# CSE 6140/ CX 4140
# Computational Science and Engineering
# ALGORITHMS

## **Coping with NP-completeness - 4**
## Local Search

Instructor: Xiuwei Zhang
Assistant Professor
School of Computational Science and Engineering

Based on slides by Prof. Ümit V. Çatalyürek and Bistra Dilkina

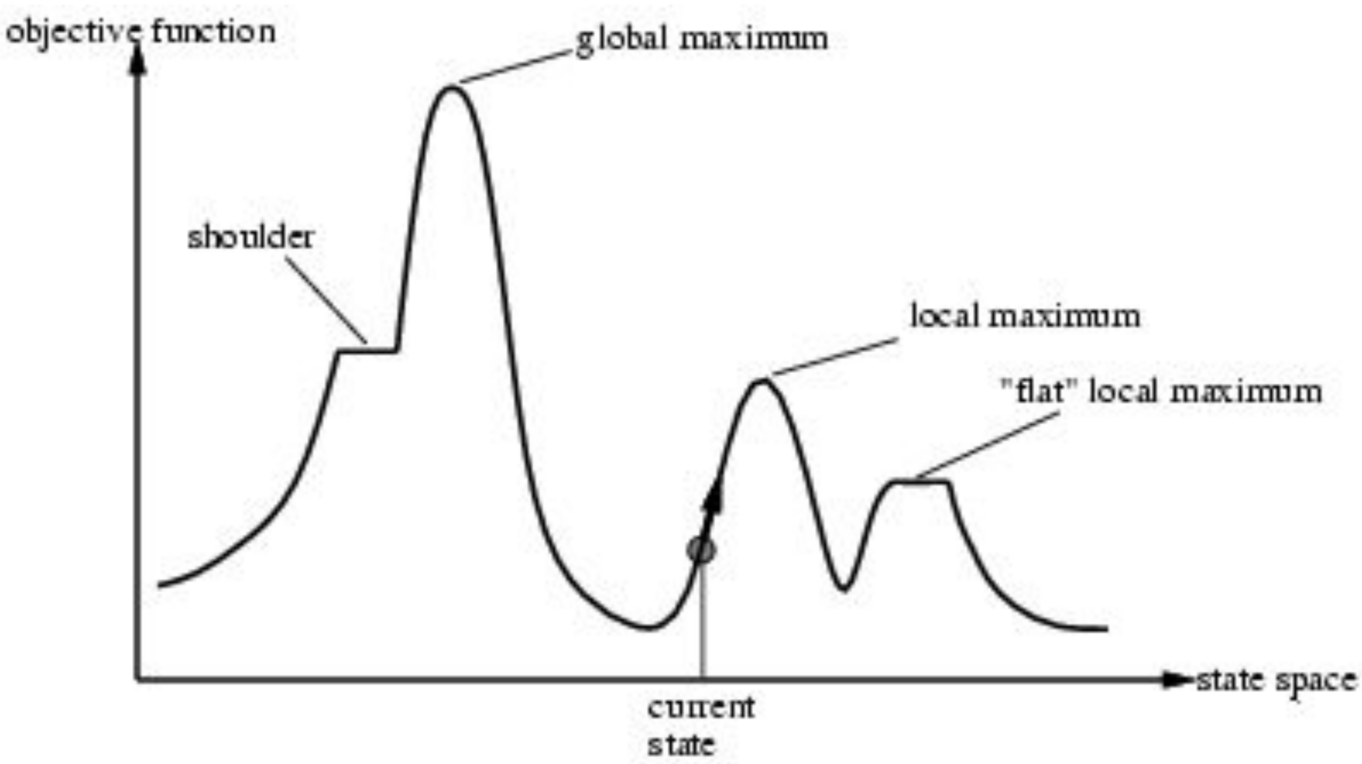
# HEURISTICS/LOCAL SEARCH

# Local Search (LS)

- Start from initial position
- Iteratively move from current position to one of neighboring positions
- Use evaluation function to choose among neighboring positions

# State space landscape

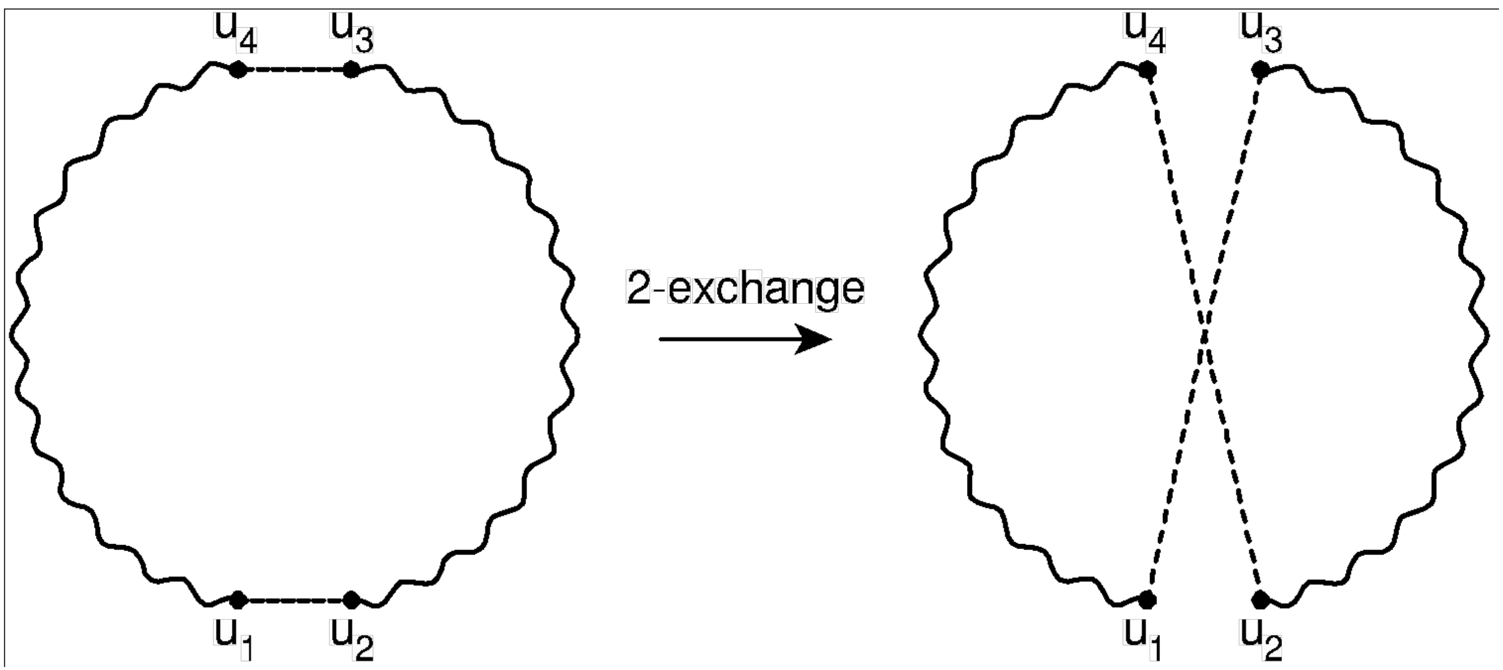
- Objective function defines *state space landscape*



# Local Search (LS) Algorithms

- search space S
  - TSP: set of all permutations of vertices (all "potential solutions")
- solution set $S' \subseteq S$
  - TSP: the tours of minimum length
- neighborhood relation $N \subseteq SxS$
  - A way to move from one potential solution to another
  - TSP: neighboring tour differ in several edges
- evaluation function g : S $\longrightarrow$ R+
  - TSP: length of tour

# Symm. TSP --- search neighborhood



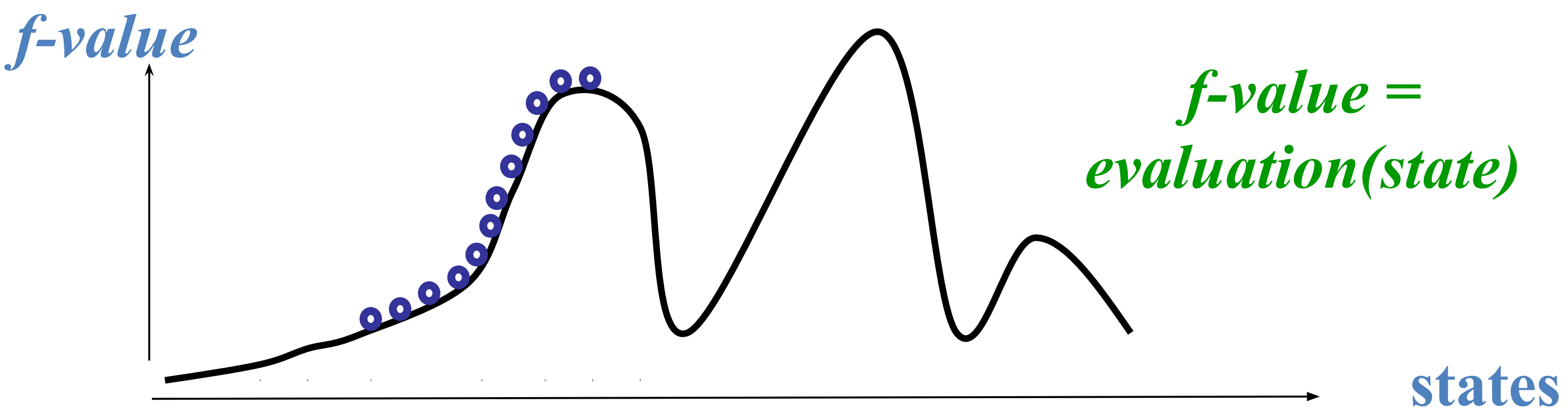

Search Space: all permutations of the cities (each defines a cycle)
3-opt – delete 3 edges and reconnect fragments into 1 cycle
k-opt – delete k edges and reconnect fragments into 1 cycle

# Iterative Improvement (Greedy Search)

- Initialize search at some point of search space
- In each step, move from the current search position to a neighboring position with better evaluation function value

# Hill climbing (Best Improvement Search)

- Choose the neighbor with the largest improvement as the next state




**while** f-value(*state*) < f-value(next-best(*state*))
    *state* := next-best(*state*)

# Hill climbing

```
function Hill-Climbing(problem) returns a solution state

current ← Make-Node(Initial-State[problem])
loop do
    next ← a highest-valued successor of current
    if  Value[next] < Value[current] then return current
    current ←next
end
```

# Problems with iterative improvement

- Advantages:
  - Very fast, works well for certain problems
- Disadvantages:
  - What if there are multiple peaks?
    - Hill climbing gets stuck at all peaks, known as local maxima
    - Optimal solution is highest peak – global maximum
    - May result in extremely suboptimal solution if many peaks
  - Being misguided by evaluation/objective function

# Stochastic Local Search

- Randomize <u>initialization</u> step
- Randomize search steps such that <u>suboptimal/worsening steps</u> are allowed

- Improved performance & robustness
- Typically, degree of randomization controlled by noise parameter

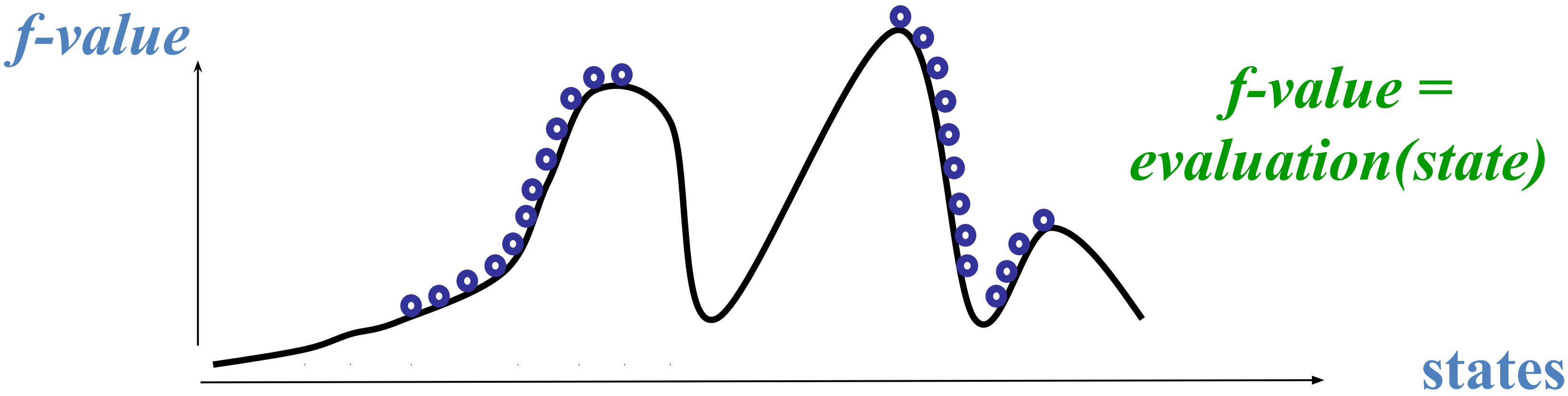
# Stochastic Local Search

- Pros:
  - for many combinatorial problems more efficient than systematic search
  - easy to implement
  - easy to parallelize

- Cons:
  - often incomplete (no guarantees for finding existing solutions)
  - highly stochastic behavior
  - often difficult to analyze theoretically/empirically

# Random restart hill climbing

- Start at random solution
- Hill-climb until local optima
- Start at another random position




***f-value = evaluation(state)***

# Randomized Iterative Improvement:

- *Idea*: escape local maxima by allowing some "bad" moves

- *Initialize* search at some point of search space

- *Search steps*:
  - With probability *p*, move from current search position to a randomly selected neighboring position
  - Otherwise, move from current search position to neighboring position with better evaluation function value.

# The WalkSAT Algorithm

```
WalkSAT(CNF, max-tries, max-flips, p) {
   for i ← 1 to max-tries do
      solution = random truth assignment
      for j ← 1 to max-flips do
         if all clauses in CNF satisfied then
            return solution
         c ← random unsatisfied clause in CNF
         with probability p
            flip a random variable in c
         else
            flip variable in c that maximizes
               number of satisfied clauses
  return failure
}
```

# Simulated annealing (SA)

- Combinatorial search technique inspired by the physical process of annealing [Kirkpatrick et al. 1983, Cerny 1985]

- Outline
  - Select a neighbor at random.
  - If better than current state, go there (improving move).
  - Otherwise, go there with some probability (worsening move).
  - Probability goes down with time (similar to temperature cooling)

- When probability is high -> diversify (many worsening moves)
- When probability is low -> intensify (focus on improving moves)

# SA analogy

- Annealing is process of heating and cooling metals in order to improve strength
- Idea: Controlled heating and cooling of metal
  - When hot, atoms move around
  - When cooled, atoms find configuration with lower internal energy (i.e. makes metal stronger)
- Analogy:
  - Temperature = probability of accepting worse neighboring solution
    - When temperature is high, likely to accept worse neighboring solutions (but may lead to better overall solution)
  - Cooling represents shrinking probability of accepting worse solutions

# SA Pseudo code

**function** Simulated-Annealing(*problem, schedule*) **returns** *solution state*

*current* ← Make-Node(Initial-State[*problem*])

**for t ← 1 to *infinity (Iters, Time cutoff)***

$T \leftarrow schedule[t]$ // T (a function of time t) goes downwards.

**if** $T = 0$ **then return** *greedy from current*

*next* ← Random-Successor*(current)*

$\Delta E \leftarrow$ f-Value[*next*] - f-Value[*current*]

**if** $\Delta E > 0$ then *current* ← *next*

**else** *current* ←*next* with probability $e^{\Delta E/T}$

**end**

# Simulated annealing (SA)

- Acceptance criterion (Metropolis condition): choose new solution s' over old solution s with probability (maximization)

$$\Pr(s', s) = \begin{cases} 1 & \text{if } f(s') > f(s) \\ \exp\left\{ \dfrac{f(s') - f(s)}{T} \right\} & \text{otherwise} \end{cases}$$

- Initial temperature $T_0$
- Annealing (cooling) schedule: how to update the temperature
  - E.g., T = a T  with a = 0.95 (geometric schedule)
  - Number of iterations, neighbourhood size
- Stopping criterion
  - E.g., no improved solution found for a number of iterations (or number of temperature values)

# SA for TSP [Johnson & McGeoch 1997]

- Baseline implementation:
    - start with random initial solution
    - use 2-exchange neighborhood
    - simple annealing schedule;
- -> relatively poor performance

- Improvements:
    - look-up table for acceptance probabilities
    - neighborhood pruning
    - low-temperature starts

# SA with restarts

- RESTARTS:
  Sometimes it is better to move back to a solution that was significantly better rather than always moving from the current state.
- The decision to restart could be based on several criteria.
  - based on a fixed number of steps
  - the current solution is much worse than the best so far
  - too many iterations without improvement
  - restarting randomly, etc.

# Summary of Simulated Annealing

- Historically important
- Easy to implement
- Has interesting theoretical properties (convergence), but these are of very limited practical relevance
- Achieves good performance (often at the cost of substantial run-times)

# Tabu Search

- Combinatorial search technique which heavily relies on the use of an explicit memory of the search process to guide search process [Glover 1989, 1990]

- Memory typically contains only specific attributes of previously seen solutions

- Simple tabu search strategies exploit only short term memory

- More complex tabu search strategies exploit long term memory

# Next lecture

- More on Tabu search
- Iterated local search
- Constructing initial solutions

- Approximation algorithms

- Hw4 deadline extended:
  Nov. 18, 11:59pm (instead of Nov. 13)
  hard deadline, no grace period!