



Iterative Search

Sukmawati NE

Departemen Informatika

UNDIP

Iterative Search



Biasa digunakan untuk pencarian masalah yang bukan berupa path

1. Hill-climbing search
2. Simulated annealing

Algoritma Iterative Search

Algorithm 4.1 Iterative search

Step 1. `Current_state = initial state`

Step 2. **while** current state does not satisfy goal test
and time limit is not exceeded

Step 2.1 generate the successors of the
current_state

Step 2.3 set as new current_state the successor
with highest promise (i.e. its heuristic
value is lowest)

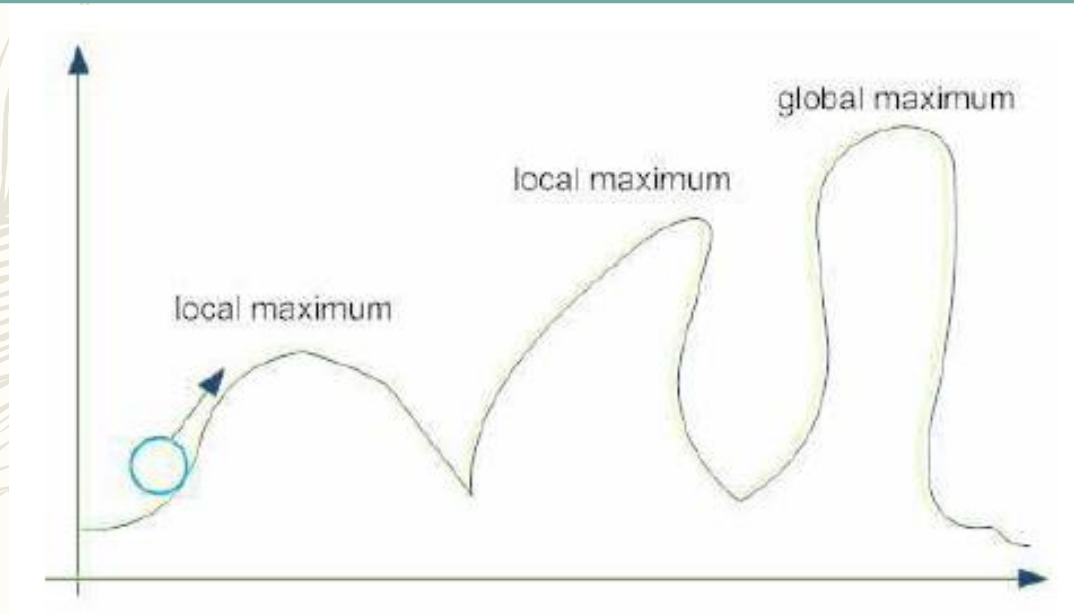
Step 3. **if** current_state satisfies goal test
then return path of actions that led to the
current_state

else return failure

end

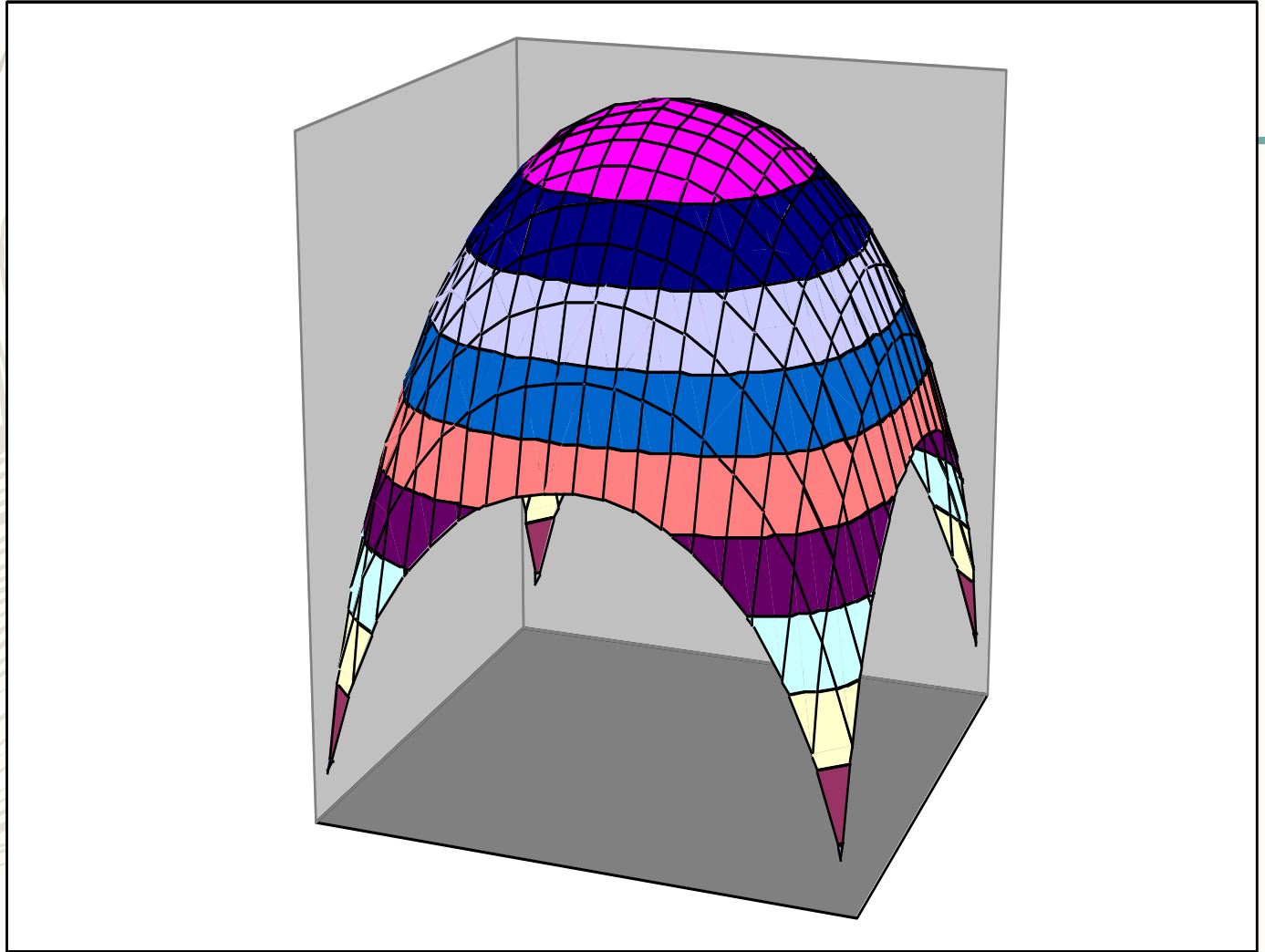
Hill Climbing

- Dikenal sebagai gradient ascent or gradient descent



- Analog dengan seseorang yang mendaki gunung tanpa adanya peta, selalu bergerak naik ke atas

Hill Climbing



Algoritma Hill Climbing

Algorithm 4.2. Hill-climbing

Step 1. Set `current_state` to take initial state (starting state).

Step 2. **loop**

Step 2.1 Generate successors of `current_state`;

Step 2.2 Get the successor with the highest value;

Step 2.3 **if** `value(successor) < value(current_state)`

then Return `current_state`

else `currentst_state = successor`

end

Hill Climbing

- Terkadang dapat menemukan puncak yang salah (non-goal node) yang merupakan local maximum → tergantung initial state

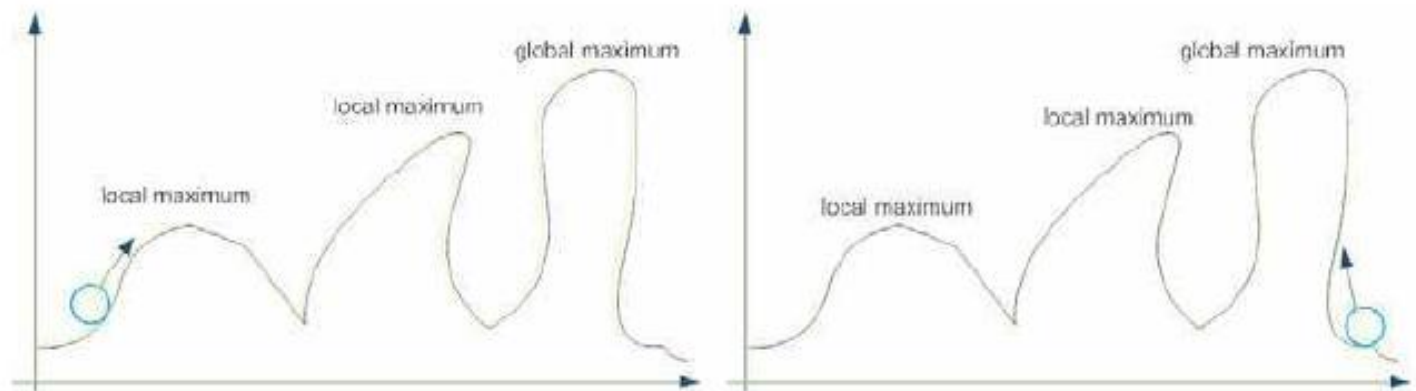


Fig. 4.2 Example of different starting states for the hill-climbing search leading to a local maximum (left) and global maximum respectively (right).

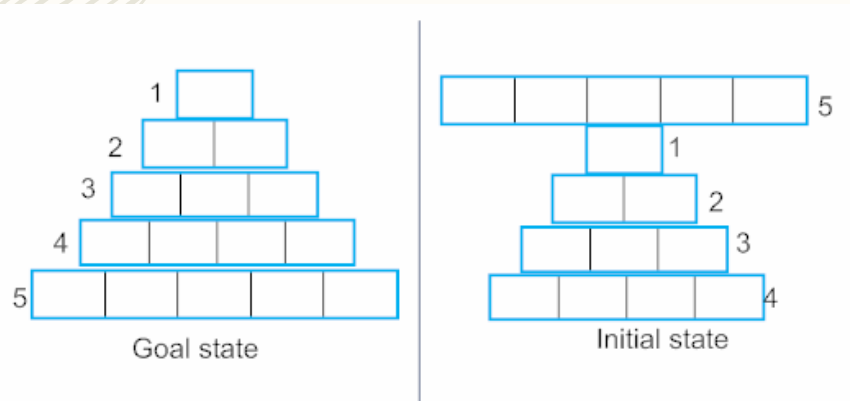
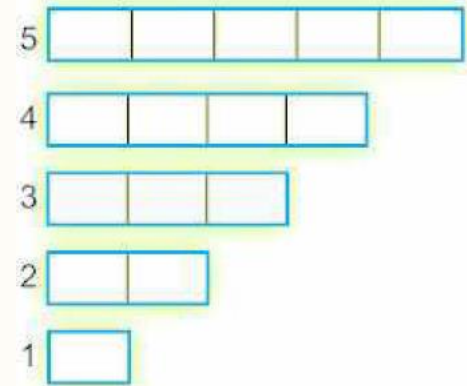
Hill Climbing



- Beberapa situasi yang perlu diperhatikan dalam Hill climbing
 - *plateau: successor states have same values, no way to choose;*
 - *foothill: local maximum, can get stuck on minor peak;*
 - *ridge: foothill where N-step look ahead might help*
- Pemilihan fungsi heuristik juga dapat berpengaruh terhadap pencapaian global maksimum

Contoh

- Terdapat gambar blok kotak sebagai berikut.

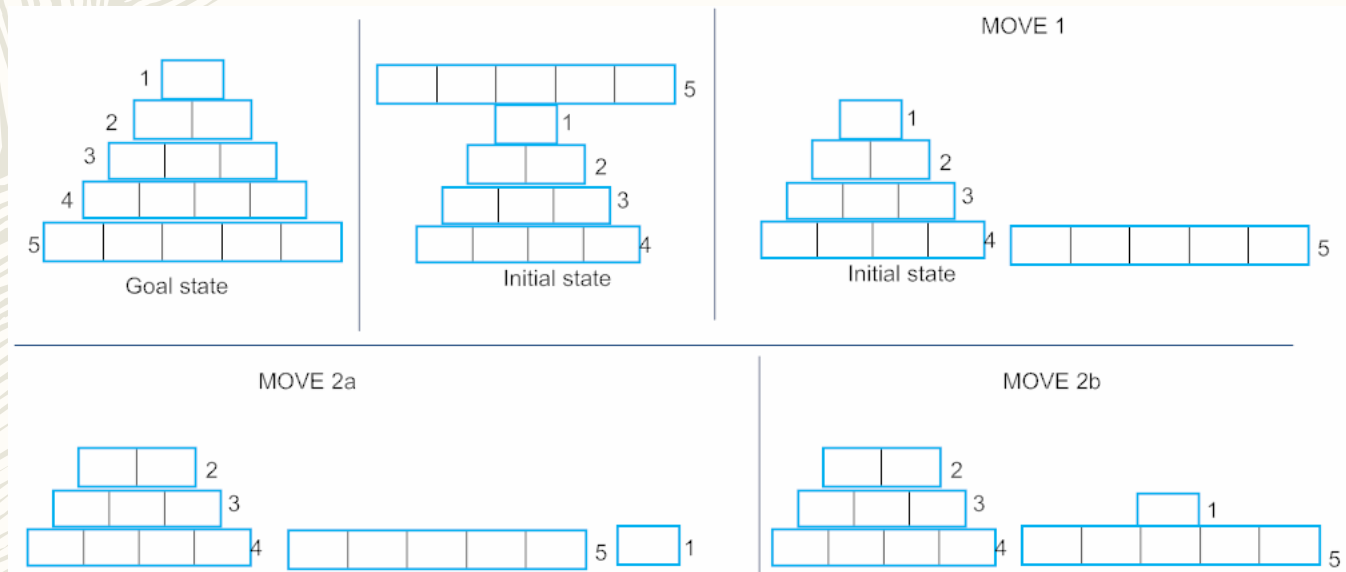


Lanj contoh : Heuristic 1

- count +1 for every figure that sits on the correct figure. The goal state has the value +5; (Setiap blok yang berada dalam diatas blok yang tepat bernilai 1)
- count -1 for every figure that sits on an incorrect figure. (Setiap blok yang berada dalam blok yang tidak tepat bernilai -1)
- Berapa nilai pada initial state?

Lanj contoh : Heuristic 1

- Nilai initial state $= +3 - 2 = 1$ (blok 1,2,3 di atas kotak yang benar, blok 4,5 di atas posisi salah)



- Move 1 (New Successor) $= +4 - 1 = 3$ (hanya blok 4 yang salah, lainnya benar)

Lanj contoh : Heuristic 1

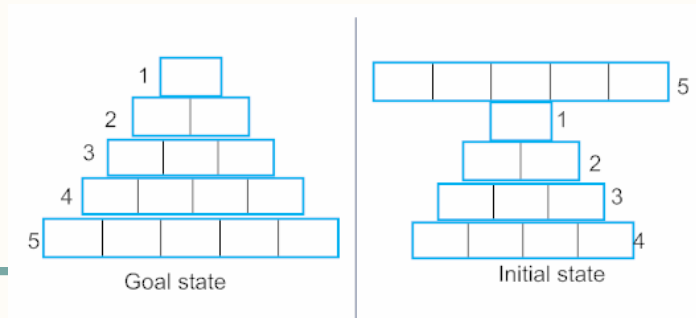
- Move 2a : $+3 - 2 = 1$ (blok 1,4 posisi salah)
- Move 2b : $+3 - 2 = 1$ (blok 1,4 posisi salah)
- Nilai move 1 > move 2 sehingga move 1 optimum.
- Merupakan local optimum bukan global optimum

Heuristic 2



- count $+n$ for every piece that sits on a correct stack of n pieces. The goal state has the value $+10$.
- Dengan kata lain : Jumlah n blok yang tepat berada di bawah blok yang di amati
- count $-n$ for every block that sits on an incorrect stack of n .
- Berapa nilai untuk initial state?

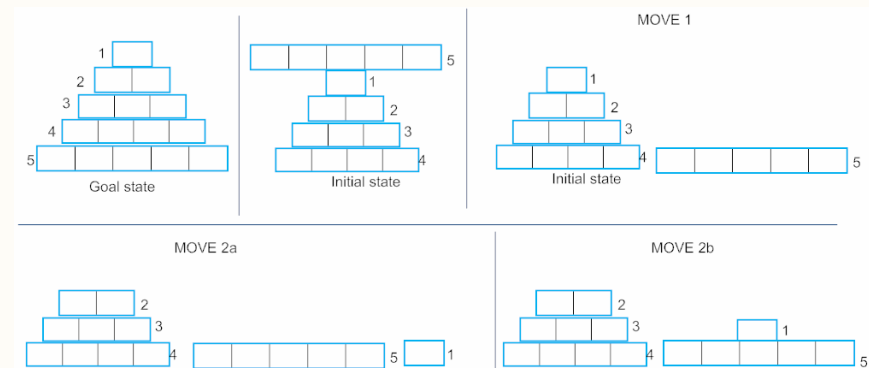
Lanj. Contoh : Heuristic 2



- Nilai di initial state = $-1 + (-2) + (-3) + (-4)$
 - Blok 3 ada di atas 1 blok salah
 - Blok 2 ada di atas 2 blok yang salah
 - Blok 1 ada di atas 3 blok yang salah
 - Blok 5 diatas 4 blok yang salah
- Berapa nilai di Move 1, Move 2a dan Move 2b?

Lanj. Contoh : Heuristic 2

- Move 1 = -6
 - Move 2a = -3
 - Move 2b = -4
-
- Local maximum dapat dihindari dengan menggunakan heuristic 2



Contoh lain

- Masalah 3 misionaris dan 3 kanibal
- Problem menyebrangi sungai dengan menggunakan satu perahu yang hanya muat 2 orang
- Mirip dengan problem 3 polisi dan 3 narapidana
- Ilustrasinya :

Left side
(3, 3, 1)
(0, 0, 0)

current state
goal state

Right side
(0, 0, 0)
(3, 3, 1)

Latihan 1



- Carilah pada saat terjadi lokal maximum pada penyelesaian masalah di atas dengan Hill Climbing!
- Catatan :
 - Hill Climbing akan berhenti jika ada move yang membuat langkahnya menjauhi goal

Contoh : 8-puzzle

– h : jumlah tile yang tidak sesuai

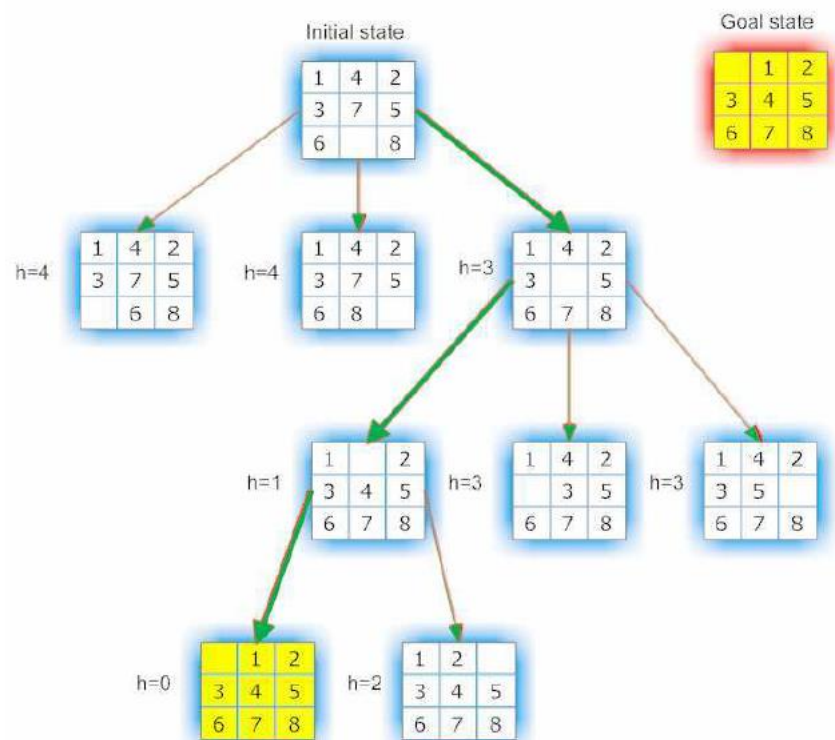


Fig. 4.5 Hill-Climbing for the 8-puzzle problem.

Latihan 2

- Carilah dengan hill climbing!

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

Contoh : TSP

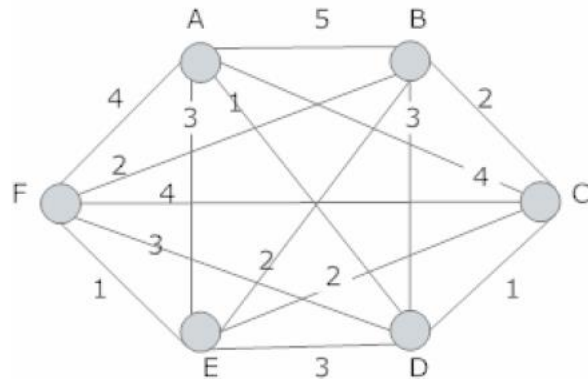


Fig. 4.6 The graph for the TSP example.

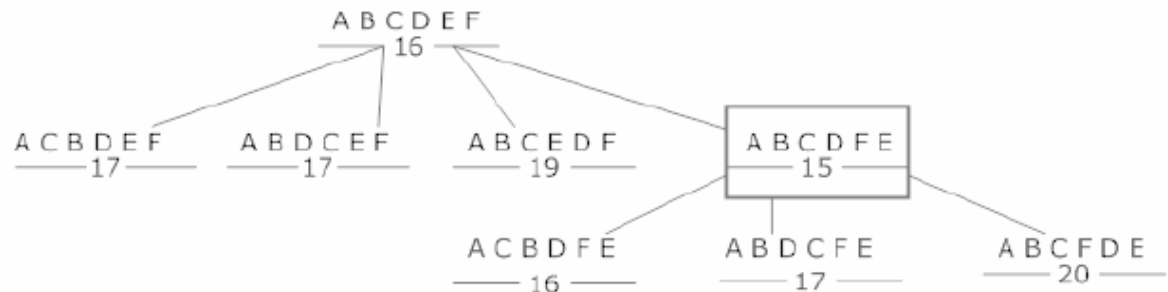


Fig. 4.7 The (local) solution obtained by hill-climbing for TSP.

Lanjutan TSP

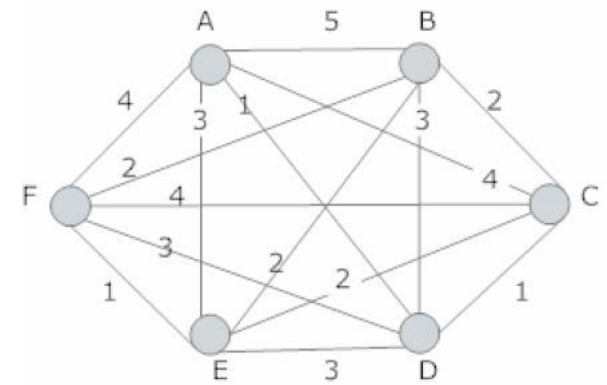


Fig. 4.6 The graph for the TSP example.

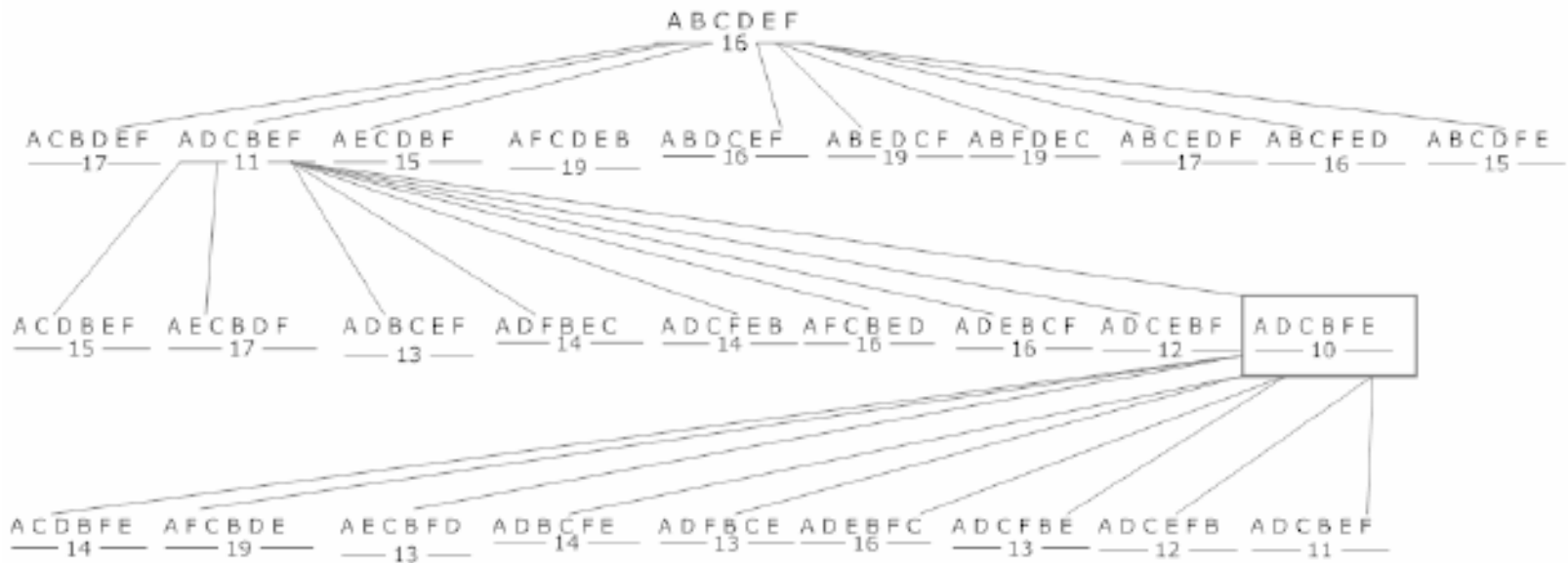


Fig. 4.8 The global solution obtained by hill-climbing for TSP.



Kelebihan dan kekurangan TSP

Advantages:

1. easy to implement; the algorithm is very simple and easy to reproduce;
 - requires no memory (since there is no backtracking);
 - since it is very simple, can be easily used to get an approximate solution when the exact one is hard or almost impossible to find (for instance, for very large TSP instances (or other similar NP Complete problems), an approximate solution may be satisfactory when the exact one is not known and difficult to find).

Disadvantages:

- the evaluation function may be sometimes difficult to design;
- if the number of moves is enormous, the algorithm may be inefficient;
- by contrary, if the number of moves is less, the algorithm can get stuck easily;
- it's often cheaper to evaluate an incremental change of a previously evaluated object than to evaluate from scratch;
- inner-loop optimization often possible.

Kelemahan Hill Climbing




- Proses akan berhenti saat menemukan local maksimum
- Hasilnya akan berupa local maksimum bukan global maksimum
- Kelemahan ini berusaha diperbaiki dengan Simulated Annealing

Simulated Annealing (SA)



-
- Salah satu algoritme untuk optimisasi yang berbasiskan probabilitas dan mekanika statistic.
 - Algoritme ini dapat digunakan untuk mencari pendekatan terhadap solusi optimum global dari suatu permasalahan.
 - Masalah yang membutuhkan pendekatan SA adalah masalah-masalah optimisasi kombinatorial, di mana ruang pencarian solusi yang ada terlalu besar, sehingga hampir tidak mungkin ditemukan solusi eksak terhadap permasalahan itu.
 - Publikasi tentang pendekatan ini pertama kali dilakukan oleh S. Kirkpatrick, C. D. Gelatt dan M. P. Vecchi, diaplikasikan pada desain optimal hardware komputer, dan juga pada salah satu masalah klasik ilmu komputer yaitu *Traveling Salesman Problem*.
 - Dasar SA adalah Annealing

Anneling

- 
-
- Satu teknik yang dikenal dalam bidang metalurgi
 - Digunakan dalam mempelajari proses pembentukan kristal dalam suatu materi. Agar dapat terbentuk susunan kristal yang sempurna, diperlukan pemanasan sampai suatu tingkat tertentu, kemudian dilanjutkan dengan pendinginan yang perlahan-lahan dan terkendali dari materi tersebut.
 - Pemanasan materi di awal proses annealing, memberikan kesempatan pada atom-atom dalam materi itu untuk bergerak secara bebas, mengingat tingkat energi dalam kondisi panas ini cukup tinggi.
 - Proses pendinginan yang perlahan-lahan memungkinkan atom-atom yang tadinya bergerak bebas itu, pada akhirnya menemukan tempat yang optimum, di mana energi internal yang dibutuhkan atom itu untuk mempertahankan posisinya adalah minimum.

Annealing



- Based on a metallurgical metaphor
 - Start with a temperature set very high and slowly reduce it.
 - Run hillclimbing with the twist that you can occasionally replace the current state with a **worse** state based on the current temperature and how much worse the new state is.

Annealing



- More formally...
 - Generate a new neighbor from current state.
 - If it's better take it.
 - If it's worse then take it with some probability proportional to the temperature and the delta between the new and old states.

Simulated Annealing



- Simulated Annealing berjalan berdasarkan analogi dengan proses annealing yang telah dijelaskan di atas.
- Pada awal proses SA, dipilih suatu solusi awal, yang merepresentasikan kondisi materi sebelum proses dimulai.
- Gerakan bebas dari atom-atom pada materi, direpresentasikan dalam bentuk modifikasi terhadap solusi awal/solusi sementara.
- Pada awal proses SA, saat parameter suhu (T) diatur tinggi, solusi sementara yang sudah ada diperbolehkan untuk mengalami modifikasi secara bebas.

Lanjutan SA



-
- Kebebasan ini secara relatif diukur berdasarkan nilai fungsi tertentu yang mengevaluasi seberapa optimal solusi sementara yang telah diperoleh.
 - Bila nilai fungsi evaluasi hasil modifikasi ini membaik (dalam masalah optimisasi yang berusaha mencari minimum berarti nilainya lebih kecil/downhill) solusi hasil modifikasi ini akan digunakan sebagai solusi selanjutnya.
 - Bila nilai fungsi evaluasi hasil modifikasi ini memburuk, pada saat temperatur annealing masih tinggi, solusi yang lebih buruk (uphill) ini masih mungkin diterima, sedangkan pada saat temperatur annealing sudah relatif rendah, solusi hasil modifikasi yang lebih buruk ini mungkin tidak dapat diterima.
 - Dalam tahapan selanjutnya saat temperatur sedikit demi sedikit dikurangi, maka kemungkinan untuk menerima langkah modifikasi yang tidak memperbaiki nilai fungsi evaluasi semakin berkurang. Sehingga kebebasan untuk memodifikasi solusi semakin menyempit, sampai akhirnya diharapkan dapat diperoleh solusi yang mendekati solusi optimal.

Simulated annealing

function SIMULATED-ANNEALING(problem, schedule) **return** a solution state

input: problem, a problem

schedule, a mapping from time to temperature

local variables: current, a node.

next, a node.

T, a “temperature” controlling the probability of downward steps

current \leftarrow MAKE-NODE(INITIAL-STATE[problem])

for t \leftarrow 1 to ∞ **do**

T \leftarrow schedule[t]

if T = 0 **then return** current

next \leftarrow a randomly selected successor of current

$\Delta E \leftarrow$ VALUE[next] - VALUE[current]

if $\Delta E > 0$ **then** current \leftarrow next

else current \leftarrow next only with probability $e^{\Delta E / T}$

Simulated Annealing: the code

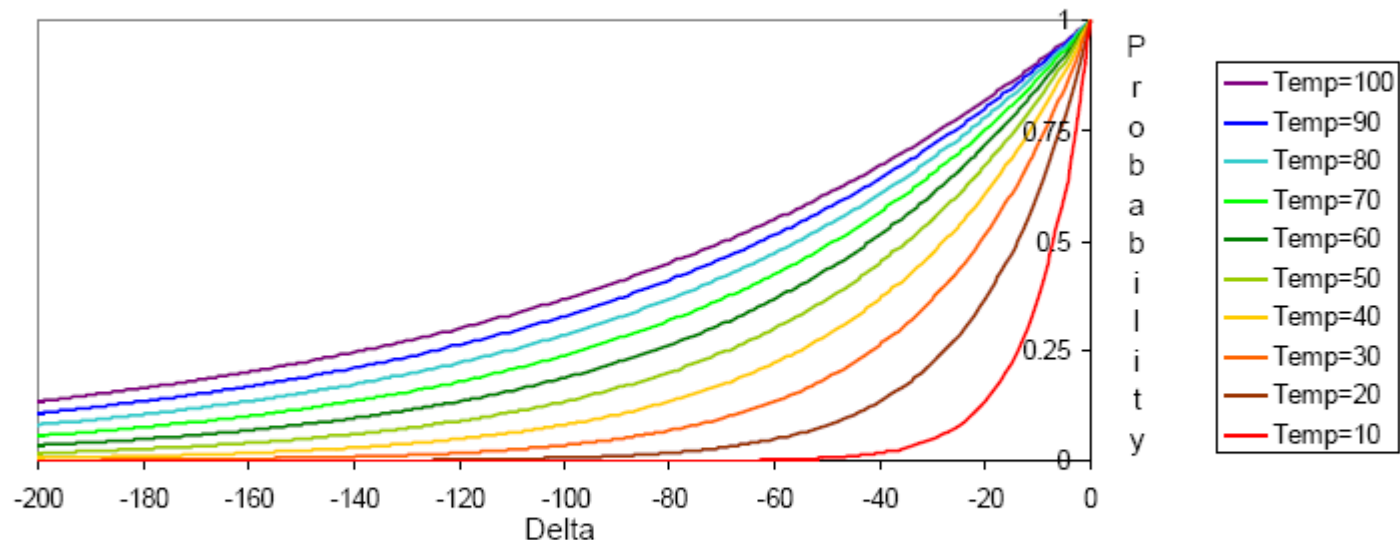
```
1. Create random initial solution  $\gamma$ 
2.  $E_{old} = \text{cost}(\gamma)$ ;
3. for(temp=tempmax; temp>=tempmin; temp=next_temp(temp) ) {
4.     for(i=0; i<imax; i++ ) {
5.         sucesor_func( $\gamma$ ); //this is a randomized function
6.          $E_{new} = \text{cost}(\gamma)$ ;
7.         delta= $E_{new} - E_{old}$ ;
8.         if(delta>0)
9.             if(random() >= exp(-delta/K*temp);
10.                undo_func( $\gamma$ ); //rejected bad move
11.             else
12.                  $E_{old} = E_{new}$  //accepted bad move
13.         else
14.              $E_{old} = E_{new}$ ; //always accept good moves
    }
}
```

Simulated Annealing

– Acceptance criterion and cooling schedule

if ($\Delta \geq 0$) accept

else if ($\text{random} < e^{\Delta / \text{Temp}}$) accept, else reject /* $0 \leq \text{random} \leq 1$ */



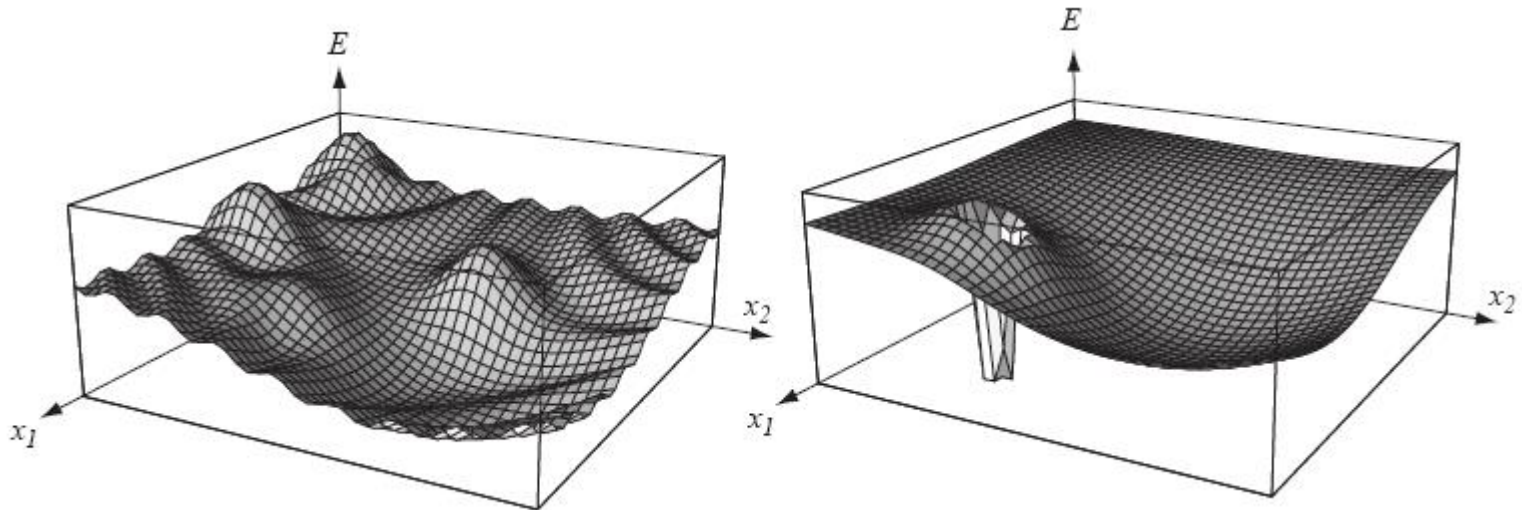
Initially temperature is very high (most bad moves accepted)

Temp slowly goes to 0, with multiple moves attempted at each temperature

Final runs with temp=0 (always reject bad moves) greedily “quench” the system

Practical Issues with simulated annealing

- Cost function must be carefully developed, it has to be “fractal and smooth”.
- The energy function of the left would work with SA while the one of the right would fail.



Applications



- Basic Problems
 - Traveling salesman
 - Graph partitioning
 - Matching problems
 - Graph coloring
 - Scheduling
- Engineering
 - VLSI design
 - *Placement*
 - *Routing*
 - *Array logic minimization*
 - *Layout*
 - Facilities layout
 - Image processing
 - Code design in information theory