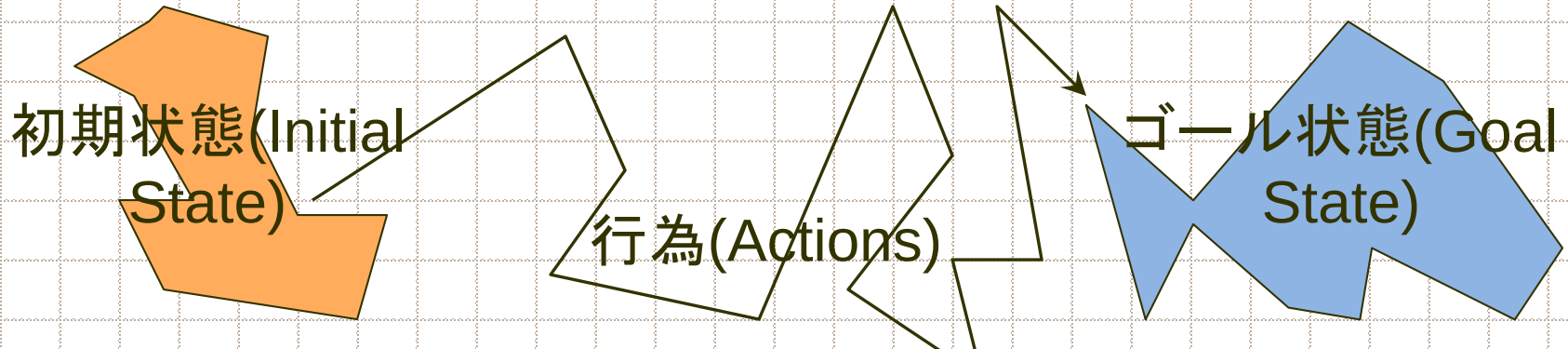
知能システム学
第11回

探索による問題解決(4)、
情報を持つ探索

ソフトウェア情報学部
David Ramamonjisoa

ゴールを用いるエージェントの構築

- エージェントの環境を状態で表現する(**state**)?
- エージェントのゴールは何か(**goal** to be achieved?)
- 可能な行為は何かある(What are the **actions**?)
- 問題を解決するためにどのような情報が状態と状態遷移に記述するべきか

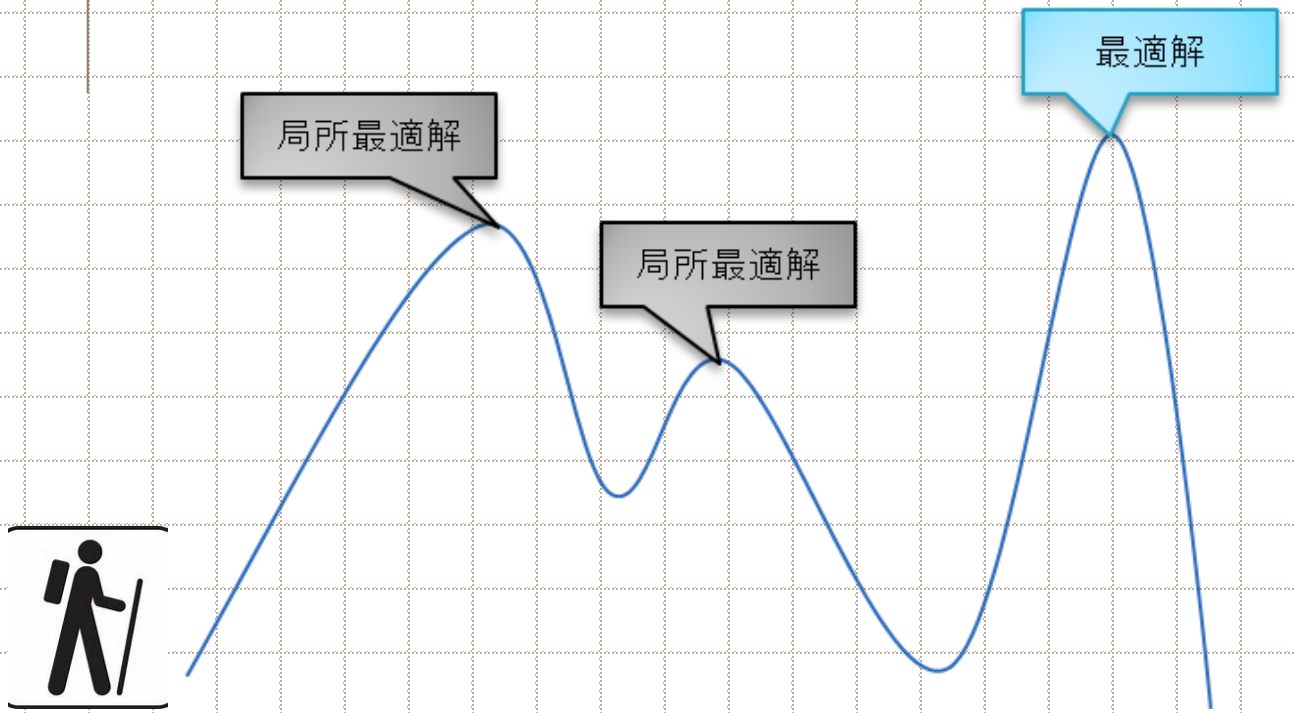


探索戦略

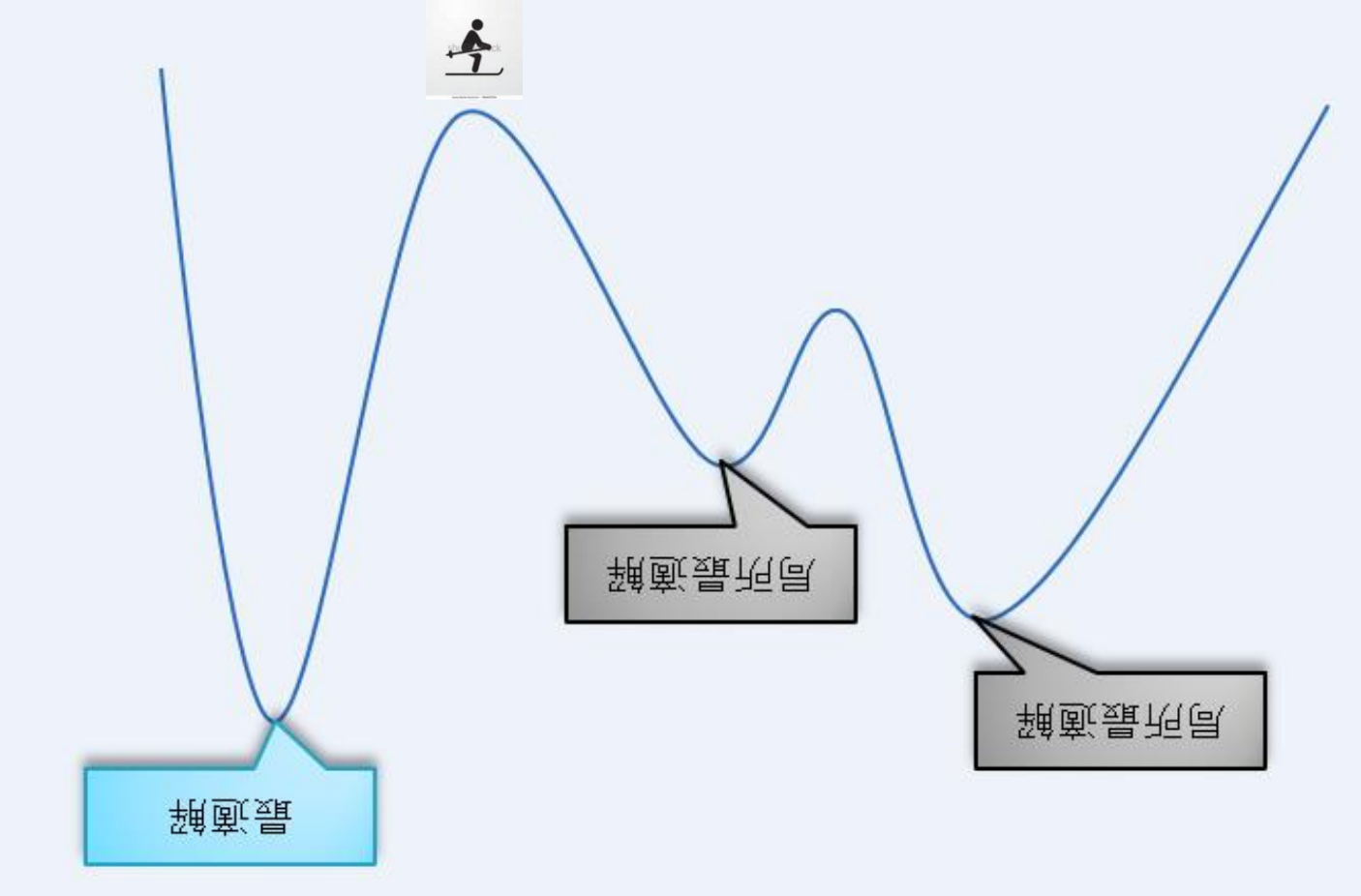
- 探索戦略選択のための四つの基準
  - 完全性： 解が存在するとき，それを見つけることが保証されているか？
  - 最適性： いくつか異なる解があるとき，戦略は最も良い解を見つけるか？
  - 時間計算量： 解を見つけるまでにどれくらい時間が掛かるか？
  - 空間計算量： 探索を行うためにどのくらいメモリを必要とするか？
- 情報を持つ探索(informed search)，ある探索
  - 現在の状態からゴールに至る順路の中で、最小コストの順路（最適順路）などを考えて効率的に行う.

情報をもつ探索戦略(Informed search strategies)

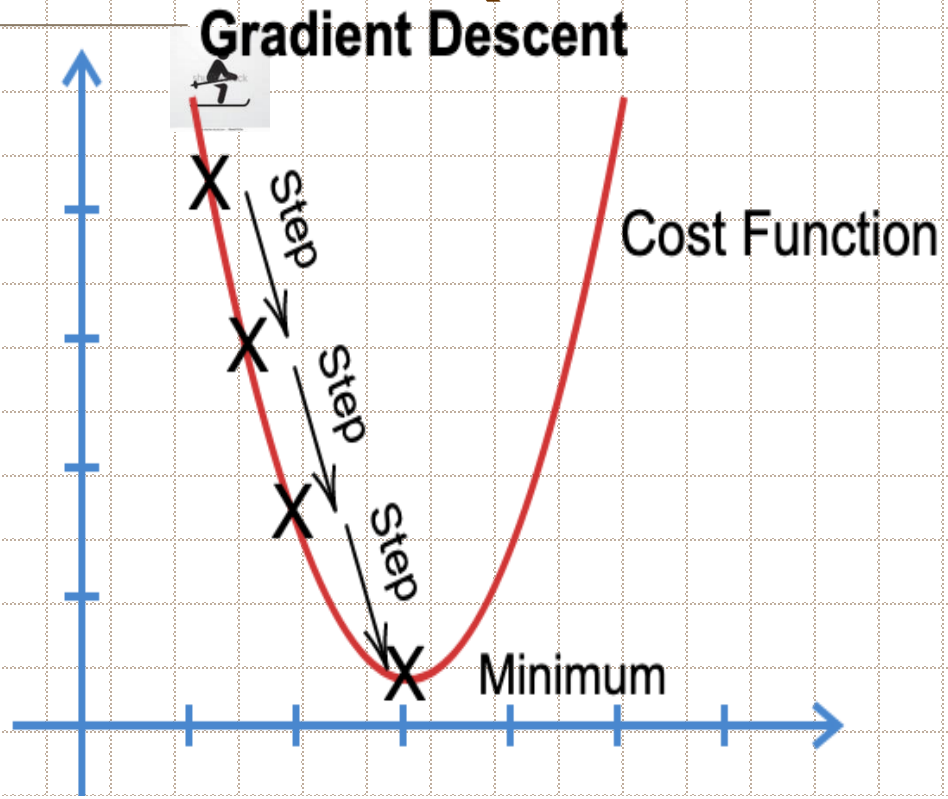
- 情報をもつ探索戦略は問題定式化の使用可能情報のみ利用する。
- 山登り法探索(hill climbing search)



勾配降下法(Gradient Descent)



勾配降下法(Gradient Descent)



- fの微分は勾配と言う。
- fの最小値は$f'(x) = 0$で分かる。0のとき、xの値を求めたい。
- 適当なx_0を初期化し、$x_n = x_{n-1} - r * f'(x_{n-1})$を計算すれば、$|x_n - x_{n-1}| < \epsilon$が満たしたら、修了

情報を持つ探索（ヒューリスティック探索）

- 情報のある探索は、ヒューリスティック探索(heuristic search)ともいう。
- 深さ優先、あるいは幅優先探索に、ヒューリスティックスを導入して、最も見込みのありそうな枝から探索を進める。
- ヒューリスティックスとしては、たとえば、ゴールに最も近づく道を選ぶ。

山登り法探索 (hill climbing search)

- ゴールへ到達するのに予想されるコストを最小化する
  - ゴール状態に最も近いと判断した状態へ到達するまでのコストを見積もりする

function HillClimbing-Search(*問題*) returns 一つの解 or 失敗
　*問題*の初期状態で，探索木を初期設定する．
　loop do
　　if 展開すべき候補ノードがない **then return** 失敗
　　*HillClimbing-Search(問題)*によって展開すべき葉ノードを選ぶ
　　if そのノードがゴール状態 **then return** 対応する解
　　else そのノードを展開して結果のノード群を探索木に加える
end

山登り法探索(HillClimbing Search)

- アルゴリズム (評価値を最小化するの方法)
  - step 1 出発点を現在の節点にする. BestNode
  - step 2 節点の評価値を求める. h(BestNode)
  - step 3 if nが目標節点である then 探索は成功, 終了.
  - step 4 if nが展開できる(子節点を持つ)

    NodeEval = 1000; NextNode=NULL

    then 展開し、すべての子節点からnへのポインタを付ける. すべての子節点をリストLに入れ、

    For Node in リストL:

    if NodeEval>h(Node):

    NodeEval = h(Node)

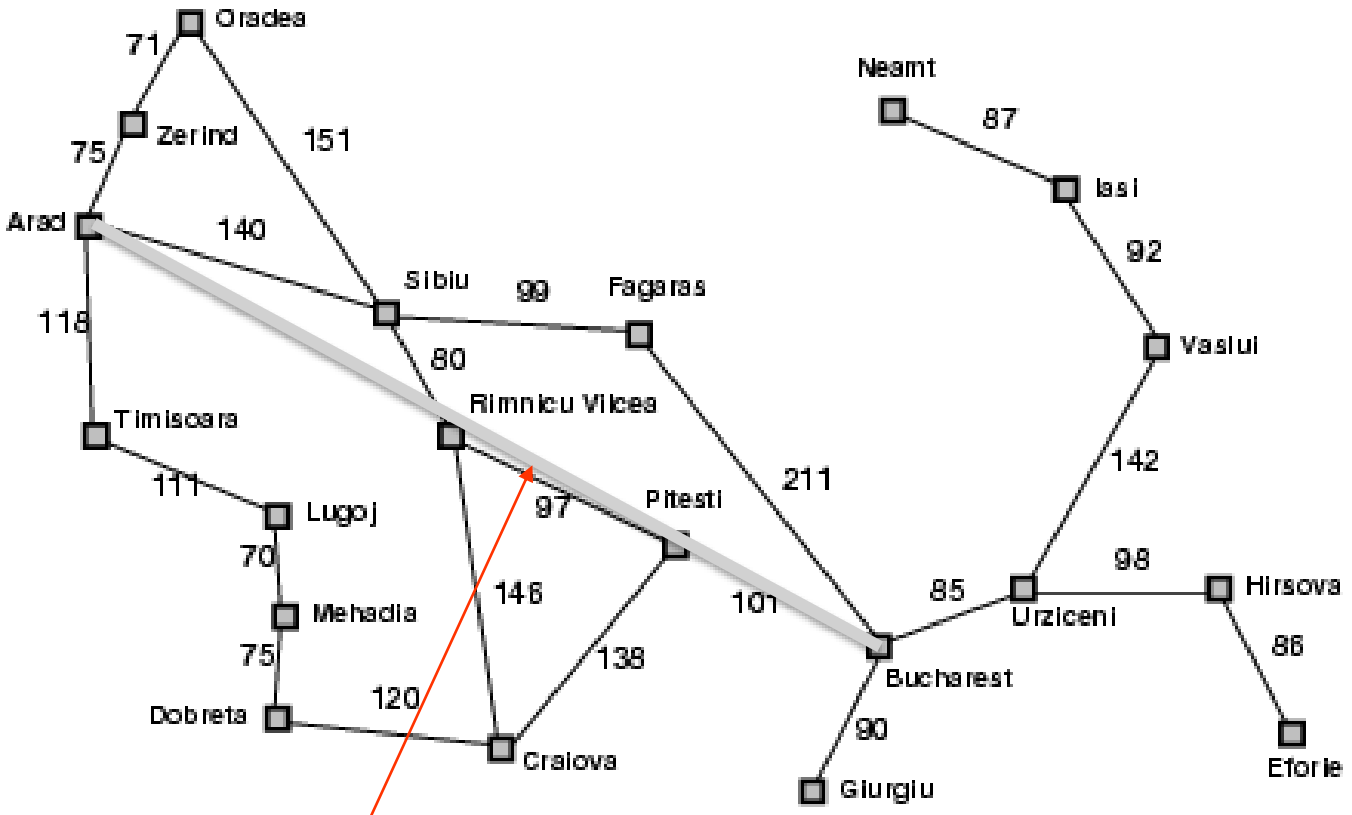
    NextNode = Node

    if h(BestNode)< NodeEval then BestNodeを返す、終了

    else BestNode=NextNode,

    step 2へ

例1：ロマニアの都市間の直線距離（コスト）(Romania with step costs in km)



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

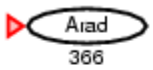
Arad-Bucharest =366km

ヒューリスティック関数の値

欲張り探索(Greedy best-first search)

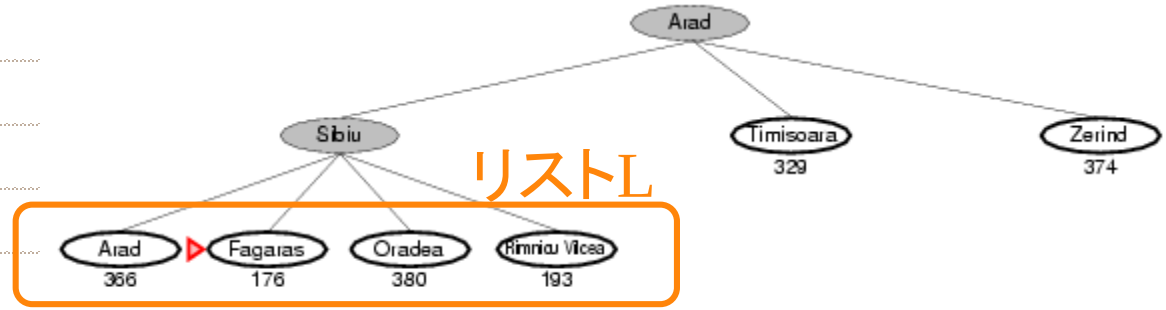
- *評価関数　f(n) = h(n)* (ヒューリスティック関数)
- *f(n) = n*からゴールまでのコスト見積もり (estimate of cost from *n* to *goal)*
- $h_{SLD}(n)$ = *n*からBucharestまでの直線距離 (straight-line distance(SLD) from *n* to Bucharest)
- 展開すべき次の節点に*f(n) を用いる欲張り探索を行う。*

欲張り探索の例(Greedy best-first search example)

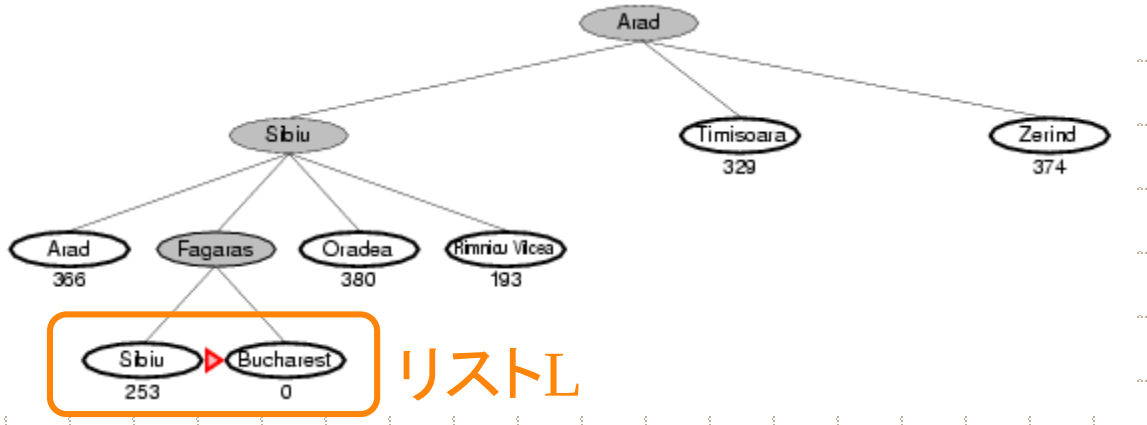


山登り法探索の例(Hill Climbing search example)

山登り法探索の例(Hill Climbing search example)



山登り法探索の例(Hill Climbing search example)



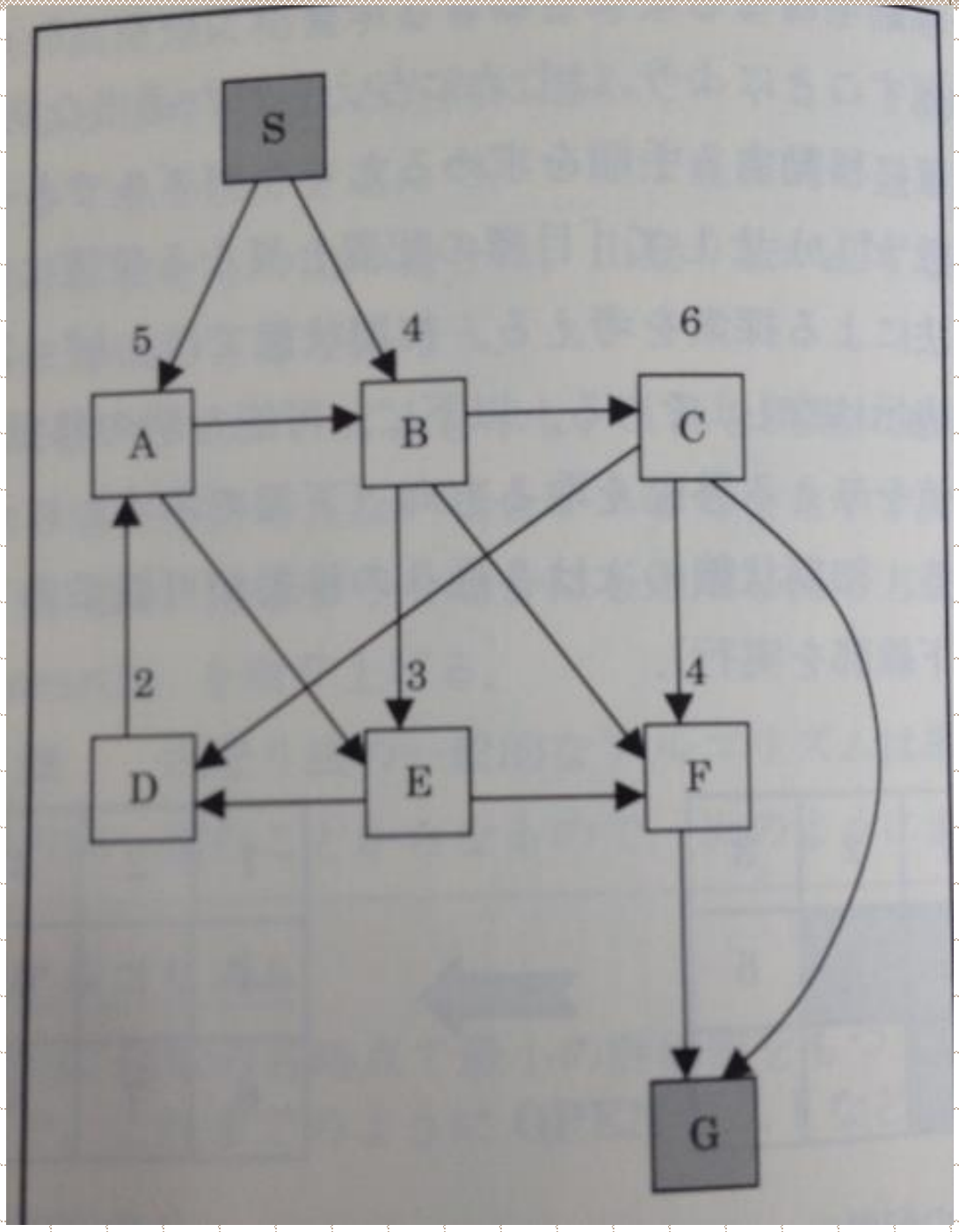
最適な解ではありません。局所最適解といいます。

山登り法探索の性質(Hill Climbing search Properties)

- 完全(Complete)? No –解は永久に求まらない。
  Iasi → Neamt → Iasi → Neamt → 行ったり、来たりになってしまう。
- 時間計算量(Time)? *O(bm)*,
-
- メモリ計算量(Space)? *O(bm)* – DFSと同じ
-
- 最適(Optimal)? No

m:探索空間の最大の深さ。
深さ優先探索と似ている。

例題2



山登り法の特徴

- 現在の状態だけを保存していけばよいので、最良優先探索法に比べ使用メモリならびに探索手間が少ない。
- バックトラック機構がないので解が得られる保証はない
- 最適解が得られるとは限らない

山登り法探索のまとめ

- 探索の方法が視界ゼロの状況で山に登るときの方法に似ていることから名付けられた。
- 局所解に落ち込んだり、解を見落としたりする可能性がある。
- 極値を探索するアルゴリズムのため、評価関数の最小値･最大値の探索手法としては不完全である。

参考ページ

- https://ja.wikipedia.org/wiki/山登り法
- 8パズルの局所解：