بسم اللّه الرّحمن الرّحیم

تذکر: لطفاً یادداشت‌برداری کنید. آنچه در کلاس مطرح می‌شود ممکن است فراتر از محتوای اسلایدها باشد.

Search, Games, and Problem Solving (Chapters 3–6)

❋ Uninformed Search Strategies (Chapter 3)

❋ Informed (Heuristic) Search Strategies (Chapter 3)

❋ Stochastic Local Search (SLS), Evolutionary Algorithms, Meta-heuristics, Swarm Intelligence (Chapter 4)

❋ Adversarial Search, Games, Multiagent Systems, Mathematical Game Theory (Chapter 5)

❋ Constraint Satisfaction Problems (Chapter 6)

Local Search Algorithms and Optimization Problems

Algorithms that perform purely local search in the state space, evaluating and modifying one or more current states rather than systematically exploring paths from an initial state.

The family of local search algorithms includes methods inspired by statistical physics (Simulated Annealing) and evolutionary biology (Genetic Algorithms).

Bio-inspired Algorithms, Nature-Inspired Optimization Algorithms, Nature-inspired Metaheuristic Algorithms, Biologically Inspired Algorithms

مزایای رویکرد جستجوی محلی نسبت به روش‌های جستجوی سیستماتیک

Although local search algorithms are not systematic, they have two key advantages: (1) they use very little memory—usually a constant amount; and (2) they can often find reasonable solutions in large or infinite (continuous) state spaces for which systematic algorithms are unsuitable.

Systematic vs Local Search

* Many SLS methods are surprisingly simple, and the respective algorithms are rather easy to understand, communicate and implement.

* These algorithms can often solve computationally hard problems very effectively and robustly.

* SLS methods are also typically quite general and flexible.

* SLS methods are among the most prominent and successful techniques for solving computationally hard problems in many areas of computer science (specifically artificial intelligence) and operations research.

الگوریتم‌های جستجوی سیستماتیک

Systematic search algorithms traverse the search space of a problem instance in a systematic manner which guarantees that eventually either a (optimal) solution is found, or, if no solution exists, this fact is determined with certainty. This typical property of algorithms based on systematic search is called completeness.

Completeness: Is the algorithm guaranteed to find a solution when there is one?

Local search algorithms start at some location of the given search space and subsequently move from the present location to a neighbouring location in the search space, where each location has only a relatively small number of neighbours, and each of the moves is determined by a decision based on local knowledge only. Typically, local search algorithms are incomplete, that is, there is no guarantee that an existing solution is eventually found, and the fact that no solution exists can never be determined with certainty.

Unfortunately, many spaces are too big for systematic search and are possibly even infinite. In any reasonable time, systematic search will have failed to consider enough of the space to give any meaningful results. We consider methods intended to work in these very large spaces. The methods do not systematically search the whole search space but they are designed to find solutions quickly on average. They do not guarantee that a solution will be found even if one exists, and so they are not able to prove that no solution exists. They are often the method of choice for applications where solutions are known to exist or are very likely to exist.

Not All Combinatorial Problems are Hard

Although many combinatorial problems are NP-hard, it should be noted that not every computational task that can be formulated as a combinatorial problem is inherently difficult. In general, there are many other combinatorial problems that can be solved by polynomial-time algorithms.

تمرین: مشخص کنید که کدام‌یک از مسائل زیر NP-hard هستند و کدام‌یک چنین نیستند؟

- ☞ The minimum-spanning-tree problem
- ☞ The hamiltonian-cycle problem
- ☞ The linear programming (LP) problem
- ☞ The clique problem
- ☞ The vertex-cover problem
- ☞ The traveling-salesman problem (TSP)
- ☞ The subset-sum problem
- ☞ The independent-set problem
- ☞ The all-pairs shortest-paths problem
- ☞ The maximum-flow problem
- ☞ The CNF-satisfiability problem
- ☞ The optimal binary search tree problem
- ☞ The 0-1 knapsack problem
- ☞ The fractional knapsack problem

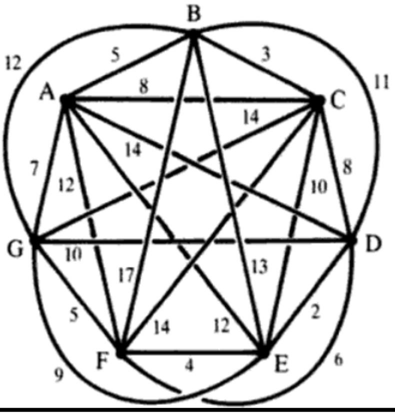
یادآوری: یک نمونهٔ ساده از مسئلهٔ CNF-SAT

$$\begin{aligned} F := {} & (\neg x_1 \vee x_2) \\ & \wedge (\neg x_2 \vee x_1) \\ & \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \\ & \wedge (x_1 \vee x_2) \\ & \wedge (\neg x_4 \vee x_3) \\ & \wedge (\neg x_5 \vee x_3) \end{aligned}$$

تمرین: یک مقداردهی ارضاکننده برای نمونهٔ بالا ارائه دهید.

یادآوری: مسئلهٔ فروشندهٔ دوره‌گرد

The Travelling Salesman Problem (TSP): Finding a minimum-weight hamiltonian cycle in a weighted complete graph.



الگوریتم‌های مبتنی بر راهبرد جستجوی محلی تصادفی را می‌توان به‌شکل زیر به دو دسته تقسیم کرد:

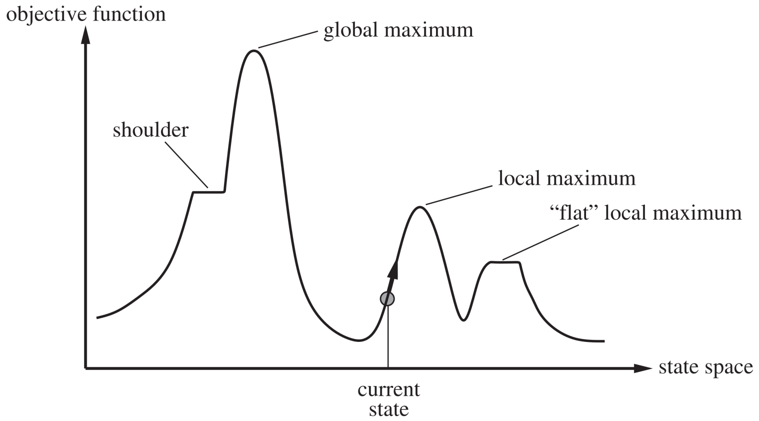
☞ **Single-solution-based SLS methods (trajectory): Manipulate only one single candidate solution (individual) of the given problem instance in each search step.**

☞ **Population-based SLS methods: Several individual candidate solutions are simultaneously maintained.**

Single-Solution-Based Metaheuristics

- Hill Climbing (HC)
- Simulated Annealing (SA)
- Tabu Search (TS)
- Iterated Local Search (ILS)
- Variable Neighbourhood Search (VNS)
- Greedy Randomized Adaptive Search Procedure (GRASP)
- Guided Local Search (GLS)

State-Space Landscape



A landscape has both “location” (defined by the state) and “elevation” (defined by the value of the heuristic cost function or objective function). Local search algorithms explore this landscape.

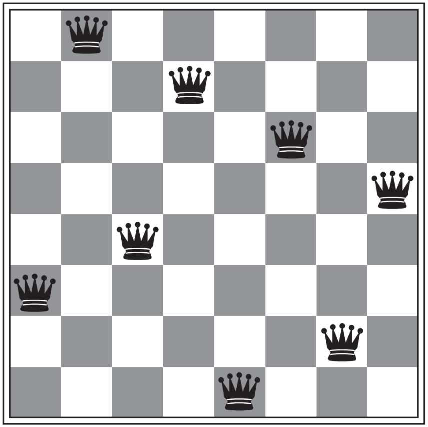
The hill-climbing search algorithm (steepest-ascent version)—iterative best improvement

Hill-climbing search modifies the current state to try to improve it, as shown by the arrow. At each step the current node is replaced by the best neighbor.

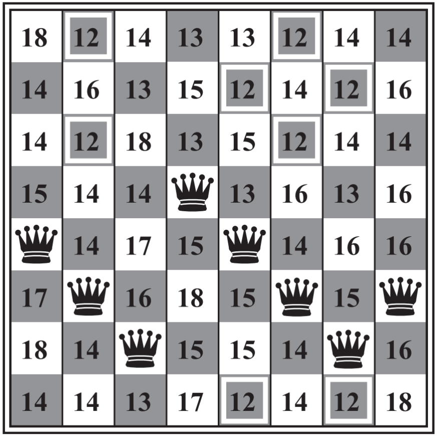
It is simply a loop that continually moves in the direction of increasing value—that is, uphill. It terminates when it reaches a "peak" where no neighbor has a higher value. The algorithm does not maintain a search tree, so the data structure for the current node need only record the state and the value of the objective function. Hill climbing does not look ahead beyond the immediate neighbors of the current state.

```
**function** HILL-CLIMBING($problem$) **returns** a state that is a local maximum
    $current \leftarrow$ MAKE-NODE($problem$.INITIAL-STATE)
    **loop do**
        $neighbor \leftarrow$ a highest-valued successor of $current$
        **if** neighbor.VALUE $\leq$ current.VALUE **then return** $current$.STATE
        $current \leftarrow neighbor$
```

8-Queens Problem



Local search algorithms typically use a complete-state formulation, where each state has 8 queens on the board, one per column.



18	12	14	13	13	12	14	14
14	16	13	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14	♛	13	16	13	16
♛	14	17	15	♛	14	16	16
17	♛	16	18	15	♛	15	♛
18	14	♛	15	15	14	♛	16
14	14	13	17	12	14	12	18

* The successors of a state are all possible states generated by moving a single queen to another square in the same column (so each state has $8 \times 7 = 56$ successors).

* The heuristic cost function h is the number of pairs of queens that are attacking each other, either directly or indirectly. The global minimum of this function is zero, which occurs only at perfect solutions.

* The figure shows a state with $h = 17$. The figure also shows the values of all its successors, with the best successors having $h = 12$.

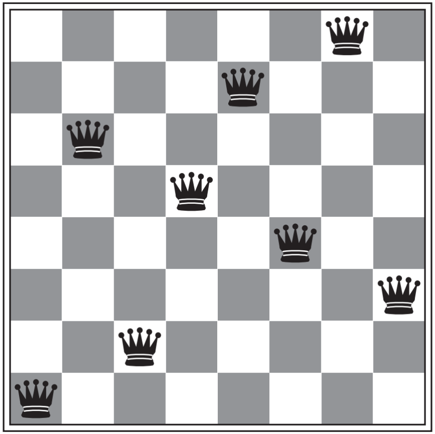
* Hill-climbing algorithms typically choose randomly among the set of best successors if there is more than one.

* Hill climbing often makes rapid progress toward a solution.

Unfortunately, hill climbing often gets stuck for the following reasons:

(1) Local maxima: a local maximum is a peak that is higher than each of its neighboring states but lower than the global maximum. Hill-climbing algorithms that reach the vicinity of a local maximum will be drawn (attracted) upward toward the peak but will then be stuck with nowhere else to go. ☞ Basin of Attraction

A local minimum in the 8-queens state space; the state has $h = 1$ but every successor has a higher cost. Every move of a single queen makes the situation worse.



(2) Plateaux: A plateau is a flat area of the state-space landscape. It can be a flat local maximum, from which no uphill exit exists, or a shoulder, from which progress is possible. A hill-climbing search might get lost on the plateau.

(3) Ridges: Ridges result in a sequence of local maxima that is very difficult for greedy algorithms to navigate.

Why ridges cause difficulties for hill climbing? **The grid of states (dark circles) is superimposed on a ridge rising from left to right, creating a sequence of local maxima that are not directly connected to each other. From each local maximum, all the available actions point downhill.**