Metaheuristic Methods and Their Applications

Lin-Yu Tseng (曾怜玉)

Department of Computer Science
Graduate Institute of Networking and Multimedia
National Chung Hsing University

OUTLINE

I. Optimization Problems

II. Strategies for Solving NP-hard Optimization Problems

III. What is a Metaheuristic ?

IV. Trajectory Methods

V . Population-Based Methods

VI. The Applications of Metaheuristics

VII. Conclusions

I. Optimization Problems

Computer Science

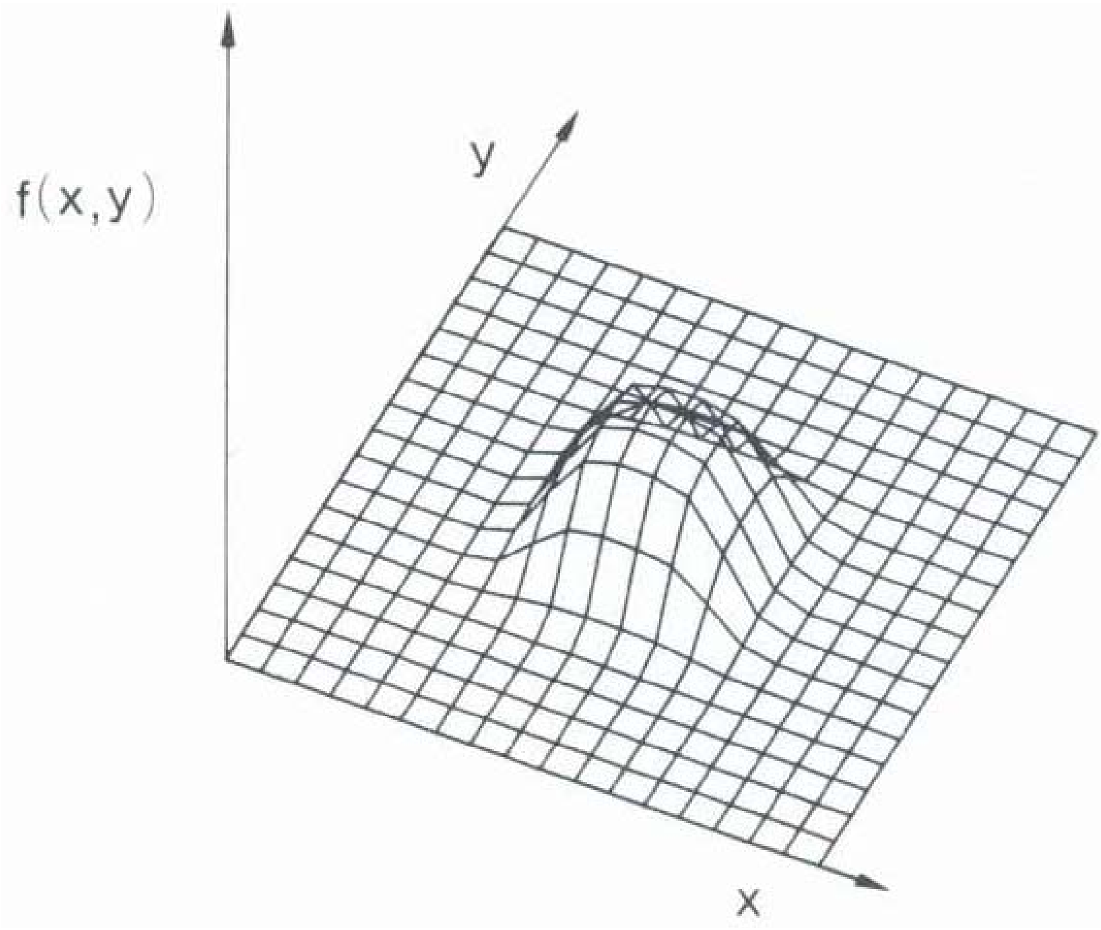
- Traveling Salesman Problem
- Maximum Clique Problem

Operational Research

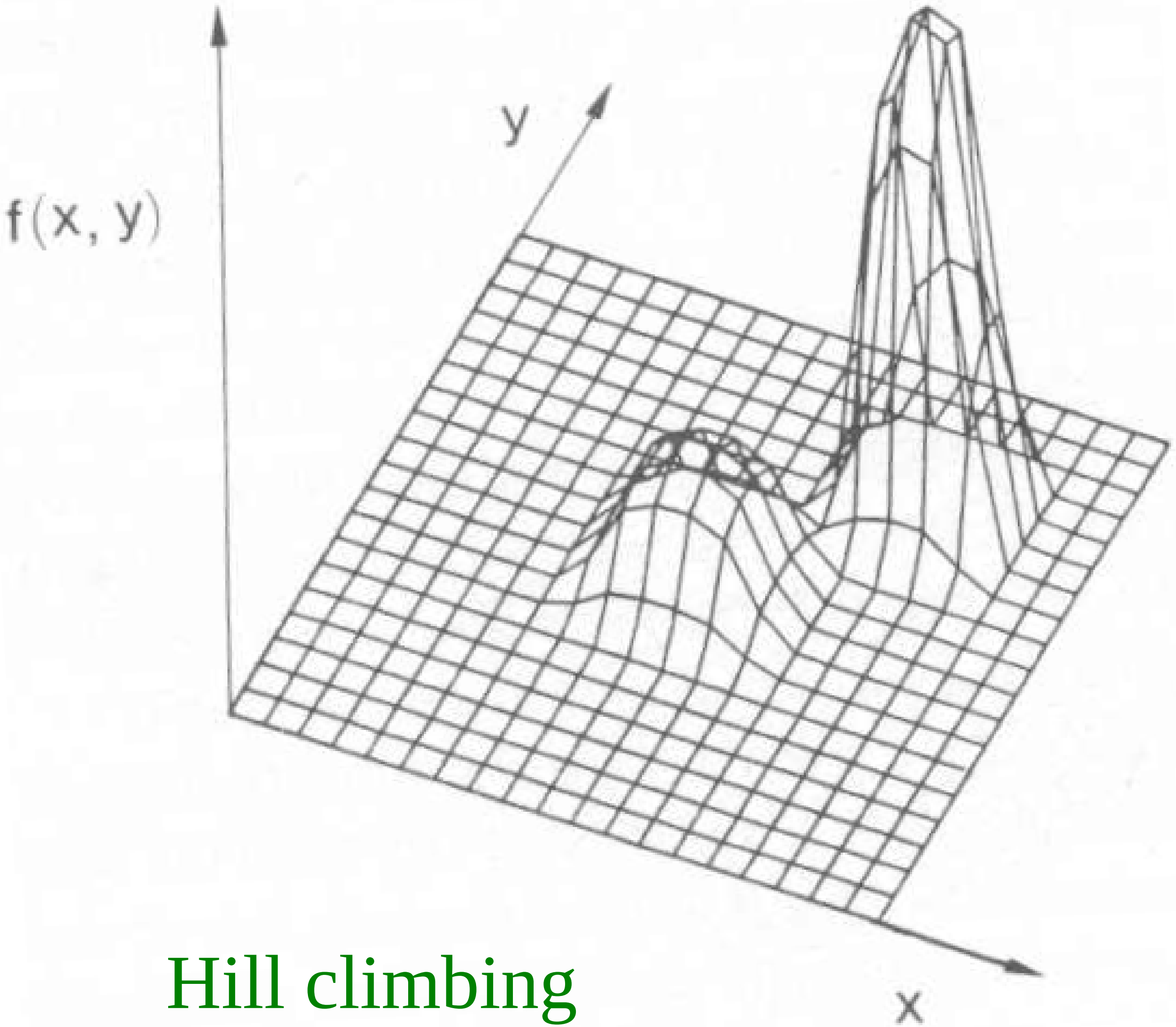
- Flow Shop Scheduling Problem
- P – Median Problem

Many optimization problems are NP-hard.

Optimization problems

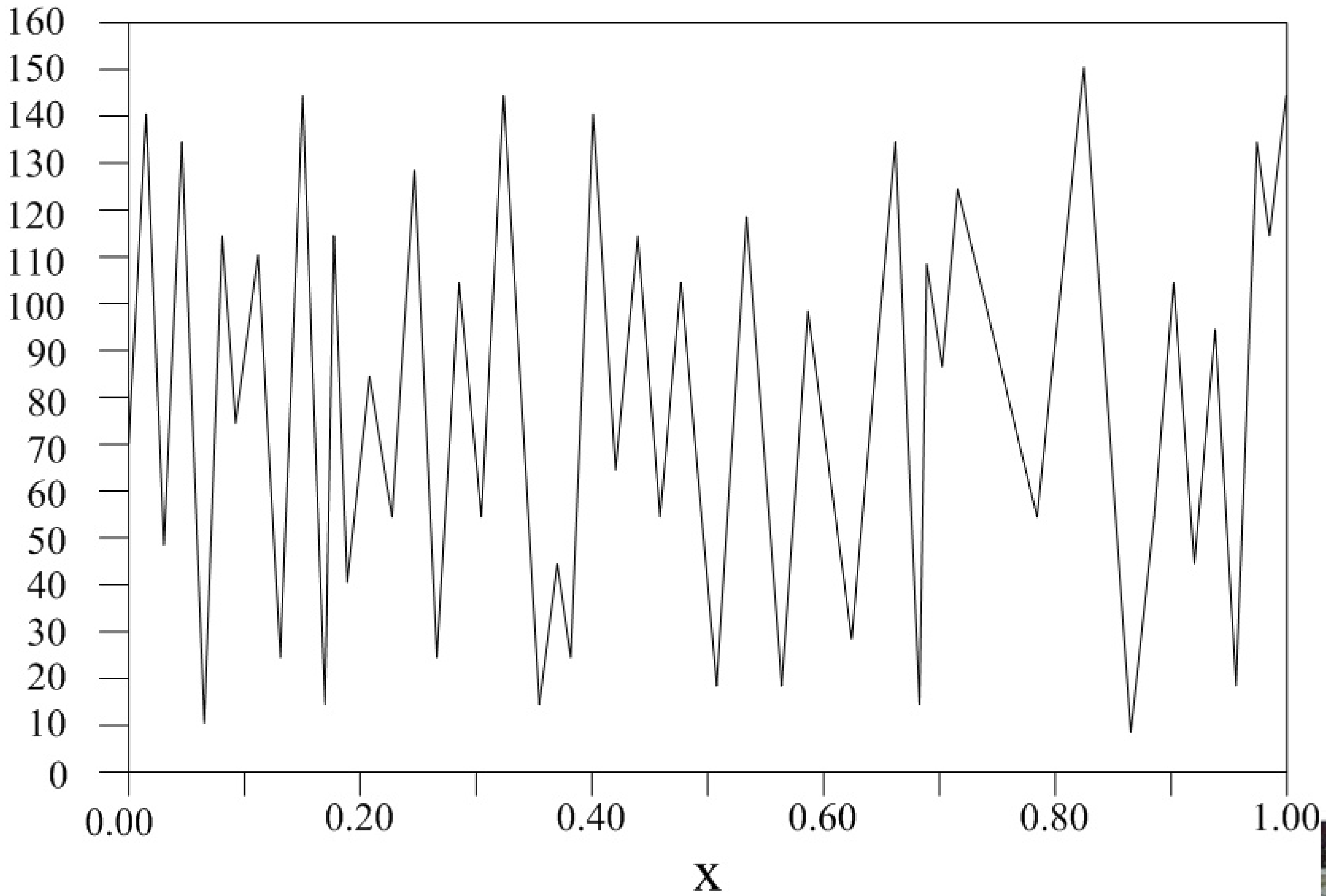


Calculus-based method

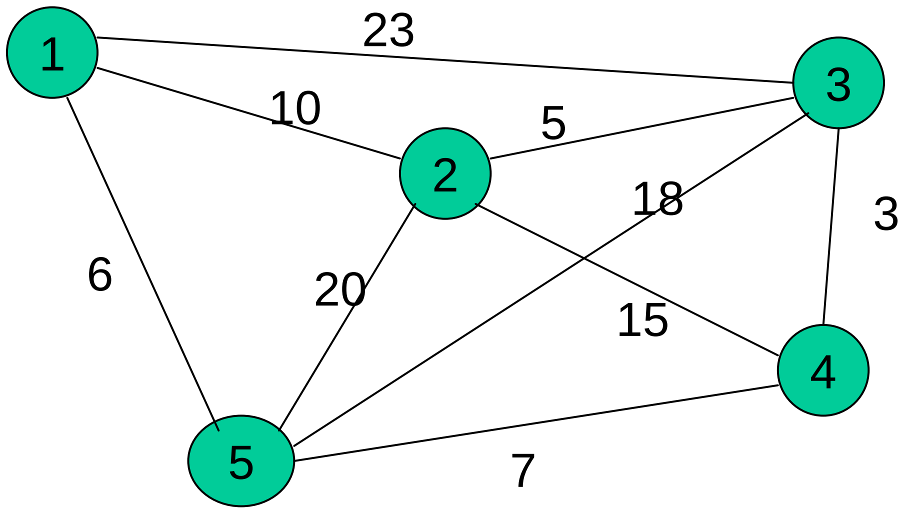


Hill climbing

How about this?



An example : TSP
(Traveling Salesman Problem)



A solution ⟷ A sequence

12345 Tour length = 31

13452 Tour length = 63

There may be (n-1)! tours in total

II. Strategies for Solving NP-hard Optimization Problems

- Branch-and-Bound → Find exact solution
- Approximation Algorithms
  - e.g. There is an approximation algorithm for TSP which can find a tour with tour length $\leqq 1.5 \times$ (optimal tour length) in $O(n^3)$ time.
- Heuristic Methods → Deterministic
- Metaheuristic Methods → Heuristic + Randomization

III. What is a Metaheruistic Method?

- Mata : in an uper level
- Heuristic : to find

A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions. [Osman and Laporte 1996].

Fundamental Properties of Metaheuristics [Blum and Roli 2003]

- Metaheuristics are strategies that “guide” the search process.
- The goal is to efficiently explore the search space in order to find (near-)optimal solutions.
- Techniques which constitute metaheuristic algorithms range from simple local search procedures to complex learning processes.
- Metaheuristic algorithms are approximate and usually non-deterministic.

Fundamental Properties of Metaheuristics (cont.)

- They may incorporate mechanisms to avoid getting trapped in confined areas of the search space.
- The basic concepts of metaheuristics permit an abstract level description.
- Metaheuristics are not problem-specific.
- Metaheuristics may make use of domain-specific knowledge in the form of heuristics that are controlled by the upper level strategy.
- Today’s more advanced metaheuristics use search experience (embodied in some form of memory) to guide the search.

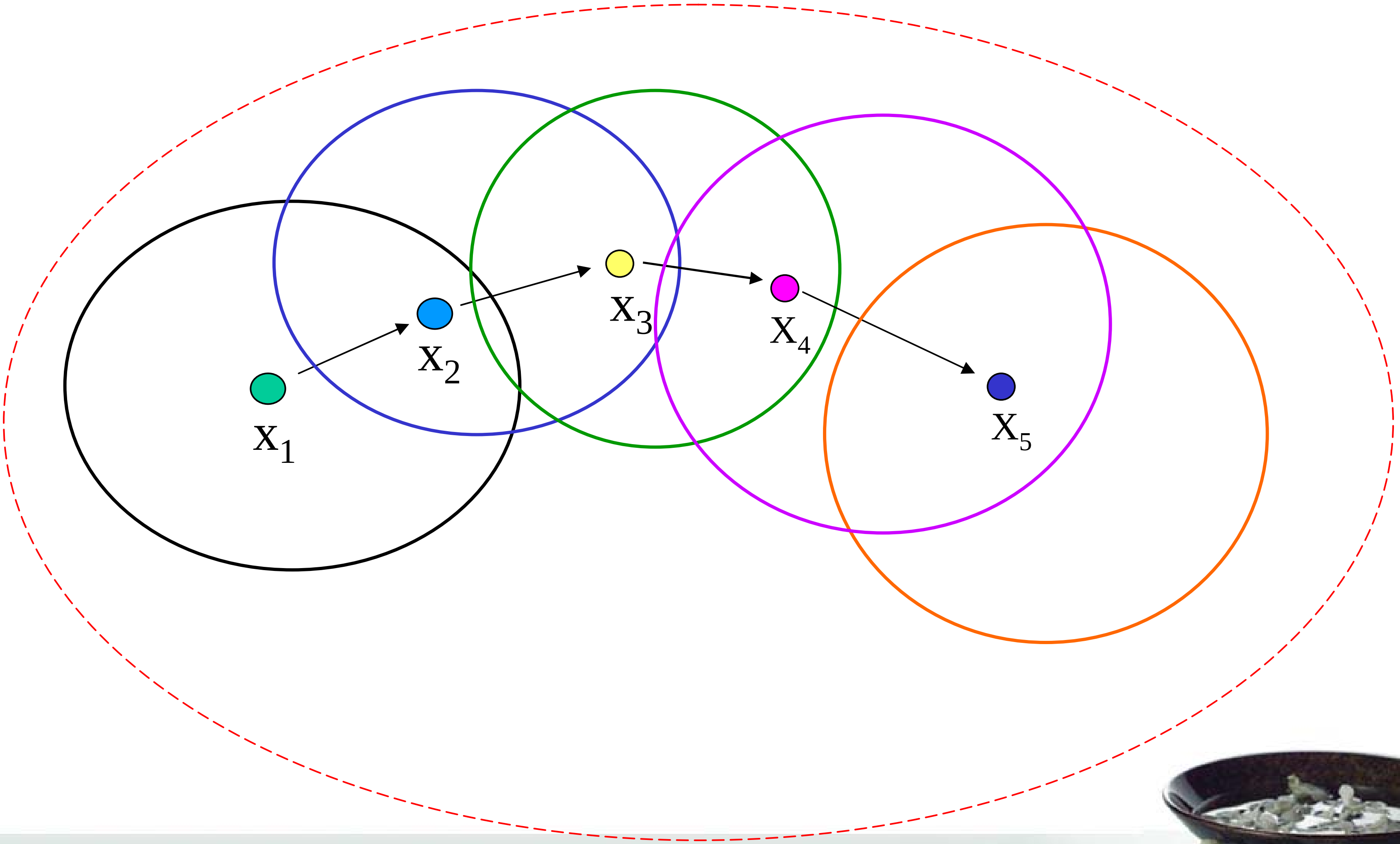
IV. Trajectory Methods

1. Basic Local Search : Iterative Improvement

$s \leftarrow$ GenerateInitialSolution()
repeat
  $s \leftarrow$ Improve($\mathcal{N}(s)$)
until no improvement is possible

- Improve (N(S)) can be
  ① First improvement
  ② Best improvement
  ③ Intermediate option

Trajectory Methods



2. Simulated Annealing – Kirkpatrick 1983

$s \leftarrow$ GenerateInitialSolution()
$T \leftarrow T_0$
while termination conditions not met **do**
  $s' \leftarrow$ PickAtRandom($\mathcal{N}(s)$)
  if $(f(s') < f(s))$ **then**
    $s \leftarrow s'$ % s' replaces s
  else
    Accept s' as new solution with probability $p(T, s', s)$
  endif
  Update(T)
endwhile

- Probability $p(T,s',s) = \exp\left(-\dfrac{f(s') - f(s)}{T}\right)$
- Temperature T may be defined as $T_{k+1} = \alpha T_k,\ 0 < \alpha < 1$
- Random walk + iterative improvement

3. Tabu Search – Glover 1986

- Simple Tabu Search

$s \leftarrow$ GenerateInitialSolution()

$TabuList \leftarrow \emptyset$

while termination conditions not met **do**

  $s \leftarrow$ ChooseBestOf($\mathcal{N}(s) \setminus TabuList$)

  Update($TabuList$)

endwhile

  - Tabu list
  - Tabu tenure: the length of the tabu list
  - Aspiration condition

Tabu Search

$s \leftarrow$ GenerateInitialSolution()
InitializeTabuLists($TL_1, \ldots, TL_r$)
$k \leftarrow 0$
while termination conditions not met **do**
  $AllowedSet(s, k) \leftarrow \{s' \in \mathcal{N}(s) \mid s \text{ does not violate a tabu condition, or it satisfies at least one aspiration condition}\}$
  $s \leftarrow$ ChooseBestOf($AllowedSet(s, k)$)
  UpdateTabuListsAndAspirationConditions()
  $k \leftarrow k + 1$
endwhile

4. Variable Neighborhood Search – Hansen and Mladenović 1999

- Composed of three phase ① shaking ② local search ③ move
- A set of neighborhood structures, $|N_1| < |N_2| < \ldots < |N_{k_{\max}}|$

$N_k^{'}, k = 1, \ldots, k_{\max}^{'}$

$k \leftarrow 1$

$k = k_{\max}^{'}$

x' $x(x' \in N_k^{'}(x))$

x'

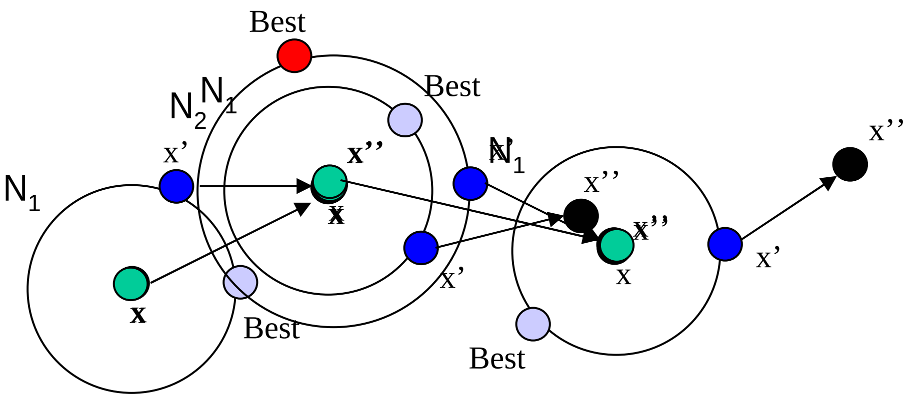
x''

x'' $x \leftarrow x''$ $k \leftarrow 1$

$k \leftarrow k + 1$

Initialization:

Variable Neighborhood Search



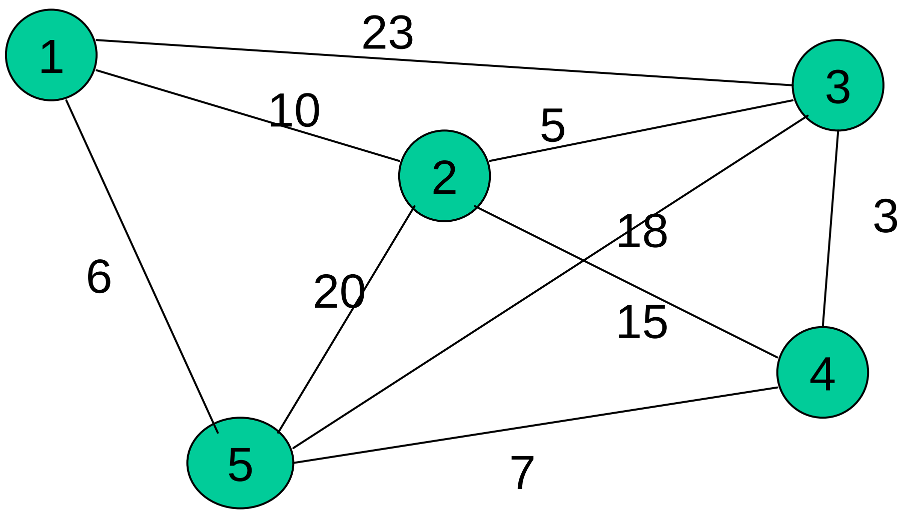
V. Population-Based Methods

1. Genetic Algorithm – Holland 1975

- Coding of a solution --- Chromosome
- Fitness function --- Related to objective function
- Initial population

An example : TSP

(Traveling Salesman Problem)



A solution ⟷ A sequence

12345 Tour length = 31

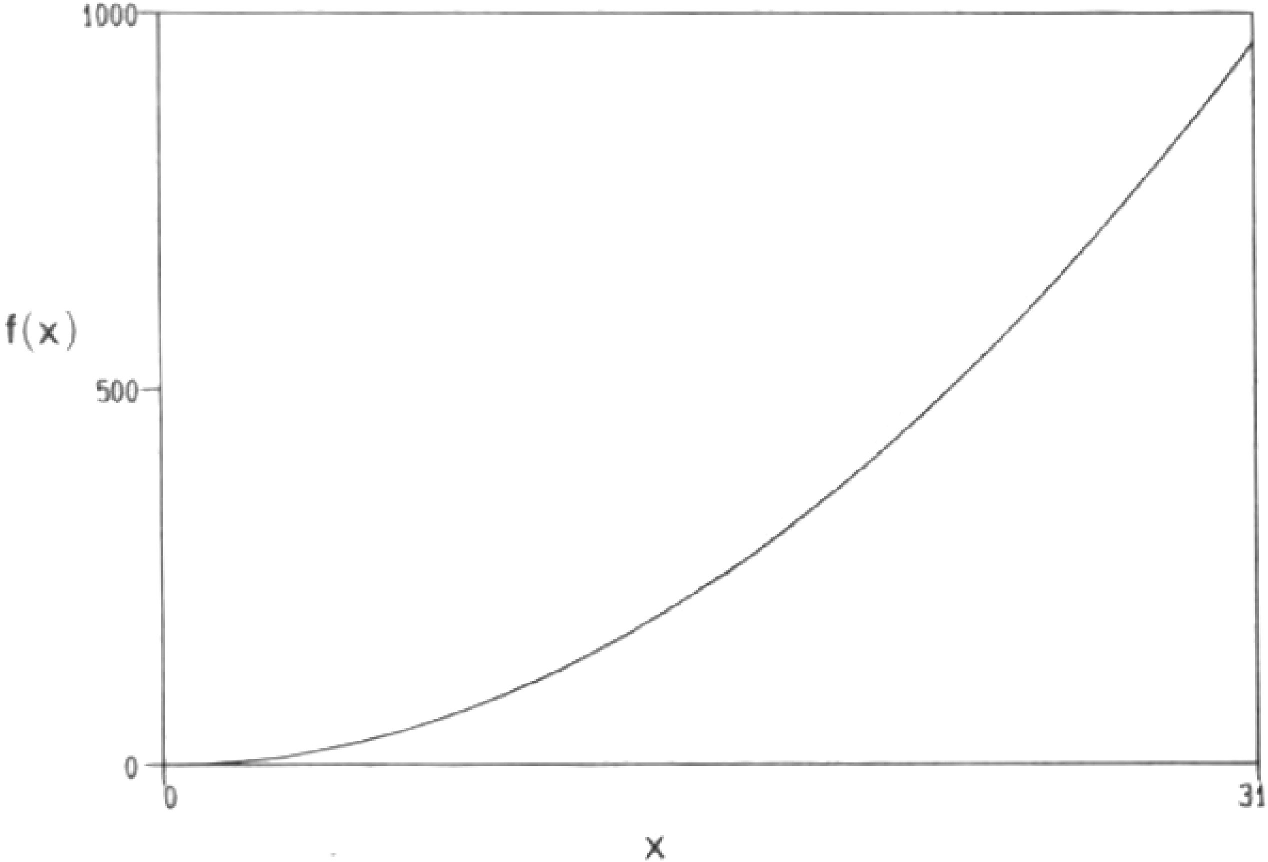
13452 Tour length = 63

Genetic Algorithm

- Reproduction (Selection)
- Crossover
- Mutation

Example 1

Maximize $f(x) = x^2$ where $x \in I$ and $0 \le x \le 31$



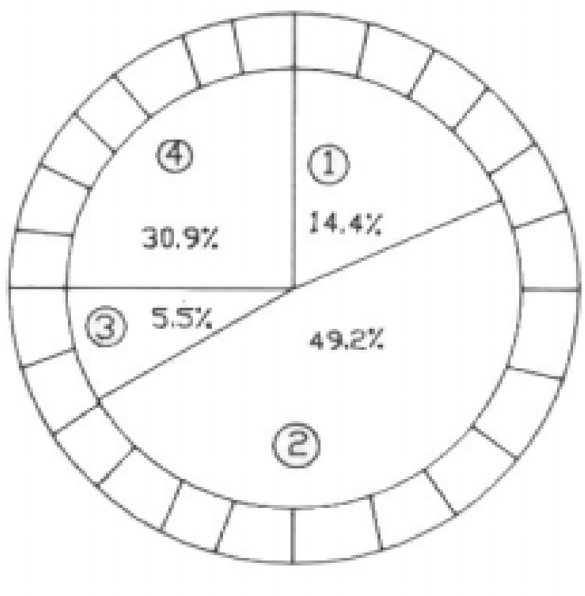
1. Coding of a solution : A five-bit integer, e.g. 01101
2. Fitness function : $F(x) = f(x) = x^2$
3. Initial population : (Randomly generated)
   01101
   11000
   01000
   10011

Reproduction

TABLE 1.1 Sample Problem Strings and Fitness Values

No.	String	Fitness	% of Total
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9
Total		1170	100.0

Roulette Wheel



Reproduction

TABLE 1.2 A Genetic Algorithm by Hand

String No.	Initial Population (Randomly Generated)	x Value (Unsigned Integer)	$f(x)$ x^2	pselect, $\frac{f_i}{\Sigma f}$	Expected count $\frac{f_i}{\bar{f}}$	Actual Count from Roulette Wheel
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4.0
Average			293	0.25	1.00	1.0
Max			576	0.49	1.97	2.0

Crossover

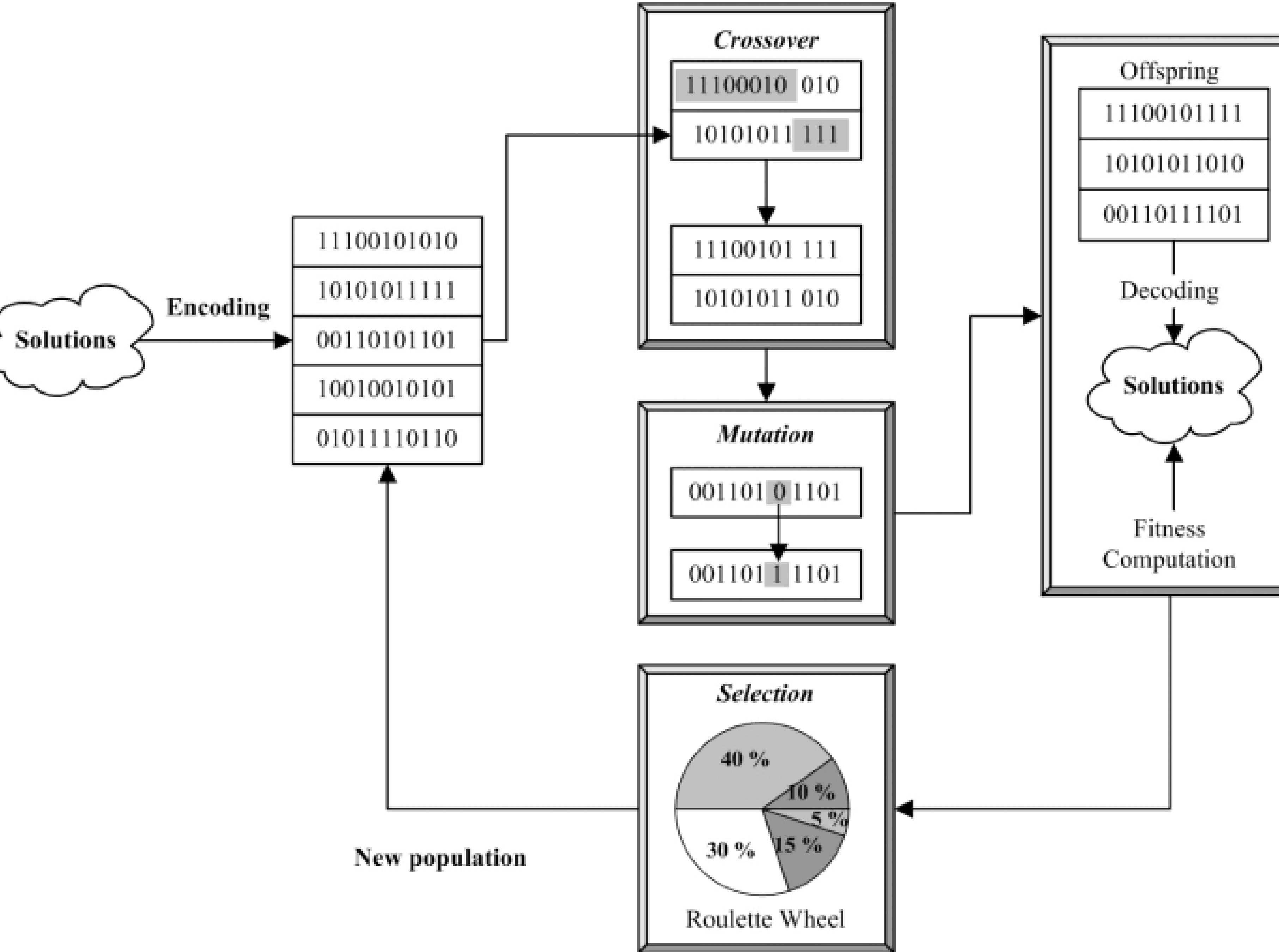
Mating Pool after Reproduction (Cross Site Shown)	Mate (Randomly Selected)	Crossover Site (Randomly Selected)	New Population	x Value	$f(x)$ x^2
0 1 1 0 | 1	2	4	0 1 1 0 0	12	144
1 1 0 0 | 0	1	4	1 1 0 0 1	25	625
1 1 | 0 0 0	4	2	1 1 0 1 1	27	729
1 0 | 0 1 1	3	2	1 0 0 0 0	16	256
					1754
					439
					729

Mutation

The probability of mutation

$Pm = 0.001$

$20 \text{ bits} * 0.001 = 0.02 \text{ bits}$



Solutions
Encoding
11100101010
10101011111
00110101101
10010010101
01011110110
Crossover
11100010 010
10101011 111
11100101 111
10101011 010
Mutation
001101 0 1101
001101 1 1101
Offspring
11100101111
10101011010
00110111101
Decoding
Solutions
Fitness
Computation
Selection
40 %
10 %
5 %
15 %
30 %
Roulette Wheel
New population

Example 2 Word matching problem

- “to be or not to be” $\rightarrow$ “tobeornottobe”
- $(1/26)^{13} = 4.03038 \times 10^{-19}$
- The lowercase letters in ASCII are represented by numbers in the range [97, 122]

[116,111,98,101,114,110, 116, 116, 111, 98,101]

Initial population

[114, 122, 102, 113, 100, 104, 117, 106, 97, 114, 100, 98, 101]
[110, 105, 101, 100, 119, 118, 121, 118, 106, 97, 104, 102, 106]
[115, 99, 121, 117, 101, 105, 115, 111, 115, 113, 118, 99, 98]
[102, 98, 102, 118, 114, 97, 109, 116, 101, 107, 117, 118, 115]
[107, 98, 117, 113, 114, 116, 106, 116, 106, 101, 110, 115, 98]
[102, 119, 121, 113, 121, 107, 107, 116, 122, 121, 111, 106, 104]
[116, 98, 120, 98, 108, 115, 111, 105, 122, 103, 103, 119, 109]
[101, 111, 111, 117, 114, 104, 100, 120, 98, 118, 116, 120, 97]
[100, 116, 114, 105, 117, 111, 115, 114, 103, 107, 109, 98, 103]
[106, 118, 112, 98, 103, 101, 109, 116, 112, 106, 97, 108, 113]

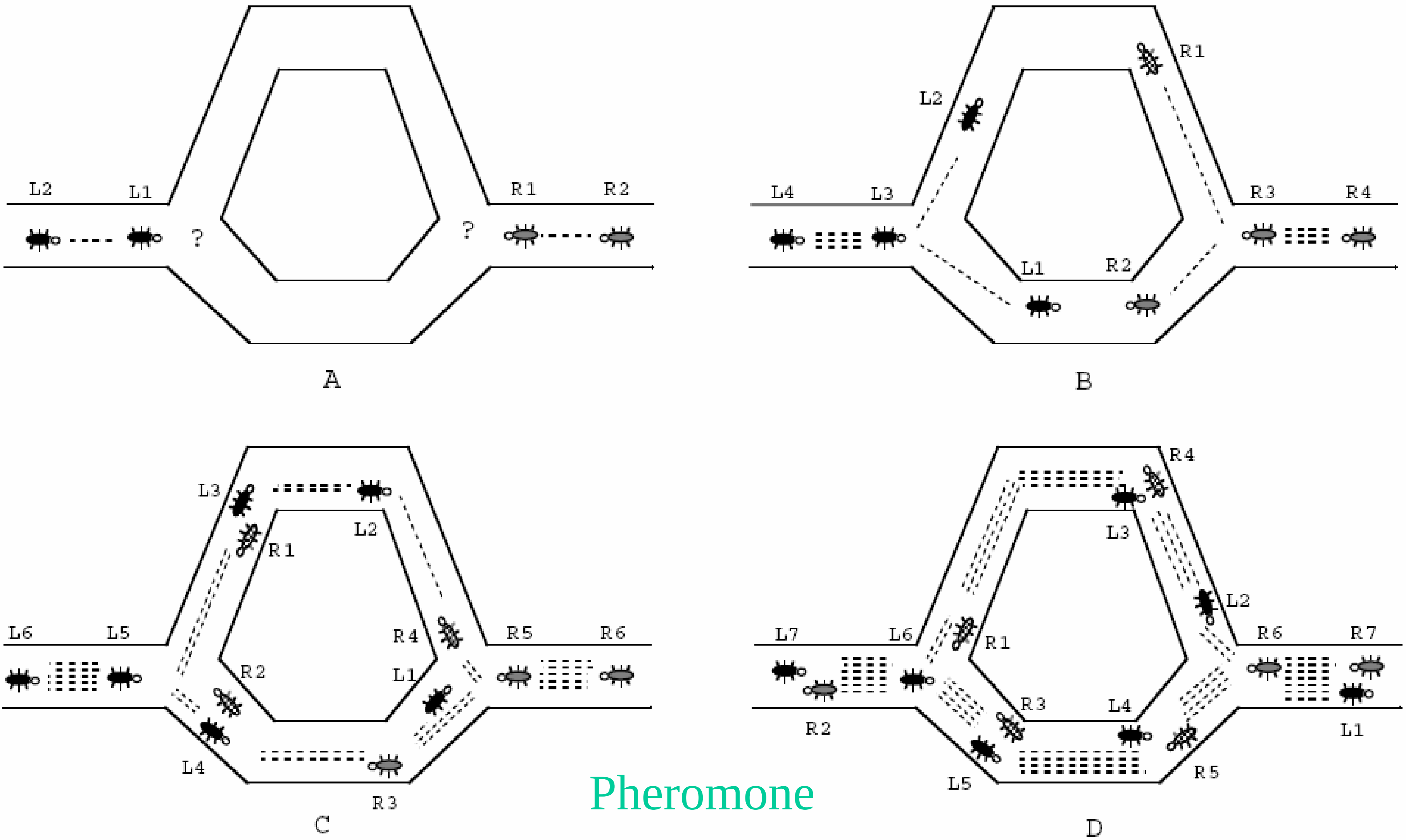
The corresponding strings

rzfqdhujardbe
niedwrvrjahfj
cyueisosqvcb
fbfvgramtekuvs
kbuqrtjtjensb
fwyqykktzyojh
tbxblsoizggwm
dtriusrgkmbg
jvpbgemtpjalq

Table 1.2 The Best String for Each Generation

Generation	String	Fitness	Generation	String	Fitness
1	rzfqdhujardbe	2	16	rzbwornottobe	10
2	rzfqdhuoardbe	3	17	rzbwornottobe	10
3	rzfqghuoatdbe	4	18	rzbwornottobe	10
4	rzfqghuoztobe	5	19	rzbwornottobe	10
5	rzfqghhottobe	6	20	robwornottobe	11
6	rzfqohhottobe	7	21	tobwornottobe	12
7	rzfqohnottobe	8	22	tobwornottobe	12
8	rzfqohnottobe	8	23	tobeornottobe	13
9	rzfqohnottobe	8	24	tobeornottobe	13
10	rzfqohnottobe	8	25	tobeornottobe	13
11	rzfqornottobe	9	26	tobeornottobe	13
12	rzfqornottobe	9	27	tobeornottobe	13
13	rwfwornottobe	9	28	tobeornottobe	13
14	rwcwornottobe	9	29	tobeornottobe	13
15	rzcwornottobe	9	30	tobeornottobe	13

2. Ant Colony Optimization – Dorigo 1992



Initialization

Loop

Loop

Each ant applies a state transition rule to incrementally build a solution and applies a local updating rule to the pheromone

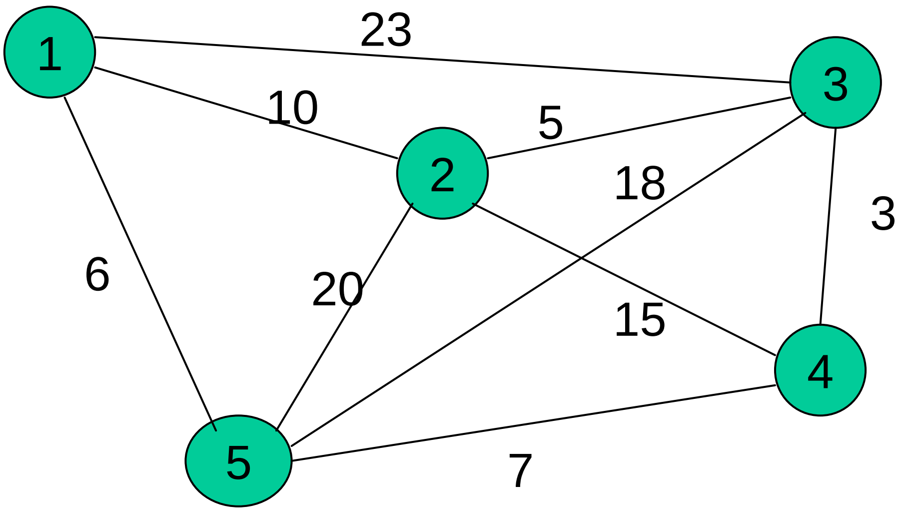
Until each of all ants has built a complete solution

A global pheromone updating rule is applied

Until End_Condition

Example : TSP

A simple heuristics – a greedy method: choose the shortest edge to go out of a node



Solution : 15432

- Each ant builds a solution using a step by step constructive decision policy.
- How ant k choose the next city to visit?

$$s = \begin{cases} \arg \quad \max_{u \in J_r} \{\tau_{ru} \cdot (\eta_{ru})^{\beta}\} & if \quad q \le q_0 \\ S, \quad otherwise & \end{cases}$$

Where $\eta = 1/\delta$, δ_{rs} be a distance measure associated with edge (r,s)

$$p_{rs}^{k} = \begin{cases} \dfrac{\tau_{rs} \cdot (\eta_{rs})^{\beta}}{\sum\limits_{u \in J_r} \tau_{ru} \cdot (\eta_{ru})^{\beta}} & if \quad s \in J_r \\ 0 \quad\quad if & s \notin J_r \end{cases}$$

Local update of pheromone

$$\tau_{rs} \leftarrow (1-\rho)\tau_{rs} + \rho \cdot \Delta\tau_{rs}, \qquad 0 < \rho < 1$$

where $\Delta\tau_{rs} = \tau_0$, τ_0 is the initial pheromone level

Global update of pheromone

$$\tau_{rs} \leftarrow (1-\alpha)\tau_{rs} + \alpha \cdot \Delta\tau_{rs}, \qquad 0 < \alpha < 1$$

where $\Delta\tau_{rs} = \begin{cases} 1/L_{gb}, & if\ (r,s) \in \\ 0, & otherwise \end{cases}$ globally best tour

3. Particle Swarm Optimization – Kennedy and Eberhart 1995

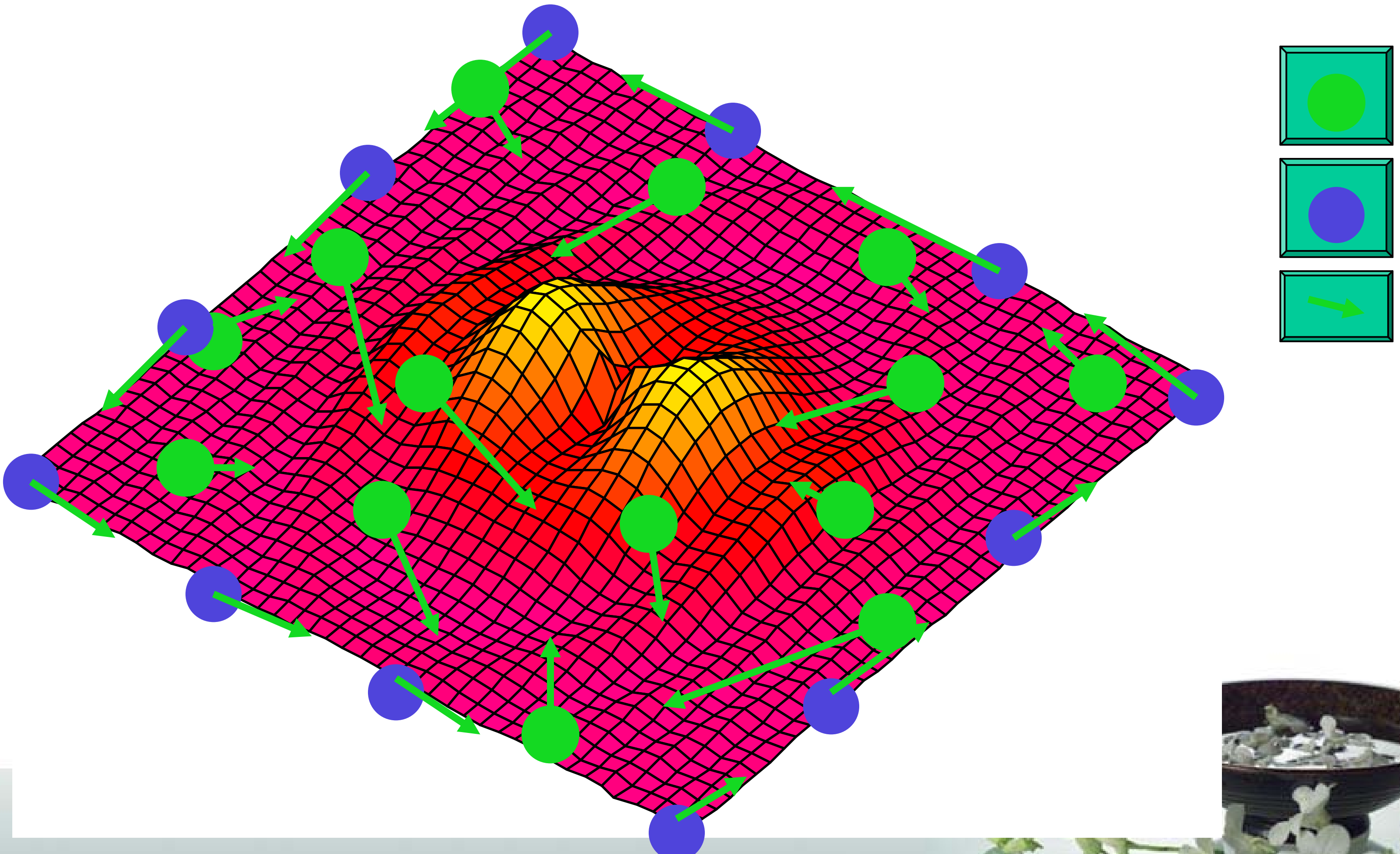
- Population initialized by assigning random **positions** *and* **velocities**; potential solutions are then *flown* through hyperspace.
- Each particle keeps track of its “best” (highest fitness) position in hyperspace.
  - This is called “**pBest**” for an individual particle.
  - It is called “**gBest**” for the best in the population.
- At each time step, each particle stochastically accelerates toward its **pBest** and **gBest**.

Particle Swarm Optimization Process

1. Initialize population in hyperspace.
   - Includes position and velocity.
2. Evaluate fitness of individual particle.
3. Modify velocities based on personal best and global best.
4. Terminate on some condition.
5. Go to step 2.

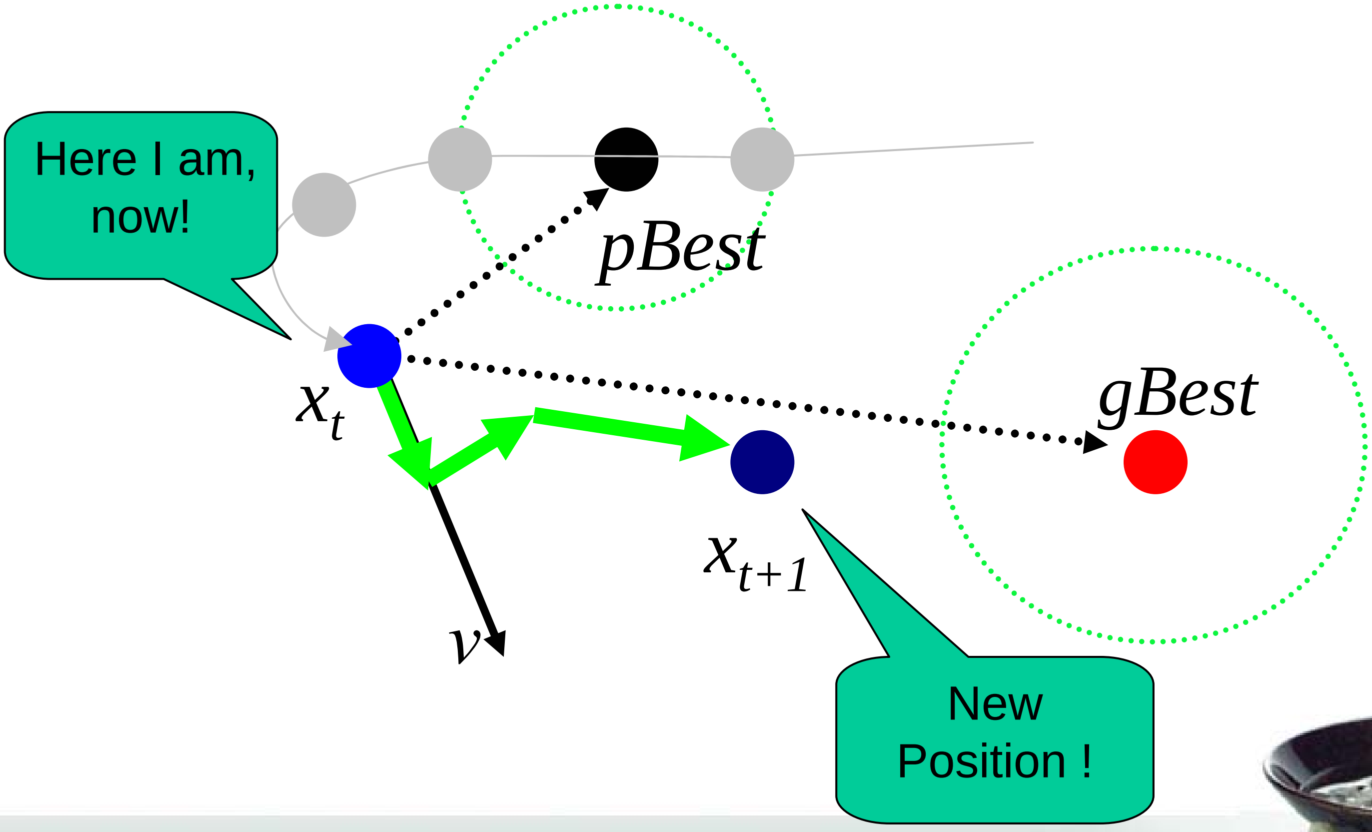
Initialization:

- Positions and velocities



Modify velocities

- based on personal best and global best.



Particle Swarm Optimization

Modification of velocities and positions

$$v_{t+1} = v_t + c_1 * rand() * (pBest - x_t) + c_2 * rand() * (gBest - x_t)$$

$$x_{t+1} = x_t + v_{t+1}$$

VI. The Applications of Metaheuristics

1. Solving NP-hard Optimization Problems
   - Traveling Salesman Problem
   - Maximum Clique Problem
   - Flow Shop Scheduling Problem
   - P-Median Problem
2. Search Problems in Many Applications
   - Feature Selection in Pattern Recognition
   - Automatic Clustering
   - Machine Learning (e.g. Neural Network Training)

VII. Conclusions

- For NP-hard optimization problems and complicated search problems, metaheuristic methods are very good choices for solving these problems.
- More efficient than branch-and-bound.
- Obtain better quality solutions than heuristic methods.
- Hybridization of metaheuristics
- How to make the search more systematic ?
- How to make the search more controllable ?
- How to make the performance scalable?

References

1. Osman, I.H., and Laporte,G. “Metaheuristics:A bibliography”. *Ann. Oper. Res. 63*, 513–623, 1996.
2. Blum, C., and Andrea R. “Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison”. *ACM Computing Surveys*, *35(3),* 268–308, 2003.
3. Kirkpatrick, S., Gelatt. C. D., and Vecchi, M. P. “Optimization by simulated annealing”, *Science, 13 May 1983 220*, 4598, 671–680, 1983.
4. Glover, F. “Future paths for integer programming and links to artificial intelligence”, *Comput. Oper. Res. 13*, 533–549, 1986.
5. Hansen, P. and Mladenović, N. An introduction to variable neighborhood search. In *Metaheuristics: Advances and trends in local search paradigms for optimization*, S. Voß, S. Martello, I. Osman, and C. Roucairol, Eds. Kluwer Academic Publishers, Chapter 30, 433–458, 1999.
6. Holland, J. H. *Adaption in natural and artificial systems*. The University of Michigan Press,Ann Harbor, MI. 1975.
7. Dorigo, M. Optimization, learning and natural algorithms *(in italian)*. Ph.D. thesis, DEI, Politecnico di Milano, Italy. pp. 140, 1992.
8. Kennedy, J. and Eberhart, R. “Particle Swarm Optimization”, *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pp. 1942-1948, IEEE Press, 1995.

謝謝！

祝各位 身體健康

心情愉快！

並祝 大會圓滿成功！