Simulated Annealing

MENTOR: Prof. Nabin Kumar Sahu

Jayesh Popat
DAIICT
Gandhinagar, India
201801003@daiict.ac.in

Srinivas Talnikar
DAIICT
Gandhinagar, India
201801406@daiict.ac.in

***Abstract*—Simulated annealing is a well-renowned meta-heuristic algorithm employed to address the black box optimization problems related to searching for global optimums. This paper explores the technique of Simulated Annealing and aims to study its extended applications in the field of communication design and network synthesis.**

I. Introduction

Simulated Annealing is a simple yet highly versatile algorithm finding myriad applications rooted in the fields of optimization and artificial intelligence. It is also extensively used in real world simulation problems such as network synthesis and communication design. This report aims to presents a brief discussion on simulated annealing reviewing its application to some pragmatic optimization problems. The nomenclature of Simulated Annealing is established upon its analogy to the physical annealing of materials. Physical Annealing is a process in which a crystalline solid is heated and then permitted to cool over a period allowing it achieve its most stable state of lattice configuration devoid of crystalline defects. Larger the cooling period, more superior the final structural integrity results. Simulated annealing incorporates this thermodynamic behaviour into its algorithm for the search of global minima in a discrete optimization problem. At each iteration of the algorithm the objective function generates a new states, eventually superior states with improved solutions are accepted, while a dynamic fraction of inferior states are permitted which enables us to escape the local optimum in search of a global maximum. Analogous to physical annealing, the dynamic fraction is computed based on the current temperature or the volatility of the algorithm. The temperature is a non-increasing parameter, initially high rendering a larger search space of solutions to explore, which is eventually decreased as we reach closer to a opitma.

II. Algorithm Analysis

Simulated annealing is established upon the metropolis acceptance criterion. Which governs how a thermodynamic system transists from its current state to a candidate state. Mathematically, at every iteration the probability that the genreated candidate state (say $\omega^{'}$) is selected as the next solution over the current state (say ω) is given by

$$P_{\omega \rightarrow \omega'} = \begin{cases} e^{[-\frac{(cost(\omega^{'})-cost(\omega))}{t_k}]} & cost(\omega^{'}) > cost(\omega) \\ 1 & cost(\omega^{'}) \leq cost(\omega) \end{cases}$$

Here t_k is the temperature parameter at the particular iteration.

Let Ω be the sample space of all the possible neighbors of current state ω. We define the probability of generating the candidate state $\omega^{'}$ as $g_k(\omega, \omega^{'})$. The transition probabilities P_t which defines the probability the algorithm transitions to $\omega^{'}$, encompasses three possible scenarios:

$$P_t(\omega, \omega^{'}) = \begin{cases} g_k(\omega, \omega^{'}) \times exp(-\frac{\delta_{cost}}{t_k}) & \omega^{'} \in \Omega \\ 0 & \omega^{'} \notin \Omega \\ 1 - \sum_{\omega^{''} \in \Omega, \omega^{''} \neq \omega} P_t(\omega, \omega^{''}) & \omega^{'} = \omega \end{cases}$$

These transition probabilities signify a series of states fabricated from an inhomogenous Markov Chain.

A. Convergence Criterion

Shoshana Anily, Awi Federgruen, using inhomogenous markov chain theory, demonstrated the necessary and sufficient conditions for the convergence of simulated annealing algorithm

- The acceptance probability function must allow any hill-climbing transition to occur with a non-zero probability, in each and every iteration of the algorithm.
- The acceptance probability function must be bounded and asymptomatically monotone, with limit zero for hill-climbing solution transition.
- In the limit, the stationary probability distribution must have zero probability mass for every non-globally optimal soluton.
- The probability of escaping from any locally optimal solution must not approach zero too quickly

B. Psuedocode

Now let's move on to the Pseudocode for simulated annealing. Assume that it starts from the initial state s_0 and the process continues until a maximum of k_{max} steps have been taken. During this process, the call neighbour should generate a randomly chosen neighbour of a given state s; the function call random(0,1) will return a value from the range [0,1] which will be a completely random value. The call temperature(x) is responsible to define the annealing schedule; based on the fraction x of the time that has been expended so far, it will provide the temperature to use.

Below is the pseudo-code for Simulated Annealing.

```
Let $s = s_0$
for k = 0 to $k_{max}$ do
    $T \leftarrow temperature(\frac{k+1}{k_{max}})$
    Pick a random neighbour $\implies$ $s_{new} \leftarrow neighbour(s)$
    if $P(E(s), E(s_{new}, T) \geq random(0,1)$ then
        $s \leftarrow s_{new}$
    end if
end for
Output : final state s
```

Fig.(1) demonstrates the flowchart which concisely dictates the flow of the simulated annealing algorithm.

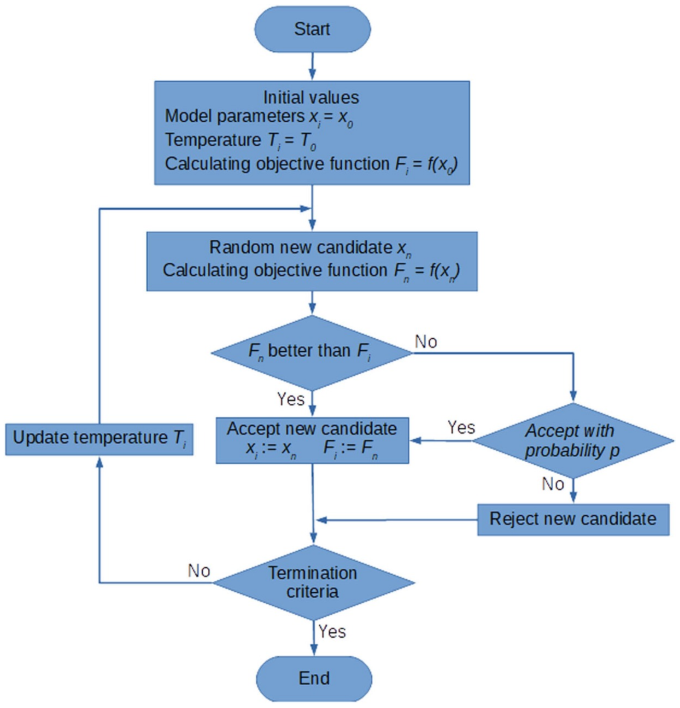
Start
Initial values
Model parameters $x_i = x_0$
Temperature $T_i = T_0$
Calculating objective function $F_i = f(x_0)$
Random new candidate x_n
Calculating objective function $F_n = f(x_n)$
F_n better than F_i
No
Yes
Accept new candidate
$x_i := x_n$ $F_i := F_n$
Yes
Accept with probability p
No
Reject new candidate
Update temperature T_i
No
Termination criteria
Yes
End

Fig. 1. Flowchart for Simulated Annealing Algoritm

III. Solving Sudoku

The first problem we solved using simulated annealing is the very well known game sudoku. The cost function we used for solving sudoku was the sum of number of duplicates in each row and column. Initially we would be provided with an unsolved sudoku and then the algorithm starts by picking up a 3x3 block of sudoku and then randomly filling integers from 1-9 which weren't used in block; once we are done with this block we pick up the next 3x3 block and do the same process. This will give us a sudoku which may/may not be solved. Now we will use the cost function and slowly decrease the temperature and try to improve the cost function; if we reach the best cost function(0 in this case) then we stop the loop and print the answer however if not then we randomly swap two integers in the block(which we have filled in the initial stage of algorithm). However, sometimes it may be possible that we get stuck somewhere from which we cannot reach solution in a feasible time (since swapping is completely random); so if we do not improve our cost in eighty iterations then we simply increase the temperature. The reason why we are again increasing the temperature is because we want to solve sudoku however this will cost us some extra time but if want a acceptable solution, then this algorithm does provide it and also the cost function of the solution would be considerably low(may/may not be zero). The link for the code of sudoku is provided in the references section.

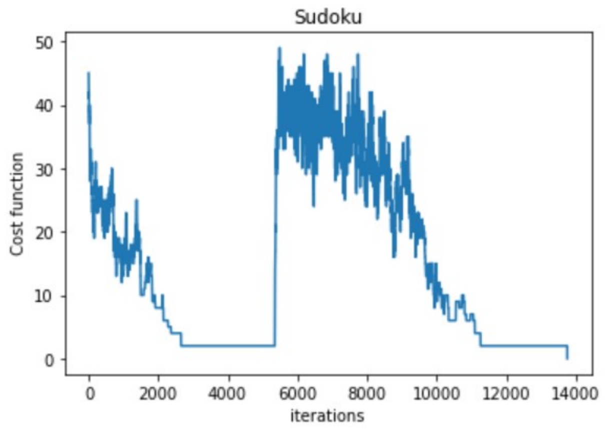

Fig. 2. Sudoku

Fig(2) demonstrates the cost of objective function of during the runtime of the algorithm. We can observe that after around 3000th iteration the cost function roughly becomes constant for awhile, indicating the algorithm is stuck in a local minima. This local minima is then escaped by reheating the cooling schedule in search of a global optimum.

IV. Communication Problem

A. Introduction

Now let us discuss on the network synthesis problem. Our aim would be to satisfy all traffic constraints at the minimum possible cost for a set of nodes. Let's say you have a set of n nodes then there are n^{n-2} different topologies possible. This is really a high number so even if you have around ten nodes there would be around a billion possibilities. The problem can be formulated based on some requirements i.e. the traffic that can be handled between every O-D(origin and destination) pair along with the cost required for the traffic on every possible path(link) between the nodes.

B. Formulation

There are many different ways that the network synthesis problem can be formulated. Let's start by discussing the traditional approach (linear programming approach) first. Assume S as the set of all routes possible with n number of nodes. Let g ϵ S. Let us represent g by its upper triangular node adjacency matrix A with elements a_{ij}. Our task is to find a memeber g* which minimizes the cost for the specific O-D pair subject to the constraints(node capacity, hop limit etc.). The total bandwidth required on the path between O-D pair $x - y$ is given by F^{x-y}. The partial flow along the k^{th} route between node x and node y is expressed as h_r^{pq} and the cost per unit flow on this route would be C_r^{pq}.

Objective Function

$$Minimize(g \in \mathbf{S}) \sum_{x=1}^{n} \sum_{y>x} \sum_{k} C_k^{yx} h_k^{yx} \tag{1}$$

such that

$$f_{ij} = \sum_{x=1}^{n} \sum_{y>x} \sum_{r} c_r^{qp} h_r^{qp} \; \forall i,j \tag{2}$$

$$\sum_{k} h_k^{yx} = F^{xy} \; \forall x,y \tag{3}$$

$$\frac{1}{2}\left[\sum_{x=1}^{n}(F^{xi} + F^{ix}) + \sum_{j \neq i} f_{ij}\right] \le u_i^{max} \tag{4}$$

$$0 \le f_{ij} \le f_{ij}^{max} \; \forall i,j \tag{5}$$

$$\sum_{(i,j)} c_{(i,j),k}^{xy} \le h^{max} \; \forall x,y,k \tag{6}$$

$$\sum_{j=1}^{n}(a_{ij} + a_{ji}) \le d_i^{max} \; \forall i \tag{7}$$

$$0 \le h_k^{xy} \; \forall x,y,k \tag{8}$$

Where:

- N is the number of nodes
- F^{xy} is the total bandwidth required between O-D pair x-y.
- $c_{ij,k}^{xy}$ is 1 if the link x-y exists on route k between nodes i,j; 0 otherwise.
- f_{ij}^{max} is the upperbound on the available capacity(total flow) on the link(i,j).

- u_i^{max} is the upperbound on the total flow at node i. This flow comprises traffic originating at, terminating at and traversing node i.
- H^{max} is the upperbound imposed on the number of links in a route(the hop limit).
- h_k^{xy} is the amount of traffic routed between x and y on route k.

The equations written above represent the model of the network synthesis problem. The first equation is the objective function that is supposed to minimize(it is basically the traffic costs). The second equation is to calculate total flow on every link. The next equation takes care of the distribution of bandwidth. Equation 4 makes sure that the node capacities are not exceeded. the fifth constraint takes care of the link capacities. The sixth and seventh equation takes care of the hop limit and ensures that the nodes meet the node degree constraints.

1) Pre-Computed Routes Approach: Here we will discuss the idea i.e. before the application of an optimisation technique; a set of routes can be precomputed. The length of the route should strictly not exceed the hop limit. There are some advantages when using this approach:

- Hop limits are automatically satisfied
- All the routes would be feasible.
- The problem reduces to selection of routes which is quite similar to items selected for a knapsack in the knapsack problem.

However there is also a disadvantage that the memory used can be quite large if the set of routes is large which can be the case if the hop limit is high. We will come back to this point later in our discussion.

With this approach we can apply common local search operators within Simulated Annealing (SA) to locate feasible solution states. We also need to allocate traffic amongst the selected routes.

Now let us talk little bit on local search transition operators that are appropriate for the network synthesis problem i.e. add,drop and change.

- Add: A route is added to the network.
- Drop: A route is removed from the network.
- Change: A route in the network is changed to another route. (basically combination of add and drop)

This is graphically demonstrated in fig 3.

In our SA implementation, all the three operators are assigned an equal probability of being selected at every iteration of the algorithm. The algorithm is used to synthesise a suitable network topology. There might be multiple routes for some O-D pair; so one of the subproblem is to determine an appropriate allocation of traffic for each route. There are two main solutions to this problem.

- exact solution: Finding the optimal allocation of the traffic.
- heuristic approach: The allocation is determined by a special purpose algorithm.

The first approach does find the optimal solution however its time complexity would be quite high and sometimes not even practically possible. In contrast to this the heuristic approach may not find the optimal allocation however it does ensure to satisfy equation 3. Here we will discuss the heuristic approach which will yield acceptable solutions in reasonable amount of computer time.

Our approach is simple and efficient. In essence, for every O-D pair and its set of routes, the algorithm proportionally loads traffic on every route according to its cost. As a result the routes which will be less expensive will be given more traffic.

Where:

- rc(n) is the cost of route n.
- h(i,j,n) is the allocation of traffic of route n between (i,j).
- d(i,j) is the demand between nodes i and j.

V. Implementing Simulated Annealing

- Selecting the size of initial route set is an imminent step for the algorithm to run in a pragmatic time constraint. Since the entire set of routes can be colossal, only a specified percentage of such routes is selected in the precomputed set.
- Reheating provides algorithm to escape its stranded local minima in search of global minimum. Ergo, reheating should be applied a numerous times for the algorithm to render desirable results. There are variety of standardised cooling schedules available for simulated annealing, which could help dictate the proper flow of algorithm, out of those Connoly's cooling schedule has consistently shown promising

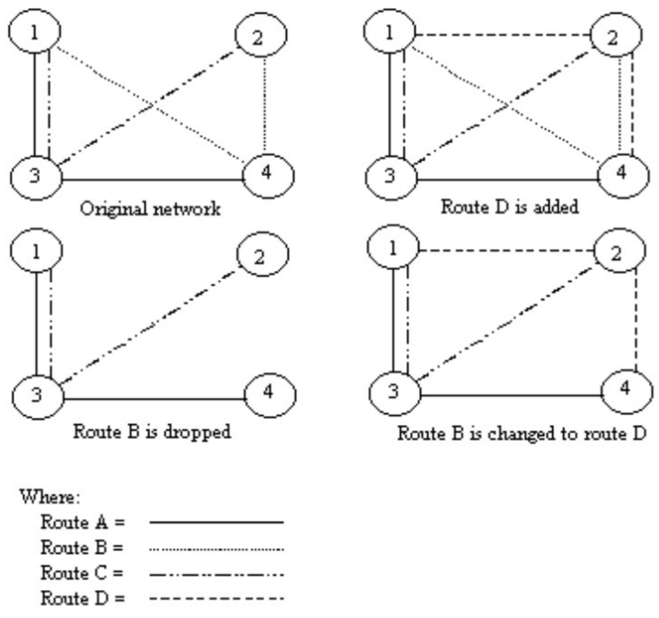

Fig. 3. The transition operators add, drop and change. The figure above demonstrates the effects of all the three operators.

```
for each O-D pair(i,j) do
    $sc \leftarrow 0$
    for n $\in$ routes in current solution between(i,j) do
        sc $\leftarrow$ sc + rc(n);
    end for
    if number of routes in the solution between(i,j)$>1$ then
        $sca \leftarrow 0$
        for n $\in$ routes in current solution between(i,j) do
            sca $\leftarrow$ sca + rc(n);
        end for
        for n $\in$ routes in current solution between(i,j) do
            h(i,j,n) $\leftarrow$ d(i,j) $\times$ (sc-rc(n)) $\div$ sca;
        end for
    else
        n $\leftarrow$ number of the single route between (i,j);

        h(i,j,n) $=$ d(i,j);
    end if
end for
```

results. Connoly's cooling schedule allows alteration in both the number of times reheating occurs and the intervals between reheats.

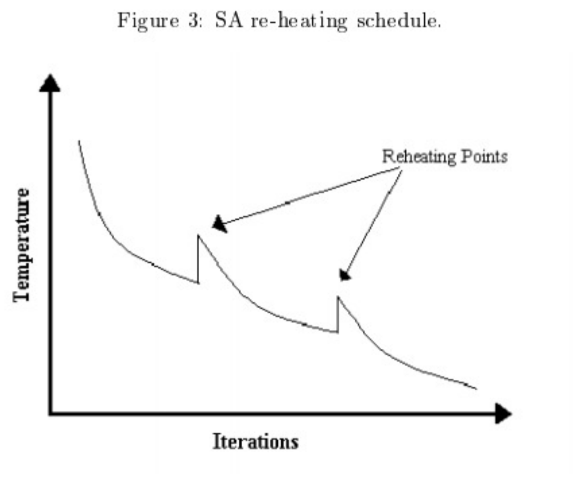

Fig. 4. Cooling Schedule Visualization

- Due to largely constrained nature of the problem, it is onerous to form an initial feasible solution. The search for an initial feasible solution can be divided into two phases.

  - The first phase involves fabricating a solution with at least one route per O-D pair.
  - The second phase involves making the above solution feasible. This can be done using Simulated Annealing, aiming to minimising the amount of constraint violation to zero. The objective function accounts the difference between LHS and RHS in the violated constraint. For instance, for a constraint relation of type LHS $\leq$ RHS, with LHS $>$ RHS, the degree of violation is the absolute difference between RHS and LHS.

VI. Conclusion

The report elaborated Simulated Annealing, a global optimization meta-heuristic. It highlights the significance of such a simple yet efficient algorithm capable of solving complex network problems with rigid and difficult constraints. This algorithms has unprecedented applications in eclectic domains of real world NP-Hard simulations. The report studies an interesting approach to solve the problem of network synthesis using Simulated Annealing. The heuristic allocates the traffic for each Origin Destination pair. It does so greedily, allocating larger traffic to less costlier routes. The reports also accentuates the necessary and sufficient conditions for the convergence of the algorithm and provides a detailed pictorial and applied analysis through the use of proper visuals and a simulation of Sudoku.

VII. Future Work

- In the future we would like to research on an adaptive discretization algorithm for the design of water usage and treatment networks and try to solve more real world problems using simulated annealing.
- We would also like to explore this niche of optimization for some more evolutionary algorithms like simulated annealing and juxtapose their performances in real world simulations. Genetic algorithm is one such technique which too is highly efficient in solving NP-Hard optimization problems. Genetic algorithms attempts to mimic the evolutionary behaviour of biological systems. Fabricating a series of population of candidate solutions, the algorithm transforms each candidate population into a descendent poplution using genetically inspired transition operators.

References

[1] Marcus Randall, Stephen J. Sugden, A Simulated Annealing Approach to Communication Network Design

[2] https://colab.research.google.com/drive/1tg2oMibRbiL_JhBzvD6K-dzOk0S-PaV7?usp=sharing

[3] Henderson, Darrall and Jacobson, Sheldon and Johnson, Alan. (2006). The Theory and Practice of Simulated Annealing.