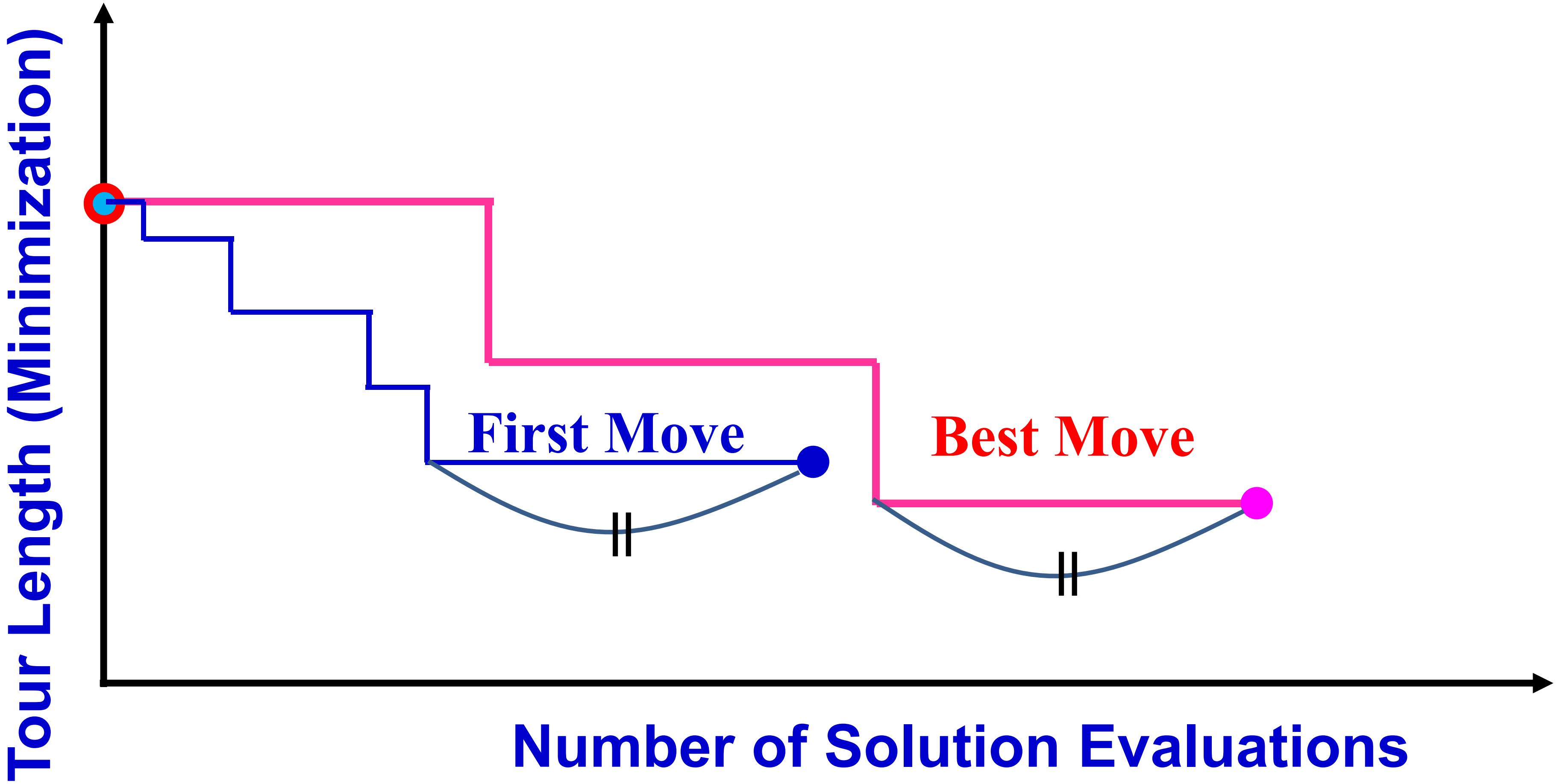
Optimization Methods

1. Introduction.
2. Greedy algorithms for combinatorial optimization.
3. LS and neighborhood structures for combinatorial optimization.
4. Variable neighborhood search, neighborhood descent, SA, TS.
5. Branch and bound algorithms, and subset selection algorithms.
6. Linear programming problem formulations and applications.
7. Linear programming algorithms.
8. Integer linear programming algorithms.
9. Unconstrained nonlinear optimization and gradient descent.
10. Newton's methods and Levenberg-Marquardt modification.
11. Quasi-Newton methods and conjugate direction methods.
12. Nonlinear optimization with equality constraints.
13. Nonlinear optimization with inequality constraints.
14. Problem formulation and concepts in multi-objective optimization.
15. Search for single final solution in multi-objective optimization.
16: Search for multiple solutions in multi-objective optimization.

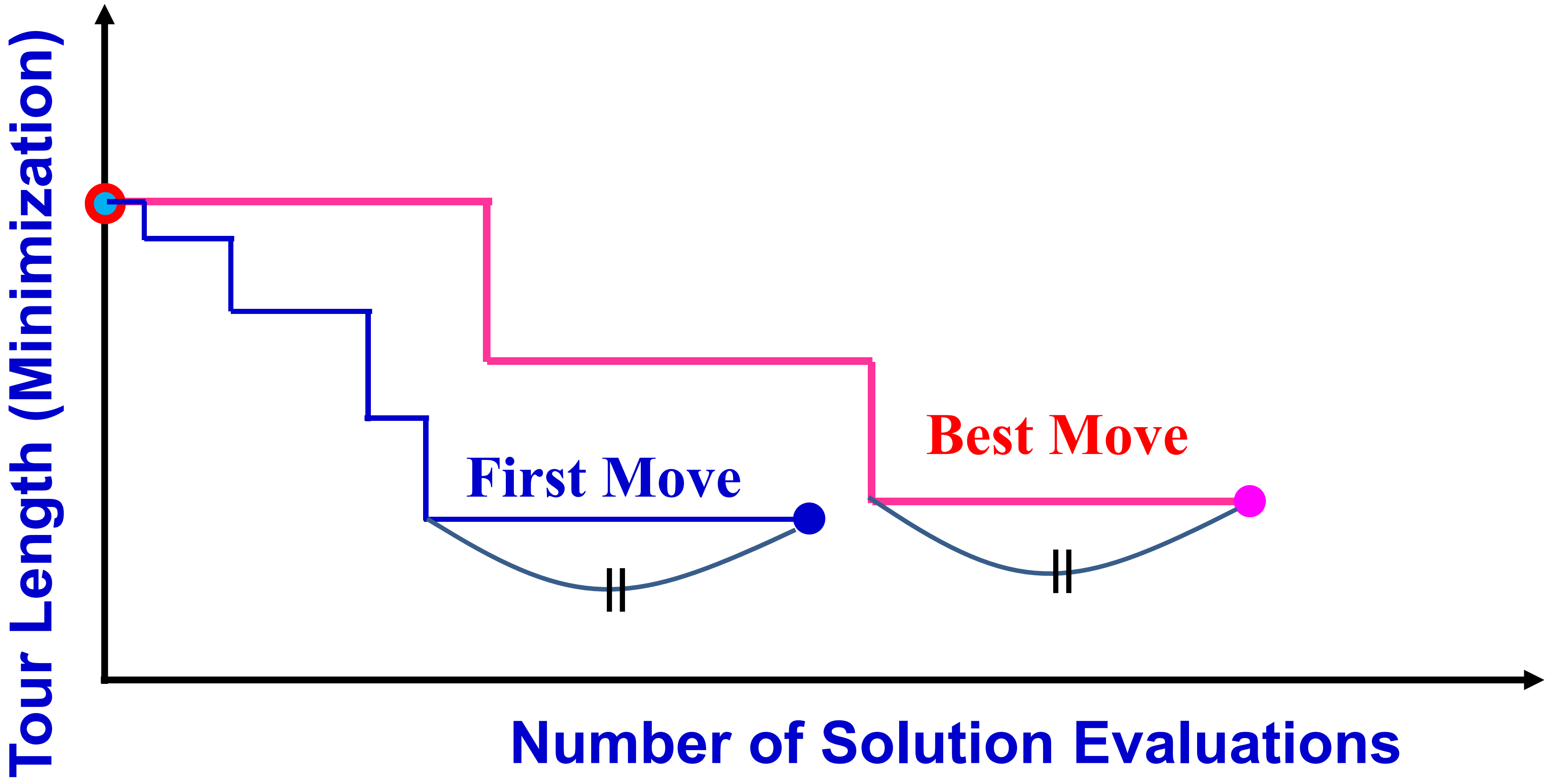
Best Move and First Move: Which is better ?



(1) In general, the first move needs less computation load.

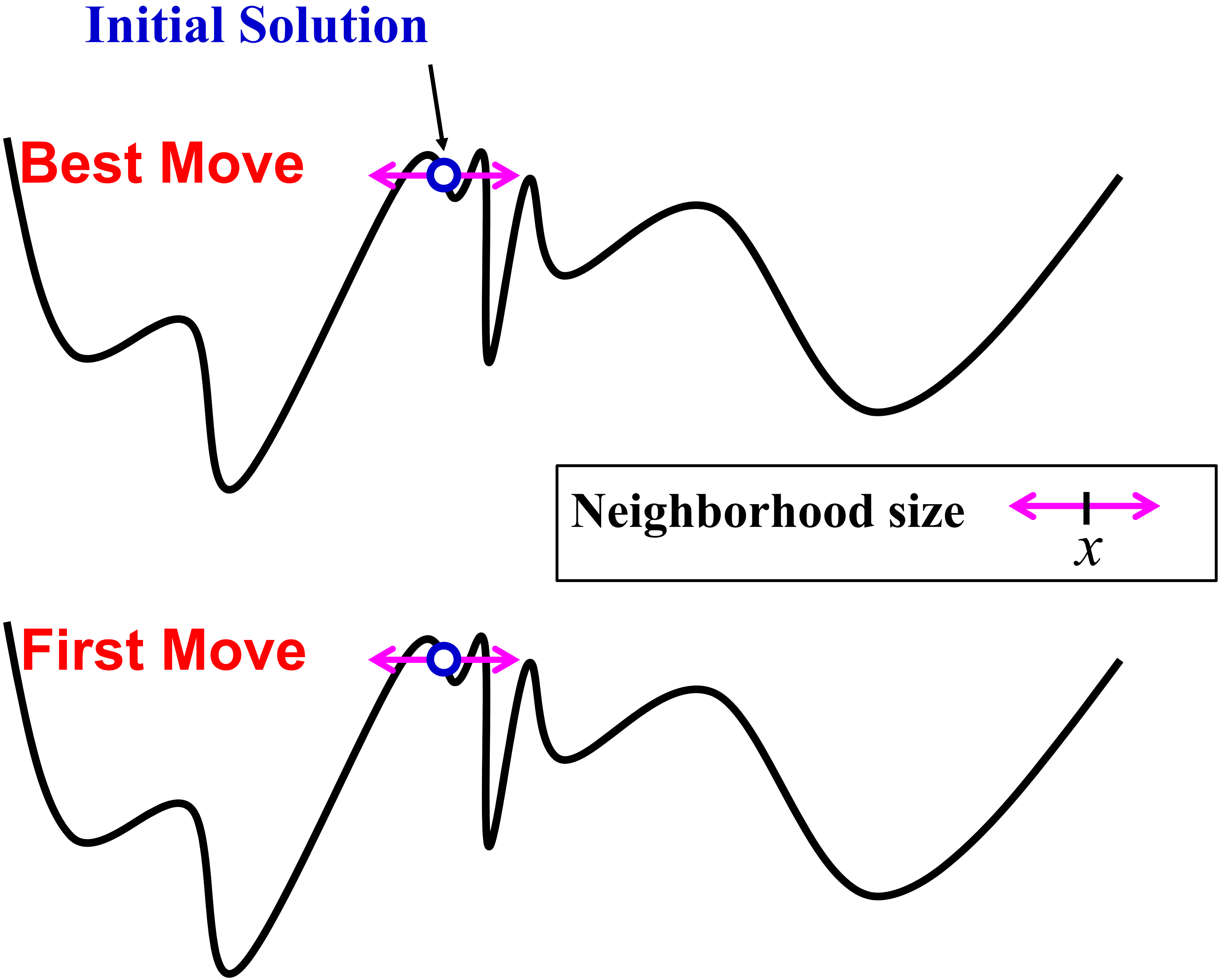
(2) With respect to the final solution quality, we cannot say which is better between these two strategies.

Best Move and First Move: Which is better ?



(1) In general, the first move needs less computation load.

(2) With respect to the final solution quality, we cannot say which is better between these two strategies.



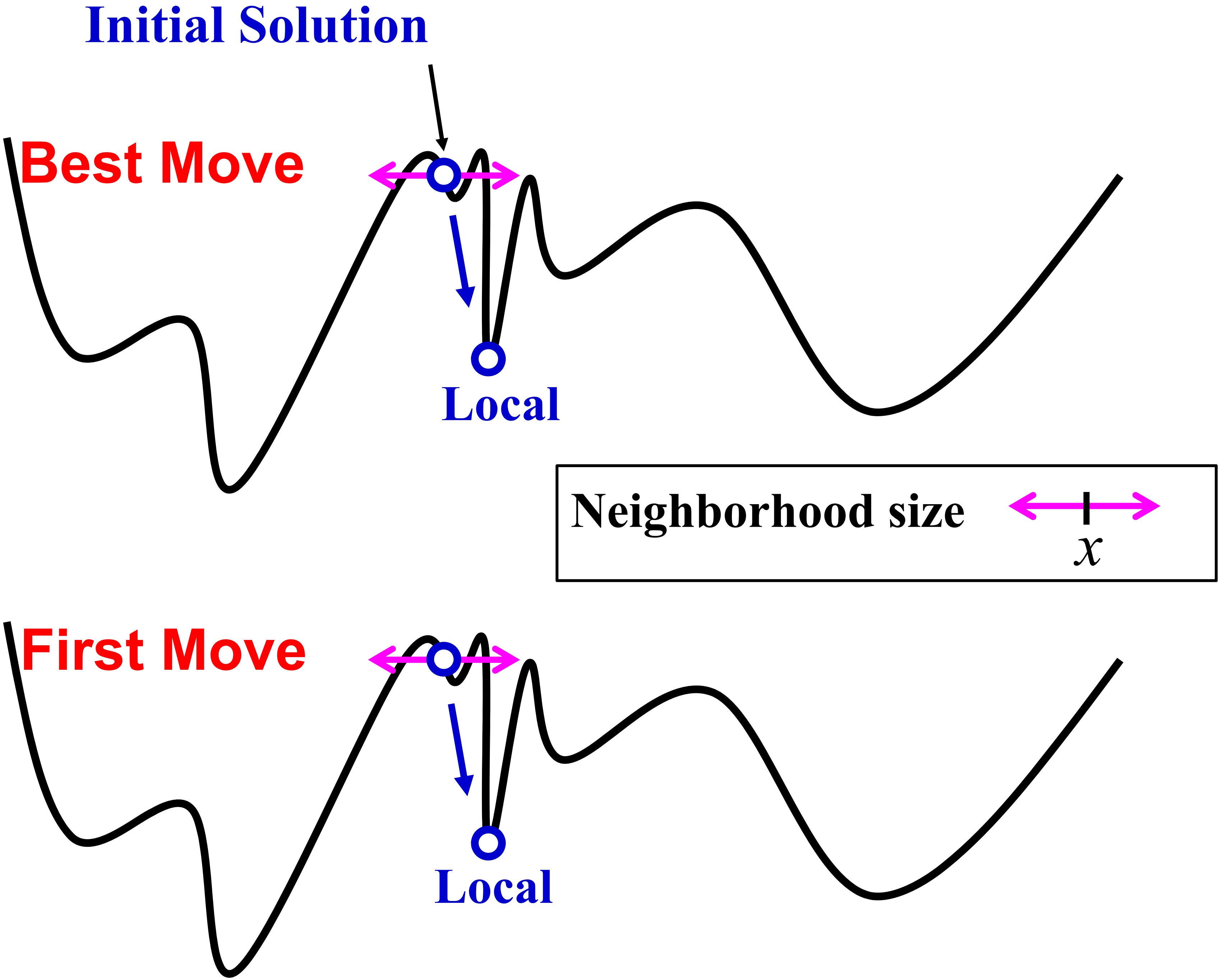
Initial Solution

Best Move

Neighborhood size

x

First Move



Initial Solution

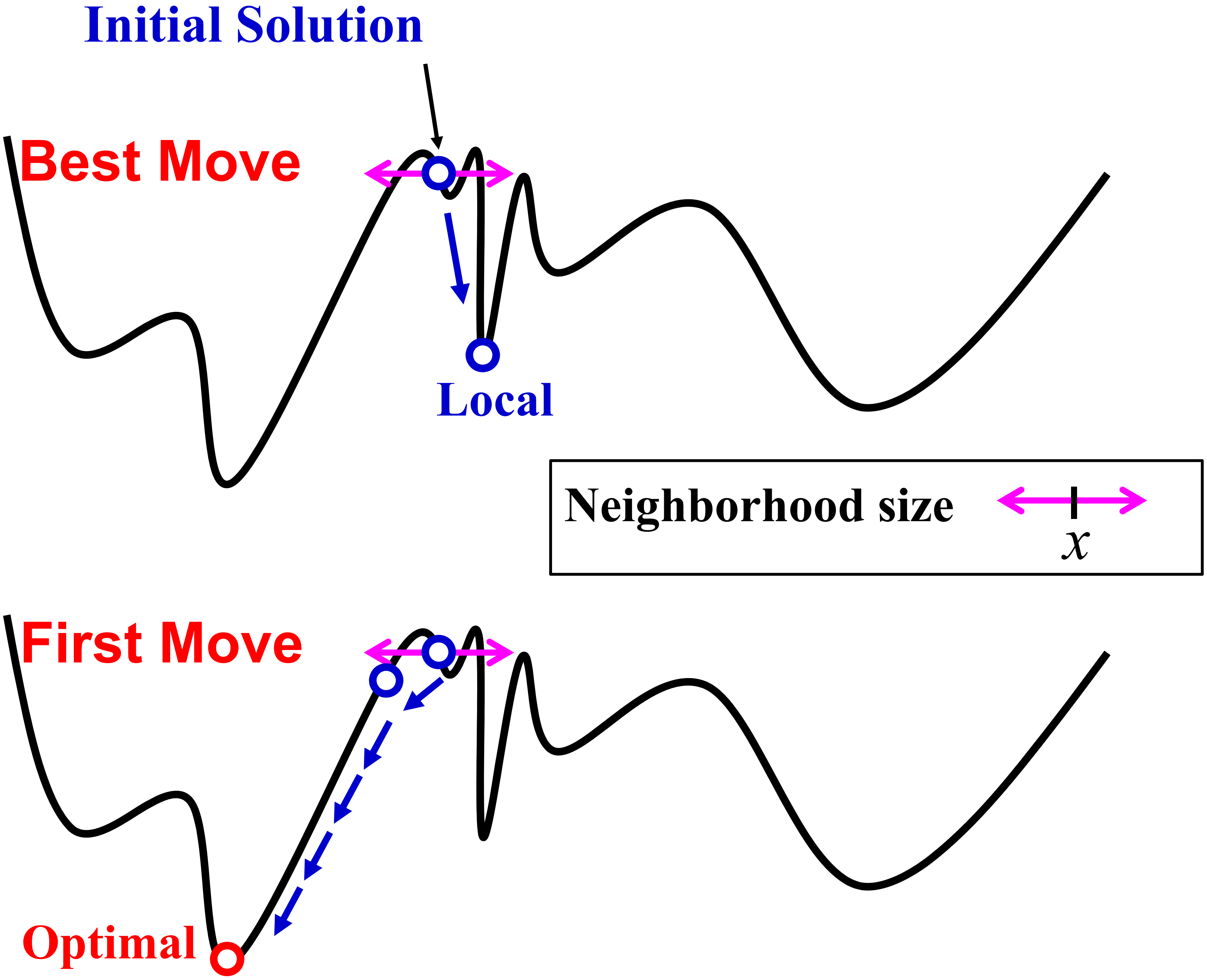
Best Move

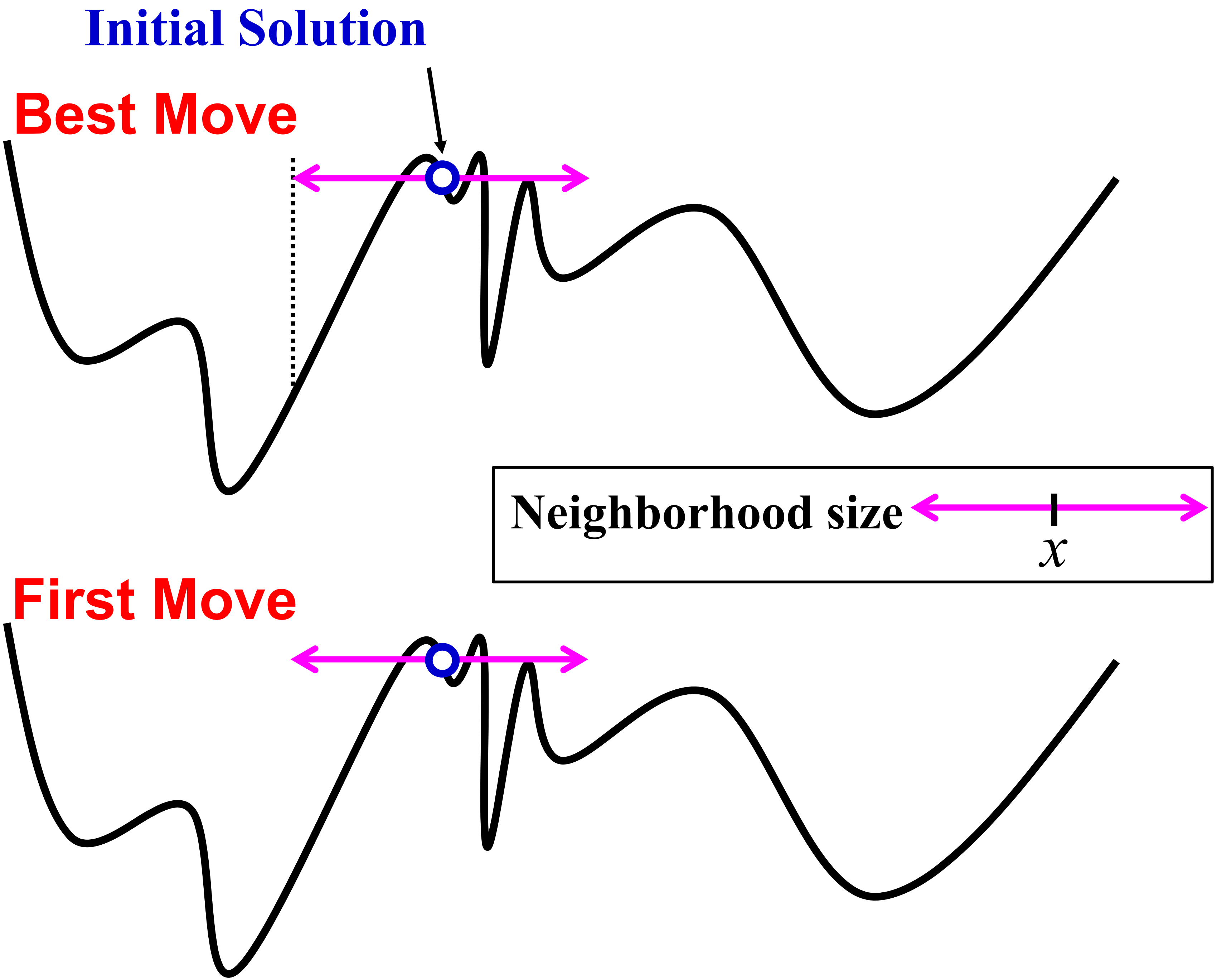
Local

Neighborhood size

First Move

Local


Initial Solution
Best Move
Local
Neighborhood size
x
First Move
Optimal

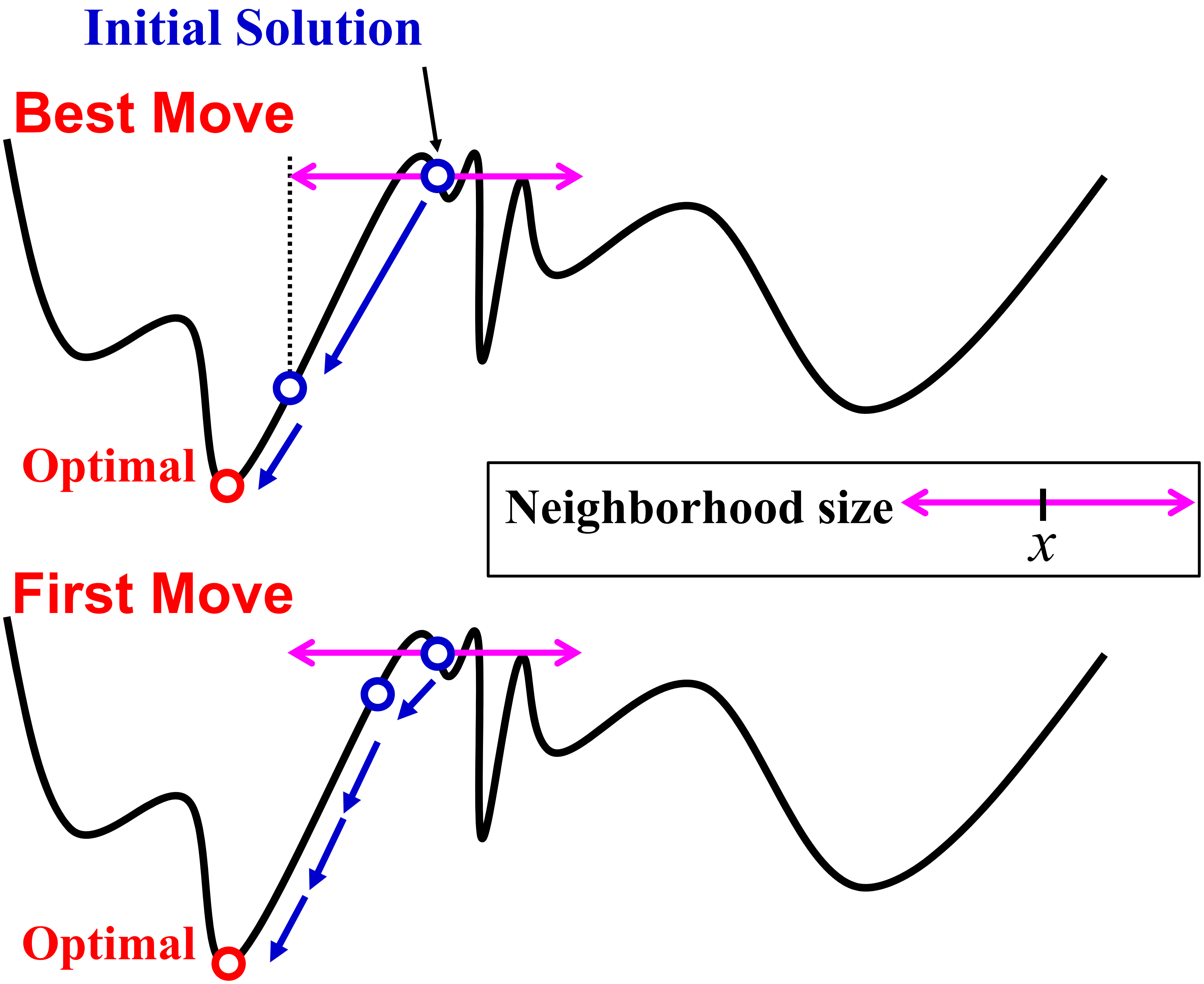


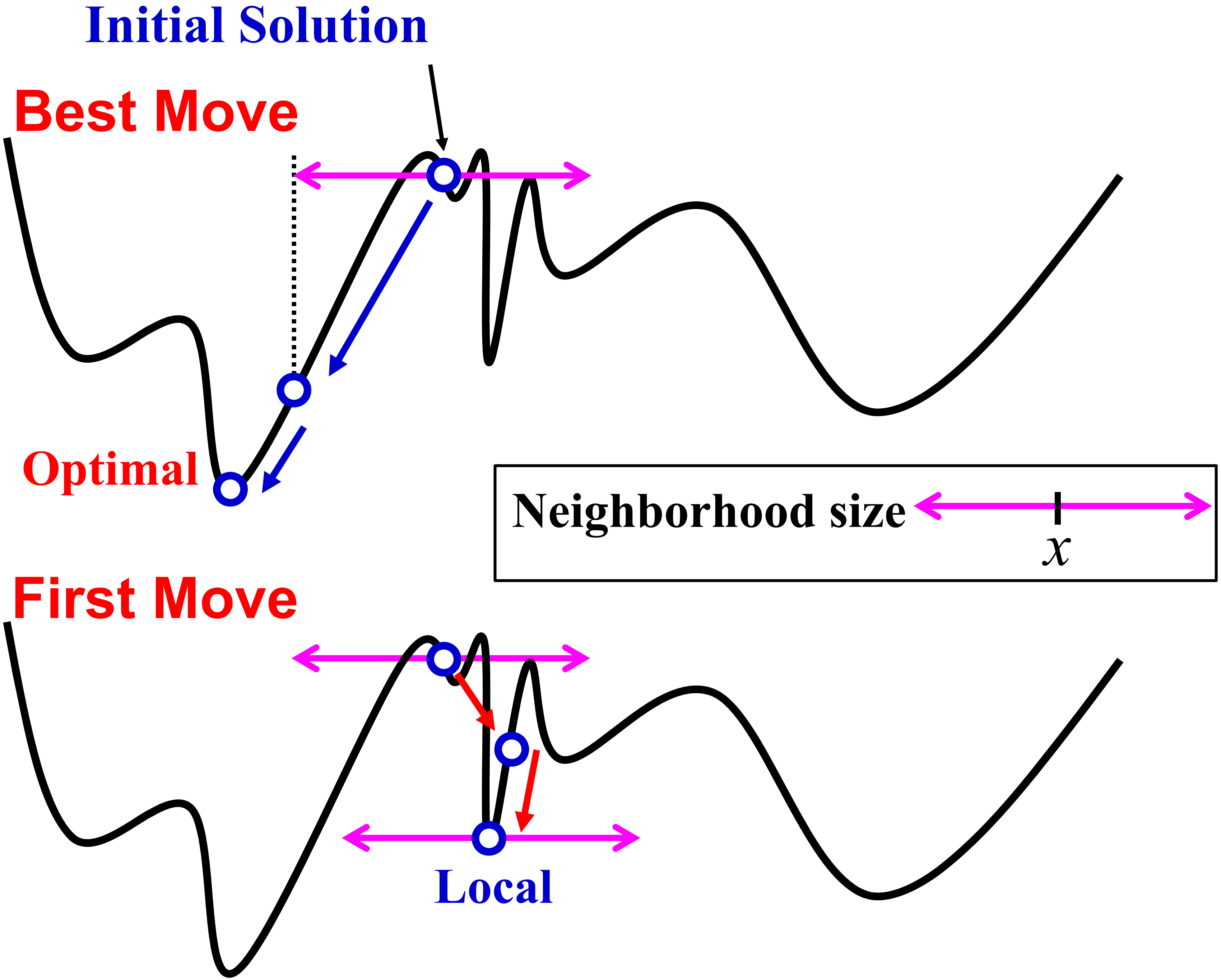
Initial Solution

Best Move

Neighborhood size x

First Move


Initial Solution
Best Move
Optimal
Neighborhood size
x
First Move
Optimal



Initial Solution

Best Move

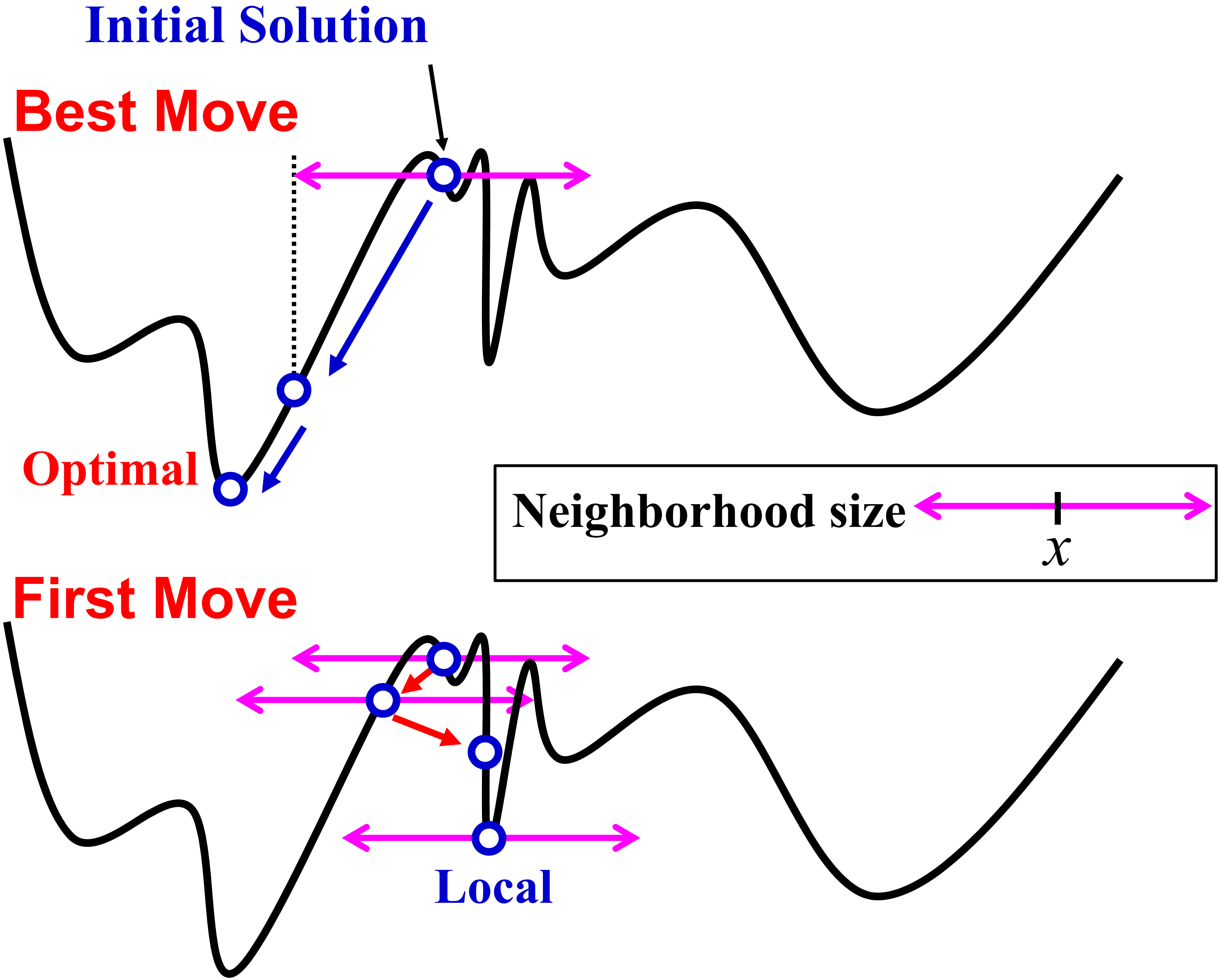
Optimal

Neighborhood size

x

First Move

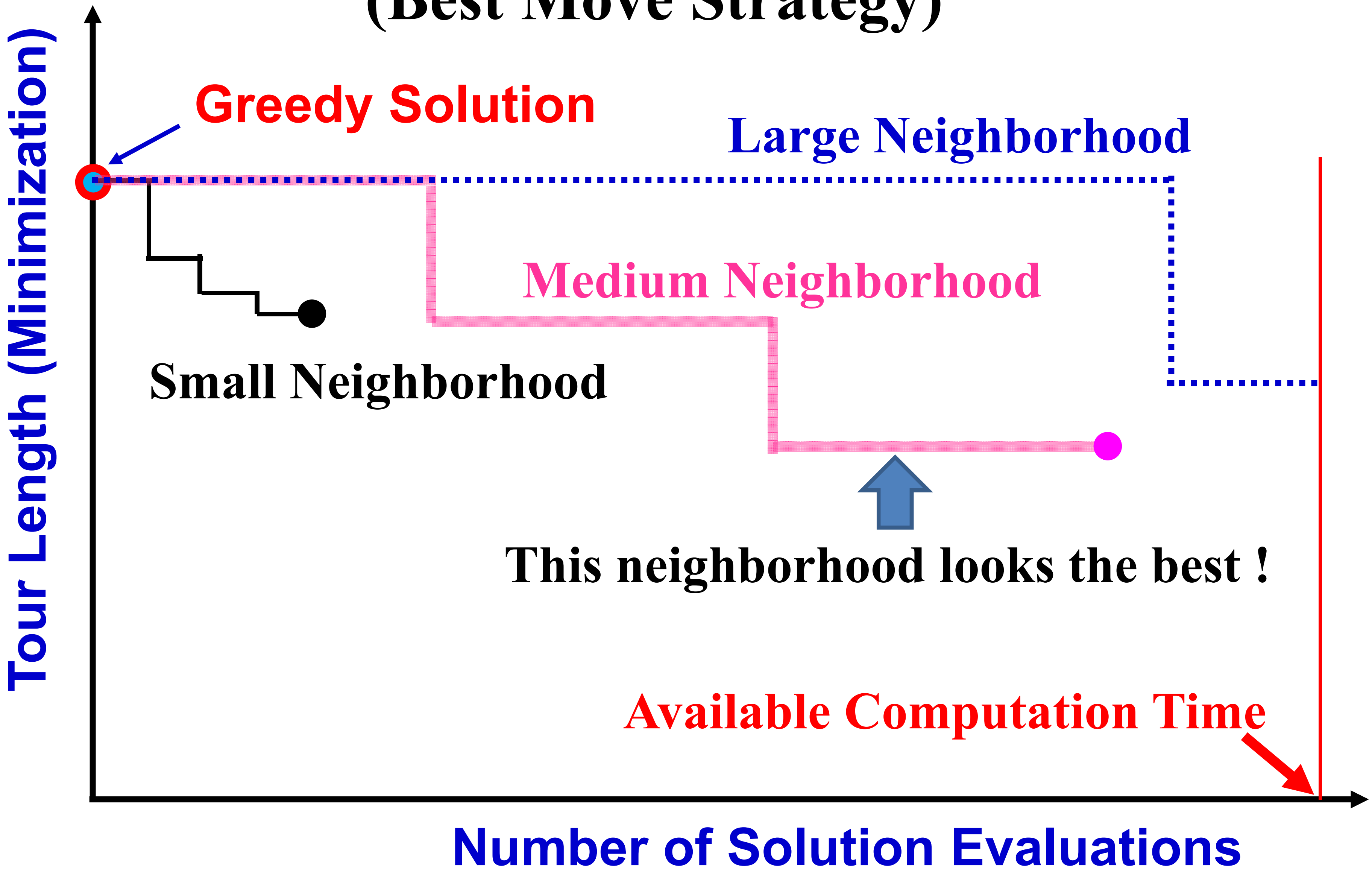
Local



Best Move

First Move

Behavior of Local Search
(Best Move Strategy)



Greedy Solution

Large Neighborhood

Medium Neighborhood

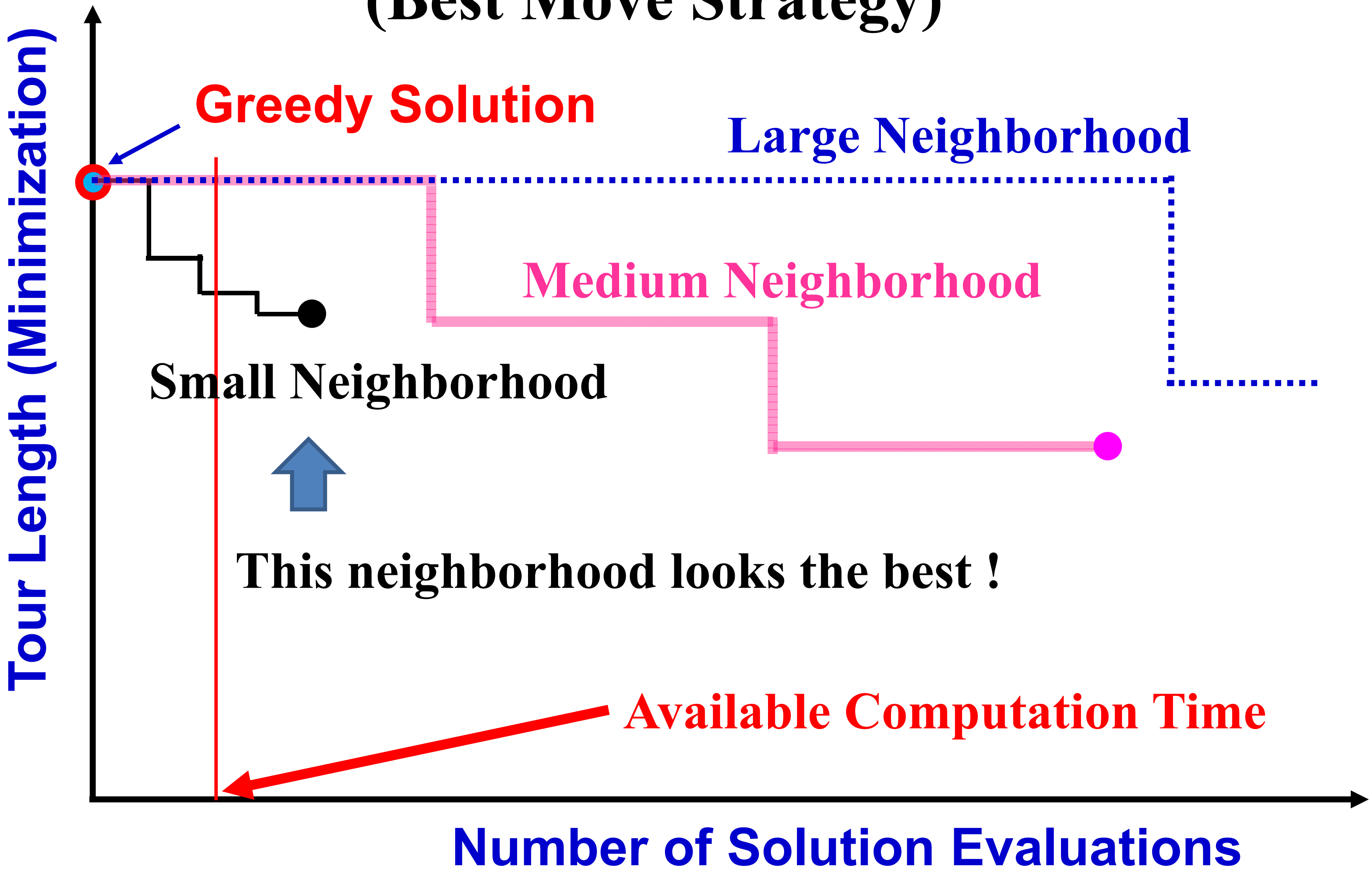
Small Neighborhood

This neighborhood looks the best !

Available Computation Time

Behavior of Local Search

(Best Move Strategy)

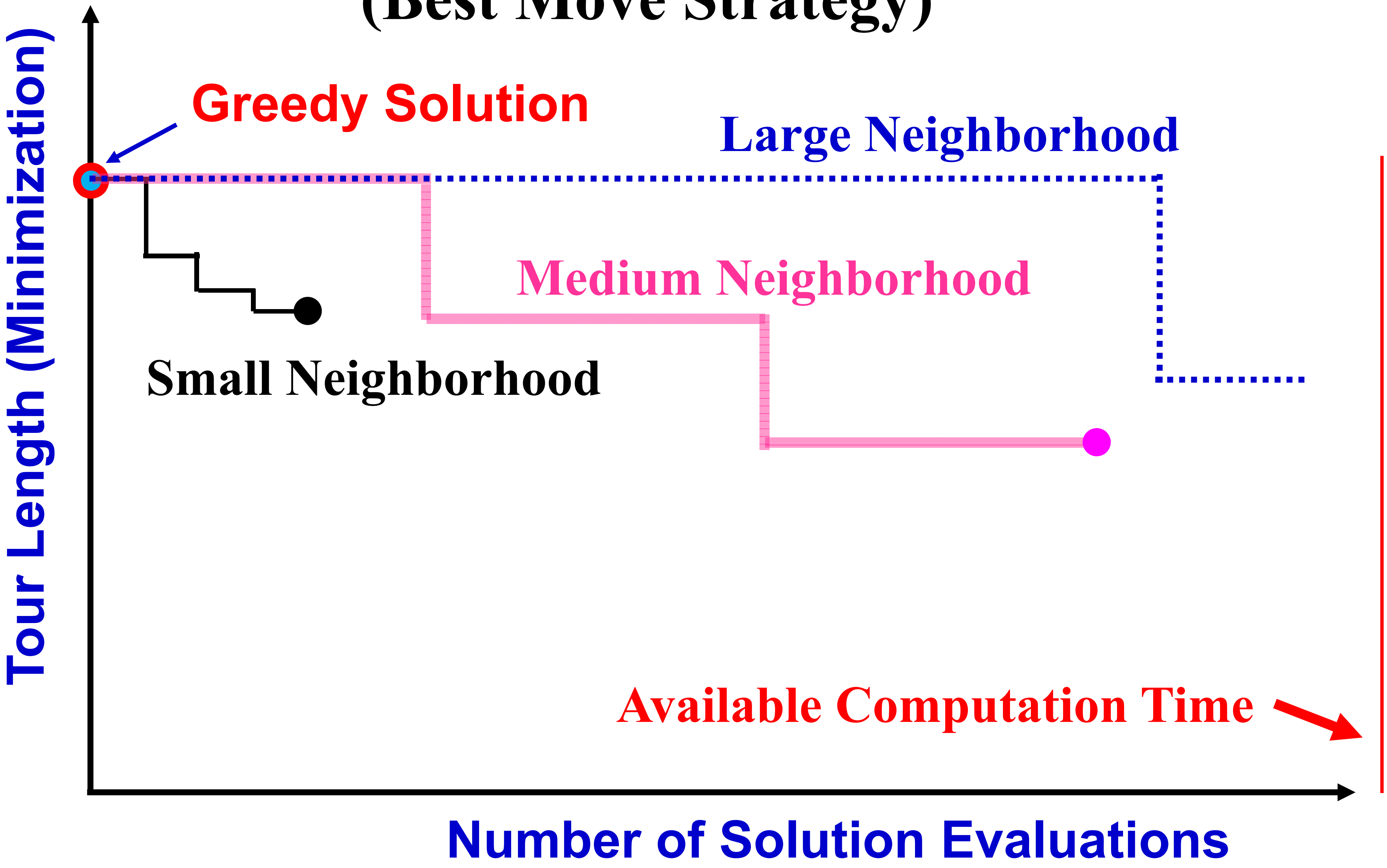


This neighborhood looks the best !

Available Computation Time

Behavior of Local Search

(Best Move Strategy)



Local Search

Implementation Issues

(1) Specification of an initial solution

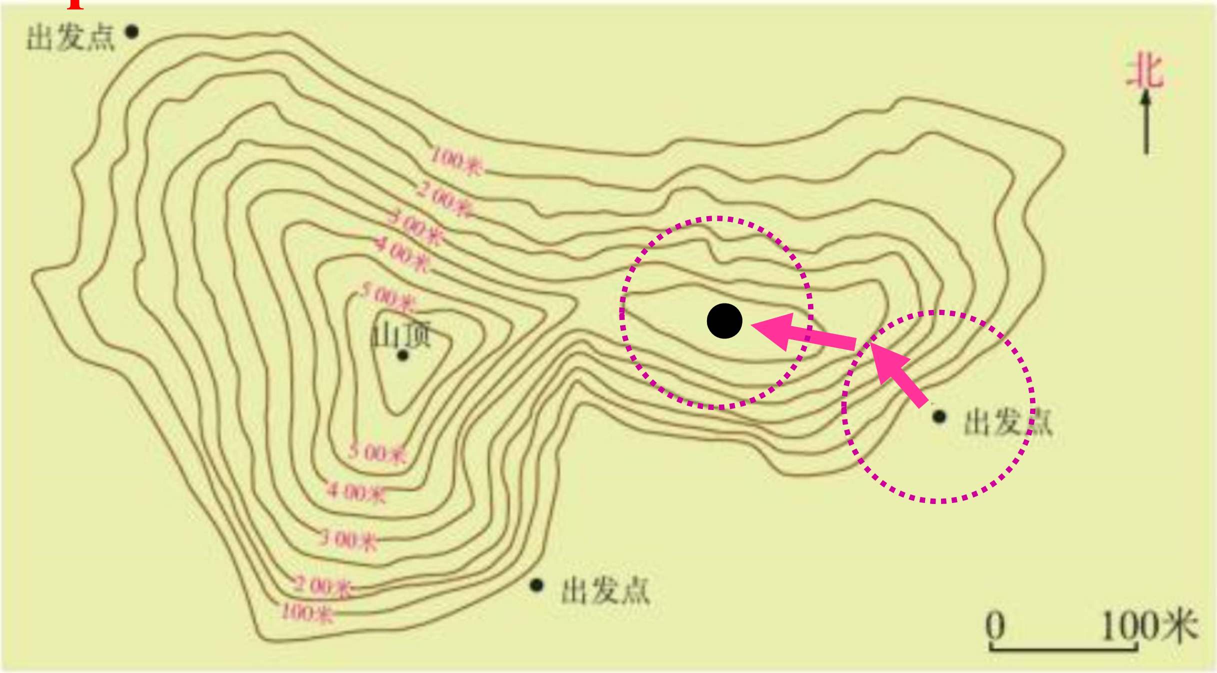
(2) Specification of a neighborhood structure

(3) Choice between the first move and the best move

⇩

Local Solution

Question: How can we improve local search ?



Local Search

Implementation Issues

(1) Specification of an initial solution

(2) Specification of a neighborhood structure

(3) Choice between the first move and the best move

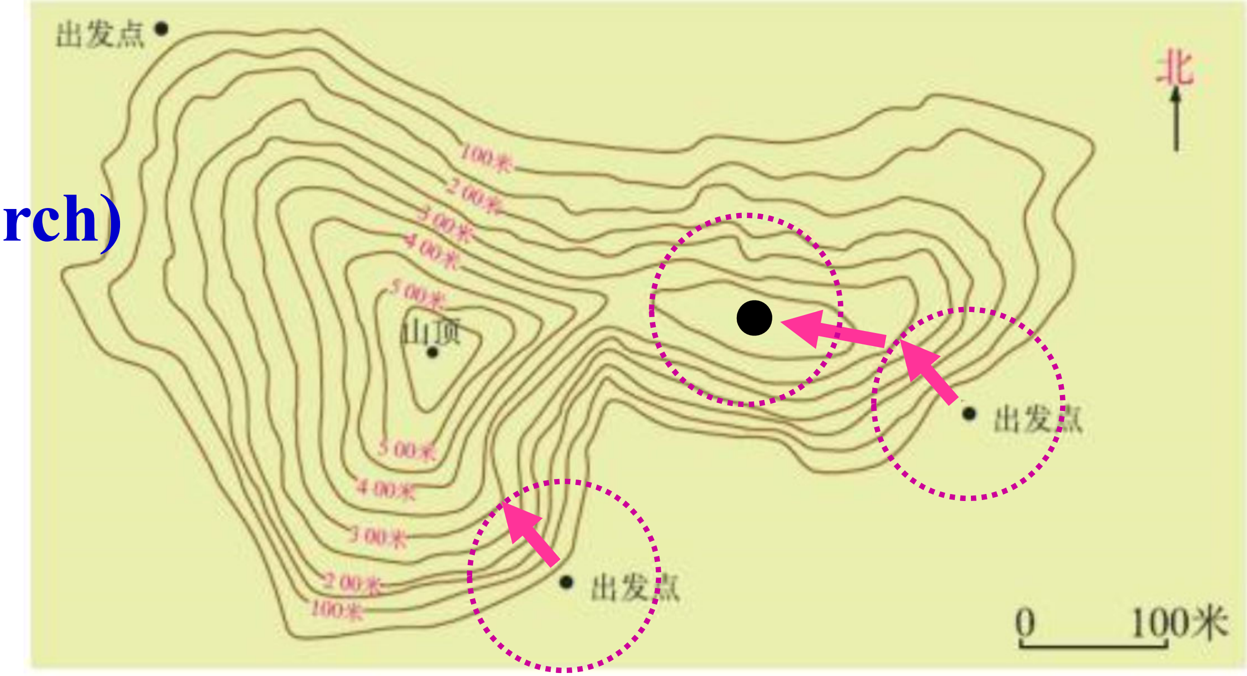
⇩

Local Solution

Next Step

- Restart

(Iterated Local Search)



Iterated Local Search

Iterate LS from Different Initial Solutions

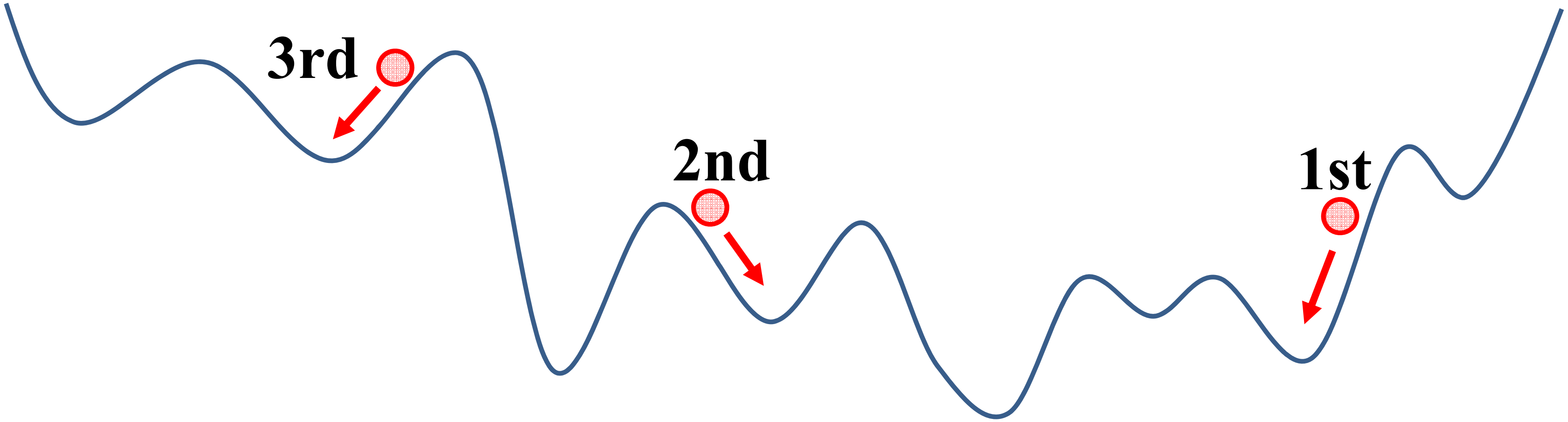
Iterate the following steps:

(1) Generate an initial solution ***x***.

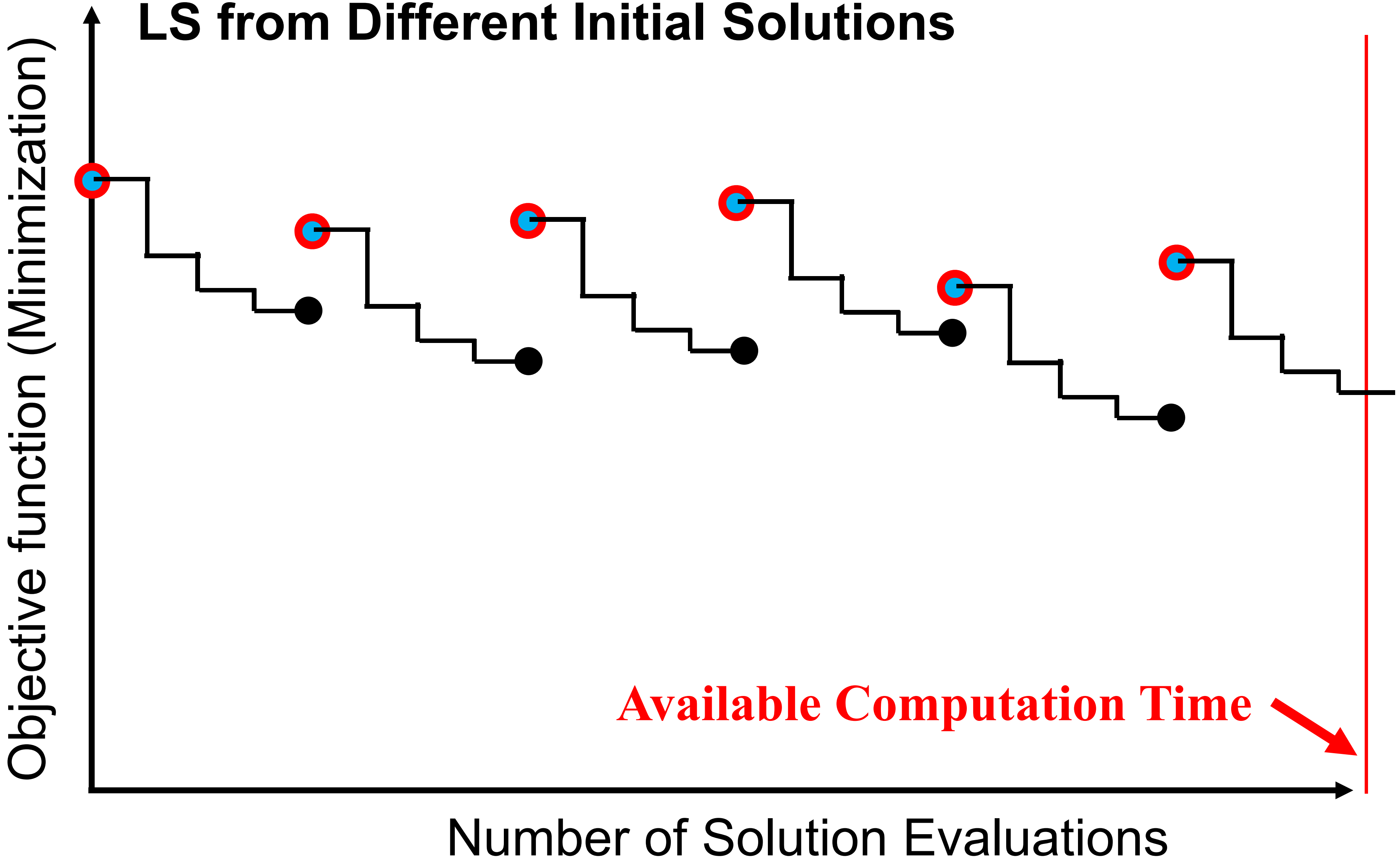
(2) Iterate the following steps:

(i) The first move or the best move (if a neighbor ***y*** of the current solution ***x*** is better than ***x***, replace ***x*** with ***y***).

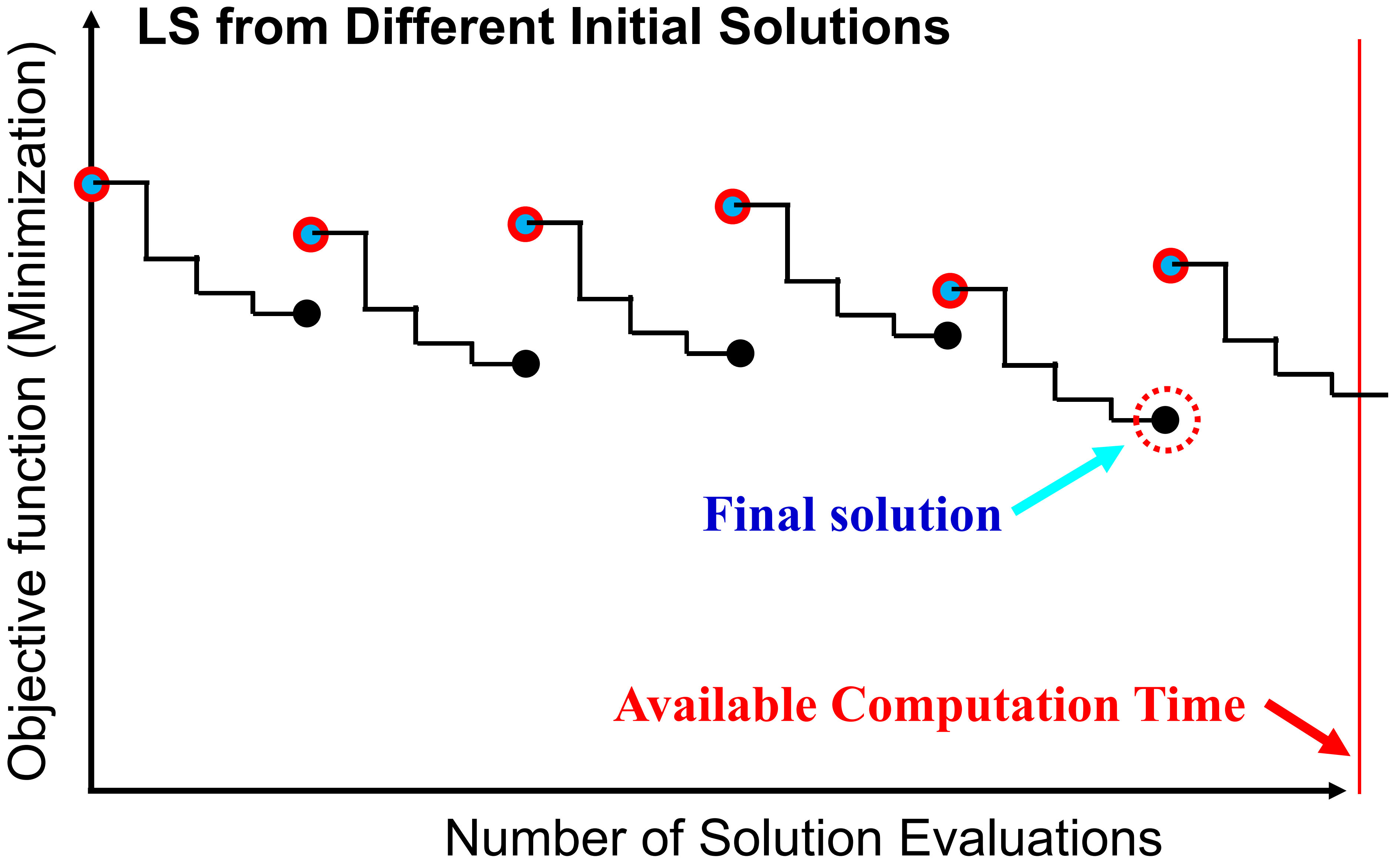
(ii) If no better solution exists in the neighborhood of ***x***, terminate the current execution of LS (i.e., restart with a new initial solution ***x*** in (1)).



Iterated Local Search



Iterated Local Search



Iterated Local Search

Iterate LS from Different Initial Solutions

Iterate the following steps:

(1) Generate an initial solution ***x***.

(2) Iterate the following steps:

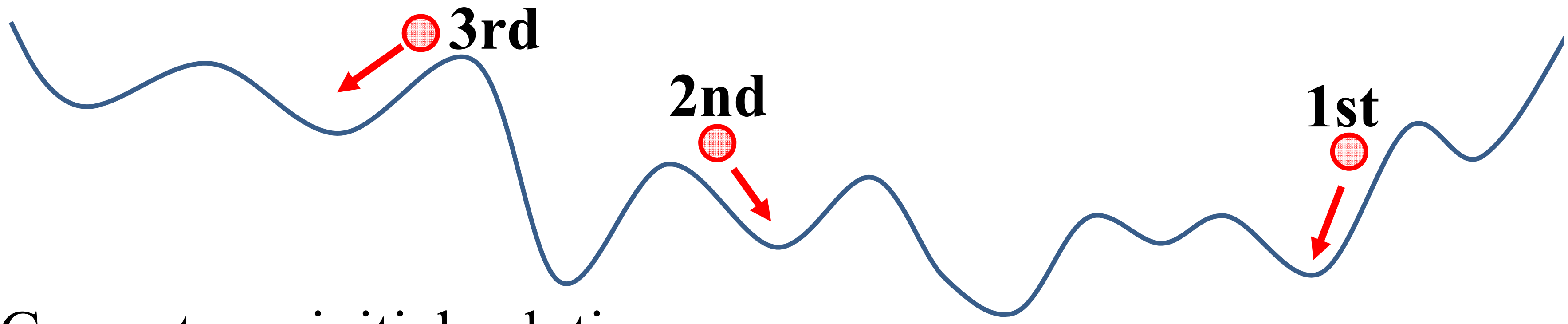
(i) The first move or the best move (if a neighbor ***y*** of the current solution ***x*** is better than ***x***, replace ***x*** with ***y***).

(ii) If no better solution exists in the neighborhood of ***x***, terminate the current execution of LS (i.e., restart with a new initial solution ***x*** in (1)).

Discussions:

How to specify an initial solution for each LS run in (1)?

- In the first LS run: **Greedy solution.**
- In the other LS runs: **????????????** .



(1) Generate an initial solution ***x***.

(2) Iterate the following steps:

(i) The first move or the best move (if a neighbor ***y*** of the current solution ***x*** is better than ***x***, replace ***x*** with ***y***).

(ii) If no better solution exists in the neighborhood of ***x***, terminate the current execution of LS (i.e., restart with a new initial solution ***x*** in (1)).

Discussions:

How to specify an initial solution for each LS run in (1)?

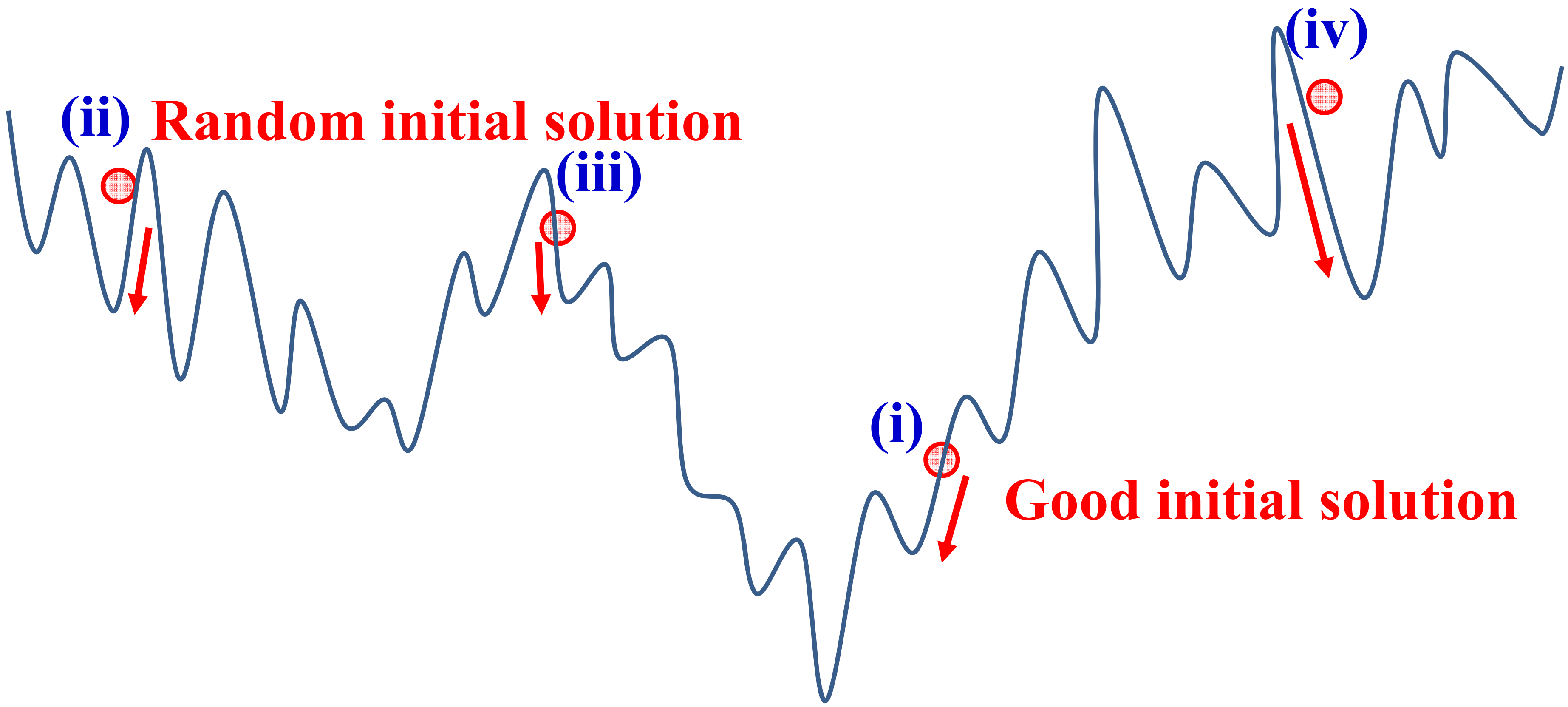
- In the first LS run: **Greedy solution.**
- In the other LS runs: ????????????? .

Main Issue

Choice of Initial Solutions:

- First initial solution: **Greedy solution**.

- Other initial solutions: **Random solutions**.

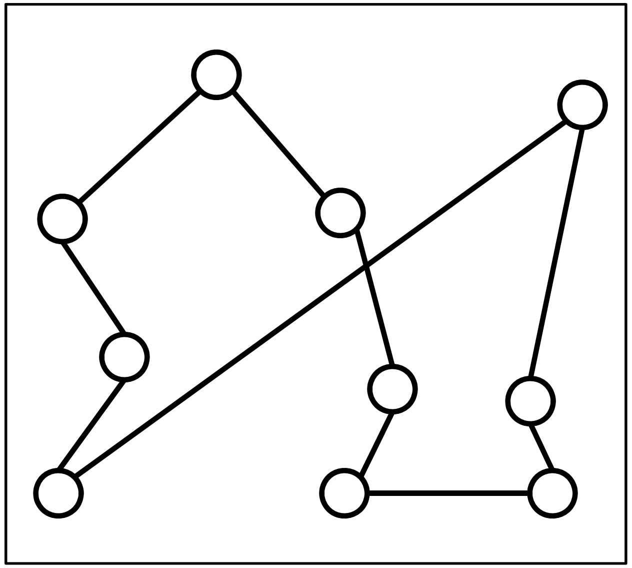


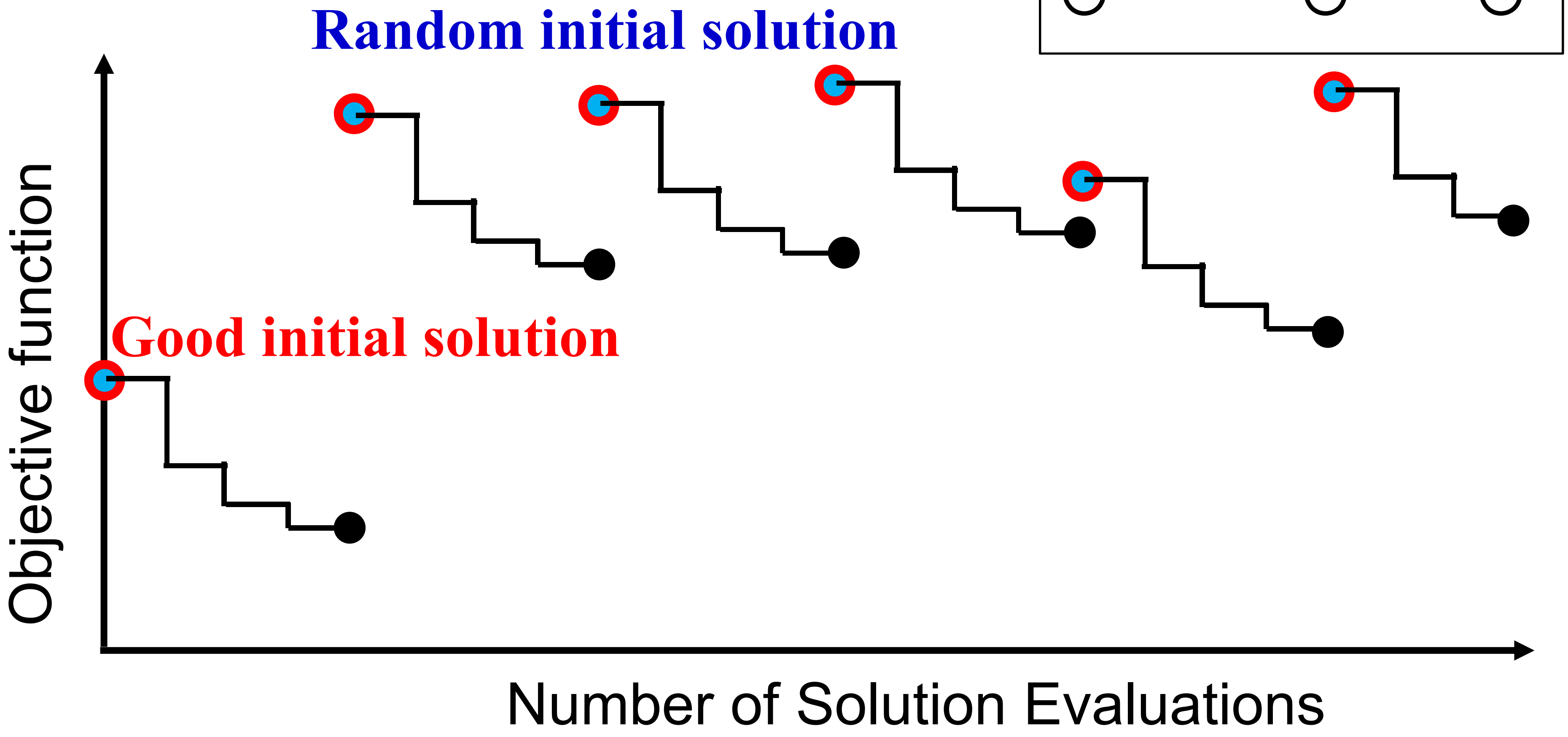
Main Issue

Choice of Initial Solutions:

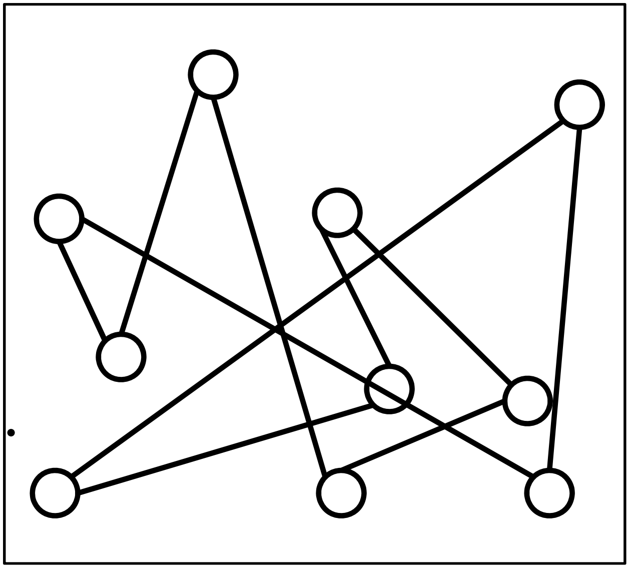
- First initial solution: **Greedy solution**.

- Other initial solutions: Random solutions.





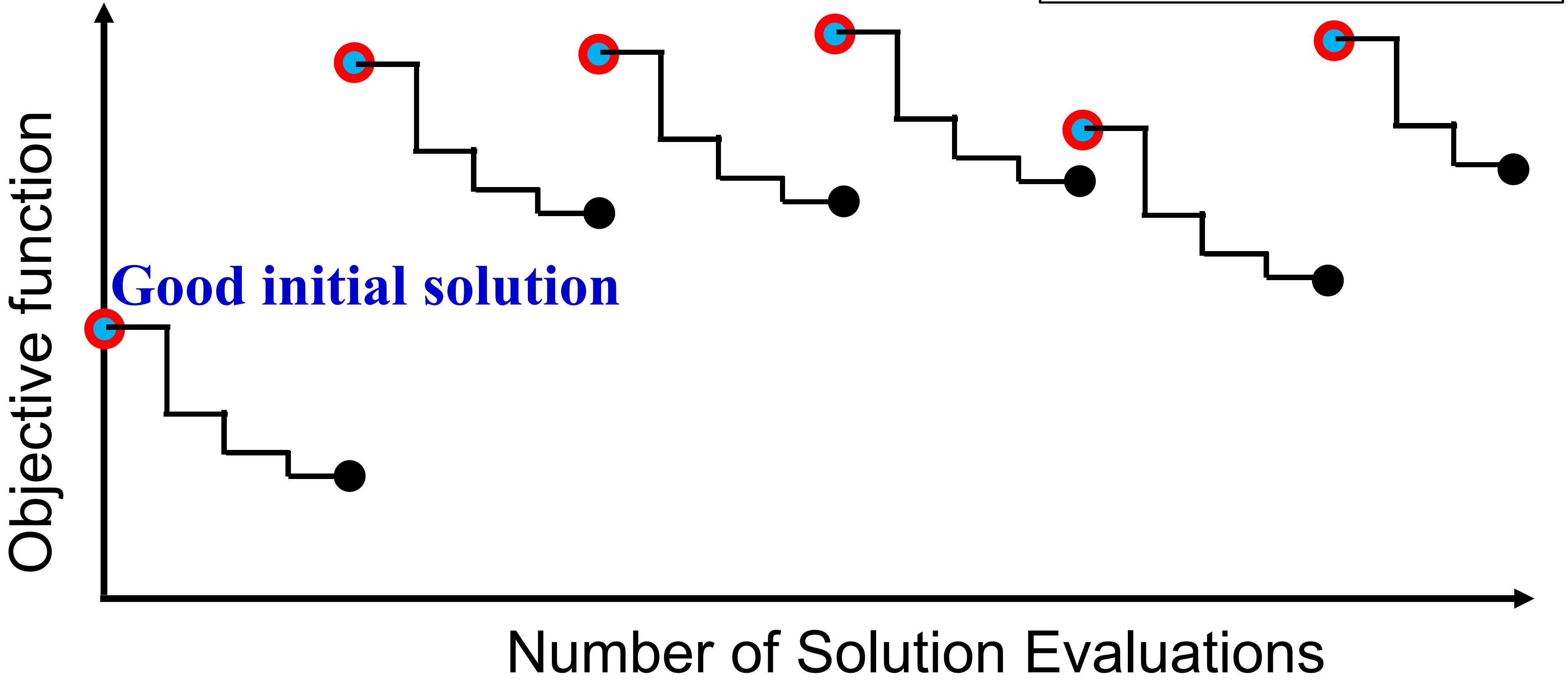
Main Issue



Choice of Initial Solutions:

- First initial solution: Greedy solution.
- Other initial solutions: **Random solutions**.

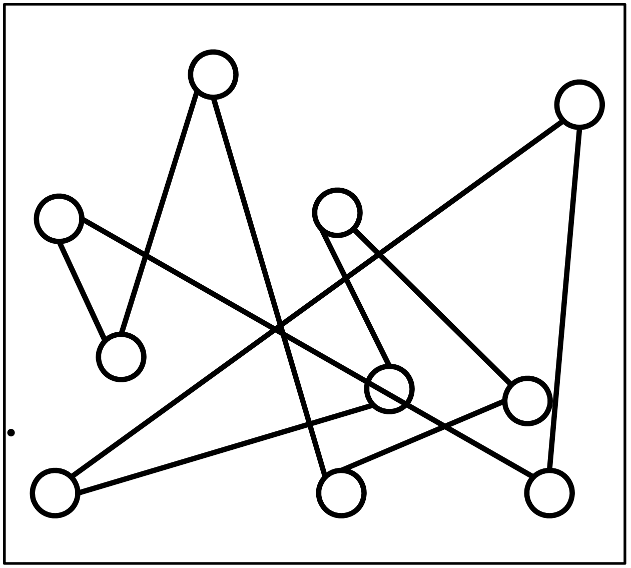
Random initial solution

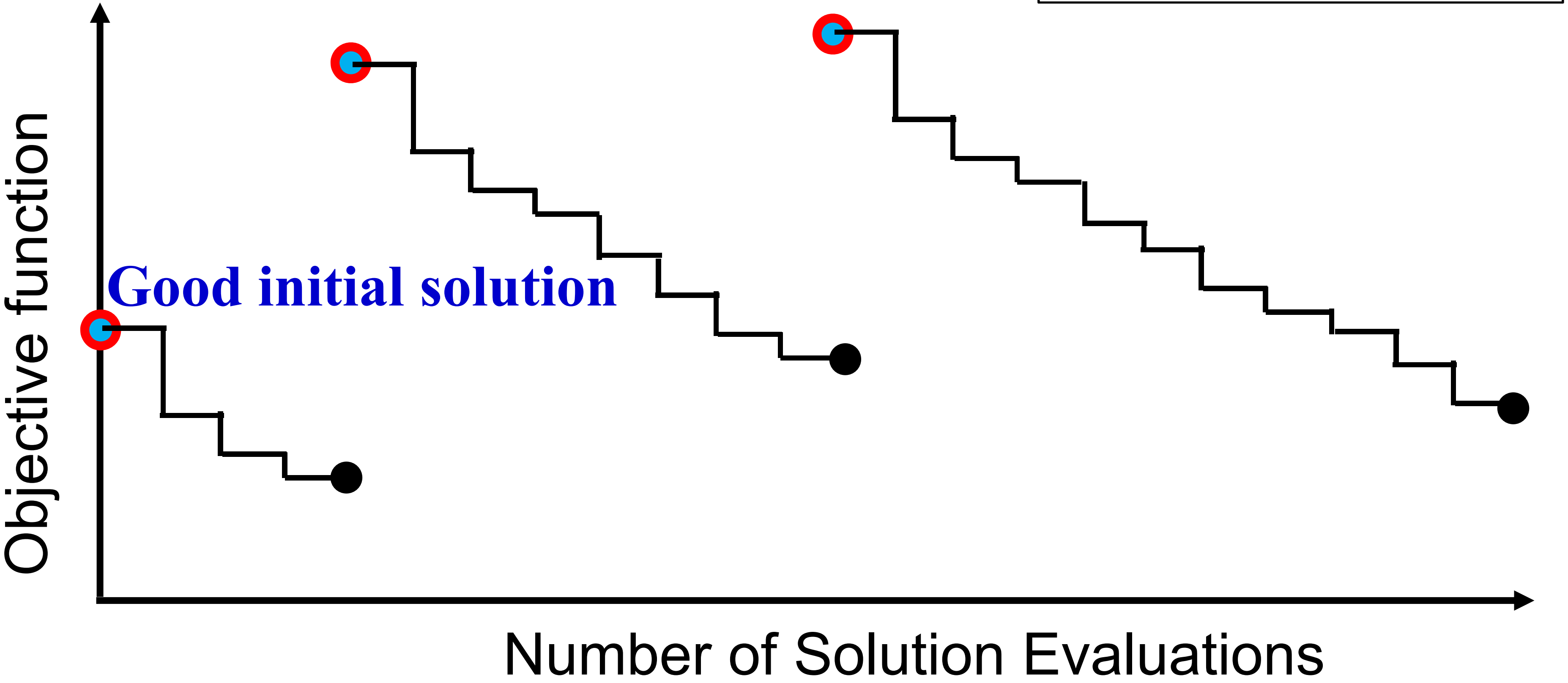


Main Issue

Choice of Initial Solutions:

- First initial solution: Greedy solution.
- Other initial solutions: **Random solutions**.

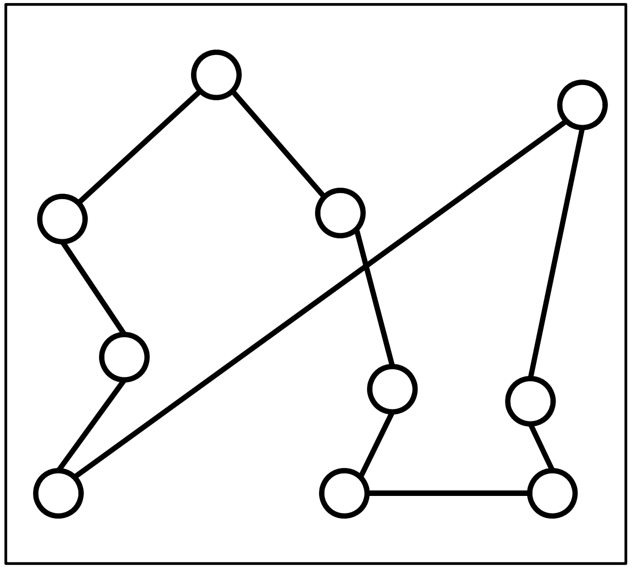




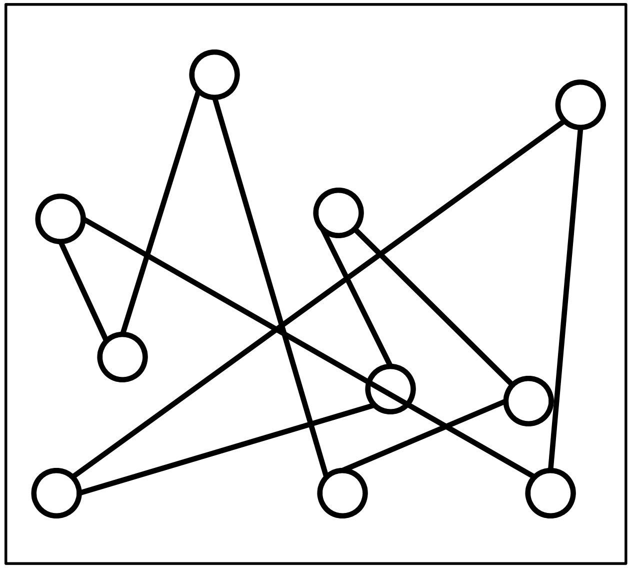
Main Issue

Choice of Initial Solutions:

- First initial solution: Greedy solution.
- Other initial solutions: Random solutions.



Good greedy solution

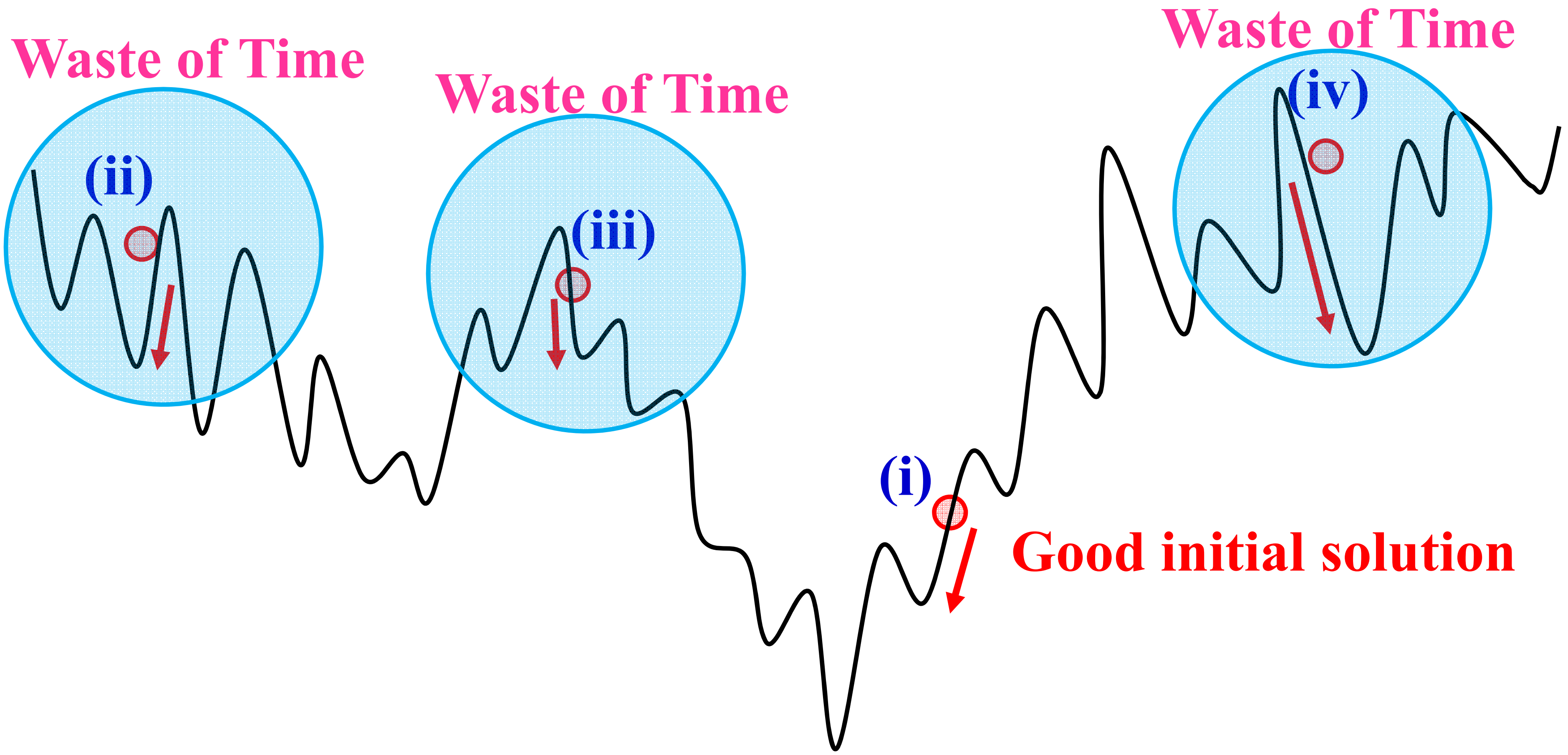


Poor random solution

Main Issue

Choice of Initial Solutions:

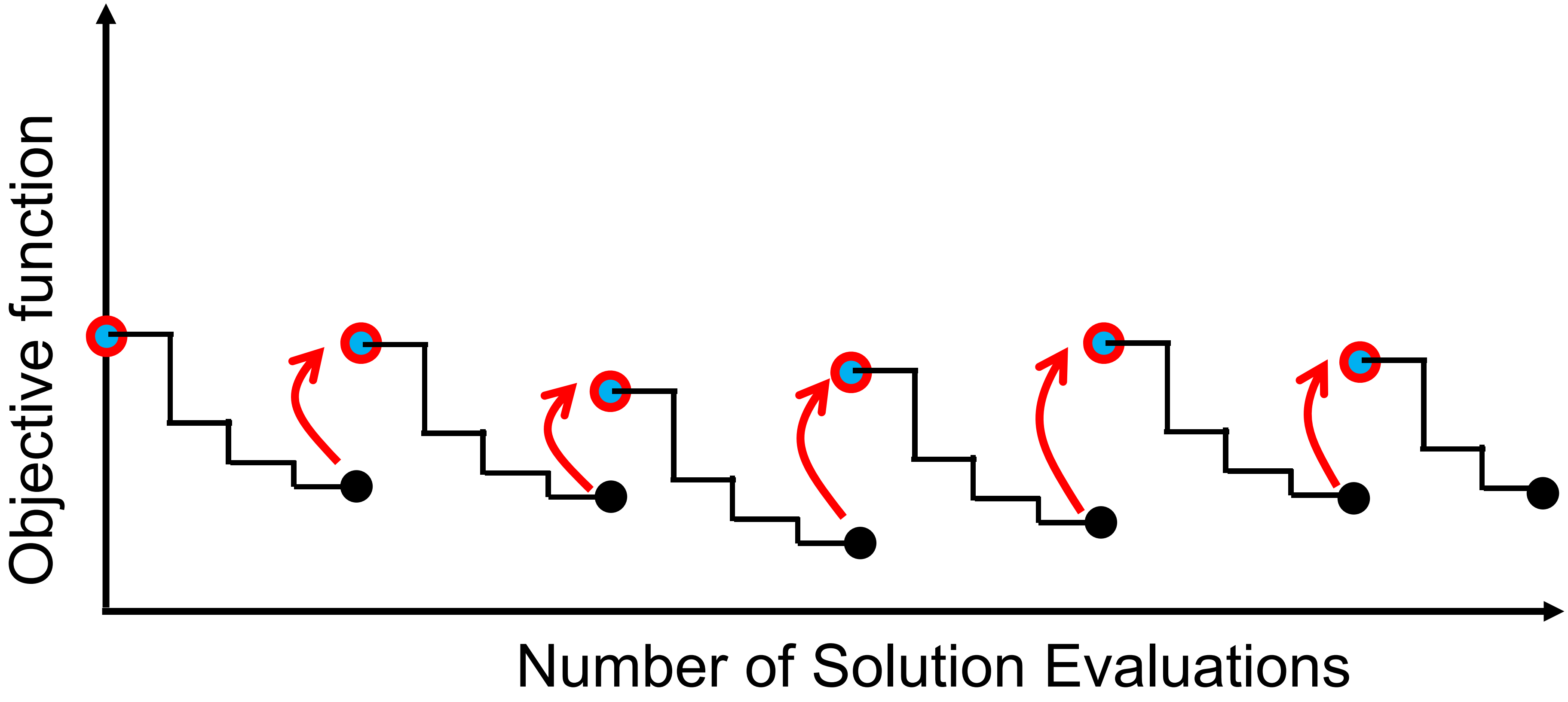
- First initial solution: Greedy solution.
- Other initial solutions: Random solutions.



One Idea

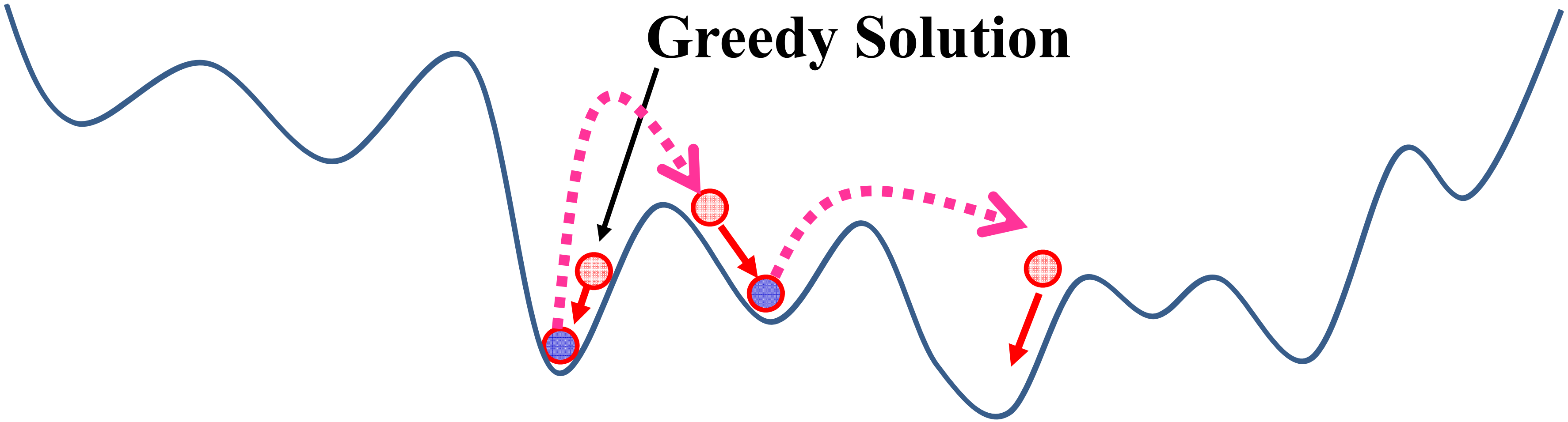
Use of the Current Solution

- First initial solution: Greedy solution.
- Other initial solutions: Generate from the final solution of the previous LS run

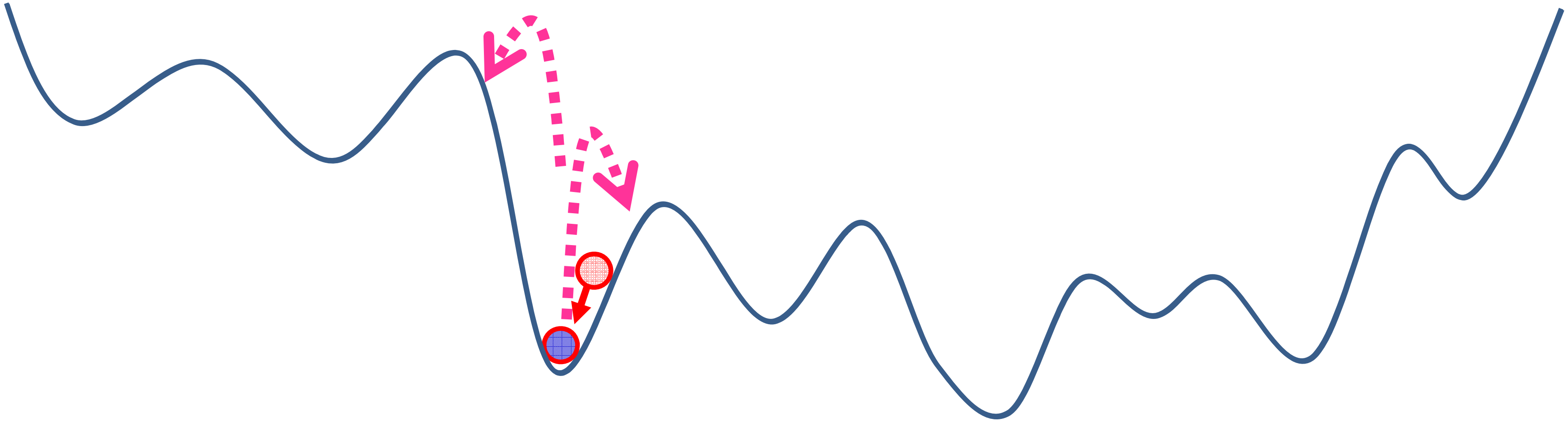


How to specify initial solutions

A new solution form the final solution

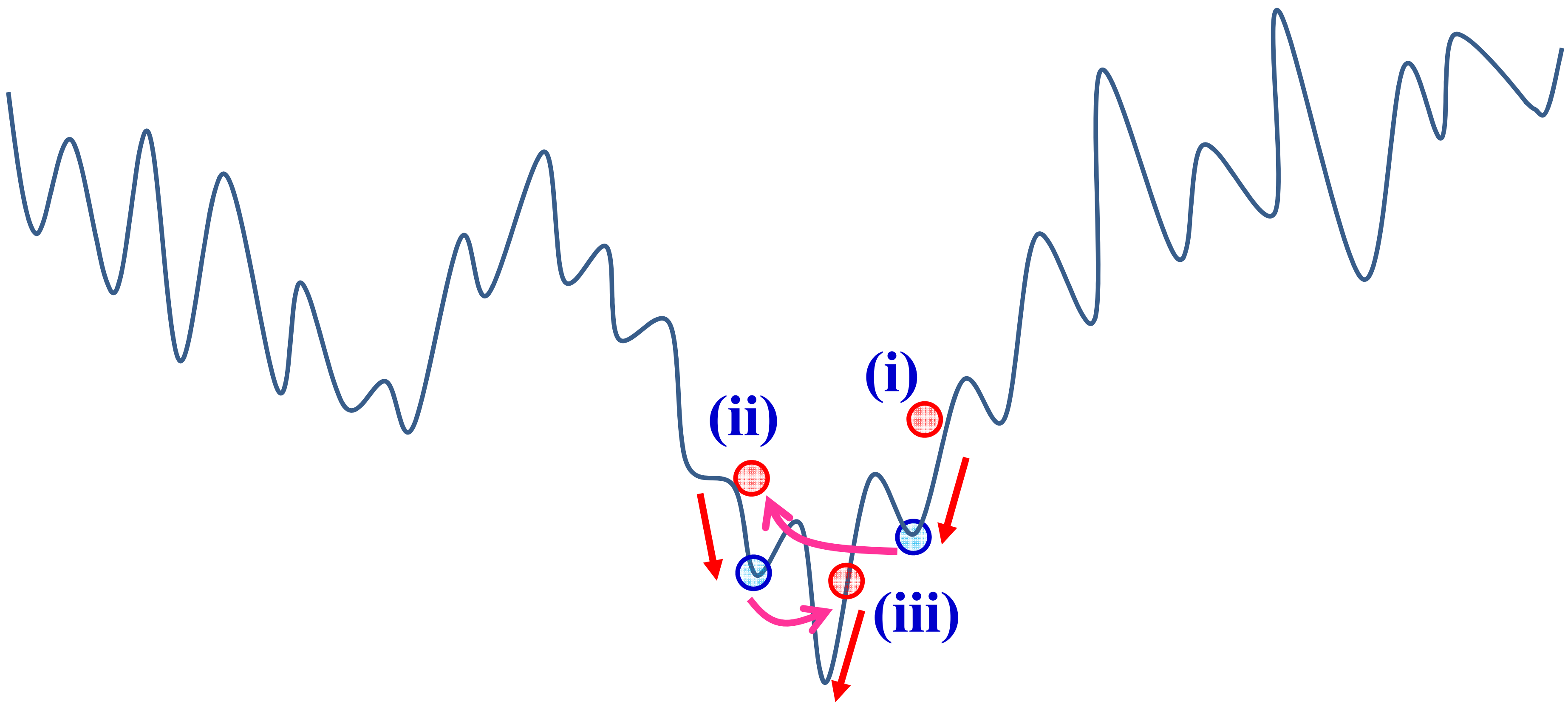


The following undesirable situation can happen.



Use of the final Solution

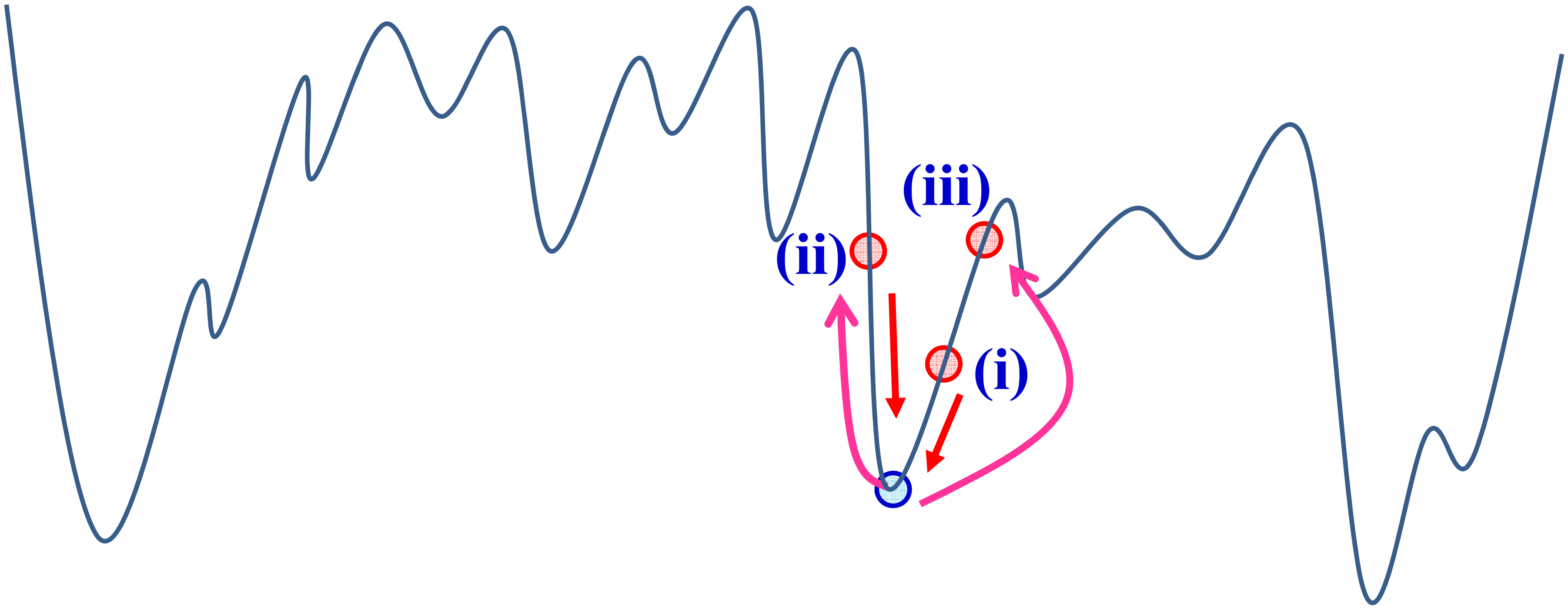
- First initial solution: Greedy solution.
- Other initial solutions: Generate from the final solution



The optimal solution can be efficiently searched by local search..

Use of the final Solution

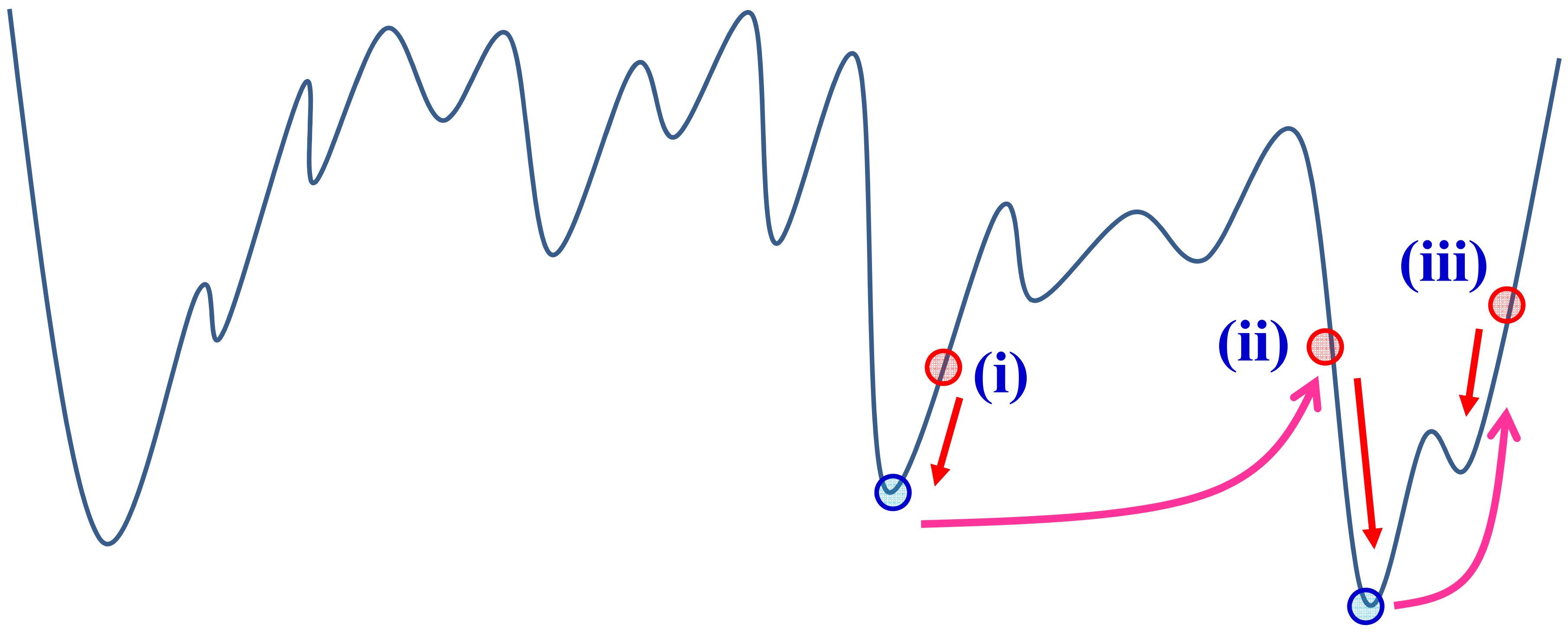
- First initial solution: Greedy solution.
- Other initial solutions: Generate from the final solution



The same solution can be obtained from different initial solutions.

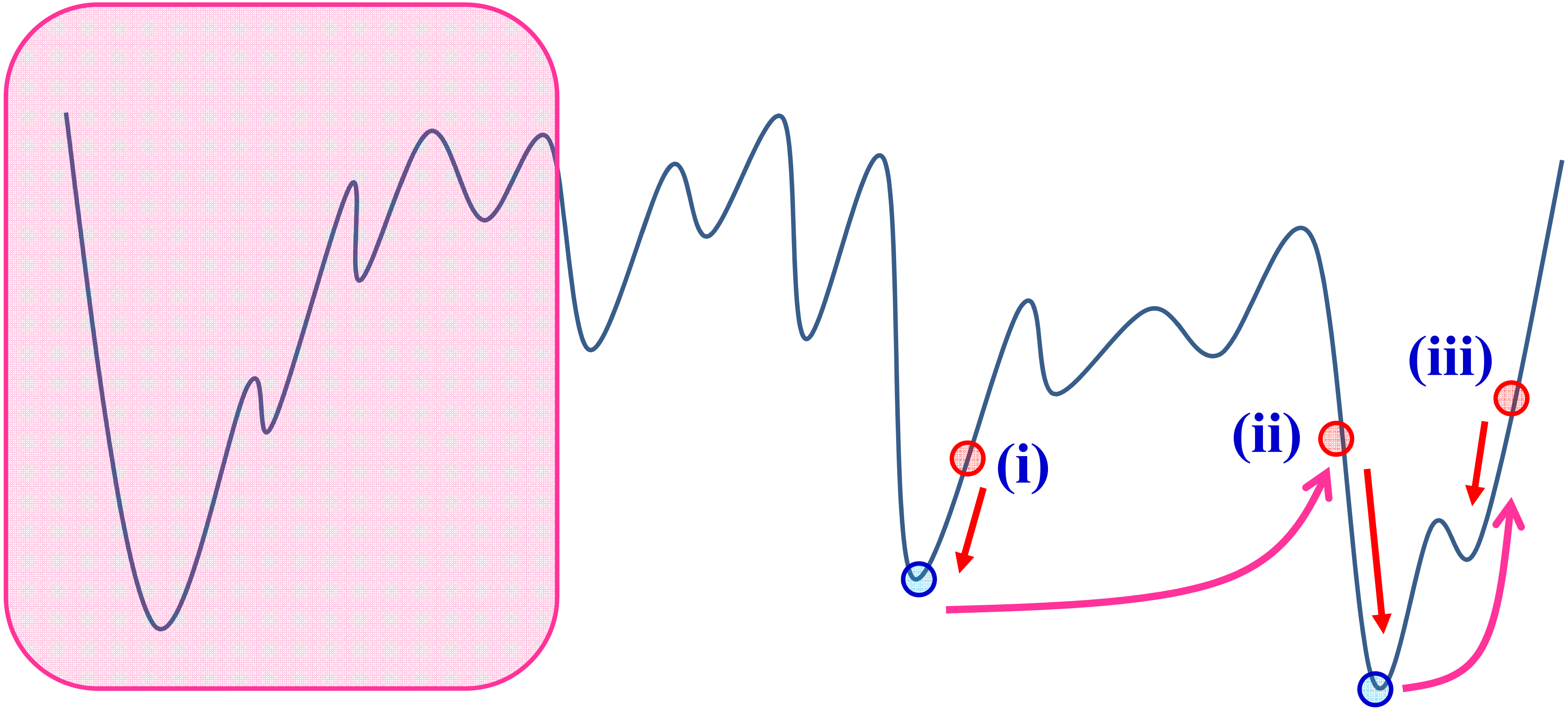
Good Initial Solutions

- Close to a good local solution
- Search for a different local solution



Good Initial Solutions

- Close to a good local solution
- Search for a different local solution

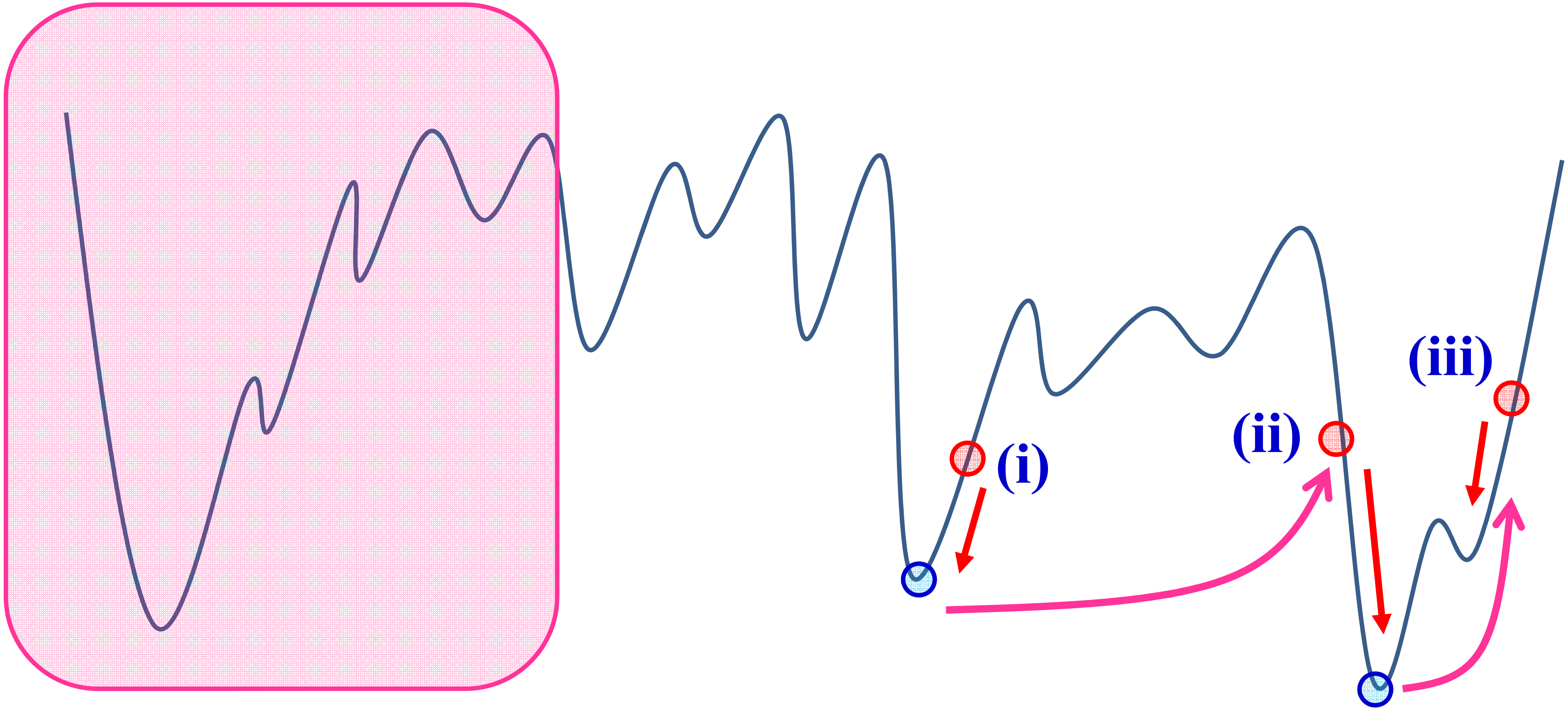


The search of this region is difficult.

Initial Solutions

Random: Wide search region & Inefficient search

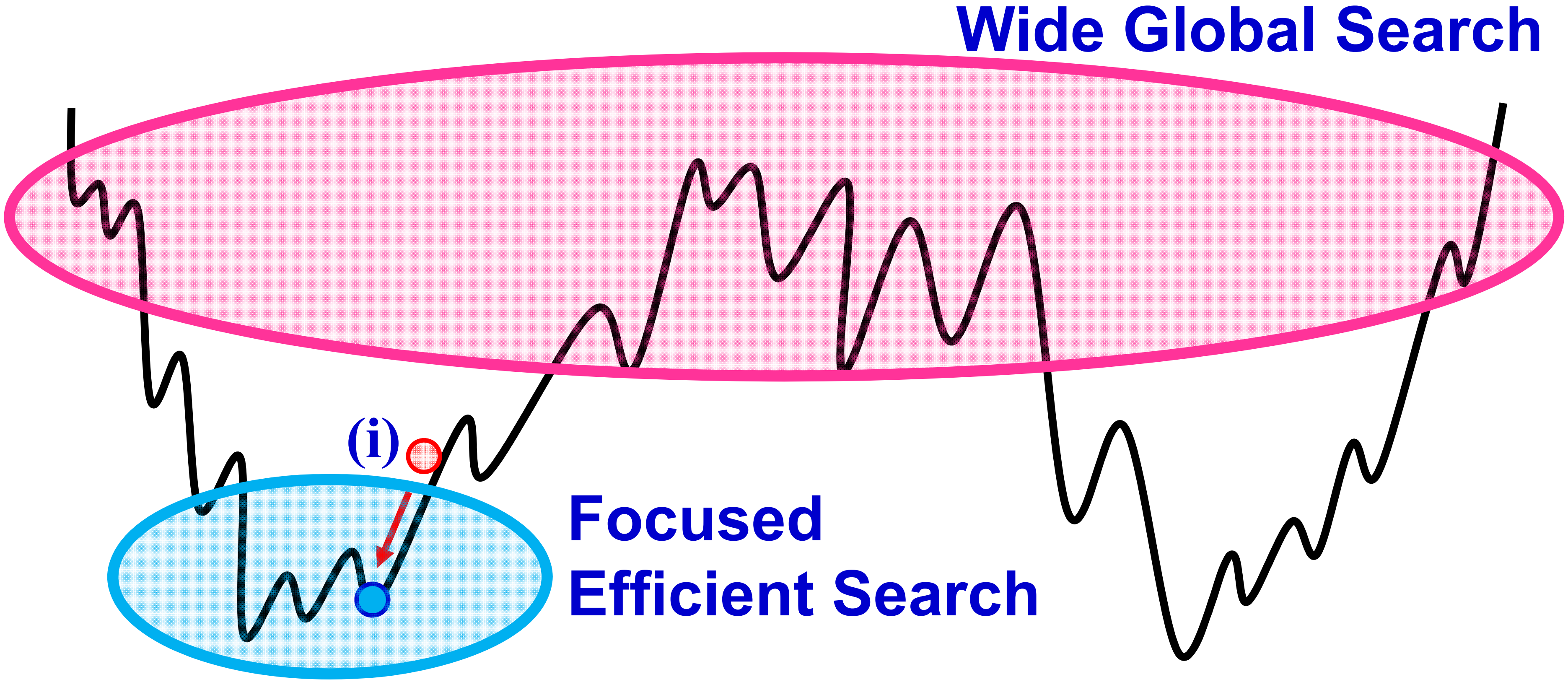
Focused: Narrow search region & Efficient search



The search of this region is difficult.

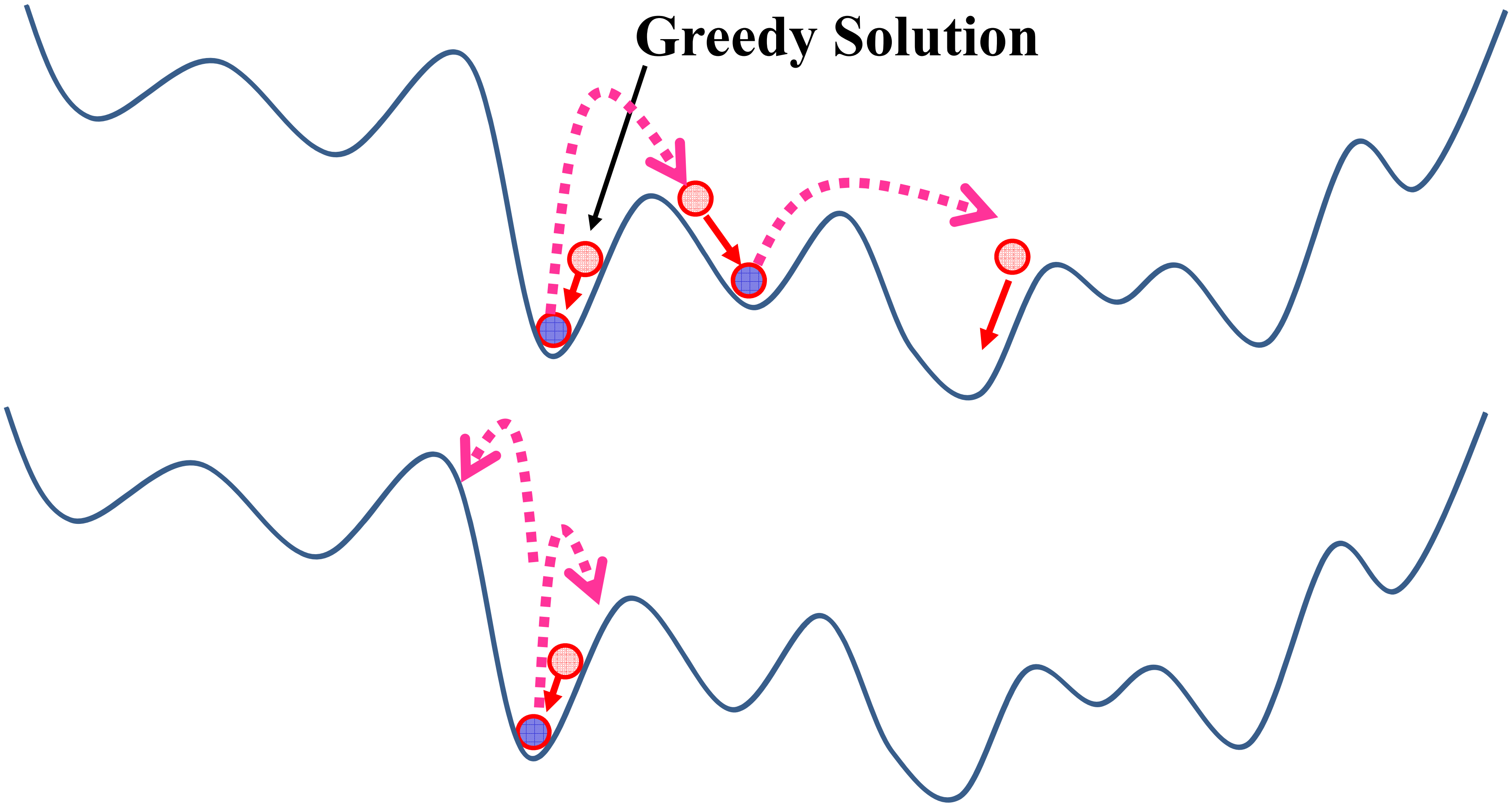
Optimization Algorithm Design:

Find a good balance between the wide global search and the focused efficient search (the good balance depends on the problem size and the available computation time)



General Guideline:

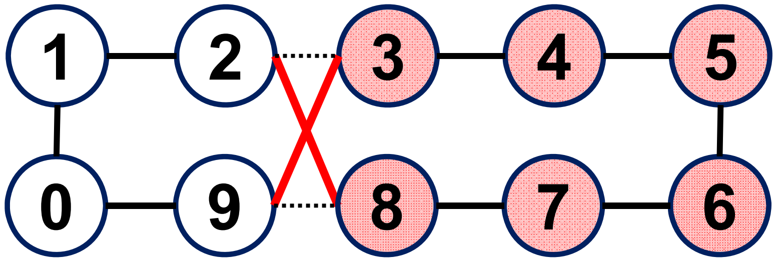
Generate good initial solutions which are not likely to go back to the same final solution.

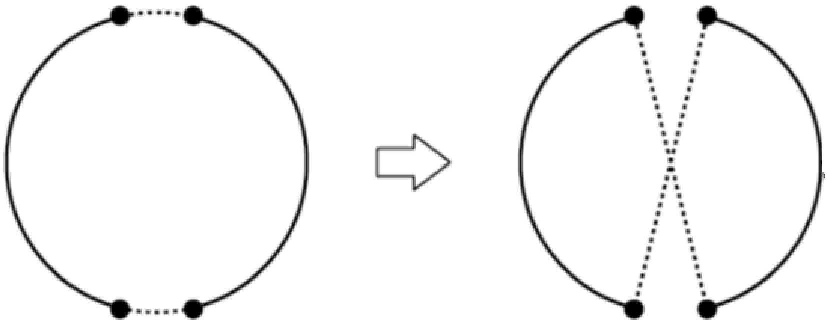


Lab Session Task 1

Local search

- Neighborhood structure: Inversion (two-edge change)





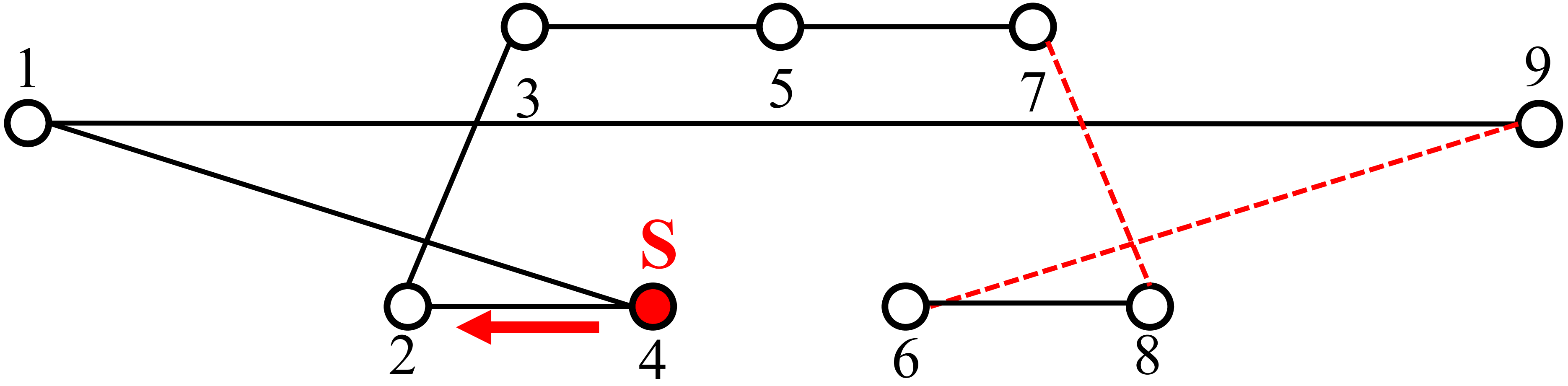
Creation of a new initial solution:

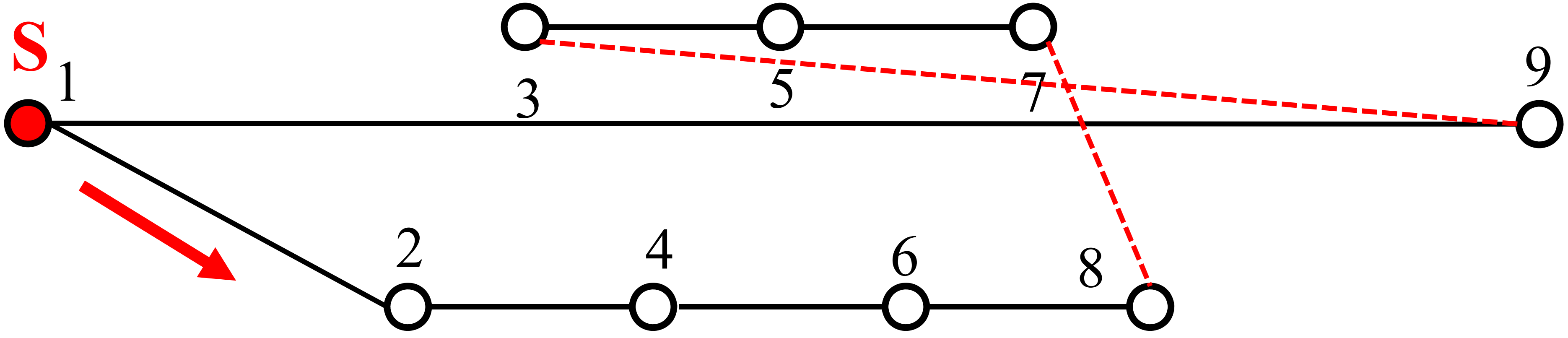
- 1st initial solution: Greedy solution

Task 1

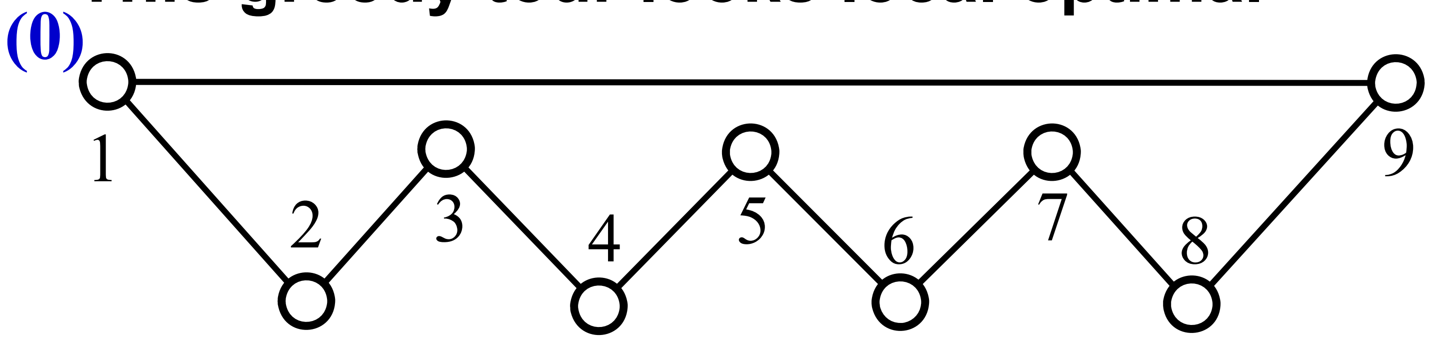
Create a TSP problem where a non-optimal greedy solution is obtained from an initial city and the obtained greedy solution cannot be improved by inversion-based local search. It is enough to show that one greedy solution cannot be improved. You do not have to show that any greedy solution from an arbitrary initial city cannot be improved.

TSP Problem: In many cases, greedy solutions can be improved by inversion-based local search.

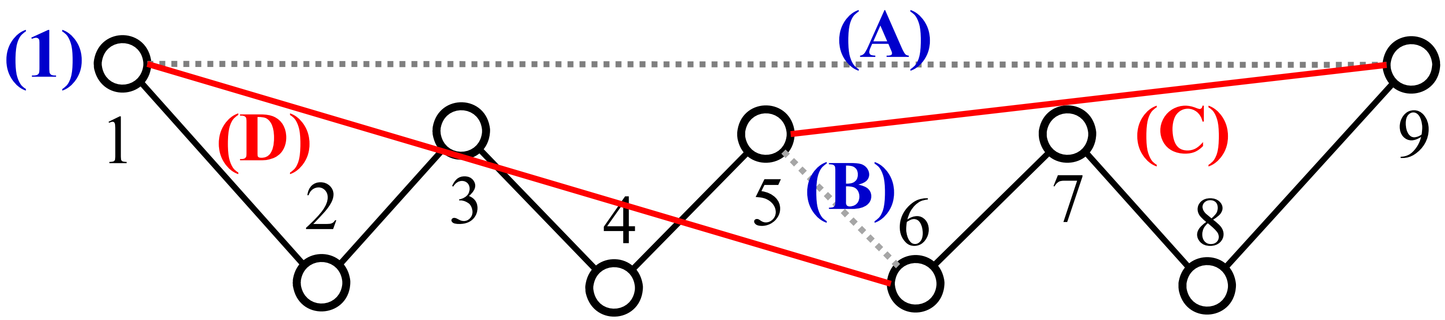




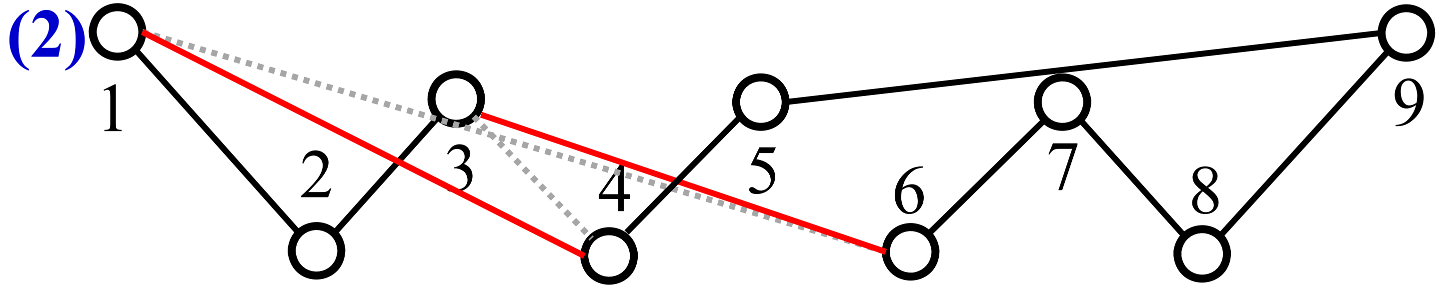
This greedy tour looks local optimal

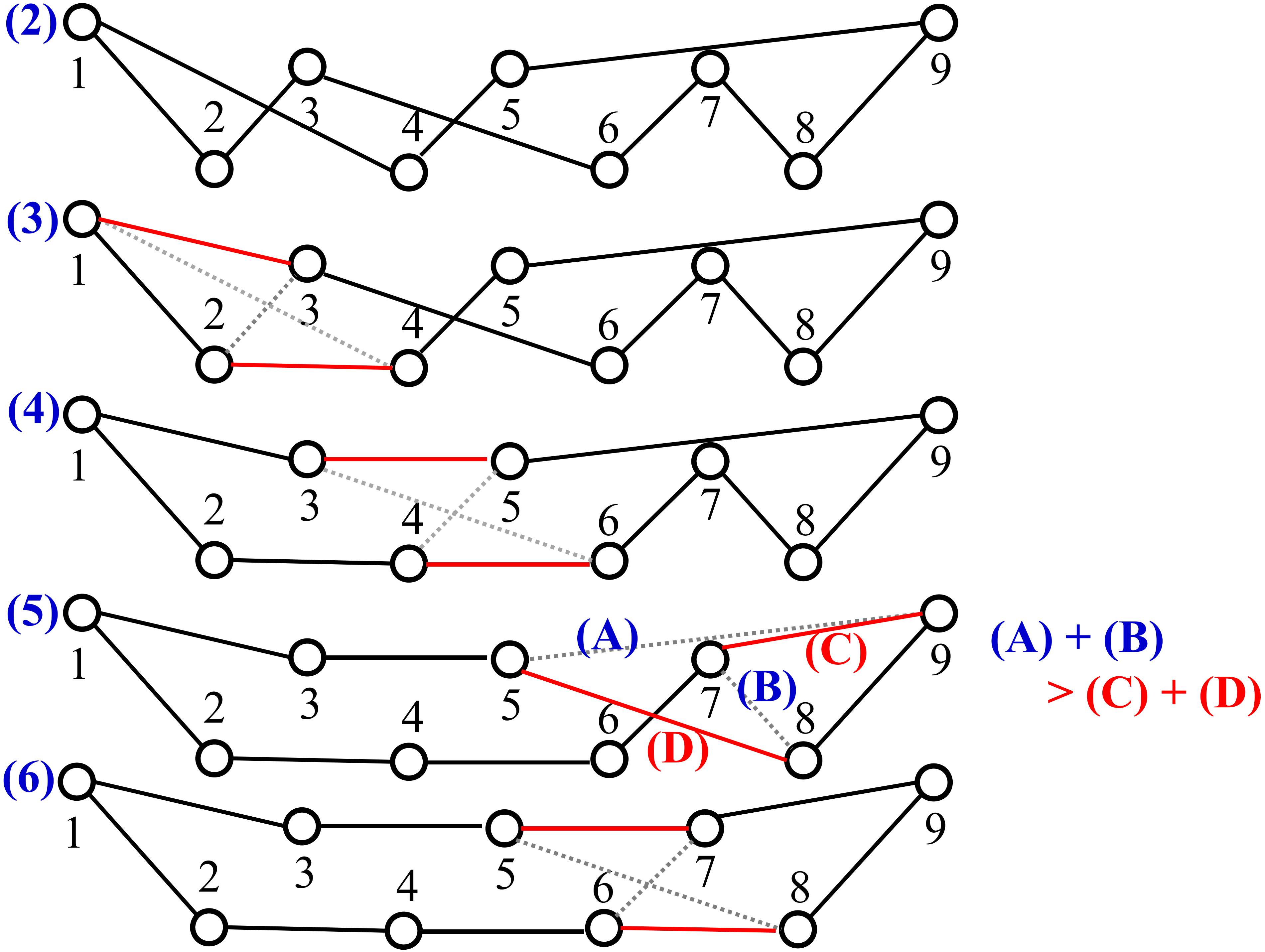


However, this can be improved by inversion-based LS.



(A) + (B)
> (C) + (D)

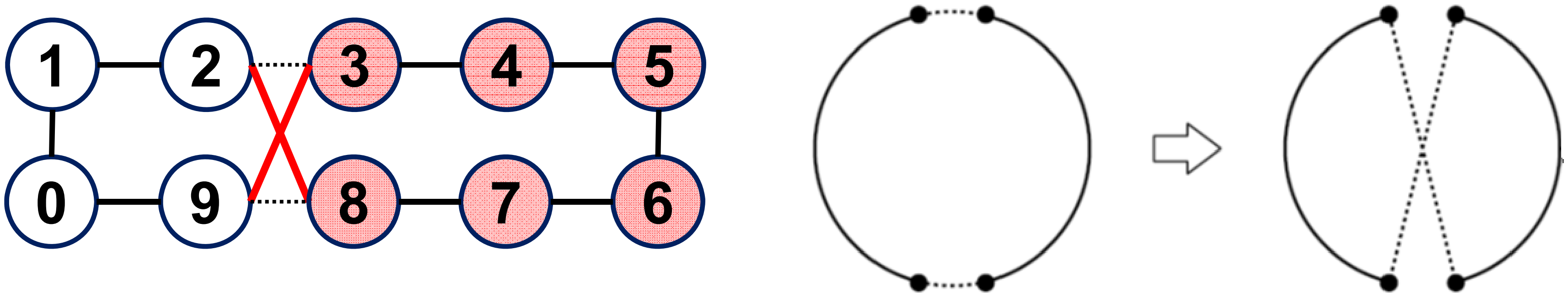



(2)
1
2
3
4
5
6
7
8
9
(3)
1
2
3
4
5
6
7
8
9
(4)
1
2
3
4
5
6
7
8
9
(5)
1
2
3
4
5
6
7
8
9
(A)
(B)
(C)
(D)
(A) + (B)
> (C) + (D)
(6)
1
2
3
4
5
6
7
8
9

Lab Session Task 2

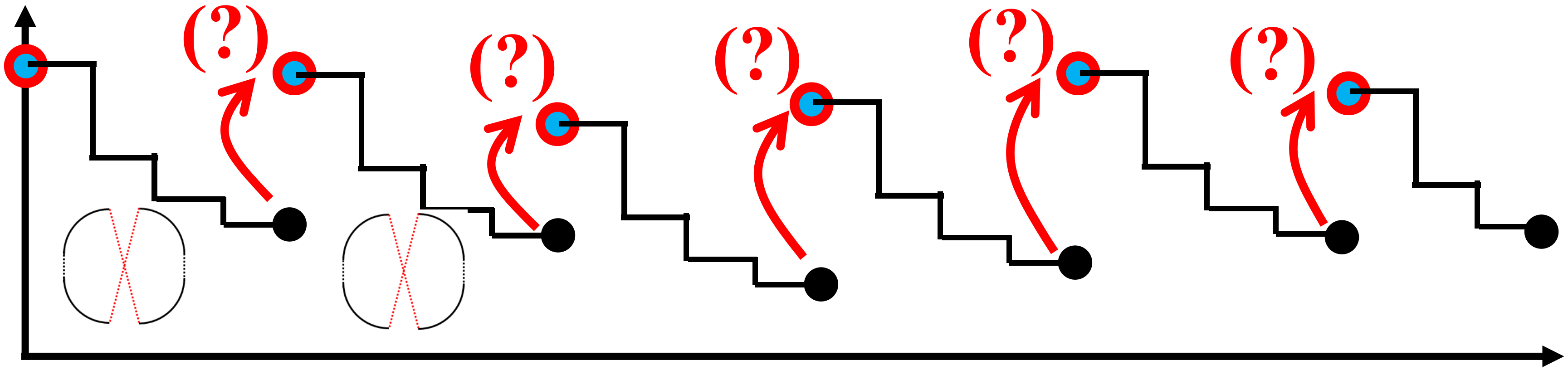
Local search

- Neighborhood structure: Inversion (two-edge change)

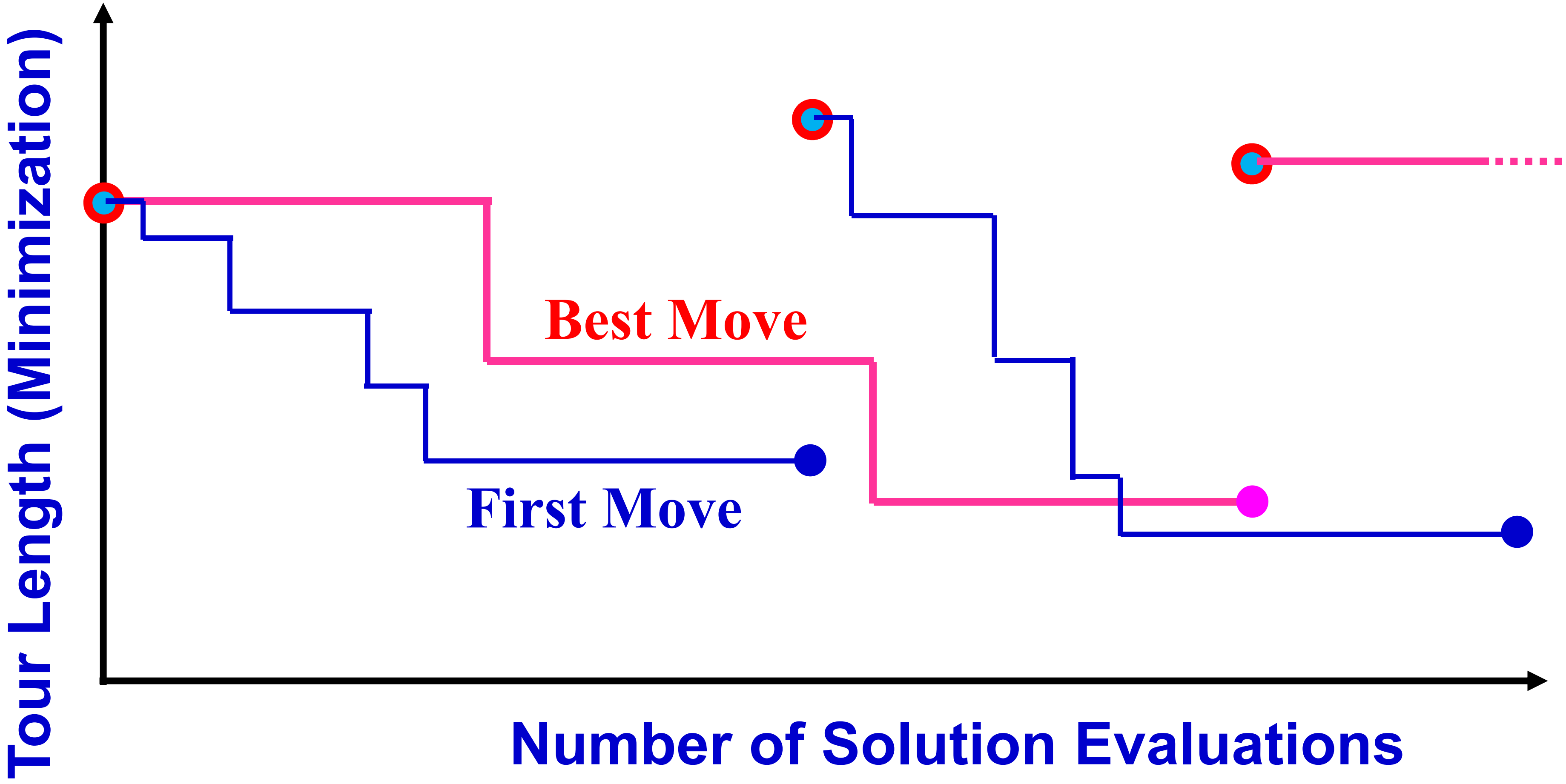


Creation of a new initial solution:

- 1st initial solution: Greedy solution
- Other initial solutions: Explain your own idea about how to create an initial solution for each local search run (Task 2).

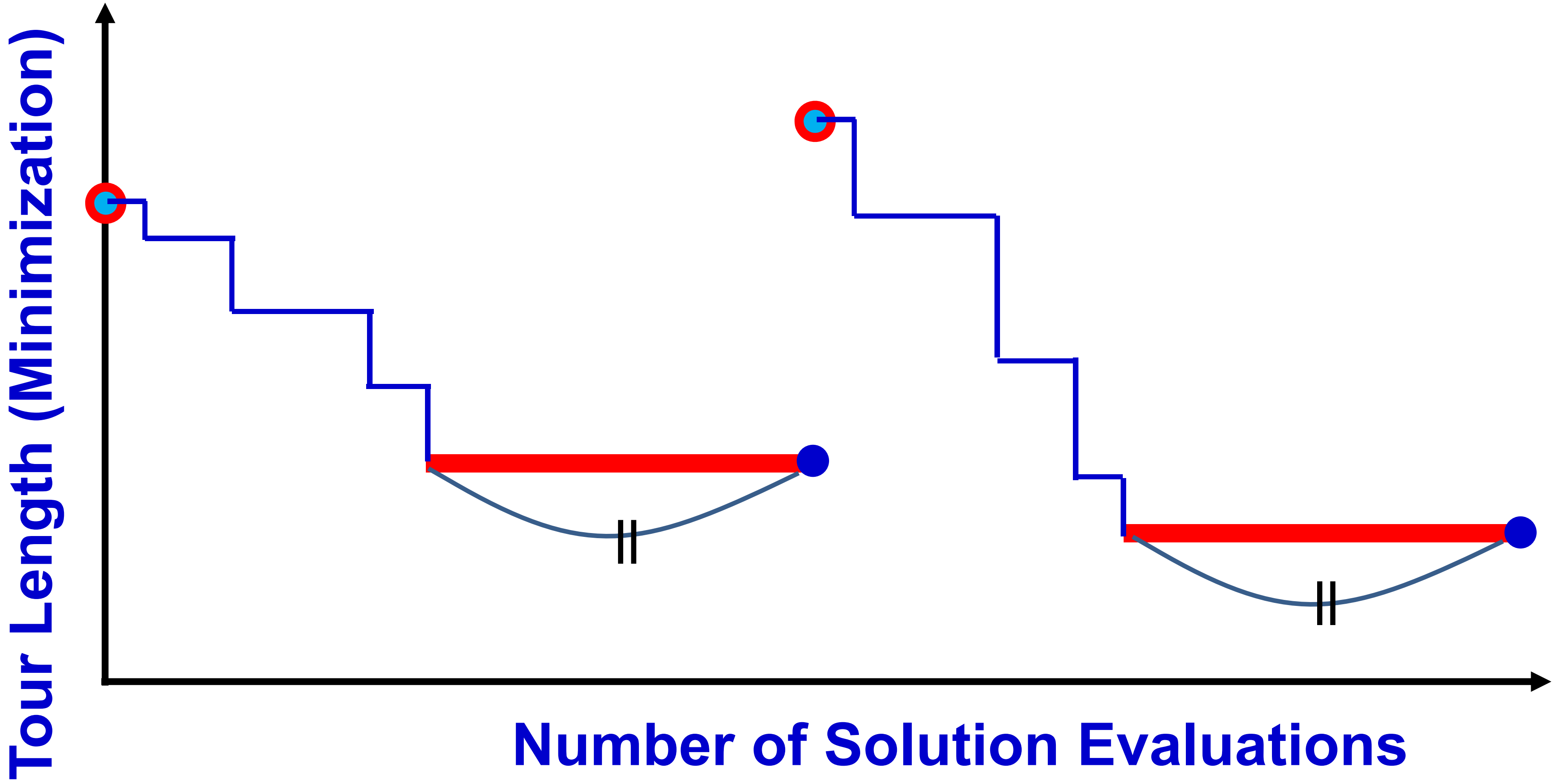


Best Move or First Move ?



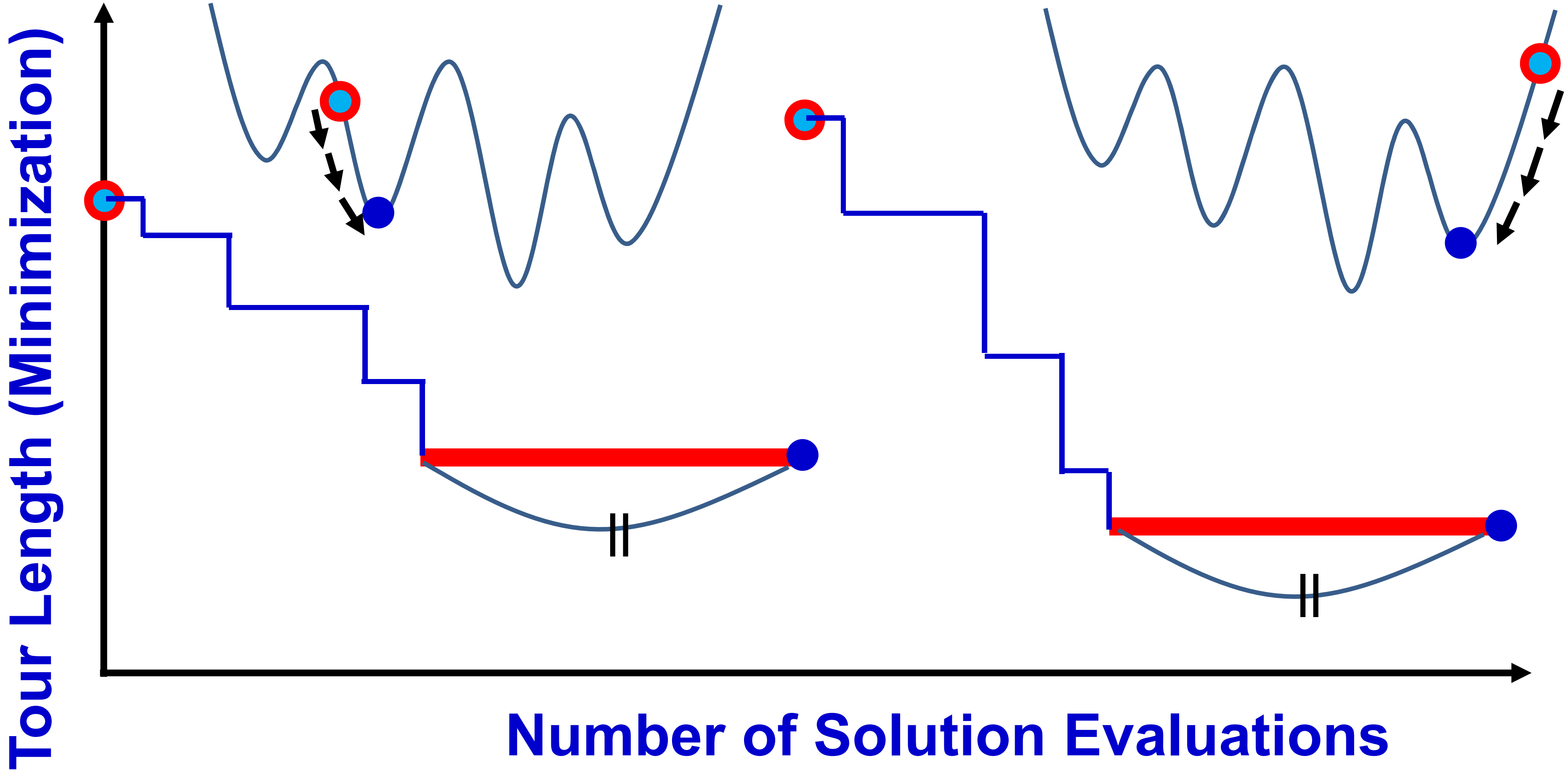
It may be a good idea to use the first move for examining more initial solutions (i.e., for finding more local solutions)

Early Termination



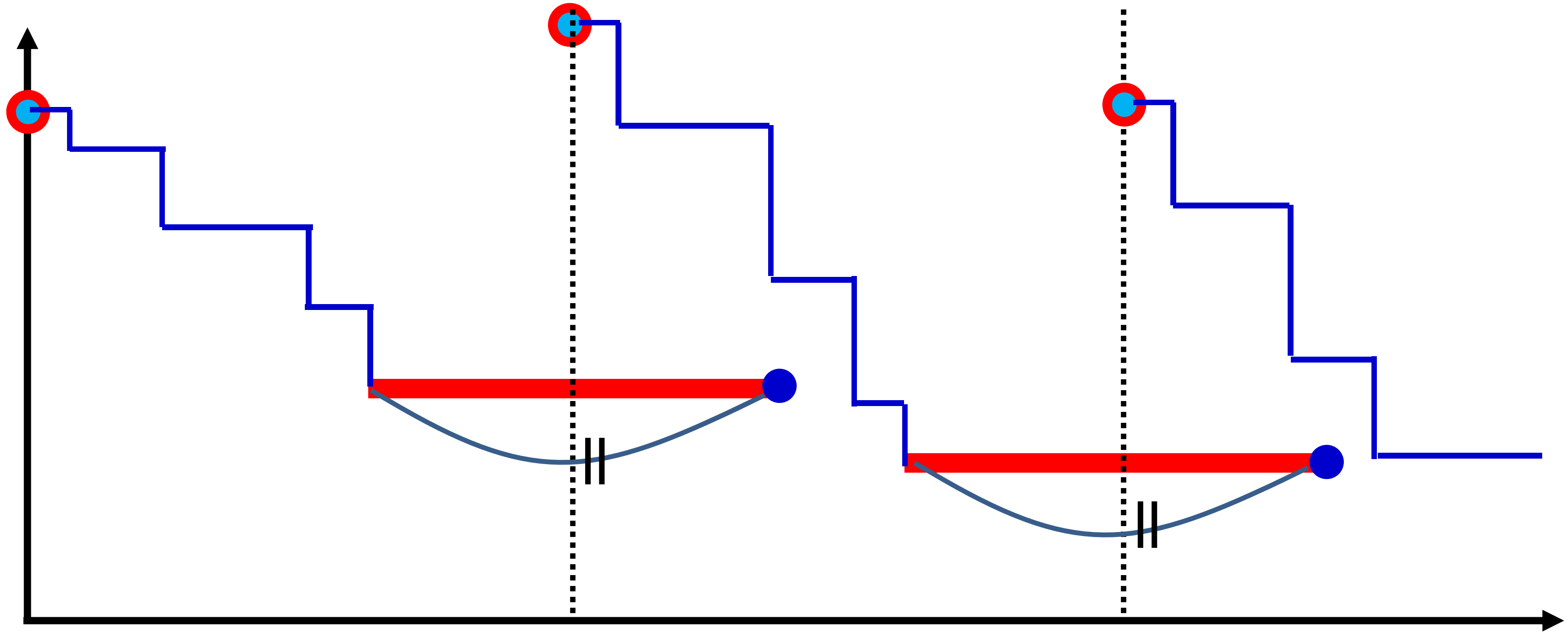
Even in the first move local search, all neighbors are examined before the termination of local search. This usually spends a long computation time with no performance improvement, which looks waste of time.

Early Termination



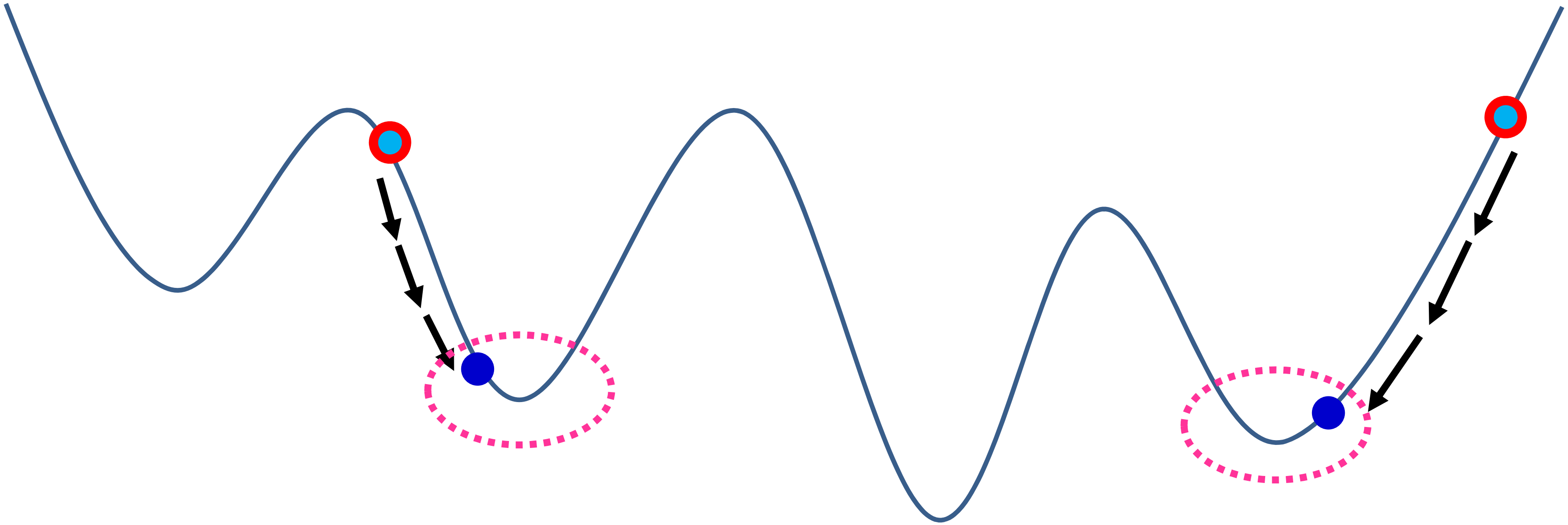
Even in the first move local search, all neighbors are examined before the termination of local search. This usually spends a long computation time with no performance improvement, which looks waste of time.

Early Termination

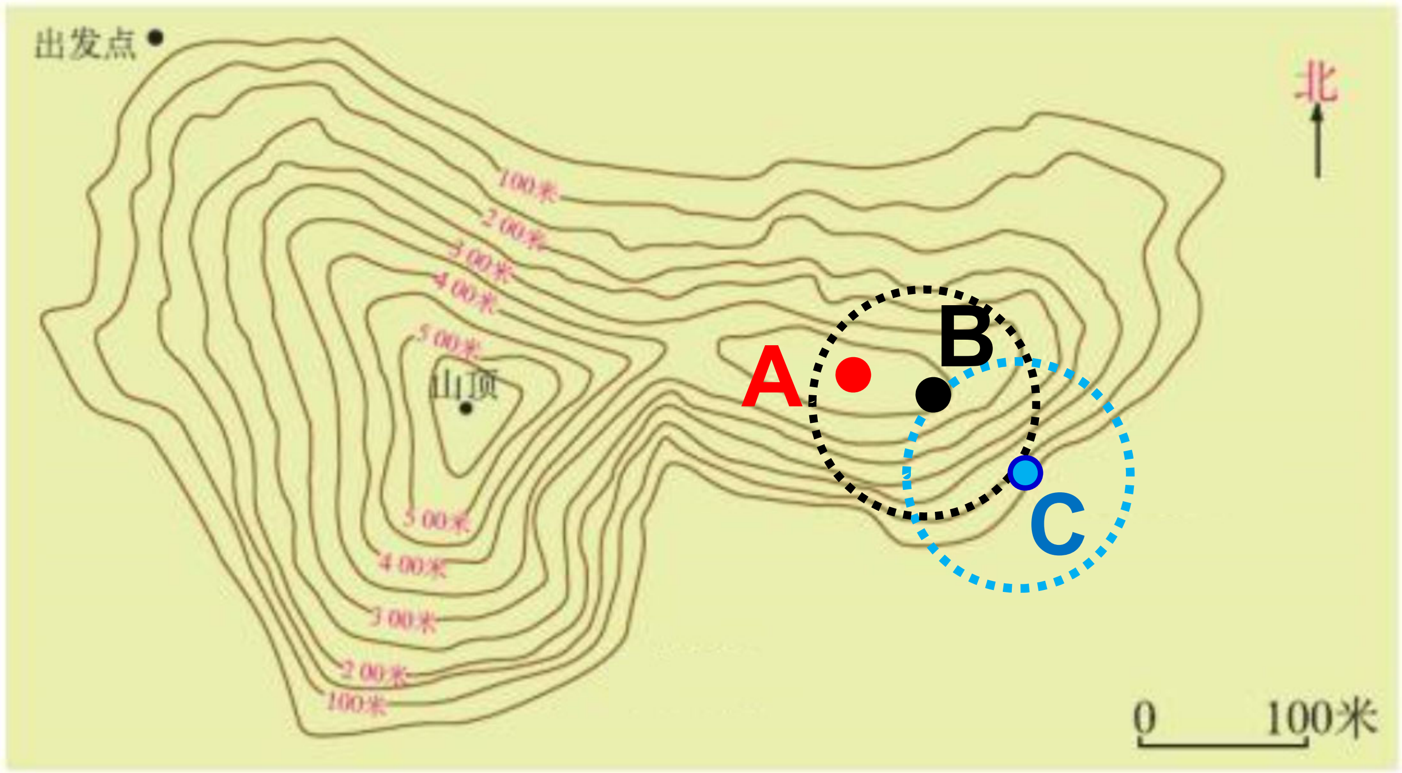


It may be a good idea for efficient search to terminate the local search before examining all neighbors (e.g., restart from a new initial solution after examining 100 neighbors or 50% of neighbors).

Early Termination



One negative effect is that the local search will terminate before finding a local solution. The question is which is a better search strategy between (i) to carefully search for a local solution around the current solution, and (ii) to quickly search for better solutions from many initial solutions. Usually, the improvement by local search around a local solution is not large. Thus, (ii) is more efficient in many cases.



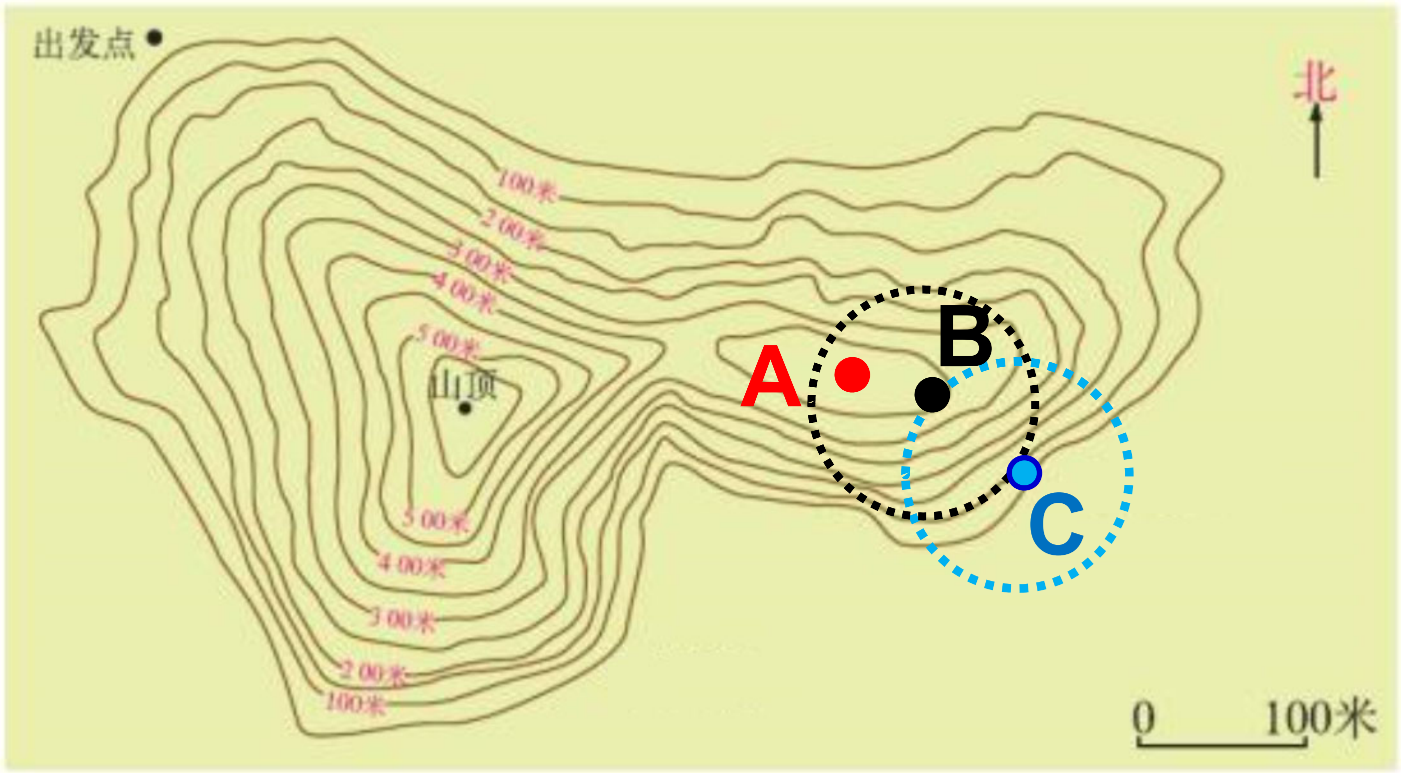
Around the local solution A: Current Solution B

(i) It is not easy to find a better solution

==> Early termination before finding the local solution.

(ii) The improvement by local search is small.

==> Early termination is not a big problem.



Around the local solution A: Current Solution B

(i) It is not easy to find a better solution

==> Early termination before finding the local solution.

(ii) The improvement by local search is small.

==> Early termination is not a big problem.

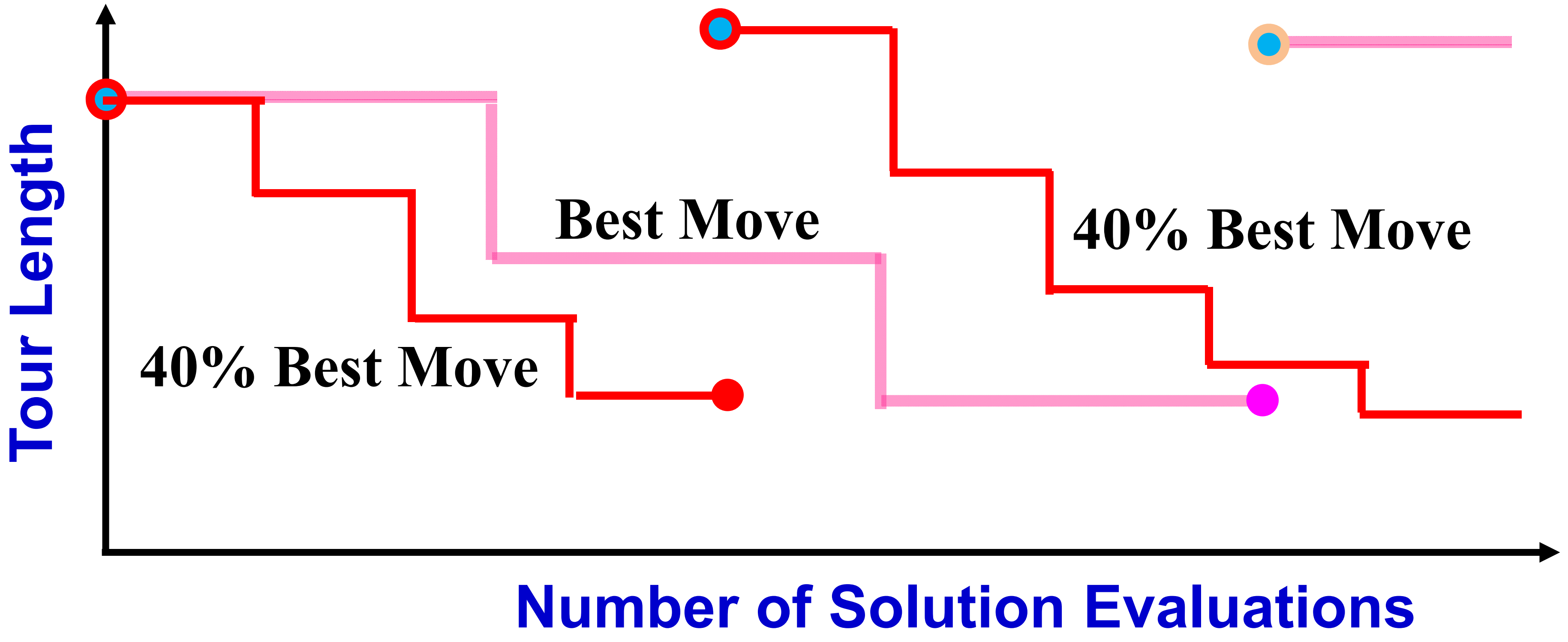
At solution C, it is easy to find a better solution.

Best Move with Early Termination

(i) Examine α% of neighbors.

(ii) Choose the best neighbor among them.

(iii) If the best neighbor is better than the current solution, move to the best neighbor. Otherwise, restart the local search from a new initial solution.



Local Search

Implementation Issues

(1) Specification of an initial solution

(2) Specification of a neighborhood structure

(3) Choice between the first move and the best move

⇩

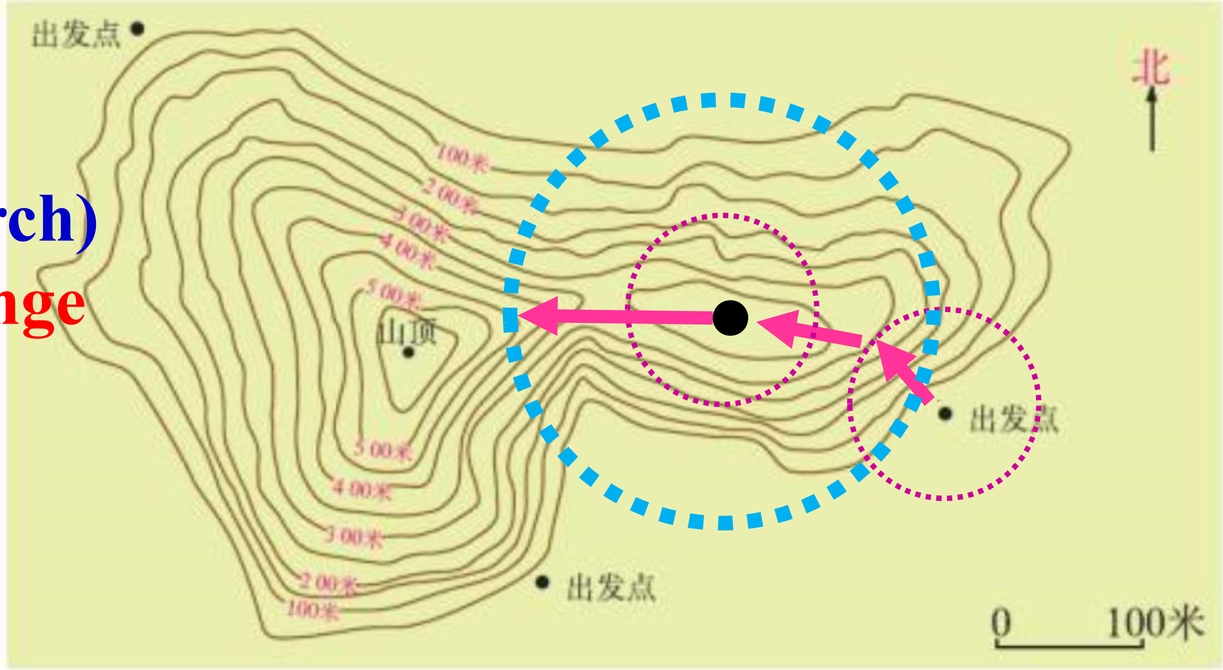
Local Solution

Next Step

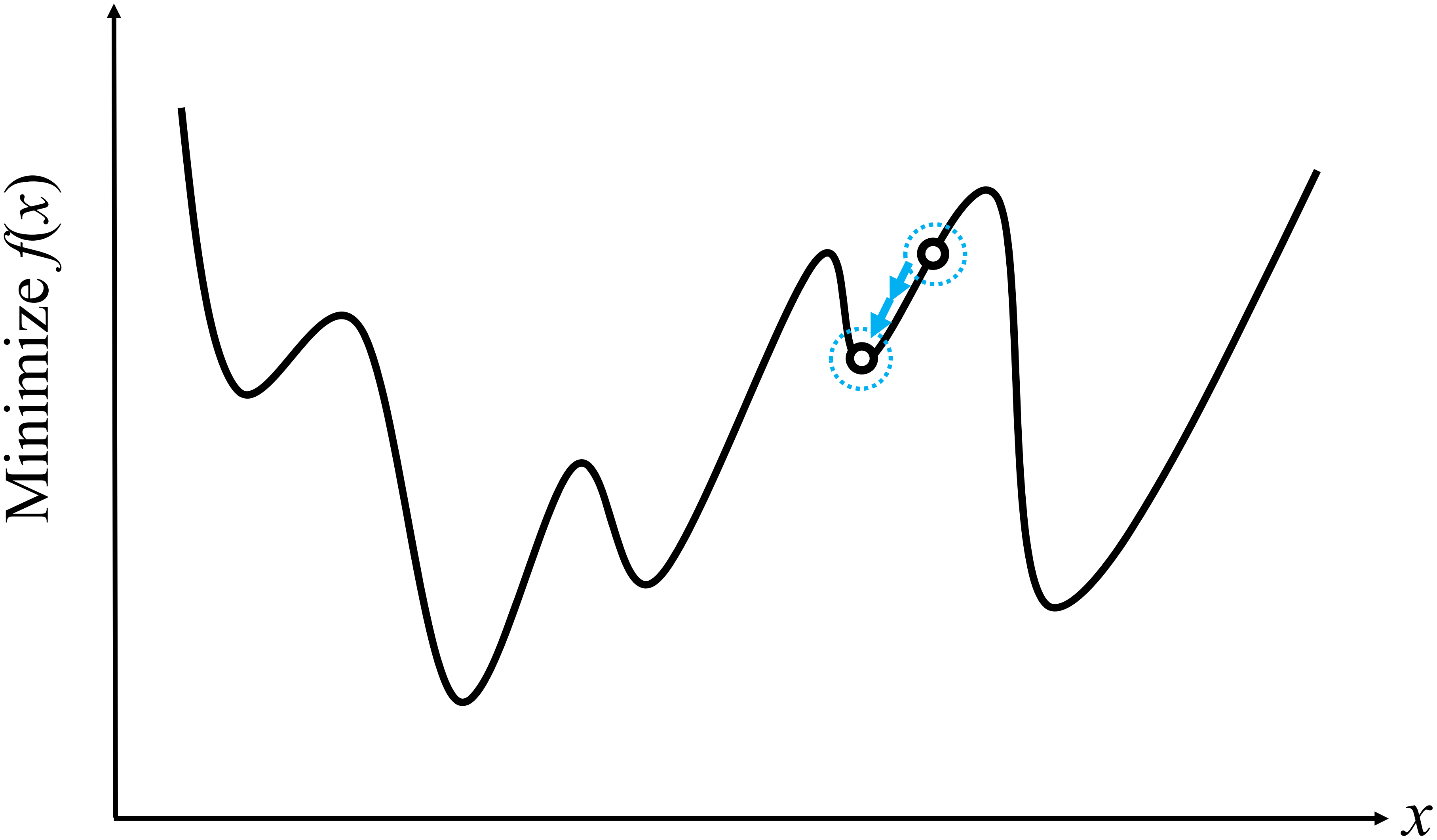
- Restart

(Iterated Local Search)

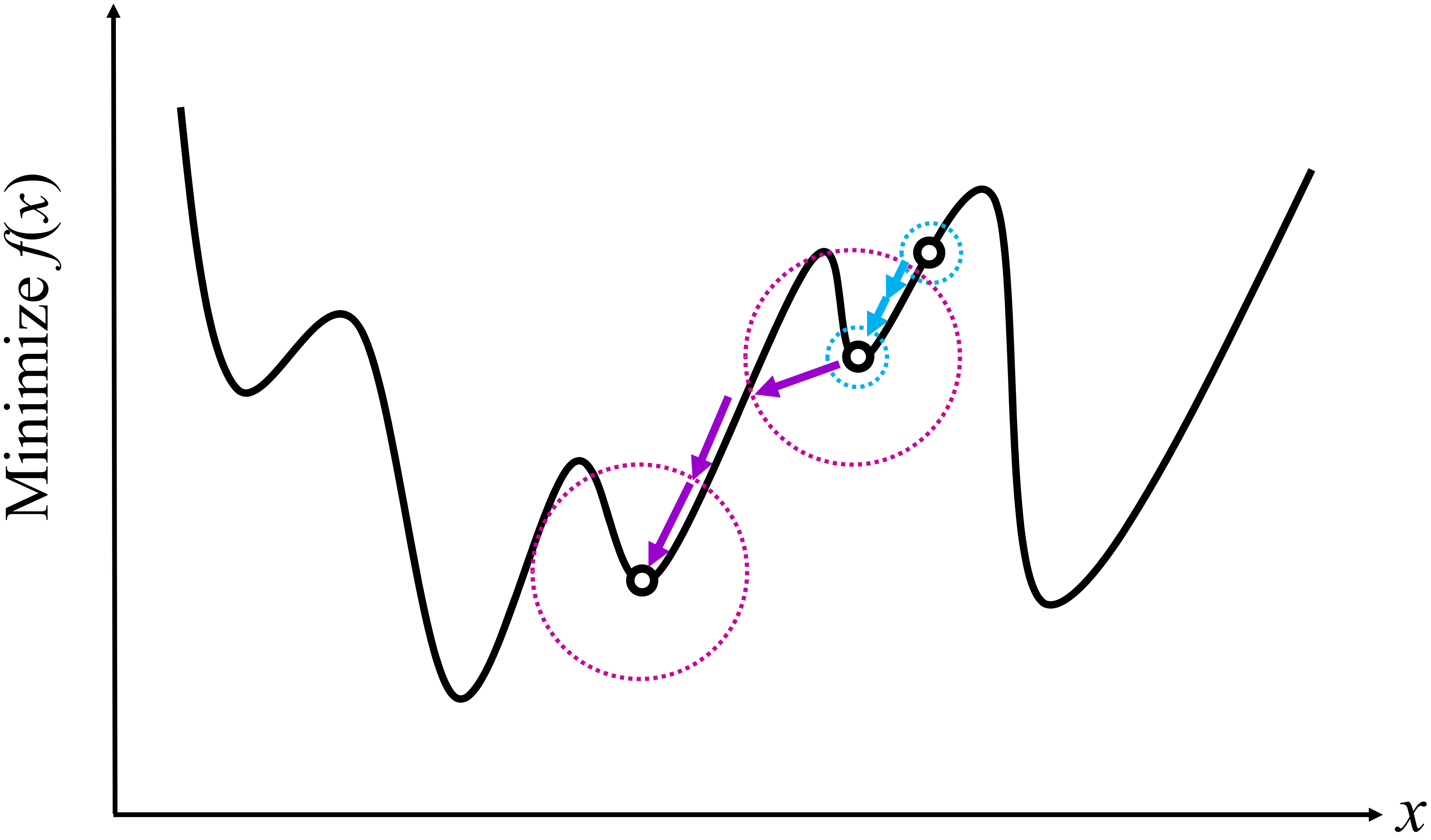
- Neighborhood Change



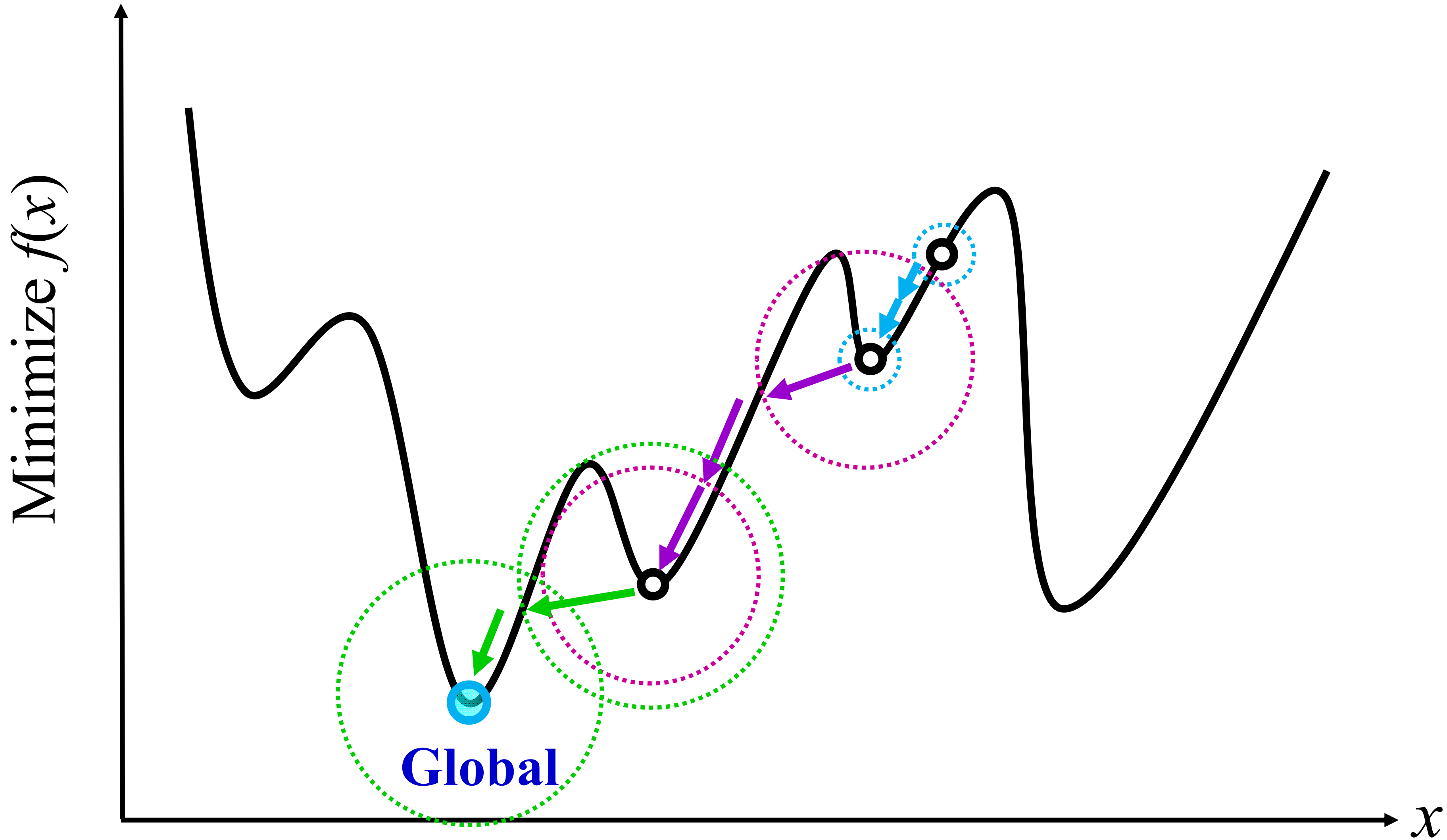
Local Search with Neighborhood Change

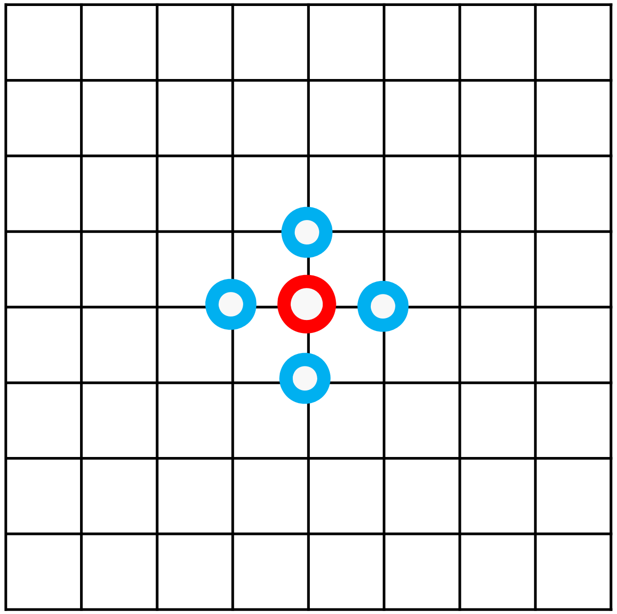


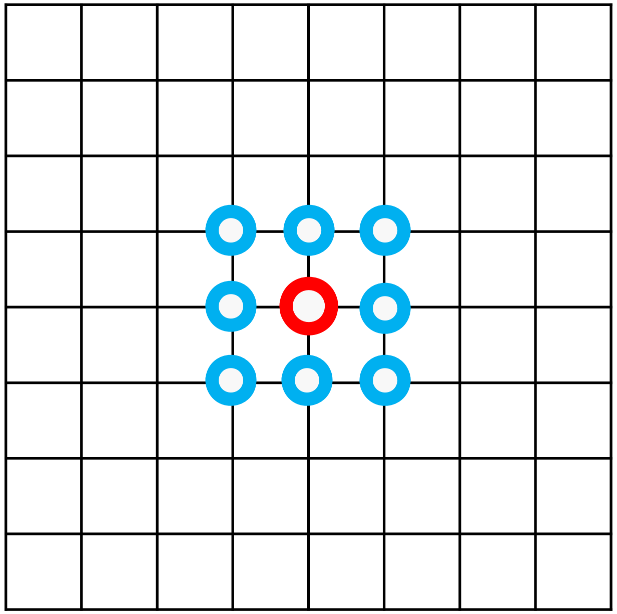
Local Search with Neighborhood Change



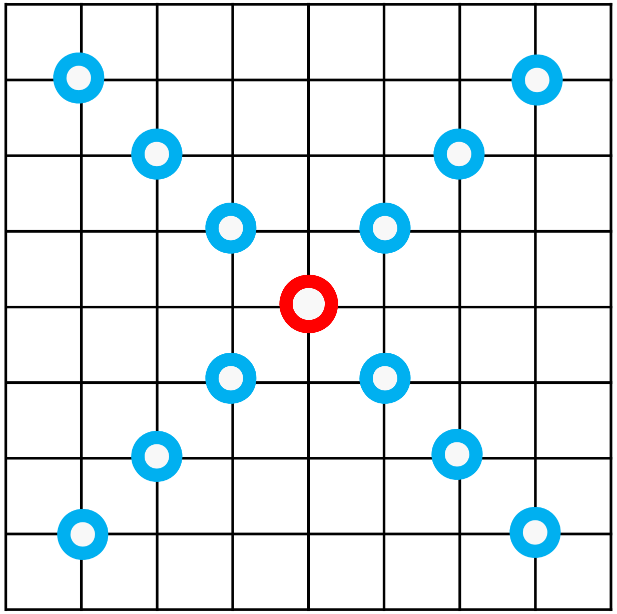
Local Search with Neighborhood Change

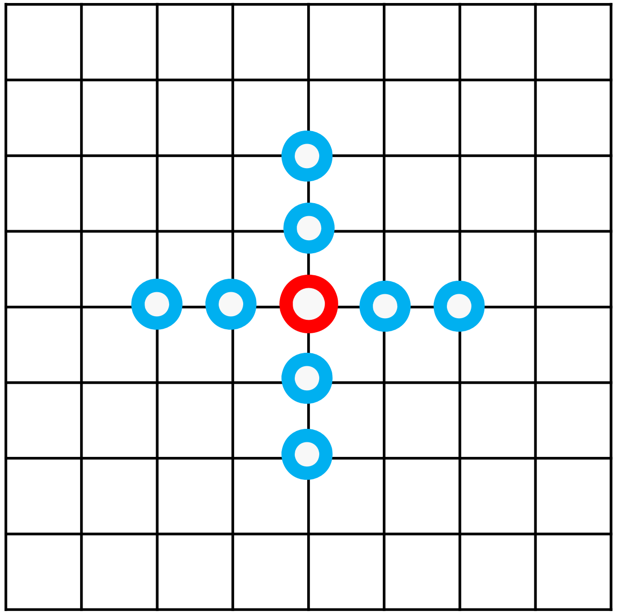






Local Search with Neighborhood Change





Adjacent two-city change

Size: n

0	1	2	3	4	5	7	6	8	9

Arbitrary two-city change

Size: $n(n-1)/2$

0	1	2	3	4	5	9	7	8	6

Insertion (Shift)

Size: $n(n-3)$

0	1	2	3	4	5	7	8	6	9

Inversion (Two-edge change)

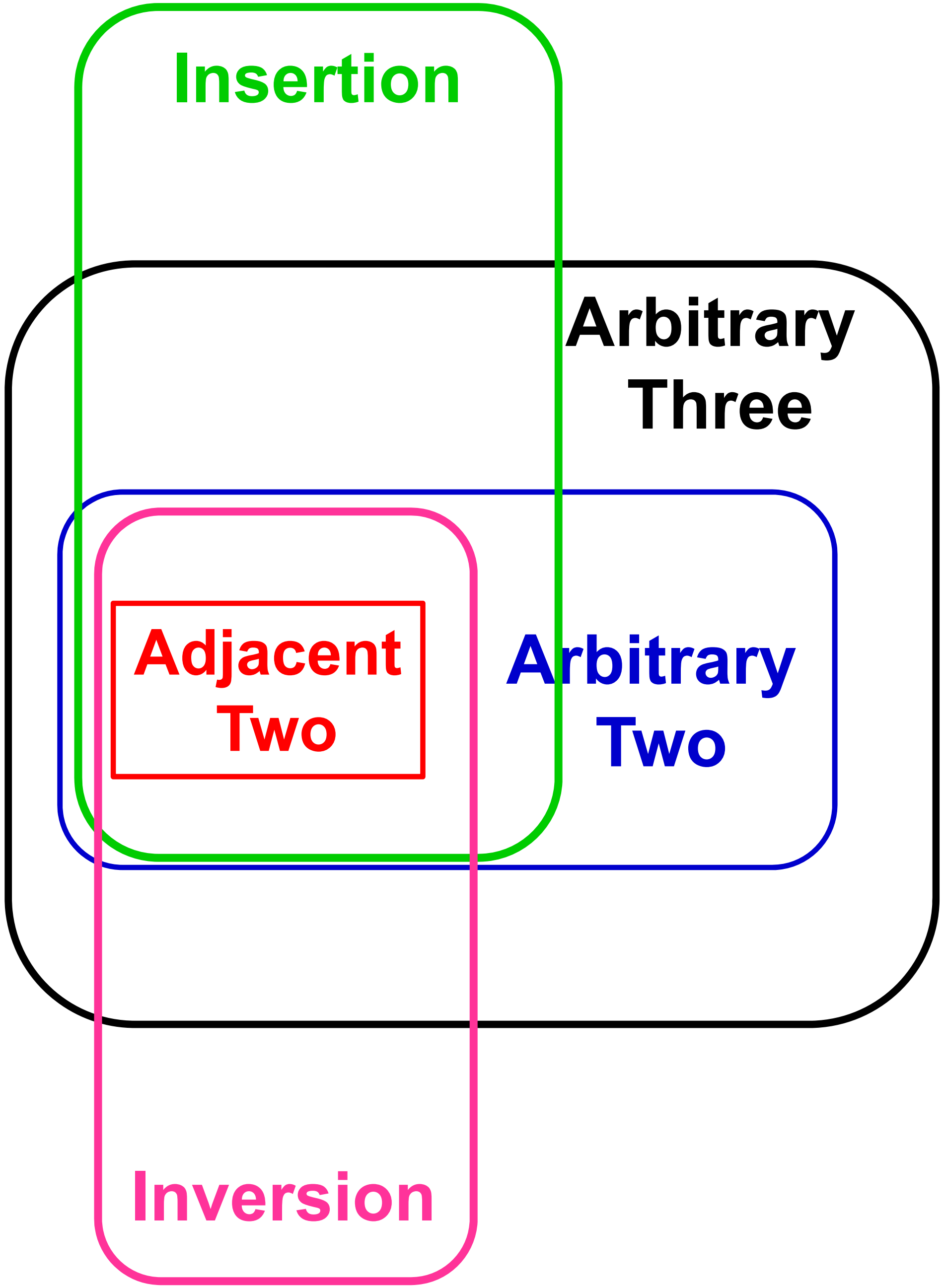
Size: $n(n-3)/2$

0	1	2	8	7	6	5	4	3	9

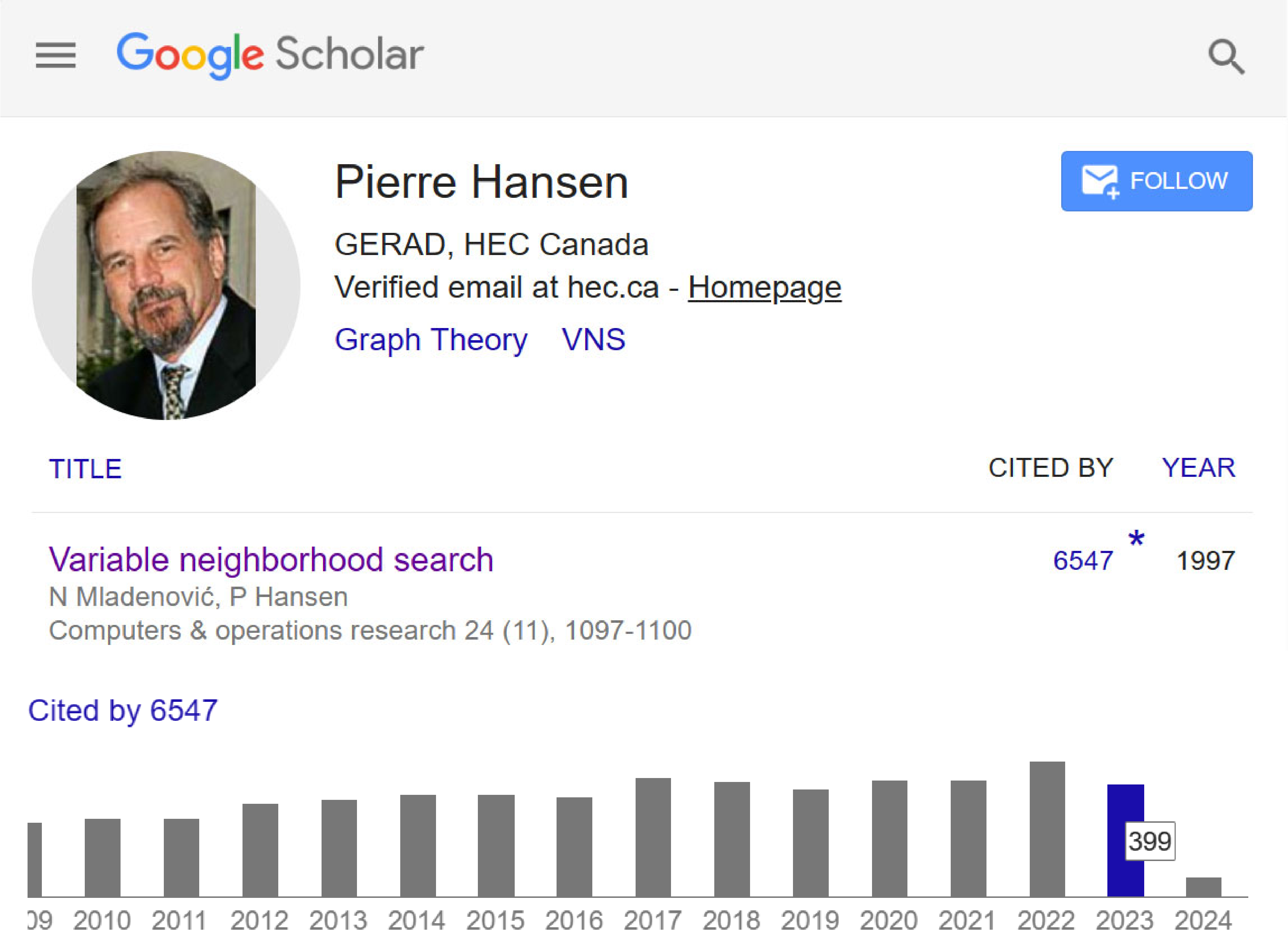
Arbitrary three-city change

Size: $n(n-1)(2n-1)/6$

0	1	2	6	4	5	9	7	8	3



Variable Neighborhood Search



Google Scholar

Pierre Hansen

GERAD, HEC Canada

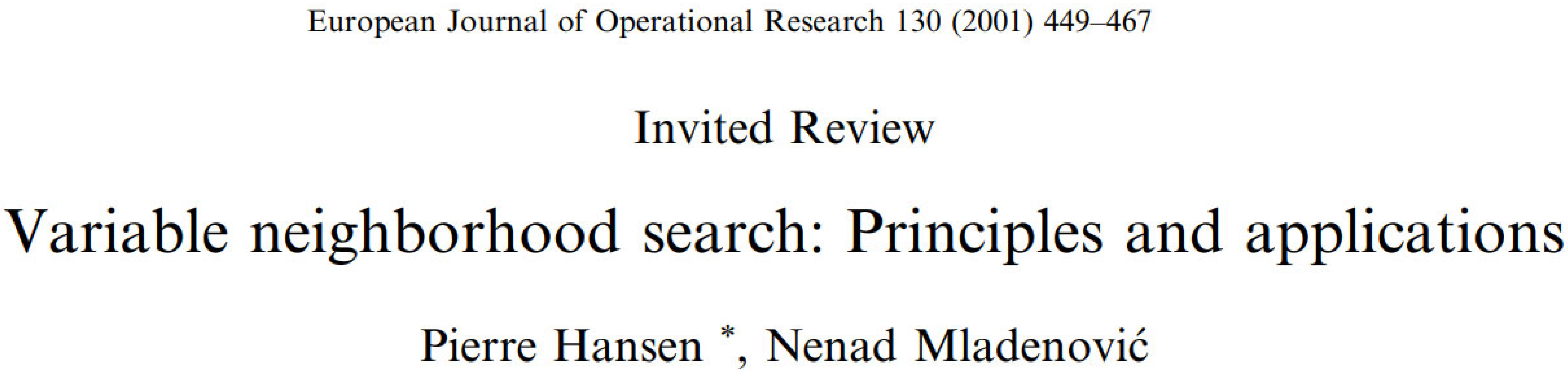
Verified email at hec.ca - Homepage

Graph Theory VNS

TITLE	CITED BY	YEAR
Variable neighborhood search N Mladenović, P Hansen Computers & operations research 24 (11), 1097-1100	6547 *	1997

Cited by 6547

Variable Neighborhood Search Algorithm



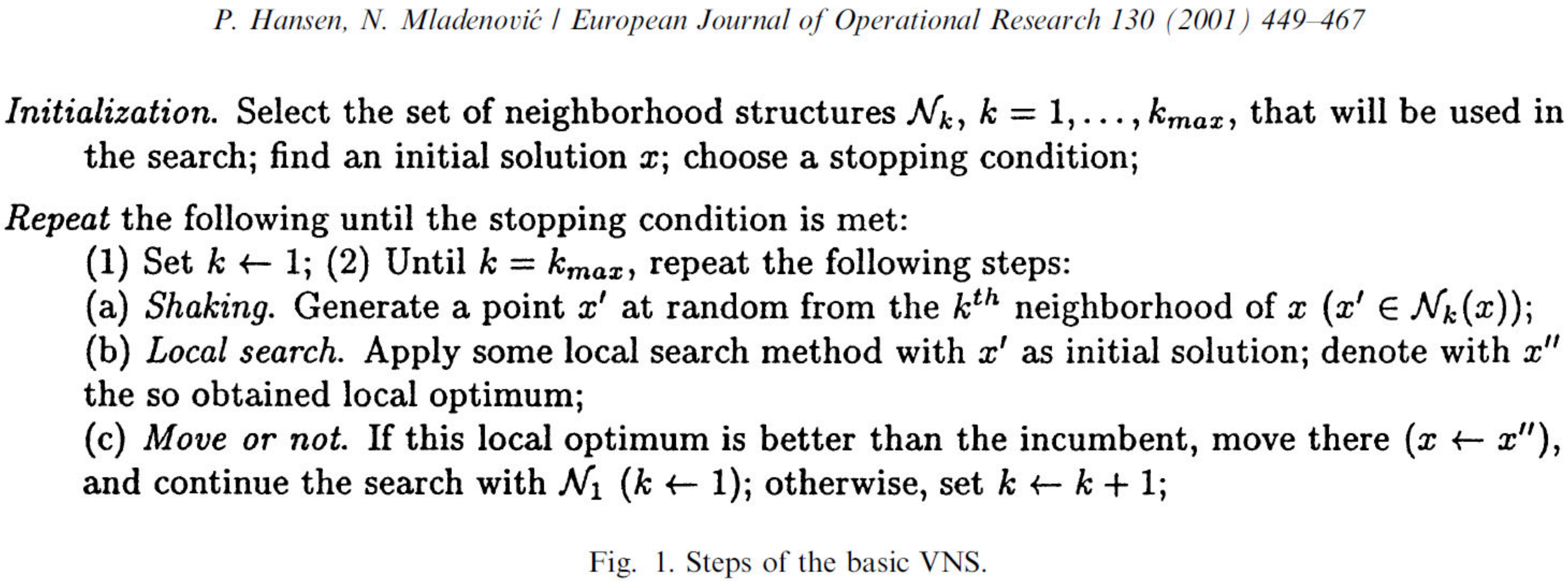
European Journal of Operational Research 130 (2001) 449–467

Invited Review

Variable neighborhood search: Principles and applications

Pierre Hansen *, Nenad Mladenović

Variable Neighborhood Search Algorithm



P. Hansen, N. Mladenović / European Journal of Operational Research 130 (2001) 449–467

Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, that will be used in the search; find an initial solution x; choose a stopping condition;

Repeat the following until the stopping condition is met:

(1) Set $k \leftarrow 1$; (2) Until $k = k_{max}$, repeat the following steps:

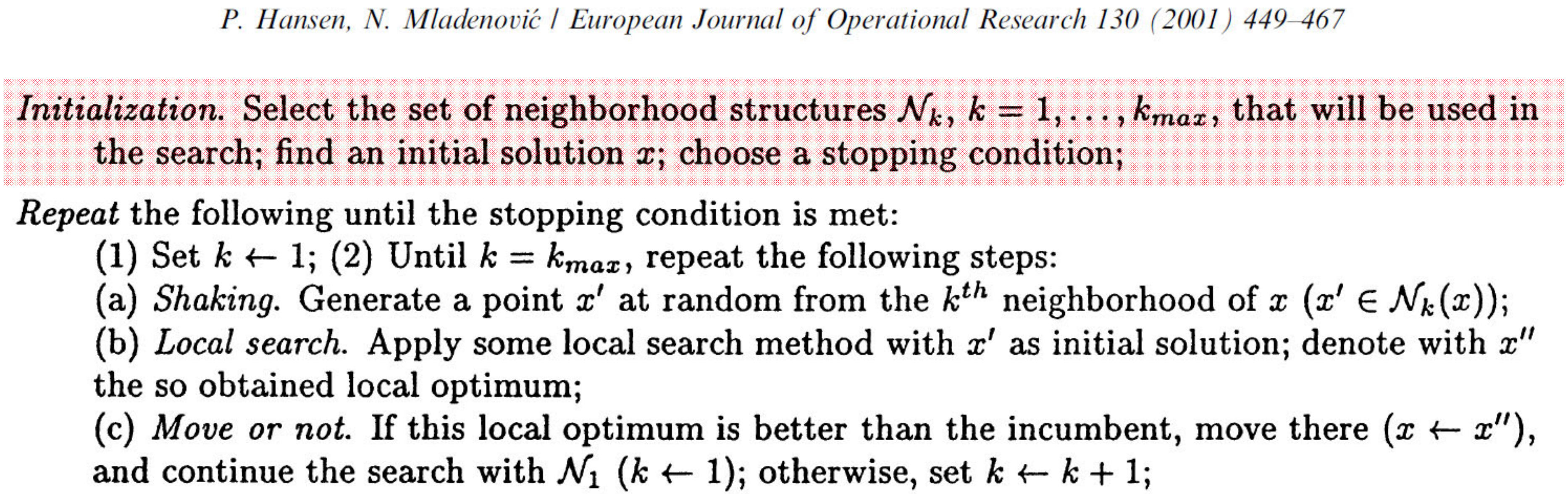
(a) *Shaking.* Generate a point x' at random from the k^{th} neighborhood of x ($x' \in \mathcal{N}_k(x)$);

(b) *Local search.* Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;

(c) *Move or not.* If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;

Fig. 1. Steps of the basic VNS.

Variable Neighborhood Search Algorithm



P. Hansen, N. Mladenović / European Journal of Operational Research 130 (2001) 449–467

Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, that will be used in the search; find an initial solution x; choose a stopping condition;

Repeat the following until the stopping condition is met:

(1) Set $k \leftarrow 1$; (2) Until $k = k_{max}$, repeat the following steps:

(a) *Shaking.* Generate a point x' at random from the k^{th} neighborhood of x ($x' \in \mathcal{N}_k(x)$);

(b) *Local search.* Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;

(c) *Move or not.* If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;

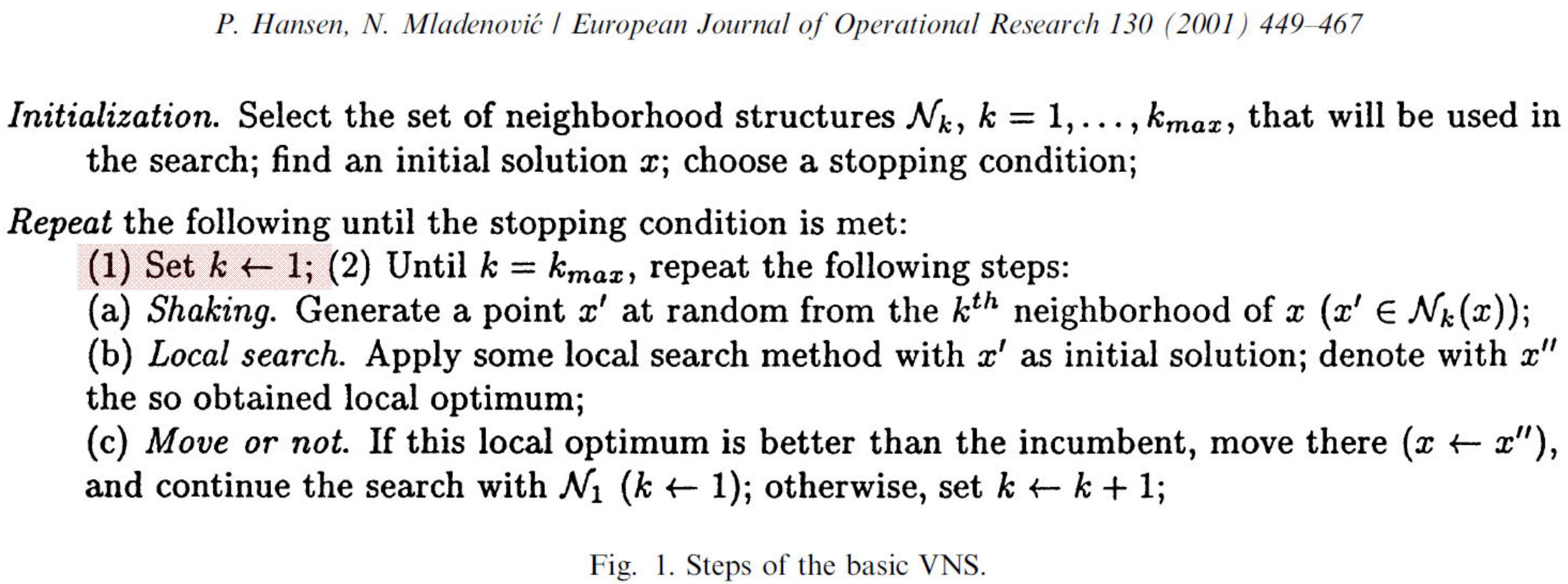
Fig. 1. Steps of the basic VNS.

Neighborhood Structures: $\mathcal{N}_k$, $k = 1, 2, \ldots, k_{max}$

$\mathcal{N}_{k+1}$ is larger than $\mathcal{N}_k$, $k = 1, 2, \ldots, k_{max} - 1$.

In some cases, $\mathcal{N}_k \subset \mathcal{N}_{k+1}$, $k = 1, 2, \ldots, k_{max} - 1$.

Variable Neighborhood Search Algorithm



P. Hansen, N. Mladenović / European Journal of Operational Research 130 (2001) 449–467

Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, that will be used in the search; find an initial solution x; choose a stopping condition;

Repeat the following until the stopping condition is met:

(1) Set $k \leftarrow 1$; (2) Until $k = k_{max}$, repeat the following steps:

(a) *Shaking.* Generate a point x' at random from the k^{th} neighborhood of x ($x' \in \mathcal{N}_k(x)$);

(b) *Local search.* Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;

(c) *Move or not.* If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;

Fig. 1. Steps of the basic VNS.

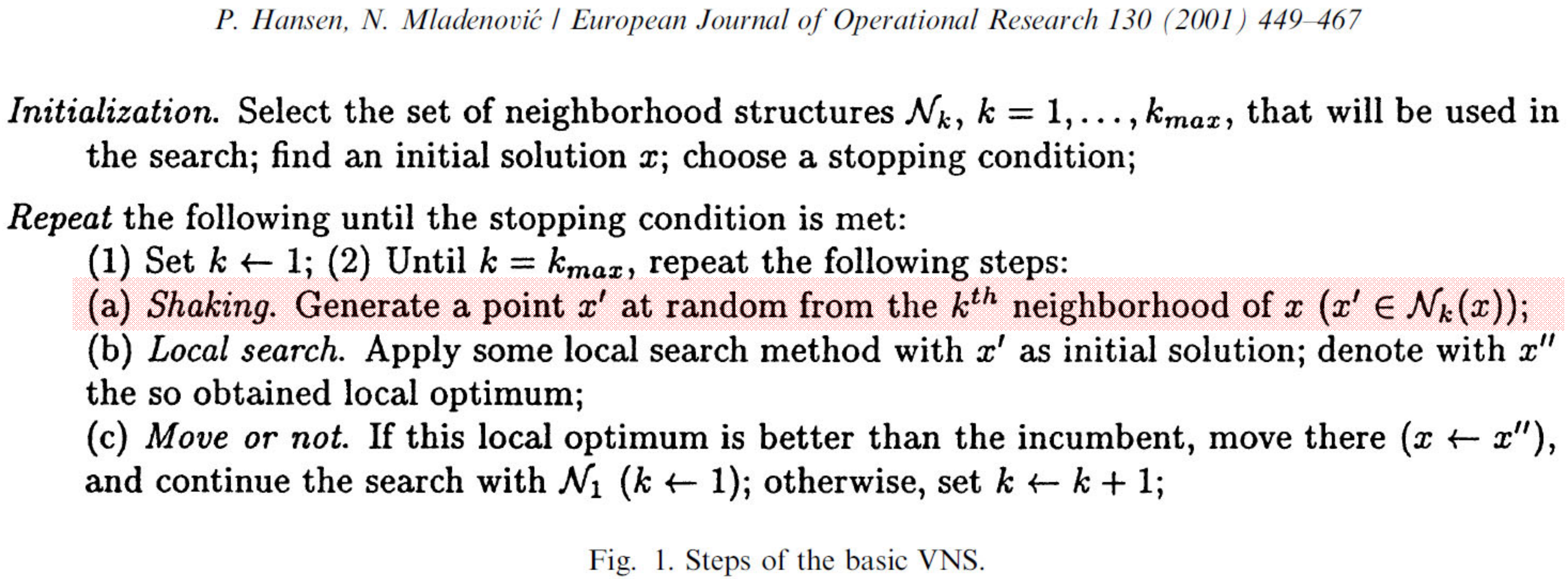
Neighborhood Structures: $\mathcal{N}_k$, $k = 1, 2, \ldots, k_{max}$

$\mathcal{N}_{k+1}$ is larger than $\mathcal{N}_k$, $k = 1, 2, \ldots, k_{max}-1$.

In some cases, $\mathcal{N}_k \subset \mathcal{N}_{k+1}$, $k = 1, 2, \ldots, k_{max}-1$.

$\mathcal{N}_1$ is used first, then $\mathcal{N}_2$, $\mathcal{N}_3$, ...

Basic Variable Neighborhood Search (VNS) Algorithm



P. Hansen, N. Mladenović / European Journal of Operational Research 130 (2001) 449–467

Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, that will be used in the search; find an initial solution x; choose a stopping condition;

Repeat the following until the stopping condition is met:

(1) Set $k \leftarrow 1$; (2) Until $k = k_{max}$, repeat the following steps:

(a) *Shaking.* Generate a point x' at random from the k^{th} neighborhood of x ($x' \in \mathcal{N}_k(x)$);

(b) *Local search.* Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;

(c) *Move or not.* If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;

Fig. 1. Steps of the basic VNS.

Neighborhood Structures: $\mathcal{N}_k$, $k = 1, 2, \ldots, k_{max}$

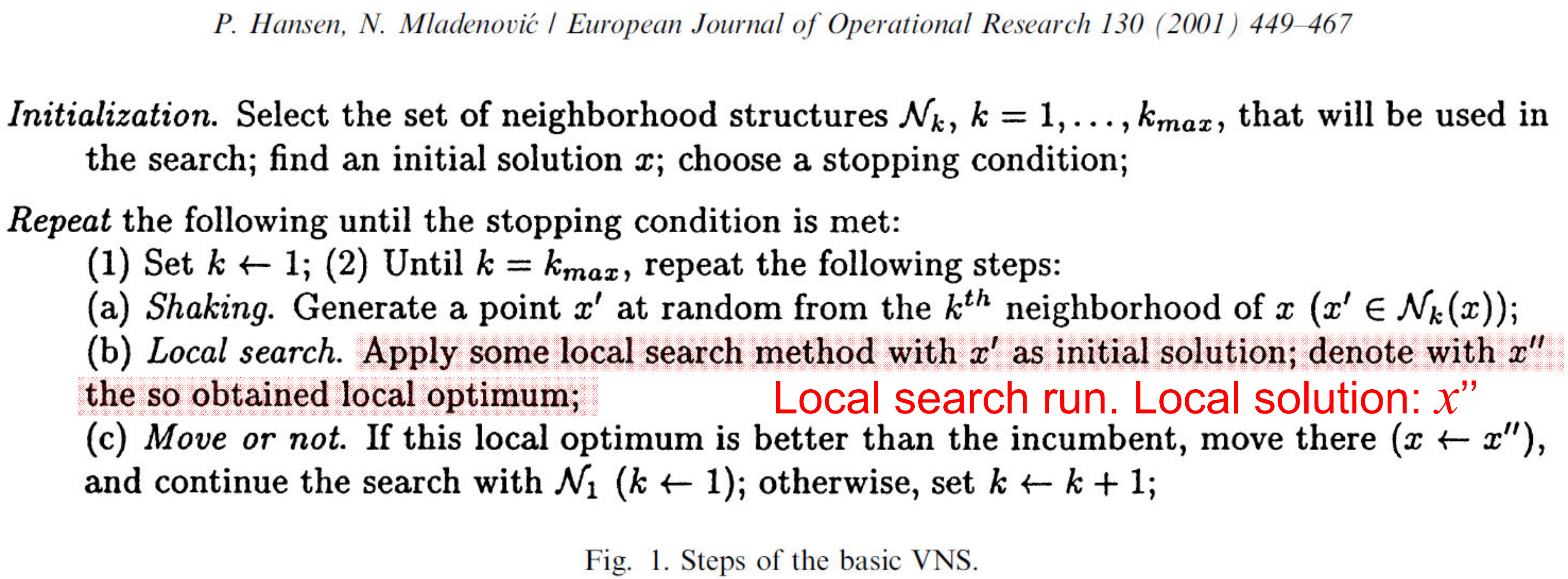
$\mathcal{N}_{k+1}$ is larger than $\mathcal{N}_k$, $k = 1, 2, \ldots, k_{max}-1$.

In some cases, $\mathcal{N}_k \subset \mathcal{N}_{k+1}$, $k = 1, 2, \ldots, k_{max}-1$.

$\mathcal{N}_1$ is used first, then $\mathcal{N}_2$, $\mathcal{N}_3$, ...

Local search is from a randomly selected neighbor in the current $\mathcal{N}_k$.

Basic Variable Neighborhood Search (VNS) Algorithm



P. Hansen, N. Mladenović / European Journal of Operational Research 130 (2001) 449–467

Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, that will be used in the search; find an initial solution x; choose a stopping condition;

Repeat the following until the stopping condition is met:

(1) Set $k \leftarrow 1$; (2) Until $k = k_{max}$, repeat the following steps:

(a) *Shaking.* Generate a point x' at random from the k^{th} neighborhood of x ($x' \in \mathcal{N}_k(x)$);

(b) *Local search.* Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;

Local search run. Local solution: x''

(c) *Move or not.* If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;

Fig. 1. Steps of the basic VNS.

Neighborhood Structures: $\mathcal{N}_k$, $k = 1, 2, \ldots, k_{max}$

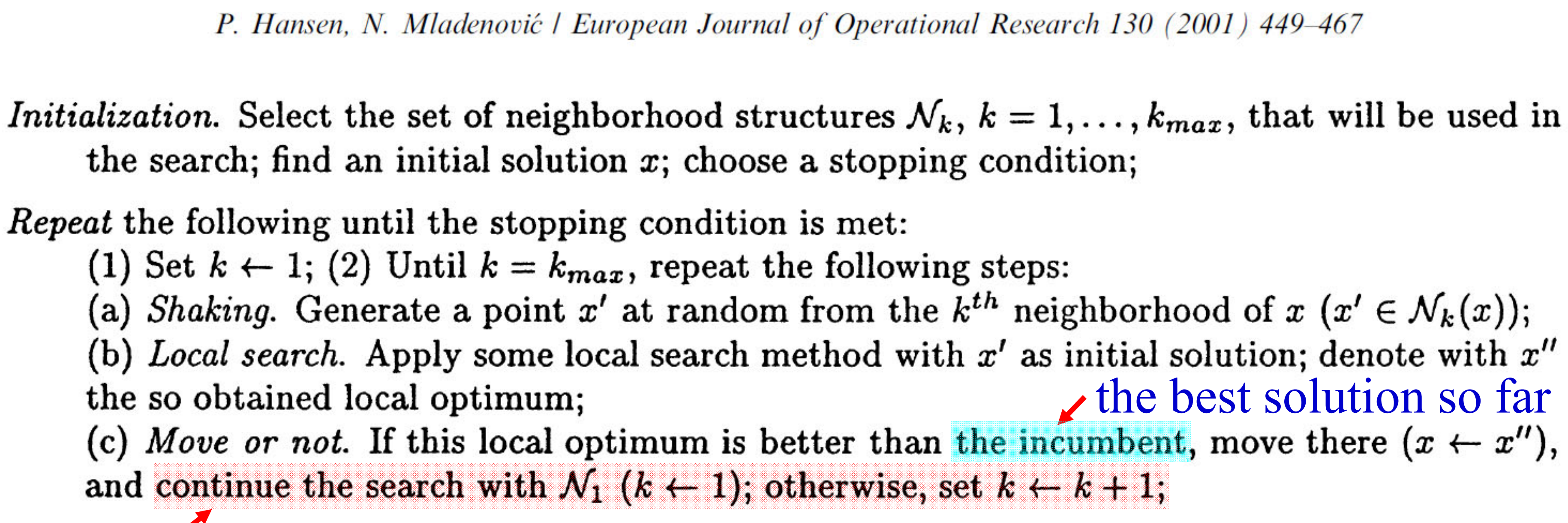
$\mathcal{N}_{k+1}$ is larger than $\mathcal{N}_k$, $k = 1, 2, \ldots, k_{max} - 1$.

In some cases, $\mathcal{N}_k \subset \mathcal{N}_{k+1}$, $k = 1, 2, \ldots, k_{max} - 1$.

$\mathcal{N}_1$ is used first, then $\mathcal{N}_2$, $\mathcal{N}_3$, ...

Local search is from a randomly selected neighbor in the current $\mathcal{N}_k$.

Basic Variable Neighborhood Search (VNS) Algorithm



P. Hansen, N. Mladenović / European Journal of Operational Research 130 (2001) 449–467

Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, that will be used in the search; find an initial solution x; choose a stopping condition;

Repeat the following until the stopping condition is met:
(1) Set $k \leftarrow 1$; (2) Until $k = k_{max}$, repeat the following steps:
(a) *Shaking.* Generate a point x' at random from the k^{th} neighborhood of x ($x' \in \mathcal{N}_k(x)$);
(b) *Local search.* Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;
(c) *Move or not.* If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;

the best solution so far

If the current solution is improved, $\mathcal{N}_k$ ==>$\mathcal{N}_1$. If not, $\mathcal{N}_k$ ==>$\mathcal{N}_{k+1}$.

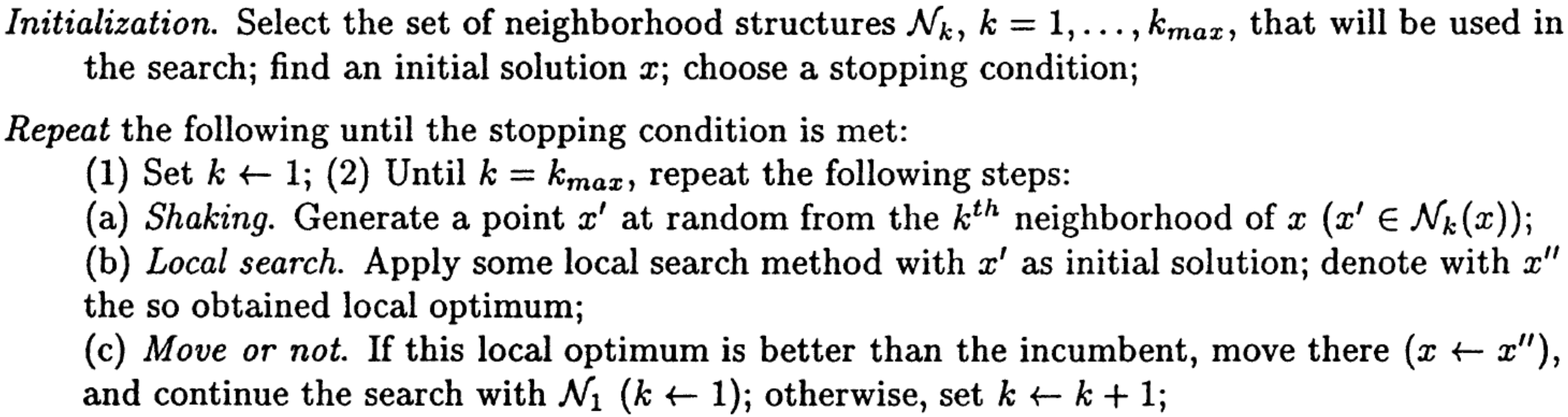
Neighborhood Structures: $\mathcal{N}_k$, $k = 1, 2, \ldots, k_{max}$

$\mathcal{N}_{k+1}$ is larger than $\mathcal{N}_k$, $k = 1, 2, \ldots, k_{max}-1$.

In some cases, $\mathcal{N}_k \subset \mathcal{N}_{k+1}$, $k = 1, 2, \ldots, k_{max}-1$.

$\mathcal{N}_1$ is used first, then $\mathcal{N}_2$, $\mathcal{N}_3$, ...

Local search is from a randomly selected neighbor in the current $\mathcal{N}_k$.



Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, that will be used in the search; find an initial solution x; choose a stopping condition;

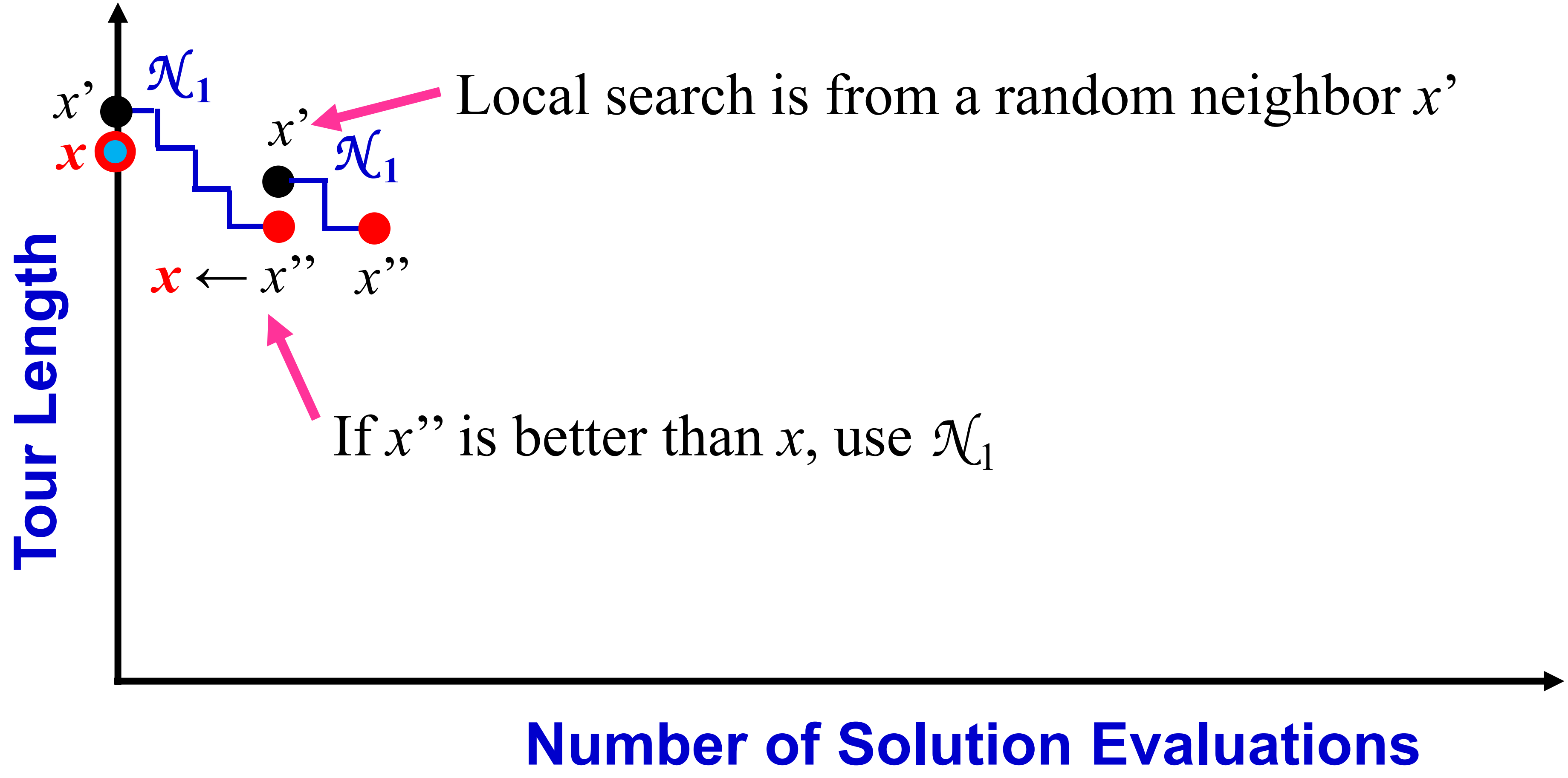
Repeat the following until the stopping condition is met:

(1) Set $k \leftarrow 1$; (2) Until $k = k_{max}$, repeat the following steps:

(a) *Shaking*. Generate a point x' at random from the k^{th} neighborhood of x ($x' \in \mathcal{N}_k(x)$);

(b) *Local search*. Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;

(c) *Move or not*. If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;



Local search is from a random neighbor *x*'

If *x*'' is better than *x*, use $\mathcal{N}_1$

Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, that will be used in the search; find an initial solution x; choose a stopping condition;

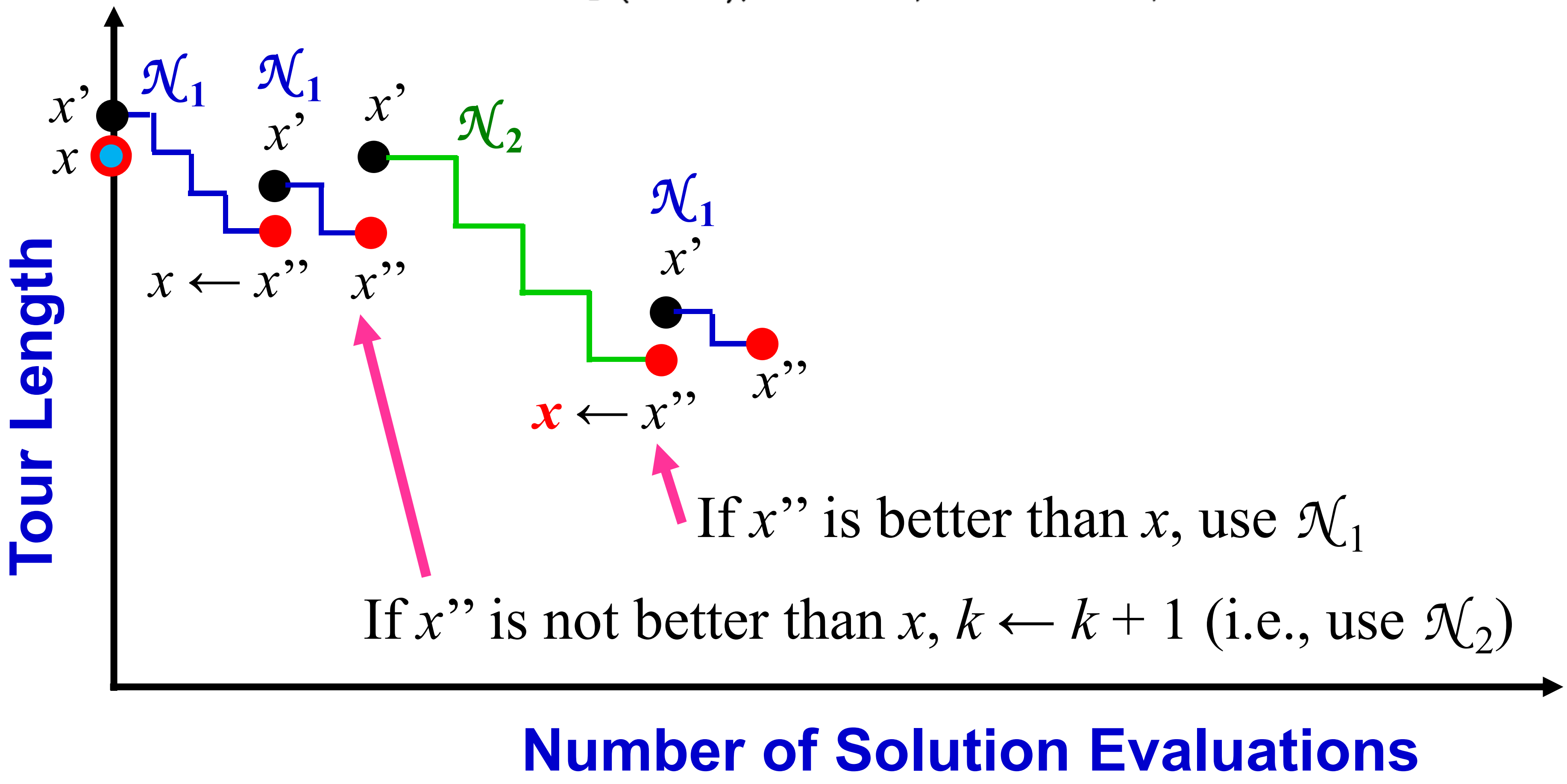
Repeat the following until the stopping condition is met:

(1) Set $k \leftarrow 1$; (2) Until $k = k_{max}$, repeat the following steps:

(a) *Shaking.* Generate a point x' at random from the k^{th} neighborhood of x ($x' \in \mathcal{N}_k(x)$);

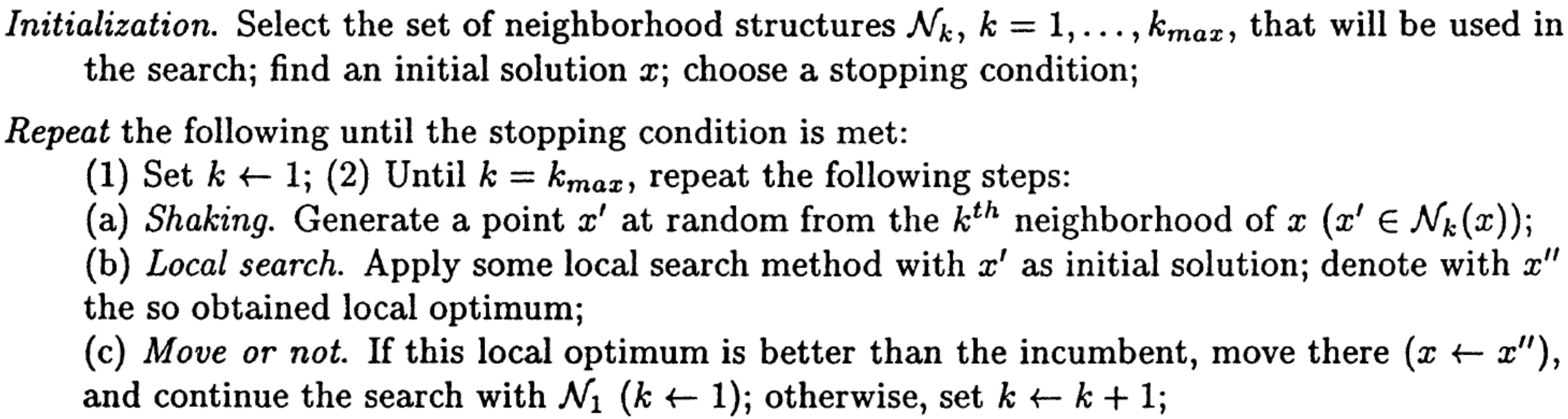
(b) *Local search.* Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;

(c) *Move or not.* If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;



If x'' is better than x, use $\mathcal{N}_1$

If x'' is not better than x, $k \leftarrow k + 1$ (i.e., use $\mathcal{N}_2$)



Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, that will be used in the search; find an initial solution x; choose a stopping condition;

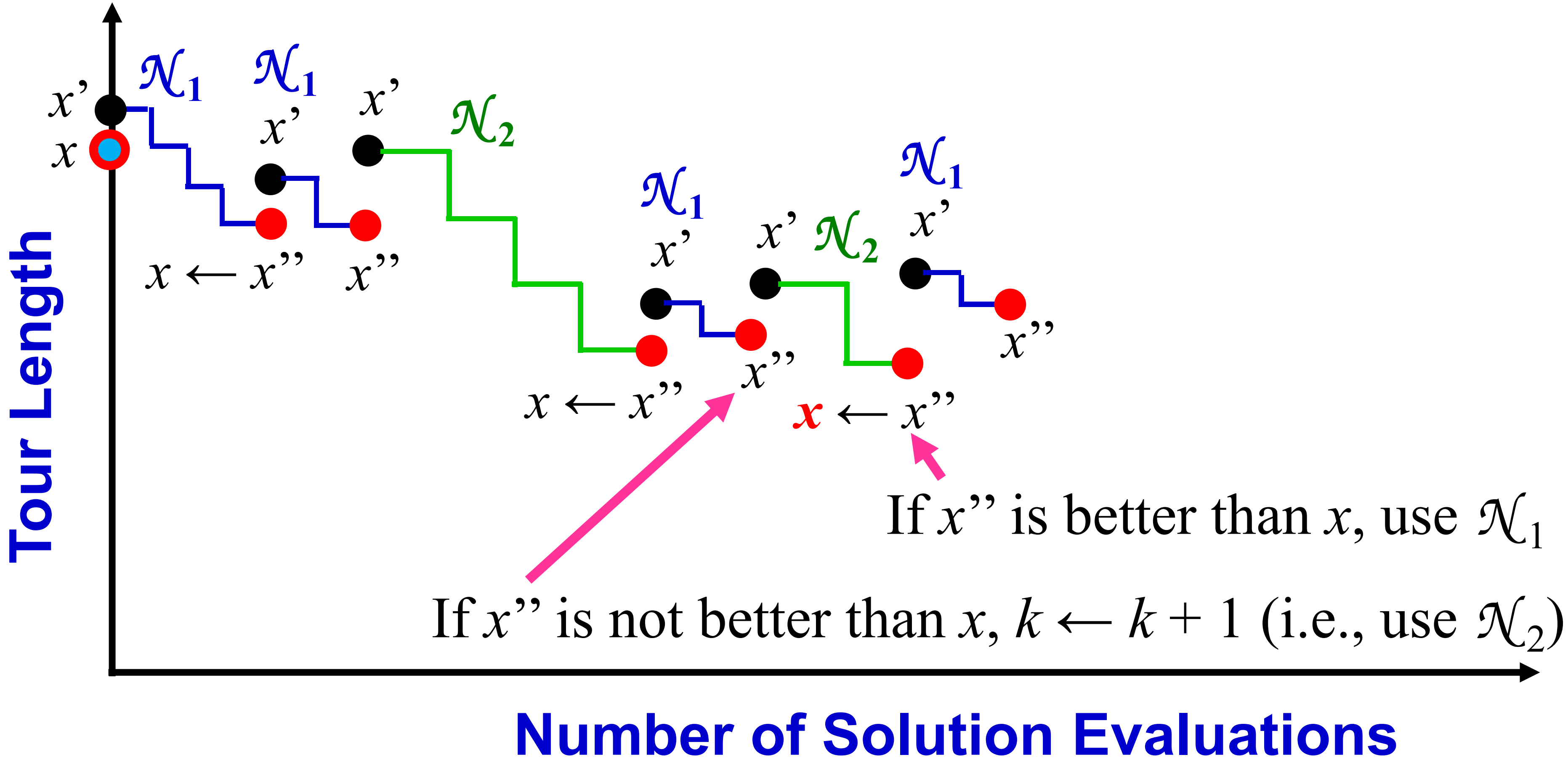
Repeat the following until the stopping condition is met:

(1) Set $k \leftarrow 1$; (2) Until $k = k_{max}$, repeat the following steps:

(a) *Shaking.* Generate a point x' at random from the k^{th} neighborhood of x ($x' \in \mathcal{N}_k(x)$);

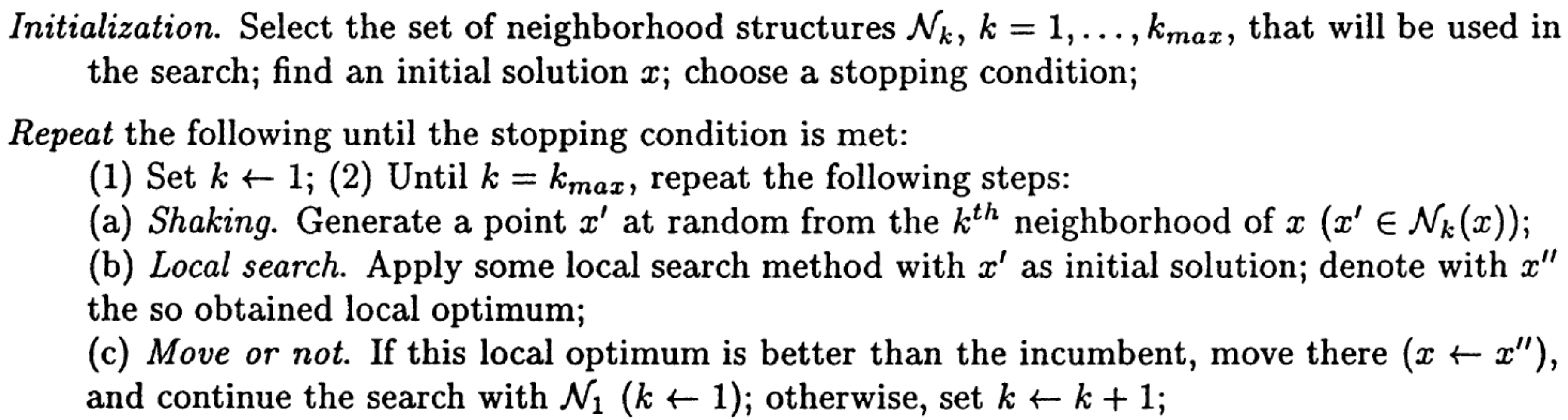
(b) *Local search.* Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;

(c) *Move or not.* If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;



If x'' is better than x, use $\mathcal{N}_1$

If x'' is not better than x, $k \leftarrow k + 1$ (i.e., use $\mathcal{N}_2$)



Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, that will be used in the search; find an initial solution x; choose a stopping condition;

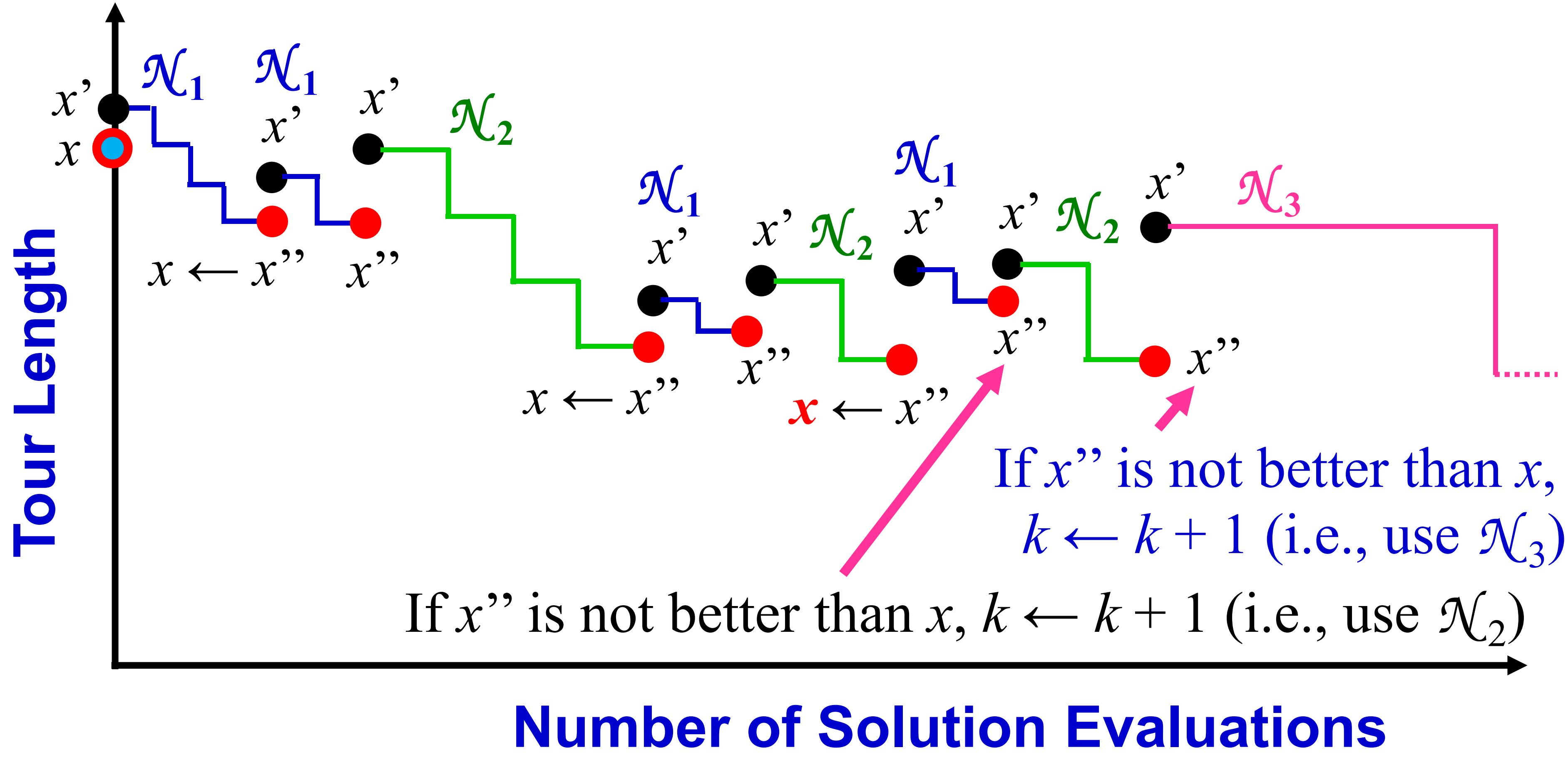
Repeat the following until the stopping condition is met:

(1) Set $k \leftarrow 1$; (2) Until $k = k_{max}$, repeat the following steps:

(a) *Shaking.* Generate a point x' at random from the k^{th} neighborhood of x ($x' \in \mathcal{N}_k(x)$);

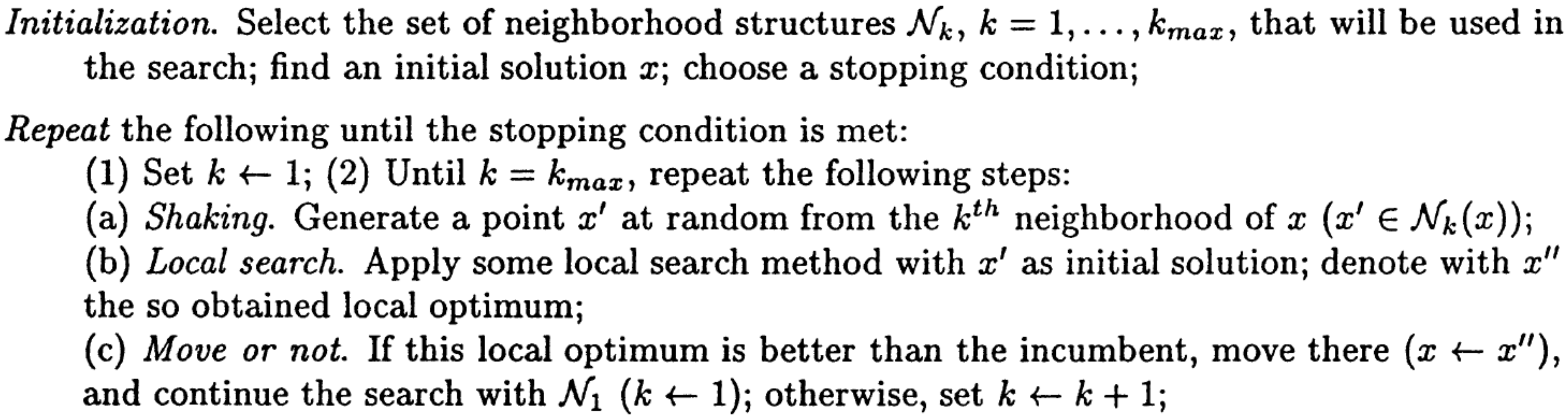
(b) *Local search.* Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;

(c) *Move or not.* If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;



If x'' is not better than x, $k \leftarrow k + 1$ (i.e., use $\mathcal{N}_3$)

If x'' is not better than x, $k \leftarrow k + 1$ (i.e., use $\mathcal{N}_2$)



Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \dots, k_{max}$, that will be used in the search; find an initial solution x; choose a stopping condition;

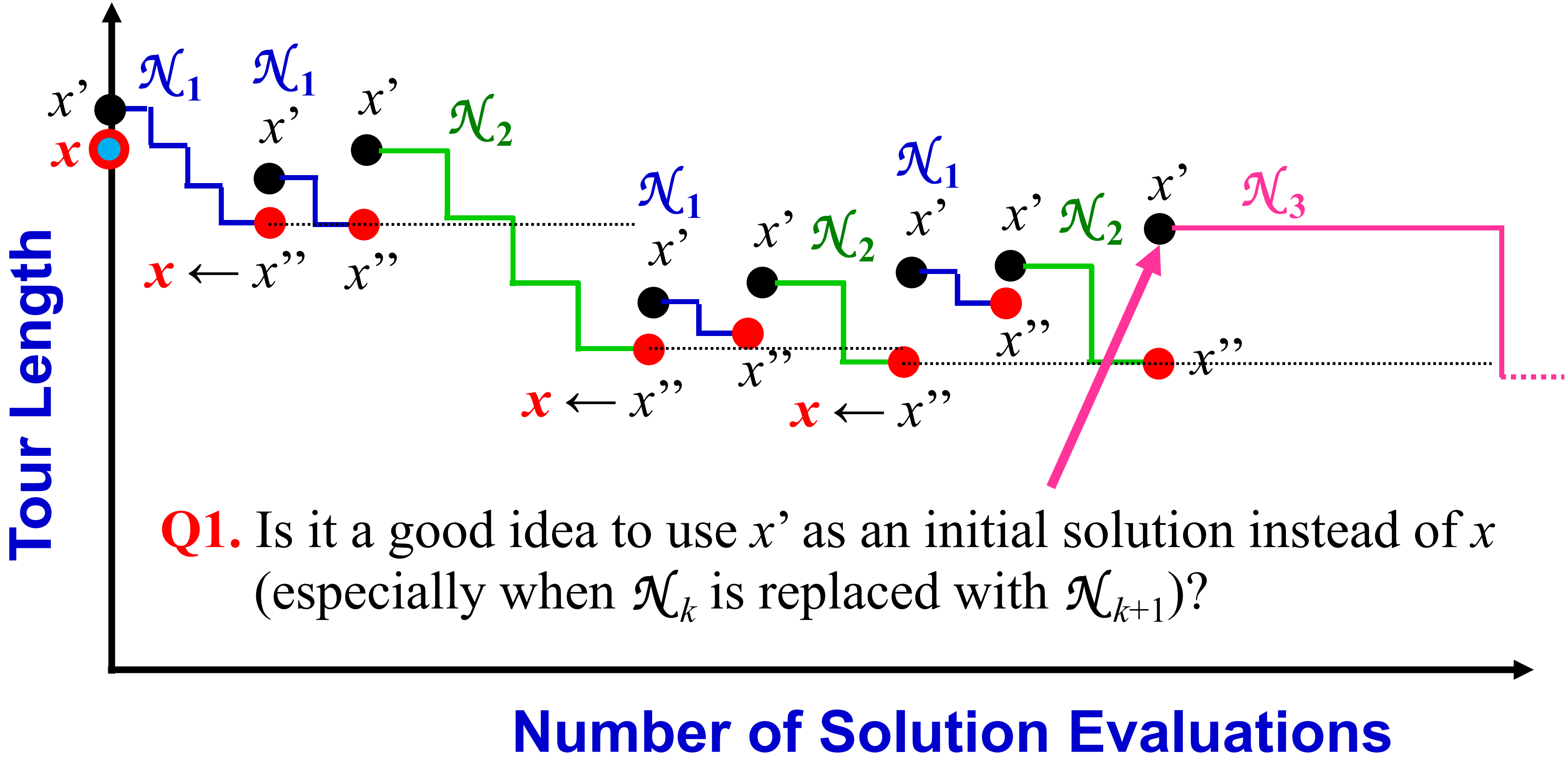
Repeat the following until the stopping condition is met:

(1) Set $k \leftarrow 1$; (2) Until $k = k_{max}$, repeat the following steps:

(a) *Shaking.* Generate a point x' at random from the k^{th} neighborhood of x ($x' \in \mathcal{N}_k(x)$);

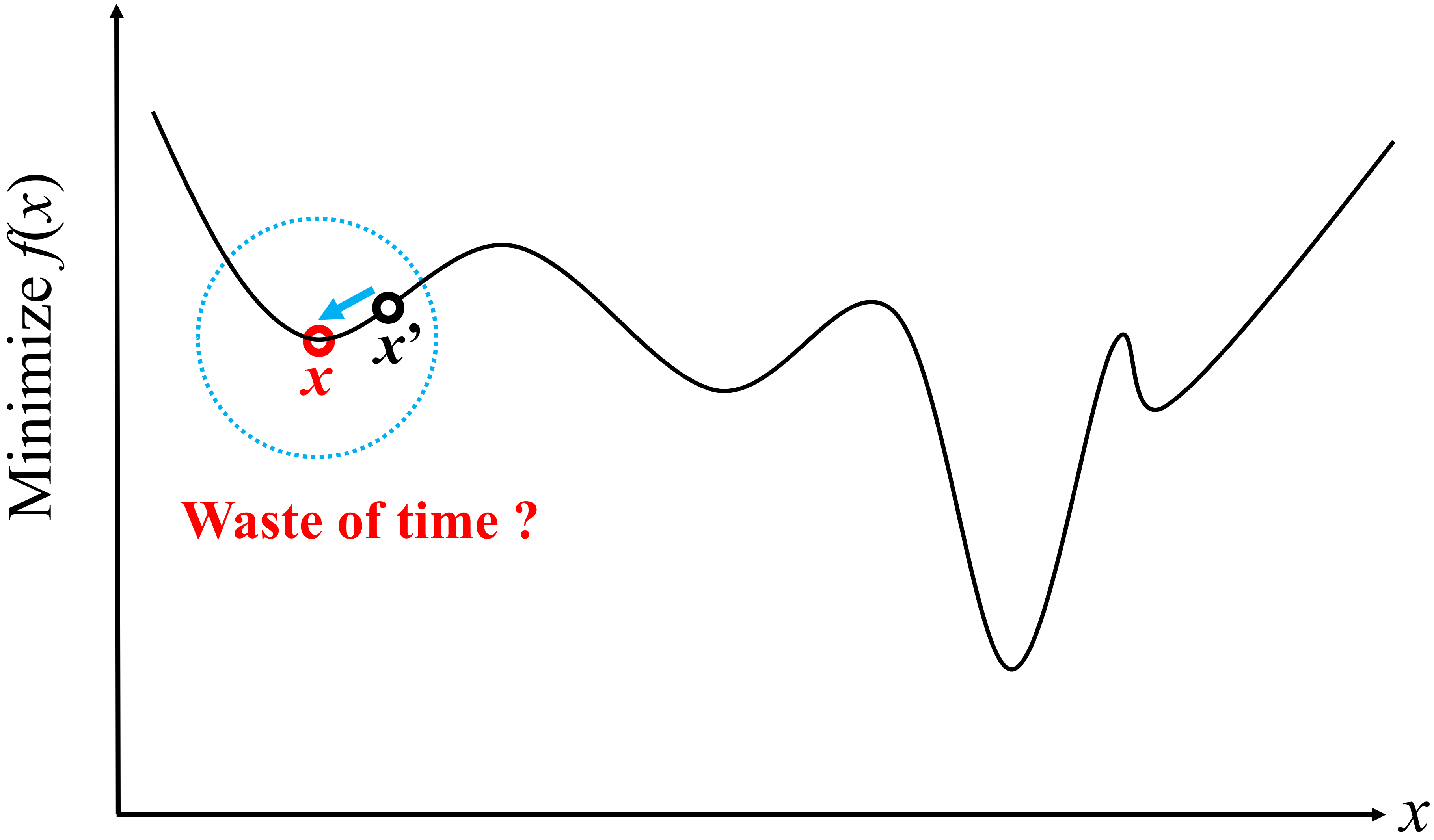
(b) *Local search.* Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;

(c) *Move or not.* If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;



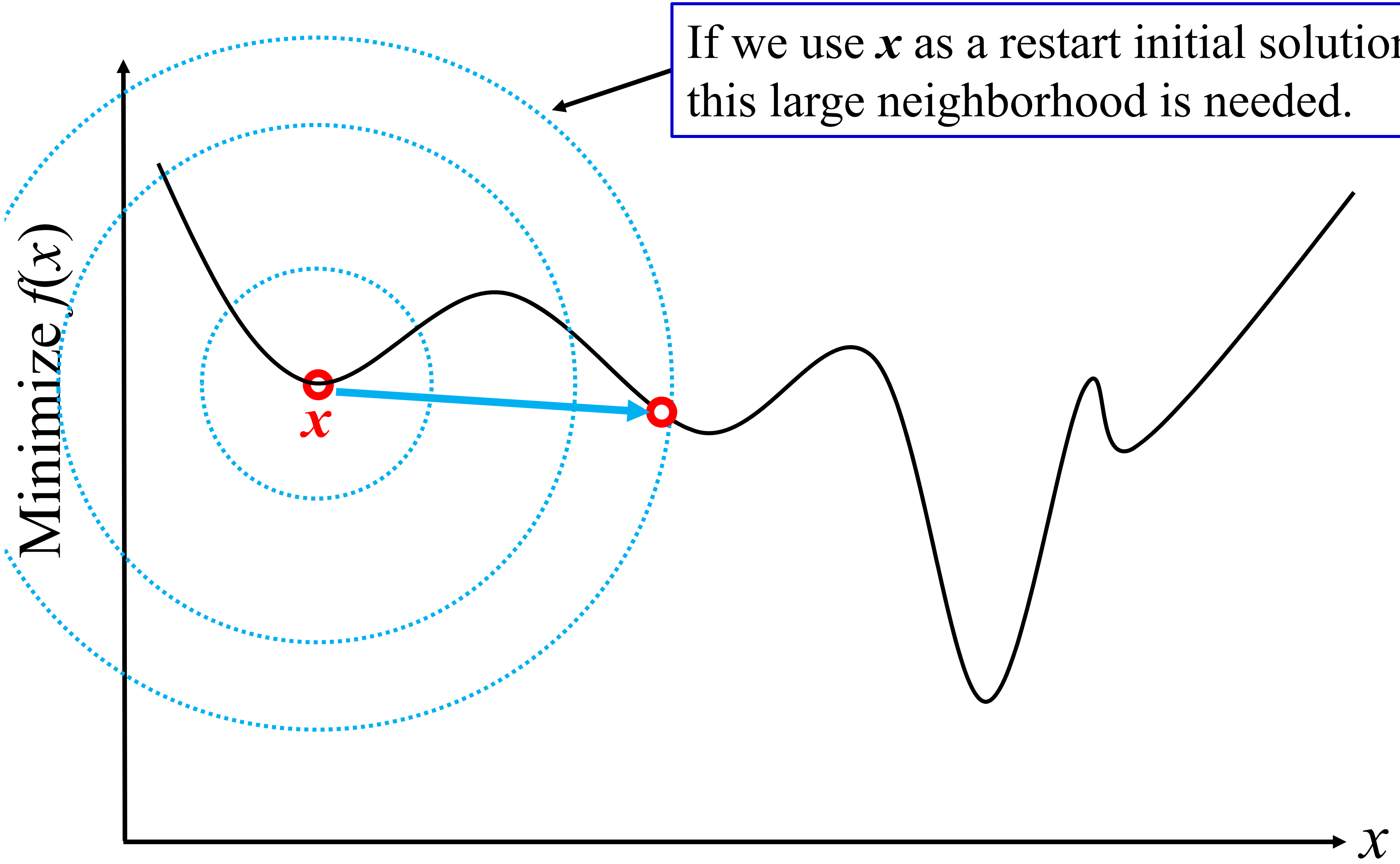
Q1. Is it a good idea to use x' as an initial solution instead of x (especially when $\mathcal{N}_k$ is replaced with $\mathcal{N}_{k+1}$)?

Q1. Is it a good idea to use *x*’ instead of *x* ?



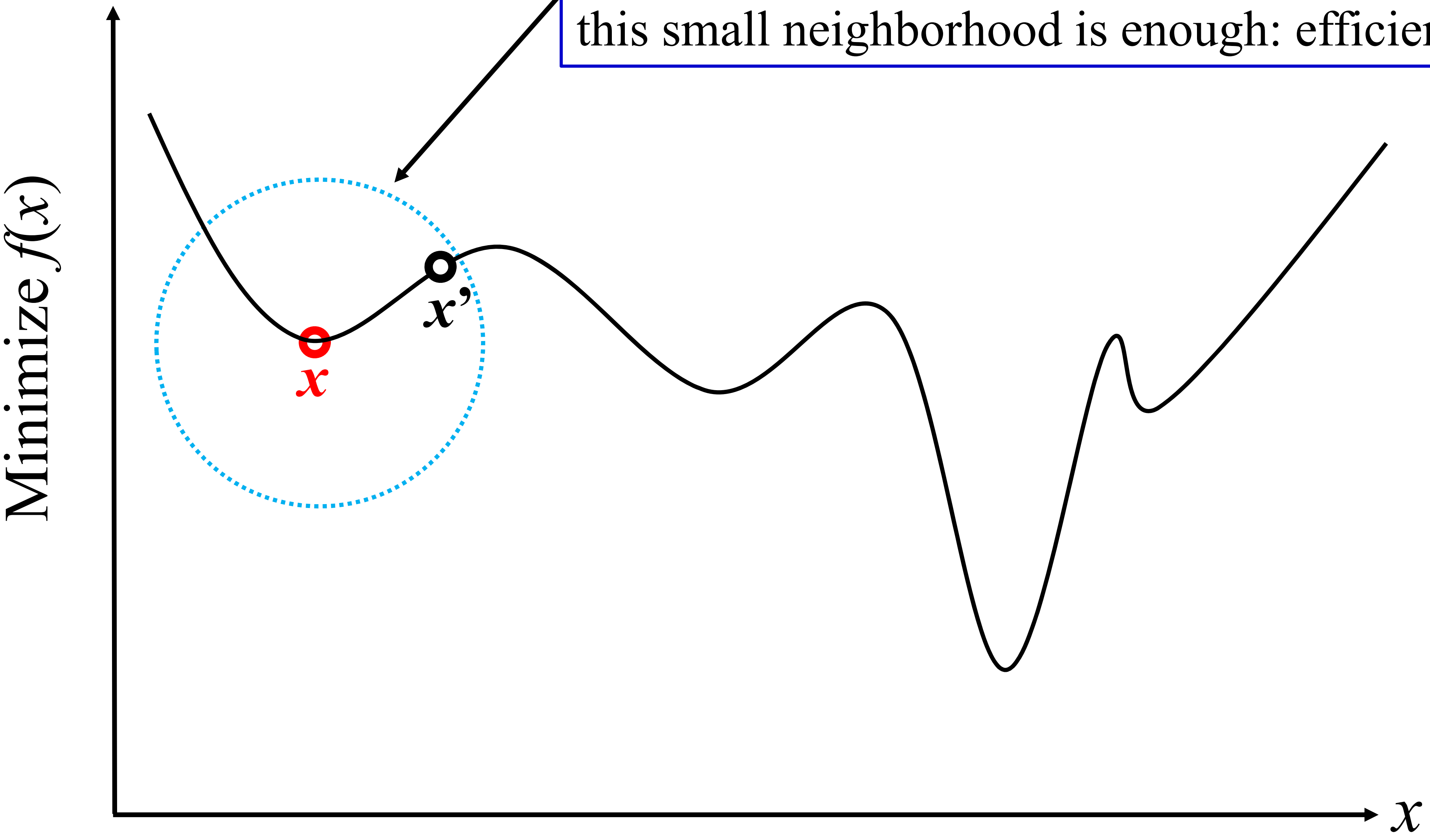
Waste of time ?

Q1. Is it a good idea to use *x*' instead of *x* ?



If we use ***x*** as a restart initial solution, this large neighborhood is needed.

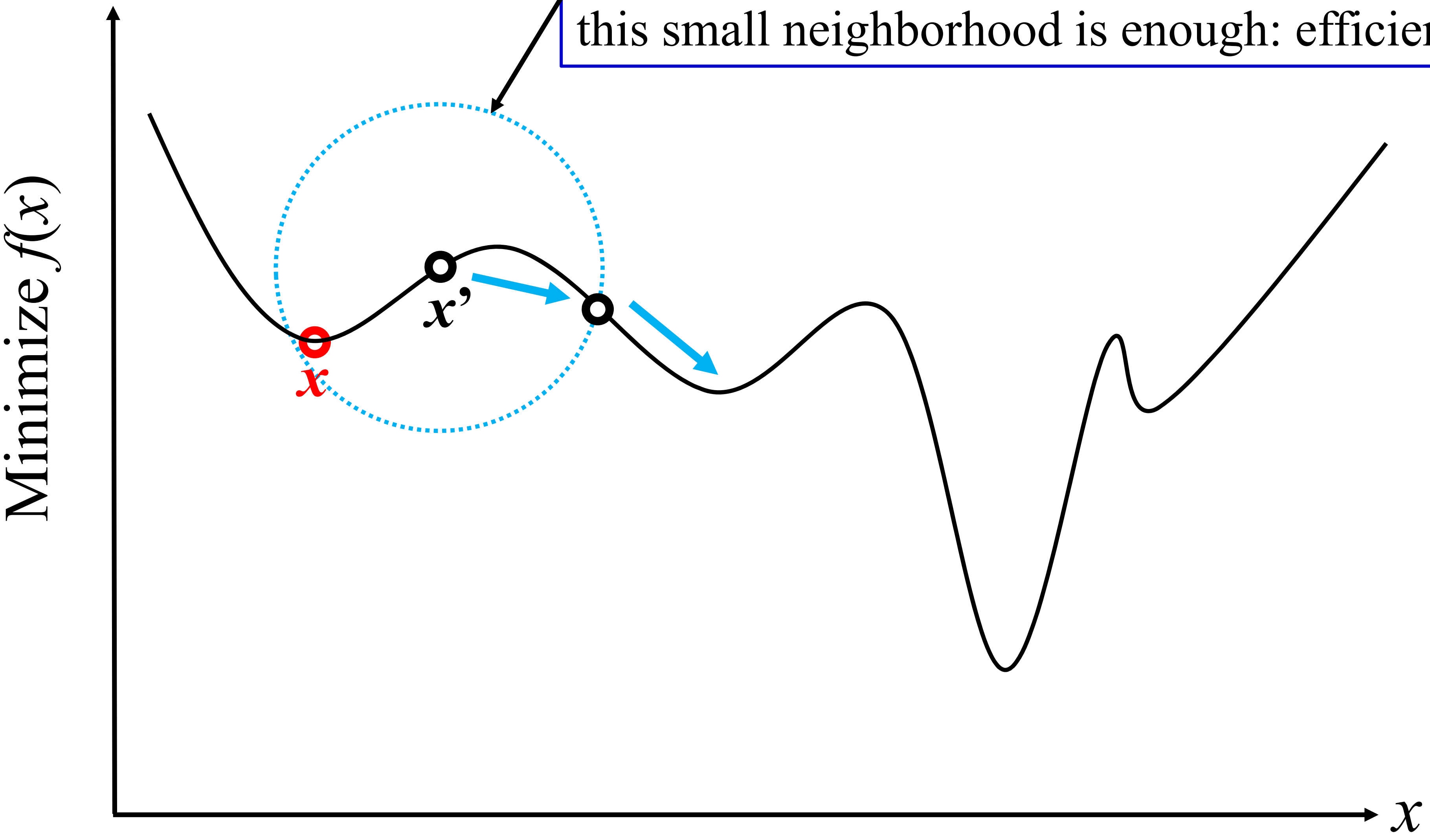
Q1. Is it a good idea to use *x*' instead of *x* ?



If we use ***x'*** as a restart initial solution, this small neighborhood is enough: efficient

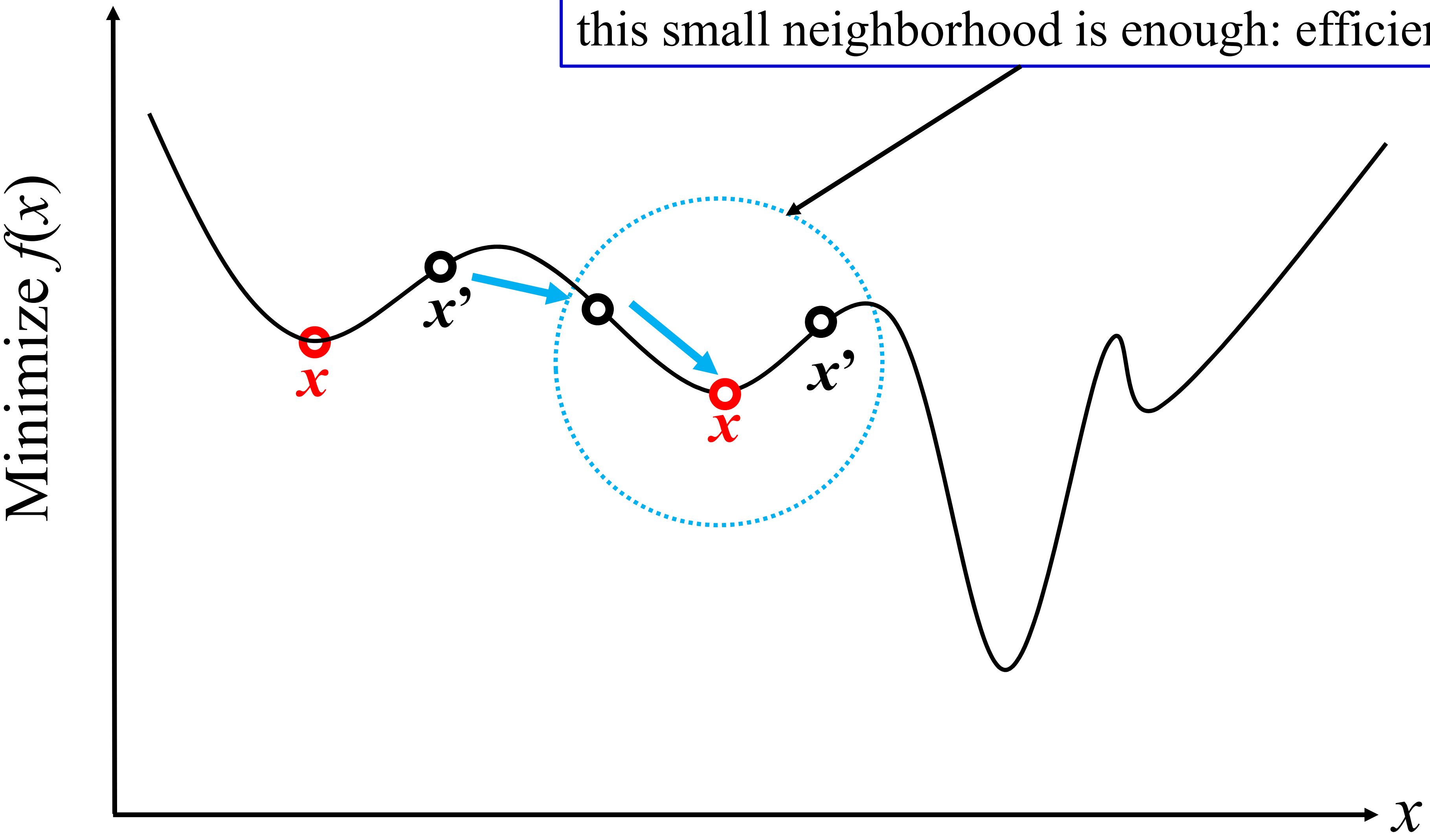
Q1. Is it a good idea to use *x*' instead of *x* ?

If we use ***x'*** as a restart initial solution,
this small neighborhood is enough: efficient



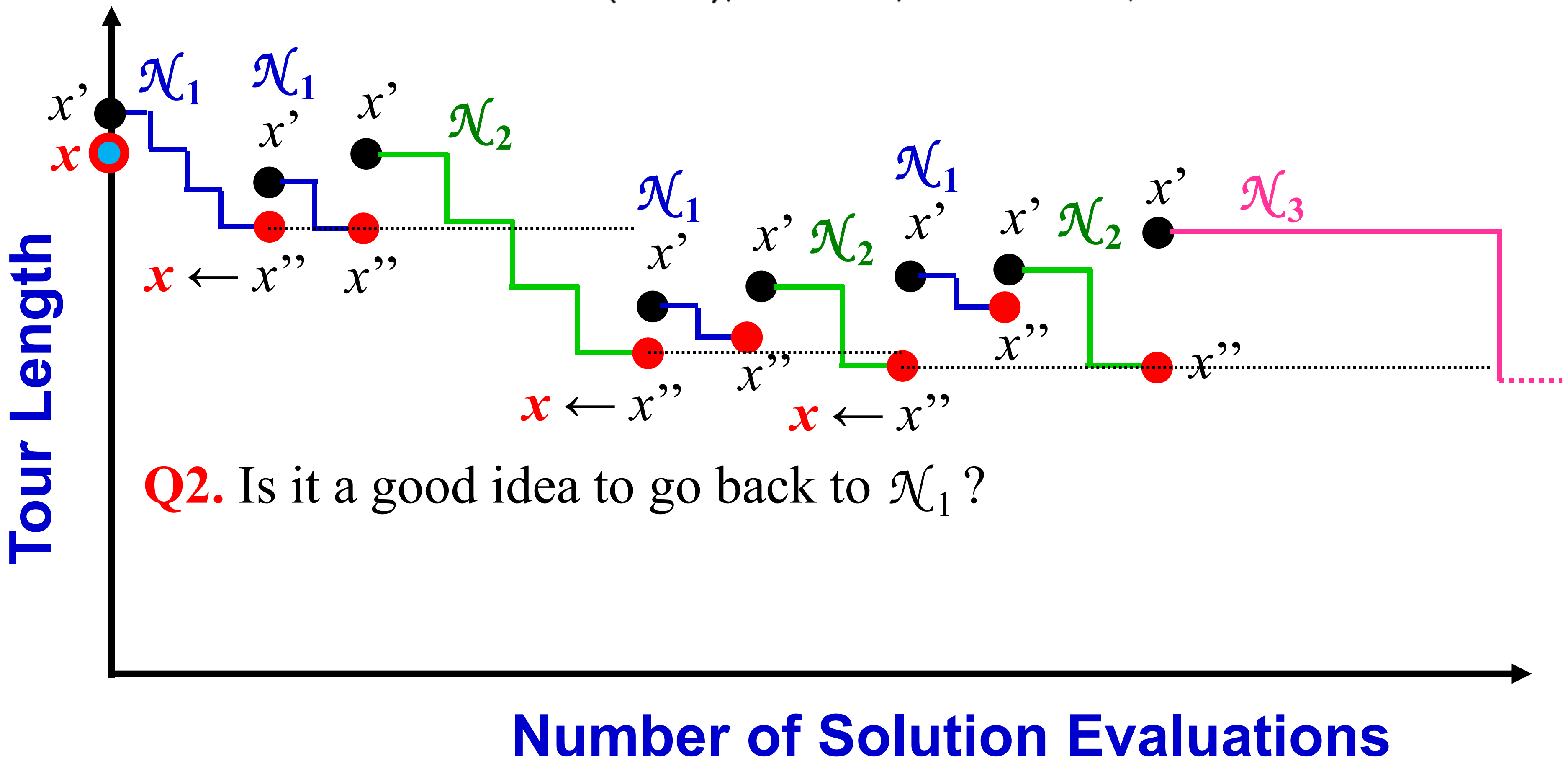
Q1. Is it a good idea to use *x*’ instead of *x* ?

If we use ***x’*** as a restart initial solution, this small neighborhood is enough: efficient

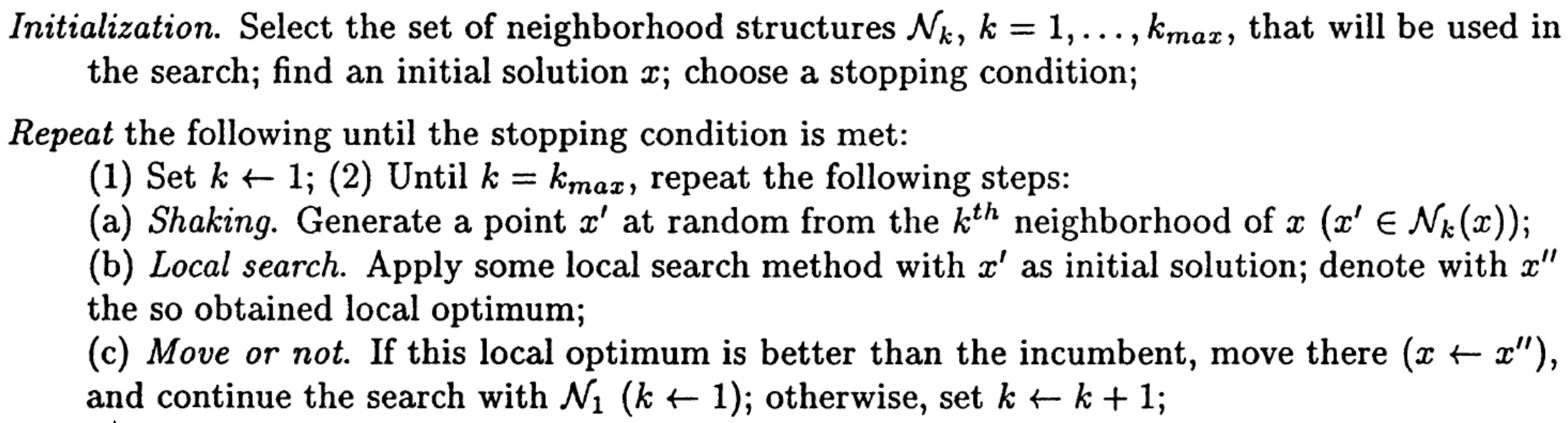


Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, that will be used in the search; find an initial solution x; choose a stopping condition;

Repeat the following until the stopping condition is met:
(1) Set $k \leftarrow 1$; (2) Until $k = k_{max}$, repeat the following steps:
(a) *Shaking.* Generate a point x' at random from the k^{th} neighborhood of x ($x' \in \mathcal{N}_k(x)$);
(b) *Local search.* Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;
(c) *Move or not.* If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;



Q2. Is it a good idea to go back to $\mathcal{N}_1$?



Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \dots, k_{max}$, that will be used in the search; find an initial solution x; choose a stopping condition;

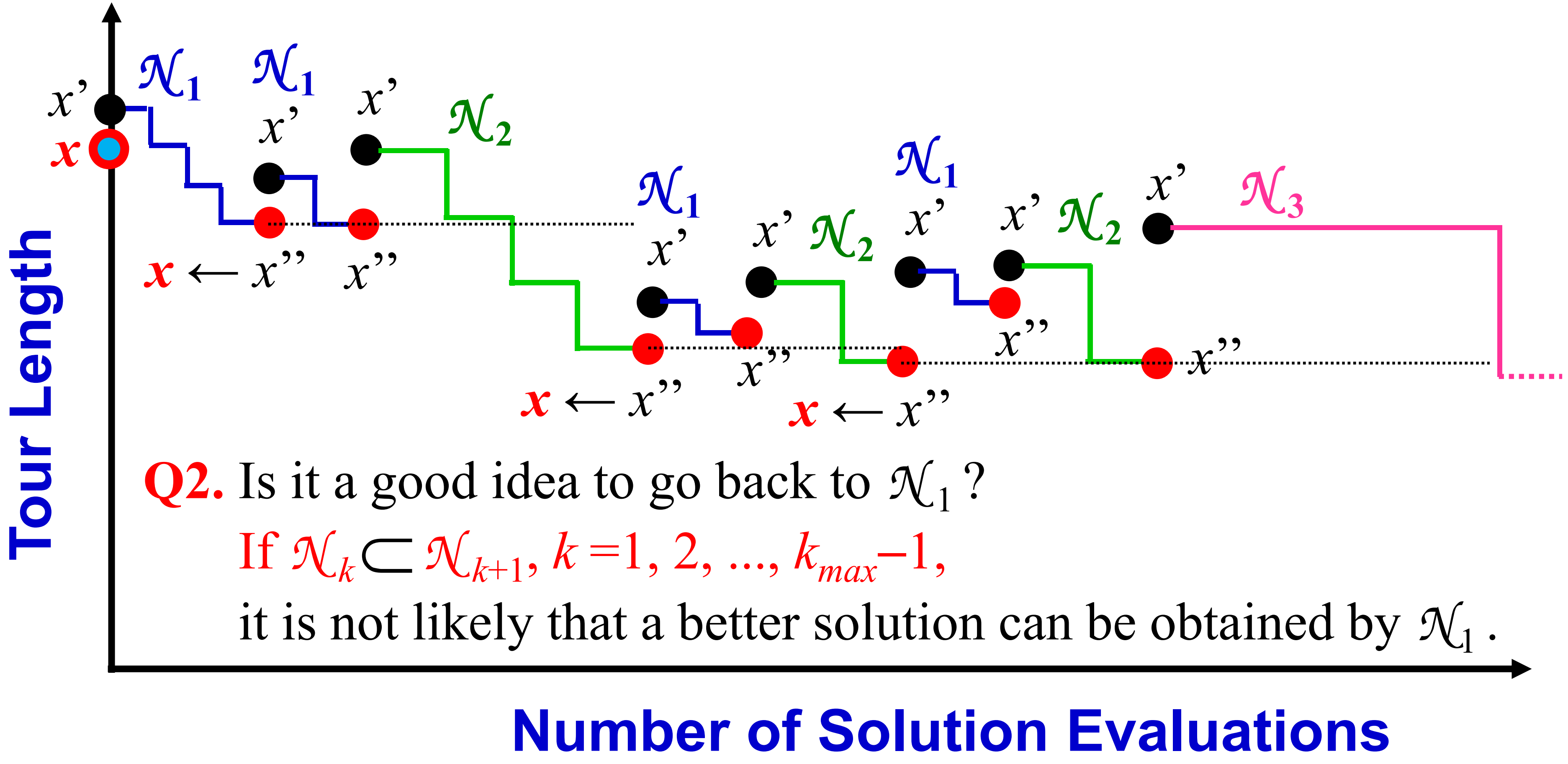
Repeat the following until the stopping condition is met:

(1) Set $k \leftarrow 1$; (2) Until $k = k_{max}$, repeat the following steps:

(a) *Shaking.* Generate a point x' at random from the k^{th} neighborhood of x ($x' \in \mathcal{N}_k(x)$);

(b) *Local search.* Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;

(c) *Move or not.* If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;



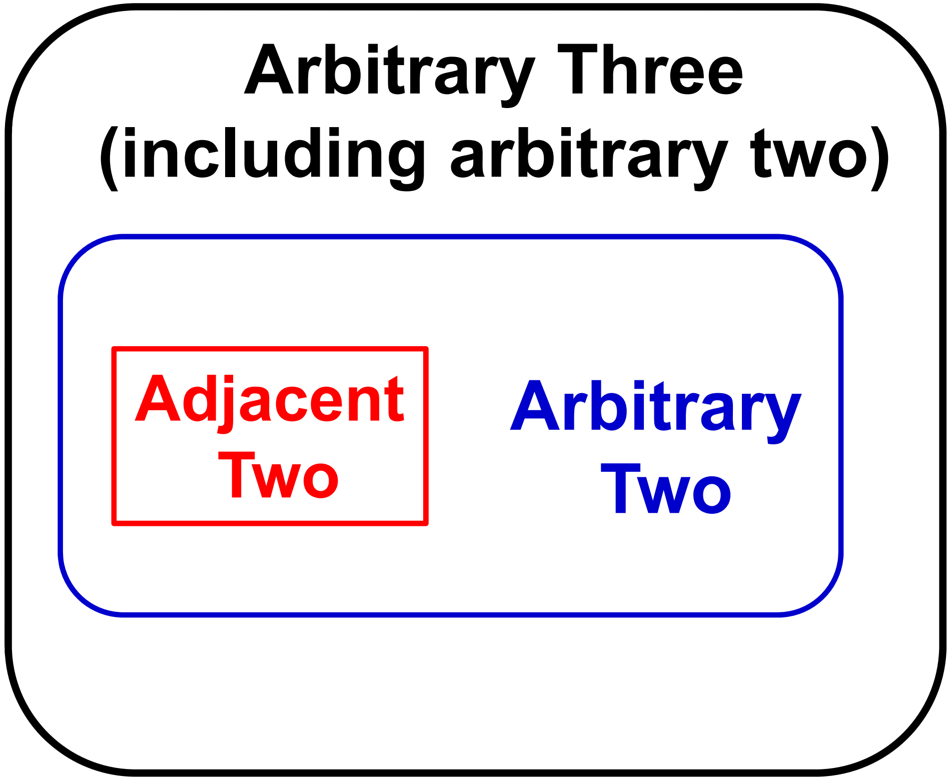
Q2. Is it a good idea to go back to $\mathcal{N}_1$?

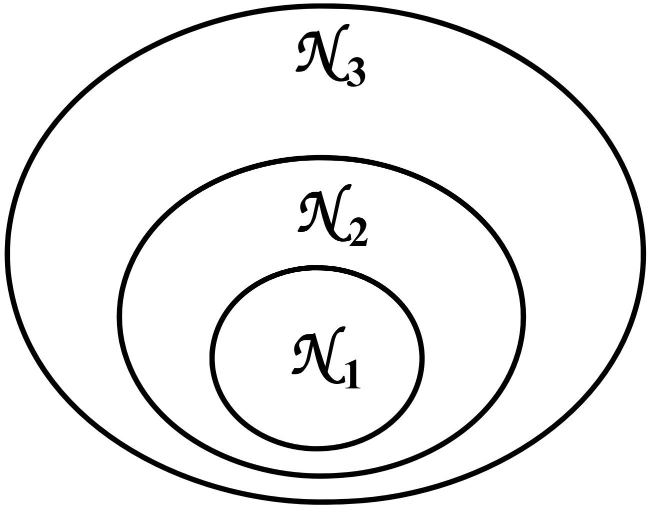
If $\mathcal{N}_k \subset \mathcal{N}_{k+1}$, $k = 1, 2, \dots, k_{max} - 1$,

it is not likely that a better solution can be obtained by $\mathcal{N}_1$.

Q2. Is it a good idea to go back to $\mathcal{N}_1$?

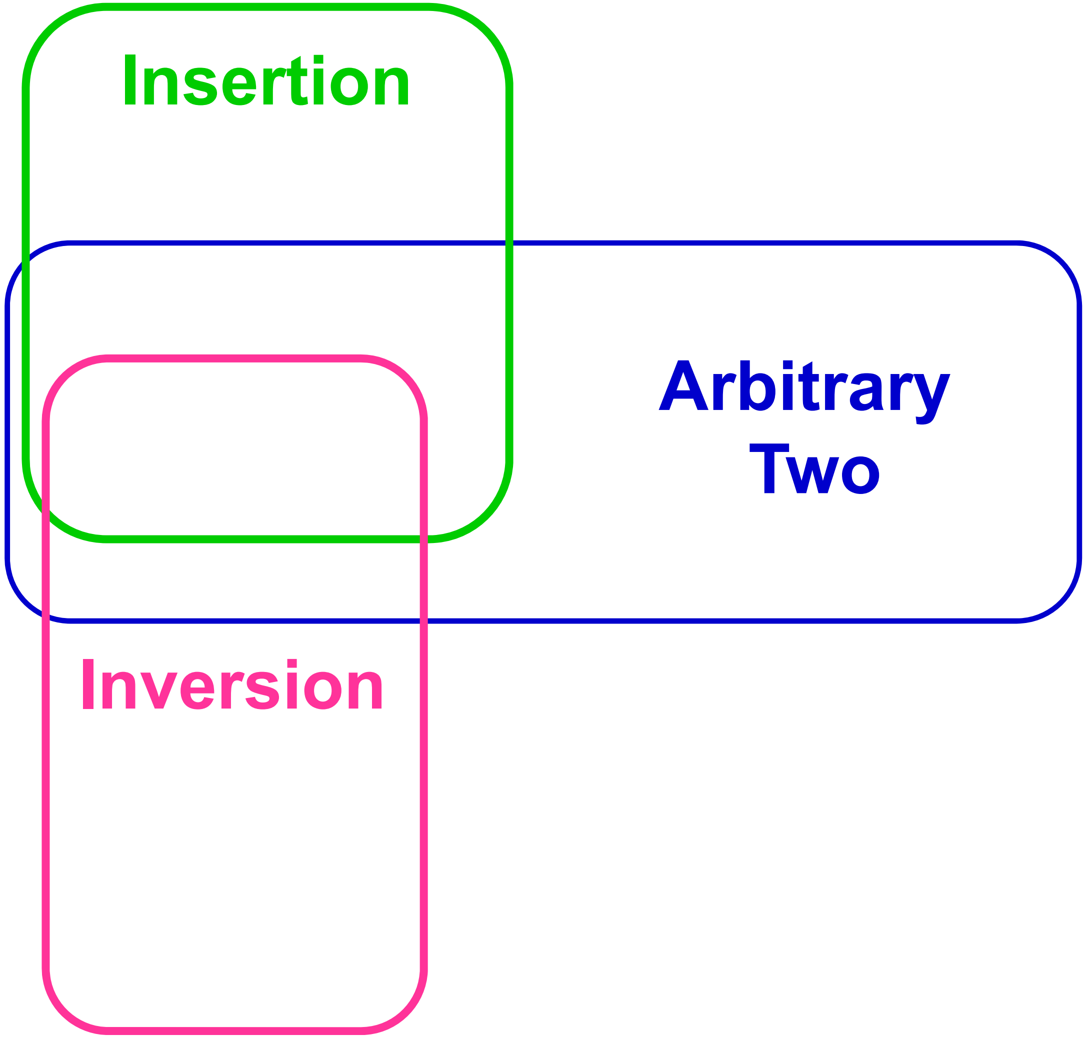
If $\mathcal{N}_k \subset \mathcal{N}_{k+1}$, $k = 1, 2, \ldots, k_{max} - 1$,
it is not likely that a better solution can be obtained by $\mathcal{N}_1$.





Q2. Is it a good idea to go back to $\mathcal{N}_1$?

If $\mathcal{N}_k \subset \mathcal{N}_{k+1}$, k =1, 2, ..., k_{max}−1,
it is not likely that a better solution can be obtained by $\mathcal{N}_1$.



Example of Neighborhood Structures of TSP (*n*-city problem)

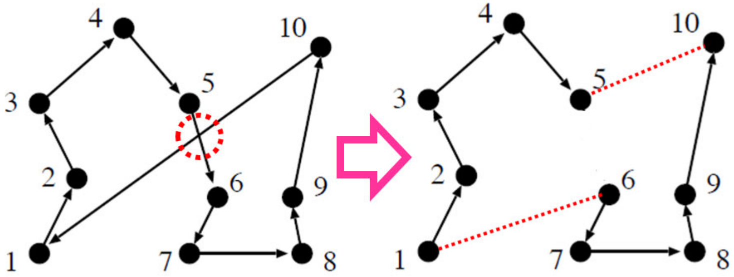
$\mathcal{N}_k$: *k*-edge change neighborhood (k = 2, 3, ..., n)

$\mathcal{N}_1 = \emptyset$

$\mathcal{N}_2$

$\mathcal{N}_3$

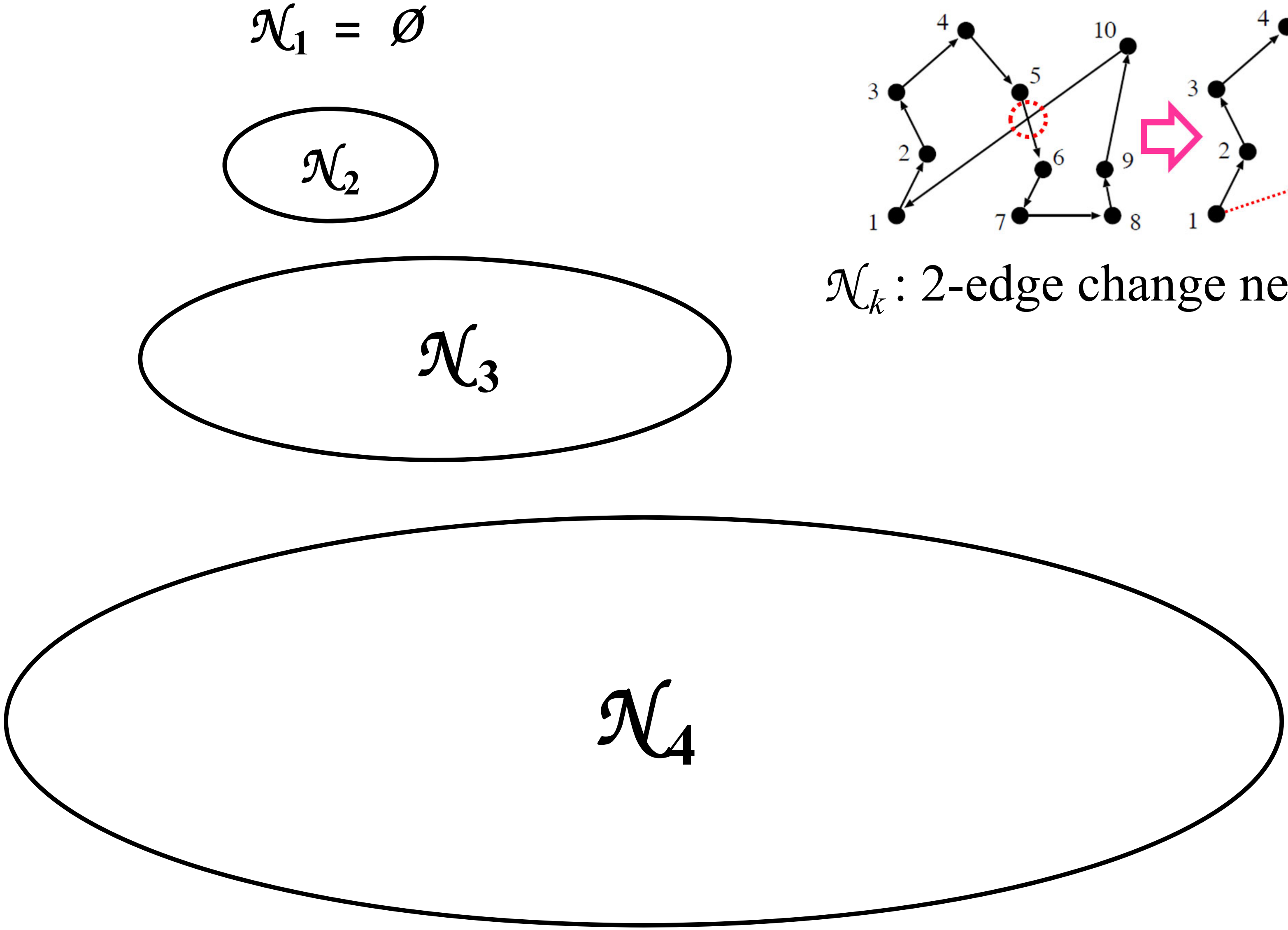
$\mathcal{N}_4$

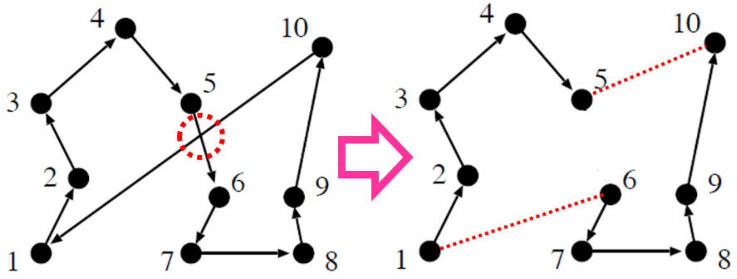


$\mathcal{N}_k$: 2-edge change neighborhood

Q2. Is it a good idea to go back to $\mathcal{N}_1$? **Yes ?** (Depends on the problem.)

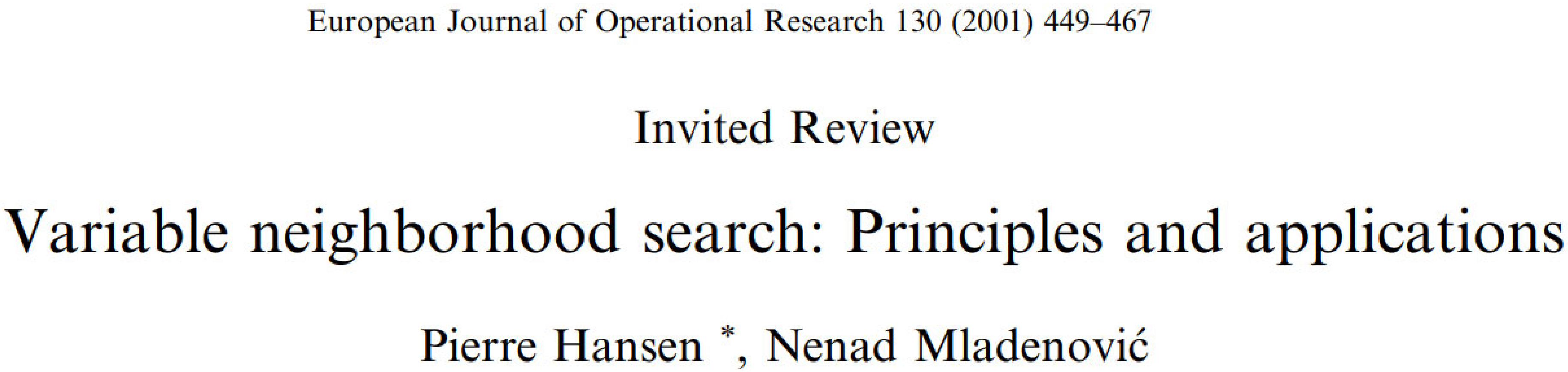
$\mathcal{N}_k$: k-edge change neighborhood ($k = 2, 3, \ldots, n$)





$\mathcal{N}_k$: 2-edge change neighborhood

Variant of Variable Neighborhood Search (VNS)
Variable Neighborhood Descent (VND)



European Journal of Operational Research 130 (2001) 449–467

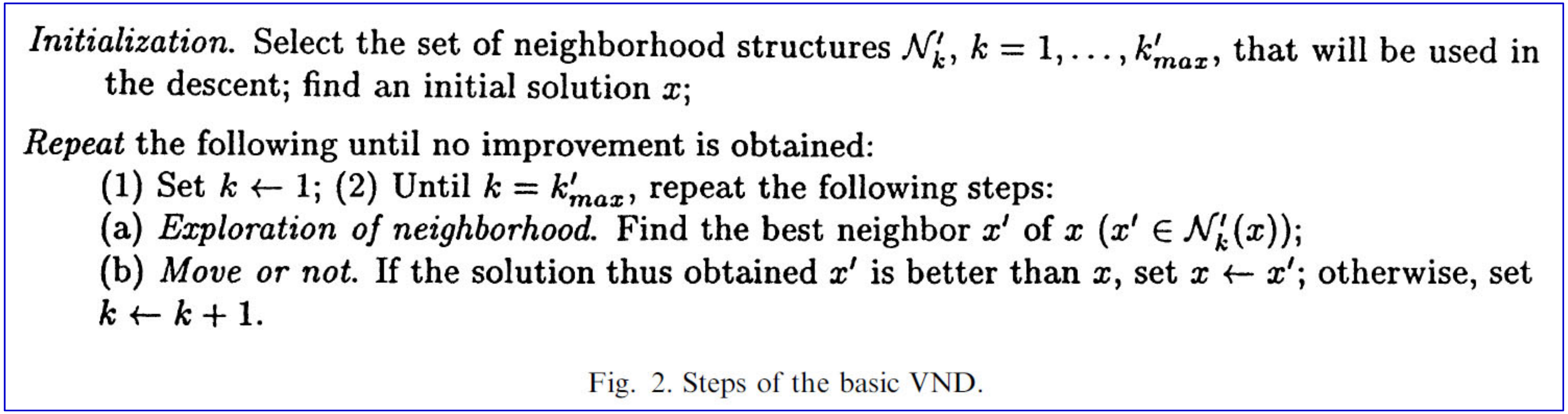
Invited Review

Variable neighborhood search: Principles and applications

Pierre Hansen *, Nenad Mladenović

Variant of Variable Neighborhood Search (VNS)
Variable Neighborhood Descent (VND)

P. Hansen, N. Mladenović / European Journal of Operational Research 130 (2001) 449–467



Initialization. Select the set of neighborhood structures $\mathcal{N}'_k$, $k = 1, \dots, k'_{max}$, that will be used in the descent; find an initial solution x;

Repeat the following until no improvement is obtained:

(1) Set $k \leftarrow 1$; (2) Until $k = k'_{max}$, repeat the following steps:

(a) *Exploration of neighborhood.* Find the best neighbor x' of x $(x' \in \mathcal{N}'_k(x))$;

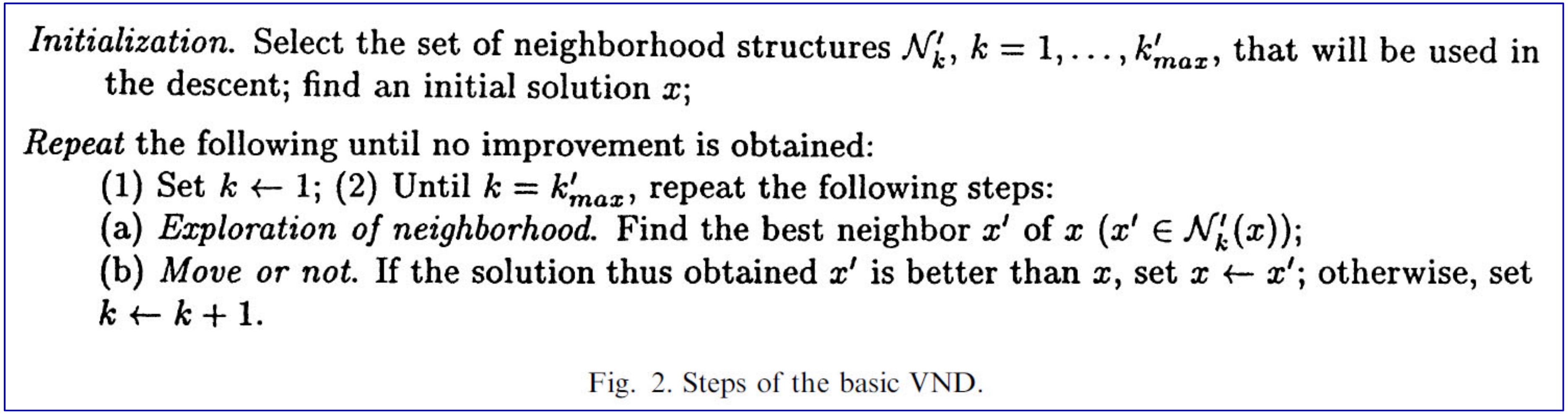
(b) *Move or not.* If the solution thus obtained x' is better than x, set $x \leftarrow x'$; otherwise, set $k \leftarrow k + 1$.

Fig. 2. Steps of the basic VND.

Variant of Variable Neighborhood Search (VNS)

Variable Neighborhood Descent (VND)

P. Hansen, N. Mladenović / European Journal of Operational Research 130 (2001) 449–467



Initialization. Select the set of neighborhood structures $\mathcal{N}'_k$, $k = 1, \ldots, k'_{max}$, that will be used in the descent; find an initial solution x;

Repeat the following until no improvement is obtained:

(1) Set $k \leftarrow 1$; (2) Until $k = k'_{max}$, repeat the following steps:

(a) *Exploration of neighborhood.* Find the best neighbor x' of x ($x' \in \mathcal{N}'_k(x)$);

(b) *Move or not.* If the solution thus obtained x' is better than x, set $x \leftarrow x'$; otherwise, set $k \leftarrow k + 1$.

Fig. 2. Steps of the basic VND.

Neighborhood Structures: $\mathcal{N}_k$, $k = 1, 2, \ldots, k_{max}$

$\mathcal{N}_{k+1}$ is larger than $\mathcal{N}_k$, $k = 1, 2, \ldots, k_{max}-1$.

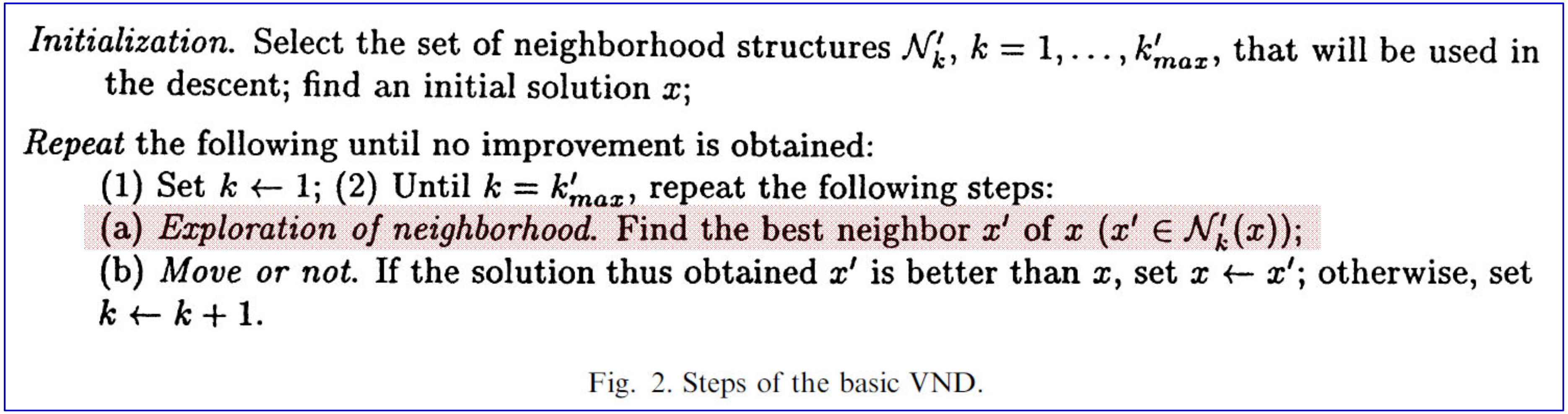
In some cases, $\mathcal{N}_k \subset \mathcal{N}_{k+1}$, $k = 1, 2, \ldots, k_{max}-1$.

$\mathcal{N}_1$ is used first, then $\mathcal{N}_2$, $\mathcal{N}_3$, ...

~~Local search is from a randomly selected neighbor in the current $\mathcal{N}_k$.~~

Variant of Variable Neighborhood Search (VNS)
Variable Neighborhood Descent (VND)

P. Hansen, N. Mladenović / European Journal of Operational Research 130 (2001) 449–467



```
Initialization. Select the set of neighborhood structures $\mathcal{N}'_k$, $k = 1, \ldots, k'_{max}$, that will be used in
    the descent; find an initial solution $x$;
Repeat the following until no improvement is obtained:
    (1) Set $k \leftarrow 1$; (2) Until $k = k'_{max}$, repeat the following steps:
    (a) Exploration of neighborhood. Find the best neighbor $x'$ of $x$ ($x' \in \mathcal{N}'_k(x)$);
    (b) Move or not. If the solution thus obtained $x'$ is better than $x$, set $x \leftarrow x'$; otherwise, set
    $k \leftarrow k + 1$.
```

Fig. 2. Steps of the basic VND.

Neighborhood Structures: $\mathcal{N}_k$, $k = 1, 2, \ldots, k_{max}$

$\mathcal{N}_{k+1}$ is larger than $\mathcal{N}_k$, $k = 1, 2, \ldots, k_{max} - 1$.

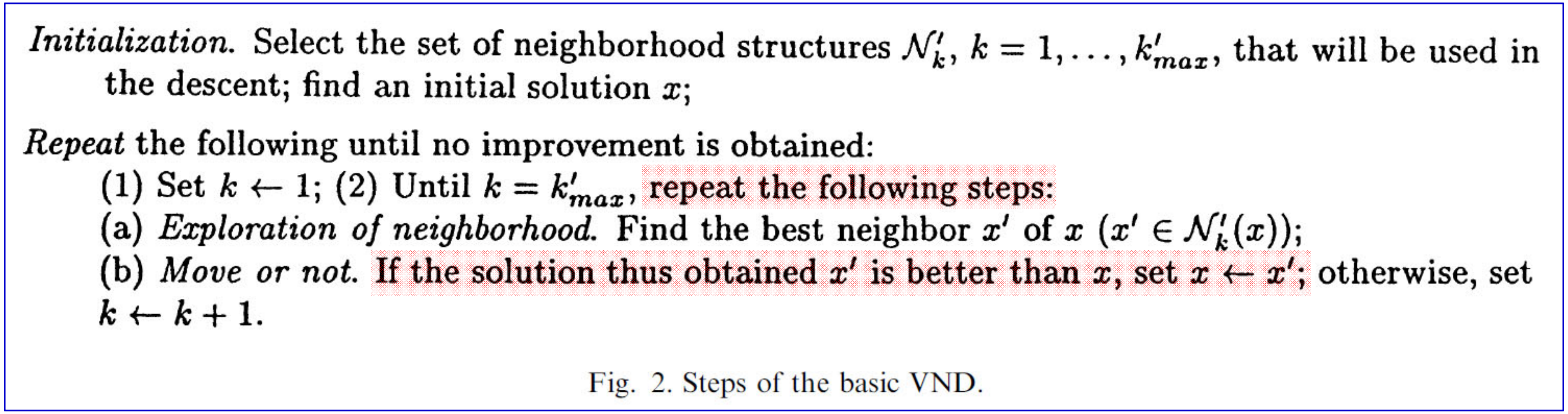
In some cases, $\mathcal{N}_k \subset \mathcal{N}_{k+1}$, $k = 1, 2, \ldots, k_{max} - 1$.

$\mathcal{N}_1$ is used first, then $\mathcal{N}_2$, $\mathcal{N}_3$, ...

One step of the best move local search.

Variant of Variable Neighborhood Search (VNS)
Variable Neighborhood Descent (VND)

P. Hansen, N. Mladenović / European Journal of Operational Research 130 (2001) 449–467



Initialization. Select the set of neighborhood structures $\mathcal{N}'_k$, $k = 1, \dots, k'_{max}$, that will be used in the descent; find an initial solution x;

Repeat the following until no improvement is obtained:

(1) Set $k \leftarrow 1$; (2) Until $k = k'_{max}$, repeat the following steps:

(a) *Exploration of neighborhood.* Find the best neighbor x' of x ($x' \in \mathcal{N}'_k(x)$);

(b) *Move or not.* If the solution thus obtained x' is better than x, set $x \leftarrow x'$; otherwise, set $k \leftarrow k + 1$.

Fig. 2. Steps of the basic VND.

Neighborhood Structures: $\mathcal{N}_k$, $k = 1, 2, \dots, k_{max}$

$\mathcal{N}_{k+1}$ is larger than $\mathcal{N}_k$, $k = 1, 2, \dots, k_{max} - 1$.

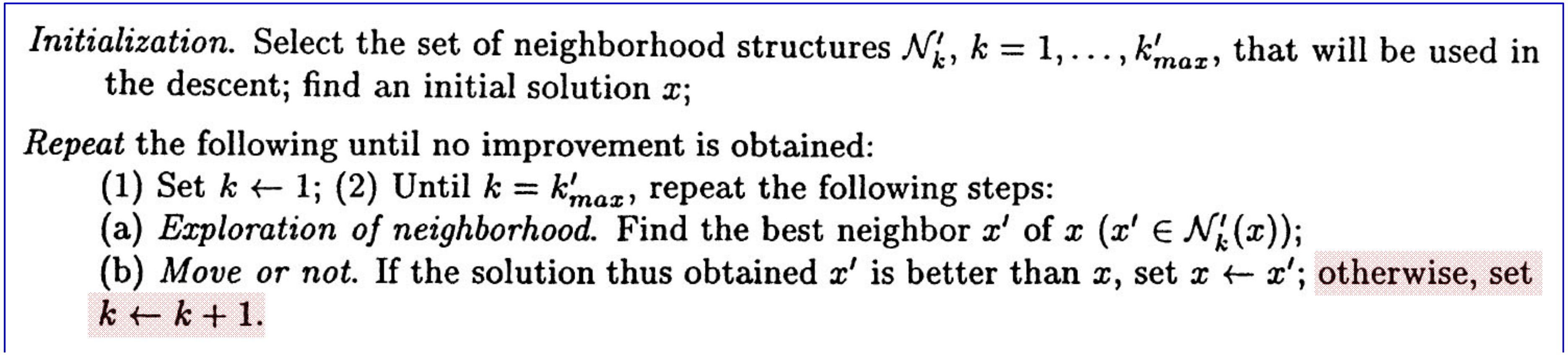
In some cases, $\mathcal{N}_k \subset \mathcal{N}_{k+1}$, $k = 1, 2, \dots, k_{max} - 1$.

$\mathcal{N}_1$ is used first, then $\mathcal{N}_2$, $\mathcal{N}_3$, ...

One step of the best move local search. (This is iterated)

Variant of Variable Neighborhood Search (VNS)
Variable Neighborhood Descent (VND)

P. Hansen, N. Mladenović / European Journal of Operational Research 130 (2001) 449–467



```
Initialization. Select the set of neighborhood structures $\mathcal{N}'_k$, $k = 1, \ldots, k'_{max}$, that will be used in
    the descent; find an initial solution $x$;
Repeat the following until no improvement is obtained:
    (1) Set $k \leftarrow 1$; (2) Until $k = k'_{max}$, repeat the following steps:
    (a) Exploration of neighborhood. Find the best neighbor $x'$ of $x$ ($x' \in \mathcal{N}'_k(x)$);
    (b) Move or not. If the solution thus obtained $x'$ is better than $x$, set $x \leftarrow x'$; otherwise, set
    $k \leftarrow k + 1$.
```

If no improvement, use the next neighborhood ($\mathcal{N}_k$ ==> $\mathcal{N}_{k+1}$).

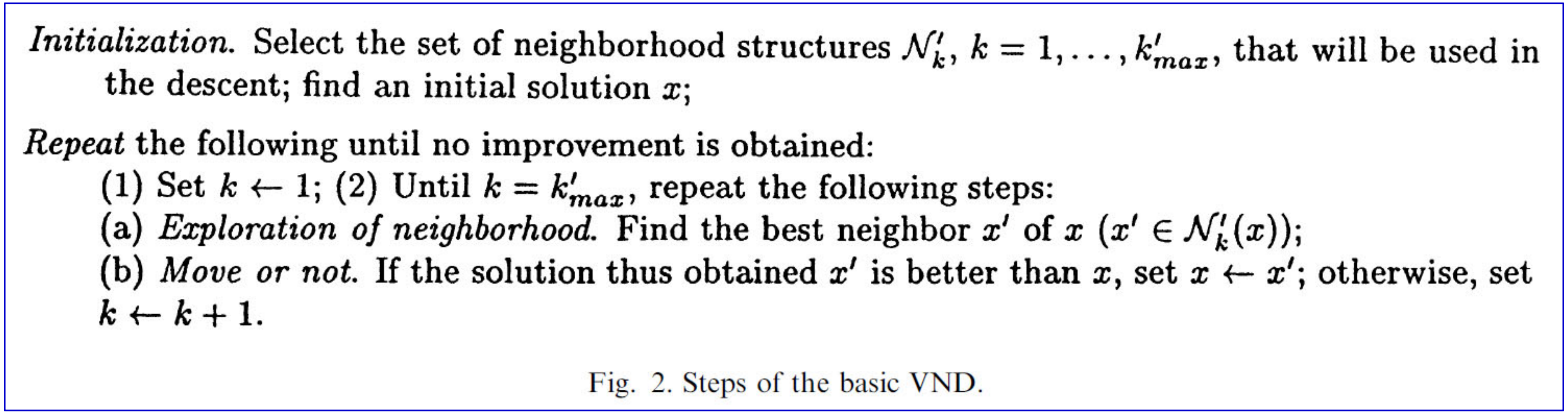
Neighborhood Structures: $\mathcal{N}_k$, $k = 1, 2, \ldots, k_{max}$

$\mathcal{N}_{k+1}$ is larger than $\mathcal{N}_k$, $k = 1, 2, \ldots, k_{max}-1$.

In some cases, $\mathcal{N}_k \subset \mathcal{N}_{k+1}$, $k = 1, 2, \ldots, k_{max}-1$.

$\mathcal{N}_1$ is used first, then $\mathcal{N}_2$, $\mathcal{N}_3$, ...

One step of the best move local search. (This is iterated)



Initialization. Select the set of neighborhood structures $\mathcal{N}'_k$, $k = 1, \ldots, k'_{max}$, that will be used in the descent; find an initial solution x;

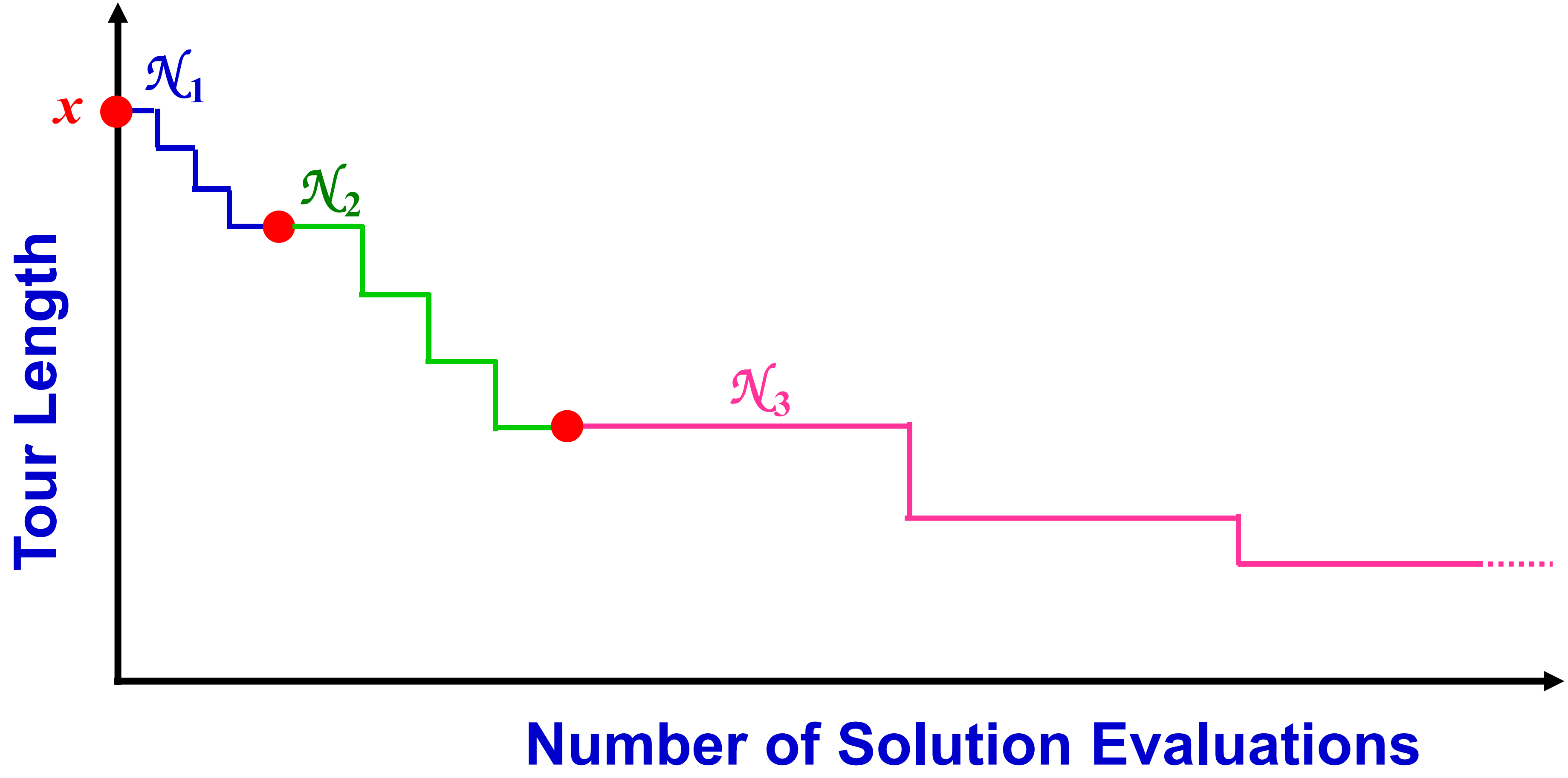
Repeat the following until no improvement is obtained:

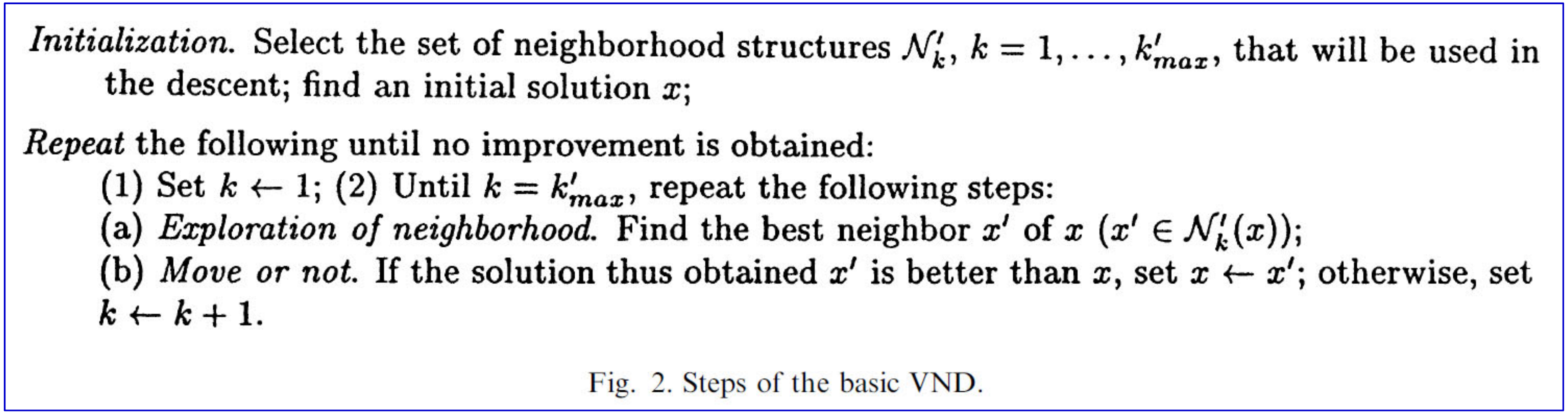
(1) Set $k \leftarrow 1$; (2) Until $k = k'_{max}$, repeat the following steps:

(a) *Exploration of neighborhood.* Find the best neighbor x' of x ($x' \in \mathcal{N}'_k(x)$);

(b) *Move or not.* If the solution thus obtained x' is better than x, set $x \leftarrow x'$; otherwise, set $k \leftarrow k + 1$.

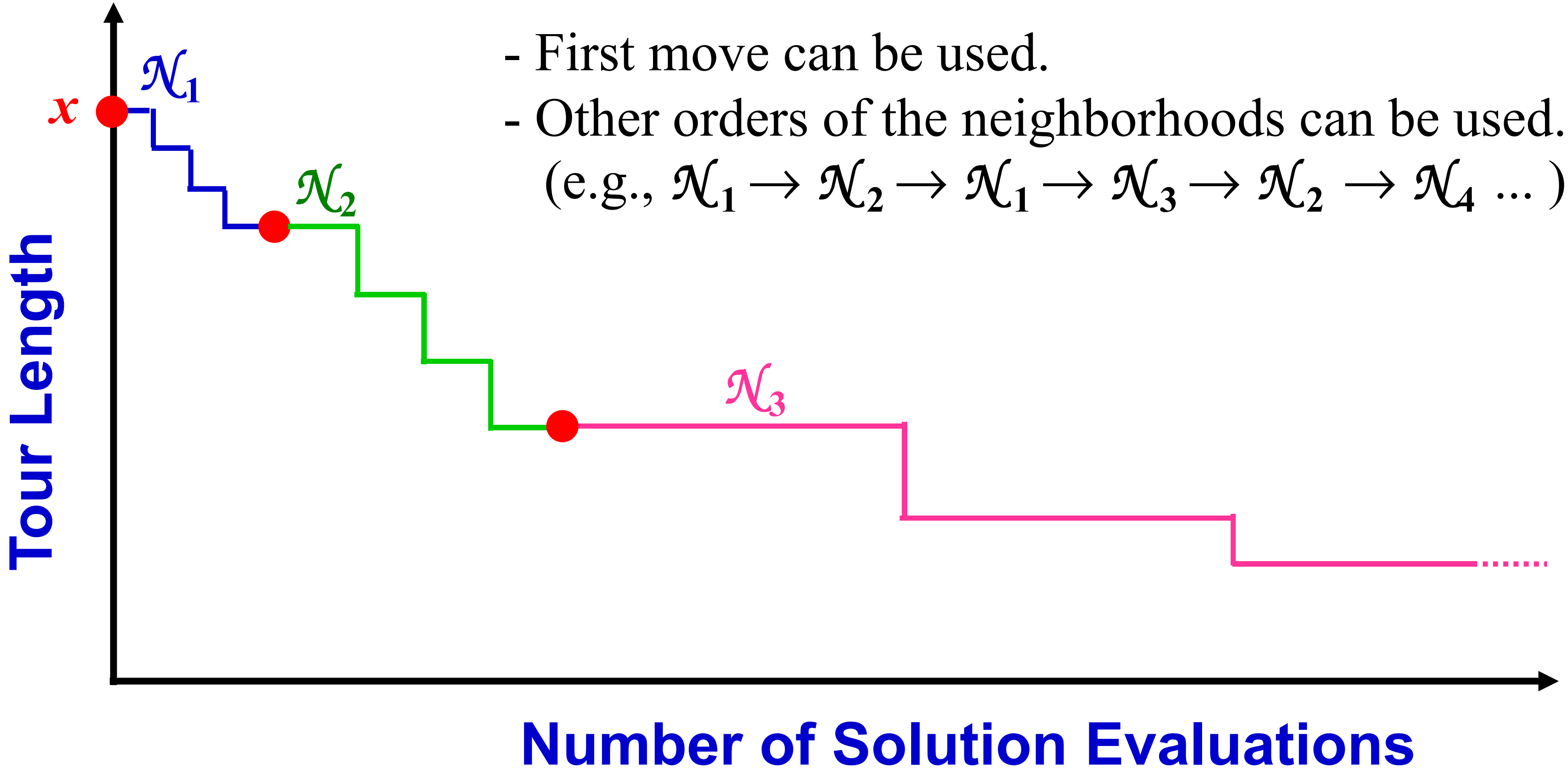
Fig. 2. Steps of the basic VND.





```
*Initialization.* Select the set of neighborhood structures $\mathcal{N}'_k$, $k = 1, \ldots, k'_{max}$, that will be used in
    the descent; find an initial solution $x$;
*Repeat* the following until no improvement is obtained:
    (1) Set $k \leftarrow 1$; (2) Until $k = k'_{max}$, repeat the following steps:
    (a) *Exploration of neighborhood.* Find the best neighbor $x'$ of $x$ ($x' \in \mathcal{N}'_k(x)$);
    (b) *Move or not.* If the solution thus obtained $x'$ is better than $x$, set $x \leftarrow x'$; otherwise, set
    $k \leftarrow k + 1$.
```

Fig. 2. Steps of the basic VND.



- First move can be used.
- Other orders of the neighborhoods can be used.
  (e.g., $\mathcal{N}_1 \rightarrow \mathcal{N}_2 \rightarrow \mathcal{N}_1 \rightarrow \mathcal{N}_3 \rightarrow \mathcal{N}_2 \rightarrow \mathcal{N}_4$ ...)

A Tutorial on Variable Neighborhood Search

Pierre Hansen
GERAD and HEC Montreal
Nenad Mladenović
GERAD and Mathematical Institute, SANU, Belgrade

Basic Ideas behind Variable Neighborhood Search

Fact 1 *A local minimum with respect to one neighborhood structure is not necessary so for another;*

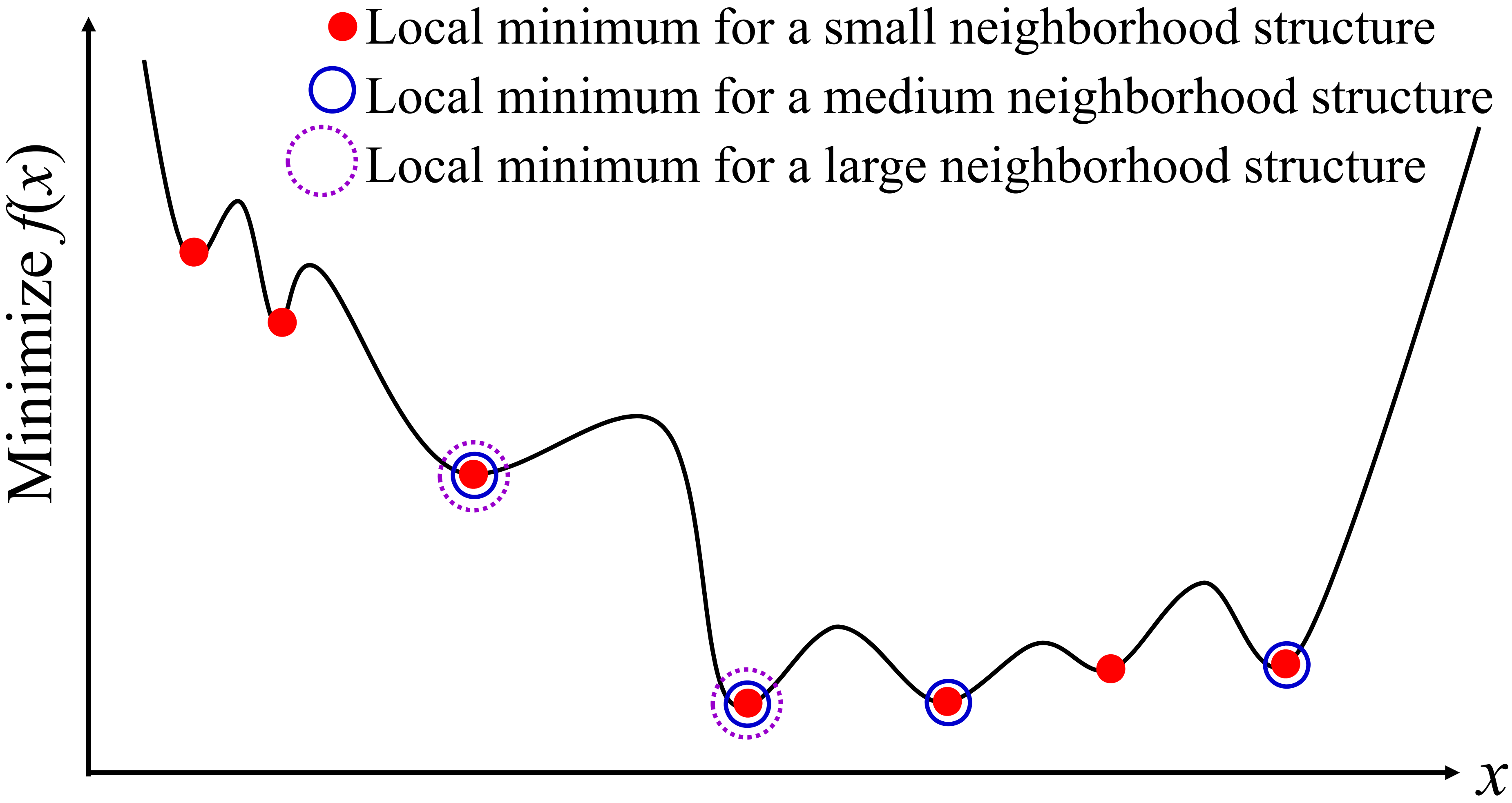
Fact 2 *A global minimum is a local minimum with respect to all possible neighborhood structures.*

Fact 3 *For many problems local minima with respect to one or several neighborhoods are relatively close to each other.*

Fact 1 *A local minimum with respect to one neighborhood structure is not necessary so for another;*

Fact 2 *A global minimum is a local minimum with respect to all possible neighborhood structures.*

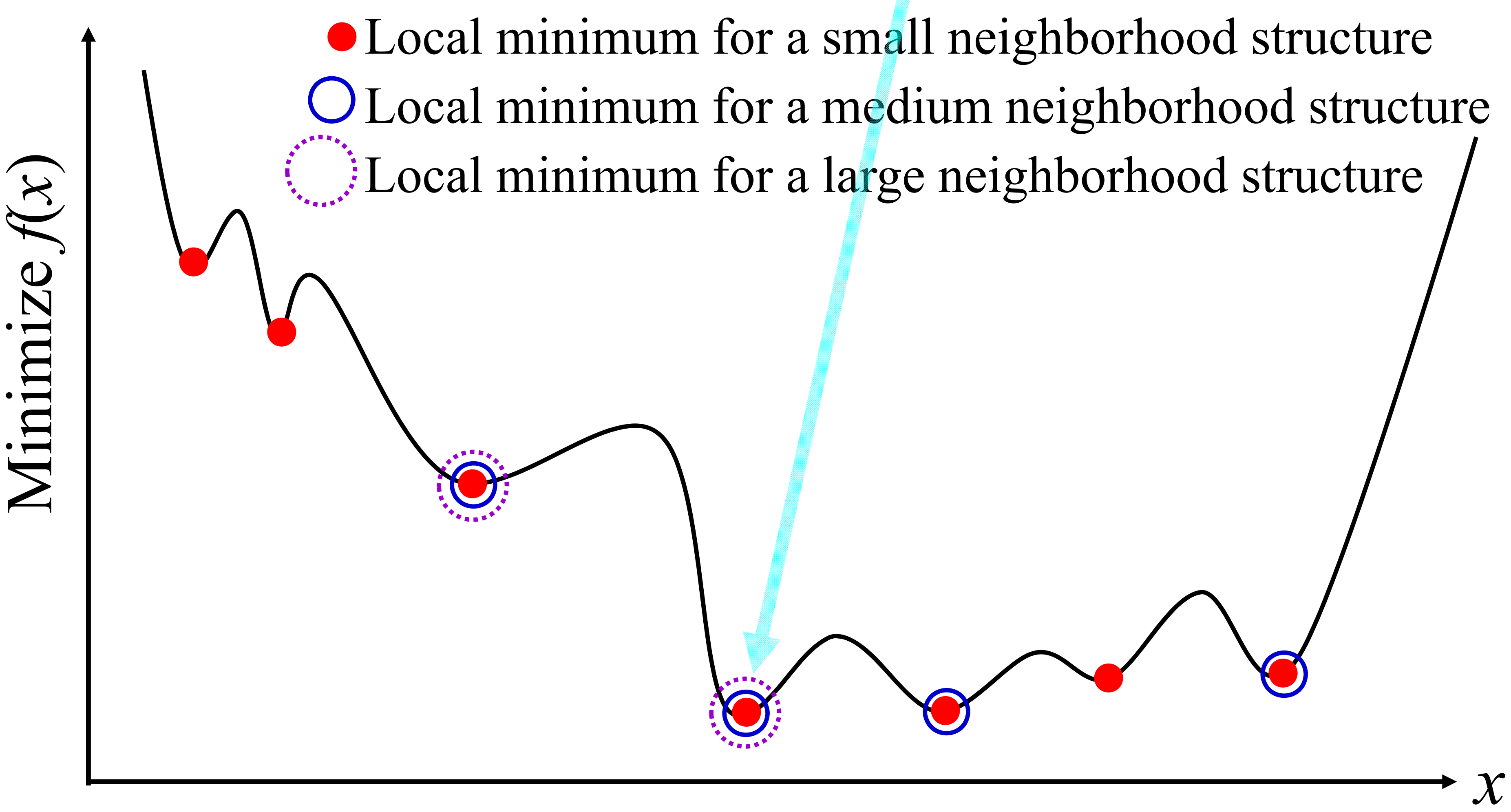
Fact 3 *For many problems local minima with respect to one or several neighborhoods are relatively close to each other.*


● Local minimum for a small neighborhood structure
○ Local minimum for a medium neighborhood structure
◌ Local minimum for a large neighborhood structure
Minimize $f(x)$
x

Fact 1 *A local minimum with respect to one neighborhood structure is not necessary so for another;*

Fact 2 *A global minimum is a local minimum with respect to all possible neighborhood structures.*

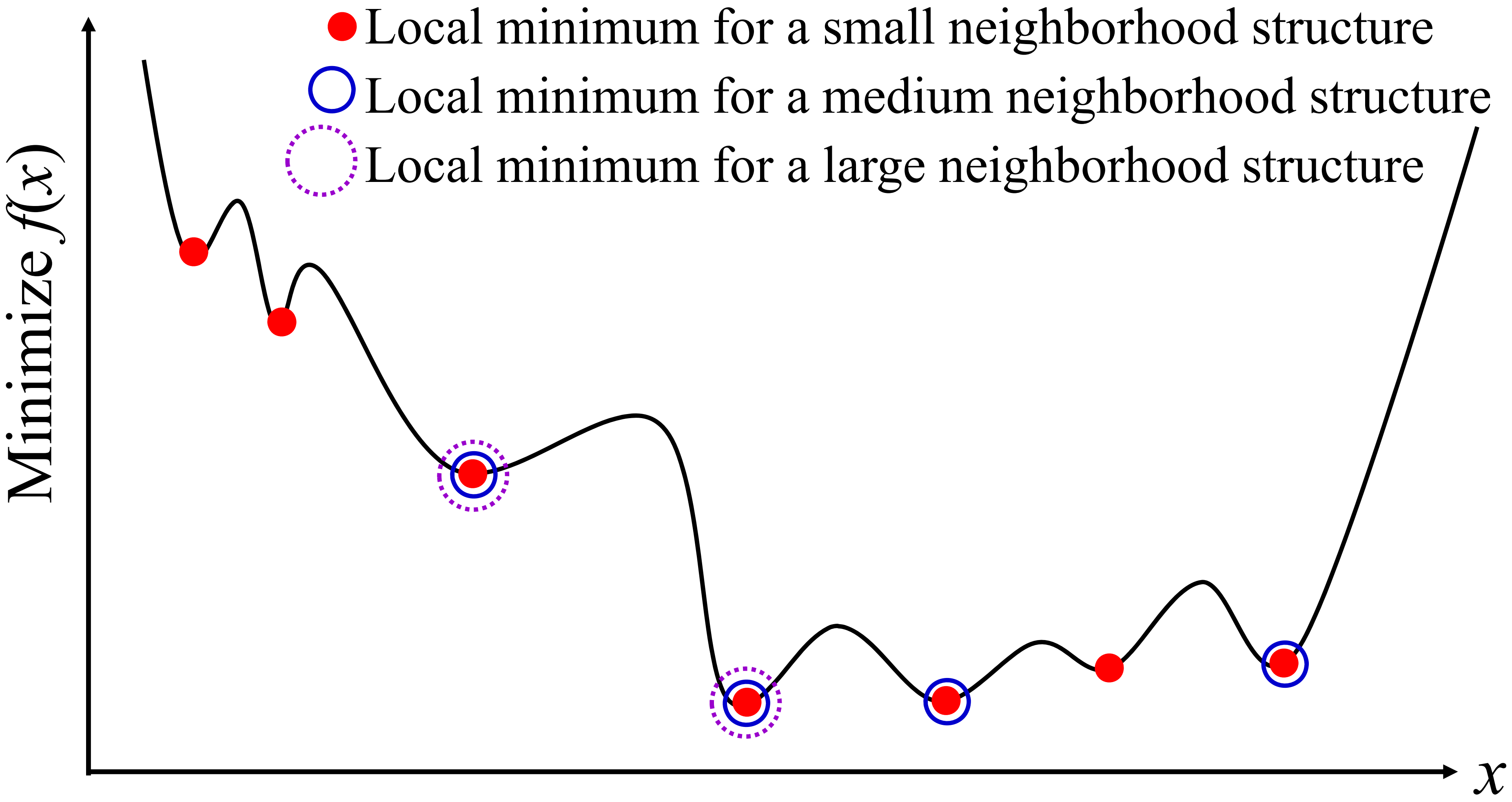
Fact 3 *For many problems local minima with respect to one or several neighborhoods are relatively close to each other.*


● Local minimum for a small neighborhood structure
○ Local minimum for a medium neighborhood structure
○ Local minimum for a large neighborhood structure
Minimize $f(x)$
x

Fact 1 *A local minimum with respect to one neighborhood structure is not necessary so for another;*

Fact 2 *A global minimum is a local minimum with respect to all possible neighborhood structures.*

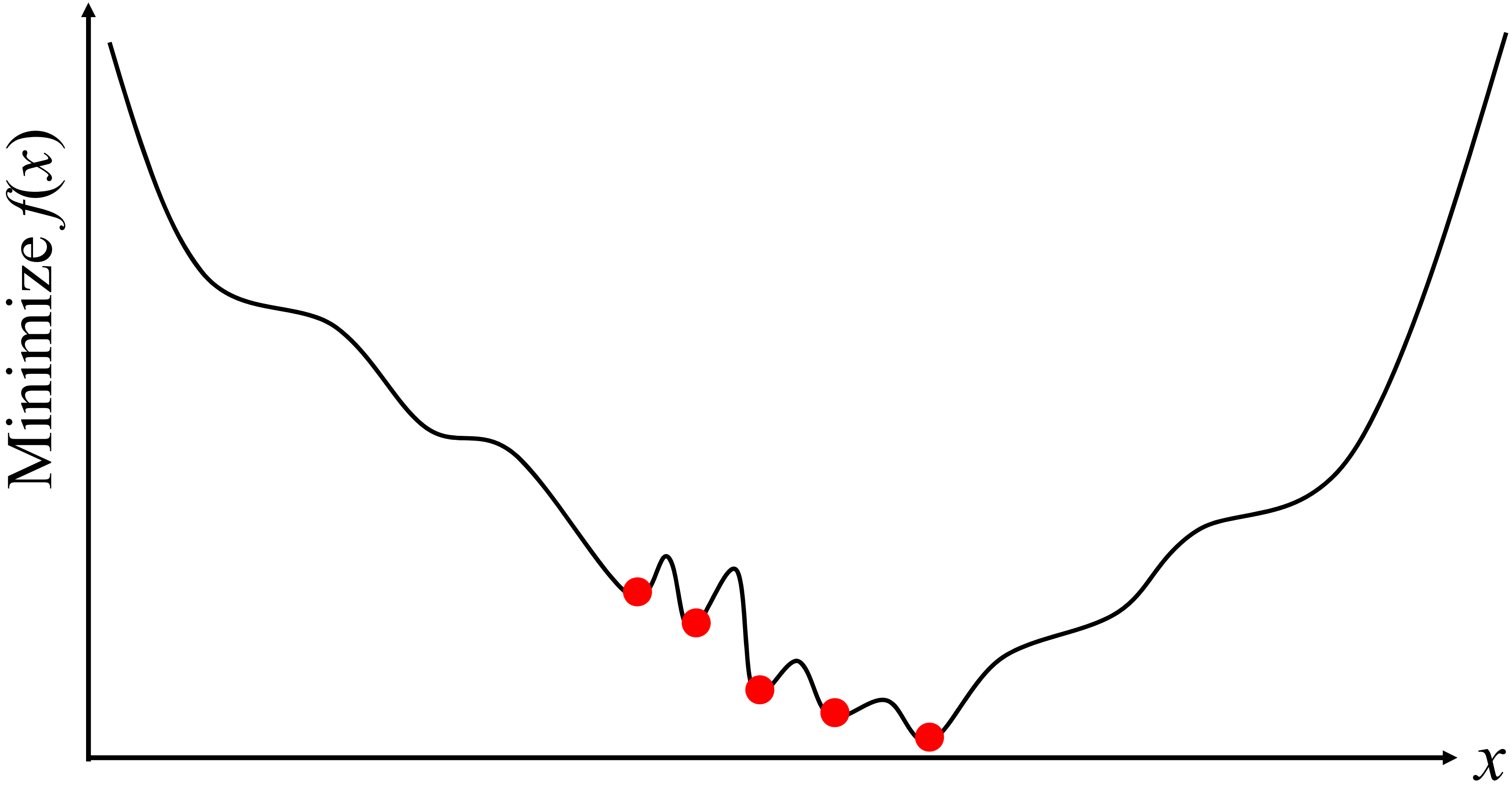
Fact 3 *For many problems local minima with respect to one or several neighborhoods are relatively close to each other.*


● Local minimum for a small neighborhood structure
○ Local minimum for a medium neighborhood structure
◌ Local minimum for a large neighborhood structure
Minimize $f(x)$
x

Fact 1 *A local minimum with respect to one neighborhood structure is not necessary so for another;*

Fact 2 *A global minimum is a local minimum with respect to all possible neighborhood structures.*

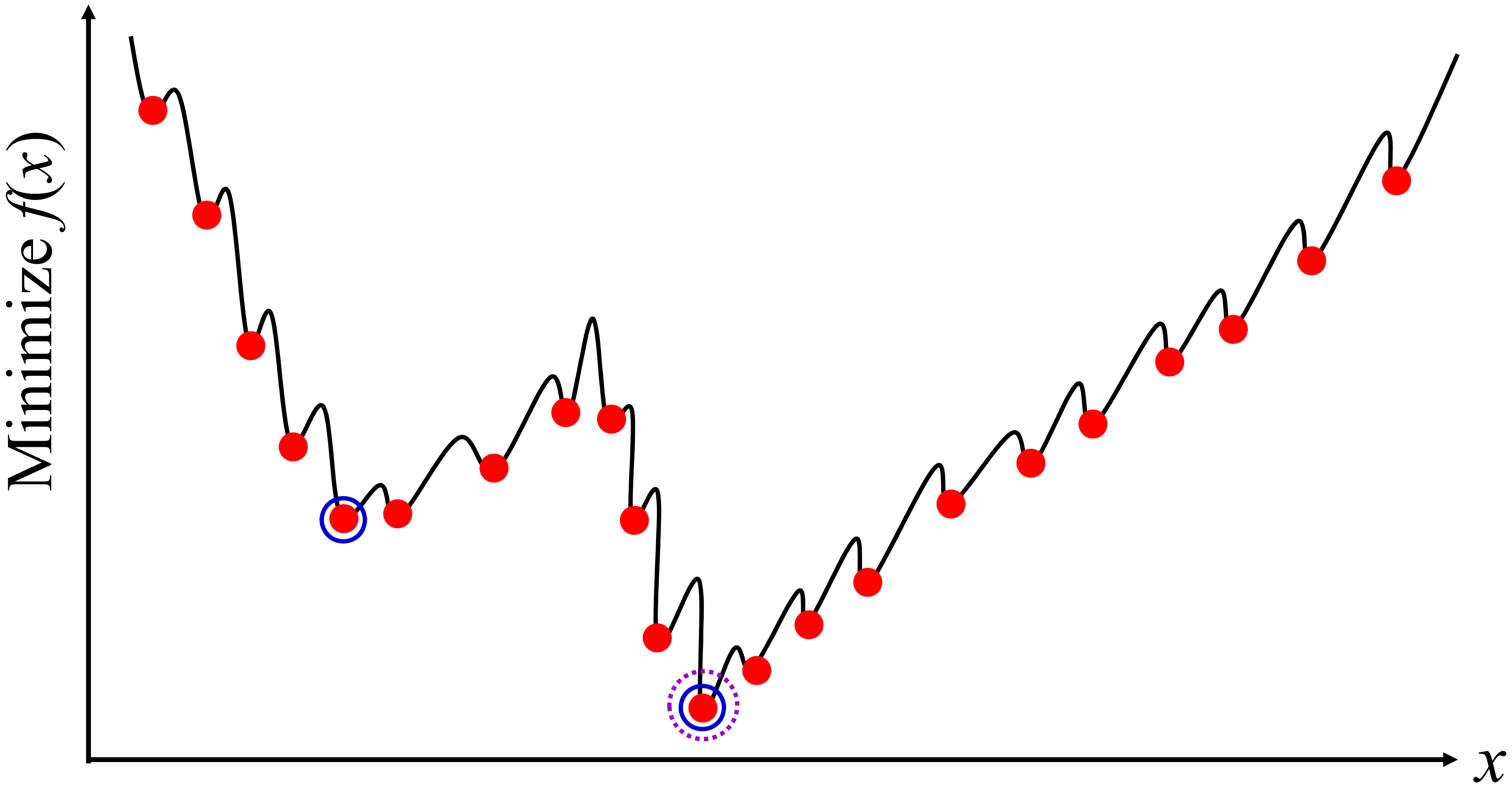
Fact 3 *For many problems local minima with respect to one or several neighborhoods are relatively close to each other.*



Fact 1 *A local minimum with respect to one neighborhood structure is not necessary so for another;*

Fact 2 *A global minimum is a local minimum with respect to all possible neighborhood structures.*

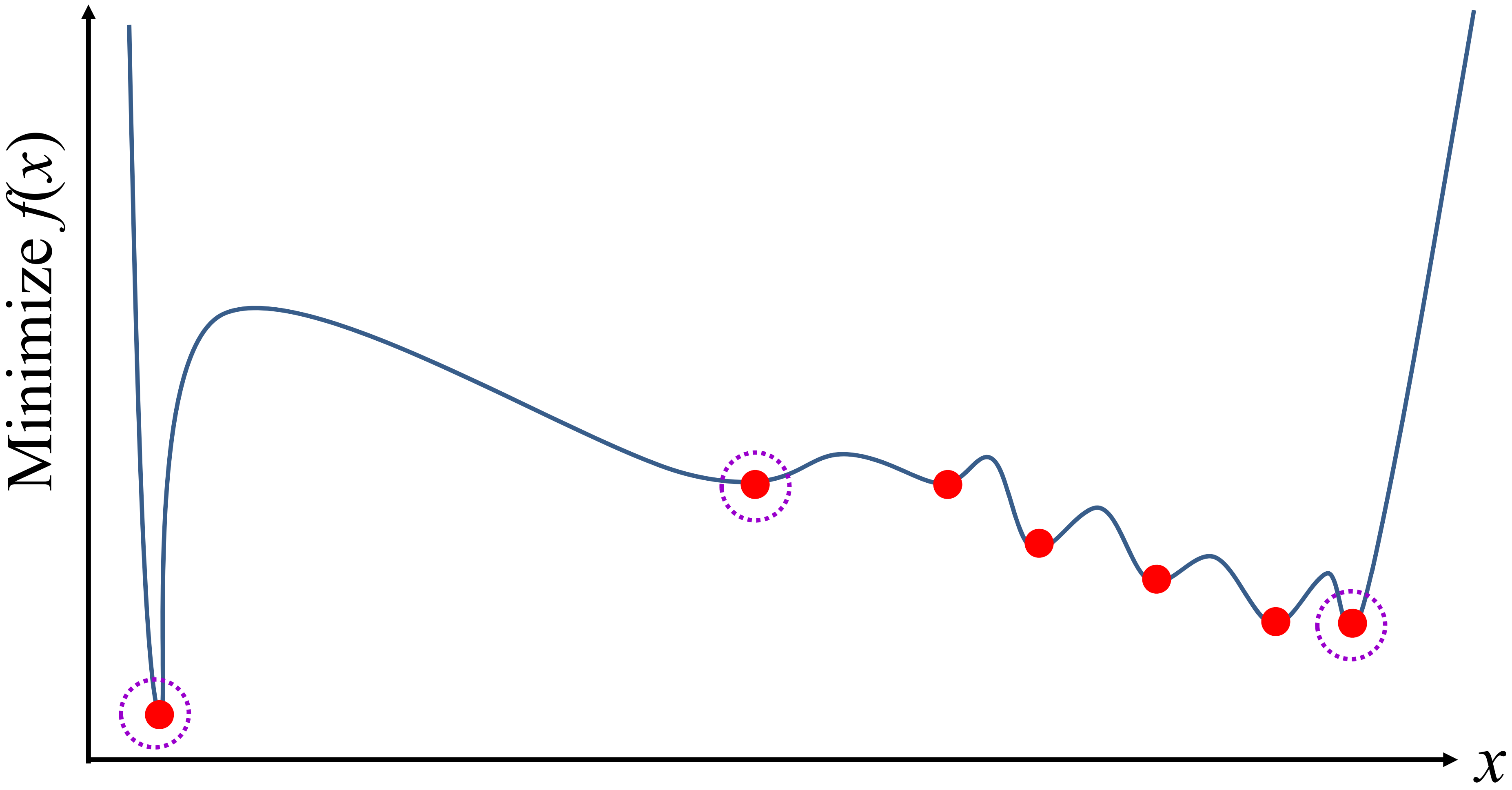
Fact 3 *For many problems local minima with respect to one or several neighborhoods are relatively close to each other.*



Fact 1 *A local minimum with respect to one neighborhood structure is not necessary so for another;*

Fact 2 *A global minimum is a local minimum with respect to all possible neighborhood structures.*

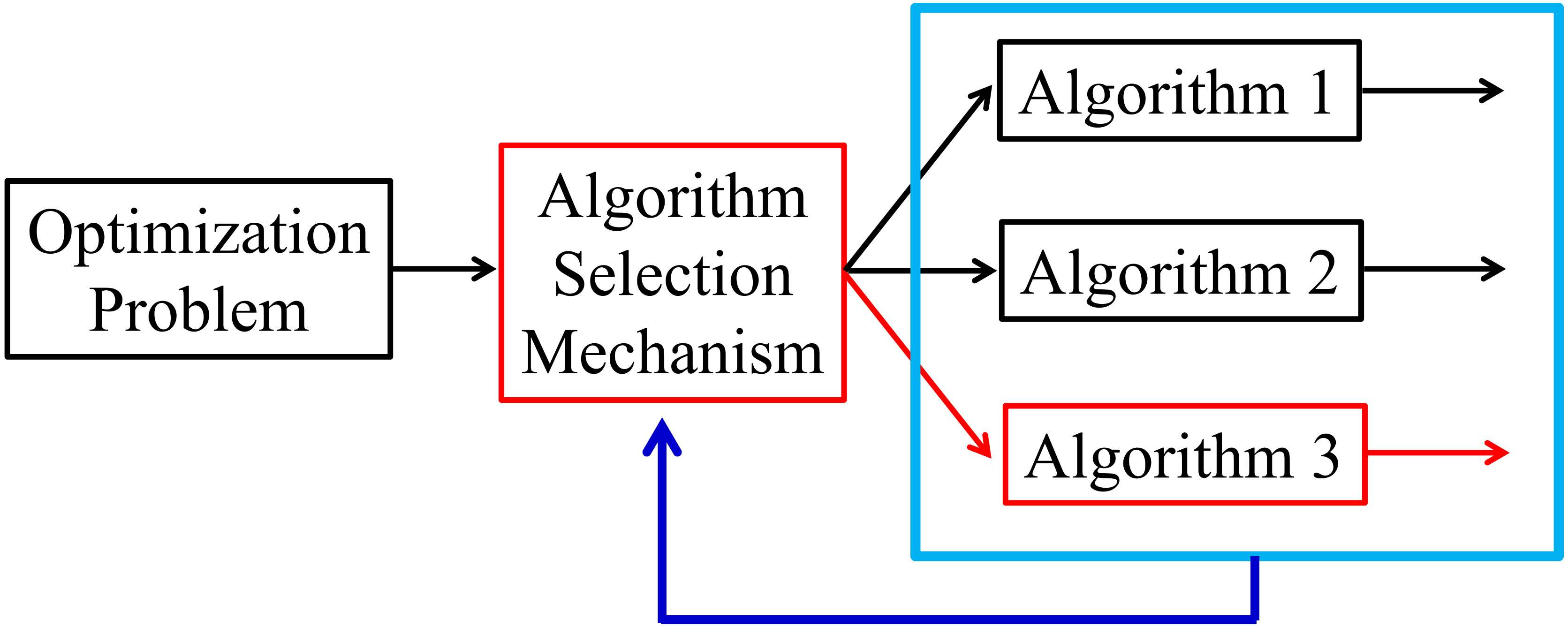
Fact 3 *For many problems local minima with respect to one or several neighborhoods are relatively close to each other.*



Variable Neighborhood Search

Algorithm Ensemble

Based on the characteristics of the problem, an appropriate algorithm will be selected.

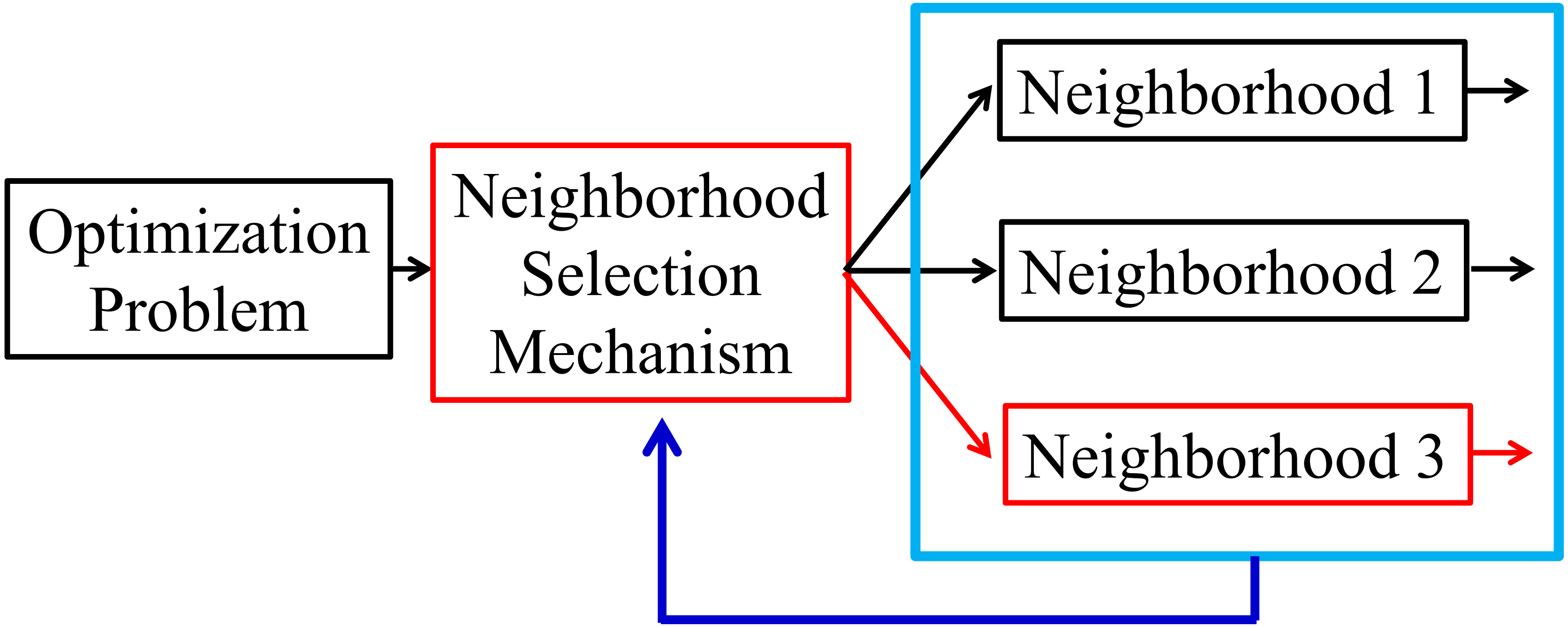


Depending on the progress of the search, the selection of an appropriate algorithm will be changed.

Variable Neighborhood Search

Algorithm Ensemble

Based on the characteristics of the problem, an appropriate algorithm will be selected.



Depending on the progress of the search, the selection of an appropriate neighborhood will be changed.

Lab Session Task 3

Specify neighborhood structures $\mathcal{N}_1$, $\mathcal{N}_2$, $\mathcal{N}_3$, ... for designing a Variable Neighborhood Descent (VND) algorithm for a TSP problem.