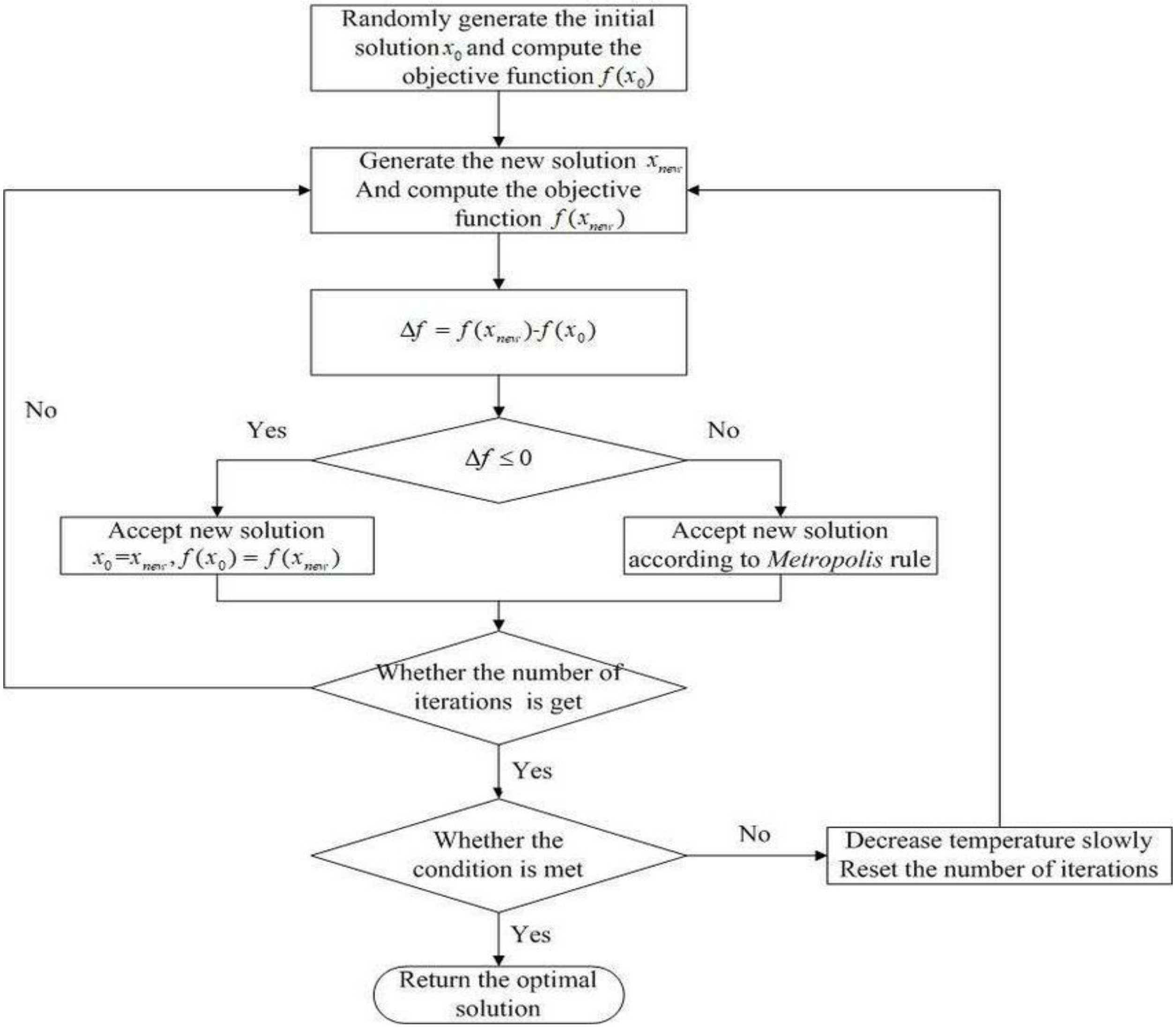
Simulated Annealing

Simulated annealing (SA) is an effective and general form of optimization and is based on simulating the annealing of solids. It is very useful in finding global optima in the presence of local-optima large numbers. The simulated-annealing algorithm starts from a higher temperature, which is called the initial temperature. When the temperature gradually decreases, the solution of the algorithm tends to be stable. However, the solution may be a local optimal solution. Then in SA, such a local optimal solution is jumped with a certain probability to find the global optimal solution of the objective function. The flow chart of the simulated-annealing algorithm is shown in Figure1.



Randomly generate the initial solution x_0 and compute the objective function $f(x_0)$
Generate the new solution x_{new} And compute the objective function $f(x_{new})$
$\Delta f = f(x_{new})$-$f(x_0)$
No
Yes
No
$\Delta f \leq 0$
Accept new solution $x_0 = x_{new}, f(x_0) = f(x_{new})$
Accept new solution according to *Metropolis* rule
Whether the number of iterations is get
Yes
Whether the condition is met
No
Decrease temperature slowly Reset the number of iterations
Yes
Return the optimal solution

Figure 1. A flowchart of the simulated-annealing algorithm

From Figure 1, we can see that the Metropolis rule is very important for SA to find the optimal solution. Suppose that state x_{old} becomes a state x_{new} when the system is subject to some disturbance. Then, in the Metropolis rule, the energy of the system also changes from $E(x_{old})$ to $E(x_{new})$, and the acceptance probability of the system changing from state x_{old} to state x_{new} is p:

$$p = \begin{cases} 1 & if\ \ \mathrm{E}(\mathrm{x}_{\mathrm{new}}) < \mathrm{E}(\mathrm{x}_{\mathrm{old}}) \\ e^{-\frac{E(\mathrm{x}_{\mathrm{new}}) - \mathrm{E}(\mathrm{x}_{\mathrm{old}})}{T}} & if\ \ \mathrm{E}(\mathrm{x}_{\mathrm{new}}) \geq \mathrm{E}(\mathrm{x}_{\mathrm{old}}) \end{cases}$$

The steps of the simulated-annealing algorithm are as follows:

```
begin:
    $X_{old}$: = initial solution
    $T_{\circ}$: = initial temperature value
    repeat
        for i: = 1 to L do
        begin
            Probabilistically generate neighbor $x_{new}$ of $x_{old}$
            if E ($x_{new}$) < E ($x_{old}$) then accept $x_{new}$
            else accept $x_{new}$ with probability $e^{-\frac{E(x_{new})-E(x_{old})}{T}}$
        end
        update i
        decrease $T_{\circ}$
    until stop criterion
end;
```

Simulated Annealing Algorithm steps for Fair Supplementary Exams Scheduling:

1. Define the problem: The goal is to schedule a fair supplementary exams table for the students of electrical engineering department given the starting and ending dates as well as the dates of common materials between departments. The hard constraint is that no student should have two consecutive exams in the same week.
2. Calculate the total number of exams that each student needs to take based on the number of failed exams
3. Determine the common exams among departments and their dates.
4. Divide the remaining exams randomly across the available time period to generate an initial solution.
5. Calculate the initial cost: Calculate the cost of the initial solution based on fairness criteria such as minimizing the number of consecutive exams for each student (hard constraint).
   Cost = number of violations of the hard constraint.
6. Set initial temperature.
7. Set cooling function: The cooling function determines how quickly the temperature decreases over time.
8. Set stopping criteria: The algorithm should stop when either a maximum number of iterations have been reached or when a satisfactory solution has been found.
9. Iterate until stopping criteria are met:

   I. Generate a new solution by randomly swapping two exams in different time slots.

   II. Calculate the cost difference between the new and current solutions.

   III. If the new solution has a lower cost, accept it as the current solution.

   IV. - If the new solution has a higher cost, accept it with probability $e^{-\frac{\text{cost difference}}{\text{temperature}}}$.

   V. Update temperature using cooling function.

   VI. Repeat until stopping criteria are met.

10. Return best solution found during iterations.

Additional links:

1. **Optimization | Simulated annealing:** **https://youtu.be/nzBAjq04dzo**
2. **Solving Sudoku using the simulated annealing algorithm:** **https://youtu.be/FyyVbuLZav8**