns a solution, or failure and a new f-cost limit
  if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
  successors ← [ ]
  for each action in problem.ACTIONS(node.STATE) do
      add CHILD-NODE(problem, node, action) into successors
  if successors is empty then return failure, ∞
  for each s in successors do /* update f with value from previous search, if any */
      s.f ← max(s.g + s.h, node.f))
  loop do
      best ← the lowest f-value node in successors
      if best.f > f_limit then return failure, best.f
      alternative ← the second-lowest f-value among successors
      result, best.f ← RBFS(problem, best, min(f_limit, alternative))
      if result ≠ failure then return result
```

Figure 3.26 The algorithm for recursive best-first search.

Busca com Memória Limitada: RBFS



Busca com Memória Limitada: RBFS



Busca com Memória Limitada: RBFS