بسم اللّه الرحمن الرحیم

☜ تذکر۱: لطفاً یادداشت‌برداری کنید. آنچه در کلاس مطرح می‌شود ممکن است فراتر از محتوای اسلایدها باشد.
☜ تذکر ۲: لطفاً در اسرع وقت ایمیل خود را در سامانهٔ درس (lms.iut.ac.ir) وارد کنید.
☜ تذکر ۳: اگر احیاناً در ترجمه یا فهم اسلایدها مشکل داشتید، لطفاً در ساعات رفع اشکال به دفتر بنده مراجعه کنید. (ساعات رفع اشکال به‌زودی تعیین خواهند شد انشاءالله.)
☜ تذکر ۴: اسلایدها و دیگر محتواهای مرتبط با درس روی سامانهٔ درس (lms.iut.ac.ir) آپلود می‌شوند.

شبه‌کد الگوریتم SA

procedure simulated annealing
begin
  $t \leftarrow 0$
  initialize T
  select a current point $\mathbf{v}_c$ at random
  evaluate $\mathbf{v}_c$
  repeat
    repeat
      select a new point $\mathbf{v}_n$
        in the neighborhood of $\mathbf{v}_c$
      if $eval(\mathbf{v}_c) < eval(\mathbf{v}_n)$
        then $\mathbf{v}_c \leftarrow \mathbf{v}_n$
        else if $random[0,1) < e^{\frac{eval(\mathbf{v}_n) - eval(\mathbf{v}_c)}{T}}$
          then $\mathbf{v}_c \leftarrow \mathbf{v}_n$
    until (termination-condition)
    $T \leftarrow g(T,t)$
    $t \leftarrow t+1$
  until (halting-criterion)
end

منطق حاکم بر رفتار الگوریتم SA

Instead of picking the best move, the SA algorithm picks a random move. If the move improves the situation, it is always accepted. Otherwise, the algorithm accepts the move with some probability less than 1.

* The probability decreases exponentially with the "badness" of the move—the amount ΔE by which the evaluation is worsened.

* The probability also decreases as the "temperature" T goes down: "bad" moves are more likely to be allowed at the start when T is high, and they become more unlikely as T decreases. If the schedule lowers T slowly enough, the algorithm will find a global optimum with probability approaching 1.

منطق حاکم بر رفتار الگوریتم SA

To control whether worsening steps are accepted, there is a positive real-valued temperature T. Suppose A is the current total assignment. Suppose that $h(A)$ is the evaluation of assignment A to be maximized. Simulated annealing selects a possible successor at random, which gives a new assignment A'. If $h(A') \geq h(A)$, it accepts the assignment and A' becomes the new assignment. Otherwise, the new assignment is accepted randomly, using a Gibbs distribution or Boltzmann distribution, with probability $e^{(h(A')-h(A))/T}$.

منطق حاکم بر رفتار الگوریتم SA

This is only used when $h(A') - h(A) < 0$, and so the exponent is always negative. If $h(A')$ is close to $h(A)$, the assignment is more likely to be accepted. If the temperature is high, the exponent will be close to zero, and so the probability will be close to 1. As the temperature approaches zero, the exponent approaches $-\infty$, and the probability approaches zero.

برای درک بهتر رفتار الگوریتم SA جدول زیر را ببینید:

Temperature	Probability of acceptance			
	1-worse	2-worse	3-worse	9-worse
100	.9900	.9802	.9704	.9139
10	.9048	.8187	.7408	.4066
1	.3679	.1353	.4979e-1	.1234e-3
0.25	.1832e-1	.3355e-3	.6144e-5	.2320e-15
0.1	.4540e-4	.2061e-8	.9358e-13	.8194e-39

The above table shows the probability of accepting worsening steps at different temperatures. In this table, k-worse means that $h(A) - h(A') = k$.

دربارهٔ تأثیر دما بر رفتار الگوریتم

If the temperature is high, as in the $T = 10$ case, the algorithm tends to accept steps that only worsen a small amount; it does not tend to accept very large worsening steps. There is a slight preference for improving steps. As the temperature is reduced (e.g., when $T = 1$), worsening steps, although still possible, become much less likely. When the temperature is low (e.g., $T = 0.1$), it is very rare that it selects a worsening step.

دما با چه منطقی باید کاهش یابد؟

Simulated annealing requires an annealing schedule, which specifies how the temperature is reduced as the search progresses. Geometric cooling is one of the most widely used schedules. An example of a geometric cooling schedule is to start with a temperature of 10 and multiply by 0.99 after each step; this will have a temperature of 0.07 after 500 steps. Finding a good annealing schedule is an art, and depends on the problem.

Tabu Search (TS)

A fundamentally different approach for escaping from local minima is to use aspects of the search history rather than random or probabilistic techniques for accepting worsening search steps. Tabu Search (TS) is a general SLS method that systematically utilises memory for guiding the search process.

In Simple Tabu Search, an iterative improvement strategy is enhanced with a short-term memory that allows it to escape from local minima. This memory is used to prevent the search from returning the most recently visited search positions for a fixed number of search steps.

Tabu Search (TS)

The local search has no memory. It does not remember anything about the search as it proceeds. A simple way to use memory to improve a local search is use **tabu search** that prevents recently changed variable assignments from being changed again. The idea is, when selecting a variable to change, not to select a variable that was changed in the last tt steps for some integer tt, called the **tabu tenure**. Tabu search prevents cycling among a few assignments. The tabu tenure is one of the parameters that can be optimized. A tabu list of size 1 is equivalent to not allowing the same assignment to be immediately revisited.

مثلاً برای مسئلهٔ SAT

After a variable x has been flipped, it cannot be flipped back within the next tt steps, where the tabu tenure, tt, is a parameter of the algorithm. In each search step, the variable to be flipped is selected based on some specific criteria, except that the choice is restricted to variables that are currently not tabu.

مقدار متغیر tt بزرگ باشد؟ کوچک باشد؟

* Intuitively, for low tt, the algorithm may not be able to escape from extensive local optima regions, while for high tt settings, all the routes to a solution may be cut off, because too many variables are tabu.

* The basic role of the tabu list is to prevent cycling. A list that is too short may not prevent cycling, but a list that is too long may create excessive restrictions. It is usually easy to determine the order of magnitude of the list size. But, for a given optimization problem, it is often difficult or even impossible to find a value that prevents cycling and does not excessively restrict the search for all instances of a given size. An effective way of circumventing this difficulty is to vary the size of the tabu list. Each element of the list belongs to it for a number of iterations bounded by given maximum and minimum values.

Population-Based Methods

The preceding local search algorithms maintain a single total assignment. Now we consider algorithms that maintain multiple total assignments.

In these algorithms, a total assignment of a value to each variable is called an individual and the set of current individuals is a population.

Remark: Our candidate solutions are complete (rather than partial) board configurations, which specify the positions of all eight queens.

Local beam search

It begins with k randomly generated states. At each step, all the successors of all k states are generated. If any one is a goal, the algorithm halts. Otherwise, it selects the k best successors from the complete list and repeats.

Local Beam search is a method similar to iterative best improvement, but it maintains up to k assignments instead of just one. It reports success when it finds a satisfying assignment. At each stage of the algorithm, it selects the k best neighbors of the current individuals (or all of them if there are less than k) and picks randomly in the case of ties. It repeats with this new set of k total assignments.

At first sight, a local beam search with k states might seem to be nothing more than running k random restarts in parallel instead of in sequence. In fact, the two algorithms are quite different. In a random-restart search, each search process runs independently of the others. In a local beam search, useful information is passed among the parallel search threads. In effect, the states that generate the best successors say to the others, "Come over here, the grass is greener!" The algorithm quickly abandons unfruitful searches and moves its resources to where the most progress is being made.

Stochastic beam search

In its simplest form, local beam search can suffer from a lack of diversity among the k states—they can quickly become concentrated in a small region of the state space, making the search little more than an expensive version of hill climbing. A variant called stochastic beam search, analogous to stochastic hill climbing, helps alleviate this problem. Instead of choosing the best k from the the pool of candidate successors, stochastic beam search chooses k successors at random, with the probability of choosing a given successor being an increasing function of its value.

Stochastic beam search is an alternative to local beam search, which, instead of choosing the best k individuals, selects k of the individuals at random; the individuals with a better evaluation are more likely to be chosen.