Simulated Annealing

Ulkei Szabolcs

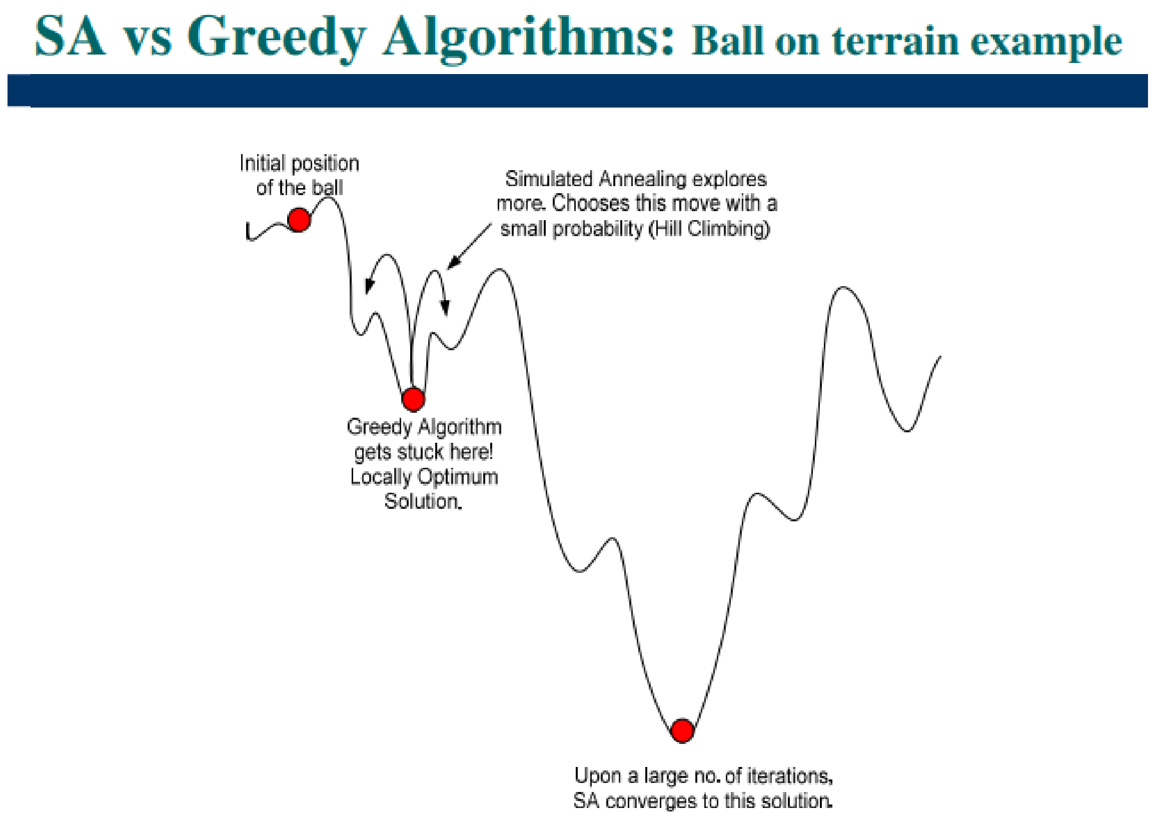
AC, CTI

7.1

General

Simulated Annealing is a stochastic, memoryless algorithm applied to solve both combinatorial and continuous optimization problems, inspired by the physical annealing process.

It is often parallelized with the greedy algorithm over which the simulated annealing has the advantage of escaping the local optima by allowing worsening moves.


SA vs Greedy Algorithms: Ball on terrain example
Initial position of the ball
Simulated Annealing explores more. Chooses this move with a small probability (Hill Climbing)
Greedy Algorithm gets stuck here! Locally Optimum Solution.
Upon a large no. of iterations, SA converges to this solution.

The algorithm had a major impact on the field of heuristic research, then got extended to deal with continuous optimization problems.

Applications

– Traveling Salesman Problem

– Graph partitioning, Matching prob., Quadratic Assignment, Linear Arrangement, Scheduling

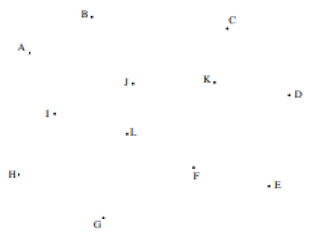


Figure 5.26: Stage 1

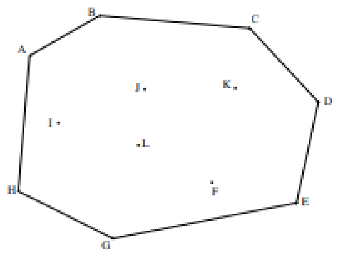


Figure 5.27: Stage 2

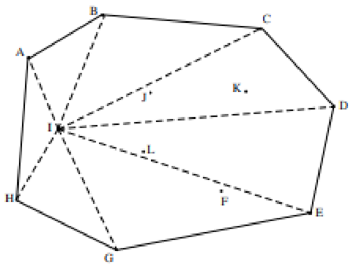


Figure 5.28: Stage 3

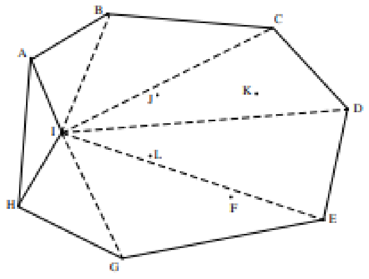


Figure 5.29: Stage 4

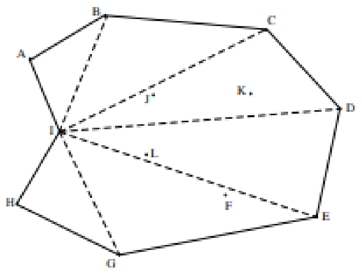


Figure 5.30: Stage 5

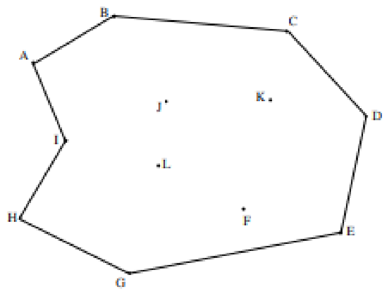


Figure 5.31: Stage 6

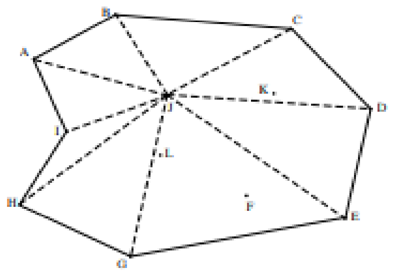


Figure 5.32: Stage 7

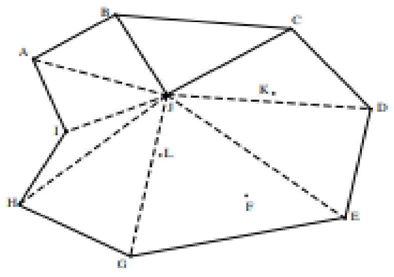


Figure 5.33: Stage 8

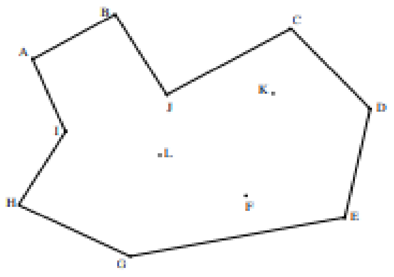


Figure 5.34: Stage 9

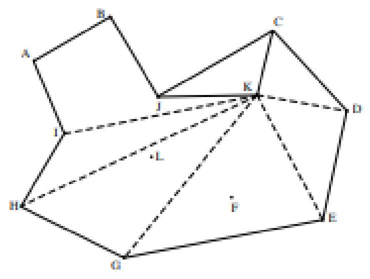


Figure 5.35: Stage 7

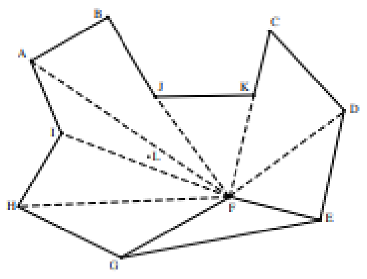


Figure 5.36: Stage 8

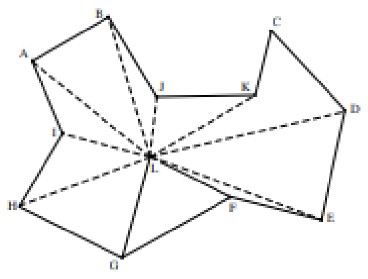


Figure 5.37: Stage 9

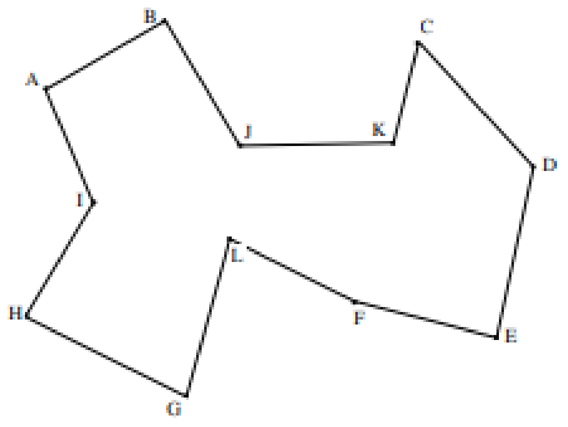
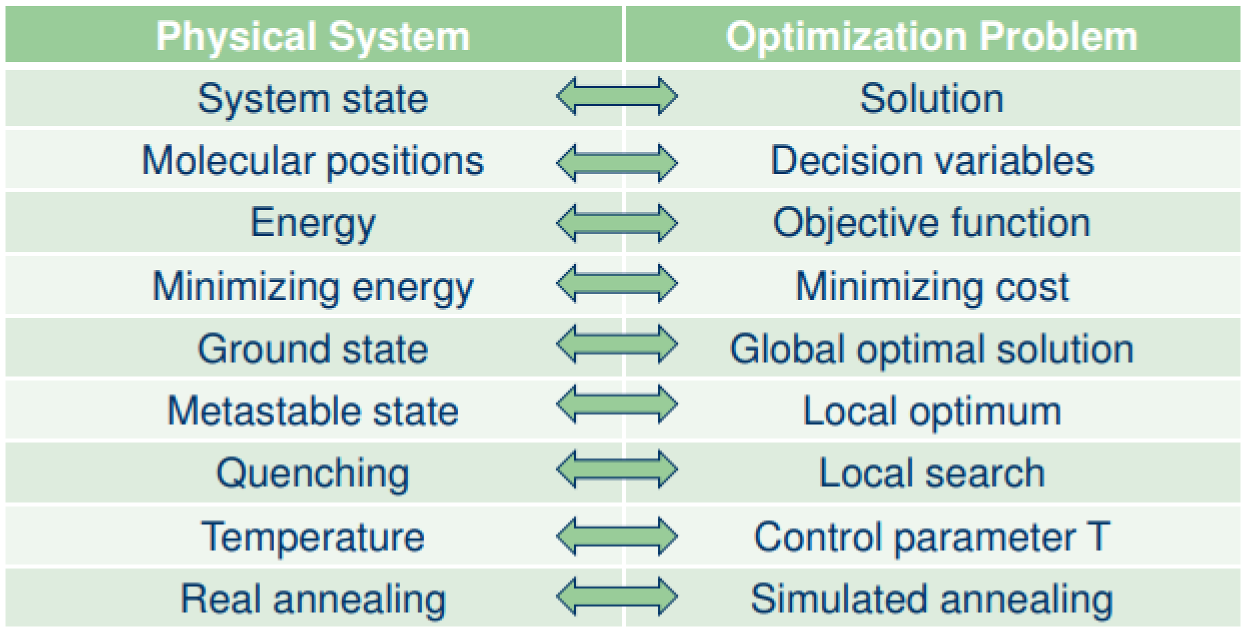


Figure 5.38: Stage 10

Analogy between the physical system and the optimization problem



Physical System	Optimization Problem
System state	Solution
Molecular positions	Decision variables
Energy	Objective function
Minimizing energy	Minimizing cost
Ground state	Global optimal solution
Metastable state	Local optimum
Quenching	Local search
Temperature	Control parameter T
Real annealing	Simulated annealing

Metropolis algorithm

In 1958 Metropolis et al. introduced a simple algorithm for simulating the evolution of a solid in a heat bath to thermal equilibrium.

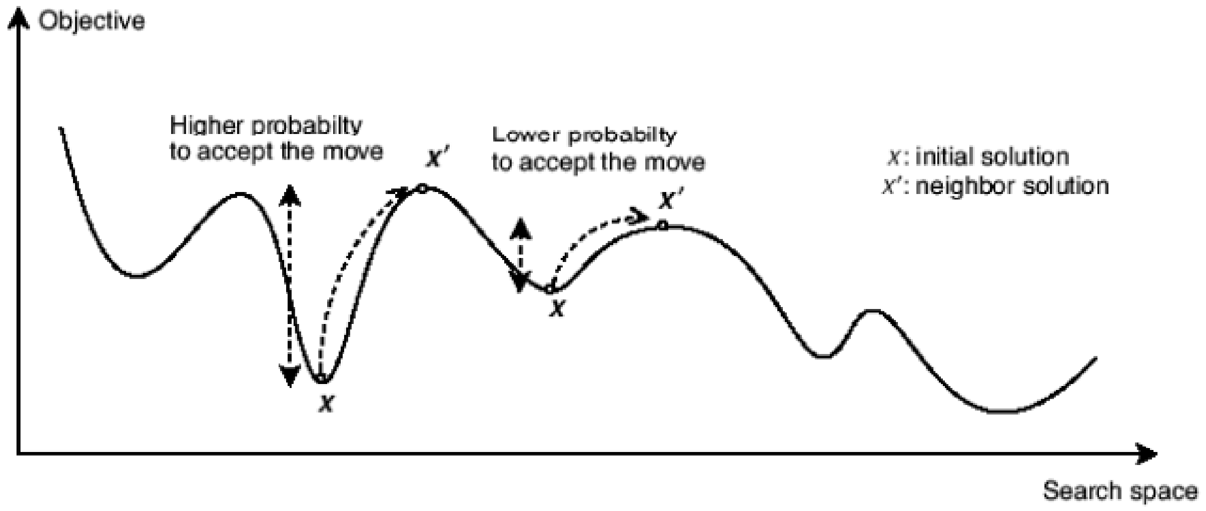
Algorithm based on Monte Carlo techniques.

On the basis given, the system is subjected to an elementary solution.

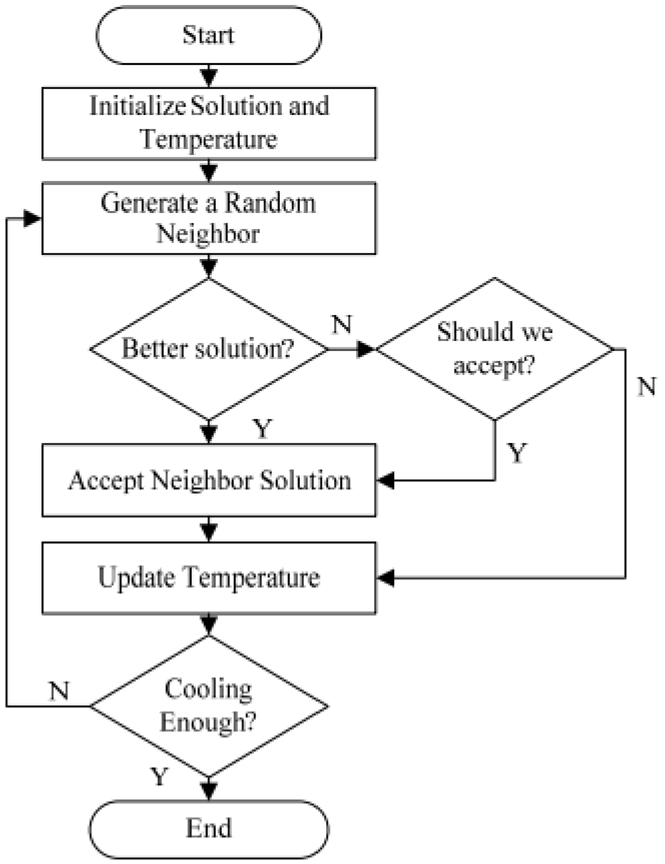
If this modification causes a decrease in the objective function of the system, it is accepted.

In case it causes an increase it is accepted with a probability of $e^{\frac{-\Delta E}{T}}$

By repeatedly observing this Metropolis rule of acceptance, a sequence of configurations is generated. With this formalism in place, it is possible to show that, when the chain is of infinite length (in practical consideration, of "sufficient" length. . .), the system can reach (in practical consideration, can approach) thermodynamic balance (Equilibrium State) at the temperature considered.


Objective
Higher probabilty to accept the move
x'
x
Lower probabilty to accept the move
x'
x
x: initial solution
x': neighbor solution
Search space

Inhomogeneous variant

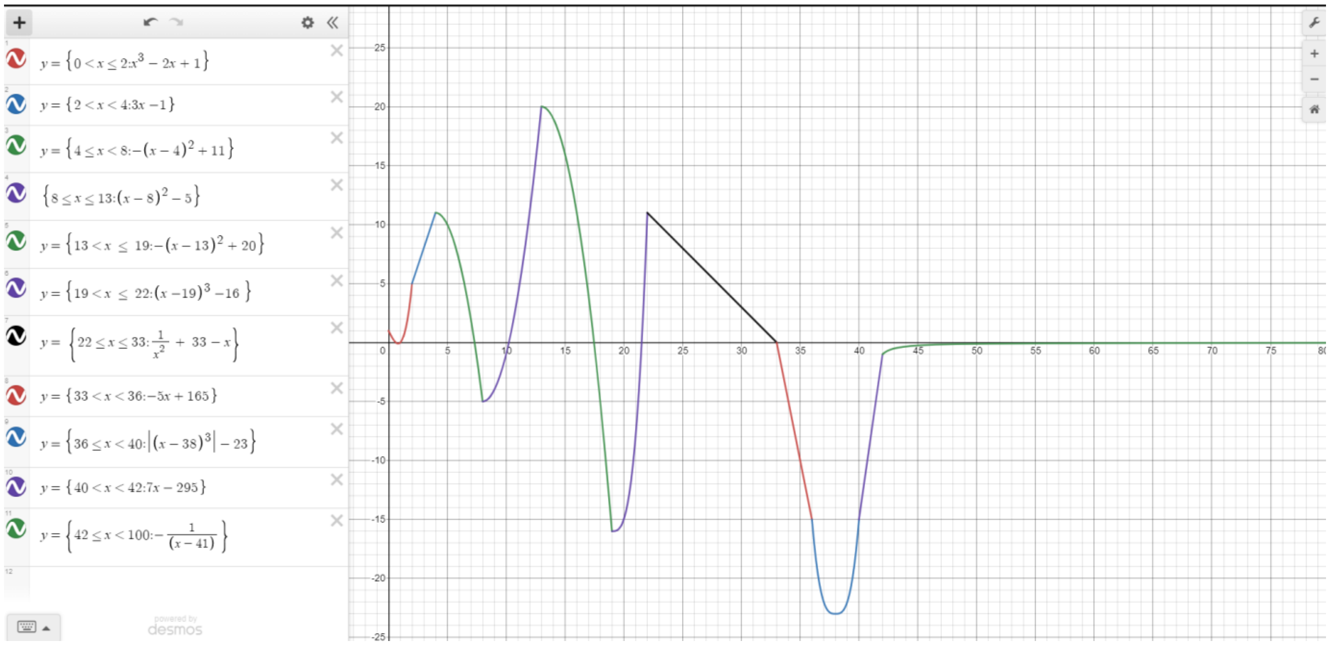


```
**Input:** Cooling schedule.
$s = s_0$ ; /* Generation of the initial solution */
$T = T_{max}$ ; /* Starting temperature */
**Repeat**
    Generate a random neighbor $s'$ ;
    $\Delta E = f(s') - f(s)$ ;
    **If** $\Delta E \leq 0$ **Then** $s = s'$ /* Accept the neighbor solution */
    **Else** Accept $s'$ with a probability $e^{\frac{-\Delta E}{T}}$ ;
    $T = g(T)$ ; /* Temperature update */
**Until** Stopping criteria satisfied /* e.g. $T < T_{min}$ */
**Output:** Best solution found.
```

Implementation

1. The implementation has been done in MIPS assembly language
2. The implementation has been done in the Missouri State University's MARS (MIPS Assembler and Runtime Simulator) application
3. Each random "neighbor" or value is read from a txt file that has been generated with a code written in C with the following annotations:
   a. Each random number is of the integer type
   b. Each random number is part of the range [0 : 200]
   c. There are 600 random numbers generated every time the generator is used
   d. The previous numbers are dicarded
4. The probability of the acceptance is an approximated number:
   a. e^-(deltaF / T) = 1 / e^[deltaF / T] * e^(deltaF / T), where [] <- symbolizes the whole part of a fractional number and () <- symbolizes the fractional part of a fractional number
   b. e^[deltaF / T] is calculated by repeated multiplication of the approximate value of e by 10 digits.
   c. e^(deltaF / T) is rounded to the nearest fraction with 1 digit precision
      i. e^0.25 = e^0.3
      ii. e^0.82 = e^0.8
      iii. This approximation can be improved by expanding on the LUT and the code that finds the closest value to the input

Results



1 $y=\{0<x\le 2:x^3-2x+1\}$

2 $y=\{2<x<4:3x-1\}$

3 $y=\{4\le x<8:-(x-4)^2+11\}$

4 $\{8\le x\le 13:(x-8)^2-5\}$

5 $y=\{13<x\le 19:-(x-13)^2+20\}$

6 $y=\{19<x\le 22:(x-19)^3-16\}$

7 $y=\left\{22\le x\le 33:\frac{1}{x^2}+33-x\right\}$

8 $y=\{33<x<36:-5x+165\}$

9 $y=\{36\le x<40:|(x-38)^3|-23\}$

10 $y=\{40<x<42:7x-295\}$

11 $y=\left\{42\le x<100:-\frac{1}{(x-41)}\right\}$

12

-multiple local minima

-global minima is (38, -23)

-The output of 10 different executions with the random numbers regenerated each time is the following:

T_0 = 1500

S_0 = 40

Nr.	Solution	Result
1	38	-23
2	39	-22
3	38	-23
4	38	-23
5	37	-22
6	39	-22
7	37	-22
8	37	-22
9	38	-23
10	38	-23