Global Focus on Knowledge Lecture (The World of Mathematics #8) 2007-06-14

Mathematics of Optimization
-Viewpoint of applied mathematics
Algorithm

Kazuo Murota

Department of Mathematical Engineering and Information Physics (Faculty of Engineering)

Department of Mathematical Informatics (Graduate School of Information Science and Tech.)

http://www.misojiro.t.u-tokyo.ac.jp/»murota

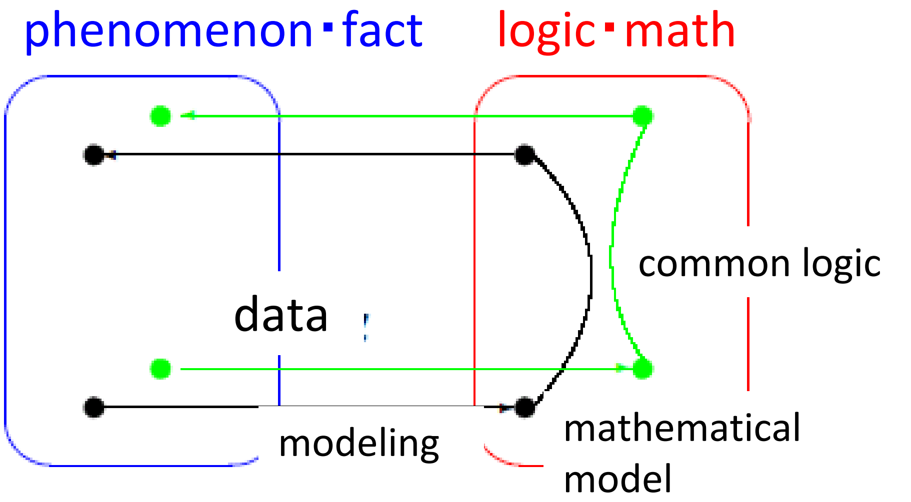
1. Algorithm

2. Calculation of optimization

The World of Optimization

(Revision)

continuity／dispersion
linear／convex／non-linear



Modeling＋ Logics＋ Algorithm

Beautiful *and* Useful

1. Algorithm

Algorithm

Finite and mechanical calculation method

Algorithm ⟵ **Muhammad AL-Khwarizmi**

(c.780-c.850; Arabia)

cf. program

```
int v;
for(v = 1; v <= n; v++){ vfirst[v] = 0; }
for(int a = m; a > 0; a--){
  int v1 = head[2*a - 1];
  adjlist[a] = vfirst[v1];
  vfirst[v1] = a;
  }
```

cf. existence theorem

There are infinitely many prime numbers.

(Proof by contradiction)

Unconstructive Existence Proof (by Contradiction)

There are infinitely many prime numbers.

(a slide by Dr. Katsura)

2. integer $\mathbb{Z}$

Prime number

a natural number that is divisible only by 1 and itself

$$2, 3, 5, 7, 11, 13, 17, 19, 23, \cdots$$

theorem There are infinitely many prime numbers

proof proof by contradiction.

If number of primes is finite,

they are written $p_1, p_2, \cdots, p_m$

and suppose $n = p_1 p_2 \cdots p_m + 1$

n can be divided by a prime

and cannot be divided by $p_1, \cdots, p_m$

repugnance

How primes are generated cannot be presumed from this proof.

Constructive Existence Proof (a Very Easy One)

theorem: The number of even primes is infinite.

proof (by the inductive method):

1) $n = 2$ is an even number.

2) If an integer n is an even number, $n + 2$ is an even number

This proves that the infinite number of even primes can be made.

An Example of an Algorithm in Math

Highest common factor   Euclidean algorithm

3. Euclidean Algorithm

(a slide by Dr. Katsura)

lemma When a, b are 2 integers and not 0

and $a = qb + r \quad (q, r$: integer)

$$\gcd(a,b) = \gcd(b,r)$$

← Property, fact

(static)

Ex.

$$\begin{aligned} 54 &= 2 \times 20 + 14 \\ 20 &= 1 \times 14 + 6 \\ 14 &= 2 \times 6 + 2 \\ 6 &= 3 \times 2 \end{aligned}$$

The highest common factor of 54 and 20 is 2.

← calculation

(dynamic)

An Example of an Algorithm in Math

highest common factor high-speed $\log n$

(Euclidean algorithm)

primality test Sieve of Eratosthenes low-speed $\sqrt{n}$

$$2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10,\ 11,\ 12,\ \cdots$$

construction problem (by a ruler and a compass)

- regular pentagon : possible regular heptagon : impossible
- bisection of an angle : possible trisection of an angle : impossible

finite basic operations | possible/ impossible | high/ low speed

The Logic of Algorithms (1)

algorithm = finite and mechanical calculation

What is a calculation? What can a computer do?

computability (1930s)

equivalence of various calculation models:

Turing machine computability ≡ **recursive function** ≡ λ computability

⇒ Proposition of Church-Turing

Halting problem (an example of non-computability)

The program e and its input value x is given, and

one must judge whether it can be finished in a finite time.

$$\mathrm{Halt}(e,x) = \begin{cases} \text{yes} & \text{(The program stops in a finite time.)} \\ \text{no} & \text{(The program never stops.)} \end{cases}$$

theorem:

There is no algorithm to calculate “Halt”.

Proof

There is No Algorithm for Halt.

$$\mathrm{Halt}(e,x) = \begin{cases} \text{yes} & \text{(It stops in a finite time.)} \\ \text{no} & \text{(It never stops.)} \end{cases}$$

$$f(e) = \begin{cases} 0 & (\mathrm{Halt}(e,e) = \text{no}) \\ \textbf{undefined} & \mathrm{Halt}(e,e) = \text{yes}) \end{cases}$$

There is an algorithm for Halt. $\Rightarrow f$ has an algorithm.

When

f is input to f calculating program…

$\mathrm{Halt}(f,f) = \text{no} \Leftrightarrow f(f) = 0$

$\Leftrightarrow$ f stops for f. $\Leftrightarrow$ $\mathrm{Halt}(f,f) = \text{yes}$

contradiction

(Computablity, Solvability)

possible/ impossible

⇓

high/ low speed

(computational complexity)

Calculation of Algorithm Dealing With Finiteness

sorting problem

input : $7,\ 15,\ 25,\ 27,\ 9,\ 10,\ 13,\ 19,\ 22,\ 2,\ 17,\ 3,\ 5,\ 14$

$$\Downarrow$$

output : $2,\ 3,\ 5,\ 7,\ 9,\ 10,\ 13,\ 14,\ 15,\ 17,\ 19,\ 22,\ 25,\ 27$

algorithm 1 :	test all sequences	$n!$
algorithm 2 :	repeat looking for minimum	n^2
algorithm 3 :	separate, sort and integrate	$n \log n$

complexity of problem — complexity of algorithm

Increases of Computation Time

input size	computation time			
	n	n^2	2^n	$n!$
10	1×10^{-9} sec	1×10^{-8} sec	1×10^{-7} sec	3.6×10^{-4} sec
20	2×10^{-9} sec	4×10^{-8} sec	1×10^{-4} sec	7.7 yr
30	3×10^{-9} sec	9×10^{-8} sec	1.1×10^{-1} sec	8.4×10^{14} yr
40	4×10^{-9} sec	1.6×10^{-7} sec	1.8 min	2.6×10^{30} yr
50	5×10^{-9} sec	2.5×10^{-7} sec	31 hrs	9.6×10^{46} yr
100	1×10^{-8} sec	1×10^{-6} sec	4.0×10^{12} yr	3.0×10^{140} yr
1000	1×10^{-7} sec	1×10^{-4} sec	....	....

Suppose the computer calculates 10^{10} times per second.

polynomial time/ exponential time

The Logic of Algorithms (2)

computational complexity

high/ low speed

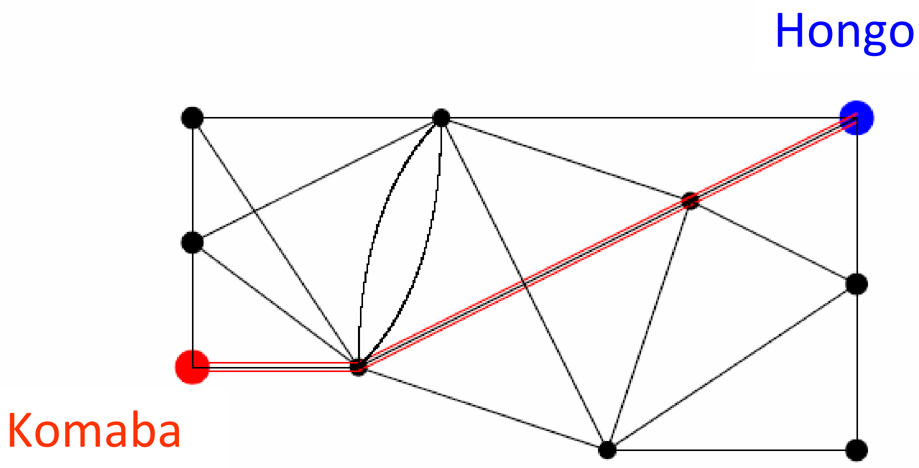
polynomial time/ exponential time

1970 s NP perfectibility (Cook, Levin)

⇒ framework of algorithm construction (target and limit)

Class P vs Class NP

Problem:
Is there a route shorter than 15km between Hongo and Komaba?



P = Polynomial

NP = Nondeterministic Polynomial

発見(Find)

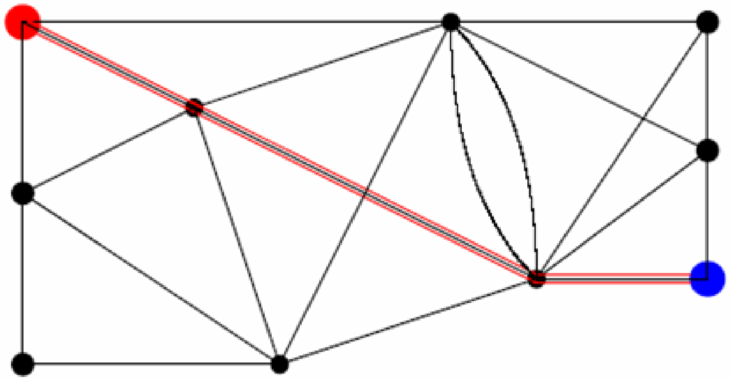
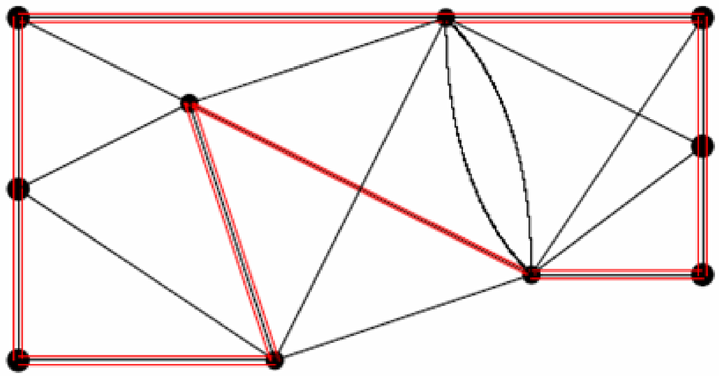
vs

確認(Check)

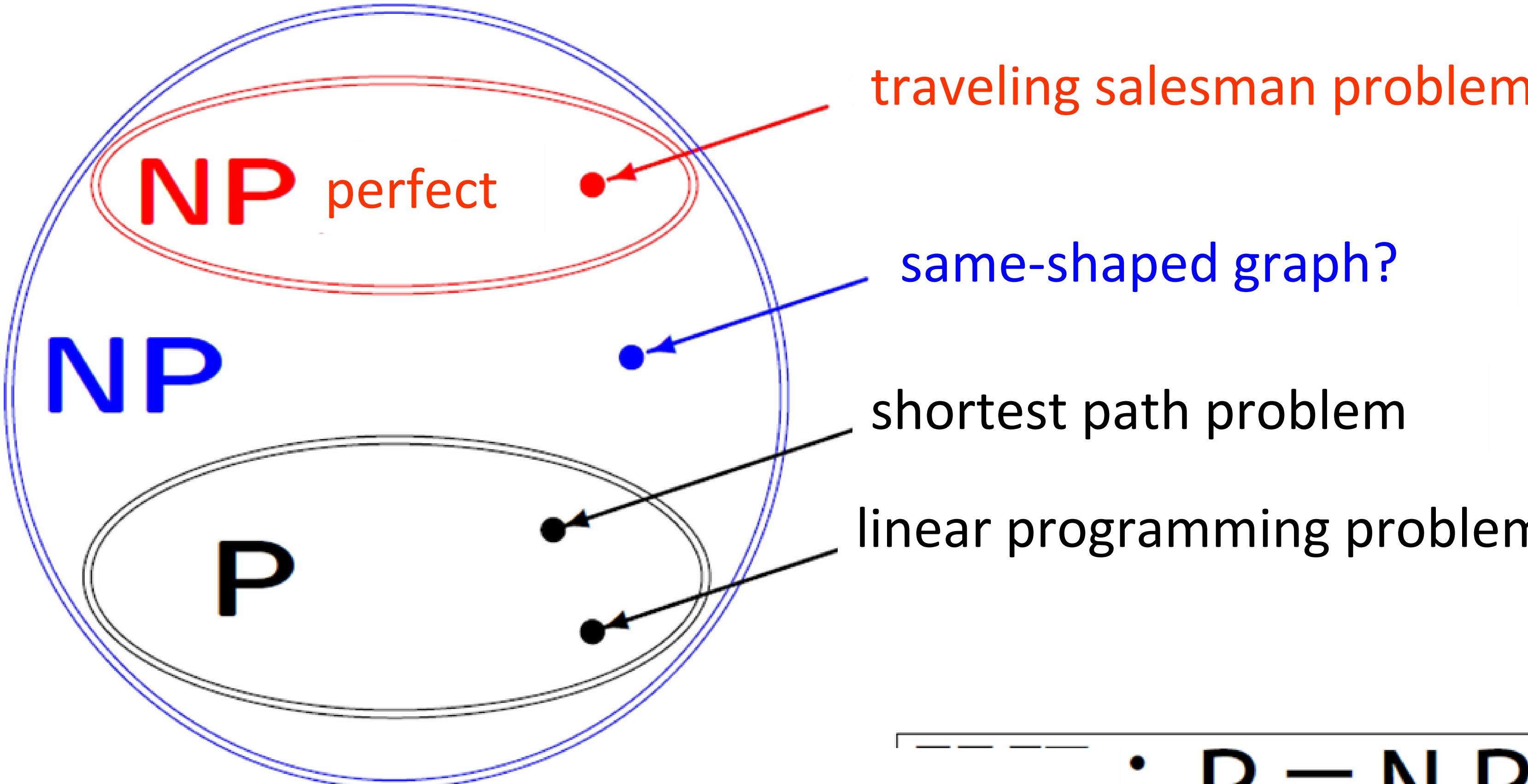
(13.7 km)

Goo map

Problem: Is There a Route Shorter than α ?

between 2 points	traveling salesman
	
easy to check	easy to check
easy to find	difficult to find
P	NP

NP Perfectibility



program : P = N P ?

Clay math institute Millennium Problem (1/7) (¥100 million)

http://www.claymath.org/

The topic is changing from here ....

Development of Calculator

abacus →

← Tiger calculator

The Birth of the Computer

electronic calculator:

1946 ENIAC (J. W. Mauchly, J.P.Eckert)

at the University of Tokyo:

1958 PC-1 (parametron type)

(Hidetoshi Takahashi, Eiichi Goto, Eiichi Wada)

The Progress of Computer (Hardware)

Moore's law : 2-fold / 1.5 yrs

2-fold / 1.5 yrs = 100-fold / 10 yrs = a billion-fold / 40 yrs

Size of a problem that can be solved in a second

# of calculation / sec.	calculating time $T(n)$			
C	n	n^2	2^n	$n!$
10^{10}	10^{10}	10^5	33	13
40 yrs ↓	↓	↓	↓	↓
10^{18}	10^{18}	10^9	60	20

n that satisfies $T(n) = C$

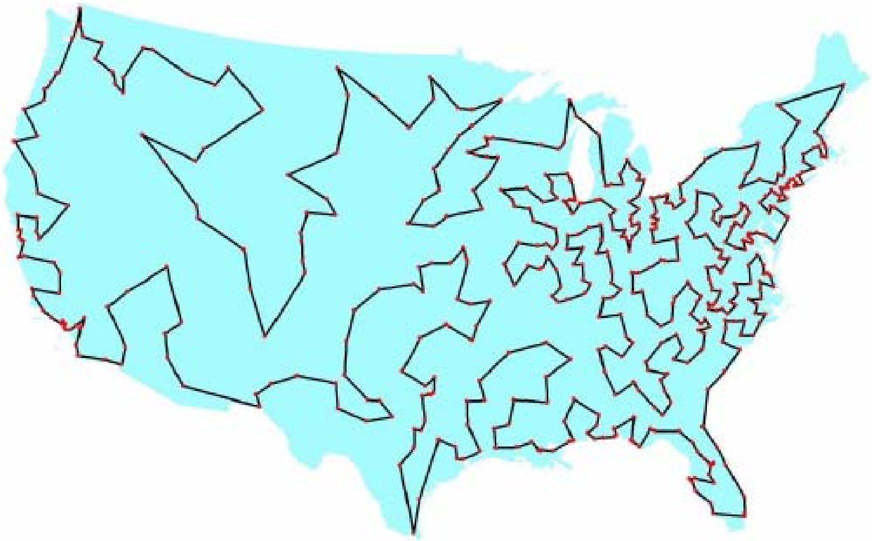
When Calculating Power is c-fold…

- polynomial time $n^2 \longrightarrow$ $n \longrightarrow n \cdot \sqrt{c}$
- exponential time $2^n \longrightarrow$ $n \longrightarrow n + \log c$

lesson 1 : Slow algorithms do not receive the benefit of the hardware's progress

lesson 2 : Difficult problems cannot be solved even if the hardware progresses.

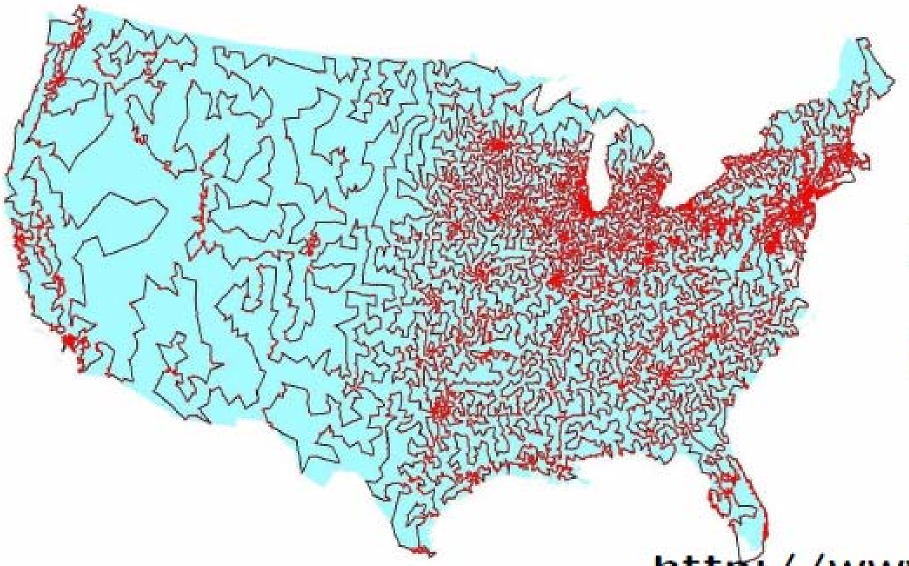
Traveling Salesman Problem (the Progress of Algorithms)



1987 : 532 cities

(M. Padberg–G. Rinaldi)

$n_0 = 532$



1998 : 13,509 cities

(D. Applegate, et al.)

$n_1 = 13509$ (25-fold)

http://www.tsp.gatech.edu/history/pictorial/

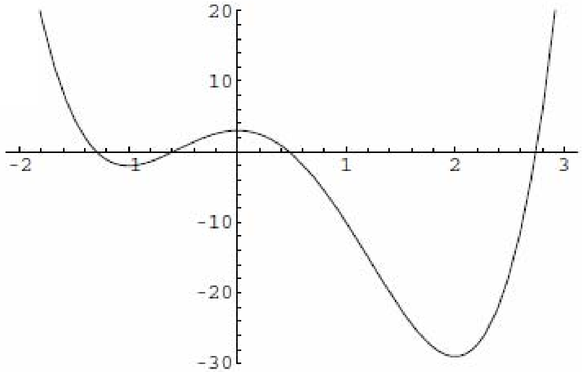
Summary up to here

Progress in environments of optimization computation

- the logic of calculation_(computability, computation amount)
- hardware
- algorithm

2. Calculation of Optimization

Calculation by Formula



$$f(x) = 3x^4 - 4x^3 - 12x^2 + 3$$

$$\Rightarrow \; f'(x) = 12x^3 - 12x^2 - 24x$$

$$= 12x(x+1)(x-2)$$

$$\Rightarrow \; f'(x) = 0 \Leftrightarrow x = 0, \; -1, \; 2$$

$$\Rightarrow \; f(0) = 3, \; f(-1) = -2, \; \boxed{f(2) = -29}$$

$$f(x) = 3x^4 - 4x^3 - 12x^2 + 3 + 0.01x$$

$$\Rightarrow \; f'(x) = 12x^3 - 12x^2 - 24x + 0.01$$

$$= 12x(x+1)(x-2) + 0.01$$

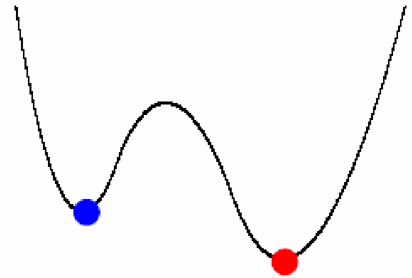
$$\Rightarrow \; f'(x) = 0 \Leftrightarrow x = 0?, \; -1?, \; 2?$$

$$\Rightarrow \; f(0?) = 3?, \; f(-1?) = -2?, \; f(2?) = -29?$$

Do what?

Calculation!

Local Search



S0: initial approximation value x^*

S1: minimize $f(x)$ at x^* 's neighborhood $\Rightarrow x^\bullet$

S2: $f(x^*) \leq f(x^\bullet)$, then stop (x^* is the local best answer)

S3: $x^* = x^\bullet$ Renew and go back to **S1**

Development of Optimization (Metric Variable)

1947 linear planning Dantzig
1960 non-linear planning, Newton method
Powell, Fletcher
1970 convex analysis, dual theorem Rockafellar
1979 ellipsoid method Khachiyan
1984 interior method Karmarkar
1995 semidefinite program
Alizadeh, Nesterov, Nemirovski

logic： linear／convex／non-linear
environment： enhancement of calculation power

Newton Method (Basic Calculation Algorithm)

Taylor series (quadratic approximation):

$$f(x) = f(a) + f'(a)(x-a) + \frac{1}{2} f''(a)(x-a)^2 + \cdots$$
$$\approx \; C \; + \; B \; (x-a) \; + \; A \; (x-a)^2$$

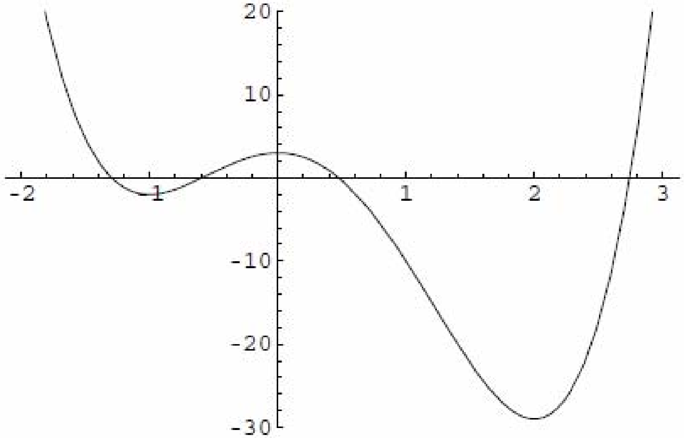
minimization:

$$\Rightarrow \quad x = a - \frac{B}{2A} = a - \frac{f'(a)}{f''(a)}$$

$$\Rightarrow \quad x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

Calculation by Newton Method

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$



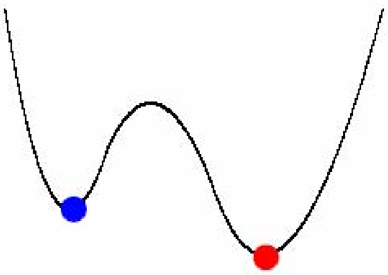
$$f(x) = 3x^4 - 4x^3 - 12x^2 + 3$$

$$\Rightarrow \ x_{k+1} = x_k - \frac{{x_k}^3 - {x_k}^2 - 2x_k}{3{x_k}^2 - 2x_k - 2}$$

k	x_k
0	3.00000
1	2.36842
2	2.07716
3	2.00452

$\Rightarrow$ expansion to polynomial function by Taylor series

Local Search —in discrete optimization—



S0: initial approximation value x^*

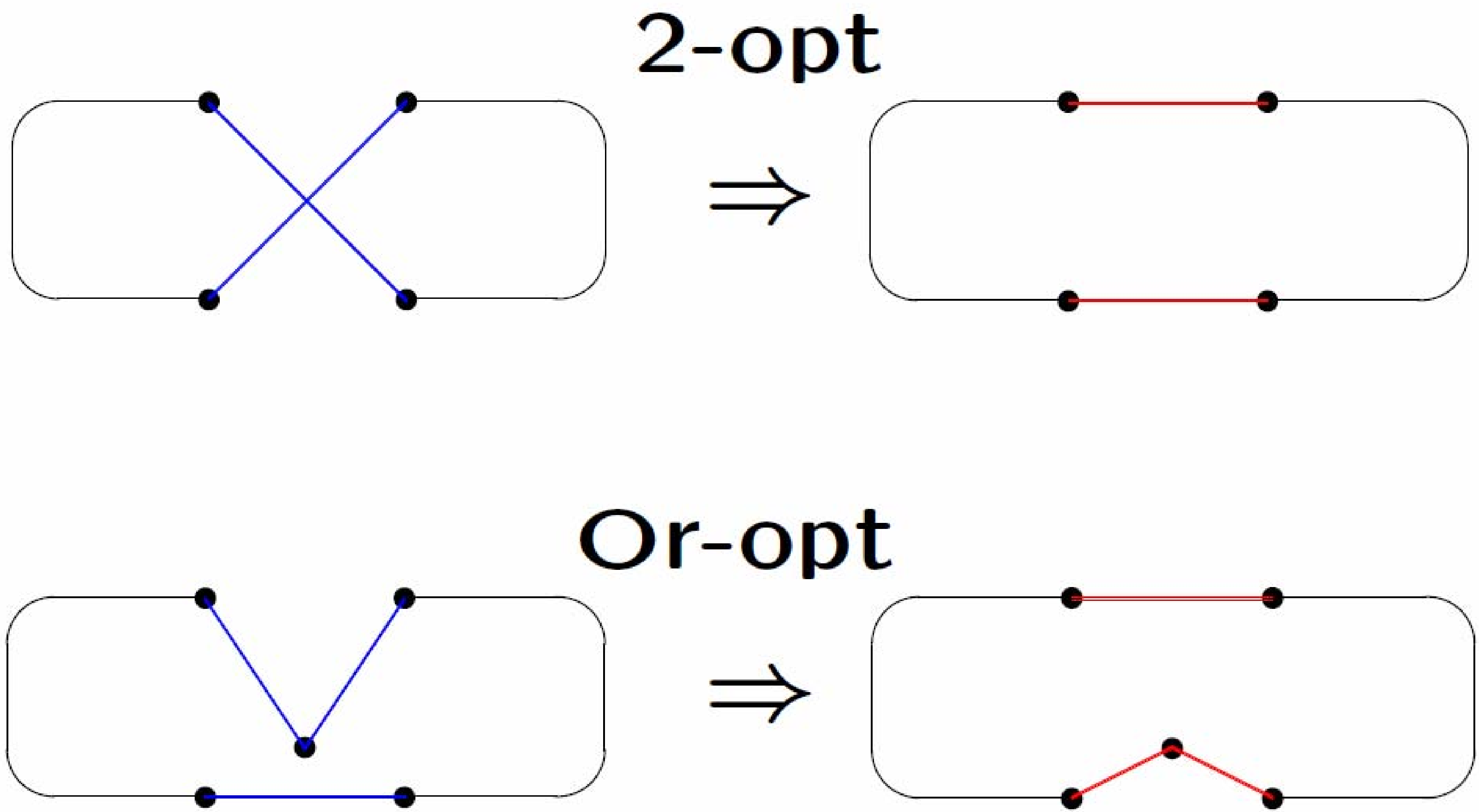
S1: minimize $f(x)$ at x^* 's neighborhood $\Rightarrow x^\bullet$

S2: $f(x^*) \le f(x^\bullet)$, then stop (x^* is the local best answer)

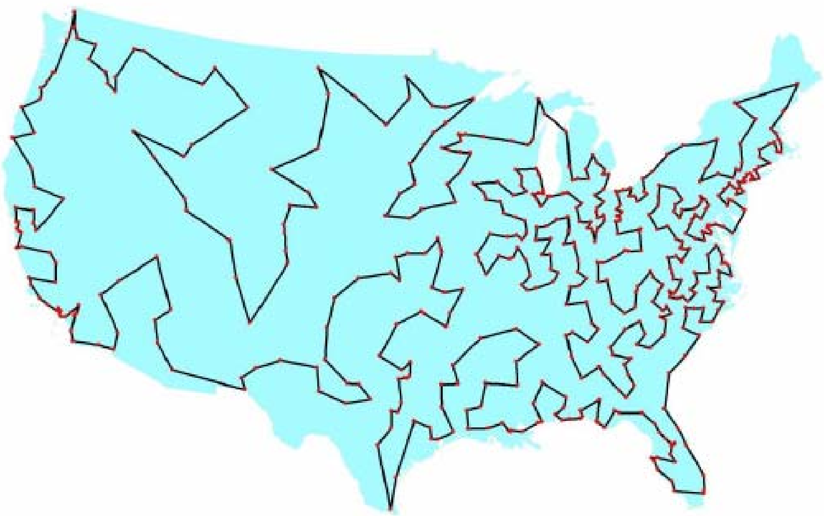
S3: $x^* = x^\bullet$ Renew and go back to **S1**

…Definition of neighborhood is the problem.

“Neighborhood” in Traveling Salesman Problem



Traveling Salesman Problem



532 cities, M. Padberg–G. Rinaldi (1987)

http://www.tsp.gatech.edu/history/pictorial/

$\Longrightarrow$ demonstration by N. Tsuchimura

(Department of Mathematical Engineering)

At last, the Relationship Between the Theory of Computational Complexity and Optimization …

【continuous】 【discrete】 (revision)

1947 —— linear programming ——

1970 convex analysis | polynomiality, NP perfectibility

dual theorem ● submodular function ●

1980 —— ellipsoid method ——

1984 interior method ●●

1995 semidefinite program ●● approximation algorithm ●

2000 discrete convex analysis ●●

Computational complexity (algorithm)

Summary of "the Math of Optimization"

dual theorem

Legendre transformation

linear programming

convex analysis

submodular

local optimization

continuous

discrete

local search

data

model

(AIC)

optimum

design

algorithm

computational theory

computer