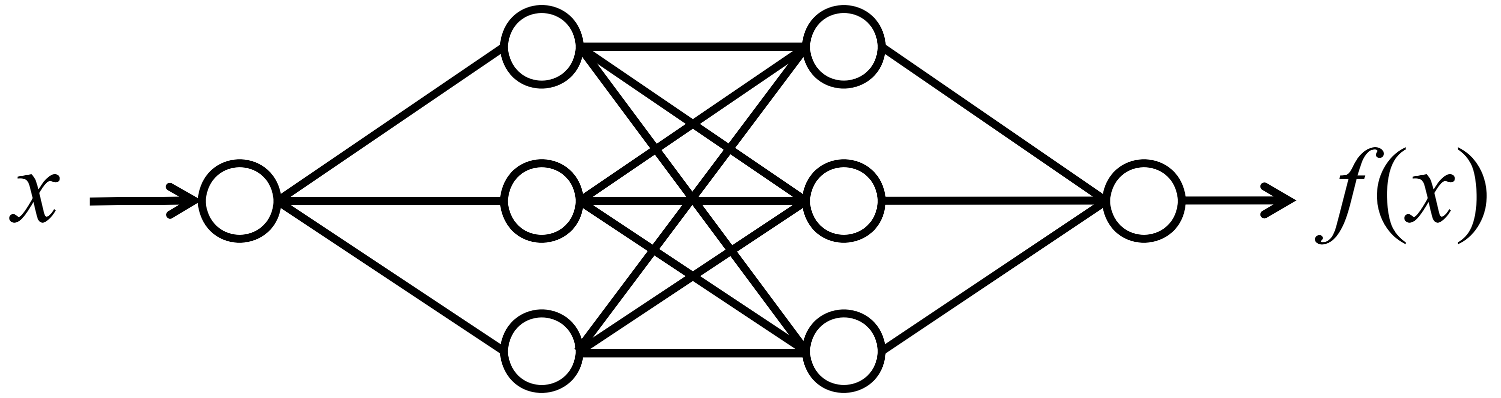
Optimization Methods

Learning and Optimization

Simple Learning Problem

Training Data: $(x_1, y_1), (x_2, y_2), \ . \ . \ . \ . \ , \ (x_m, y_m)$

Testing Data: $(x_{m+1}, y_{m+1}), (x_{m+2}, y_{m+2}), \ . \ . \ . \ . \ , \ (x_{m+n}, y_{m+n})$



Learning: To adjust the connection weights to minimize the following cost function (training data accuracy):

$$z(\mathbf{w}) = [(y_1 - f(x_1))^2 + \ . \ . \ . + (y_m - f(x_m))^2]/m$$

Evaluation: Testing data accuracy:

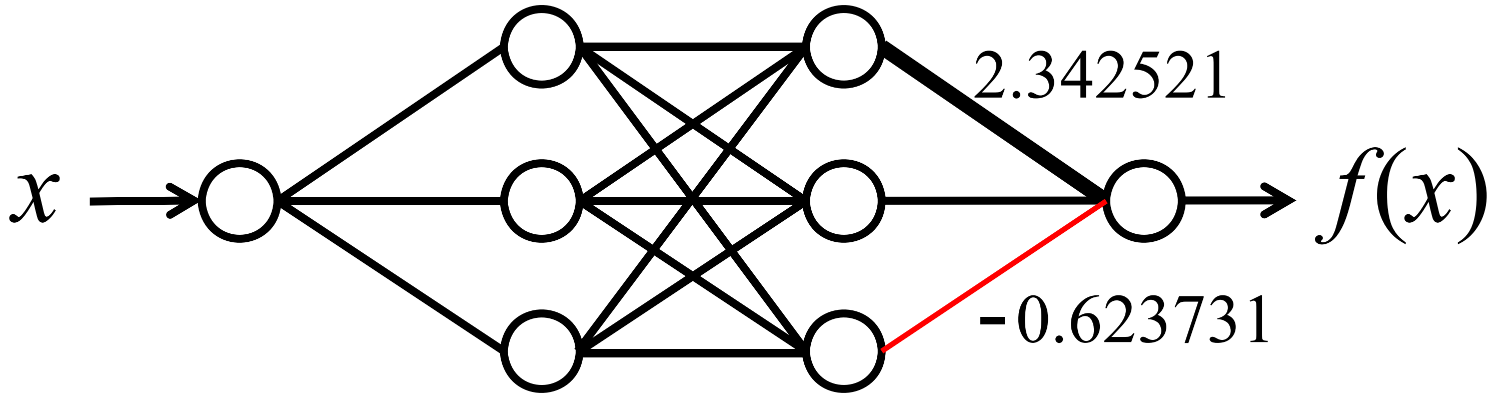
$$z(\mathbf{w}) = [(y_{m+1} - f(x_{m+1}))^2 + \ . \ . \ . \ (y_{m+n} - f(x_{m+n}))^2]/n$$

Learning and Optimization

Simple Learning Problem

Training Data: $(x_1, y_1), (x_2, y_2), \;\ldots,\; (x_m, y_m)$

Testing Data: $(x_{m+1}, y_{m+1}), (x_{m+2}, y_{m+2}), \;\ldots,\; (x_{m+n}, y_{m+n})$



Learning: To adjust the connection weights to minimize the following cost function (training data accuracy):

$$z(\mathbf{w}) = [(y_1 - f(x_1))^2 + \;\ldots + (y_m - f(x_m))^2]/m$$

Evaluation: Testing data accuracy:

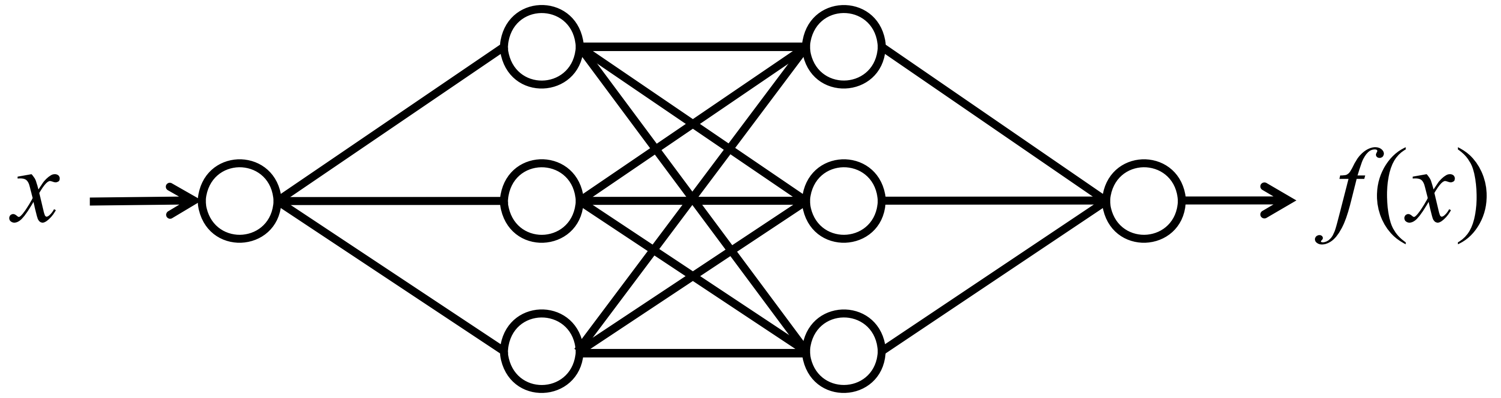
$$z(\mathbf{w}) = [(y_{m+1} - f(x_{m+1}))^2 + \;\ldots\; (y_{m+n} - f(x_{m+n}))^2]/n$$

Learning and Optimization

Optimization Problem

Training Data: $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$

~~Testing Data: $(x_{m+1}, y_{m+1}), (x_{m+2}, y_{m+2}), \ldots, (x_{m+n}, y_{m+n})$~~



Optimization: To adjust the connection weights to minimize the following cost function (training data accuracy):

$$z(\mathbf{w}) = [(y_1 - f(x_1))^2 + \ldots + (y_m - f(x_m))^2]/m$$

Evaluation: Training data accuracy:

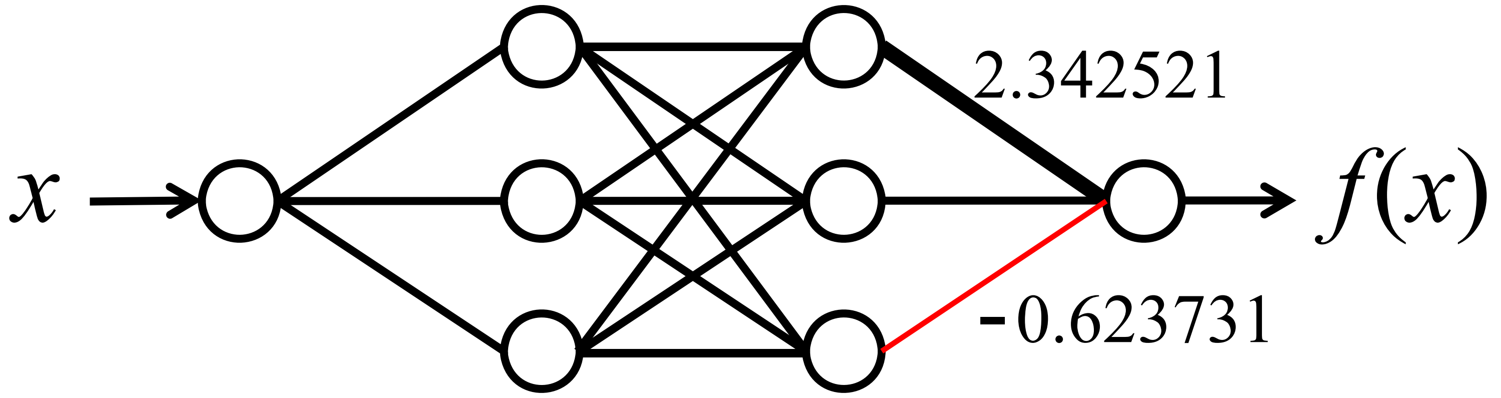
$$z(\mathbf{w}) = [(y_1 - f(x_1))^2 + \ldots (y_m - f(x_m))^2]/m$$

Learning and Optimization

Optimization Problem

Training Data: $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$

~~Testing Data: $(x_{m+1}, y_{m+1}), (x_{m+2}, y_{m+2}), \ldots, (x_{m+n}, y_{m+n})$~~



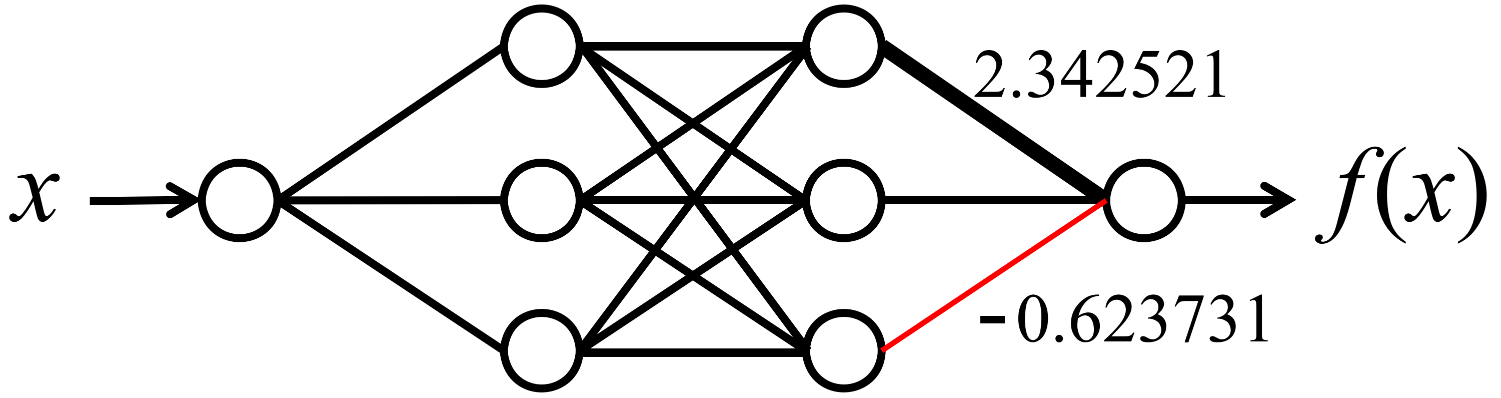
Optimization: To adjust the connection weights to minimize the following cost function (training data accuracy):

$$z(\mathbf{w}) = [(y_1 - f(x_1))^2 + \ldots + (y_m - f(x_m))^2]/m$$

Evaluation: Training data accuracy:

$$z(\mathbf{w}) = [(y_1 - f(x_1))^2 + \ldots\ (y_m - f(x_m))^2]/m$$

Learning and Optimization



Learning: To adjust the connection weights to minimize the following cost function (training data accuracy):

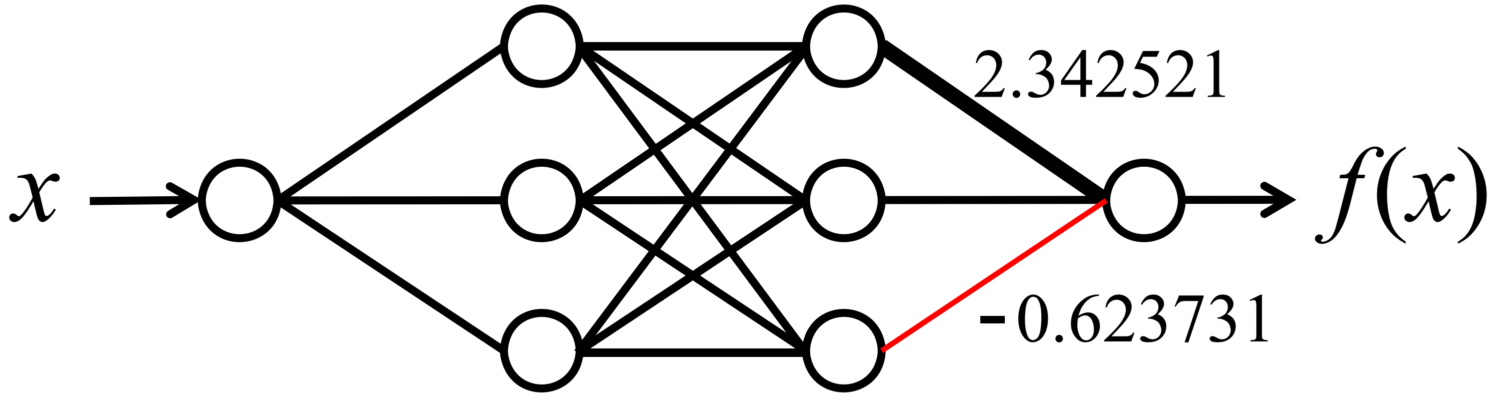
$$z(\mathbf{w}) = [(y_1 - f(x_1))^2 + \ \dots + (y_m - f(x_m))^2]/m$$

Optimization: To adjust the connection weights to minimize the following cost function (training data accuracy):

$$z(\mathbf{w}) = [(y_1 - f(x_1))^2 + \ \dots + (y_m - f(x_m))^2]/m$$

Basically, learning can be viewed as optimization (whereas generalization instead of optimization is emphasized in learning).

Learning and Optimization



Evaluation in learning: Testing data accuracy:

$$z(\mathbf{w}) = [(y_{m+1} - f(x_{m+1}))^2 + \ . \ . \ . \ (y_{m+n} - f(x_{m+n}))^2]/n$$

Evaluation in optimization: Training data accuracy:

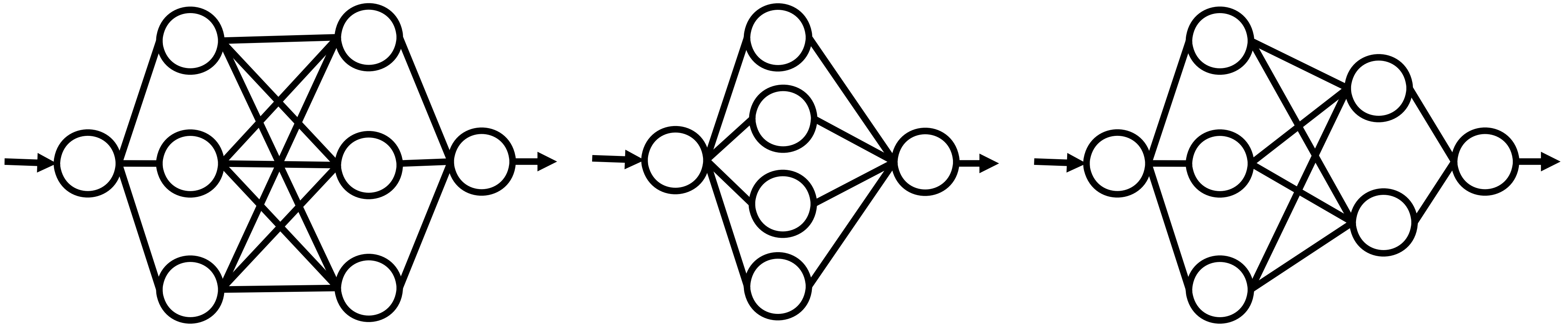
$$z(\mathbf{w}) = [(y_1 - f(x_1))^2 + \ . \ . \ . \ (y_m - f(x_m))^2]/m$$

The final goals are different: Unseen testing data accuracy in learning, and given training data accuracy in optimization.

Neural Architecture Search

Training Data: $(x_1, y_1), (x_2, y_2), \ \ldots, \ (x_m, y_m)$

Testing Data: $(x_{m+1}, y_{m+1}), (x_{m+2}, y_{m+2}), \ \ldots, \ (x_{m+n}, y_{m+n})$



Learning: To find an architecture and connection weights to minimize the following cost function:

$$z(\mathbf{w}) = [(y_1 - f(x_1))^2 + \ \ldots + (y_m - f(x_m))^2]/m$$

Evaluation: Testing data accuracy:

$$z(\mathbf{w}) = [(y_{m+1} - f(x_{m+1}))^2 + \ \ldots \ (y_{m+n} - f(x_{m+n}))^2]/n$$

Question. Is this task learning or optimization ?

Categorization of Optimization Problems

Decision Variable-Based Categorization: Continuous or Discrete (Combinatorial)?

1. Continuous Optimization Problems

- Linear Programming Problems
- Nonlinear Programming Problems
- Learning of Neural Networks (Weight Adjustment)

2. Discrete (Combinatorial) Optimization Problems

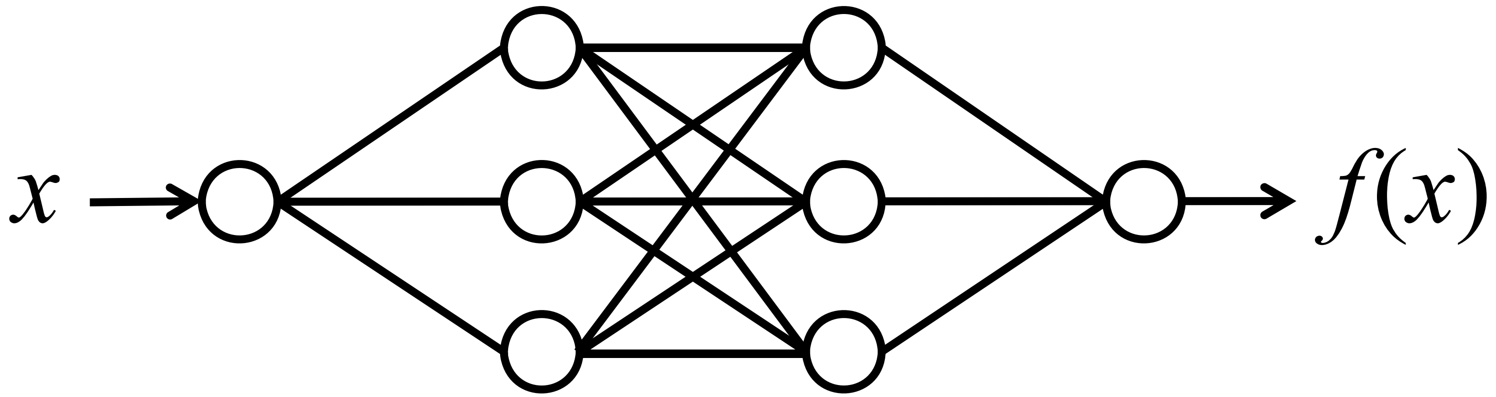
- Integer Programming Problems
- Scheduling Problems, Knapsack Problems, TSP
- Structure Learning of Neural Networks

3. Mixture of 1 and 2

- Mixed Integer Programming Problems
- Design Optimization (Length, Choice of Parts & Materials)
- Neural Network Design (Structure & Weights)

Continuous Optimization Problem

Training Data: $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$

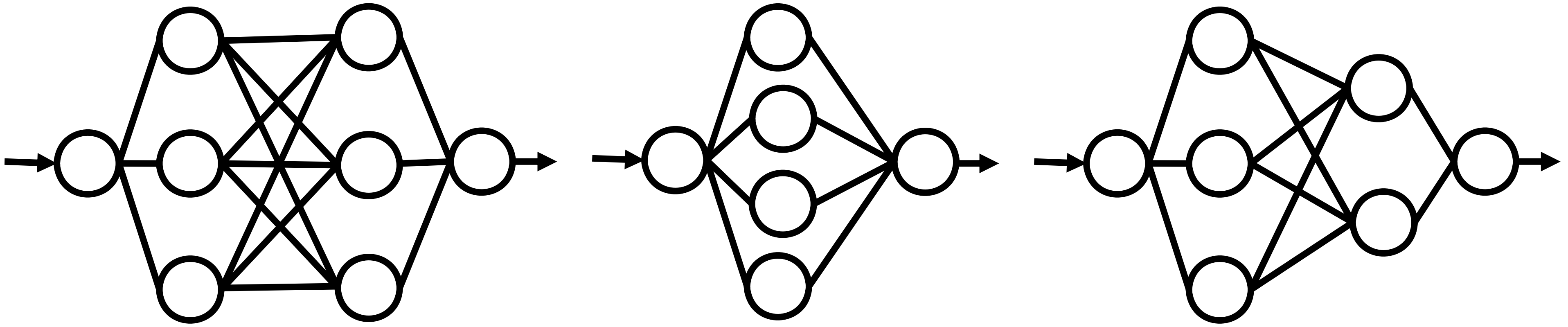


Optimization: To adjust the connection weights to minimize the following cost function (training data accuracy):

$$z(\mathbf{w}) = [(y_1 - f(x_1))^2 + \ldots + (y_m - f(x_m))^2]/m$$

Mixed Optimization Problem (Continuous & Combinatorial)

Training Data: $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$



Optimization: To find an architecture and connection weights to minimize the following cost function:

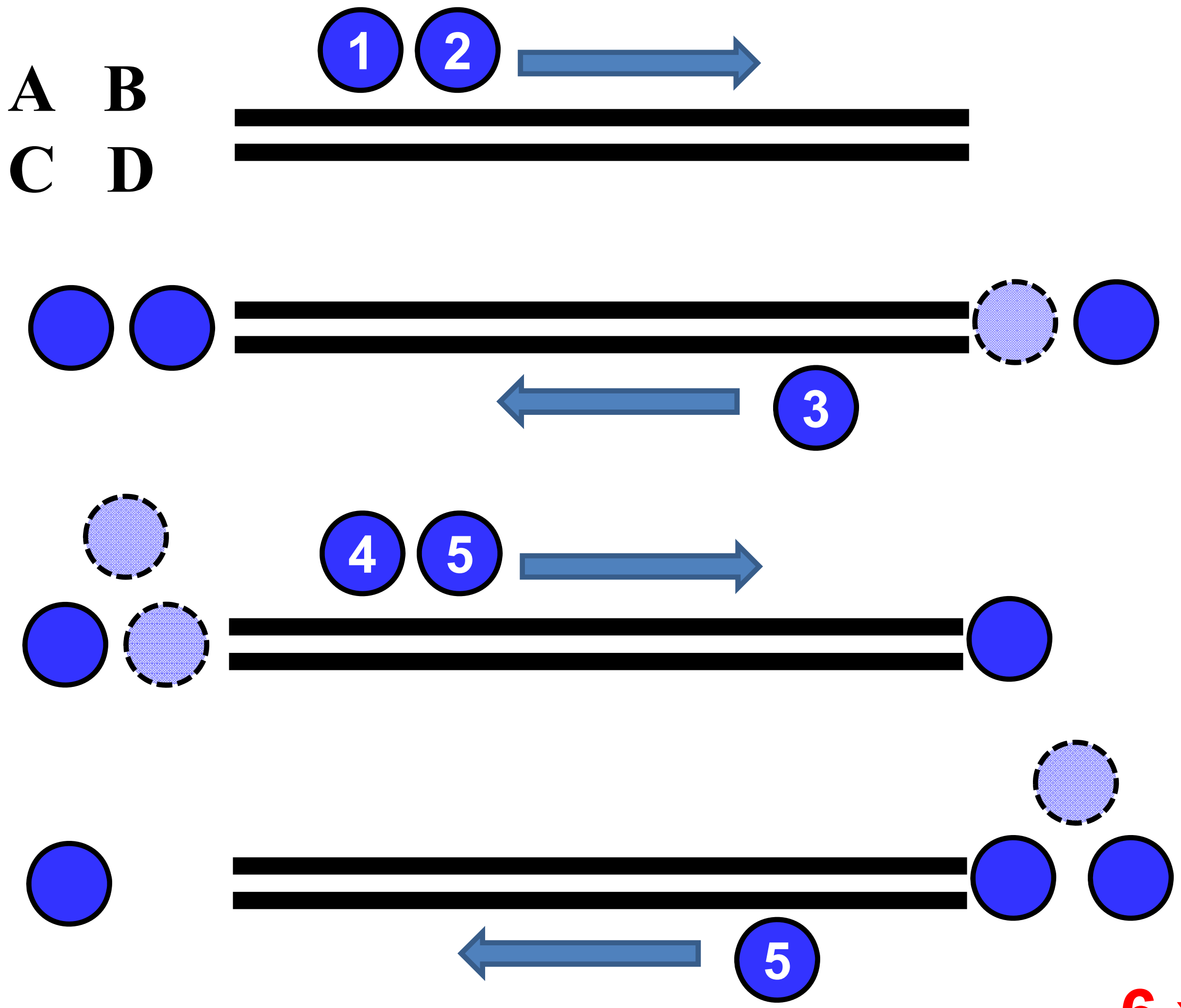
$$z(\mathbf{w}) = [(y_1 - f(x_1))^2 + \ldots + (y_m - f(x_m))^2]/m$$

Quiz to Warm Up (Wake Up)

Four travelers (Mr. A, B, C, and D) have to cross a bridge over a deep ravine. It is a very dark night and the travelers only have one oil lamp. The lamp is essential for successfully crossing the ravine because the bridge is very old and has plenty of holes and loose boards. What is worse, its construction is quite weak and it can only support two men at any time. It turns out that each traveler needs a different amount of time to cross the bridge. Mr. A is young and fast, and only needs two minutes to cross the bridge. Mr. D, on the other hand, is an old man who recently had a hip replacement and will need 11 minutes to get across the bridge. Mr. B and Mr. C need three and six minutes, respectively. And since each traveler needs the light to cross, it is the slower man in a pair who determines the total time required to make the crossing. The question is how should the men schedule (i.e., organize) themselves to cross the bridge in the shortest possible time? **Please send your answer to the TA: ?? minutes.** (This is a combinatorial optimization problem: scheduling).

Search Space Size:

Total number of possible plans: 108 .



$${}_4C_2 = 6$$

$${}_2C_1 = 2$$

$${}_3C_2 = 3$$

$${}_3C_1 = 3$$

6 x 2 x 3 x 3 = 108

Categorization of Optimization Problems

Objective Function-Based Categorization 1: Number of Objectives: Single or Multiple?

1. Single-Objective Optimization Problems

Many textbook optimization problems (test problems) have only a single objective such as the profit maximization in a knapsack problem and the processing time minimization in a scheduling problem. In machine learning, the accuracy maximization can be viewed as a single-objective optimization problem.

2. Multi-Objective Optimization Problems

Almost all real-world problems have multiple objectives (e.g., cost minimization and performance maximization). In the structure learning of neural networks, the complexity minimization and the accuracy maximization can be viewed as a two-objective problem. However, in machine learning, they are combined as a weighted sum in order to handle them in the single-objective framework.

Categorization of Optimization Problems

Objective Function-Based Categorization 2: Objective Value: Exact Value or Others?

1. Standard Optimization Problems

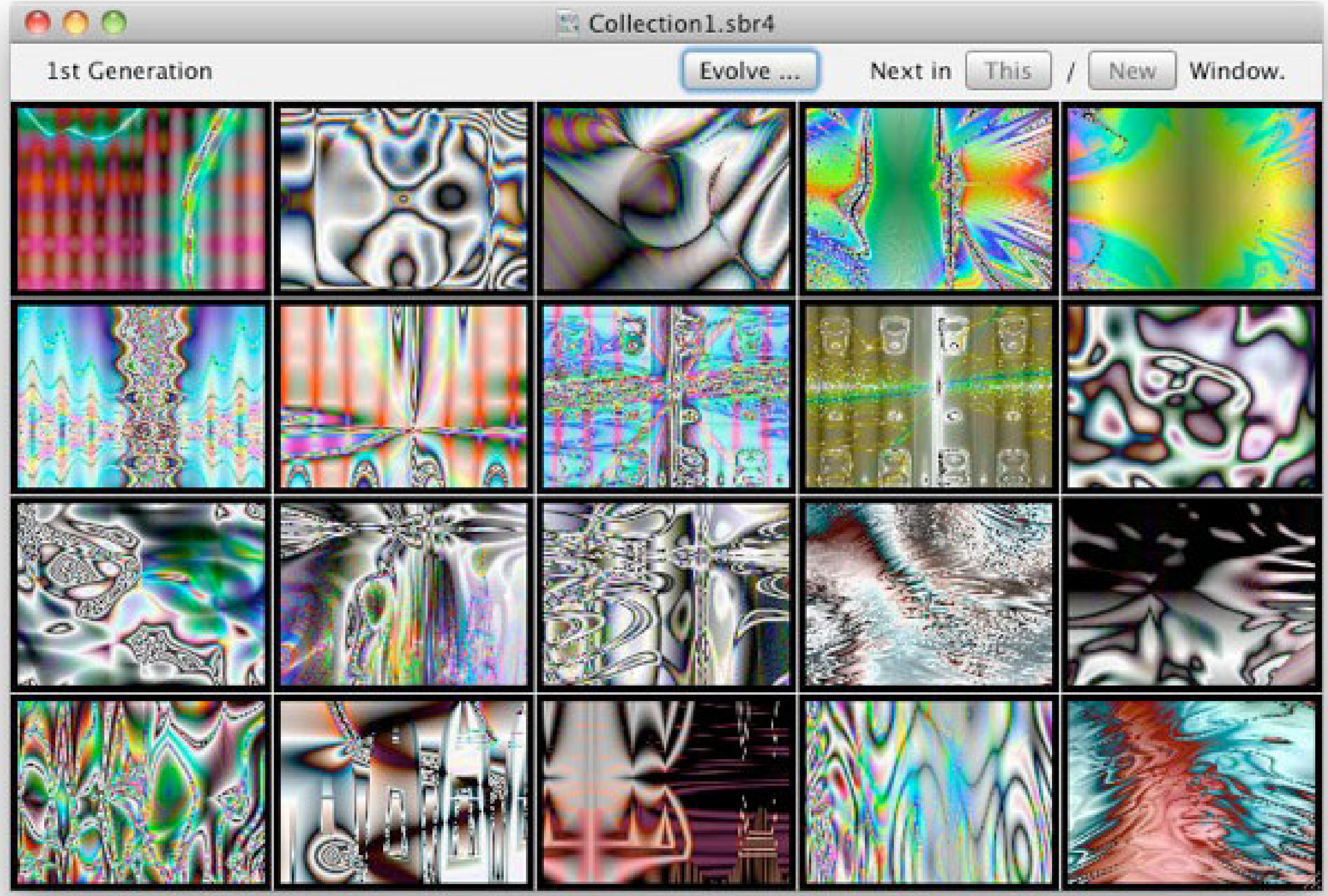
Objective values are exact values (e.g., real numbers).

2. Optimization Problems with Some Uncertainty

- Interval Programming Problems (e.g., [100$, 110$]).
- Stochastic Programming Problems (e.g., $N(100,1^2)$).
- Fuzzy Programming Problems (e.g., about 100$).
- Noisy Optimization Problems
  (e.g., for the same solution, its objective value is 97, 102, 98, ...).
- Simulation-based Optimization Problems (e.g., multi-agent)
  (Simulation results depend on randomness and parameters).
- Interactive Optimization Problems (e.g., CG design)
  (Solutions are evaluated by human: noisy, inconsistent, tired).

Interactive Evolution of Computer Graphics

http://www.intlab.soka.ac.jp/~unemi/sbart/4/ja/



Categorization of Optimization Problems

Constraint Condition-Based Categorization: With or Without Constraint Conditions?

1. Unconstrained Optimization Problems

Optimization problems with no constraint conditions.

- Simple test problems such as “minimization of $(x-1)^2$”.
- TSP (Travelling Salesman Problem)
- Some Scheduling Problems

2. Constrained Optimization Problems

Optimization problems with constraint conditions.

- Linear Programming Problems
- Knapsack Problems
- Almost All Real-World Problems

Categorization of Optimization Algorithms

Is the obtained solution always optimal?

1: Exact Optimization Algorithms

- Linear Programming
- Dynamic Programming
- Branch-and-Bound Method

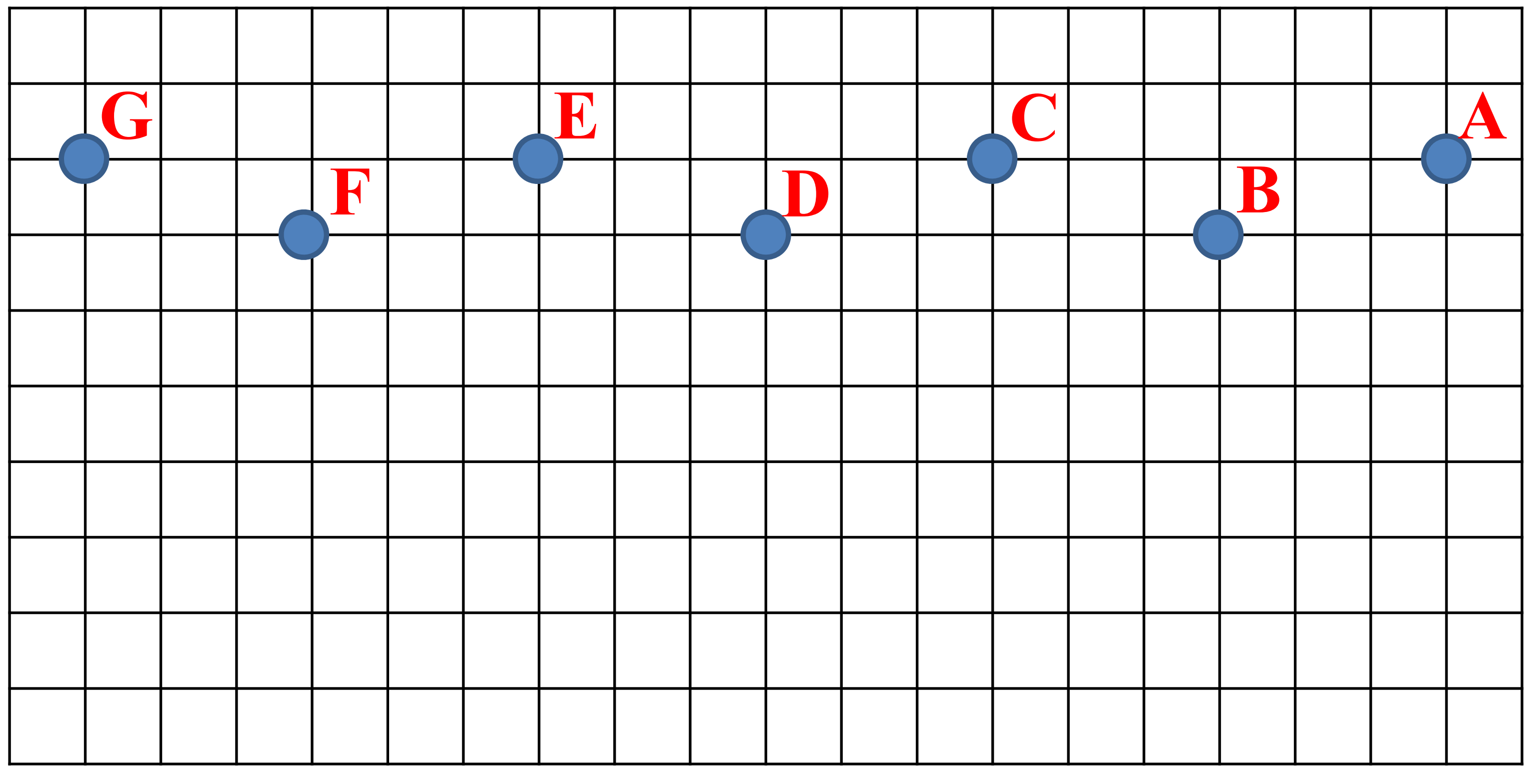
2: Approximation Algorithms

- Greedy Algorithms
- Local Search
- Genetic Algorithms
- Almost All Nonlinear Optimization Algorithms
- Learning of Connection Weights in Neural Networks

Quiz to Warm Up (Wake Up)

Travelling Salesman Problem (TSP): To find the shortest tour to visit all cities and return to the start city. Our start city is City A, and the next city is City B.

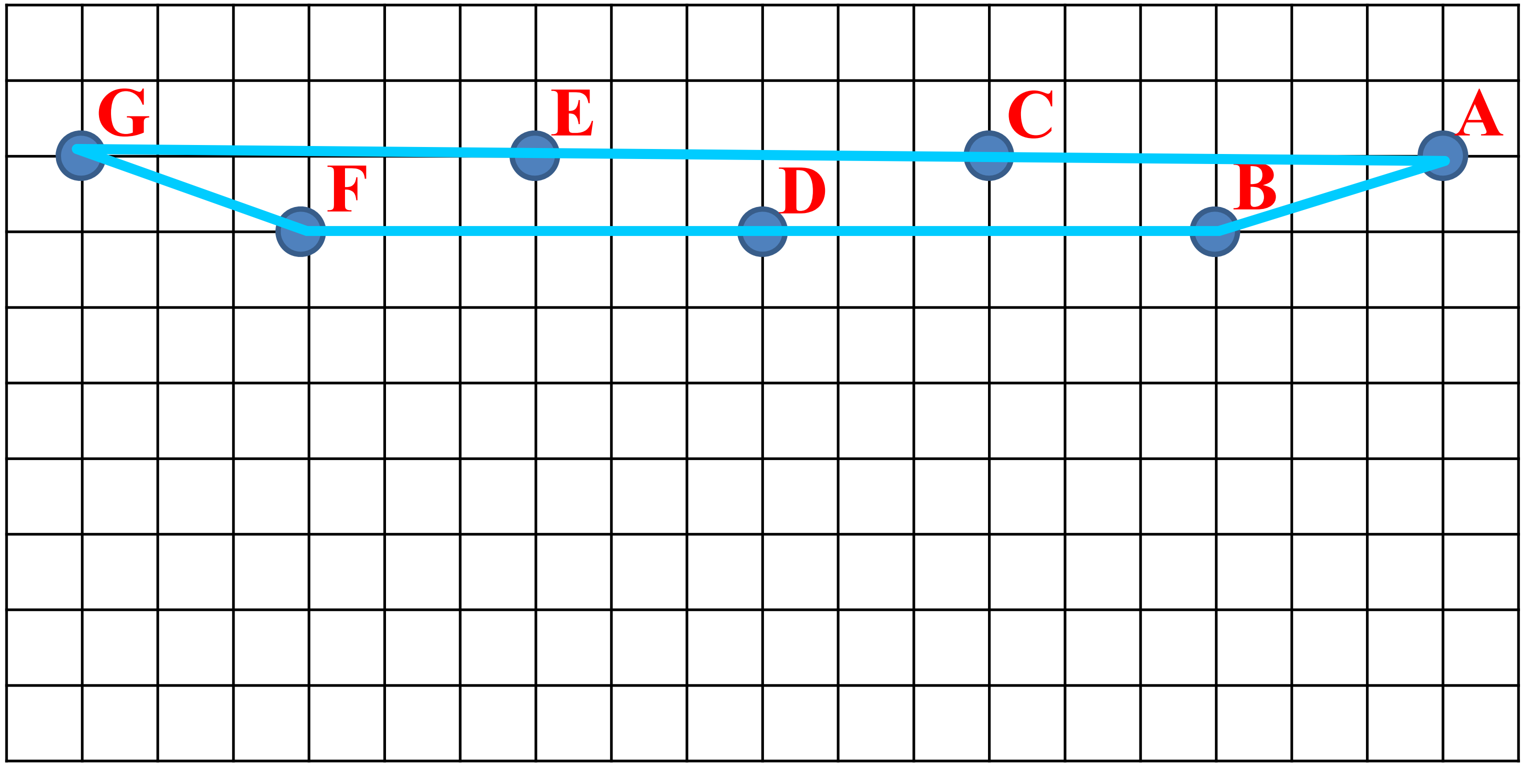
Your answer: A B ? ? ? ? ? A



Quiz to Warm Up (Wake Up)

Travelling Salesman Problem (TSP): To find the shortest tour to visit all cities and return to the start city. Our start city is City A, and the next city is City B.

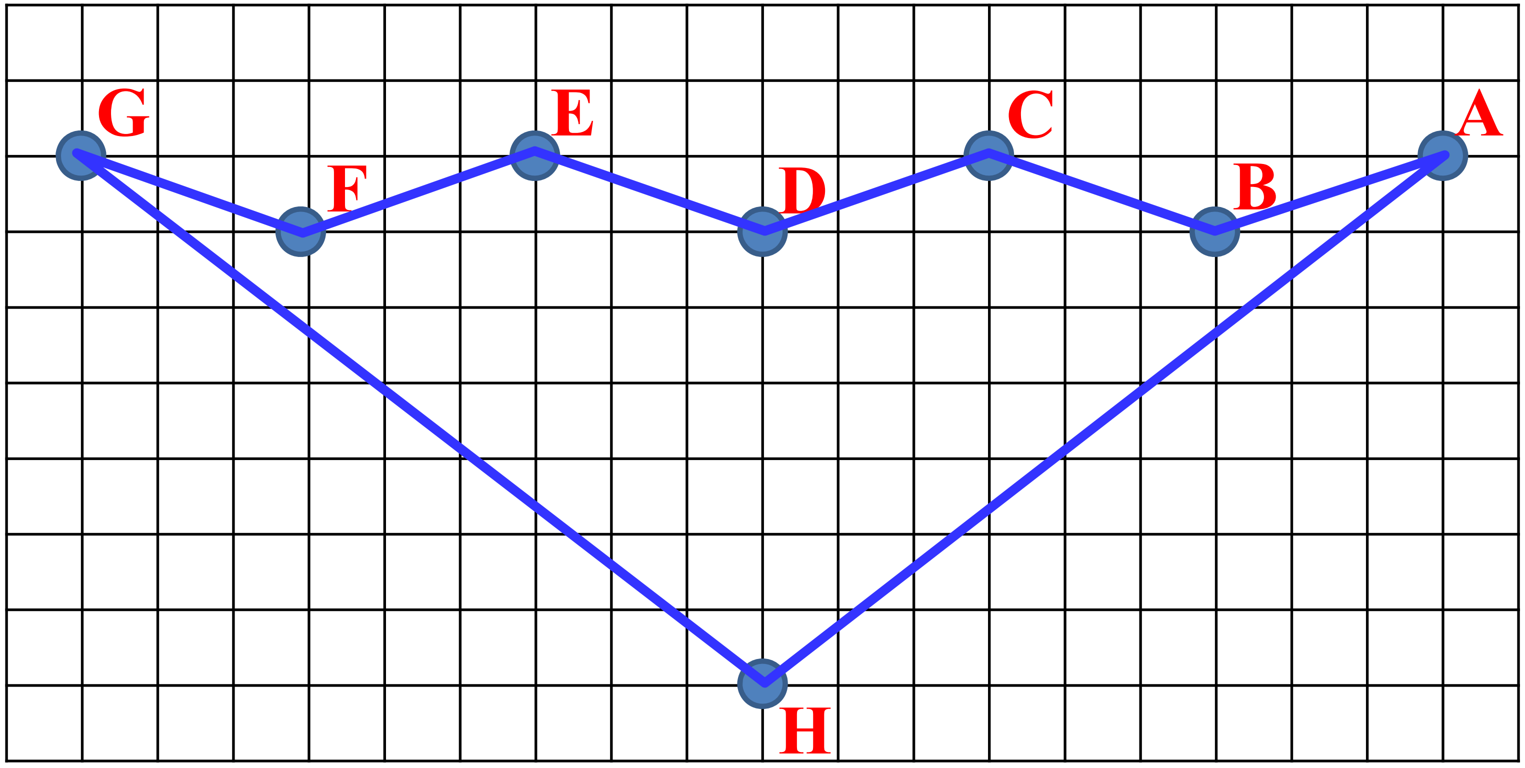
Your answer: A B D F G E C A



Quiz to Warm Up (Wake Up)

Travelling Salesman Problem (TSP): To find the shortest tour to visit all cities and return to the start city. Our start city is City A, and the next city is City B.

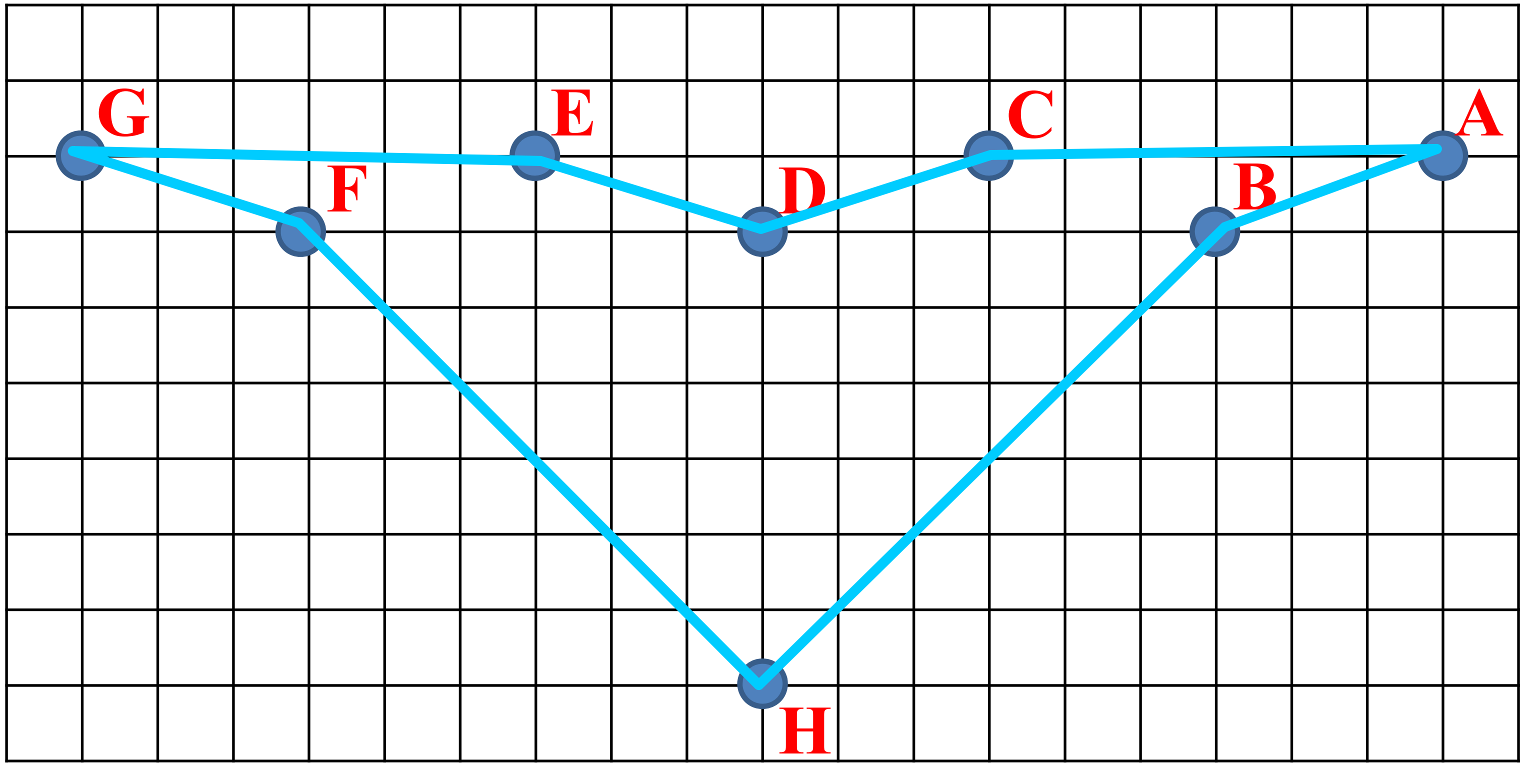
Your answer: A B ? ? ? ? ? ? A



Quiz to Warm Up (Wake Up)

Travelling Salesman Problem (TSP): To find the shortest tour to visit all cities and return to the start city. Our start city is City A, and the next city is City B.

Your answer: A B ? ? ? ? ? ? A



Discussions about Computation Load

For combinatorial optimization problems:

1: Exact Optimization Algorithms

Try to find the optimal solution without examining all possible combinations. However, in the worst case, the order of computation load is similar to the calculation of all possible combinations.

Example 1: Choice of a group of 5 students from 10 students. This is a simple problem. You can find the optimal solution by examining all combinations: 252 combinations (10x9x8x7x6/(5x4x3x2x1)). If you can examine **one combination in one minute**, you will be able to examine all combinations and choose the best one **within 5 hours**.

Example 2: Choice of a group of 5 students from 100 students. This is basically the same problem as Example 1. The number of possible combinations is 75,287,520 (100x99x98x97x96/(5x4x3x2x1)). If your computer can examine **1000 combinations in one second**, it can examine all combinations **within a day**.

Discussions about Computation Load

For combinatorial optimization problems:

1: Exact Optimization Algorithms

Example 3: Choice of a group of 10 students from 1000 students. This is basically the same problem as Examples 1-2. The number of possible combinations is $\boxed{\qquad\qquad\qquad\qquad}$.
Even if your computer can examine **1 million combinations in one second**, it needs **about** $\boxed{\qquad}$ **years** to examine all combinations.

Calculate now!

2: Simple Heuristic Algorithms

Example 3: Choice of a group of 10 students from 1000 students. First we evaluate each student independently. Then we choose the best 10 students. If we can use your one-million/second computer in Example 3, this calculation is within a second.
Difficulty: No consideration about interaction among students. The group of the best 10 students is not always the best team. However, this choice is usually much better than random choice.

Discussions about Computation Load

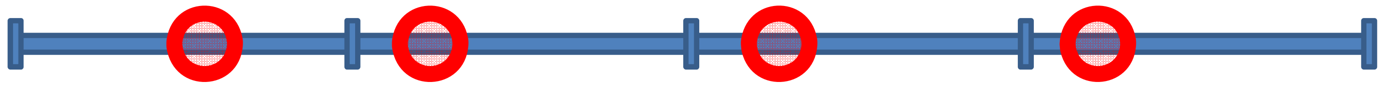
For combinatorial optimization problems:

3: Greedy Algorithms

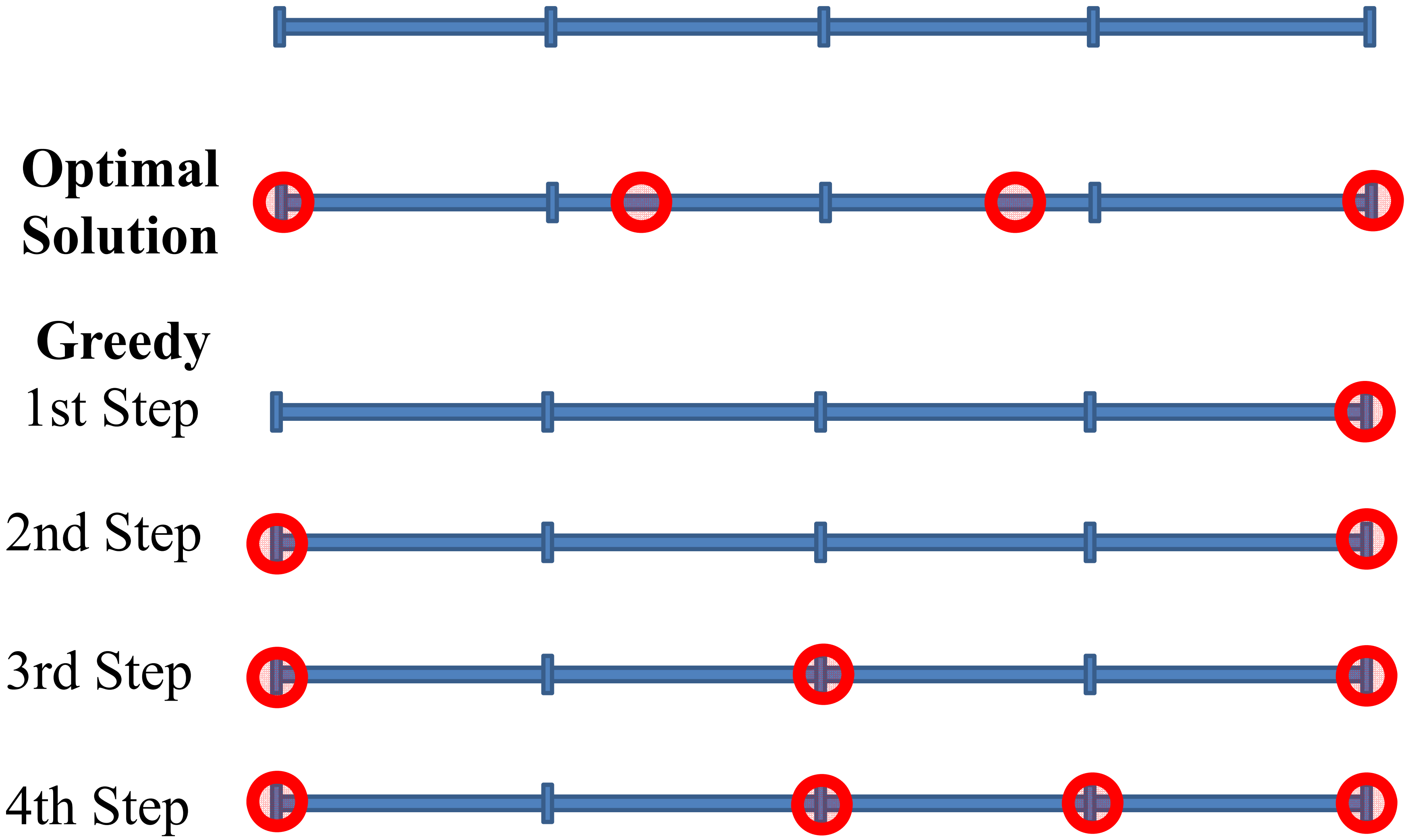
Example 3: Choice of a group of 10 students from 1000 students. First we evaluate each student independently, and choose the best student. Next we evaluate all combinations (999 combinations) of the selected student and another student, and choose the best combination. Then we evaluate all combinations (998 combinations) of the selected two students and another student, and choose the best combination. In this manner, we can choose a good combination of 10 students. If we can use your one-million/second computer in Example 3, this calculation is within a second.

Advantages: Maybe better than the choice of the best 10 students, still very fast. **Disadvantages:** The selection result can be totally different from the optimal combination since the selection is based on only a small number of possible combinations.

Simple Example: To select four points on the line to maximize the minimum distance between two points.



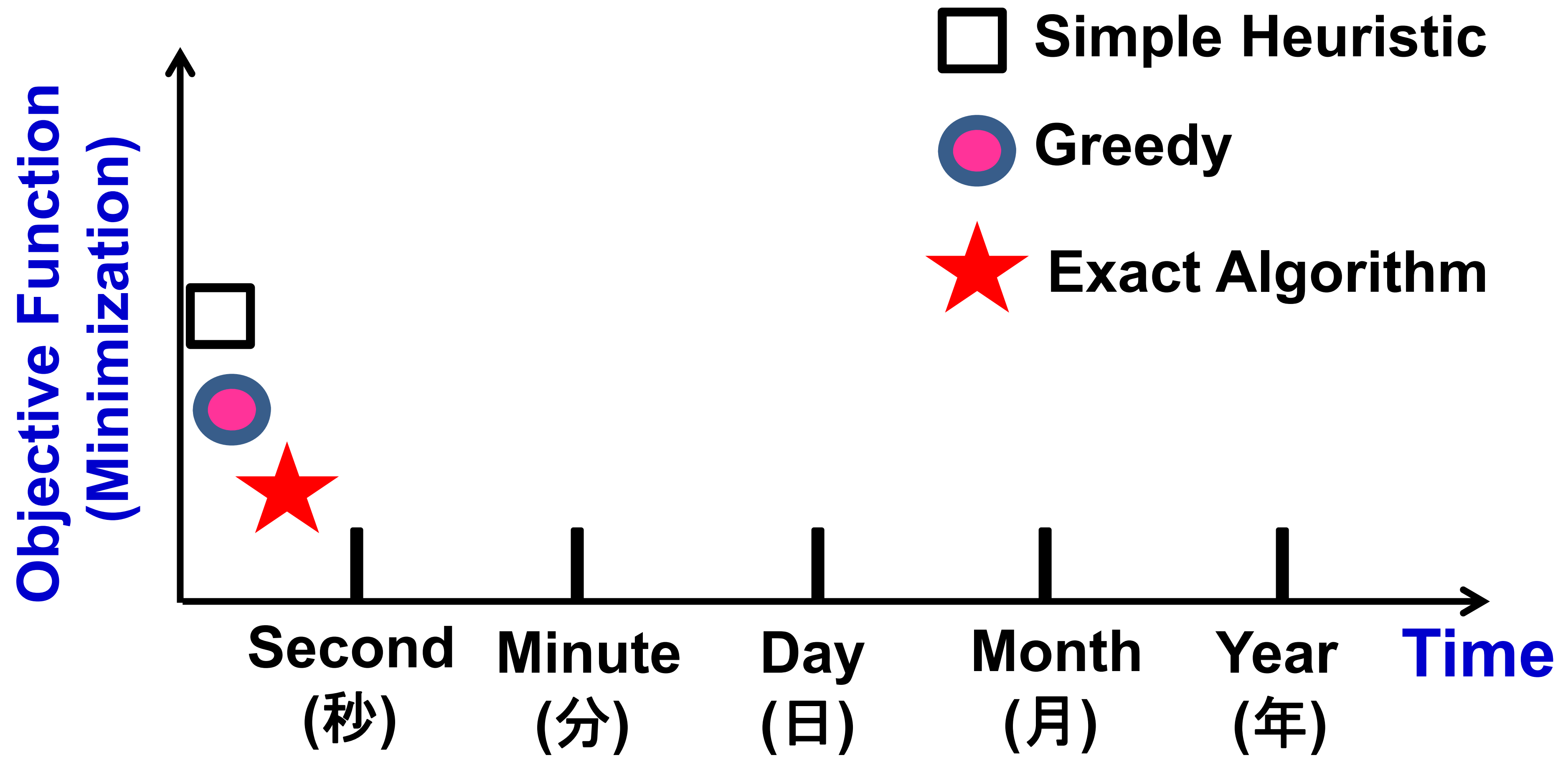
Simple Example: To select four points on the line to maximize the minimum distance between two points.



Discussions about Computation Load

For combinatorial optimization problems:

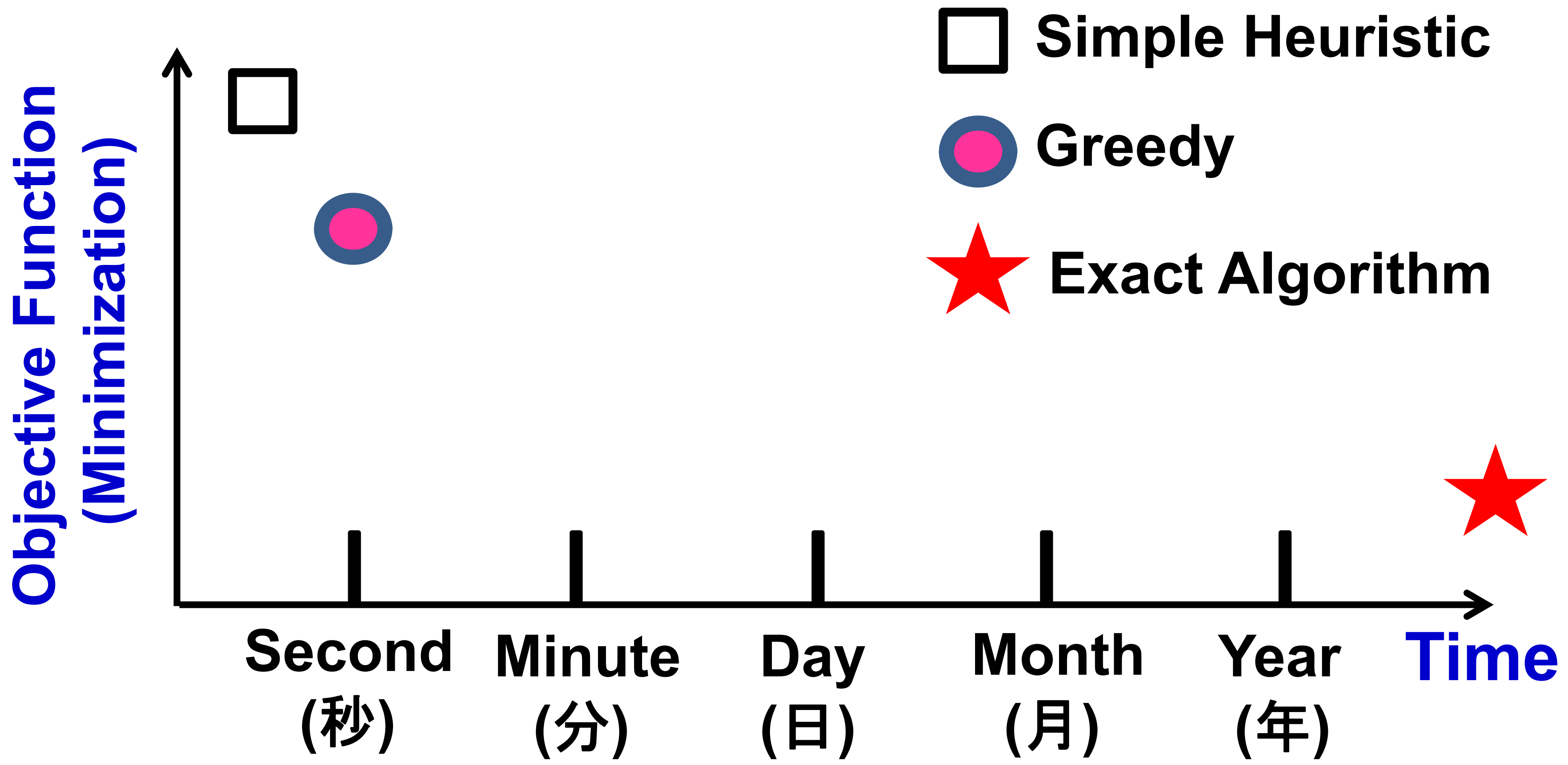
For Small Problem (Selection of 5 from 10)



Discussions about Computation Load

For combinatorial optimization problems:

For Large Problem (selection of 10 from 1000)



Discussions about Computation Load

For combinatorial optimization problems:

1: Exact Optimization Algorithms

Example 4: Assignment of 4 research topics to 4 students. The total number of different combinations is only 4x3x2x1=24.

Example 5: Assignment of 20 research topics to 20 students. The total number of different combinations is 20x19x...x1=2.4x10^{18}. Even if we can use your **one-million/second** computer, we need **77,000 years** to examine all combinations.

2: Simple Heuristic Algorithms

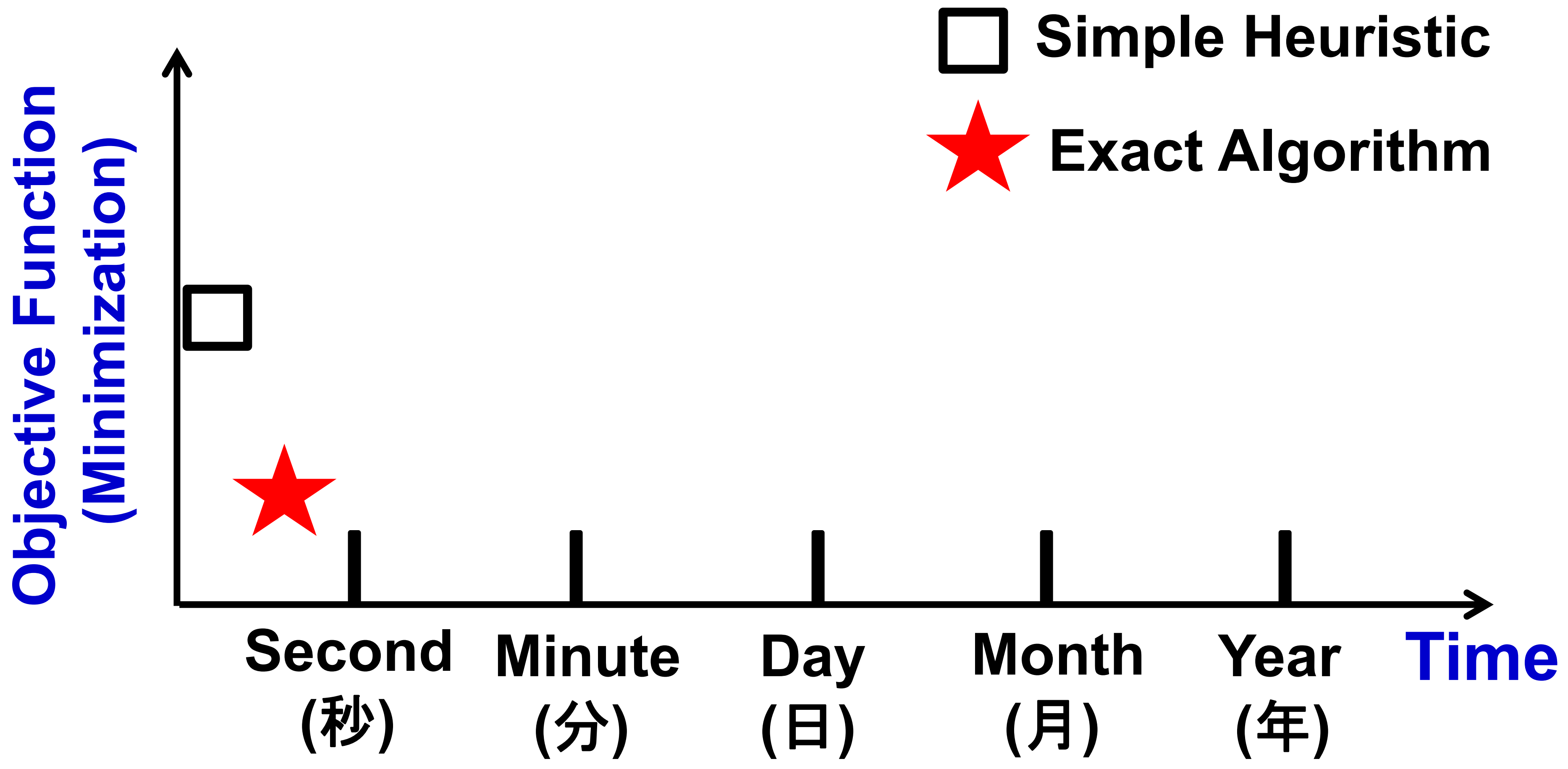
Example 5: Evaluate each research topic and sort them from the most difficult topic to the simplest topic. Evaluate each student and sort them from the best student to the worst student. Then we can assign the 20 topics to the 20 students depending on the topic difficulty.

Advantage: Very fast

Discussions about Computation Load

For combinatorial optimization problems:

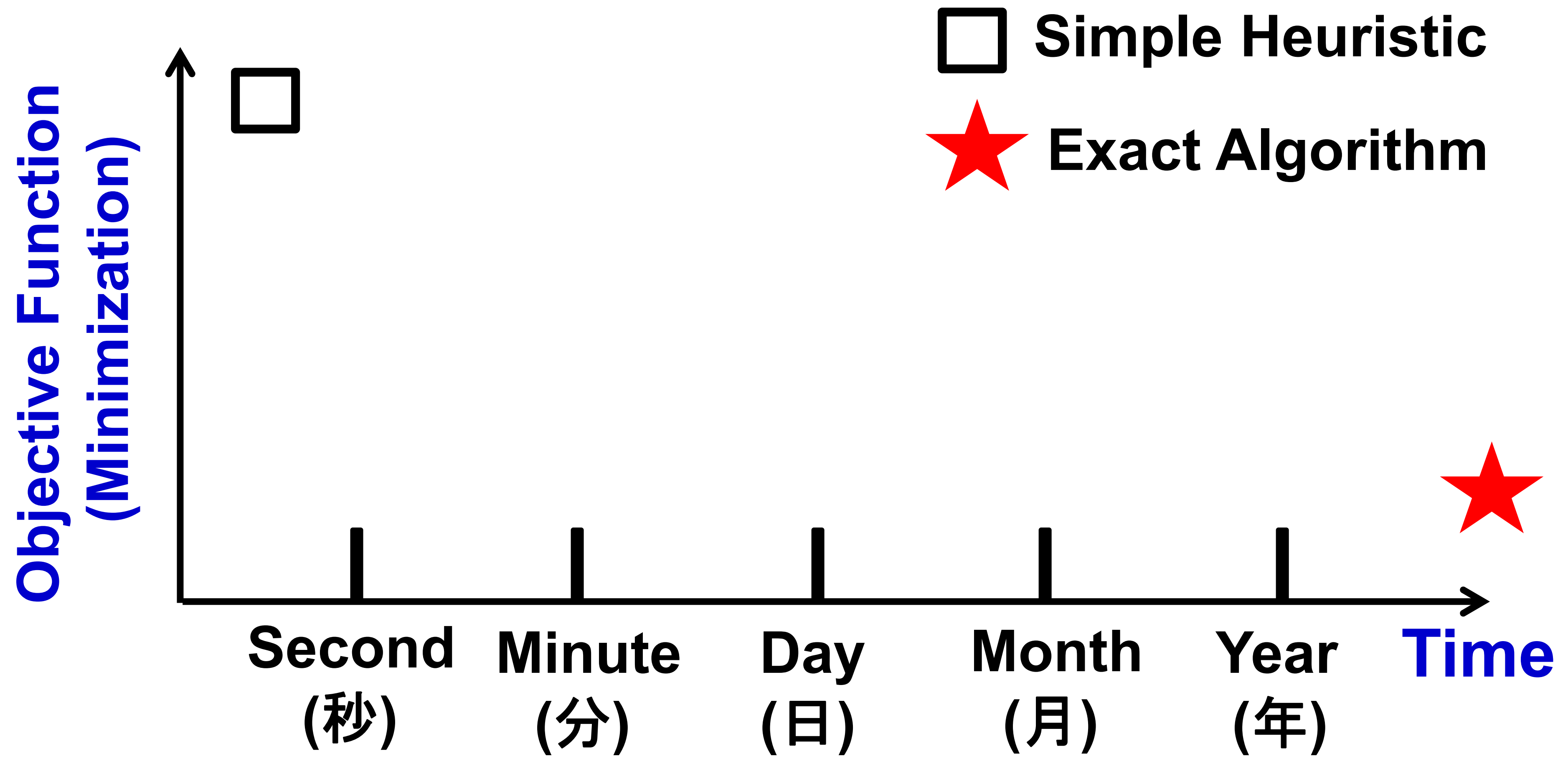
For Small Problem (Assignment of 4 topics)



Discussions about Computation Load

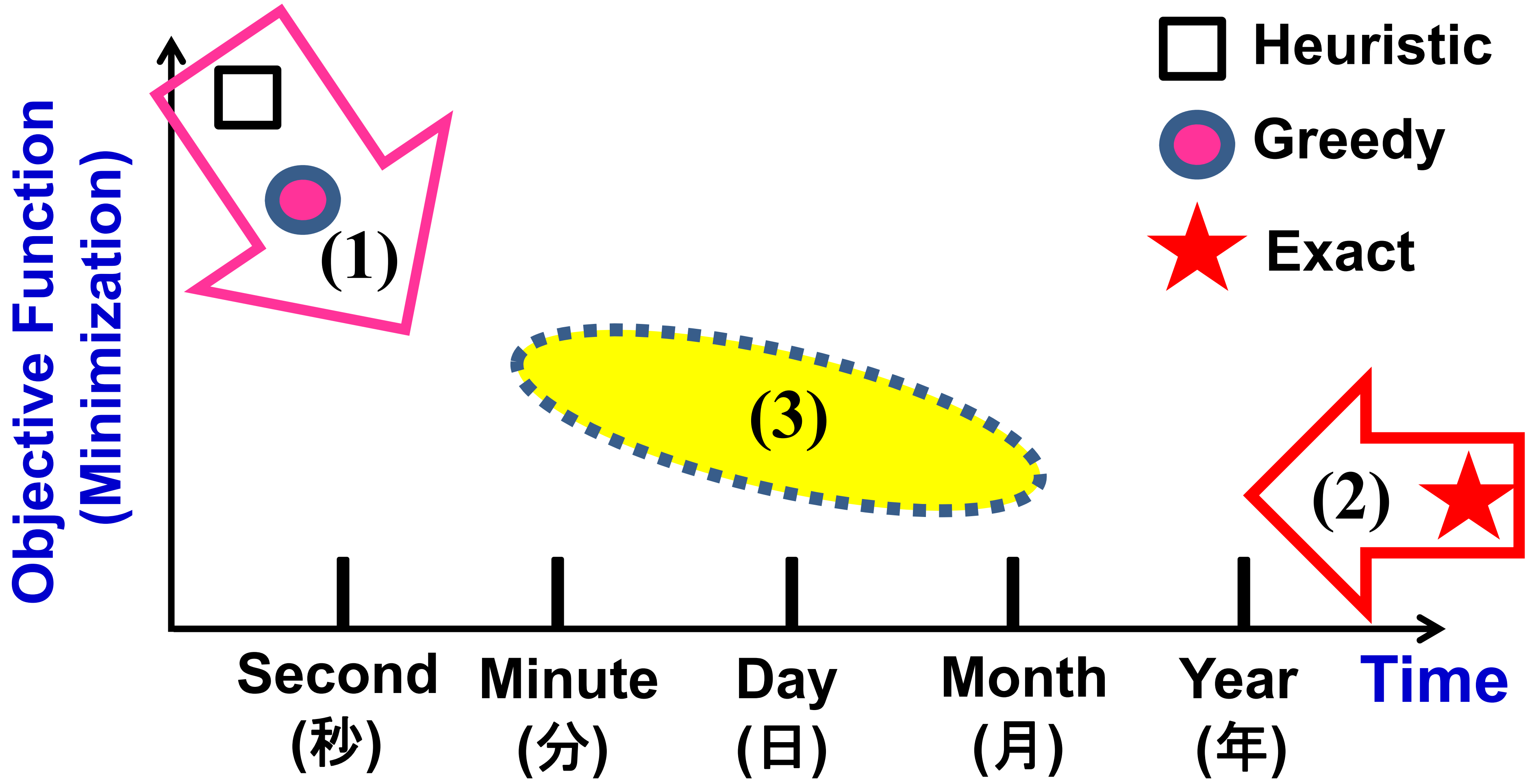
For combinatorial optimization problems:

For Large Problem (Assignment of 20 topics)



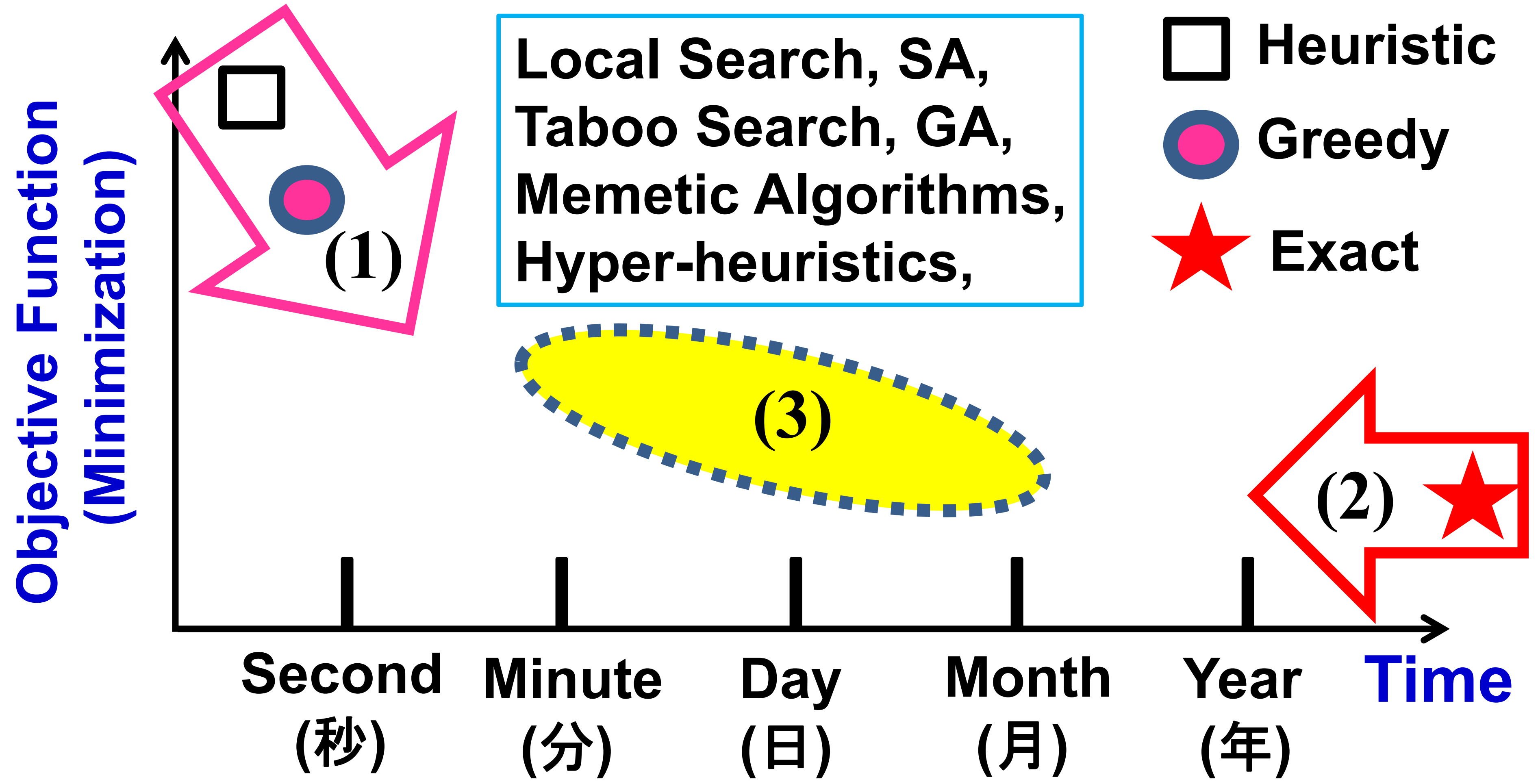
Research Directions in Optimization

(1) Accuracy improvement
(2) Efficiency improvement
(3) Huge gap between (1) and (2)



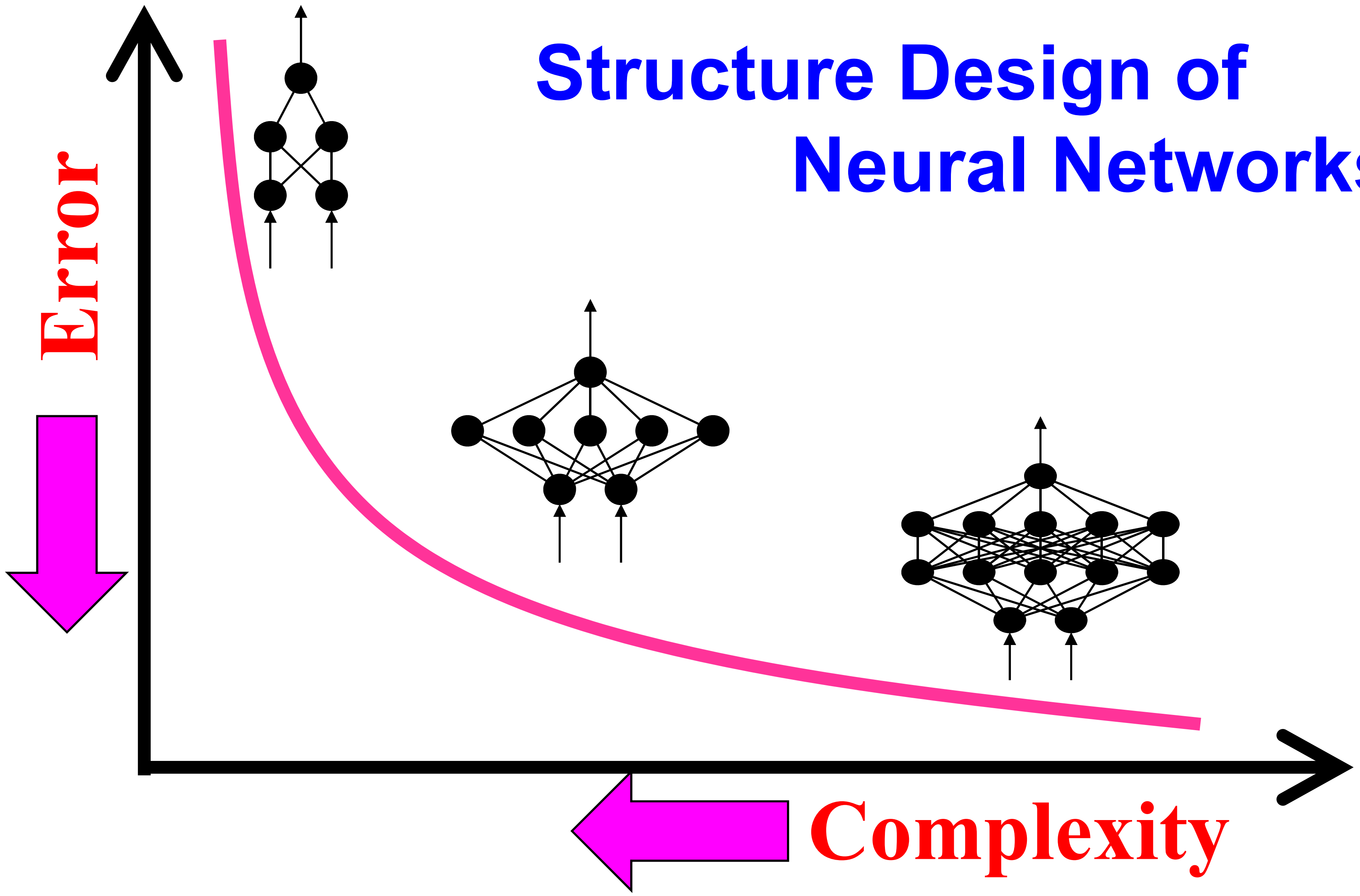
Focus of this course: Huge gap between (1) and (2)

- We have much more time than (1): One solution.
- We do not have enough time for (2): All solutions.



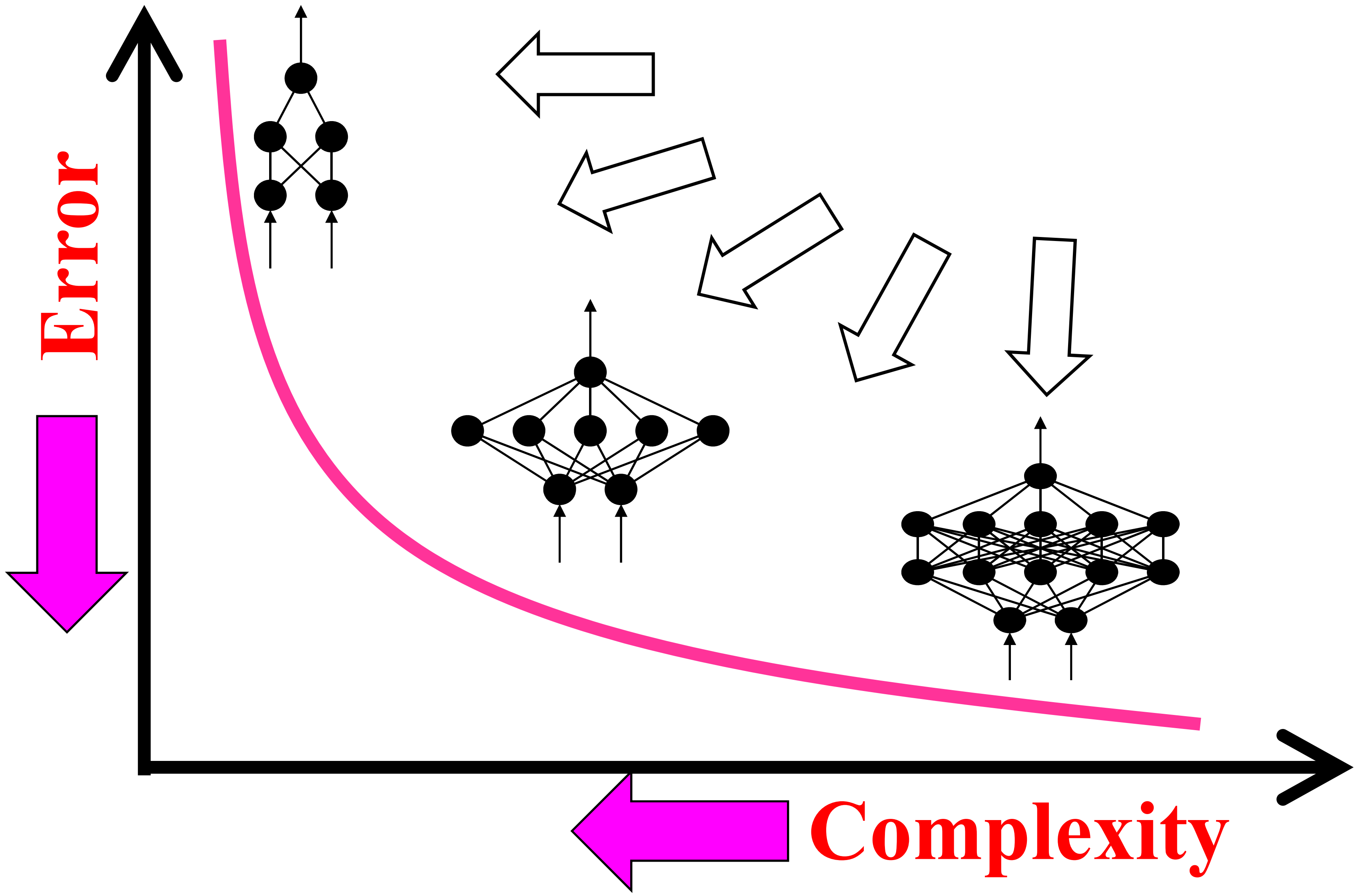
The Other Focus: Multi-Objective Optimization

Almost all problems have multiple objectives



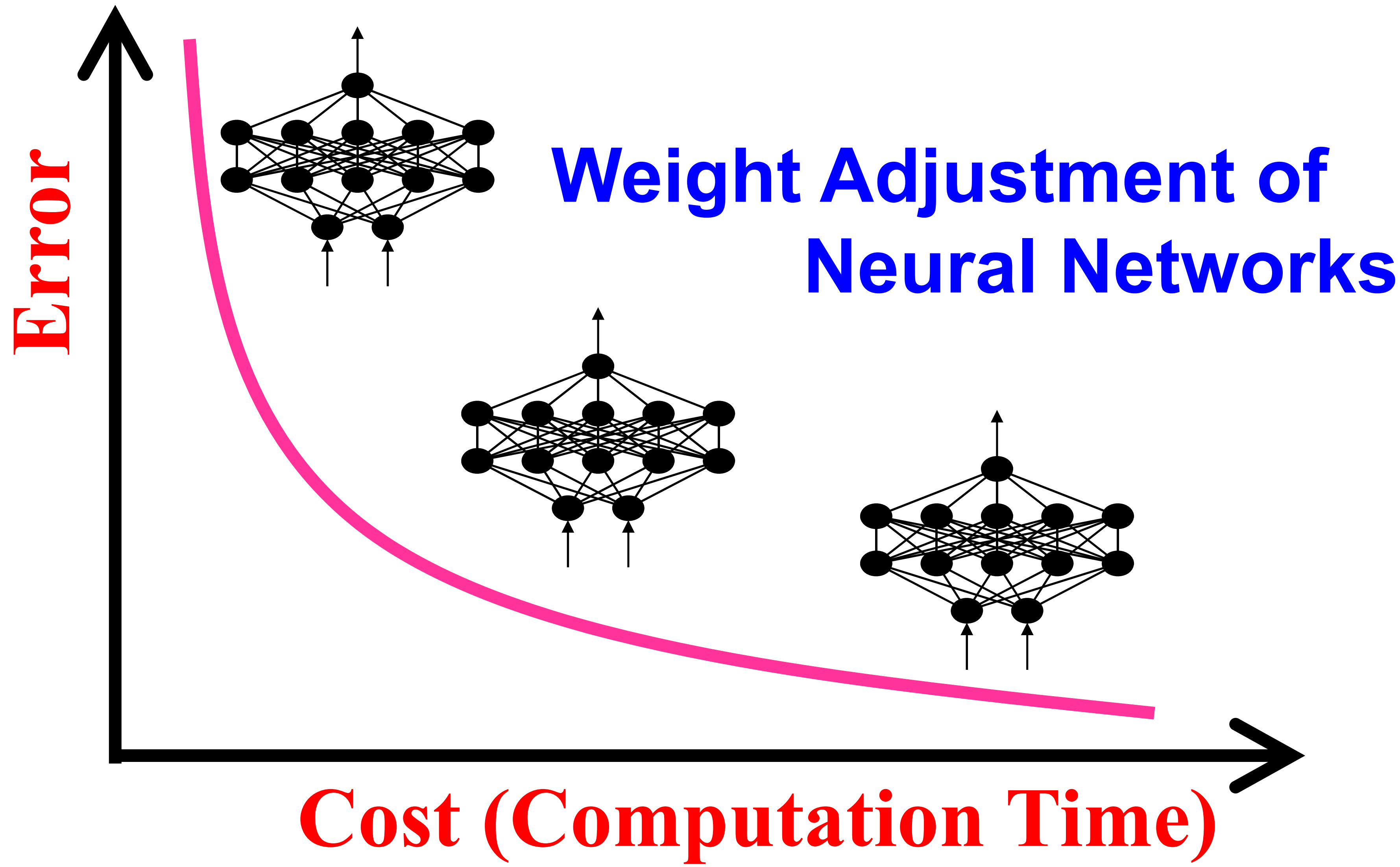
The Other Focus: Multi-Objective Optimization

Almost all problems have multiple objectives



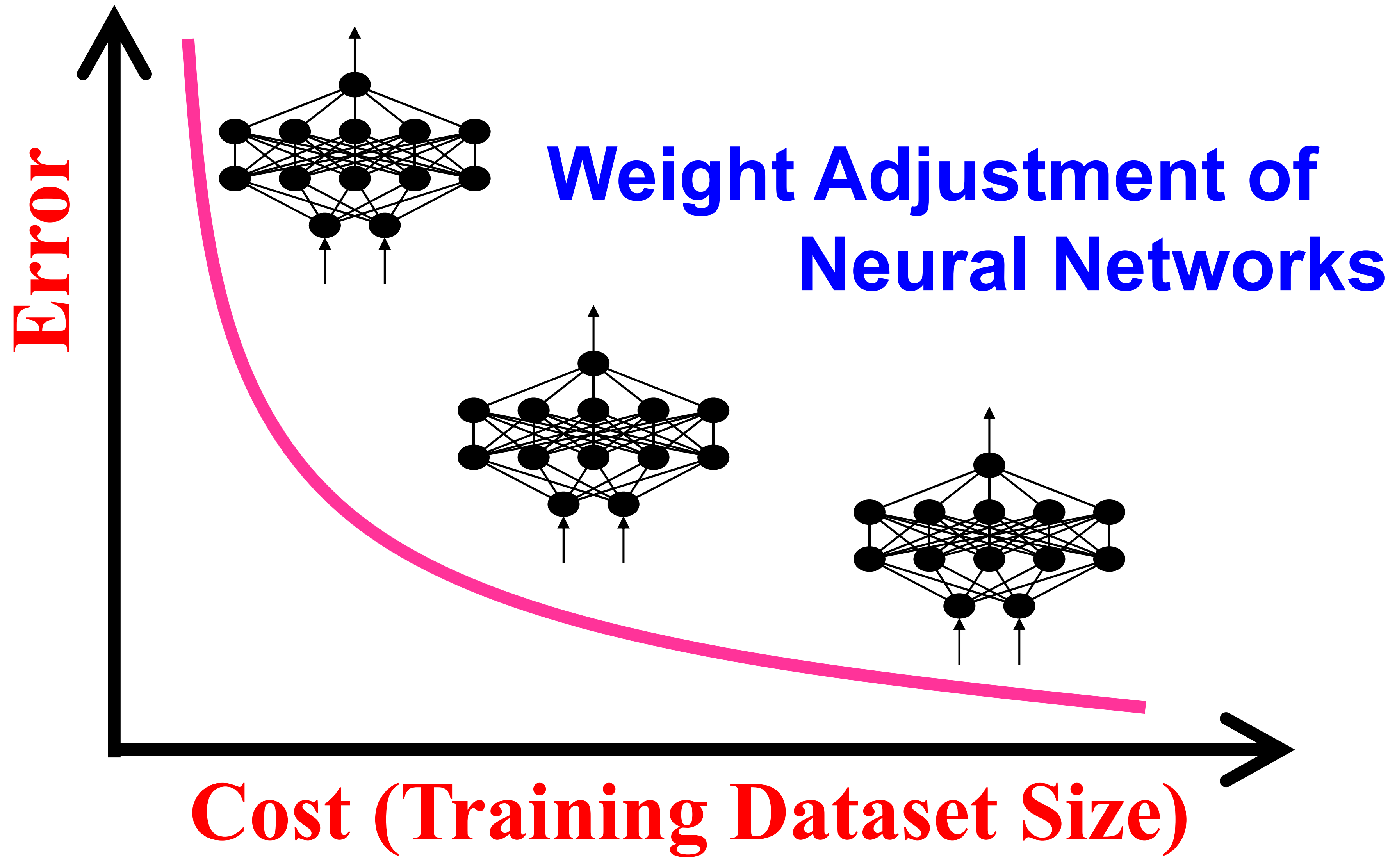
The Other Focus: Multi-Objective Optimization

Almost all problems have multiple objectives



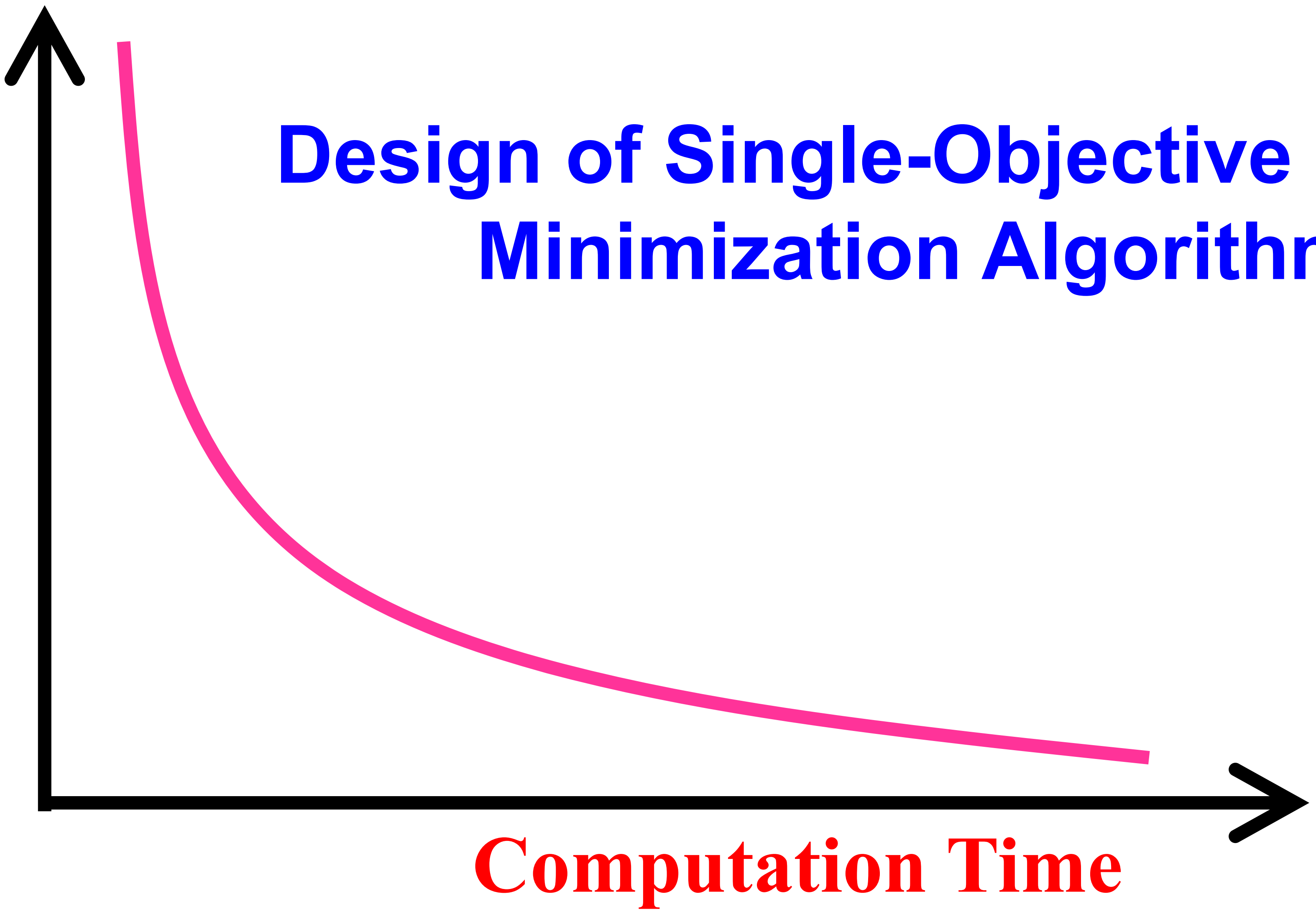
The Other Focus: Multi-Objective Optimization

Almost all problems have multiple objectives


Error
Weight Adjustment of Neural Networks
Cost (Training Dataset Size)

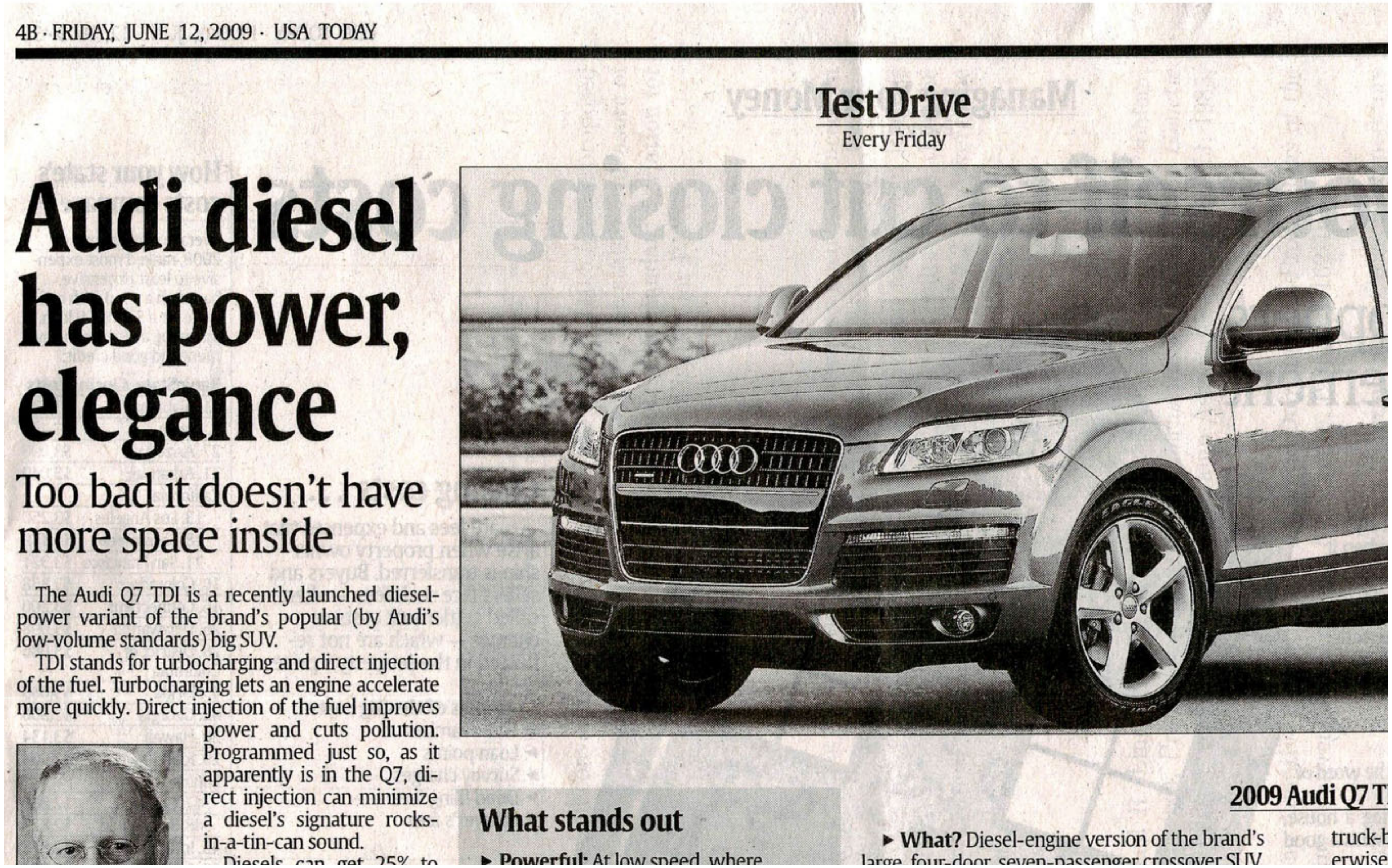
The Other Focus: Multi-Objective Optimization

Almost all problems have multiple objectives



The Other Focus: Multi-Objective Optimization

Almost all problems have multiple objectives



4B · FRIDAY, JUNE 12, 2009 · USA TODAY

Test Drive
Every Friday

Audi diesel has power, elegance

Too bad it doesn't have more space inside

The Audi Q7 TDI is a recently launched diesel-power variant of the brand's popular (by Audi's low-volume standards) big SUV.

TDI stands for turbocharging and direct injection of the fuel. Turbocharging lets an engine accelerate more quickly. Direct injection of the fuel improves power and cuts pollution. Programmed just so, as it apparently is in the Q7, direct injection can minimize a diesel's signature rocks-in-a-tin-can sound.

Diesels can get 25% to

What stands out

▸ **Powerful:** At low speed, where

2009 Audi Q7 T

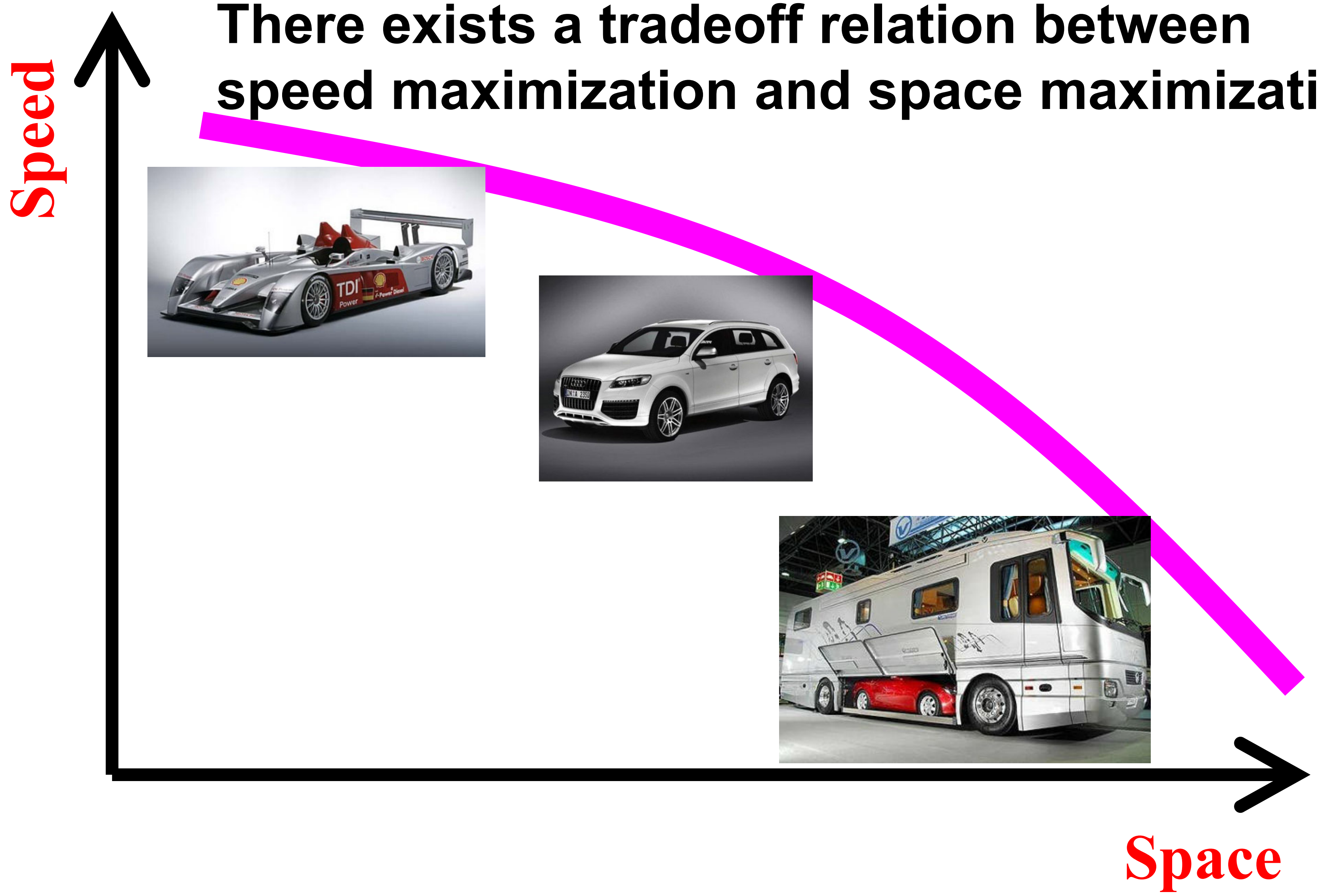
▸ **What?** Diesel-engine version of the brand's large, four-door, seven-passenger crossover SUV.

truck-b
erwise

The Other Focus: Multi-Objective Optimization

Almost all problems have multiple objectives

There exists a tradeoff relation between speed maximization and space maximization.

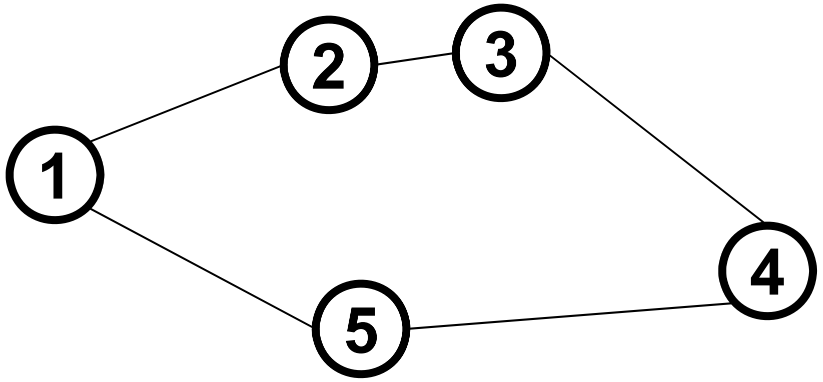


1st Week Lab Session

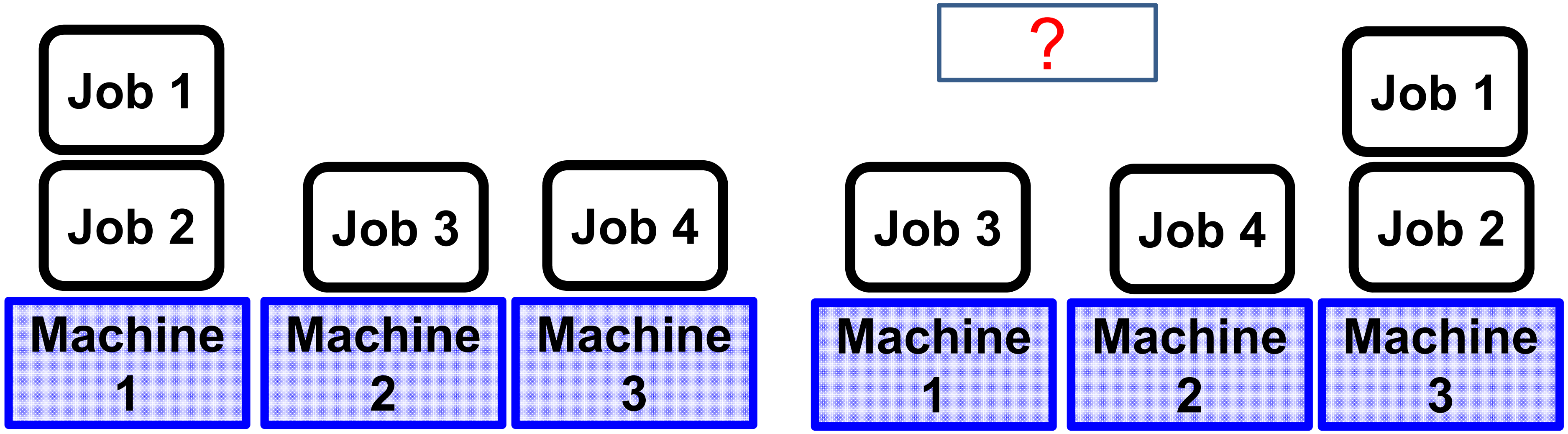
Prepare a PowerPoint file to explain how to solve each question. You will be asked to explain how to solve each question using your file at the end of lab session (e.g., 20:20 - 20:50)

Question 1: An *n*-city TSP problem (i.e., to find the shortest tour to visit all the given *n* cities and return to the start city). The problem size is the total number of different tours. This is not *n*! since many tours can be viewed as the same tour (with respect to the tour length). For example, in the following 5-city TSP problem, the tour 123451 is the same as 154321, 234512, 215432 and some other tours. How many different tours does an *n*-city TSP problem have?

$\boxed{?}$

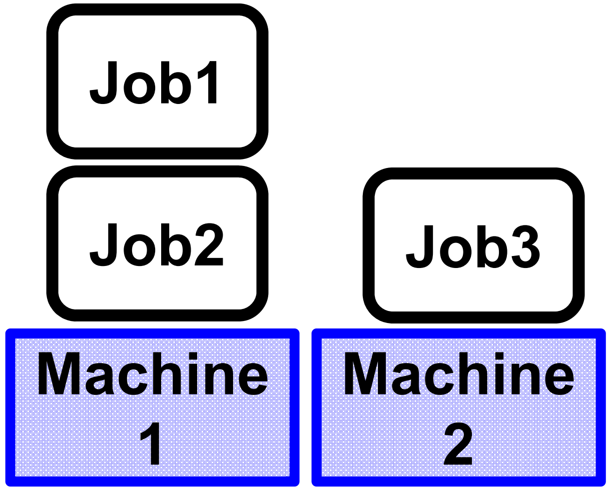


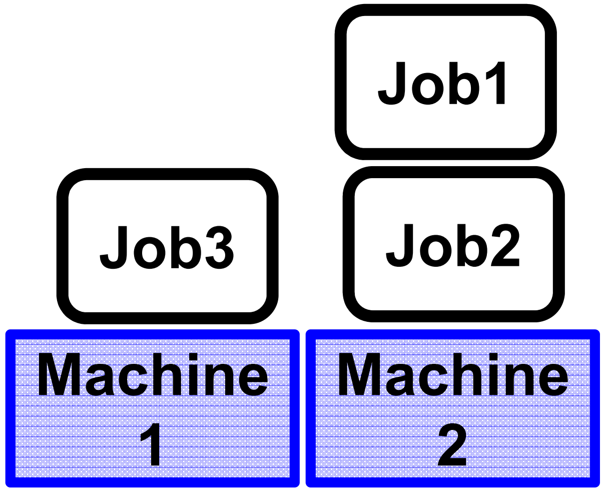
Question 2: A 3-machine 4-job load balancing problem. This problem is to find the best assignment of four jobs to three identical machines. Note that some different assignments are viewed as the same solution since all machines are identical. For example, the assignment of all jobs to Machine 1 is viewed as the same solution as the assignment of all jobs to Machine 2 (or Machine 3). The following two assignments are viewed as the same solution. How many different solutions does the 3-machine 4-job load balancing problem have?



This problem is the same as a grouping problem of four jobs into three groups: {{J1, J2}, {J3}, {J4}} is the same as {{J3}, {J4}, {J1, J2}}.

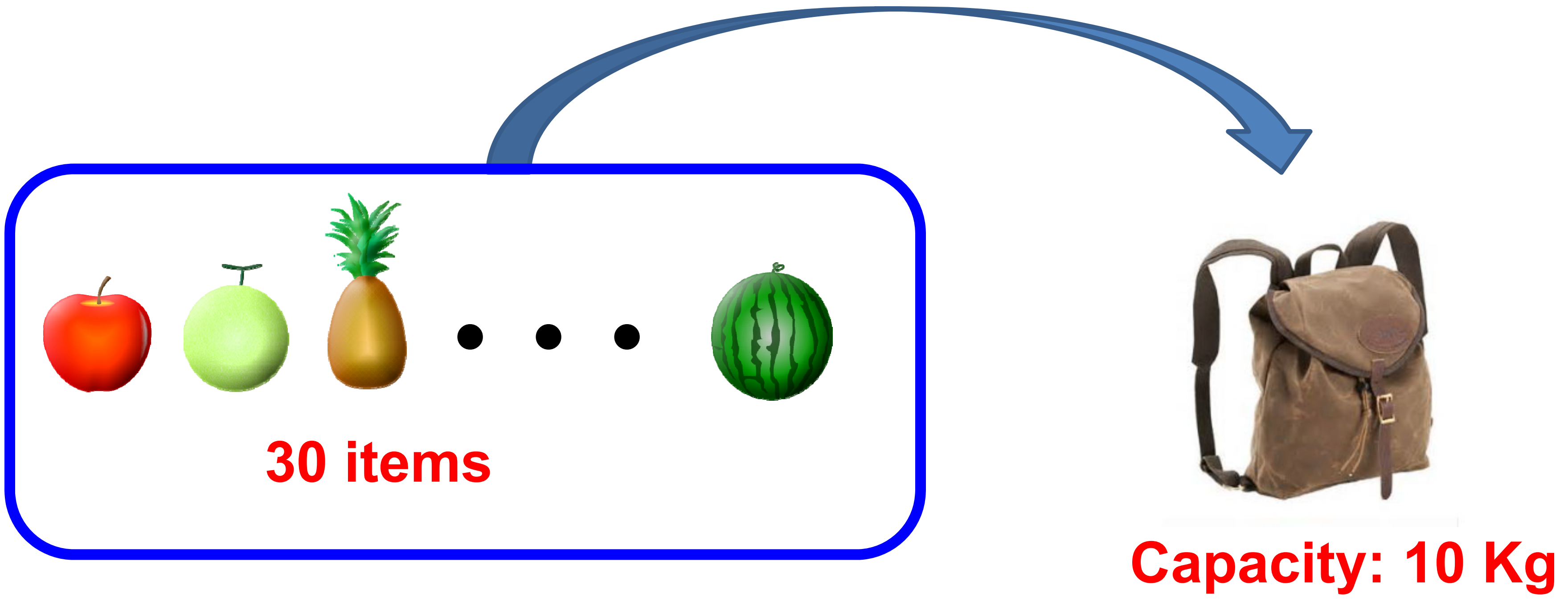
Question 3: A 2-machine *n*-job load balancing problem. As in Example 2, some different assignments (e.g., two assignments below) are viewed as the same solution since the two machines are identical. How many different solutions does the 2-machine *n*-job load balancing problem have? $\boxed{?}$





A 2-machine *n*-job load balancing problem is different from subset selection from *n* jobs. In the above tow figures, the selection of {Job 1 and Job 2} is different from the selection of {Job 3} in subset selection (whereas these two solutions are the same in load balancing).

Question 4: A 30-item knapsack problem (to find the best item set of the given 30 items under a capacity constraint). The set of all different solutions includes the selection of no items and the selection of all items. How many different solutions does the 30-item knapsack problem have? $\boxed{?}$



This problem is the same as subset selection.

Question 5: How many items are needed in an *m*-item knapsack problem to have a similar search space size to a 1000-city TSP problem.

Question 6: How many cities are needed in an *n*-city TSP problem to have a similar search space size to a 1000-item knapsack problem.