



CUGS Advanced Course

Heuristic Algorithms for Combinatorial Optimization Problems

Petru Eles and Zebo Peng
Embedded Systems Laboratory (ESLAB)
Linköping University



INSTITUTE OF TECHNOLOGY
LINKÖPING UNIVERSITY

Objectives

- Introduction to combinatorial optimization problems.
- Basic principles of heuristic techniques.
- Modern heuristic algorithms:
 - Simulated annealing.
 - Tabu search.
 - Genetic algorithms.
- Evaluation of heuristic algorithms.
- Application of heuristic techniques to design automation and software engineering.

Course Organization

- Introductory lectures
 - Lectures in 3 blocks.
 - The future time slots?
 - Mostly on principles and basic algorithms.
- Project part:
 - Implementation of one or two heuristic algorithms.
 - You can select the application area, e.g., related to your current research topic.
 - Documentation of the implementation work in a term paper.
 - Presentation of the results in a common seminar.

Reference Literature

- C. R. Reeves, "Modern Heuristic Techniques for Combinatorial Problems," Blackwell Scientific Publications, 1993.
- Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs" Springer-Verlag, 1992.
- A. H. Gerez, "Algorithms for VLSI Design Automation," John Wiley & Sons, 1999.

Additional Papers

- A. Colomi et al., “Heuristics from Nature for Hard Combinatorial Optimization Problems,” Int. Trans. Operations Research, Vol. 3, No. 1, 1996.
- S. Kirkpatrick et al., “Optimization by Simulated Annealing,” Science, Vol. 220, No. 4598, 1983.
- ... (to be distributed in each lecture block).
- Lecture notes.

Lecture I



- Combinatorial optimization
- Overview of optimization heuristics
 - Neighborhood search
- Evaluation of heuristics

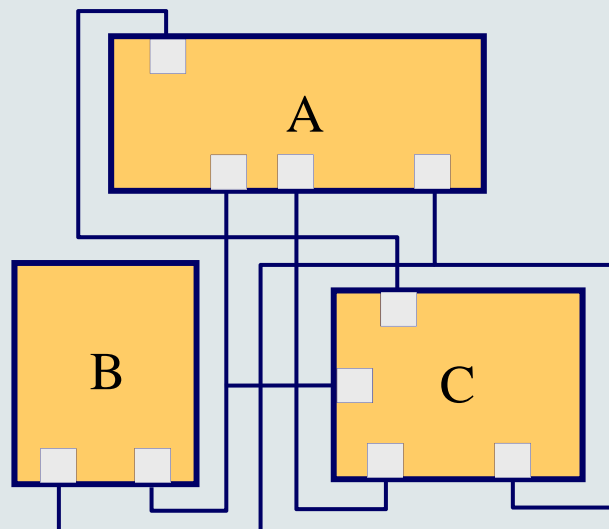
Introduction

- Many computer science problems deal with the choice of a best set of parameters to achieve some goal.

Ex. Placement and routing problem in VLSI design:

- Given a set of VLSI cells, with ports on the boundaries, and a collection of nets, which are sets of ports that need to be wired together
- Find a way to place the cells and run the wires so that the total wiring distance is minimized and each wire is shorter than a given constant.

Placement and Routing



Design Space Exploration

- The majority of design space exploration tasks can be viewed as optimization problems:

To find

- the architecture (type and number of processors, memory modules, and communication blocks, as well as their interconnections),
- the mapping of functionality onto the architecture components, and
- the schedules of basic functions and communications,

such that

- a cost function (in terms of implementation cost, performance, power, etc.) is minimized; and
- a set of constraints are satisfied.

Mathematical Optimization

- The optimization problems can usually be formulated as to

Minimize $f(\mathbf{x})$

Subject to $g_i(\mathbf{x}) \geq b_i; \quad i = 1, 2, \dots, m;$

where

\mathbf{x} is a vector of decision variables;

f is the cost (objective) function;

g_i 's are a set of constraints.

- If f and g_i 's are linear functions, we have a *Linear Programming* problem.
- LP problems can be solved by, e.g., the simplex algorithm, which is an *exact* method, i.e., it will always identify the optimal solution if it exists.

Type of Solutions

- A solution to an optimization problem specifies the values of the decision variables, \mathbf{x} , and therefore also the value of the objective function, $f(\mathbf{x})$.
- A feasible solution satisfies all constraints.
- An optimal solution is feasible and gives the best objective function value.
- A near-optimal solution is feasible and provides a superior objective function value, but not necessarily the best.

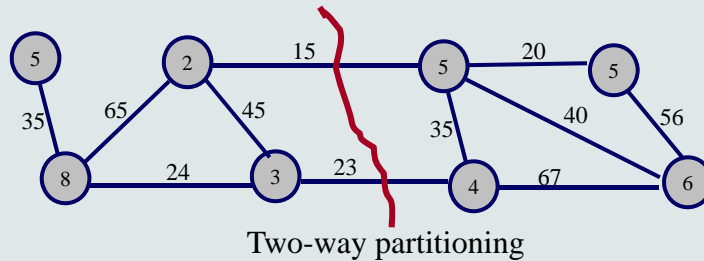
Combinatorial Optimization (CO)

- There are two types of optimization problems:
 - Continuous, with an infinite number of feasible solutions;
 - Combinatorial, with a finite number of feasible solutions.
- In an CO problem, the decision variables are discrete, i.e., where the solution is a set, or a sequence, of integers or other discrete objects.

Ex. System partitioning can be formulated as follows:

Given a graph with costs on its edges, partition the nodes into k subsets no larger than a given maximum size, to minimize the total cost of the cut edges.

The System Partitioning Problem



A feasible solution for the k-way partitioning can be represented as:

$$x_i = j; \quad j \in \{1, 2, \dots, k\}, \quad i = 1, 2, \dots, n.$$

Features of CO Problems

- Most CO problems, e.g., system partitioning with constraints, for digital system designs are NP-complete.
- The time needed to solve an NP-complete problem grows exponentially with respect to the problem size n .
- For example, to enumerate all feasible solutions for a scheduling problem (all possible permutation), we have:
 - 20 tasks in 1 hour (assumption);
 - 21 tasks in 20 hour;
 - 22 tasks in 17.5 days;
 - ...
 - 25 tasks in 6 centuries.

An Exact Approach to CO

- Many CO problems can be formulated as an *Integer Linear Programming* (ILP) problem, and solved by an ILP solver.
- It is inherently more difficult to solve an ILP problem than the corresponding Linear Programming problem.
 - Because there is no derivative information and the surface are not smooth.
- The size of problem that can be solved successfully by ILP algorithms is an order of magnitude smaller than the size of LP problems that can be easily solved.

Simple vs Hard Problems

- | | |
|--|--------------------------------------|
| ● Few decision variables | ● Many decision variables |
| ● Independent variables | ● Dependent variables |
| ● Single objective | ● Multi objectives |
| ● Objective easy to calculate (additive) | ● Objective difficult to calculate |
| ● No or light constraints | ● Severely constraints |
| ● Feasibility easy to determine | ● Feasibility difficult to determine |
| ● Deterministic | ● Stochastic |

Approach to CO

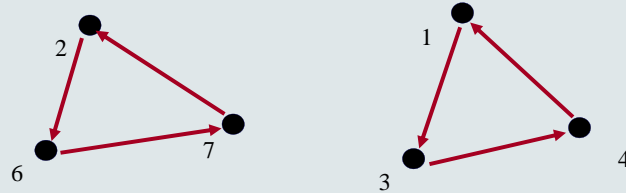
- Why not solve the corresponding LP and round the solutions to the closest integer?
 - Ex. if $x_1 = 2,75$, x_1 will be set of 3.
- This will be plausible if the solution is expected to contain large integers and therefore is insensitive to rounding.
- Otherwise, rounding could be as hard as solving the original problem from scratch, since rounding does not usually even give a feasible solution!

The Rounding-Off Problem

Ex. To maximize $f(x_1, x_2) = 5x_1 + 8x_2$
 subject to $x_1 + x_2 \leq 6$,
 $5x_1 + 9x_2 \leq 45$,
 $x_1, x_2 \geq 0$, and be integers.

	Continuous optimum	Round off	Nearest feasible point	Integer optimum
x_1	2.25	2	2	0
x_2	3.75	4	3	5
f	41.25	Infeasible solution	34	40

Additional TSP Constraints



To avoid disjoint sub-tours, $(2n - 1)$ additional constraints must be added.

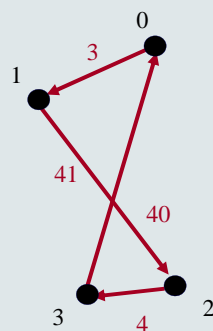
Ex.

$$x_{2,1} + x_{2,3} + x_{2,4} + x_{6,1} + x_{6,3} + x_{6,4} + x_{7,1} + x_{7,3} + x_{7,4} \geq 1.$$

The Branch-and-Bound Approach

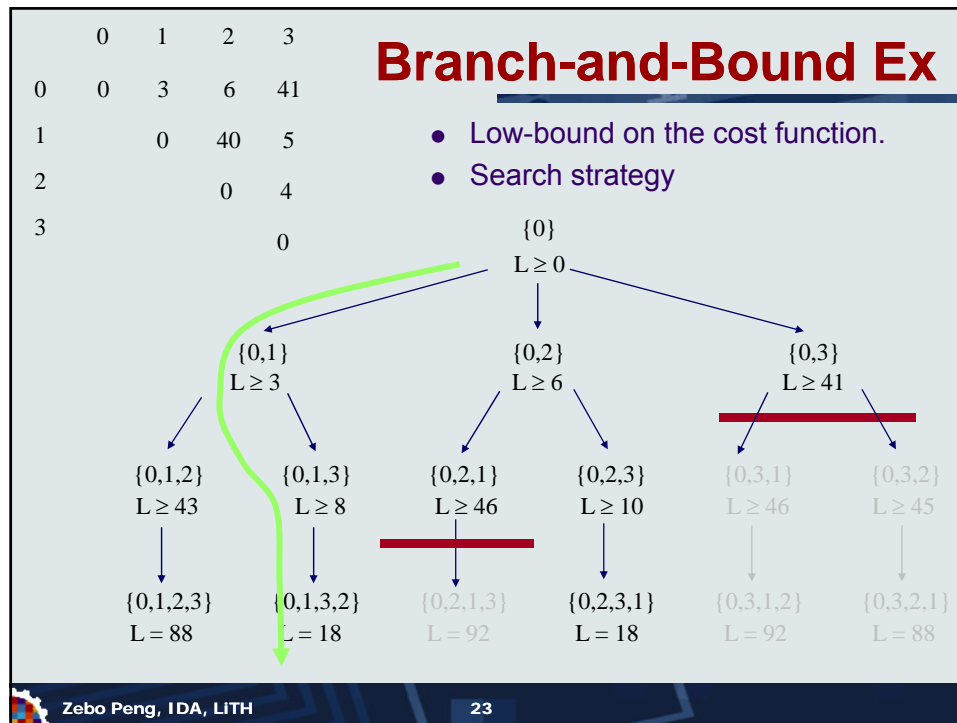
- Traverse an implicit tree to find the best leaf (solution).

4-City TSP



	0	1	2	3
0	0	3	6	41
1		0	40	5
2			0	4
3				0


Total cost of this solution = 88



Some Conclusions

- The integer programming problem is inherently more difficult to solve than the simple linear programming problem.
- We need heuristics, which seek near-optimal solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality.
- Heuristic techniques are widely used to solve many NP-hard problems.

Lecture I



- Combinatorial optimization
- Overview of optimization heuristics
- Neighborhood search
- Evaluation of heuristics

Zebo Peng, IDA, LITH 25

Heuristic: adj. & n.

Webster's 3rd New International Dictionary:
 Greek *heuriskein*: to find.

“providing aid or direction in the solution of a problem but otherwise unjustified or incapable of justification.”

“of or relating to exploratory problem-solving techniques that utilize self-educating techniques to improve performance.”

The Concise Oxford Dictionary:
 “serving to discover; (of computer problem-solving) proceeding by trial and error.”

Zebo Peng, IDA, LITH 26

Why Heuristics

- Many exact algorithms involve a huge amount of computation effort.
- The decision variables have frequently complicated interdependencies.
 - To improve a design, e.g., one might have to simultaneously change the values of several parameters.
- We have often nonlinear cost functions and constraints, even no mathematical functions.
 - Ex. The cost function f can, for example, be defined by a computer program (e.g., for power estimation).
- Approximation of the model for optimization.
 - A near optimal solution is usually good enough and could be even better than the theoretical optimum.



Heuristic Approaches to CO

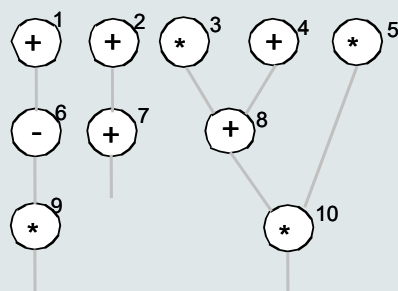
	Problem specific	Generic methods
Constructive	<ul style="list-style-type: none"> • Clustering • List scheduling • Left-edge algorithm 	<ul style="list-style-type: none"> • Branch and bound • Divide and conquer
Transformational (Iterative improvement)	<ul style="list-style-type: none"> • Kernighan-Lin algorithm 	<ul style="list-style-type: none"> • Neighborhood search • Simulated annealing • Tabu search • Genetic algorithms



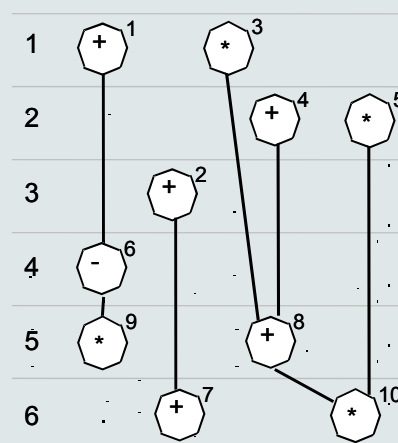
List Scheduling

- Resource-constrained (RC) scheduling problem:
 - To find a (optimal) schedule for a set of operations that obeys the data dependency and utilizes only the available functional units.
- For each control step, the operations that are available to be scheduled are kept in a list.
- The list is ordered by some priority function:
 - The length of path from the operation to the end of the block;
 - Mobility: the number of control steps from the earliest to the latest feasible control step.
- Each operation on the list is scheduled one by one if the resources it needs are free; otherwise it is deferred to the next control step.
- Until the whole schedule is constructed, no information about the schedule length is available.

List Scheduling Example



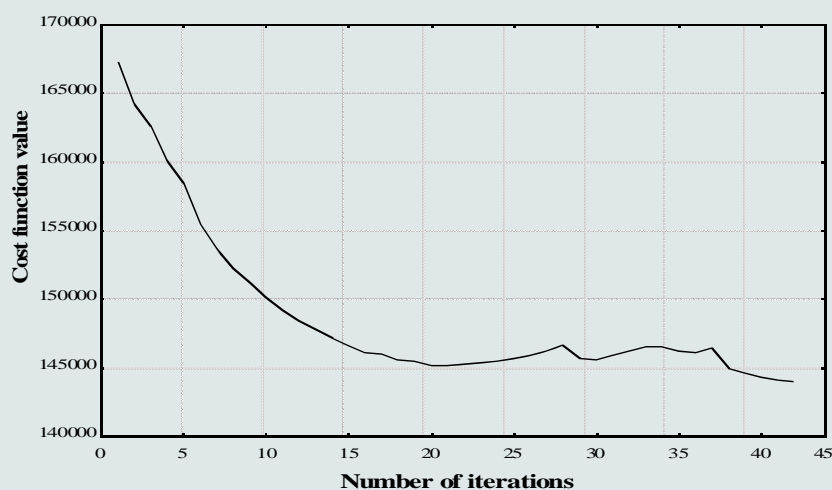
Control Steps



The Kernighan-Lin Algorithm (KL)

- A graph is partitioned into two clusters of arbitrary size, by minimizing a given objective function.
- KL is based on an iterative partitioning strategy:
 - The algorithm starts with two arbitrary clusters C1 and C2.
 - The partitioning is then iteratively improved by moving nodes between the clusters.
 - At each iteration, the node which produces the minimal value of the cost function is moved; this value can, however, be greater than the value before moving the node.

KL Execution Trace Example



0	1	2	3	
0	0	3	6	41
1		0	40	5
2			0	4
3				0

Branch-and-Bound Ex

- Low-bound on the cost function.
- Search strategy

{0} L ≥ 0

├── {0,1} L ≥ 3

│ ├── {0,1,2} L ≥ 43

│ │ └── {0,1,2,3} L = 88

│ └── {0,1,3} L ≥ 8

│ └── {0,1,3,2} L = 18

├── {0,2} L ≥ 6

│ ├── {0,2,1} L ≥ 46

│ │ └── {0,2,1,3} L = 92

│ └── {0,2,3} L ≥ 10

│ └── {0,2,3,1} L = 18

└── {0,3} L ≥ 41

├── {0,3,1} L ≥ 46


│ └── {0,3,1,2} L = 92

└── {0,3,2} L ≥ 45

└── {0,3,2,1} L = 88

Zebo Peng, IDA, LITH 33

Lecture I



- Combinatorial optimization
- Overview of optimization heuristics
 - Neighborhood search
- Evaluation of heuristics

Zebo Peng, IDA, LITH 34

Search as Heuristics

- **Search** — the term used for constructing or improving solutions to obtain the optimum or near-optimum.
- **Solution** — Encoding (representing the solution).
- **Neighborhood** — Nearby solutions (in the encoding or solution space).
- **Move** — Transforming current solution to another (usually neighboring) solution.
- **Evaluation** — To compute the solutions' feasibility and objective function value.
- **Local search** — based on greedy heuristic (local optimizers).



Neighborhood Search Method

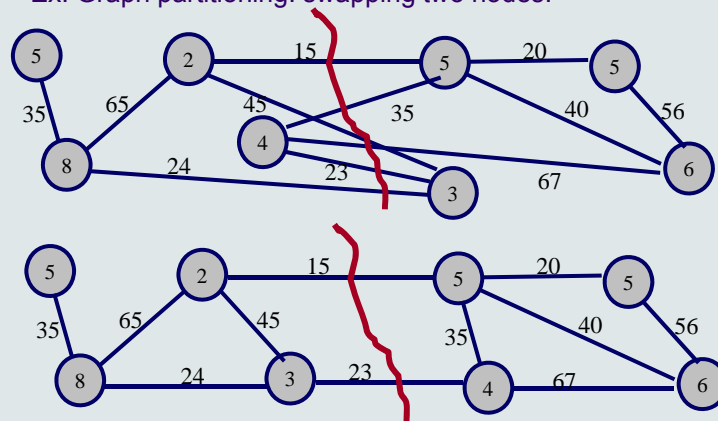
- Step 1 (Initialization)
 - (A) Select a starting solution $\mathbf{x}^{now} \in \mathbf{X}$.
 - (B) $\mathbf{x}^{best} = \mathbf{x}^{now}$, $best_cost = \mathbf{c}(\mathbf{x}^{best})$.
- Step 2 (Choice and termination)
 - Choose a solution $\mathbf{x}^{next} \in \mathbf{N}(\mathbf{x}^{now})$.
 - If no solution can be selected or the terminating criteria apply, then the method stop.
- Step 3 (Update)
 - Re-set $\mathbf{x}^{now} = \mathbf{x}^{next}$.
 - If $\mathbf{c}(\mathbf{x}^{now}) < best_cost$, perform Step 1(B).
 - Goto Step 2.

$\mathbf{N}(\mathbf{x})$ denotes the neighborhood of \mathbf{x} , which is a set of solutions reachable from \mathbf{x} by a simple transformation (move).



Neighborhood Search Method

- The neighborhood search method is very attractive for many CO problems as they have a natural neighborhood structure, which can be easily defined and evaluated.
 - Ex. Graph partitioning: swapping two nodes.

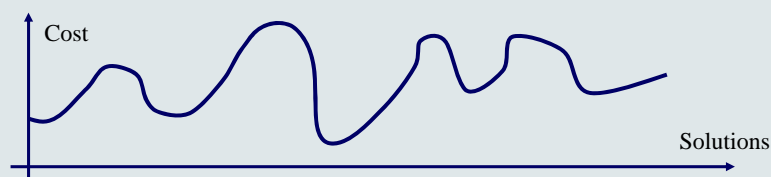


Zebo Peng, IDA, LITH

37

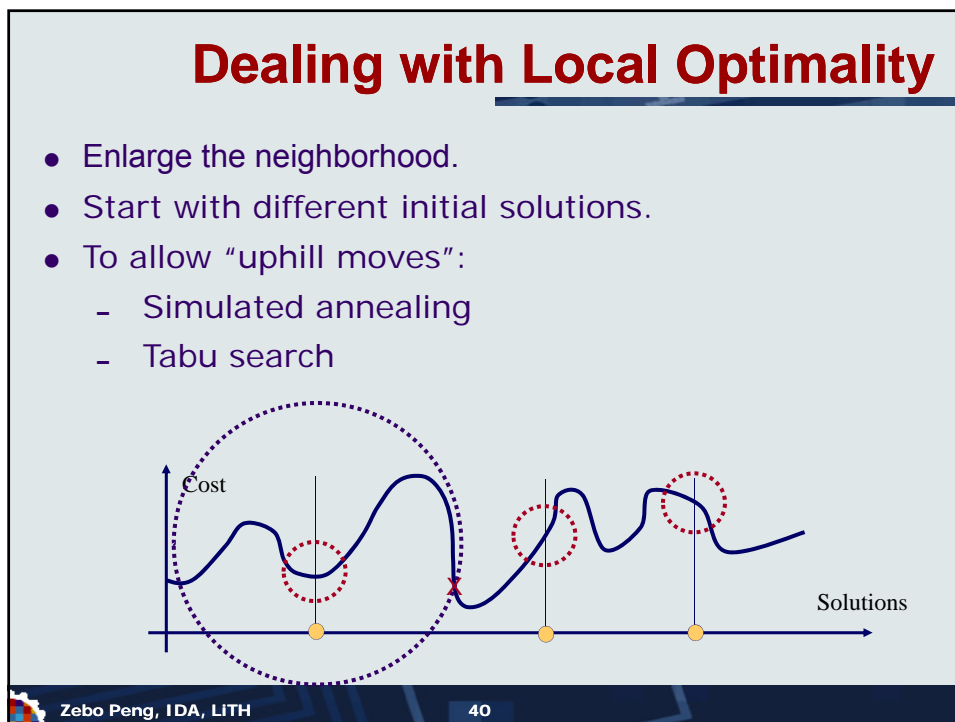
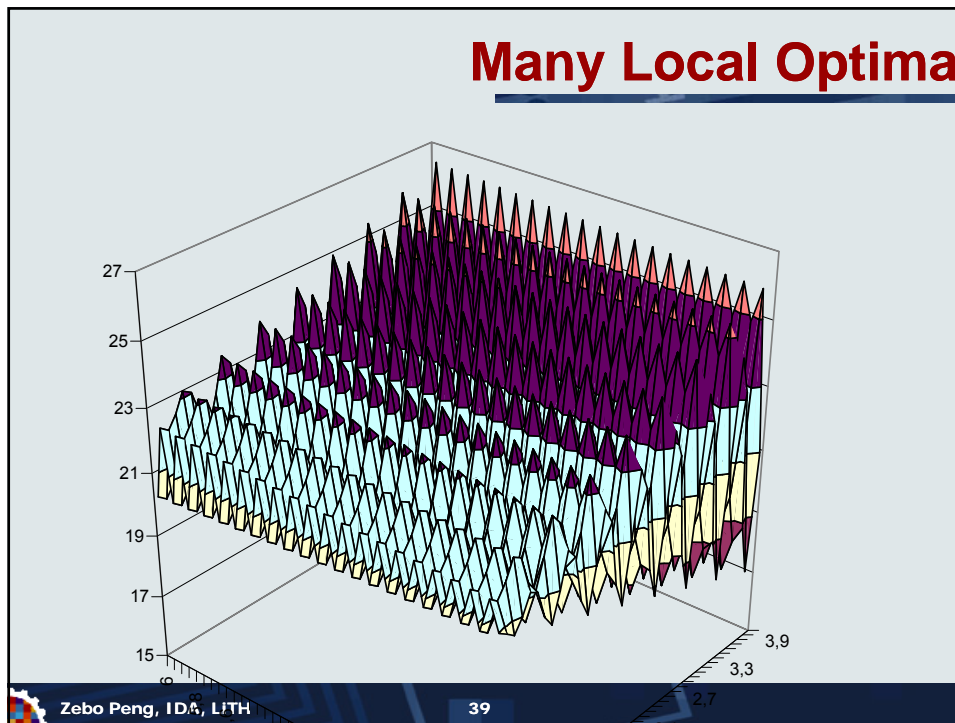
The Descent Method

- Step 1 (Initialization)
- Step 2 (Choice and termination)
 - Choose $\mathbf{x}^{next} \in \mathbf{N}(\mathbf{x}^{now})$ such that $\mathbf{c}(\mathbf{x}^{next}) < \mathbf{c}(\mathbf{x}^{now})$, and terminate if no such \mathbf{x}^{next} can be found.
- Step 3 (Update)
- The descent process can easily be stuck at a local optimum:



Zebo Peng, IDA, LITH

38



Heuristics from Nature

- Model (loosely) a phenomenon existing in nature, derived from physics, biology and social sciences:
 - Annealing.
 - Evolution.
- Usually non-deterministic, including some randomness features.
- Often has implicit parallel structure or enable parallel implementation.

Some Heuristics from Nature

- Genetic algorithms — evolution process of nature.
- Simulated annealing — heat treatment of materials.
- Tabu search — search and intelligent uses of memory.
- Neural nets — mimic biological neurons.
- Ant system — ant colony.
 - Simulating the behavior of a set of agents that cooperate to solve an optimization problem by means of very simple communications.
 - Ants leave trails on paths they visit, which will be followed by other ants, probabilistically.
 - The shorter paths will be enhanced by many ants.

Advantages of Meta-Heuristics

- Very flexible
- Often global optimizers
- Often robust to problem size, problem instance and random variables
- May be the only practical alternative

Lecture I

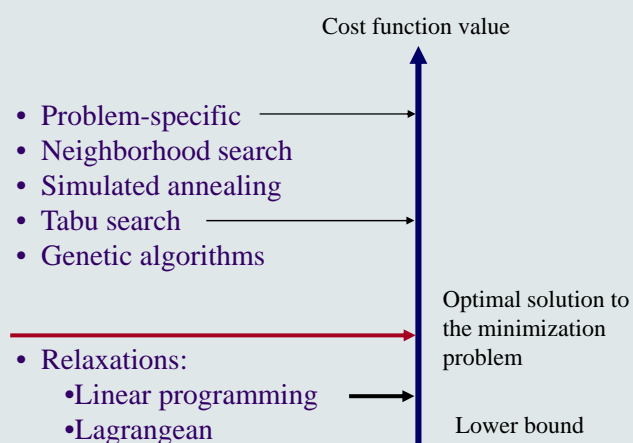


- Combinatorial optimization
- Overview of optimization heuristics
- Neighborhood search
- Evaluation of heuristics

Evaluation of Heuristics

- It is usually very difficult to state how close a heuristic solution is to the optimum.
- One technique is to find a lower bound which is as close as possible to the optimal solution for minimization problem.
- Such lower bound may usually be found by relaxation.

Evaluation Approach



Summary

- Combinatorial optimization is the mathematical study of finding an optimal arrangement, grouping, ordering, or selection of discrete objects usually finite in numbers.

- Lawler, 1976

- In practice, combinatorial problems are often very difficult to solve, because there is no derivative information and the surfaces are not smooth.
- We need therefore to develop heuristic algorithms that seek near-optimal solutions.

