

# Implementasi Simulated Annealing Untuk Menyelesaikan Traveling Salesman Problem (TSP)

**Achmad Basuki**

Politeknik Elektronika Negeri Surabaya

PENS-ITS Surabaya 2005



- Gambar Permasalahan TSP
- Definisi State
- Definisi Energi
- Flowchart Simulated Annealing Untuk TSP
- Pembangkitan State Awal
- Update State
- Cooling Schedule



- *Traveling salesman problem* (TSP) adalah suatu permasalahan untuk mendapatkan rute terpendek yang harus dilalui seorang sales yang harus melewati semua kota ( $n$ ) dengan setiap kota harus dilalui satu kali sampai dia kembali ke kota asalnya.
- TSP banyak digunakan dalam penerapannya untuk bidang transportasi, komunikasi dan teknologi informasi.
- Dalam TSP, tujuan yang dicapai adalah rute dengan jarak terpendek, dan batasannya adalah semua kota harus dilalui dan setiap kota hanya dilalui satu kali.



- Simulated Annealing pada TSP digunakan untuk menelusuri dan mencari setiap rute yang mungkin, kemudian mendapatkan rute yang jaraknya paling pendek.
- Model Simulated Annealing untuk menyelesaikan TSP adalah model state yang dibangun untuk menyatakan rute yang mungkin dan definisi energi yang dinyatakan dengan total jarak yang ditempuh.



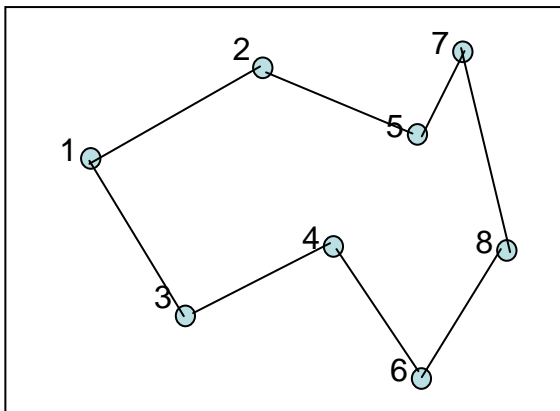
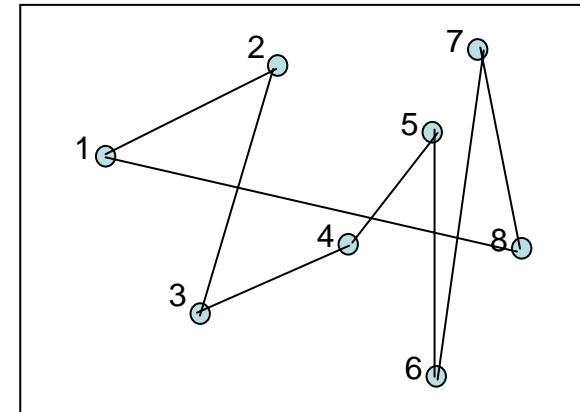
- State didefinisikan dengan rute yang ditempuh untuk melewati semua kota sampai kembali ke kota asal dengan syarat setiap kota harus dilalui satu kali.
- Bila setiap kota dinyatakan dengan indeks  $1, 2, 3, 4, \dots, n$  maka state didefinisikan sebagai permutasi dari indeks kota.
- Definisi state adalah:

$$S = \left\{ s_i \in N \mid \left( s_i \neq s_j \right)_{i \neq j} \right\}$$



Contoh state untuk TSP dengan 8 kota, dimana setiap kota dinyatakan dengan indeks 1,2,3,4,5,6,7, dan 8 adalah sebagai berikut:

**1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 1**



**1 - 2 - 5 - 7 - 8 - 6 - 4 - 3 - 1**

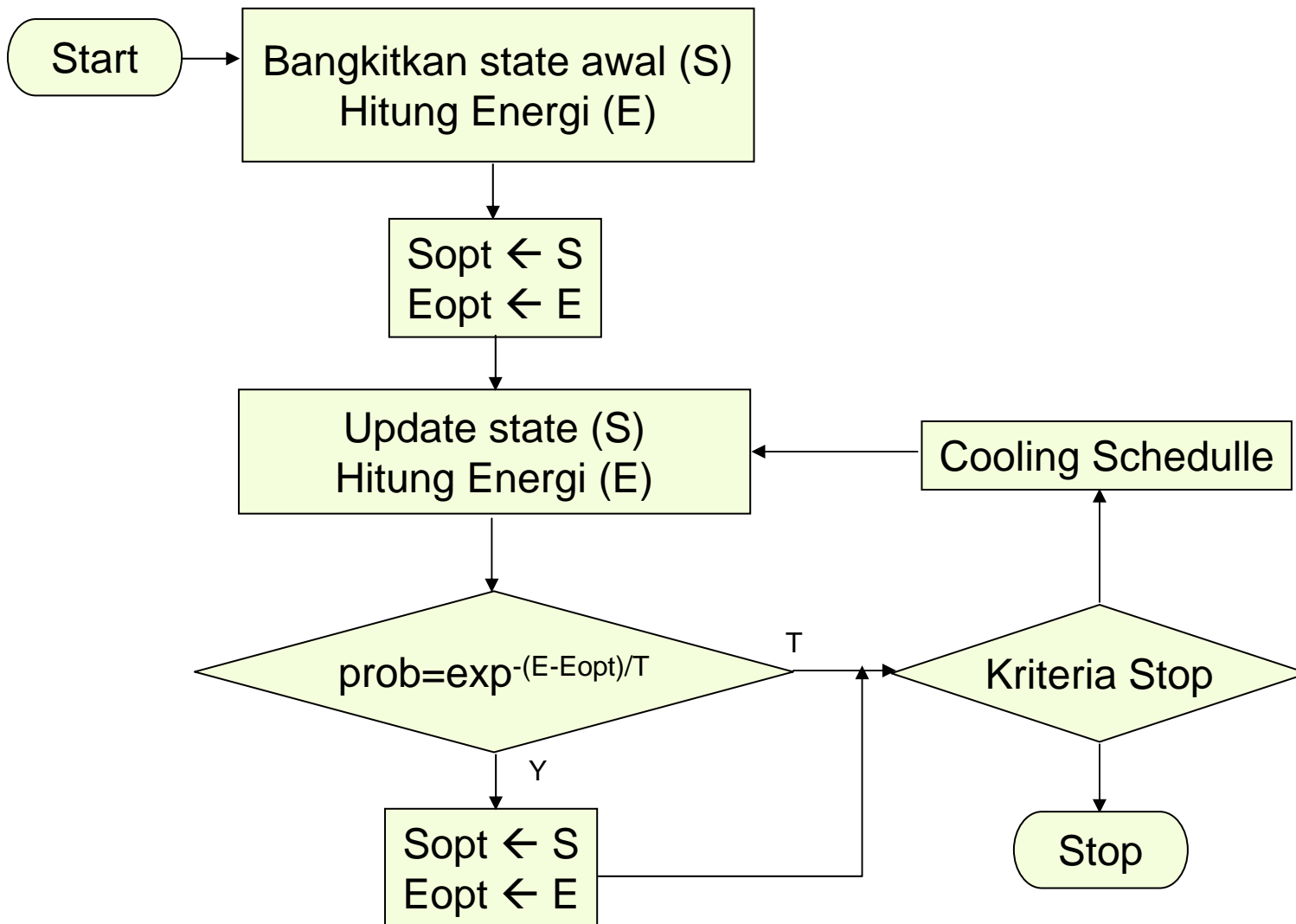


- Karena di dalam TSP dicari rute dengan jarak terpendek, maka energi didefinisikan dengan total jarak yang ditempuh.
- Energi didefinisikan:

$$E = \sum_{i=1}^n d_i$$

$d_i$  adalah jarak kota ke  $s(i)$  dan  $s(i+1)$ , bila posisi dinyatakan sebagai koordinat 2 dimensi  $(x,y)$  maka  $d_i$  dapat dihitung dengan:

$$d_i = \sqrt{(s_x(i) - s_x(i+1))^2 + (s_y(i) - s_y(i+1))^2}$$

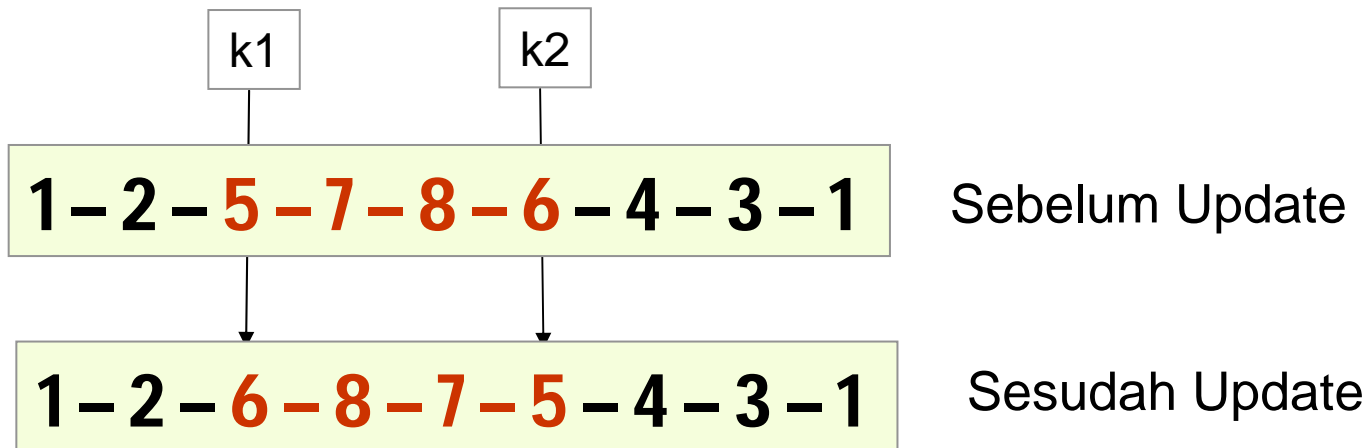




- Proses membangkitkan state awal dapat dilakukan dengan mengurutkan nomor indeks kota misalkan  $1, 2, 3, 4, \dots, n$  (cara ini yang dijelaskan dalam modul ini karena sangat mudah dilakukan)
- Cara lain yang bisa dilakukan dengan menggunakan acak permutasi.



- Update state adalah suatu proses untuk mengubah susunan rute dengan memilih sebagian komposisi urutan (nomor kota k1 sampai dengan nomor kota k2) secara acak.
- Pengubahan dapat dilakukan dengan cara swap pada komposisi yang ditentukan.





```
#include <fstream>
#include <stdlib.h>
#include <math.h>

float x[100], y[100];
int s[100], sOpt[100], nKota;
float e, eOpt;

// Membangkitkan data kota
// dan koordinatnya secara acak
void BangkitkanDataKota(){
    cout << "Jumlah kota = "; cin >> nKota
    for(int i=0;i<n;i++){
        x[i]=(float)(10*rand()/RAND_MAX);
        y[i]=(float)(10*rand()/RAND_MAX);
    }
}
```



```
// Membangkitkan state awal
// dengan mengurutkan langsung
void BangkitkanStateAwal() {
    for(int i=0;i<n;i++) s[i]=i;
    hitungEnergi();
}

// State Optimal
void StateOptimal() {
    for(int i=0;i<n;i++) sOpt[i]=s[i];
    eOpt=e;
}

// Menampilkan State
Void TampilkanState() {
    for(int i=0;i<n;i++) cout << sOpt[i] << " ";
    cout << "    energi = " << eOpt << endl;
}
```



```
float hitungEnergi(){
    float jarak=0;
    int i,j;
    for(i=0;i<n-1;i++){
        dx=sqr(x[s[i]]-x[s[i+1]]);
        dy=sqr(y[s[i]]-y[s[i+1]]);
        d=sqrt(dx+dy);
        jarak=jarak+d;
    }
    dx=sqr(x[s[n-1]]-x[s[0]]);
    dy=sqr(y[s[n-1]]-y[s[0]]);
    d=sqrt(dx+dy);
    jarak=jarak+d;
    return jarak;
}
```



```
// Update state
void UpdateState(){
    int k1,k2,i;
    for(i=0;i<n;i++) s[i]=sOpt[i];
    k1=int(nKota*(float)rand()/RAND_MAX);
    k2=int((nKota-k1)*(float)rand()/RAND_MAX)+k1;
    for(i=k1;i<k2;i++){
        s[i]=sOpt[k2+k1-i];
    }
    hitungEnergi();
}
```



```
void main(){
    int i, maxIter;
    float p, To, Tn, T;
    cout << "jumlah iterasi max = ";
    cin >> maxIter;
    To=0.1; Tn=0.0001;
    BangkitkanDataKota();
    BangkitkanStateAwal();
    StateOptimal();
    TampilkanState();
    for(i=0;i<maxIter;i++){
        UpdateState();
        p=rand;
        if(p<exp(-(e-eOpt)/T)) StateOptimal();
        TampilkanState();
        T=To*(float)pow(Tn/To,i/n);
    }
}
```



福叔