Simulated Annealing
Part 1: Basic Concepts

Fall 2009

Instructor: Dr. Masoud Yaghini

Outline

- Introduction
- Real Annealing and Simulated Annealing
- Metropolis Algorithm
- Template of SA
- A Simple Example
- Acceptance Function
- Initial Temperature
- Equilibrium State
- Cooling Schedule
- Stopping Condition
- Handling Constraints
- References

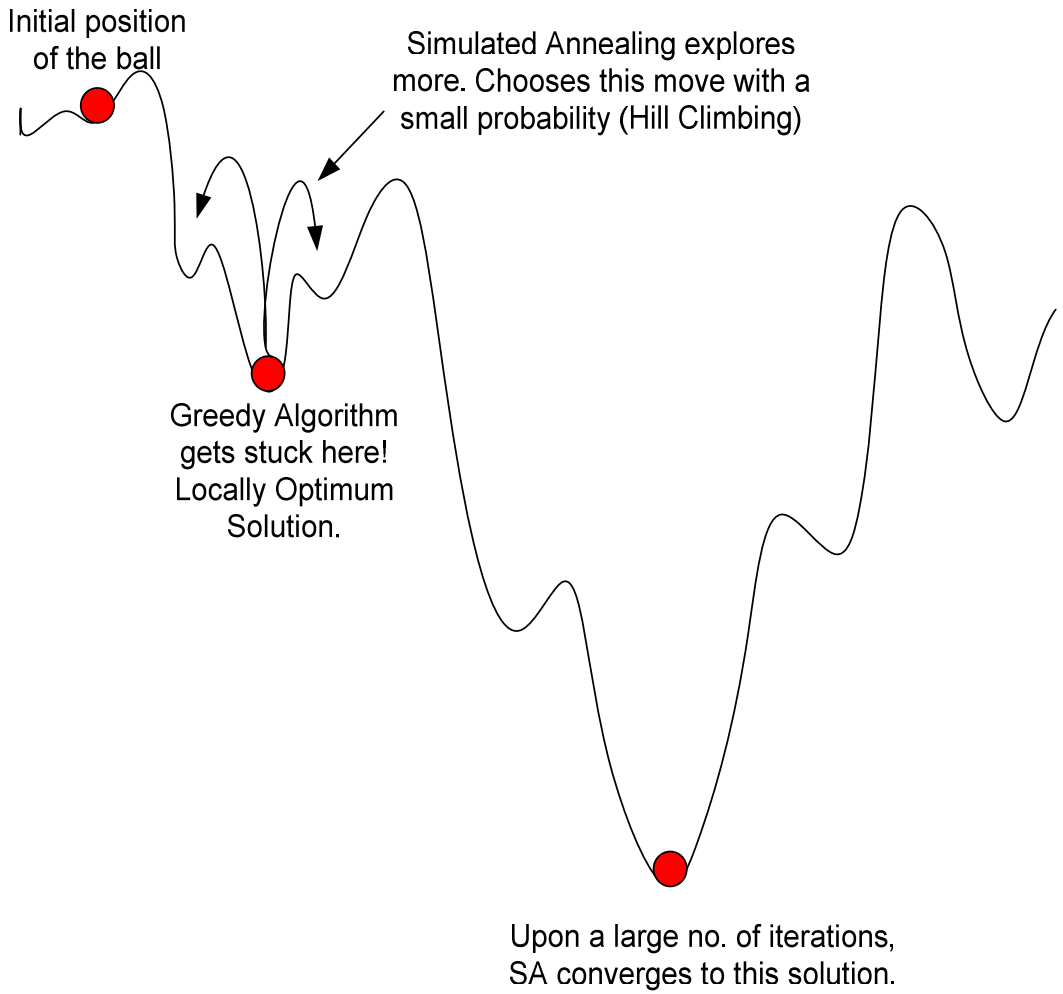
Introduction

What Is Simulated Annealing?

- Simulated Annealing (SA)
  - applied to solve optimization problems
  - is a stochastic algorithm
  - escaping from local optima by allowing worsening moves
  - is a memoryless algorithm in the sense that the algorithm does not use any information gathered during the search
  - is applied for both combinatorial and continuous optimization problems
  - is simple and easy to implement.
  - motivated by the physical annealing process
  - Mathematical proven to converge to global optimum

Simulated Annealing: Part 1

SA vs Greedy Algorithms: Ball on terrain example



History

- Numerical simulation of annealing, Metropolis et al. 1953.

N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. Journal of Chemical Physics, 21:1087–1092, 1953.

History

- SA for combinatorial problems
  - Kirkpatrick et. al, 1986
  - Cerny, 1985

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

V. Cerny, Thermodynamical approach to the traveling salesman problem : an efficient simulation algorithm. J. of Optimization Theory and Applications, 45(1):41–51, 1985.

History

- Originally, the use of simulated annealing in combinatorial optimization
- In the 1980s, SA had a major impact on the field of heuristic search for its simplicity and efficiency in solving combinatorial optimization problems.
- Then, it has been extended to deal with continuous optimization problems
- SA was inspired by an analogy between the physical annealing process of solids and the problem of solving large combinatorial optimization problems.

Applications

- **Basic problems**
  - Traveling Salesman Problem
  - Graph partitioning
  - Matching prob.
  - Quadratic Assignment
  - Linear Arrangement
  - Scheduling
  - ….

Applications

- **Engineering problem**
  - VLSI: Placement, routing…
  - Facilities layout
  - Image processing
  - Code design
  - Biology
  - Physics
  - ….

Real Annealing and Simulated Annealing

Real Annealing Technique

- **Annealing Technique** is known as a thermal process for **obtaining low-energy state** of a solid in a heat bath.
- The process consists of the following two steps:
  - Increase the **temperature** of the heat bath to a maximum value at which the solid melts.
  - Decrease carefully the temperature of the heat bath until the **particles** arrange themselves in the **ground state** of the solid.

Real Annealing Technique

- In the liquid phase all **particles** arrange themselves randomly, whereas in the ground state of the solid, the particles are arranged in a highly structured lattice, for which the corresponding energy is minimal.
- The **ground state** of the solid is obtained only if:
  - the maximum value of the temperature is sufficiently high and
  - the cooling is done sufficiently slow.

Real Annealing Technique

- If the initial temperature is not sufficiently high or a fast cooling is applied, **metastable states** (imperfections) are obtained.
- The process that leads to metastable states is called **quenching**
- Strong solid are grown from careful and slow cooling.
- If the **lowering of the temperature** is done sufficiently slow, the solid can reach **thermal equilibrium** at each temperature.

Real Annealing and Simulated Annealing

- The 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

Real Annealing and Simulated Annealing

- The objective function of the problem is analogous to the energy state of the system.
- A solution of the optimization problem corresponds to a system state.
- The decision variables associated with a solution of the problem are analogous to the molecular positions.
- The global optimum corresponds to the ground state of the system.
- Finding a local minimum implies that a metastable state has been reached.

Metropolis Algorithm

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**.
- Their algorithm is based on Monte Carlo techniques, and generates a sequence of states of the solid in the following way.
- Given a current state i of the solid with energy E_i, a subsequent state j is generated by applying a perturbation mechanism that transforms the current state into a next state by a small distortion, for instance, by a displacement of a single particle.

Metropolis Algorithm

- The energy of the next state is E_j .
- If the energy difference, $E_j - E_i$, is less than or equal to 0, the state j is accepted as the current state.
- If the energy difference is greater than 0, then state j is accepted with a probability given by

$$\exp\left(\frac{E_i - E_j}{k_B T}\right)$$

  - where T denotes the temperature of the heat bath and
  - k_B a physical constant known as the Boltzmann constant.

Metropolis Algorithm

- The acceptance rule described above is known as the Metropolis criterion (Metropolis rule) and the algorithm that goes with it is known as the Metropolis algorithm.
- In the Metropolis algorithm thermal equilibrium is achieved by generating a large number of transitions at a given temperature value.

Template of SA

Template of SA

- Using Metropolis algorithm to simulate the evolution of a physical system towards its thermodynamic balance at a given temperature:
  - On the basis of a given,
  - the system is subjected to an elementary,
  - if this modification causes a decrease in the objective function of the system, it is accepted;
  - if it causes an increase ΔE of the objective function, it is also accepted, but with a probability

$$e^{\frac{-\Delta E}{T}}$$

Template of SA

- 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

Template of SA

- At high temperature, $e^{\frac{-\Delta E}{T}}$ is close to 1,
  - therefore the majority of the moves are accepted and the algorithm becomes equivalent to a simple random walk in the configuration space .
- At low temperature, $e^{\frac{-\Delta E}{T}}$ is close to 0,
  - therefore the majority of the moves increasing energy is refused.
- At an intermediate temperature,
  - the algorithm intermittently authorizes the transformations that degrade the objective function

Template of SA

- SA can be viewed as a sequence of Metropolis algorithms, evaluated at decreasing values of the temperature.

Template of SA

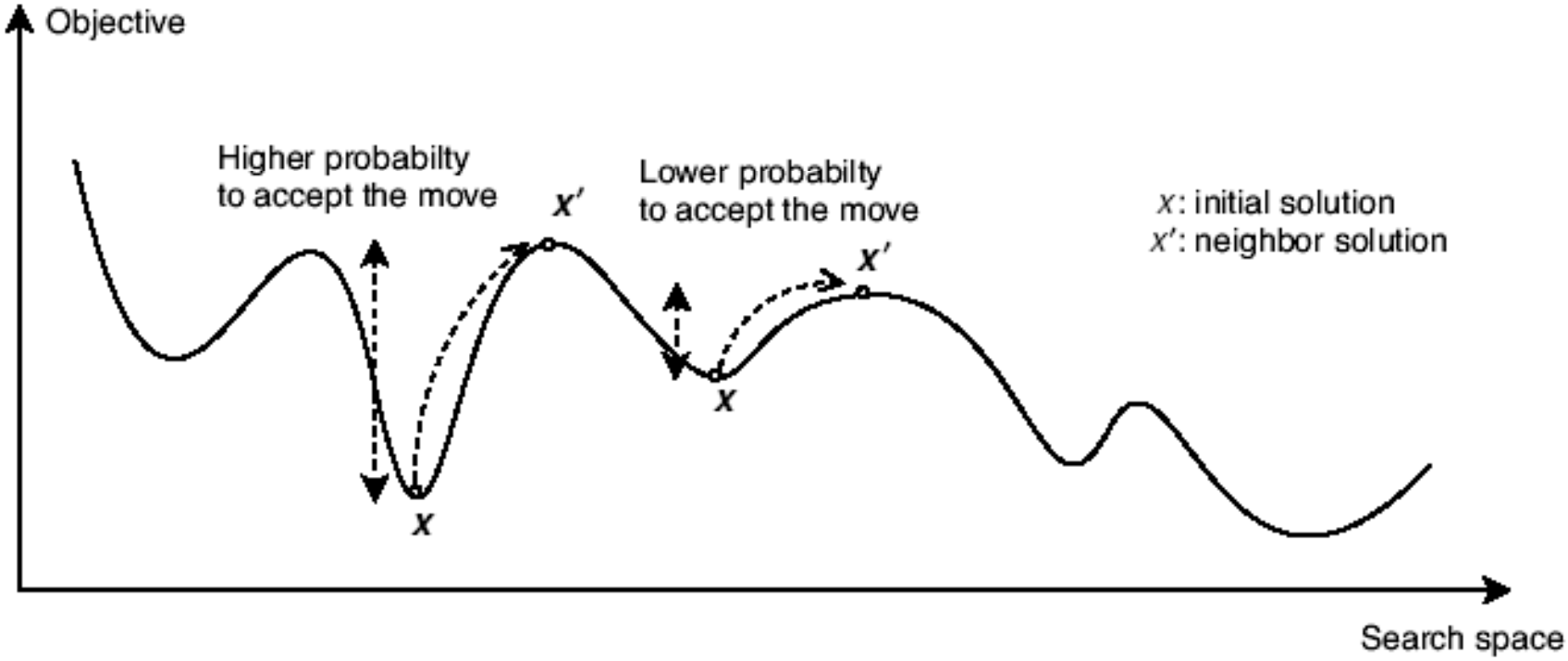
- From an initial solution, SA proceeds in several iterations.
- At each iteration, a random neighbor is generated.
- Moves that improve the cost function are always accepted.
- Otherwise, the neighbor is selected with a given probability that depends on the current temperature and the amount of degradation ΔE of the objective function.
- ΔE represents the difference in the objective value (energy) between the current solution and the generated neighboring solution.

Template of SA

- The higher the temperature, the more significant the probability of accepting a worst move.
- At a given temperature, the lower the increase of the objective function, the more significant the probability of accepting the move.

Template of SA

- As the algorithm progresses, the probability that such moves are accepted decreases.



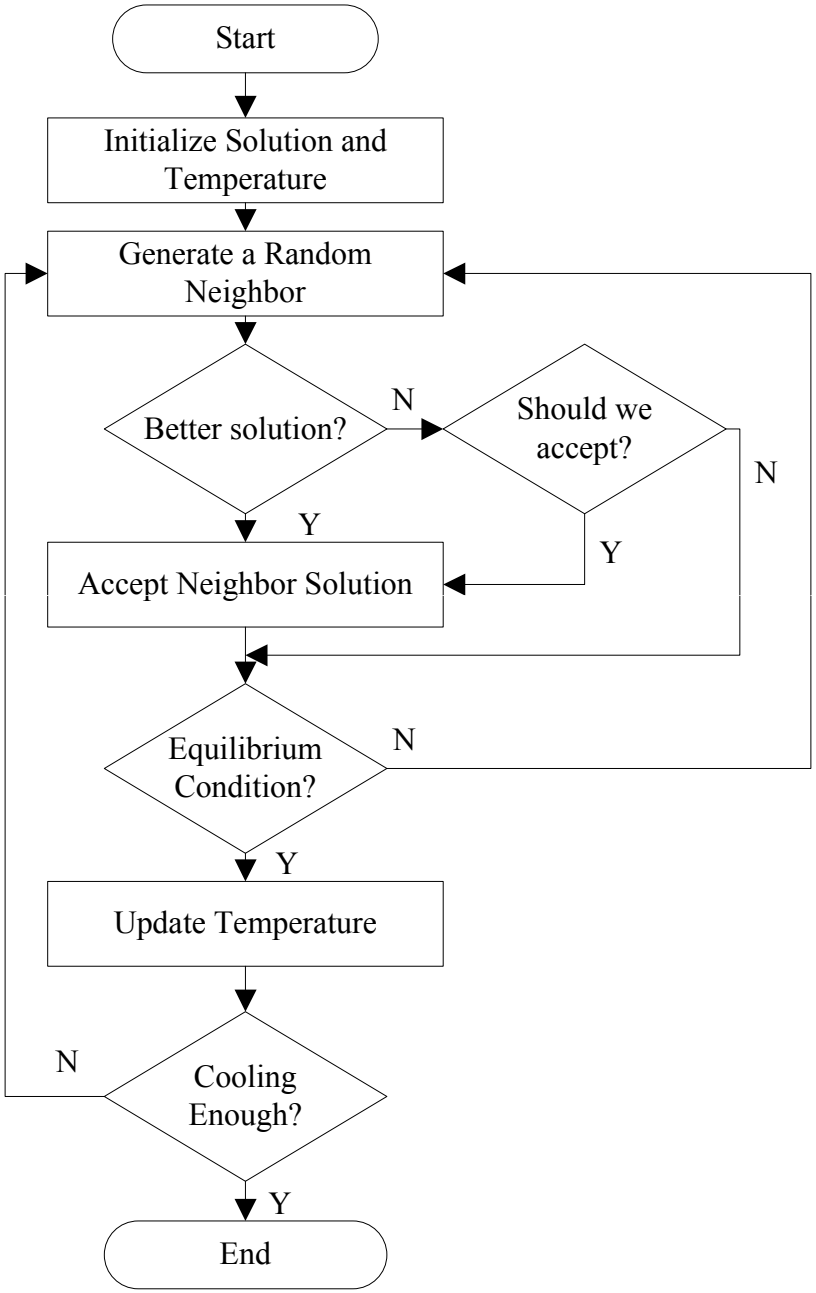
Template of SA

- **The acceptance probability function**, in general, the Boltzmann distribution:

$$P(\Delta E, T) = \mathrm{e}^{-\frac{f(s')-f(s)}{T}}$$

- It uses a control parameter, called temperature, to determine the probability of accepting nonimproving solutions.
- At a particular level of temperature, many trials are explored.
- Once an equilibrium state is reached, the temperature is gradually decreased according to a cooling schedule such that few nonimproving solutions are accepted at the end of the search.

Simulated Annealing: Part 1



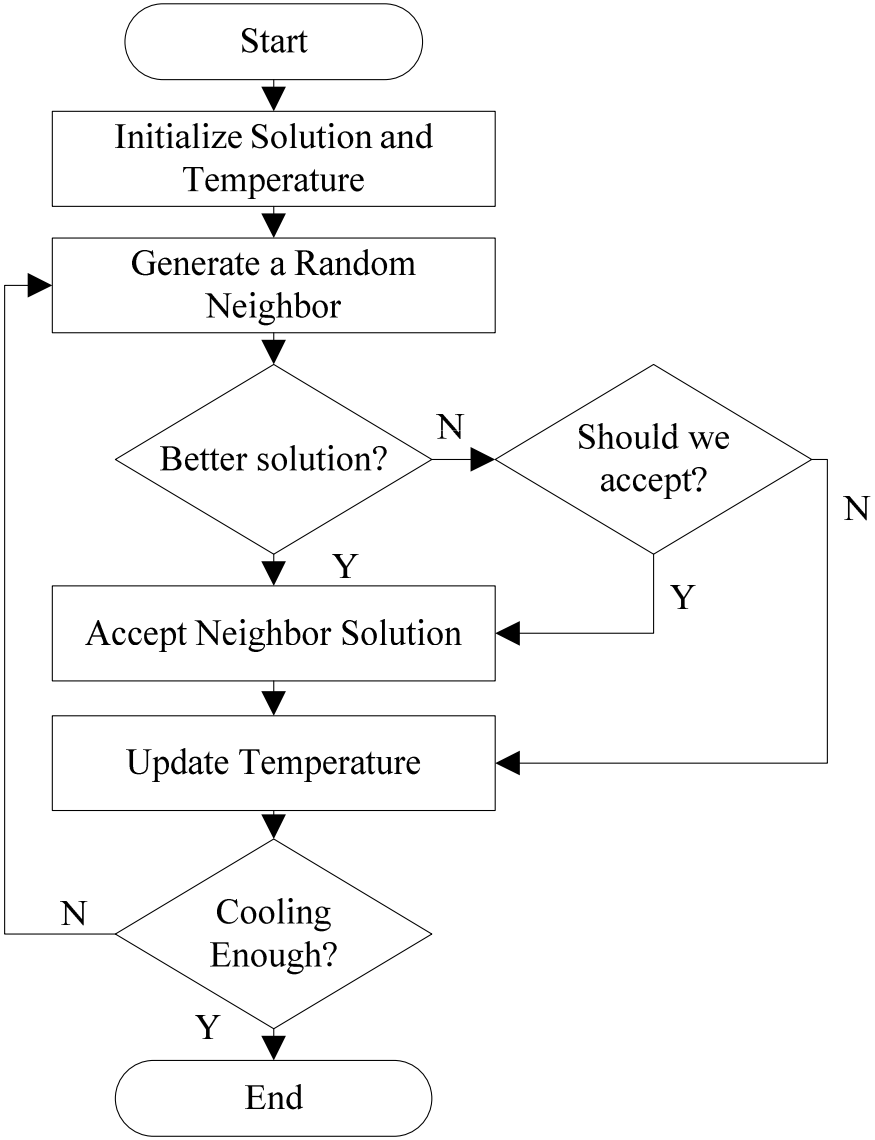
Template of SA

Input: Cooling schedule.
$s = s_0$; /* Generation of the initial solution */
$T = T_{max}$; /* Starting temperature */
Repeat
  Repeat /* At a fixed temperature */
    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 $e^{\frac{-\Delta E}{T}}$;
  Until Equilibrium condition
  /* e.g. a given number of iterations executed at each temperature T */
  $T = g(T)$; /* Temperature update */
Until Stopping criteria satisfied /* e.g. $T < T_{min}$ */
Output: Best solution found.

Inhomogeneous vs. Homogeneous Algorithm

- SA has two variants:
  - **Homogeneous variant**
    - Previous algorithm is the homogeneous variant
    - T is kept constant in the inner loop and is only decreased in the outer loop
  - **Inhomogeneous variant**
    - There is only one loop
    - T is decreased each time in the loop, but only very slightly

Inhomogeneous variant



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.

Cooling Schedule

- The cooling schedule defines for each step of the algorithm i the temperature T_i.
- It has a great impact on the success of the SA optimization algorithm.
- The parameters to consider in defining a cooling schedule are the starting temperature, the equilibrium state, a cooling function, and The final temperature that defines the stopping criteria

Template of SA

- Main components of SA:
  - **Acceptance Function**
  - **Initial Temperature**
  - **Equilibrium State**
  - **Cooling Function**
  - **Stopping Condition**

A Simple Example

A Simple Example

- Let us maximize the continuous function

  $f(x) = x^3 - 60x^2 + 900x + 100.$
- A solution x is represented as a string of 5 bits.
- The neighborhood consists in flipping randomly a bit.
- The global maximum of this function is:

  01010 (x = 10, f (x) = 4100)
- The initial solution is 10011 (x = 19, f (x) = 2399)
- Testing two sceneries:
  - First scenario: initial temperature T_0 equal to 500.
  - Second scenario: initial temperature T_0 equal to 100.
- Cooling: T = 0.9 . T

A Simple Example

- In addition to the current solution, the best solution found since the beginning of the search is stored.
- Few parameters control the progress of the search, which are:
  - The temperature
  - The number of iterations performed at each temperature

A Simple Example

- First Scenario T = 500 and Initial Solution (10011)

T	Move	Solution	f	Δf	Move?	New Neighbor Solution
500	1	00011	2287	112	Yes	00011
450	3	00111	3803	<0	Yes	00111
405	5	00110	3556	247	Yes	00110
364.5	2	01110	3684	<0	Yes	01110
328	4	01100	3998	<0	Yes	01100
295.2	3	01000	3972	16	Yes	01000
265.7	4	01010	**4100**	<0	Yes	01010
239.1	5	01011	4071	29	Yes	01011
215.2	1	11011	343	3728	No	01011

A Simple Example

- Second Scenario: T = 100 and Initial Solution (10011).

T	Move	Solution	f	Δf	Move?	New Neighbor Solution
100	1	00011	2287	112	No	10011
90	3	10111	1227	1172	No	10011
81	5	10010	2692	< 0	Yes	10010
72.9	2	11010	516	2176	No	10010
65.6	4	10000	**3236**	< 0	Yes	10000
59	3	10100	2100	1136	Yes	10000

- When Temperature is not High Enough, Algorithm Gets Stuck

Acceptance Function

Acceptance Function

- The system can escape from local optima due to the probabilistic acceptance of a nonimproving neighbor.
- The probability of accepting a nonimproving neighbor is proportional to the temperature T and inversely proportional to the change of the objective function ΔE.

Acceptance Function

- The acceptance probability of a nonimproving move is:

$$P(\Delta E, T) = e^{\frac{-\Delta E}{T}} > R$$

  - where E is the change in the evaluation function,
  - T is the current temperature, and
  - R is a uniform random number between 0 and 1.

Acceptance Function

- At high temperatures,
  - the probability of accepting worse moves is high.
  - If $T = \infty$, all moves are accepted, which corresponds to a random local walk in the landscape.
- At low temperatures,
  - the probability of accepting worse moves decreases.
  - If $T = 0$, no worse moves are accepted and the search is equivalent to local search (i.e., hill climbing).
- Moreover, the probability of accepting a large deterioration in solution quality decreases exponentially toward 0 according to the Boltzmann distribution.

To accept or not to accept?

Change	Temp	exp(-ΔE/T)		Change	Temp	exp(-ΔE/T)
0.2	0.95	0.810157735		0.2	0.1	0.135335283
0.4	0.95	0.656355555		0.4	0.1	0.018315639
0.6	0.95	0.531751530		0.6	0.1	0.002478752
0.8	0.95	0.430802615		0.8	0.1	0.000335463

Initial Temperature

Initial Temperature

- If the starting temperature is very high,
  - the search will be a random local search for a period of time
  - accepting all neighbors during the initial phase of the algorithm.
  - The main drawback of this strategy is its high computational cost.
- If the initial temperature is very low,
  - the search will be a local search algorithm.
- Temperature must be high enough to allow moves to almost neighborhood state.
- Problem is finding a suitable starting temperature

Initial Temperature

- **Acceptance deviation**
  - The starting temperature is computed using preliminary experimentations by:

    $k\sigma$
  - where σ represents the standard deviation of difference between values of objective functions and
  - $k = -3/\ln(p)$ with the acceptance probability of p, which is greater than 3σ

Initial Temperature

- **Tuning for initial temperature**
  - Start high, reduce quickly until about 60% of worse moves are accepted.
  - Use this as the starting temperature

Equilibrium State

Equilibrium State

- Once an equilibrium state is reached, the temperature is decreased.
- To reach an equilibrium state at each temperature, a number of sufficient transitions (moves) must be applied.
- The number of iterations must be set according to:
  - The size of the problem instance and
  - Particularly proportional to the neighborhood size $|N(s)|$

Equilibrium State

- The strategies that can be used to determine the number of transitions visited:
  - **Static strategy**
  - **Adaptive strategy**

Equilibrium State

- **Static strategy**
  - The number of transitions is determined before the search starts.
  - For instance, a given proportion y of the neighborhood N(s) is explored.
  - Hence, the number of generated neighbors from a solution s is $y \cdot |N(s)|$.
  - The more significant the ratio y, the higher the computational cost and the better the results.

Equilibrium State

- **Adaptive strategy**
  - The number of generated neighbors will depend on the characteristics of the search.
  - One adaptive approach is an improving neighbor solution is generated.
  - This feature may result in the reduction of the computational time without compromising the quality of the obtained solutions
  - Another approach is achieving a predetermined number of iterations without improvement of the best found solution in the inner loop with the same temperature

Cooling Schedule

Cooling Schedule

- In the SA algorithm, the temperature is decreased gradually such that $T_i > 0, \forall i$
- There is always a compromise between the quality of the obtained solutions and the speed of the cooling schedule.
- If the temperature is decreased slowly, better solutions are obtained but with a more significant computation time.

Cooling Schedule

- The temperature T can be updated in different ways:
  - **Static Strategy**
    - **Linear**
  - **Dynamic Strategy**
    - **Geometric**
    - **Logarithmic**
  - **Adaptive Strategy**

Cooling Schedule

- **Linear**
  - In the trivial linear schedule, the temperature T is updated as $T = T - \beta$, where β is a specified constant value.
  - Hence, we have

    $$T_i = T_0 - i \times \beta$$

  - where T_i represents the temperature at iteration i.
  - β is a specified constant value
  - T_0 is the initial temperature

Cooling Schedule

- **Geometric**
  - In the geometric schedule, the temperature is updated using the formula

    $T_{i+1} = \alpha.T_i$
  - where $\alpha \in]0, 1[$.
  - It is the most popular cooling function.
  - Experience has shown that α should be between 0.5 and 0.99.

Cooling Schedule

- **Logarithmic**
  - The following formula is used:

$$T_i = \frac{T_0}{\log(i+10)}$$

  - This schedule is too slow to be applied in practice but has the property of the convergence proof to a global optimum.

Cooling Schedule

- **Adaptive Strategy**
  - Most of the cooling schedules are static or dynamic in the sense that the cooling schedule is defined completely a **priori**.
  - In this case, the cooling schedule is “blind” to the characteristics of the search landscape.
  - In an adaptive cooling schedule, the decreasing rate is depends on some information obtained during the search.

Stopping Condition

Stopping Condition

- Concerning the stopping condition, theory suggests a final temperature equal to 0.
- In practice, one can stop the search when the probability of accepting a move is negligible.

Stopping Condition

- The following stopping criteria may be used:
  1. Reaching a final temperature T_F is the most popular stopping criteria.
     - This temperature must be low (e.g., $T_{min} = 0.01$).
  2. Achieving a predetermined number for successive temperature values no improvement in solution quality
  3. After a fixed amount of CPU time
  4. When the objective reaches a pre-specified threshold value

Handling Constraints

Handling Constraints

- Constraints cannot handled implicitly
  - Penalty function approach should be used

- **Constraints**
  - **Hard Constraints**: these constraints cannot be violated in a feasible solution
  - **Soft Constraints**: these constraints should, ideally, not be violated but, if they are, the solution is still feasible

Handling Constraints

- Hard constraints are given a large weighting.
  - The solutions which violate those constraints have a high cost function
- Soft constraints are weighted depending on their importance
- Weightings can be dynamically changed as the algorithm progresses.
  - This allows hard constraints to be accepted at the start of the algorithm but rejected later

References

References

- El-Ghazali Talbi, **Metaheuristics : From Design to Implementation, John Wiley & Sons, 2009.**
- J. Dreo A. Petrowski, P. Siarry E. Taillard, **Metaheuristics for Hard Optimization**, Springer-Verlag, 2006.

The End