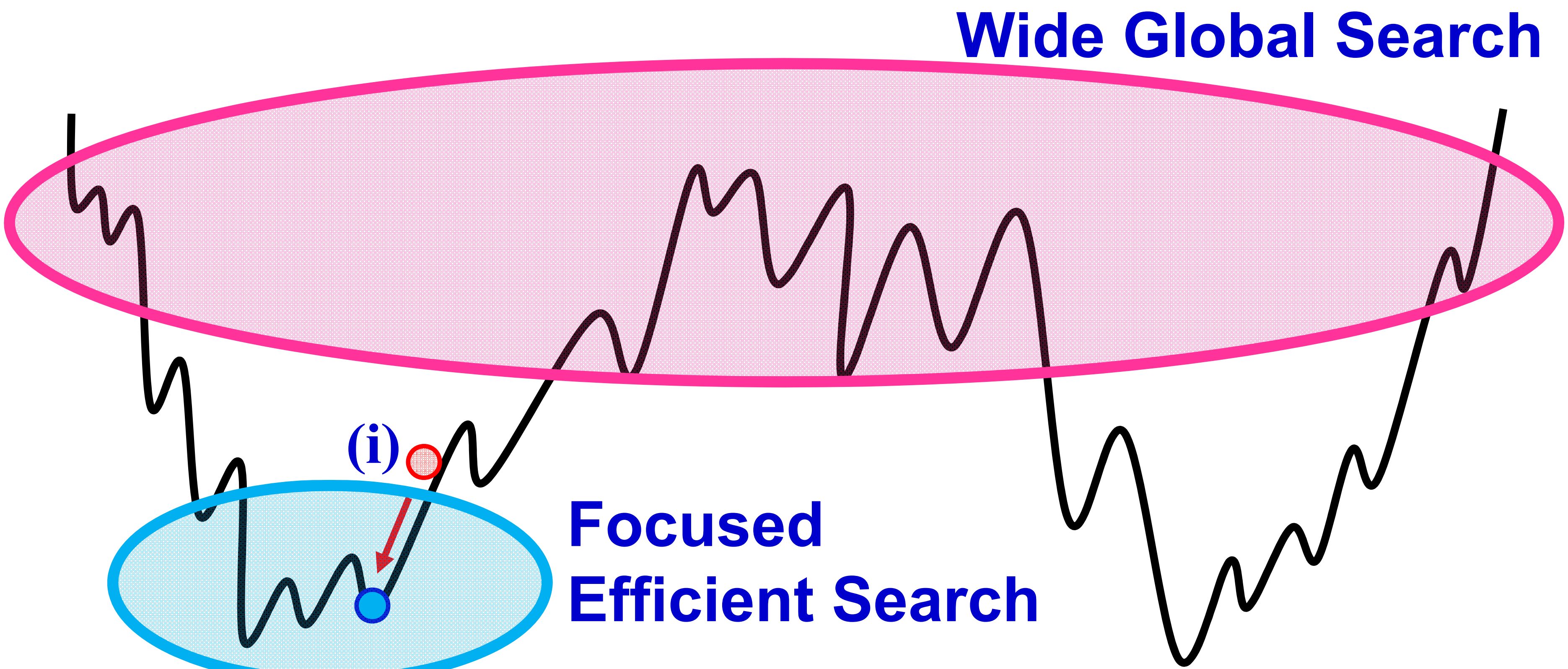
Optimization Methods

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.

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)

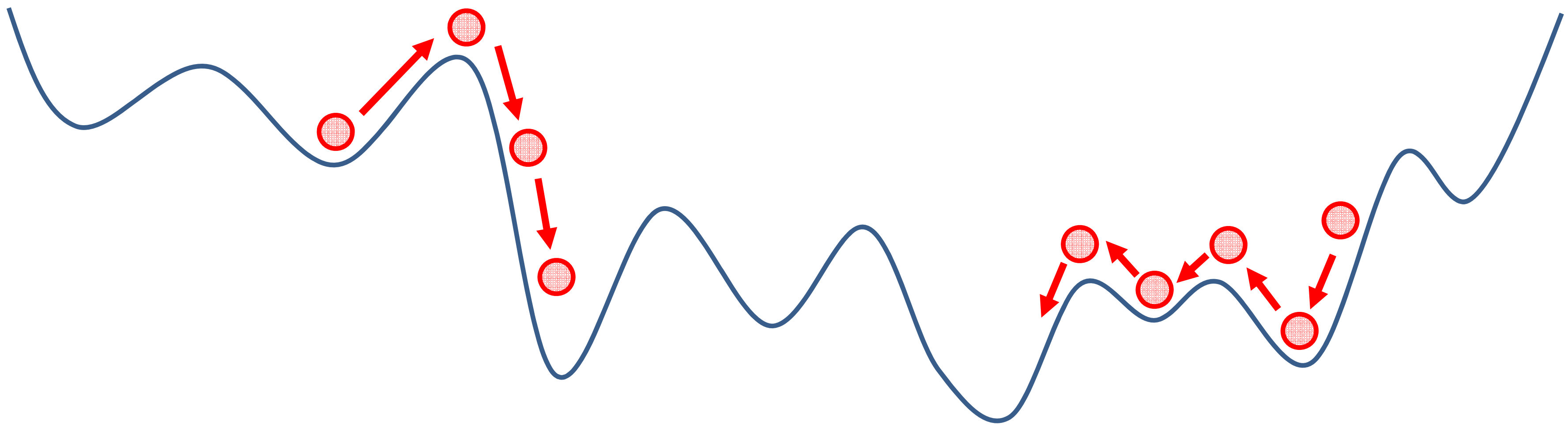


Move to a Better Solution

- Local Search (LS)
- Iterated Local Search (ILS)
- Variable Neighborhood Search (VNS)

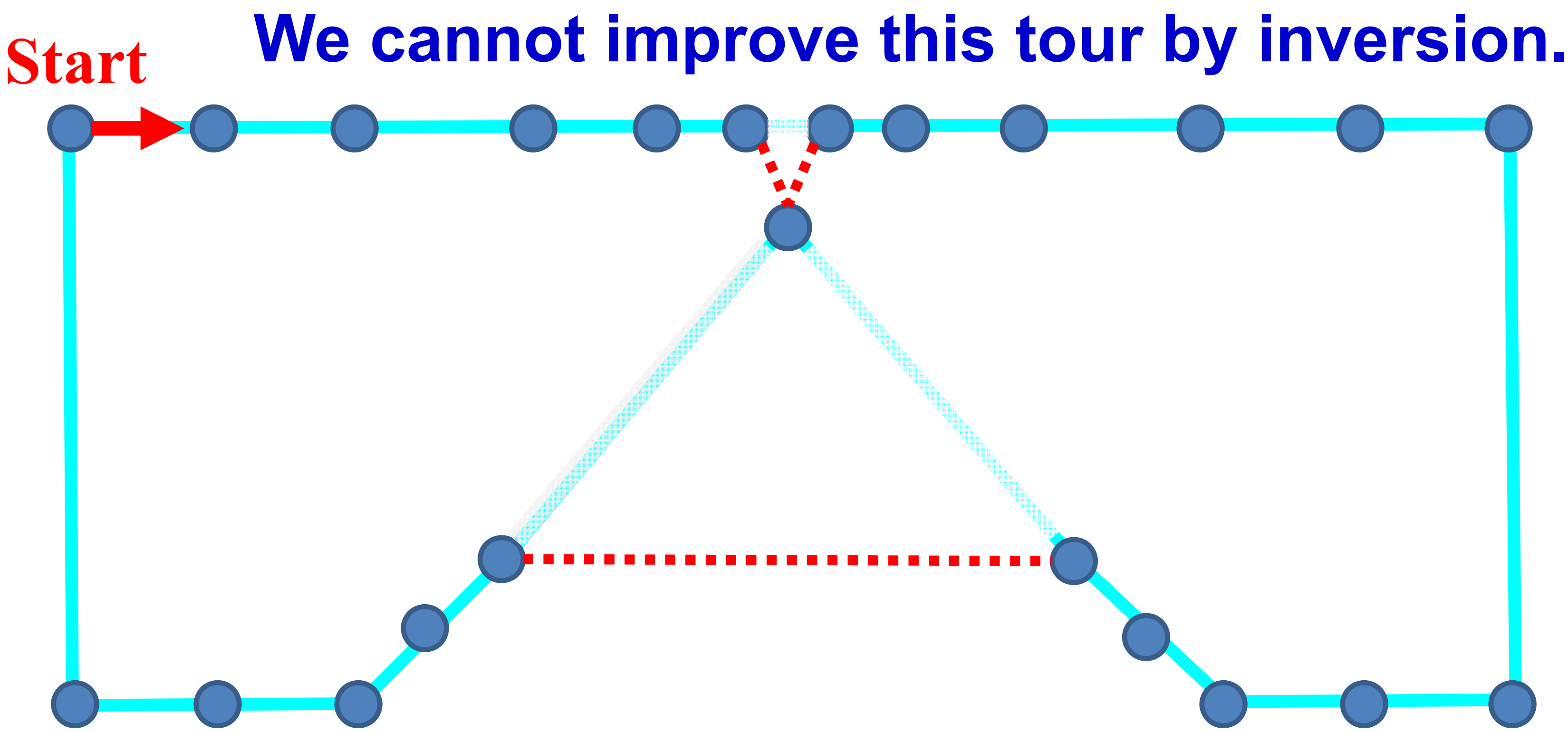
Allow the Move to a Worse Solution

- Simulated Annealing (SA)
- Tabu Search (TS)



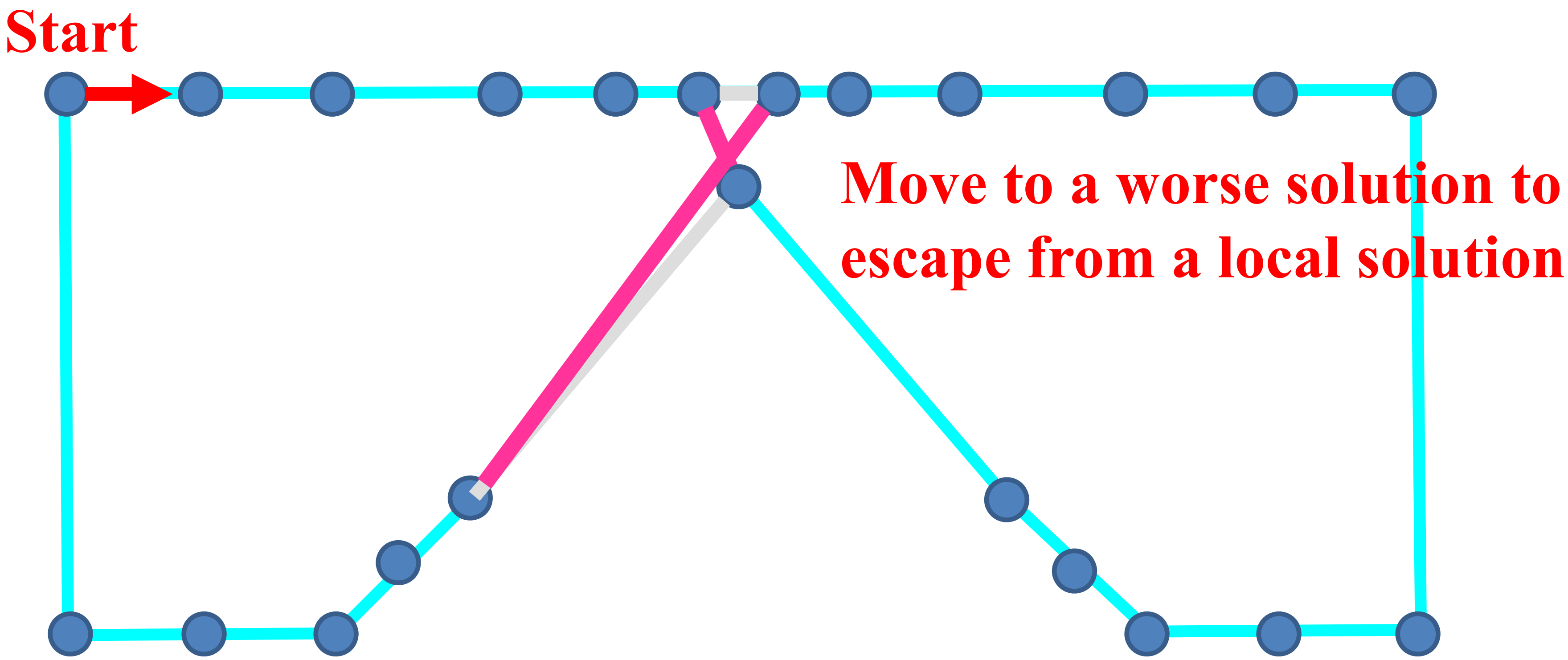
Task 1

Create a TSP problem where a greedy solution obtained from an initial city 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.



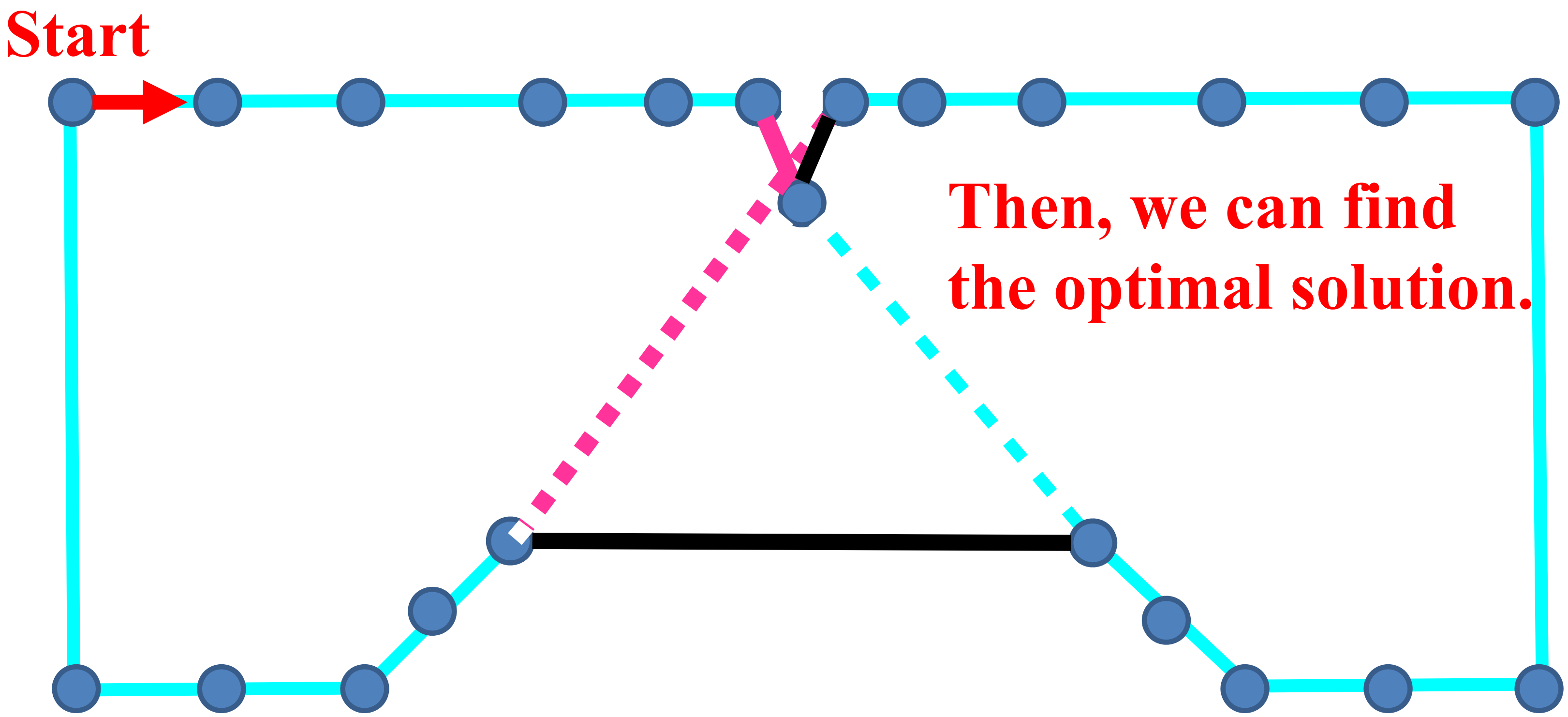
Task 1

Create a TSP problem where a greedy solution obtained from an initial city 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.



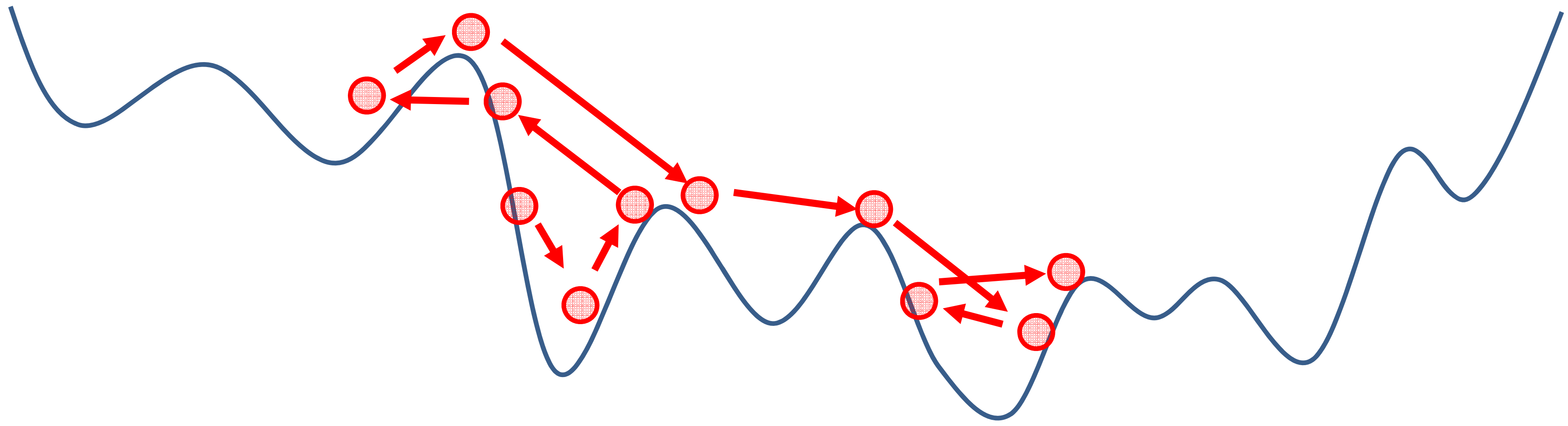
Task 1

Create a TSP problem where a greedy solution obtained from an initial city 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.



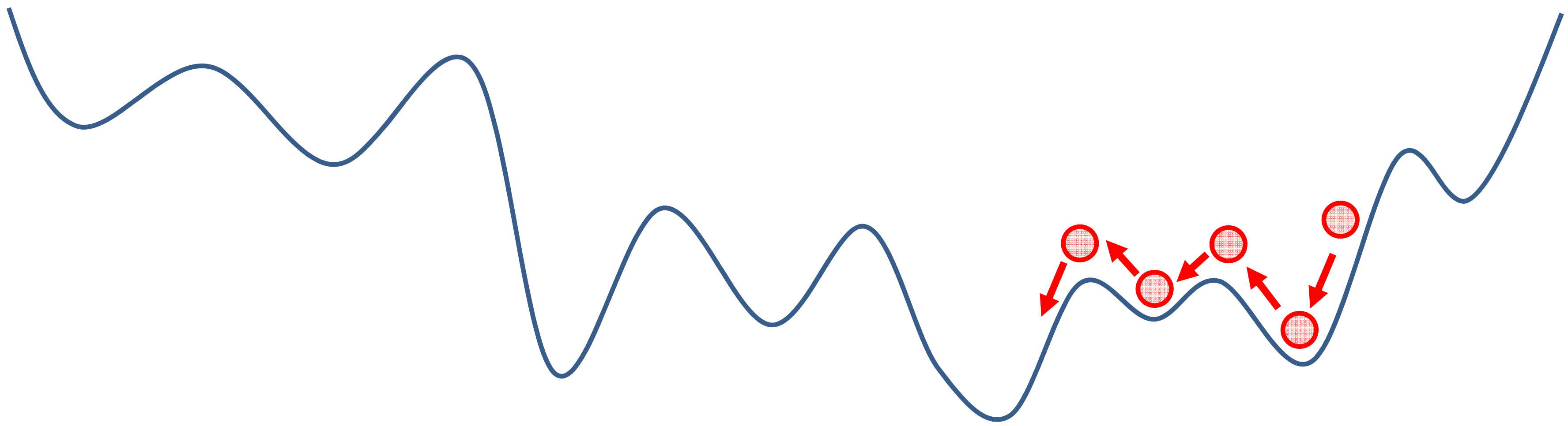
Simulated Annealing (SA)

Find a promising region in the early stage of search.



Tabu Search (TS)

Carefully search for the best solution from a good initial solution.



Google Scholar

Simulated Annealing

Scott Kirkpatrick

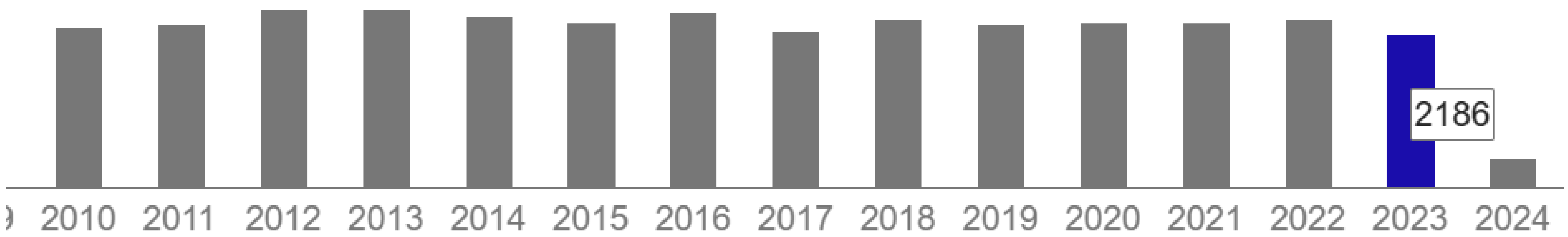
FOLLOW

Professor, School of Engineering and Computer Science, Hebrew University
Verified email at cs.huji.ac.il - Homepage

condensed matter physics   computer science   optimization   networks

TITLE	CITED BY	YEAR
Optimization by simulated annealing S Kirkpatrick, CD Gelatt Jr, MP Vecchi science 220 (4598), 671-680	59477	1983

Cited by 59477



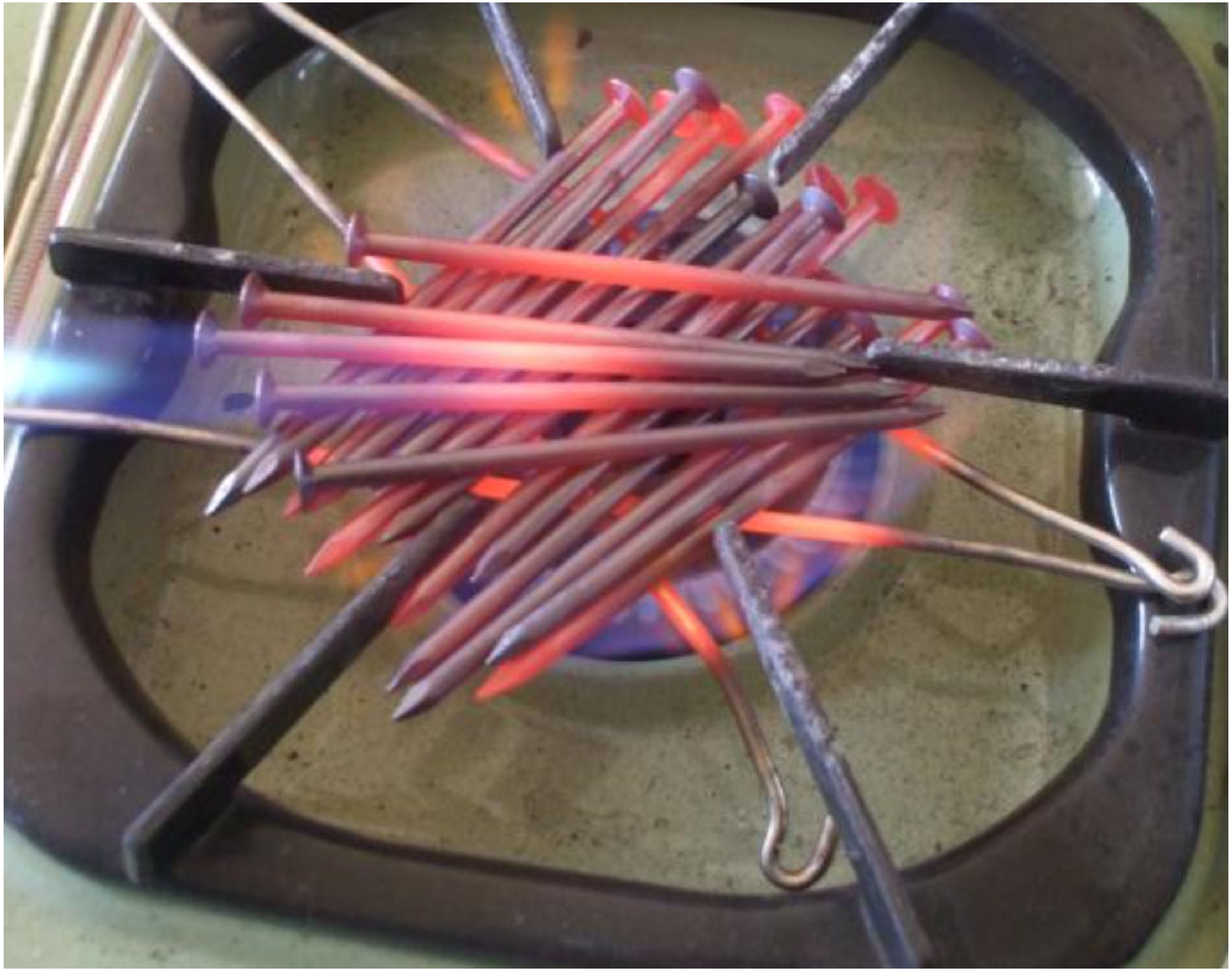
ARTICLES

Optimization by Simulated Annealing

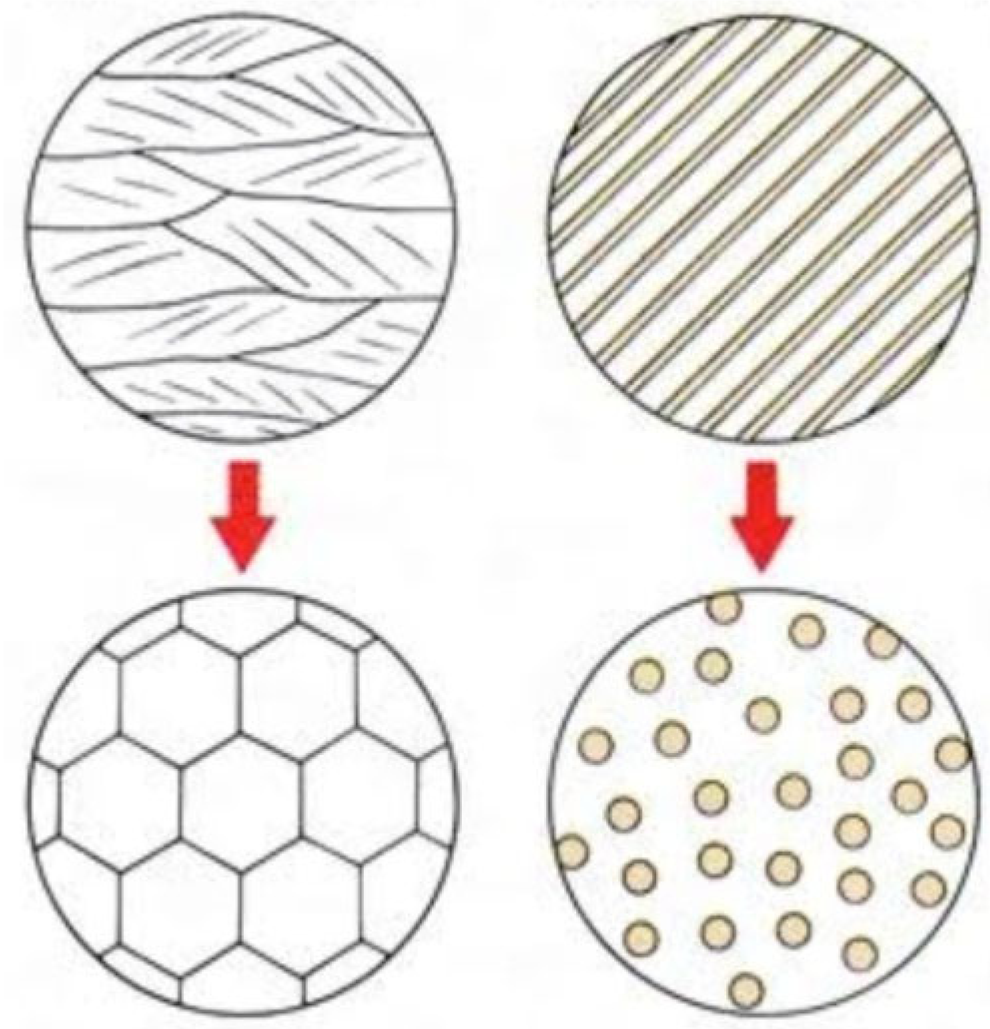
S. Kirkpatrick[1], C. D. Gelatt Jr.[1], M. P. Vecchi[2]
+ See all authors and affiliations

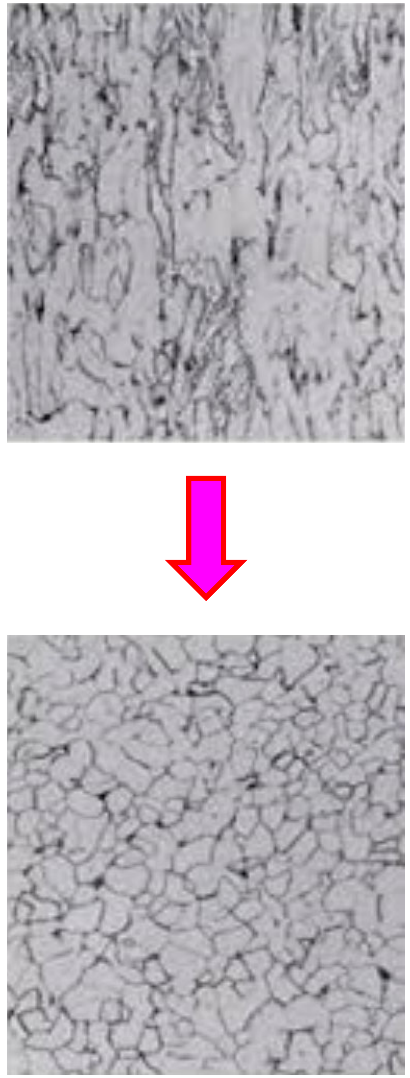
Science 13 May 1983:
Vol. 220, Issue 4598, pp. 671-680
DOI: 10.1126/science.220.4598.671

Annealing



Annealing





Simulated Annealing

$f(\boldsymbol{x})$: Objective function

$\boldsymbol{x}$: Current solution

$\boldsymbol{y}$: Neighbor solution of $\boldsymbol{x}$

Δf : Deterioration of the objective value by the move from $\boldsymbol{x}$ to $\boldsymbol{y}$.

for minimization problems: $\Delta f = f(\boldsymbol{y}) - f(\boldsymbol{x})$

for maximization problems: $\Delta f = f(\boldsymbol{x}) - f(\boldsymbol{y})$

Acceptance probability of $\boldsymbol{y}$:

$$P(\boldsymbol{x} \rightarrow \boldsymbol{y}) = \min\left\{1, \exp\left(\frac{-\Delta f}{T}\right)\right\}$$

where T is a control parameter called the temperature.

Simulated Annealing

Δf : Deterioration of the objective value by the move from $\boldsymbol{x}$ to $\boldsymbol{y}$.

Acceptance probability of $\boldsymbol{y}$:

$$P(\boldsymbol{x} \rightarrow \boldsymbol{y}) = \min\left\{1, \exp\left(\frac{-\Delta f}{T}\right)\right\}$$

where T is a control parameter called the temperature.

The acceptance probability can be rewritten as

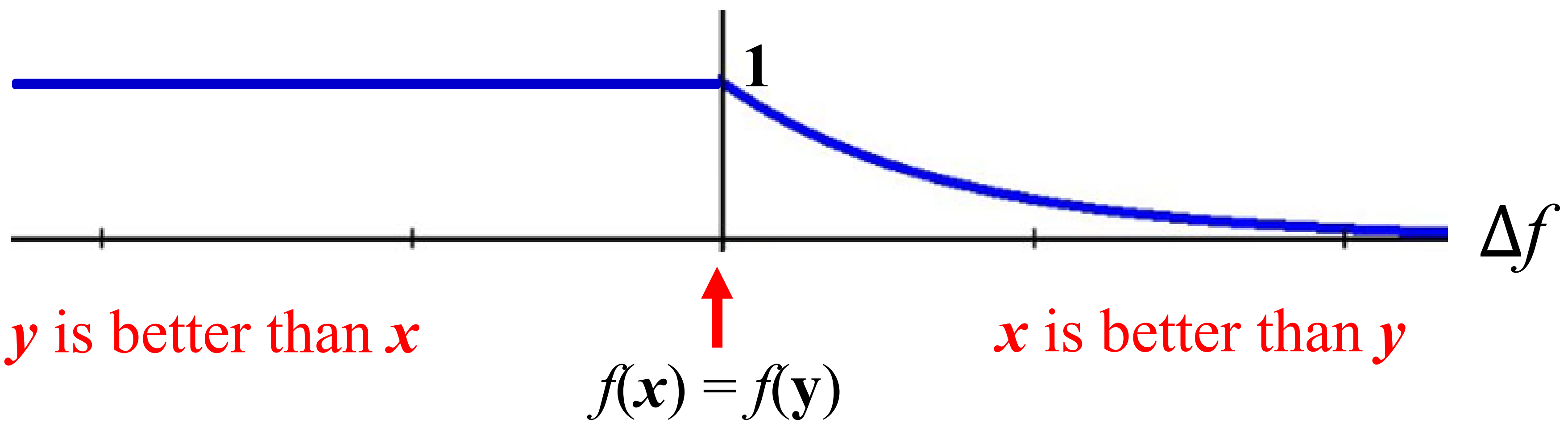
$$P(\boldsymbol{x} \rightarrow \boldsymbol{y}) = \begin{cases} 1, & \text{if } \boldsymbol{y} \text{ is better than } \boldsymbol{x} \text{ (i.e., if } \Delta f < 0) \\ \exp\left(\frac{-\Delta f}{T}\right), & \text{otherwise} \end{cases}$$

Simulated Annealing

Δf: Deterioration of the objective value by the move from $\boldsymbol{x}$ to $\boldsymbol{y}$.

Acceptance probability of $\boldsymbol{y}$:

$$P(\boldsymbol{x} \to \boldsymbol{y}) = \begin{cases} 1, & \text{if } \boldsymbol{y} \text{ is better than } \boldsymbol{x} \text{ (i.e., if } \Delta f < 0) \\ \exp\left(\dfrac{-\Delta f}{T}\right), & \text{otherwise} \end{cases}$$



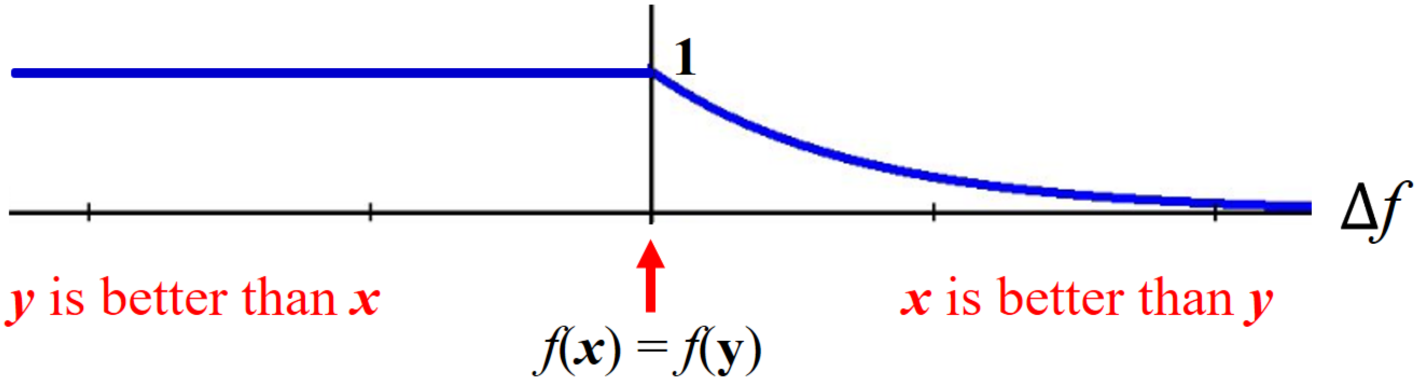
Simulated Annealing

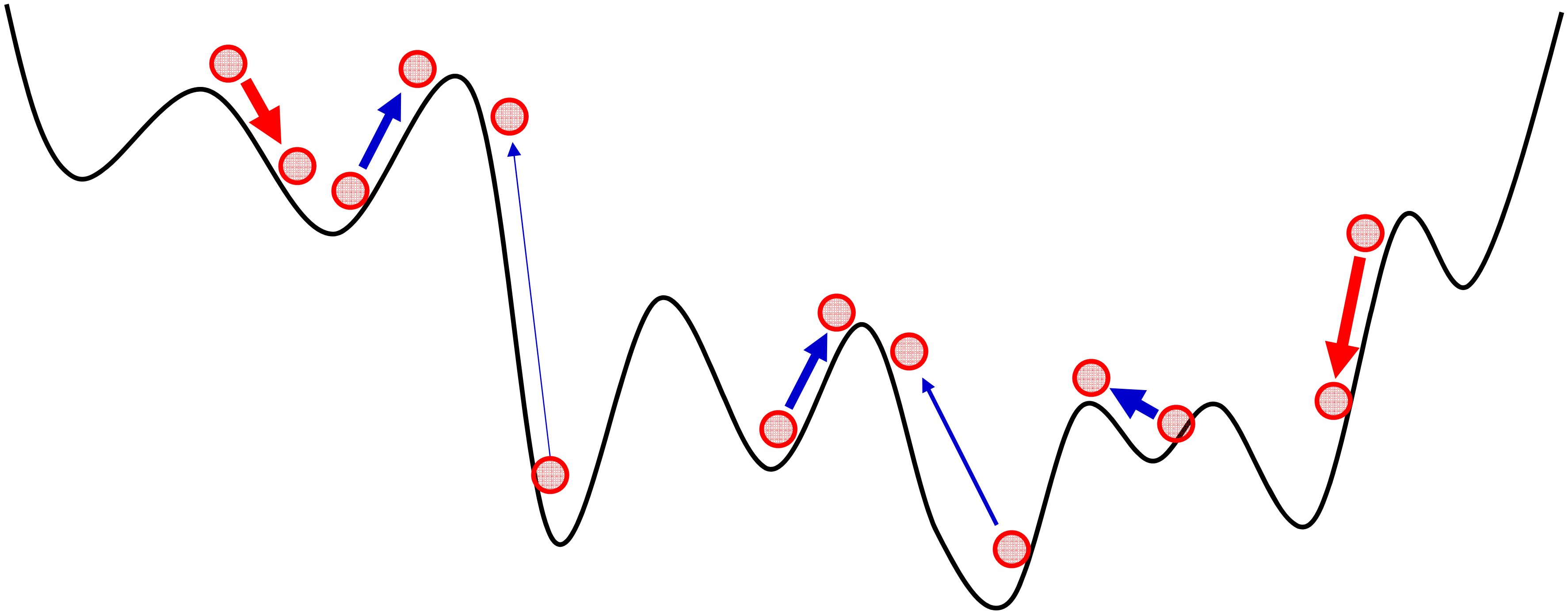
Δf : Deterioration of the objective value by the move from $\boldsymbol{x}$ to $\boldsymbol{y}$.

Acceptance probability of $\boldsymbol{y}$:

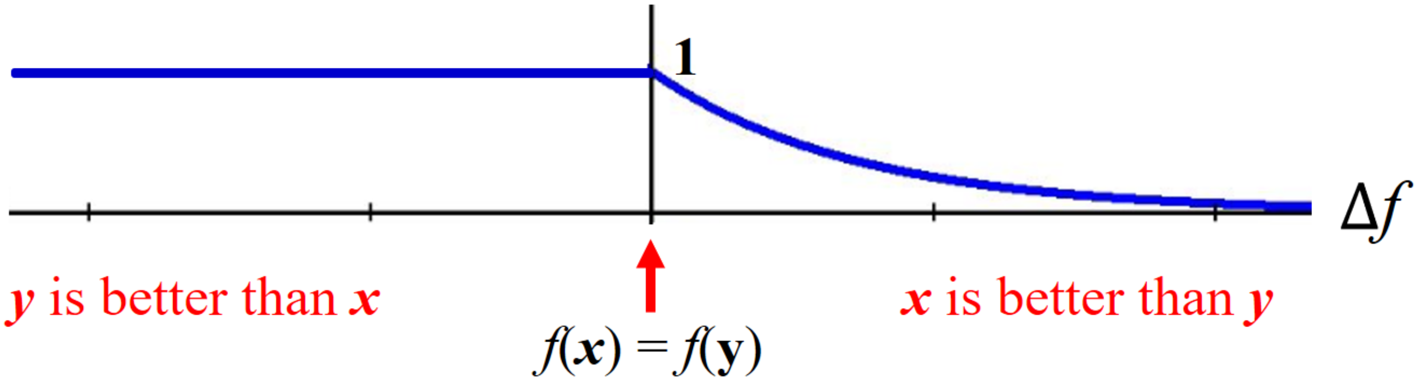
$$P(\boldsymbol{x} \rightarrow \boldsymbol{y}) = \begin{cases} 1, & \text{if } \boldsymbol{y} \text{ is better than } \boldsymbol{x} \text{ (i.e., if } \Delta f < 0) \\ \exp\left(\dfrac{-\Delta f}{T}\right), & \text{otherwise} \end{cases}$$

- A **better solution** (or same quality solution) is always accepted.
- A **solution with a smaller deterioration** has a higher acceptance probability than a solution with a larger deterioration.





- A **better solution** (or same quality solution) is always accepted.
- A **solution with a smaller deterioration** has a higher acceptance probability than a solution with a larger deterioration.



Simulated Annealing

Δf: Deterioration of the objective value by the move from $\boldsymbol{x}$ to $\boldsymbol{y}$.

Acceptance probability of $\boldsymbol{y}$:

$$P(\boldsymbol{x} \to \boldsymbol{y}) = \begin{cases} 1, & \text{if } \boldsymbol{y} \text{ is better than } \boldsymbol{x} \text{ (i.e., if } \Delta f < 0) \\ \exp\left(\dfrac{-\Delta f}{T}\right), & \text{otherwise} \end{cases}$$

- A **better solution** (or same quality solution) is always accepted.
- A **solution with a smaller deterioration** has a higher acceptance probability than a solution with a larger deterioration.

Acceptance probability of $\boldsymbol{y}$:

Δf: Deterioration of the objective value by the move from $\boldsymbol{x}$ to $\boldsymbol{y}$.

$$P(\boldsymbol{x} \rightarrow \boldsymbol{y}) = \begin{cases} 1, & \text{if } \boldsymbol{y} \text{ is better than } \boldsymbol{x} \text{ (i.e., if } \Delta f < 0) \\ \exp\left(\dfrac{-\Delta f}{T}\right), & \text{otherwise} \end{cases}$$

- A **better solution** (or same quality solution) is always accepted.
- A **solution with a smaller deterioration** has a higher acceptance probability than a solution with a larger deterioration.
- **Explain the search behavior (i.e., the move of the current solution) when *T* is very large:_______(Your answer)___________.**

Acceptance probability of $\boldsymbol{y}$:

Δf : Deterioration of the objective value by the move from $\boldsymbol{x}$ to $\boldsymbol{y}$.

$$P(\boldsymbol{x} \rightarrow \boldsymbol{y}) = \begin{cases} 1, & \text{if } \boldsymbol{y} \text{ is better than } \boldsymbol{x} \text{ (i.e., if } \Delta f < 0) \\ \exp\left(\dfrac{-\Delta f}{T}\right), & \text{otherwise} \end{cases}$$

- A **better solution** (or same quality solution) is always accepted.
- A **solution with a smaller deterioration** has a higher acceptance probability than a solution with a larger deterioration.
- **Explain the search behavior (i.e., the move of the current solution) when *T* is very large: Similar to random search**.
- **Explain the search behavior (i.e., the move of the current solution) when *T* is very small:________(Your answer)_______________.**

Acceptance probability of $\boldsymbol{y}$:

Δf: Deterioration of the objective value by the move from $\boldsymbol{x}$ to $\boldsymbol{y}$.

$$P(\boldsymbol{x} \to \boldsymbol{y}) = \begin{cases} 1, & \text{if } \boldsymbol{y} \text{ is better than } \boldsymbol{x} \text{ (i.e., if } \Delta f < 0) \\ \exp\left(\dfrac{-\Delta f}{T}\right), & \text{otherwise} \end{cases}$$

- A **better solution** (or same quality solution) is always accepted.
- A **solution with a smaller deterioration** has a higher acceptance probability than a solution with a larger deterioration.
- **Explain the search behavior (i.e., the move of the current solution) when T is very large: Similar to random search.**
- **Explain the search behavior (i.e., the move of the current solution) when T is very small: Similar to first move local search.**
- **Explain the search behavior (i.e., the move of the current solution) when T is medium:_________(Your answer)___________.**

Acceptance probability of $\boldsymbol{y}$:

Δf: Deterioration of the objective value by the move from $\boldsymbol{x}$ to $\boldsymbol{y}$.

$$P(\boldsymbol{x} \rightarrow \boldsymbol{y}) = \begin{cases} 1, & \text{if } \boldsymbol{y} \text{ is better than } \boldsymbol{x} \text{ (i.e., if } \Delta f < 0) \\ \exp\left(\dfrac{-\Delta f}{T}\right), & \text{otherwise} \end{cases}$$

- A **better solution** (or same quality solution) is always accepted.
- A **solution with a smaller deterioration** has a higher acceptance probability than a solution with a larger deterioration.
- **Explain the search behavior (i.e., the move of the current solution) when *T* is very large: Similar to random search.**
- **Explain the search behavior (i.e., the move of the current solution) when *T* is very small: Similar to first move local search.**
- **Explain the search behavior (i.e., the move of the current solution) when *T* is medium: Probabilistic search**.

Simulated Annealing

Δf : Deterioration of the objective value by the move from $\boldsymbol{x}$ to $\boldsymbol{y}$.

Acceptance probability of $\boldsymbol{y}$:

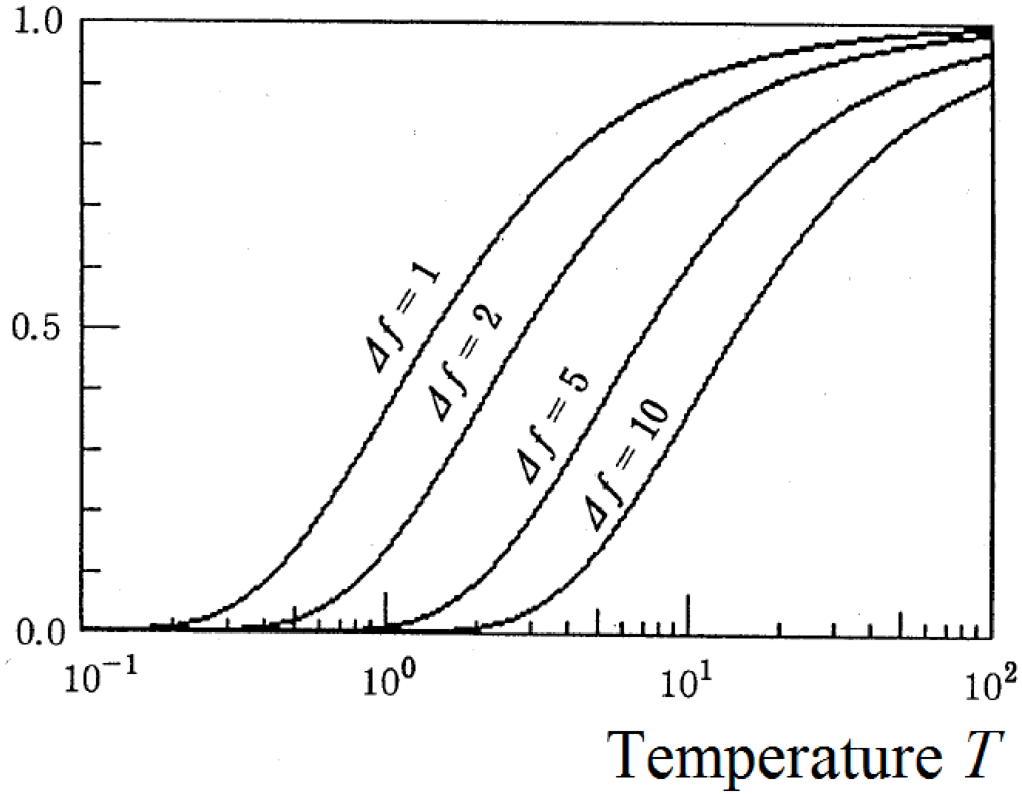
$$P(\boldsymbol{x} \rightarrow \boldsymbol{y}) = \begin{cases} 1, & \text{if } \boldsymbol{y} \text{ is better than } \boldsymbol{x} \text{ (i.e., if } \Delta f < 0) \\ \exp\left(\dfrac{-\Delta f}{T}\right), & \text{otherwise} \end{cases}$$

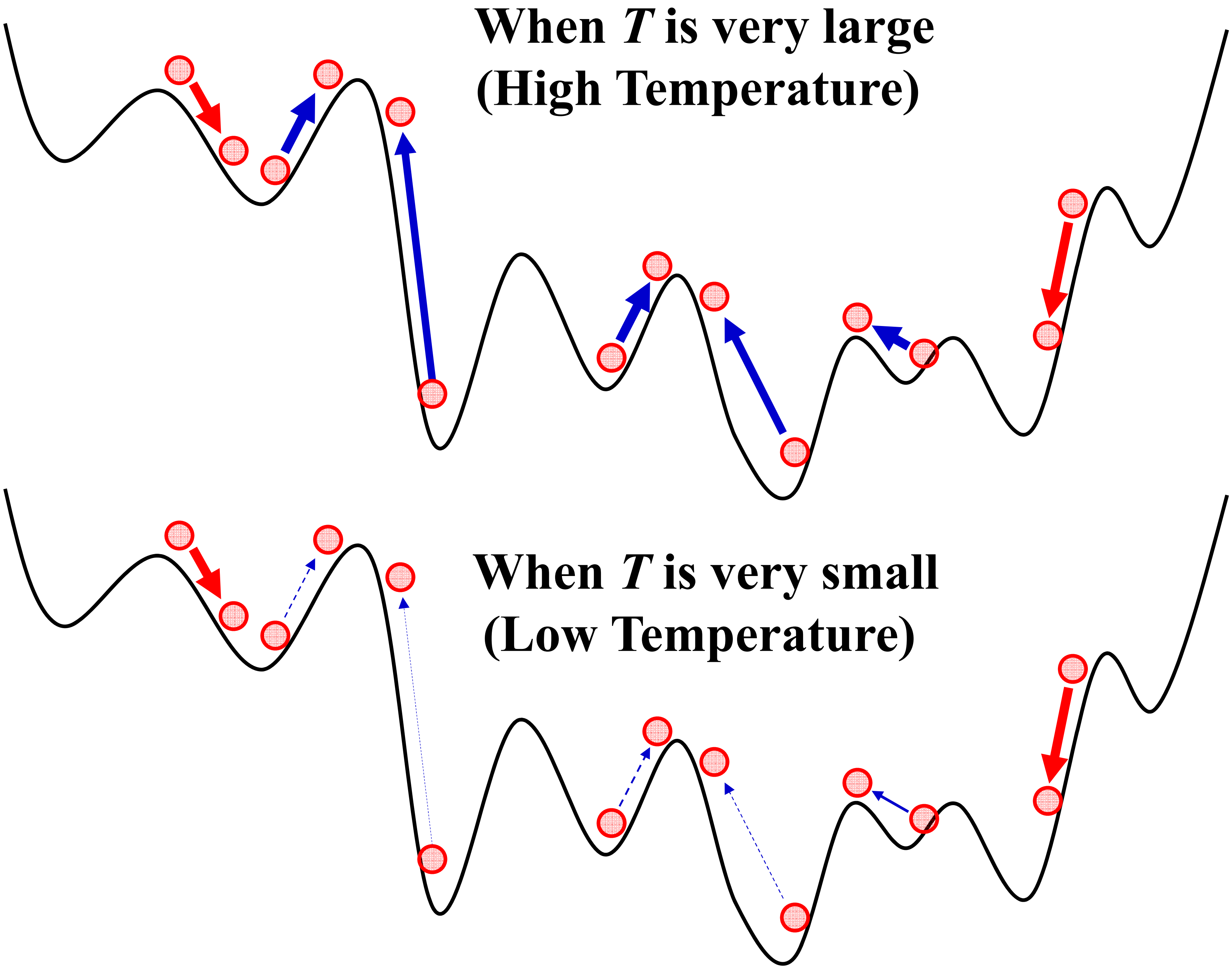
- A **better solution** (or same quality solution) is always accepted.
- A **solution with a smaller deterioration** has a higher acceptance probability than a solution with a larger deterioration.
- **When *T* is very large**, even a solution with a large deterioration has a large acceptance probability (easy to move to a worse solution).
- **When *T* is very small**, even a solution with a small deterioration has a small acceptance probability (difficult to move to a worse solution).

Simulated Annealing

Δf : Deterioration of the objective value by the move from $\boldsymbol{x}$ to $\boldsymbol{y}$.

- **When *T* is very large**, even a solution with a large deterioration has a large acceptance probability (easy to move to a worse solution).
- **When *T* is very small**, even a solution with a small deterioration has a small acceptance probability (difficult to move to a worse solution).





When *T* is very large
(High Temperature)
When *T* is very small
(Low Temperature)

Simulated Annealing

$$P(\boldsymbol{x} \to \boldsymbol{y}) = \min\left\{1,\ \exp\left(\frac{-\Delta f}{T}\right)\right\}$$

where T is a control parameter called the temperature.

Theory: When T is gradually decreased from a large value to zero under some conditions, the search converges to the optimal solution **with the probability 1** after an infinite number of iterations (i.e., the final solution is always the optimal solution with the probability 1 after an infinite number of iterations).

$$T(t) \geq \frac{T_0}{\ln(1+t)}, \qquad t = 1, 2, \ldots$$

Simulated Annealing

$$P(\boldsymbol{x} \rightarrow \boldsymbol{y}) = \min\left\{1, \exp\left(\frac{-\Delta f}{T}\right)\right\}$$

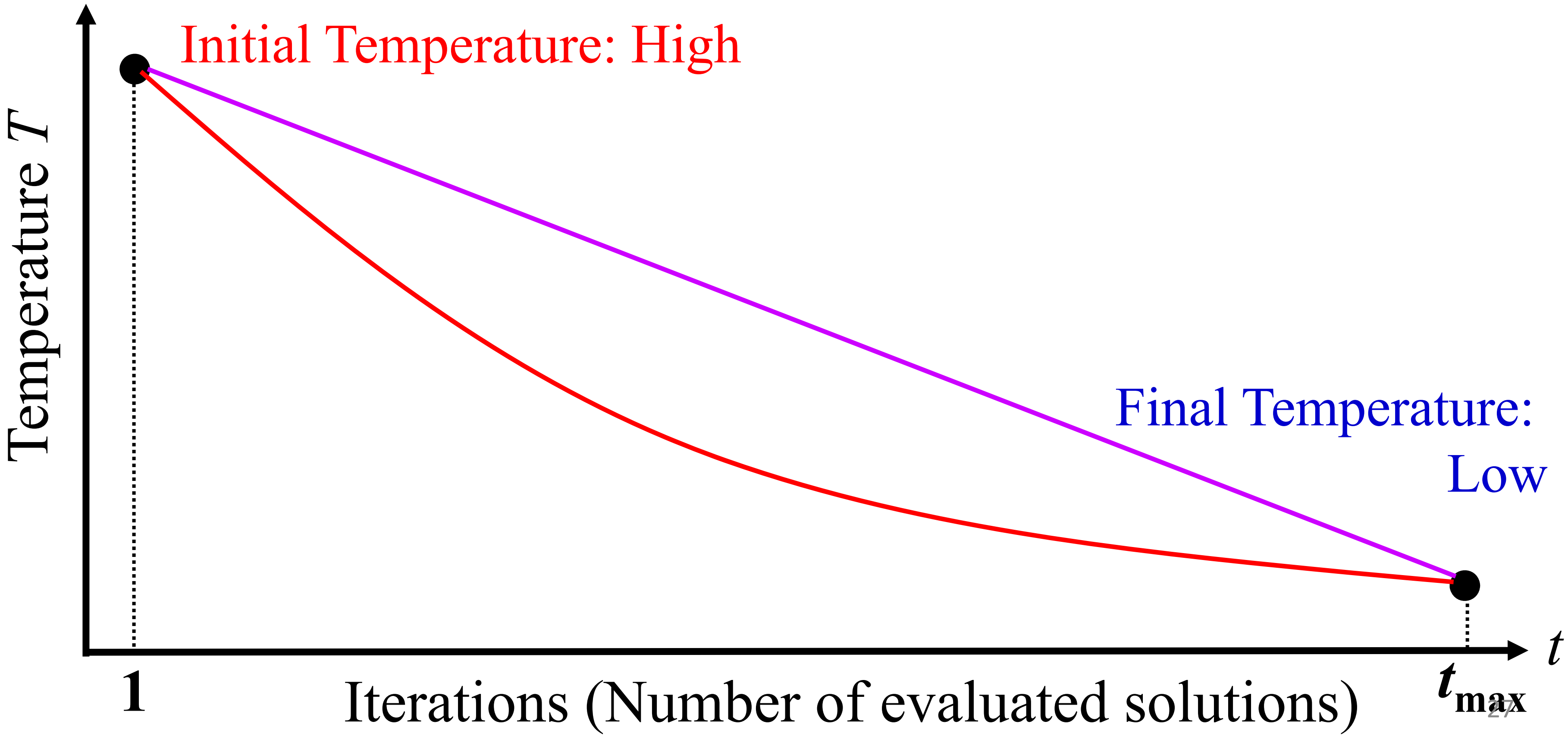
where T is a control parameter called the temperature.

Theory: When T is gradually decreased from a large value to zero under some conditions, the search converges to the optimal solution **with the probability 1** after an infinite number of iterations (i.e., the final solution is always the optimal solution with the probability 1 after an infinite number of iterations).

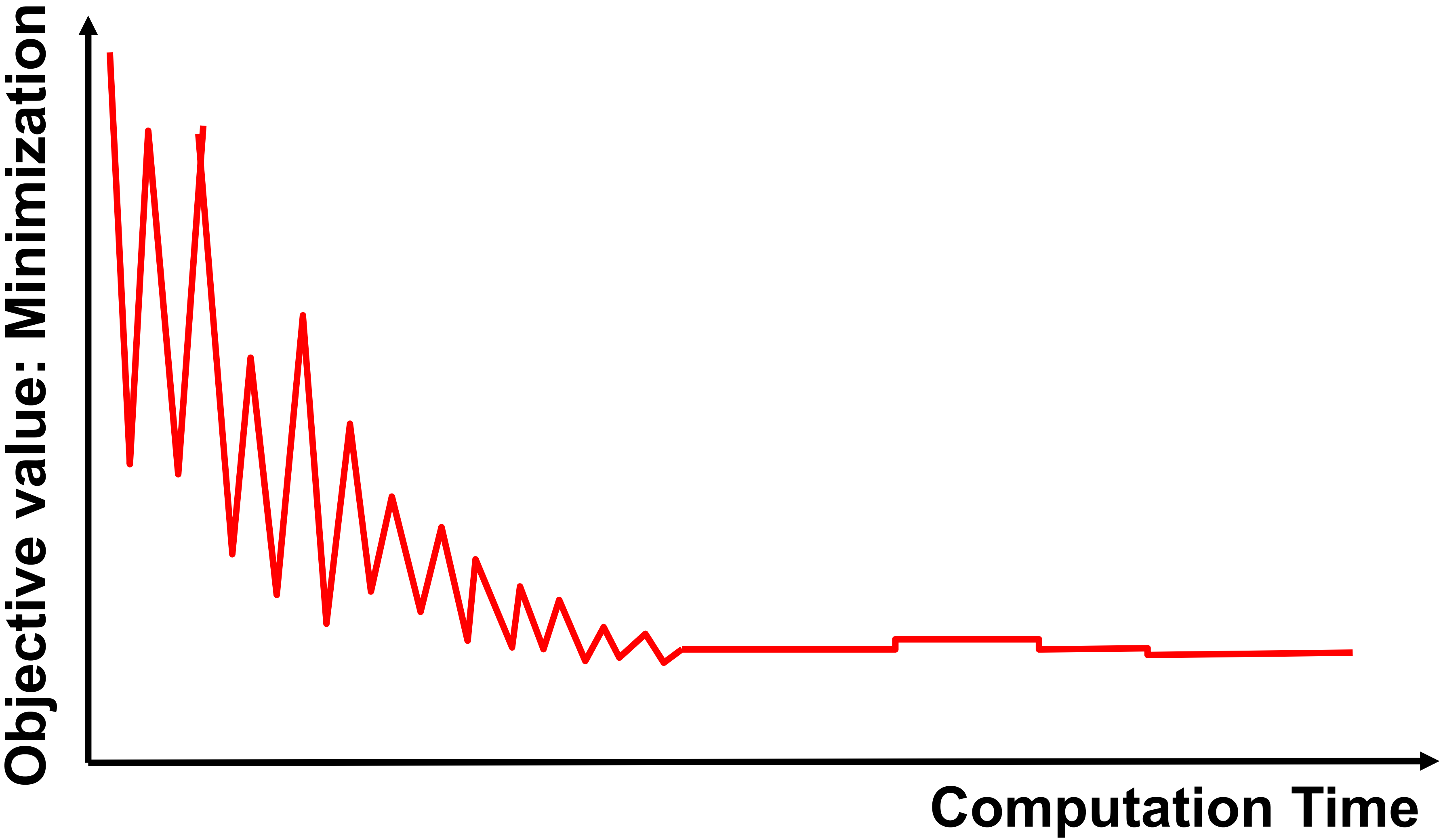
The theory is beautiful but unrealistic. We need a realistic cooling schedule of *T* and a realistic termination criterion.

==> Approximation Algorithm

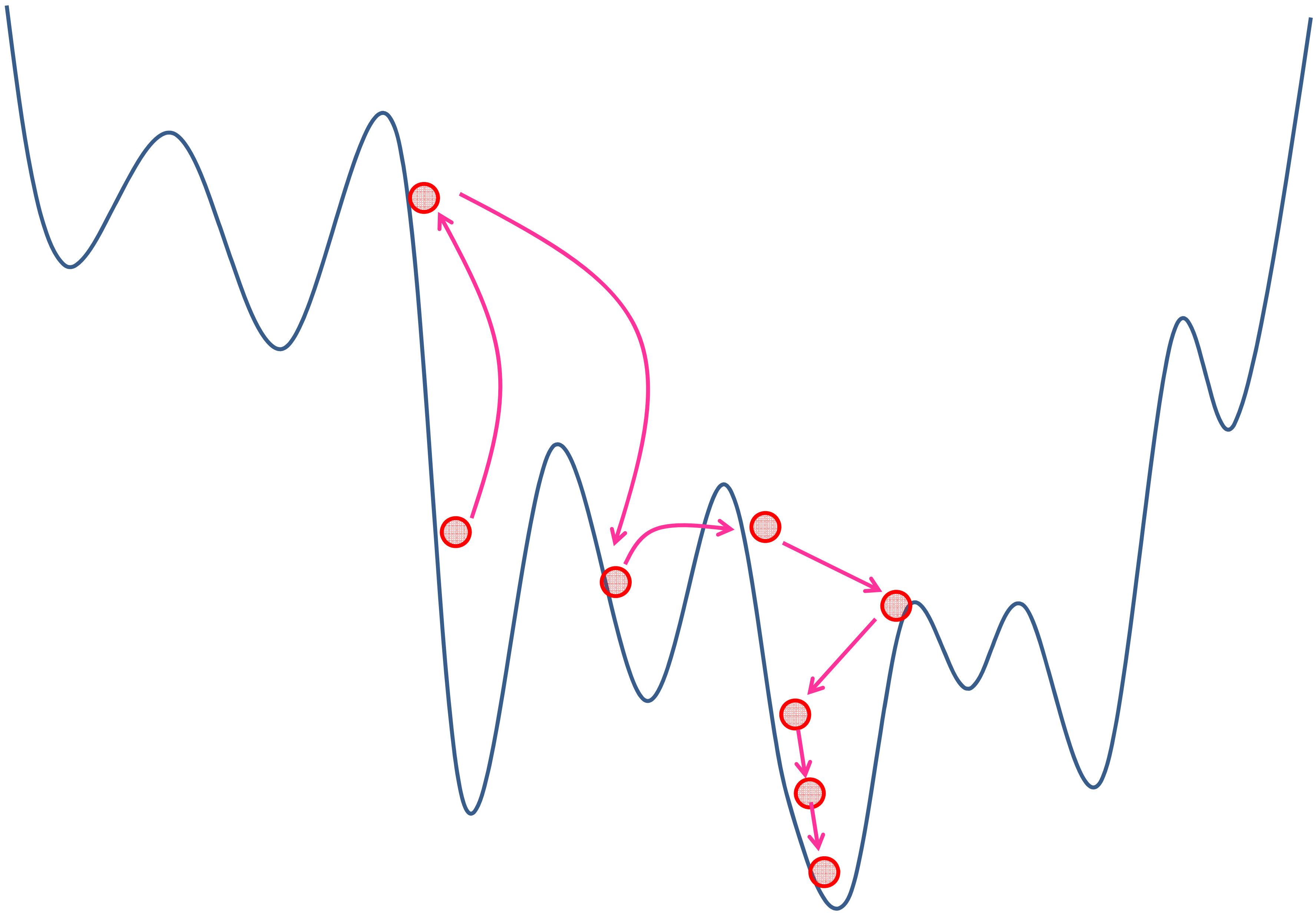
Cooling Schedule



Explanation of Search by Simulated Annealing

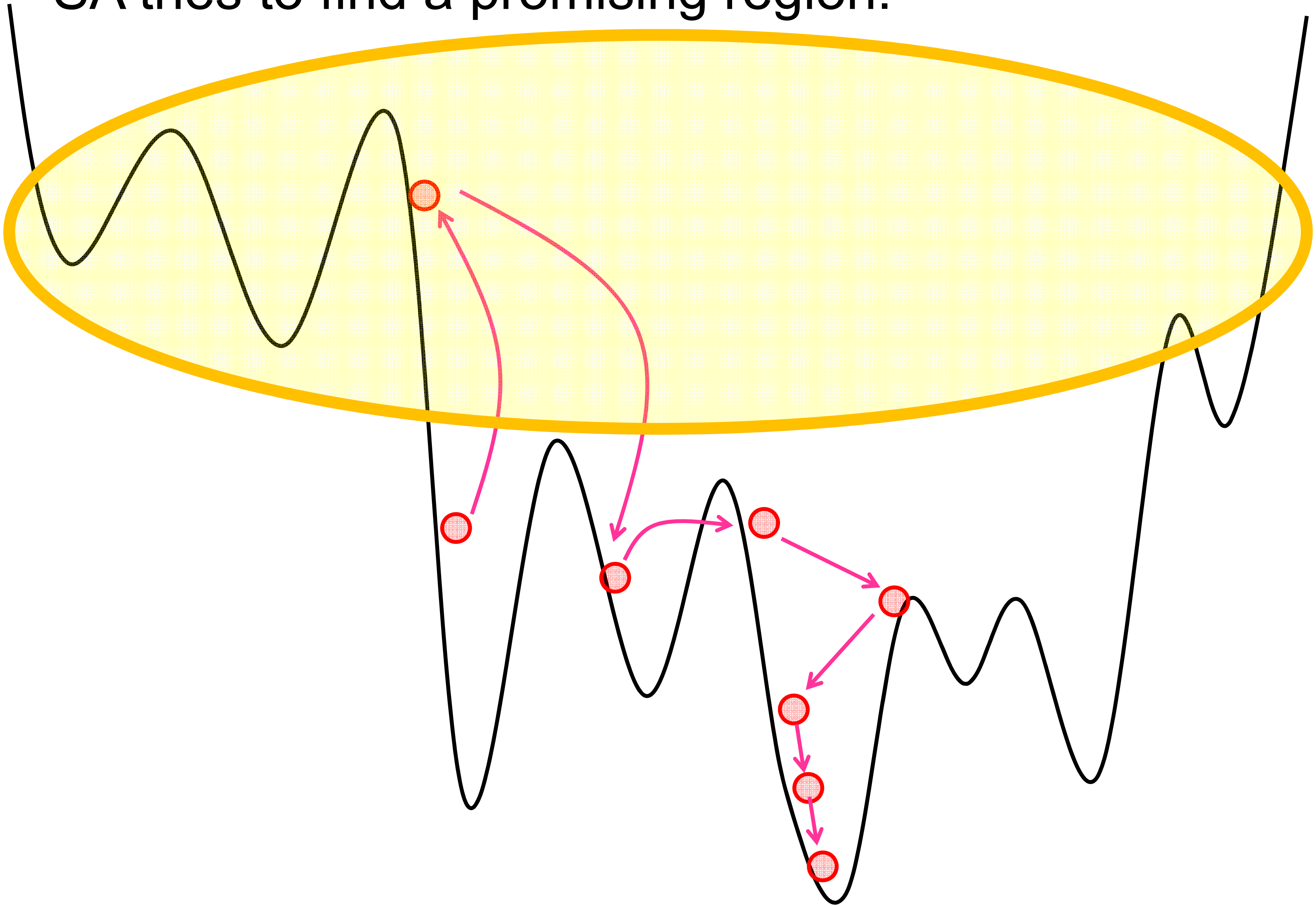


Explanation of Search by Simulated Annealing



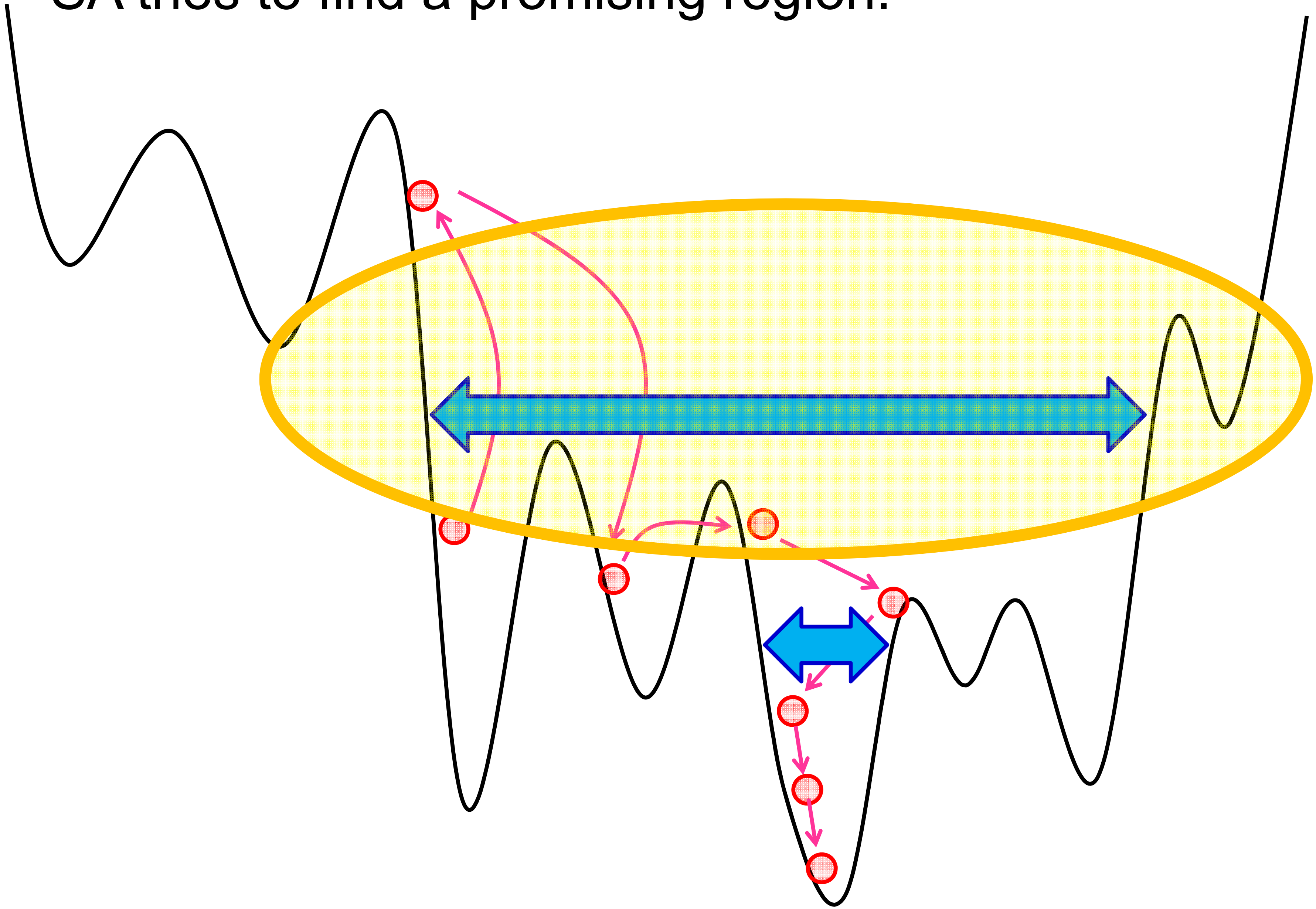
In the early stage of search with high temperature

SA tries to find a promising region.



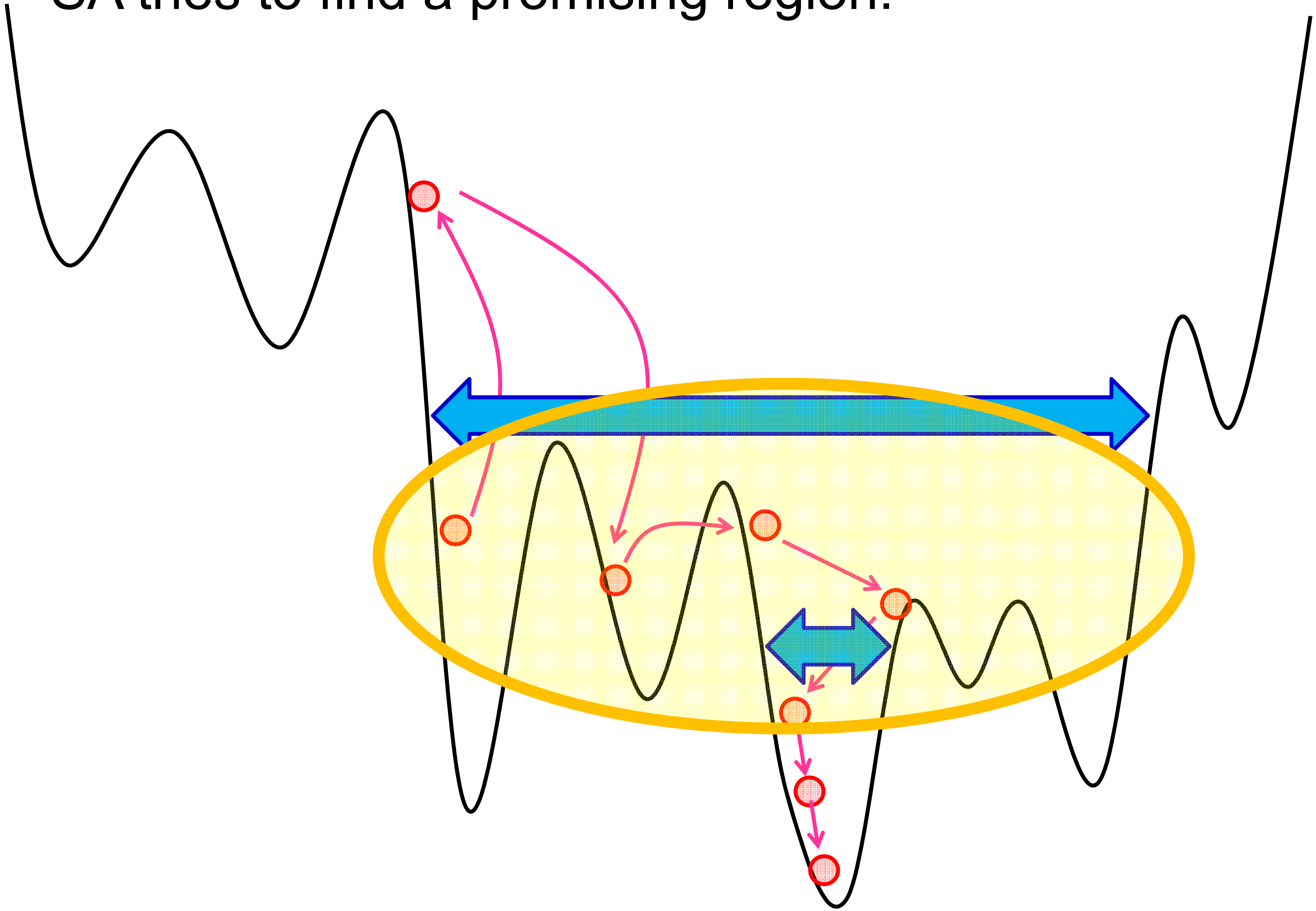
In the early stage of search with high temperature

SA tries to find a promising region.



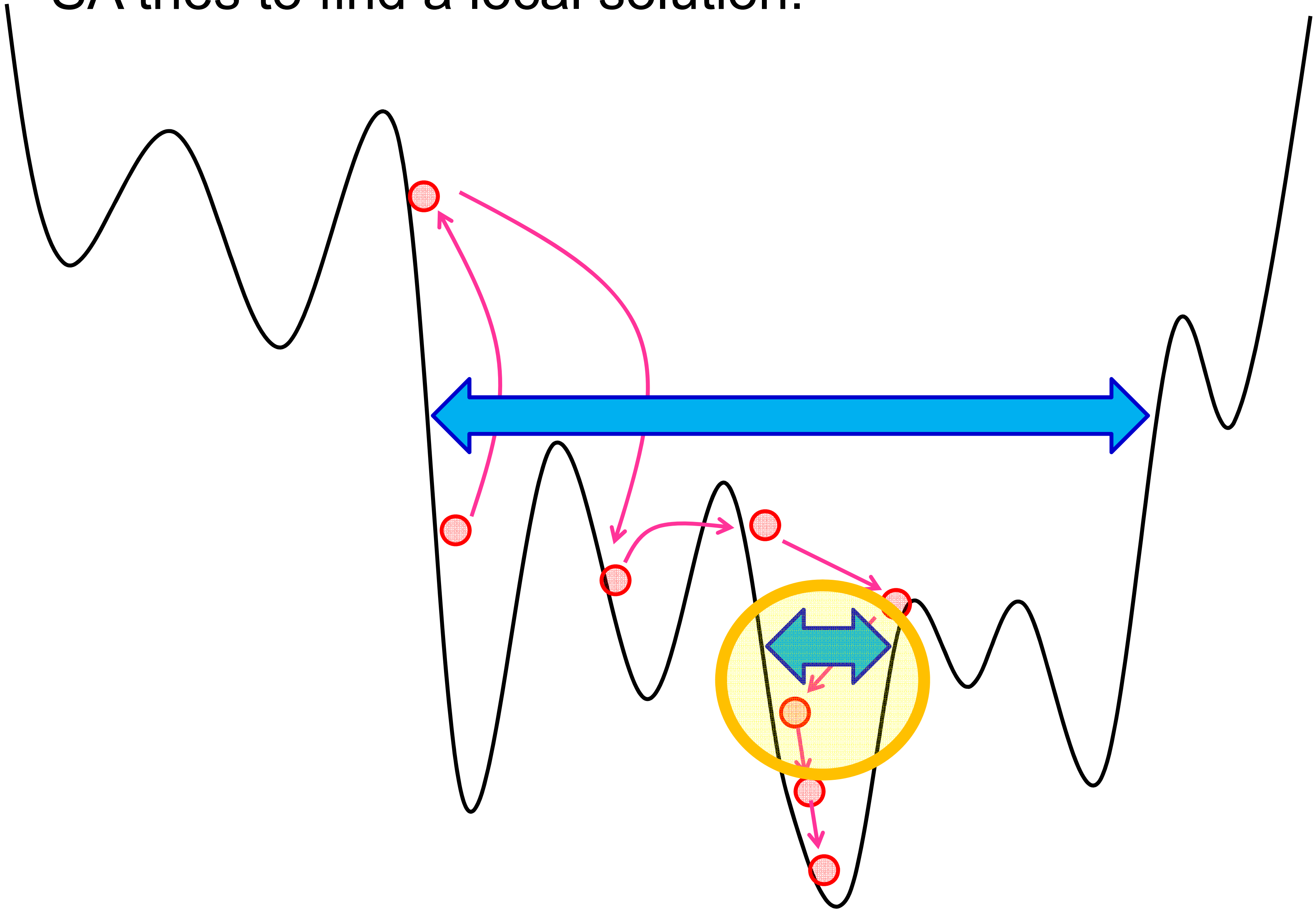
In the early stage of search with high temperature

SA tries to find a promising region.



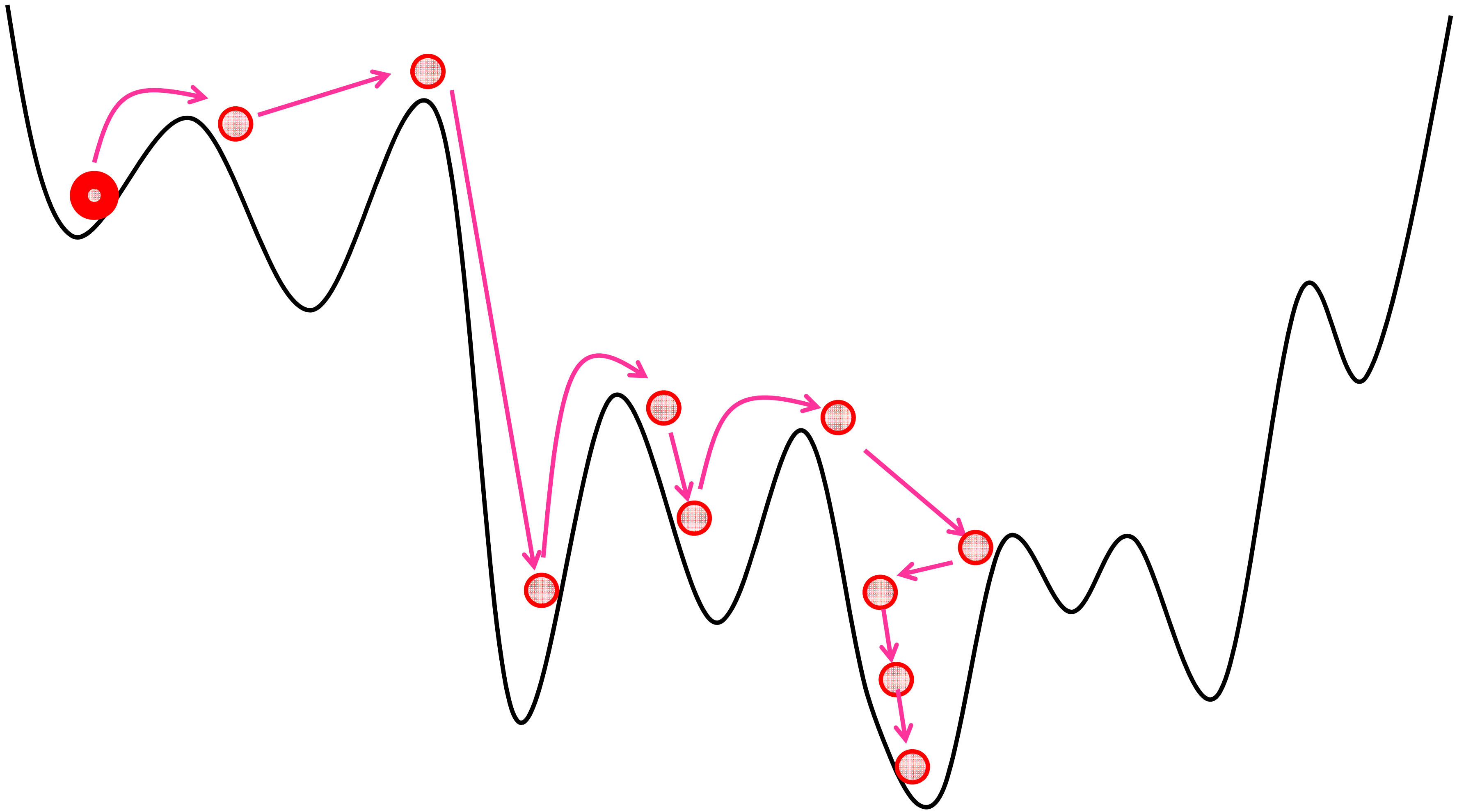
In the final stage of search with low temperature

SA tries to find a local solution.



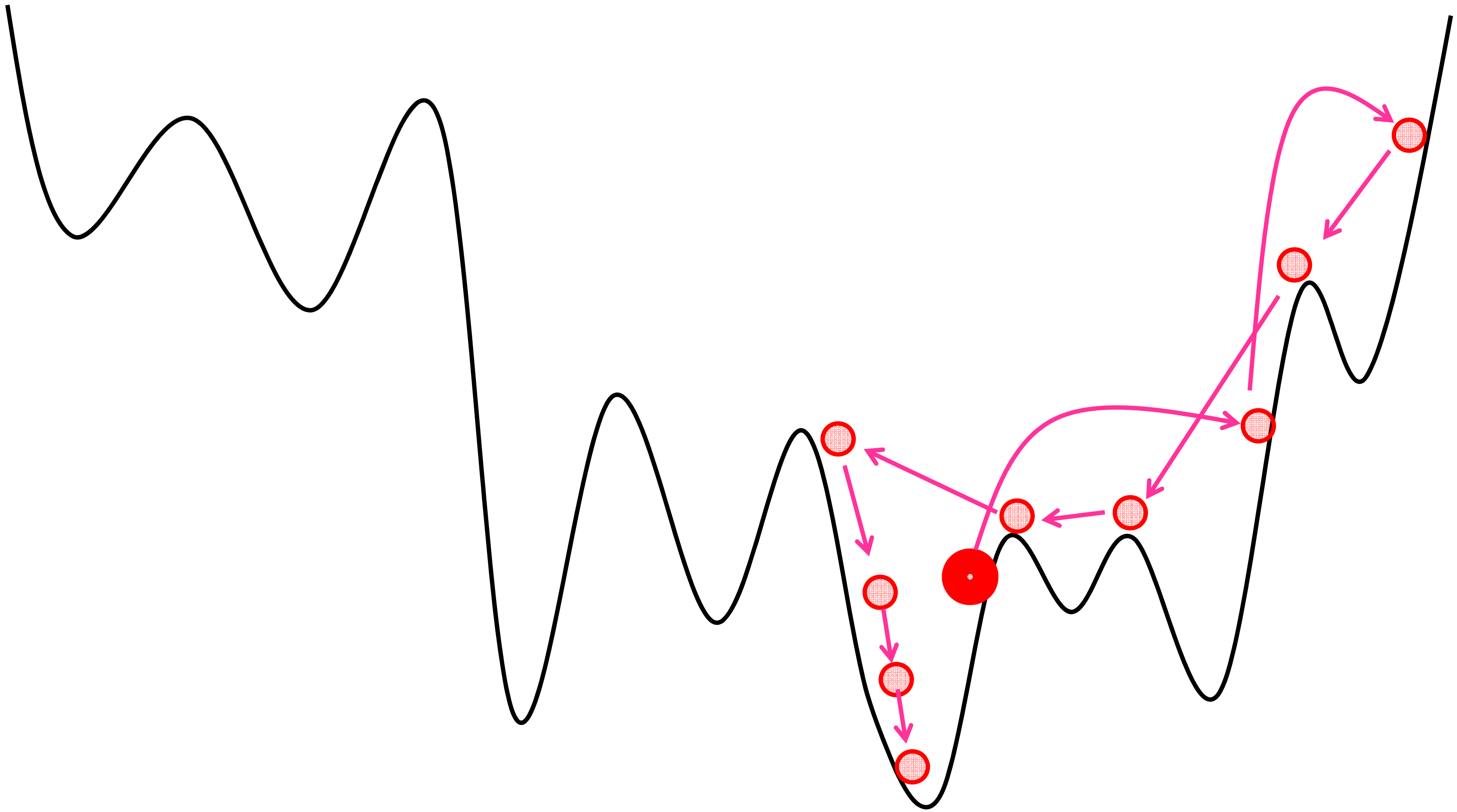
Simulated Annealing

- Choice of an initial solution is not important.



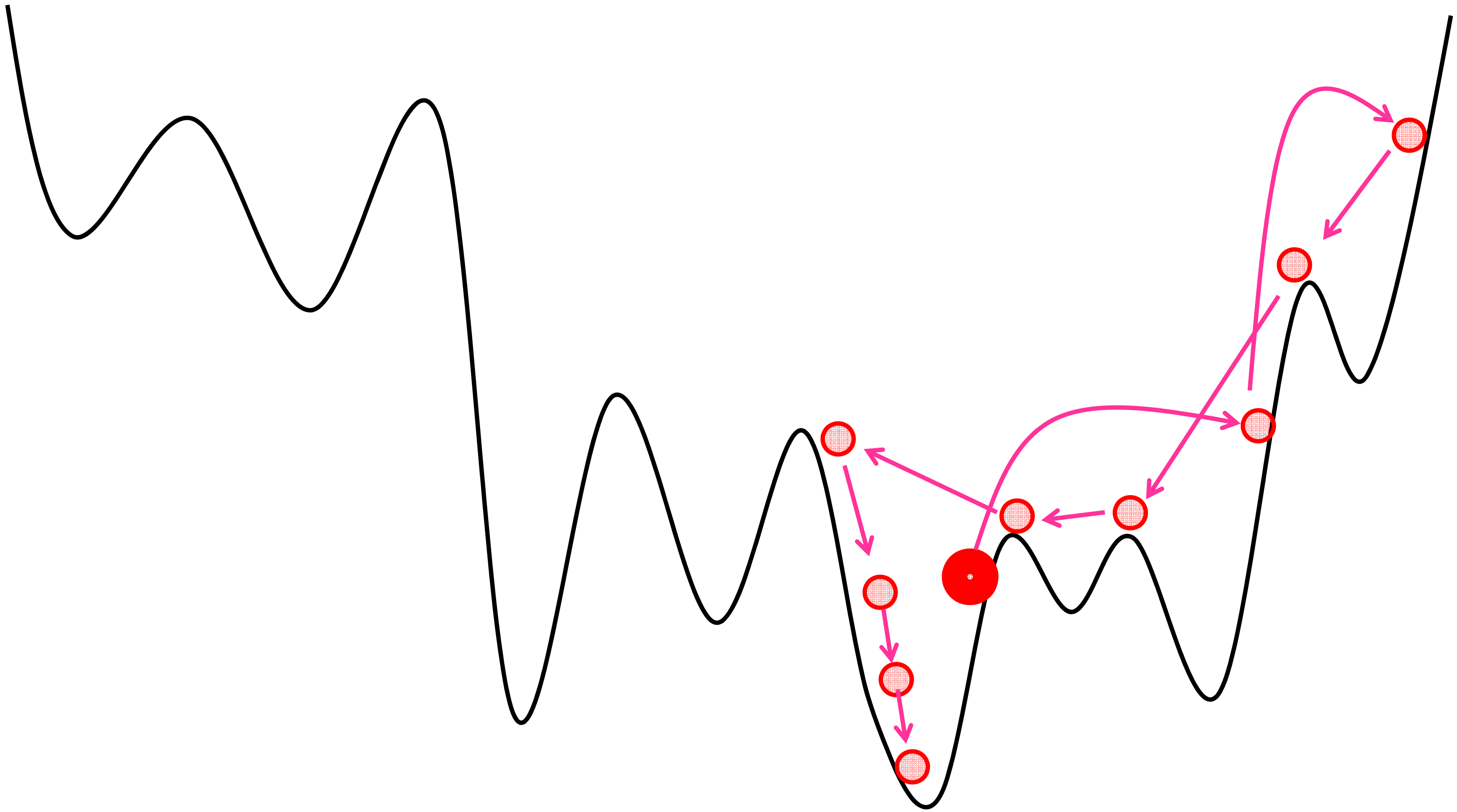
Simulated Annealing

- Choice of an initial solution is not important.



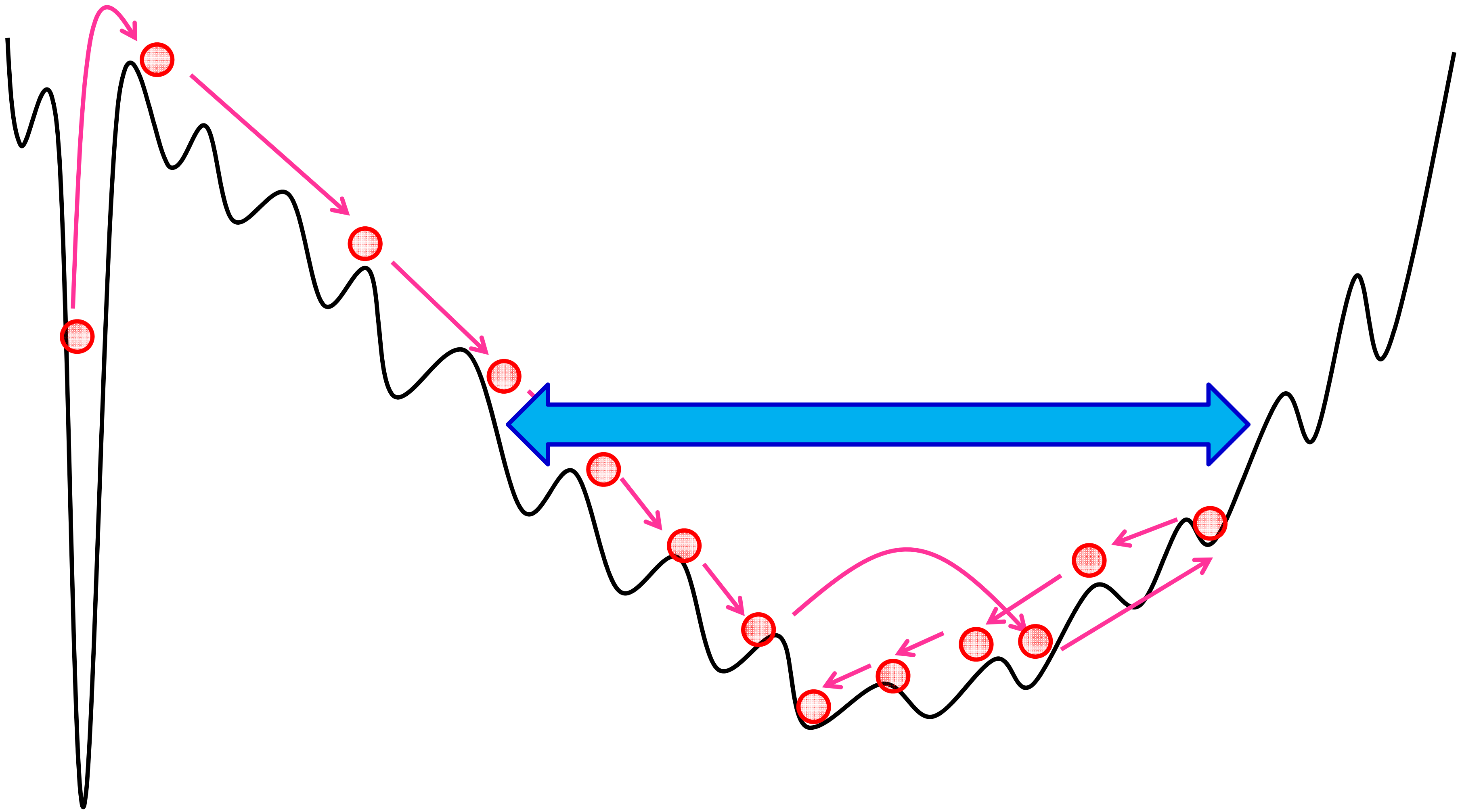
Simulated Annealing

- Choice of an initial solution is not important.
- It is difficult to efficiently utilize a good initial solution.



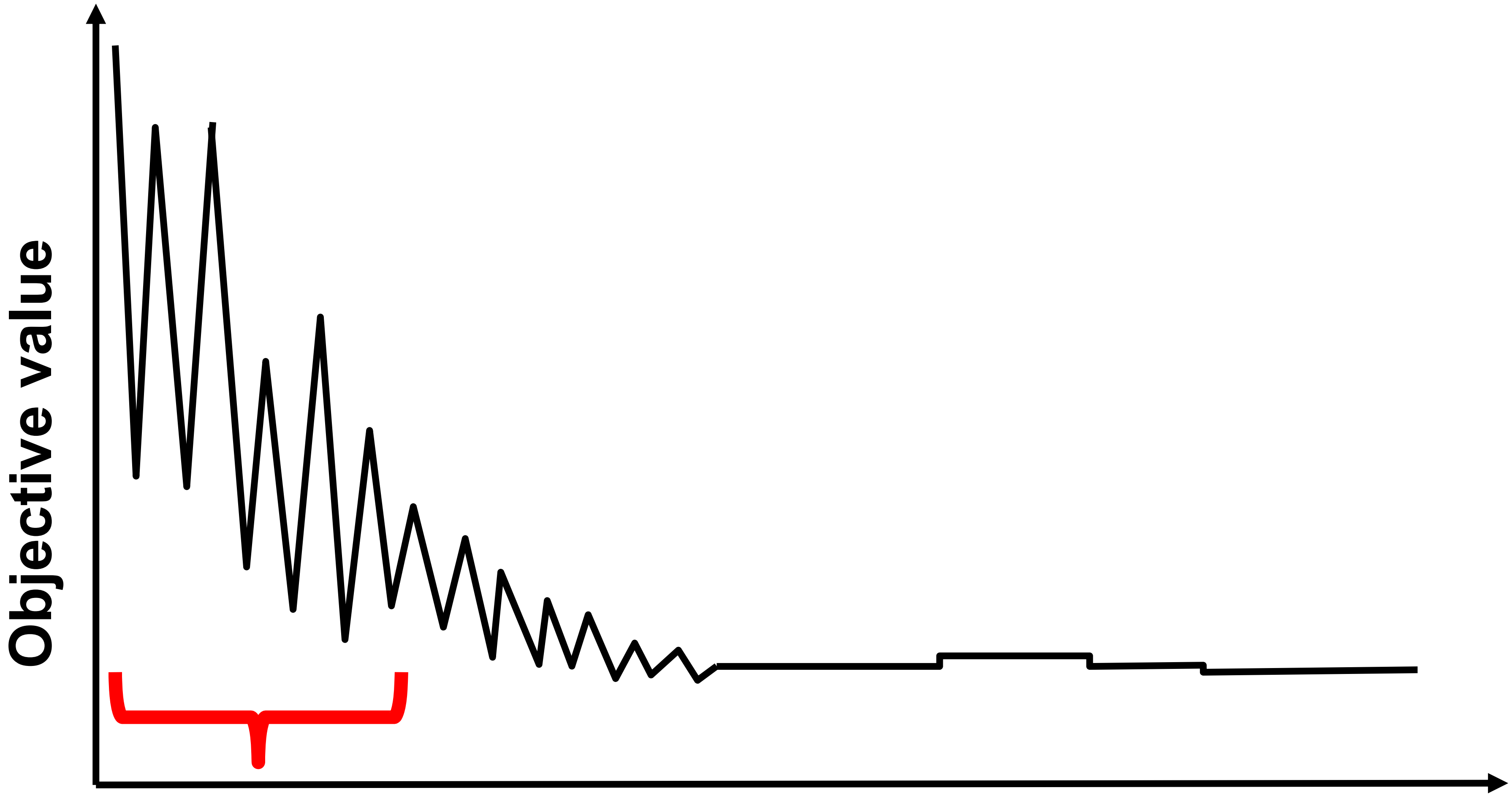
Simulated Annealing

- Choice of an initial solution is not important.
- It is difficult to efficiently utilize a good initial solution.

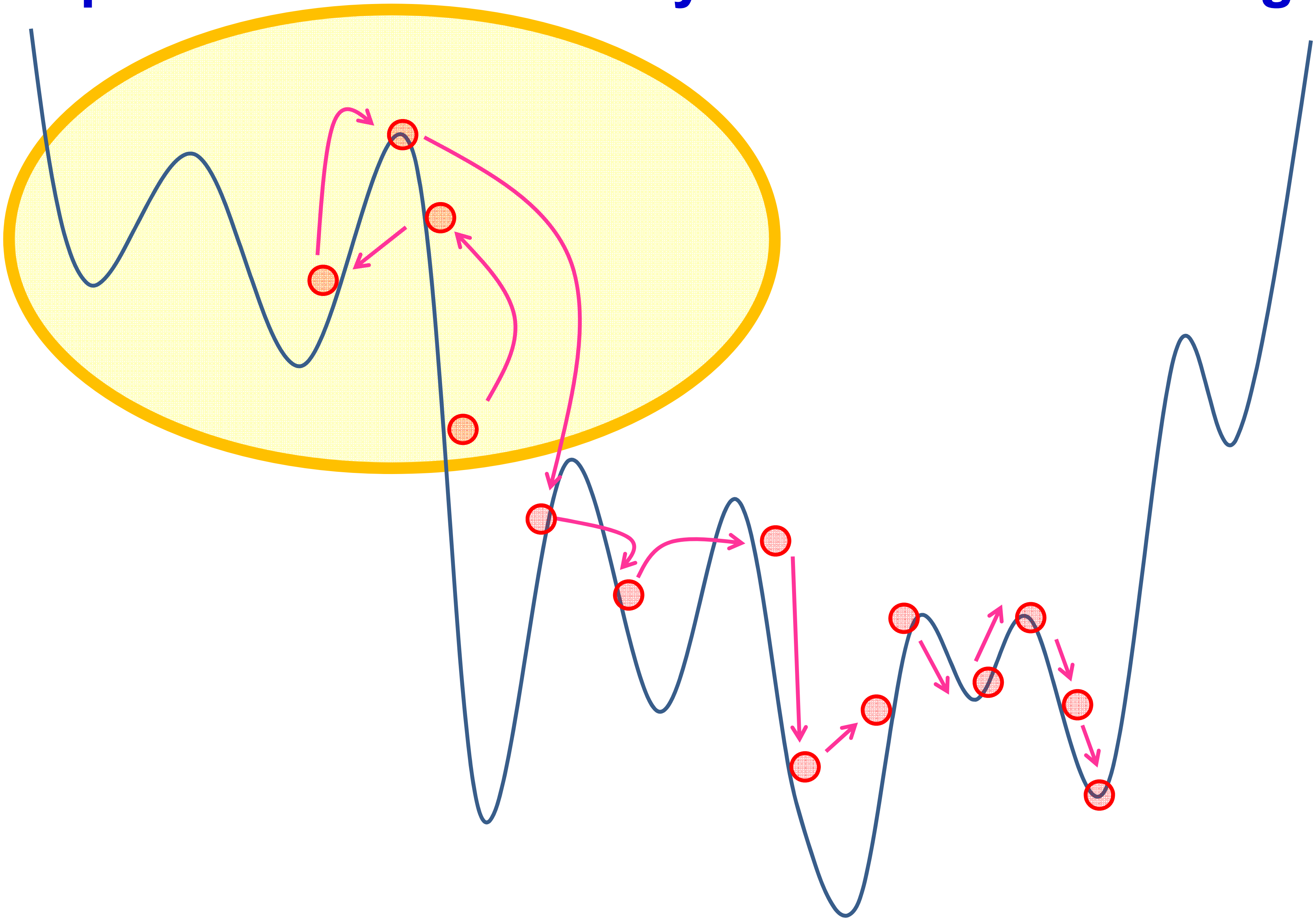


Simulated Annealing

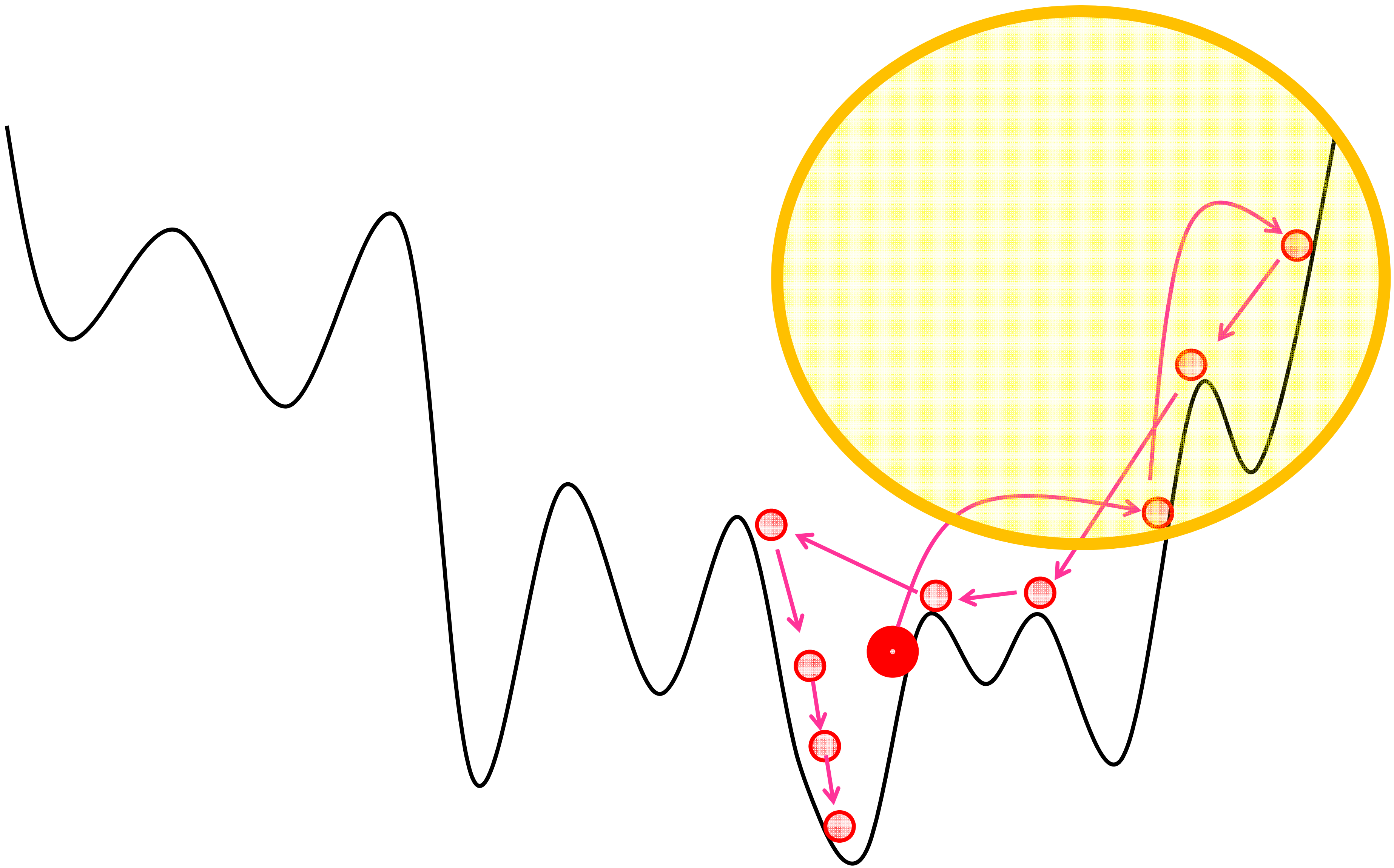
- Choice of an initial solution is not important.
- It is difficult to efficiently utilize a good initial solution.
- Some initial steps look useless (since they are almost random)



Explanation of Search by Simulated Annealing



Explanation of Search by Simulated Annealing

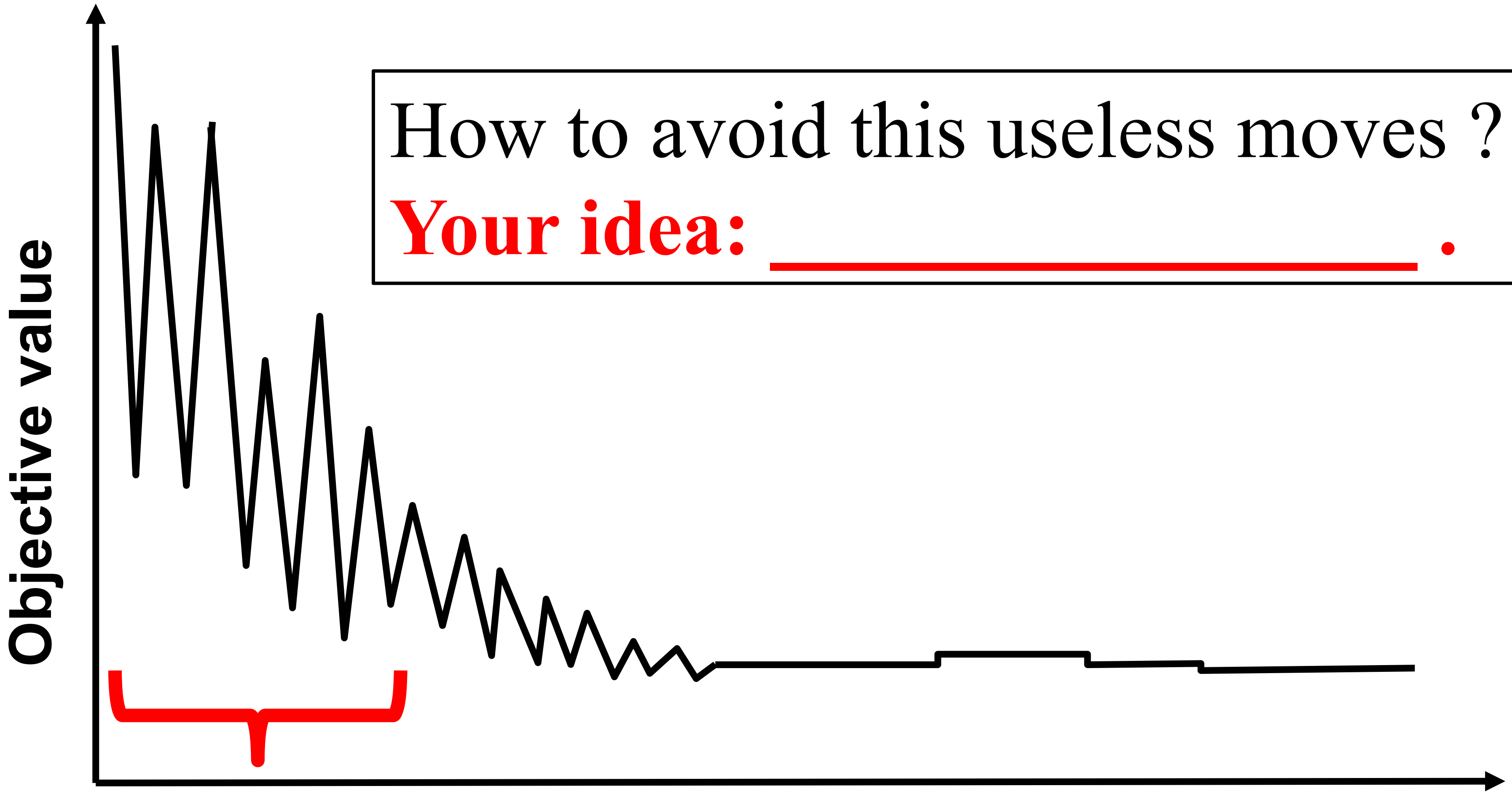


Simulated Annealing

- Choice of an initial solution is not important.
- It is difficult to efficiently utilize a good initial solution.
- Some initial steps look useless (since they are almost random)

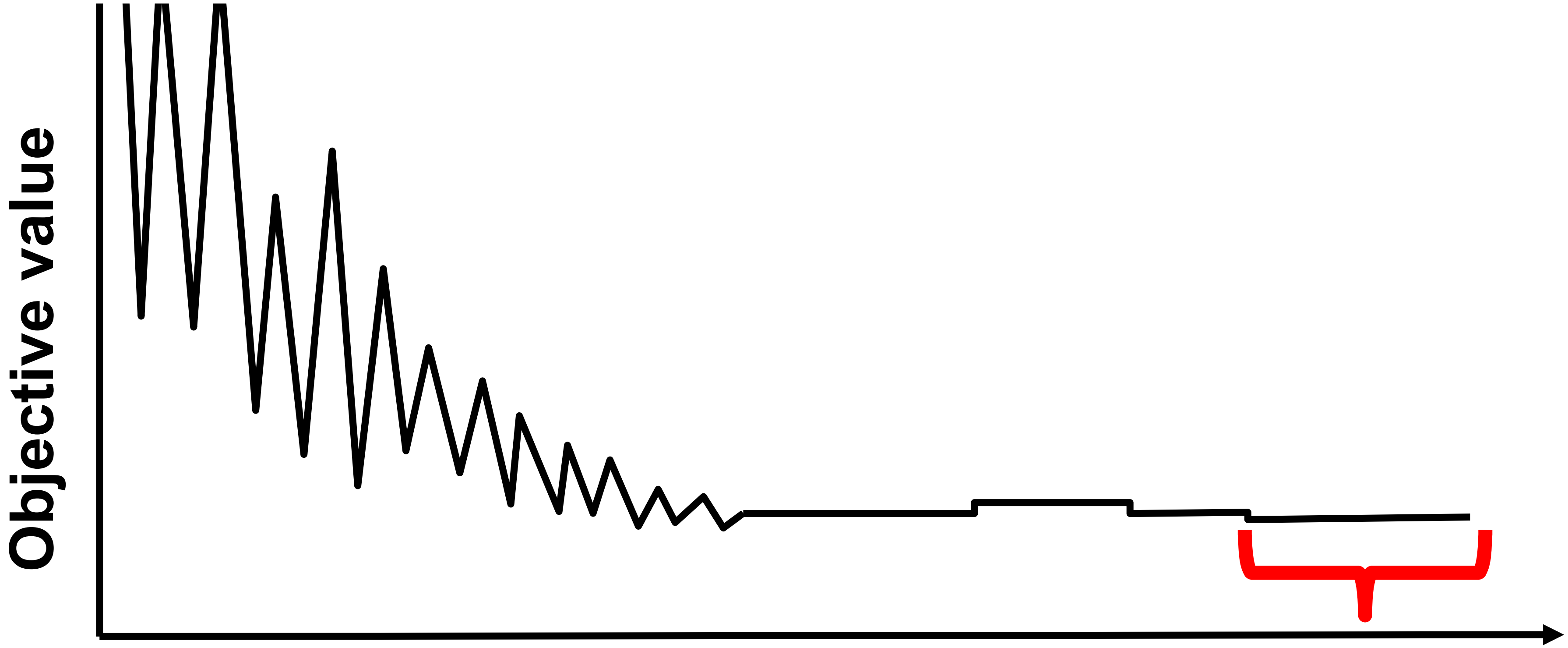
How to avoid this useless moves ?

Your idea: ______________________ .

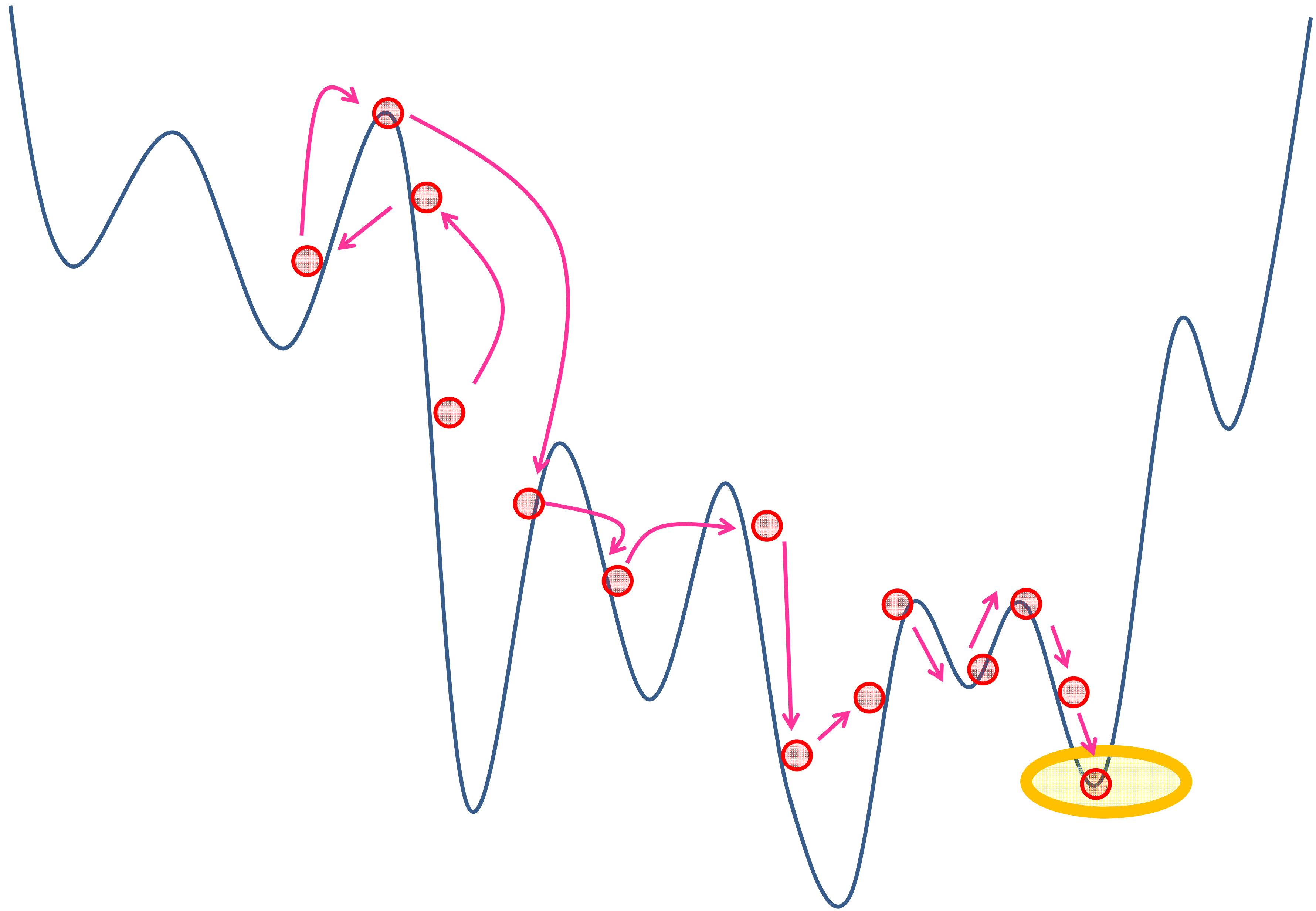


Simulated Annealing

- Choice of an initial solution is not important.
- It is difficult to efficiently utilize a good initial solution.
- Some initial steps look useless (since they are almost random)
- Many final steps look useless (since they cannot move from a local solution)

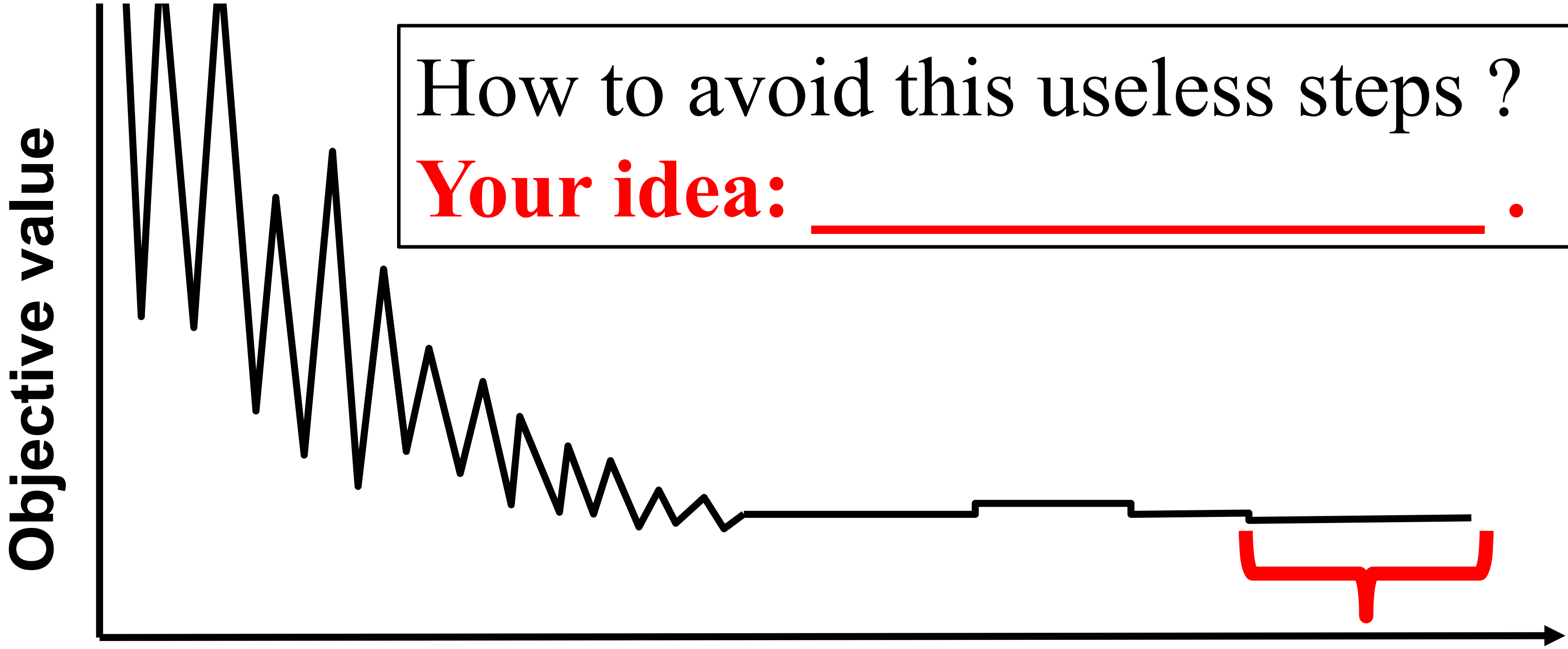


Explanation of Search by Simulated Annealing



Simulated Annealing

- Choice of an initial solution is not important.
- It is difficult to efficiently utilize a good initial solution.
- Some initial steps look useless (since they are almost random)
- Many final steps look useless (since they cannot move from a local solution)



How to avoid this useless steps ?

Your idea: ____________________ .

Simulated Annealing

- Choice of an initial solution is not important.
- It is difficult to efficiently utilize a good initial solution.
- Some initial steps look useless (since they are almost random)
- Many final steps look useless (since they cannot move from a local solution)

The Main Implementation Issue: Cooling Schedule

$$T = T(t),\ t = 1, 2, \ldots, t_{\max} \quad (t\text{: iteration index})$$

Initial value $T(1)$: Almost all moves are accepted.

Final value $T(t_{\max})$: No deterioration is accepted.

Simulated Annealing

- Choice of an initial solution is not important.
- It is difficult to efficiently utilize a good initial solution.
- Some initial steps look useless (since they are almost random)
- Many final steps look useless (since they cannot move from a local solution)

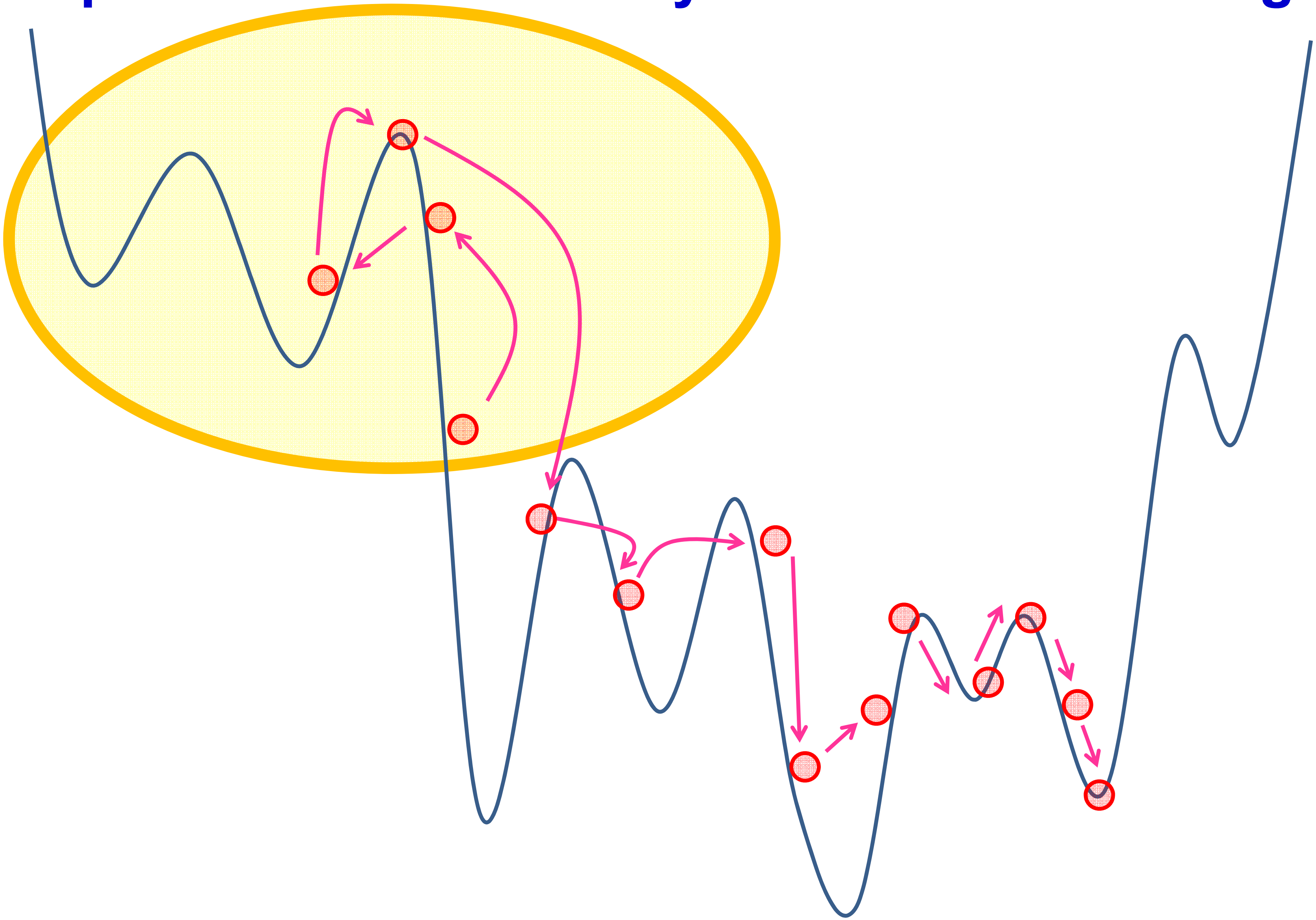
The Main Implementation Issue: Cooling Schedule

$$T = T(t),\ t = 1, 2, \ldots, t_{\max} \quad (t\text{: iteration index})$$

Initial value $T(1)$: Almost all moves are accepted.

Final value $T(t_{\max})$: No deterioration is accepted.

Explanation of Search by Simulated Annealing



Simulated Annealing

- Choice of an initial solution is not important.
- It is difficult to efficiently utilize a good initial solution.
- Some initial steps look useless (since they are almost random)
- Many final steps look useless (since they cannot move from a local solution)

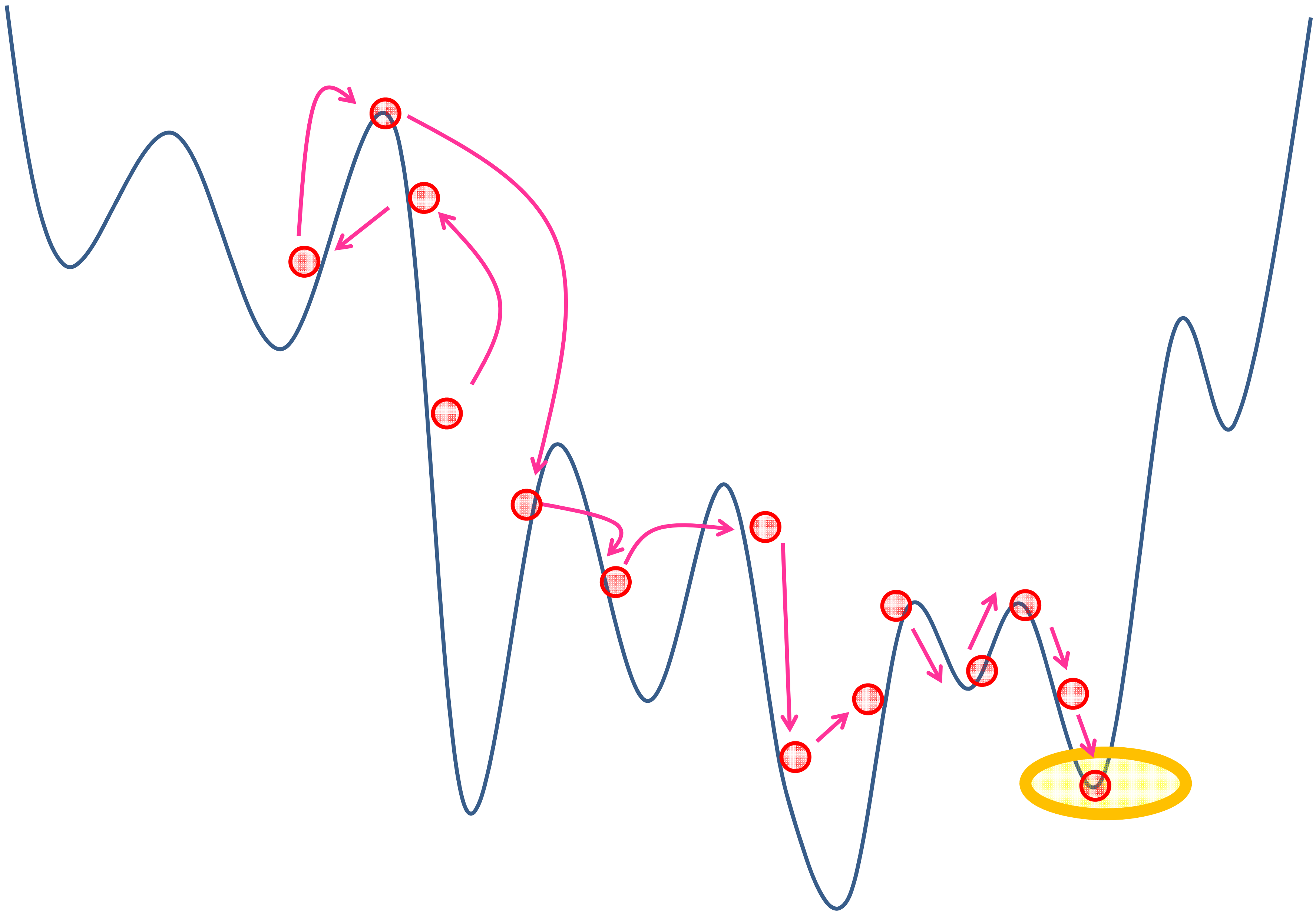
The Main Implementation Issue: Cooling Schedule

$$T = T(t),\ t = 1, 2, \ldots, t_{\max} \quad (t\text{: iteration index})$$

Initial value $T(1)$: Almost all moves are accepted.

Final value $T(t_{\max})$: No deterioration is accepted.

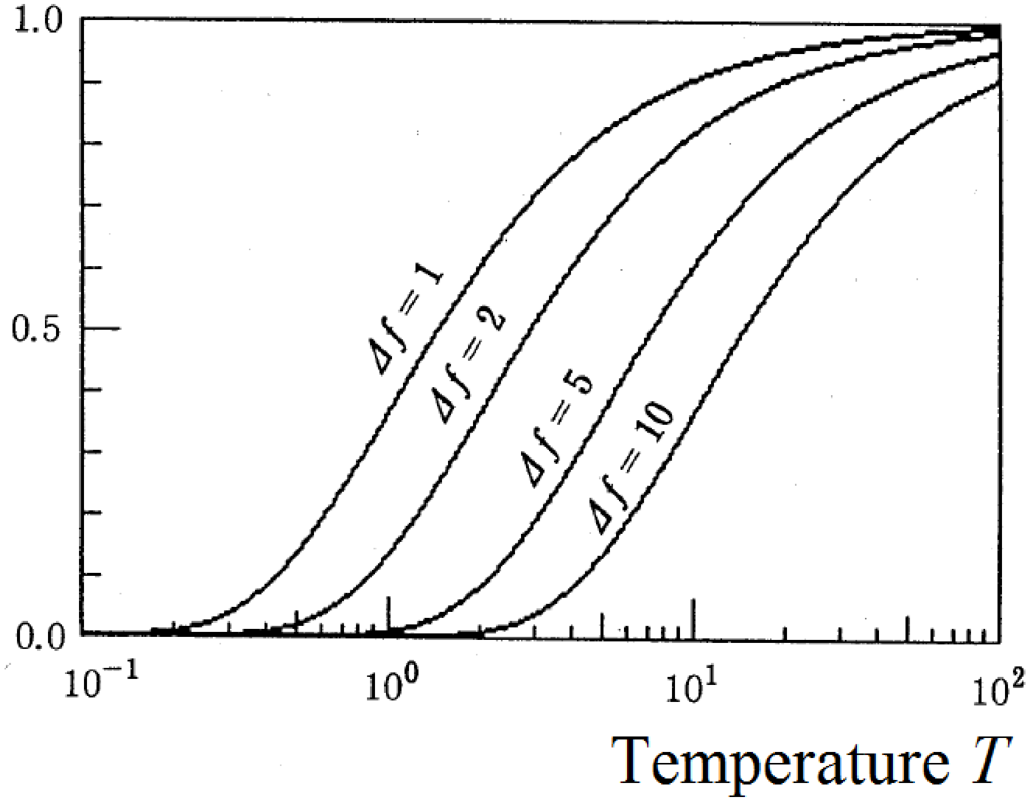
Explanation of Search by Simulated Annealing



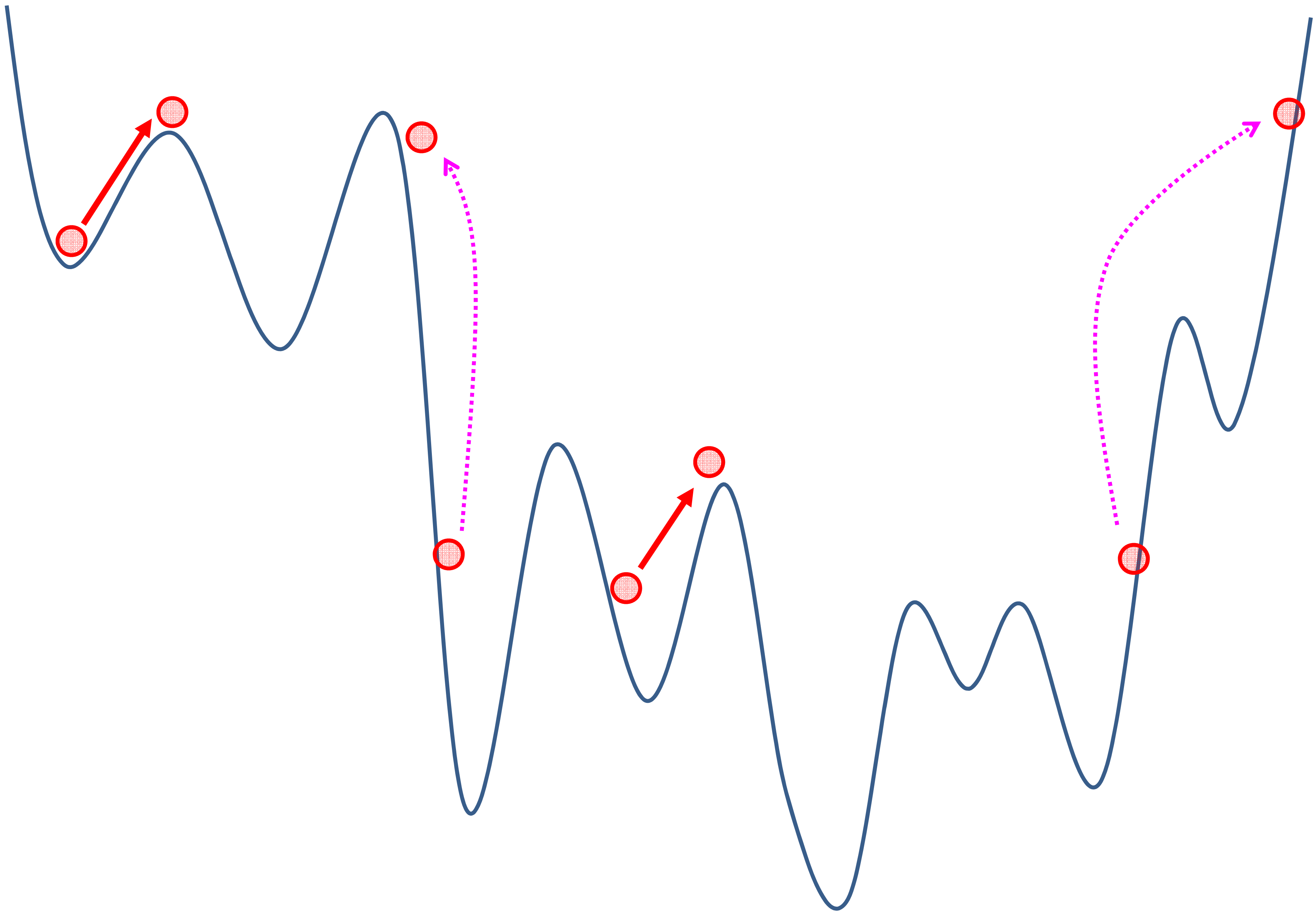
The Main Implementation Issue: Cooling Schedule

$T = T(t)$, $t = 1, 2, ..., t_{max}$ (t: iteration index)

Initial value $T(1)$: **About α% of moves are accepted.**



Explanation of Search by Simulated Annealing

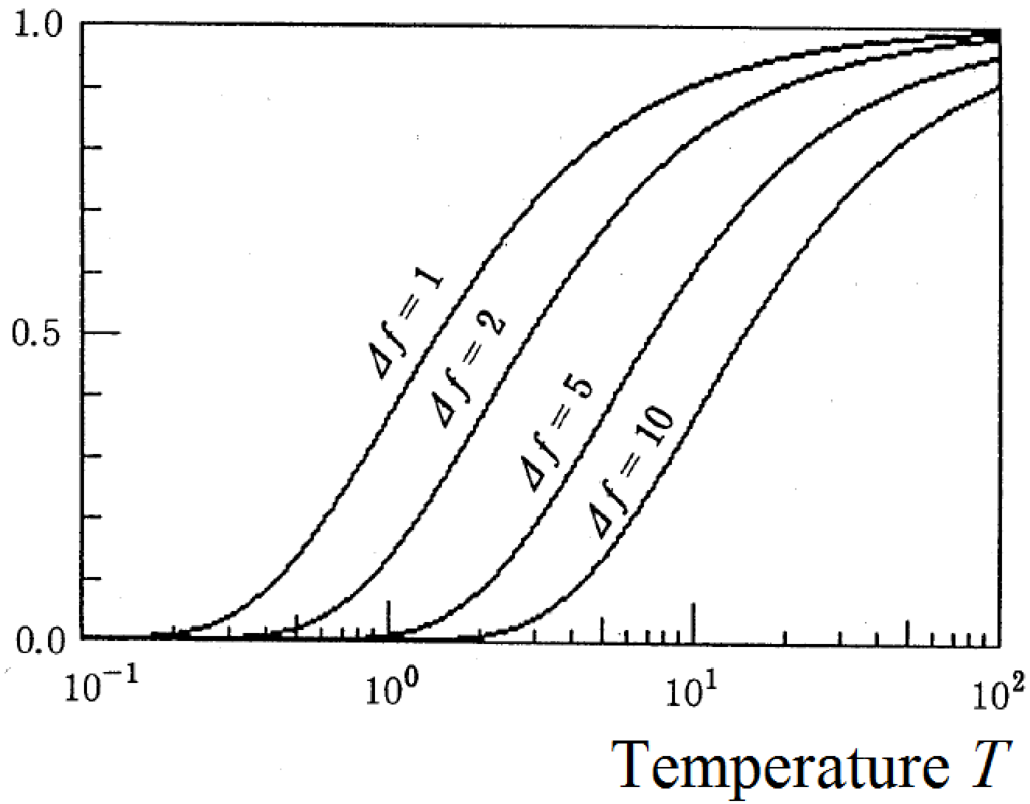


The Main Implementation Issue: Cooling Schedule

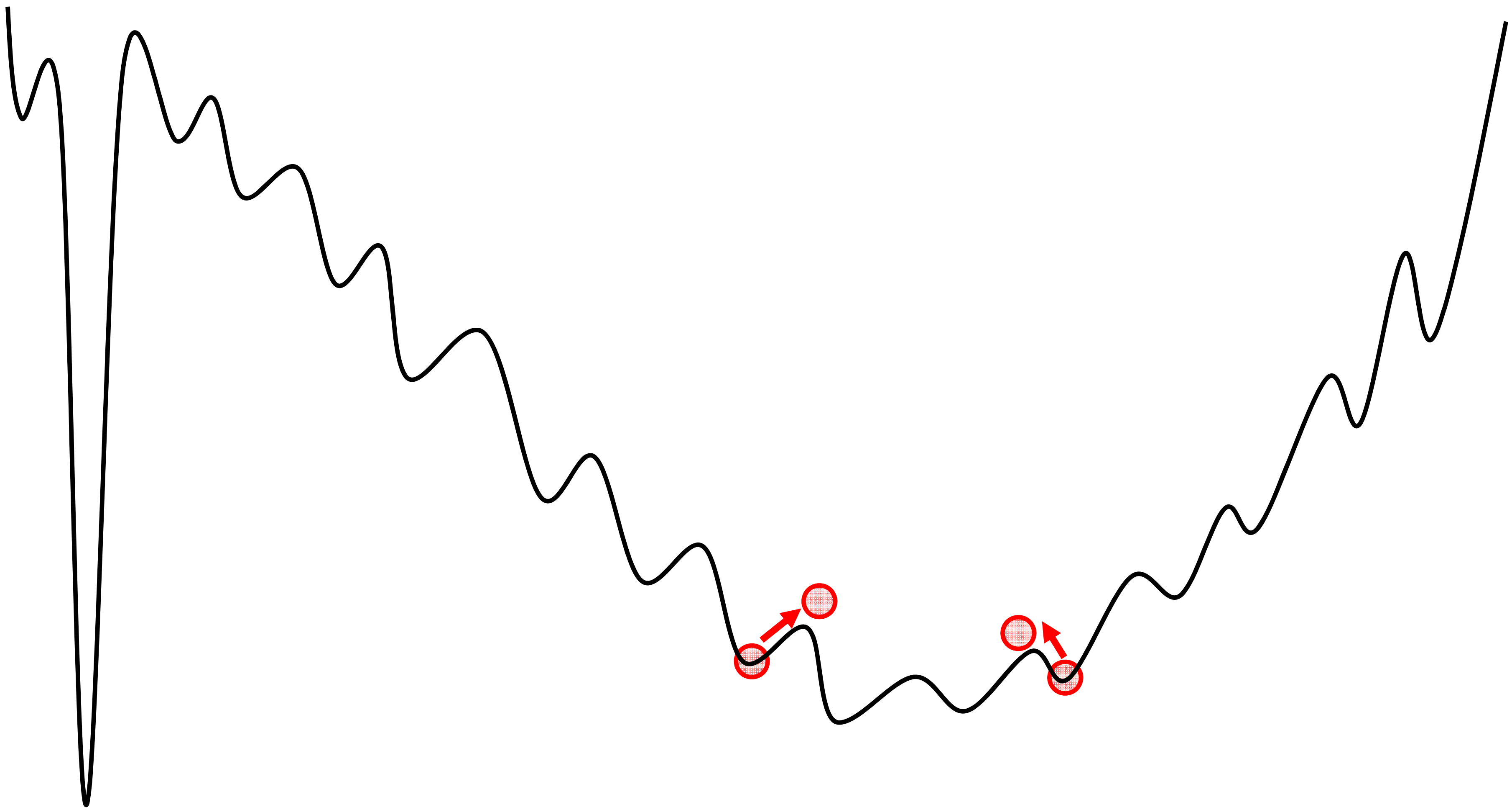
$T = T(t), t = 1, 2, ..., t_{max}$ **(t: iteration index)**

Initial value $T(1)$: **About α% of moves are accepted.**

Final value $T(t_{max})$: **The minimum deterioration can be accepted with the probability β.**



Explanation of Search by Simulated Annealing



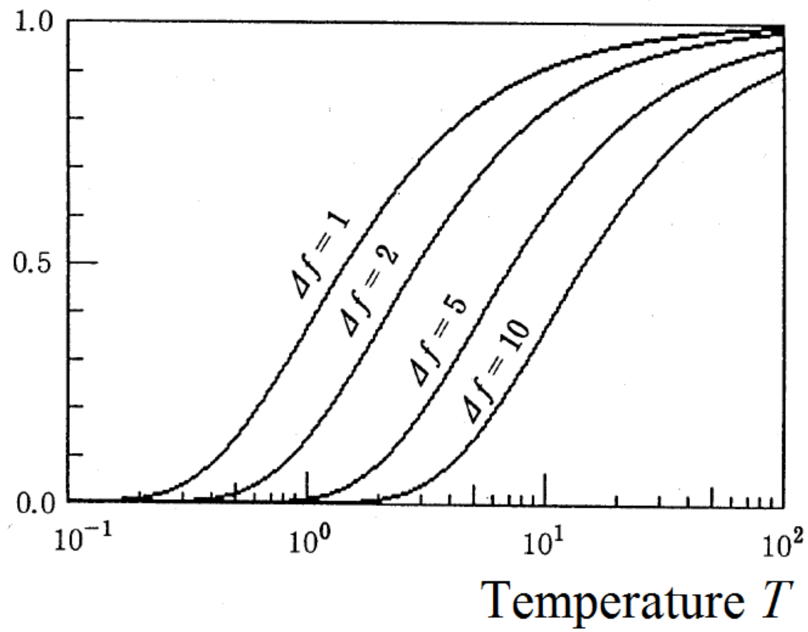
The Main Implementation Issue: Cooling Schedule

$T = T(t)$, $t = 1, 2, ..., t_{max}$ (t: iteration index)

Initial value $T(1)$: **About α% of moves are accepted.**

Final value $T(t_{max})$: **The minimum deterioration can be accepted with the probability β.**

$T(1)$ can be specified by sampling some neighbors.
$T(t_{max})$ can be specified from the problem characteristics for some problems (knapsack, scheduling).

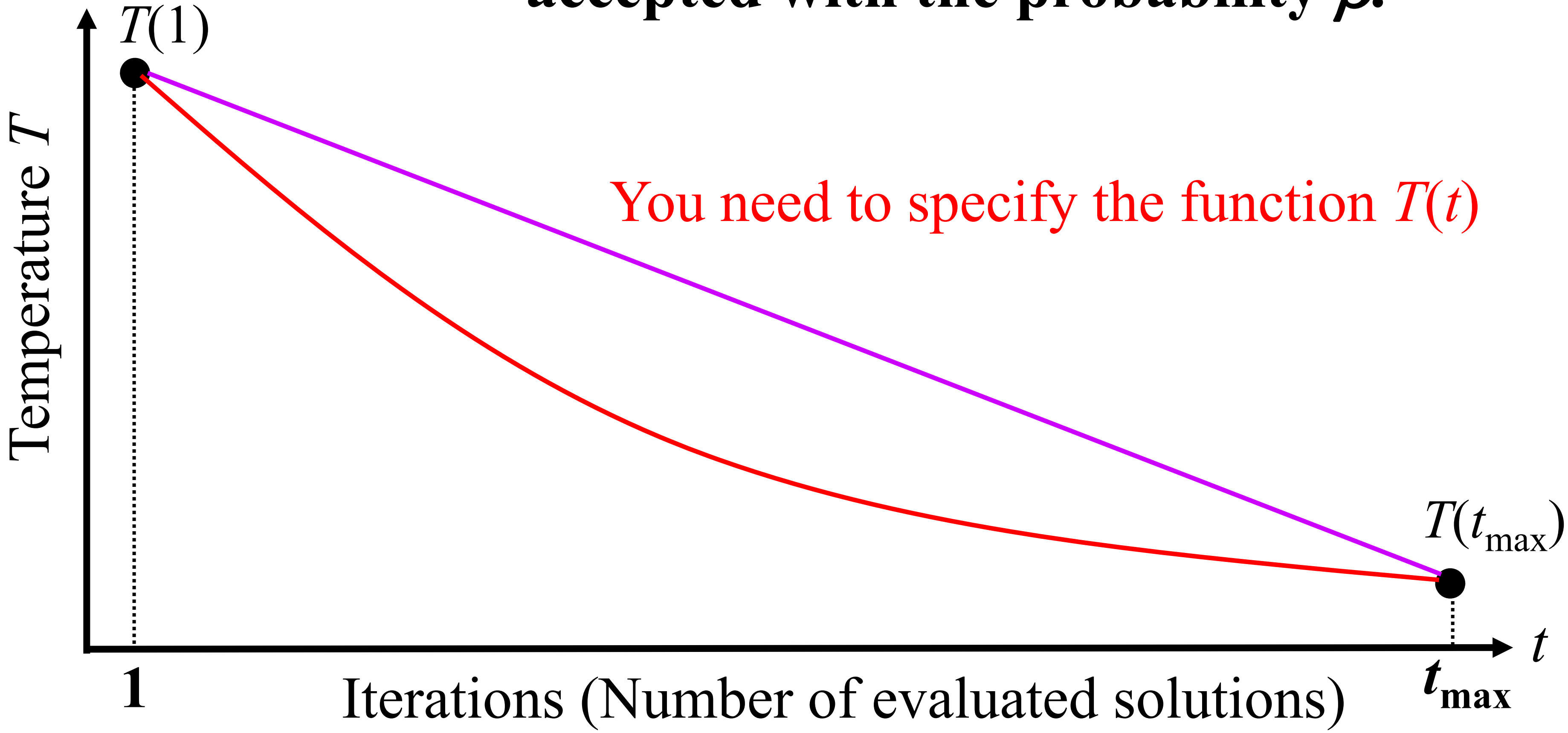


The Main Implementation Issue: Cooling Schedule

$T = T(t),\ t = 1, 2, \ldots, t_{max}$ (*t*: iteration index)

Initial value $T(1)$: **About α% of moves are accepted.**

Final value $T(t_{max})$: **The minimum deterioration can be accepted with the probability β.**

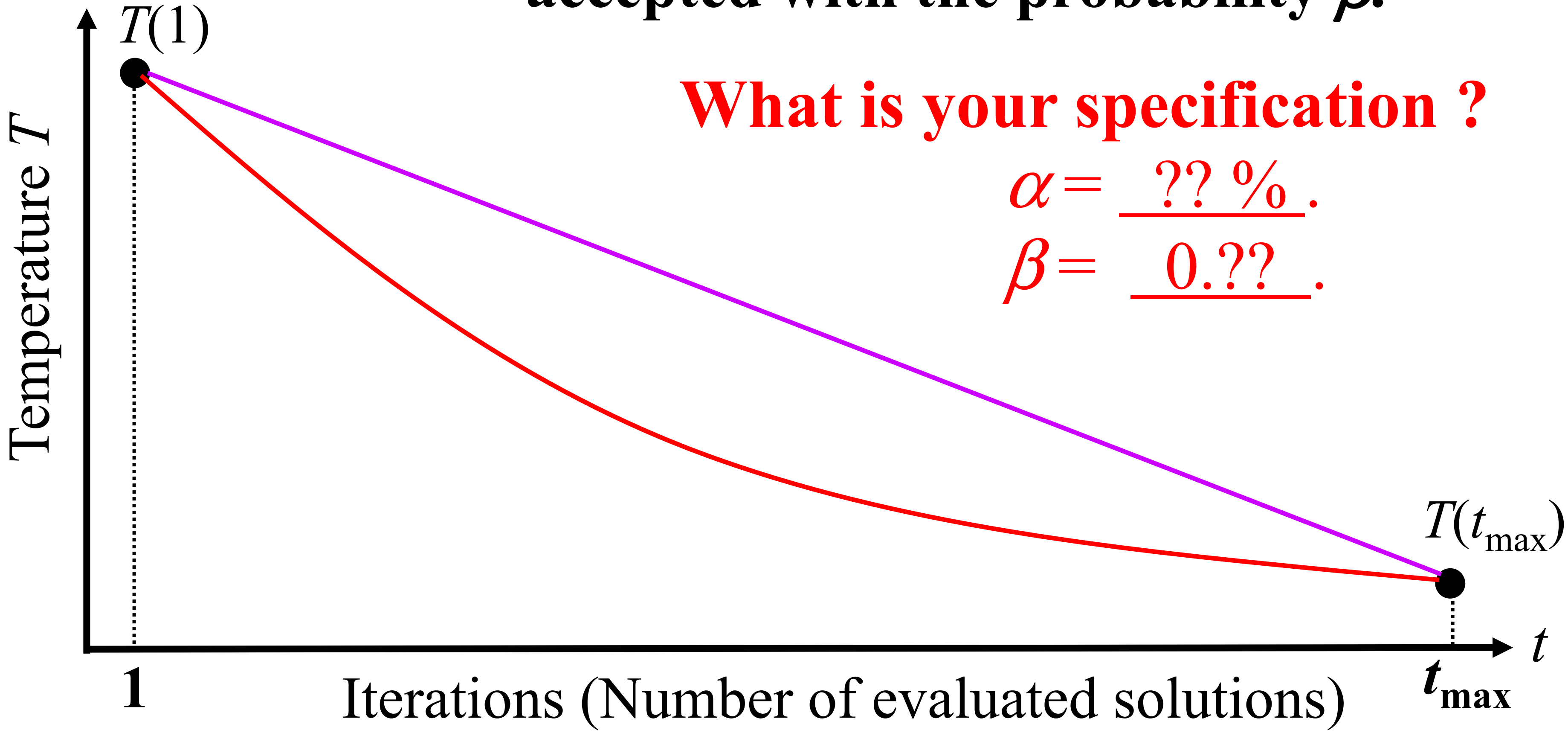


The Main Implementation Issue: Cooling Schedule

$T = T(t)$, $t = 1, 2, ..., t_{max}$ (t: iteration index)

Initial value $T(1)$: **About α% of moves are accepted.**

Final value $T(t_{max})$: **The minimum deterioration can be accepted with the probability β.**



What is your specification ?

$\alpha =$?? %.

$\beta =$ 0.??.

Simulated Annealing

- Choice of an initial solution is not important.
- It is difficult to efficiently utilize a good initial solution.
- Some initial steps look useless (since they are almost random)
- Many final steps look useless (since they cannot move from a local solution)

Some Simple Modifications:

(i) Use a fixed temperature (to escape from a local solution).

Simulated Annealing

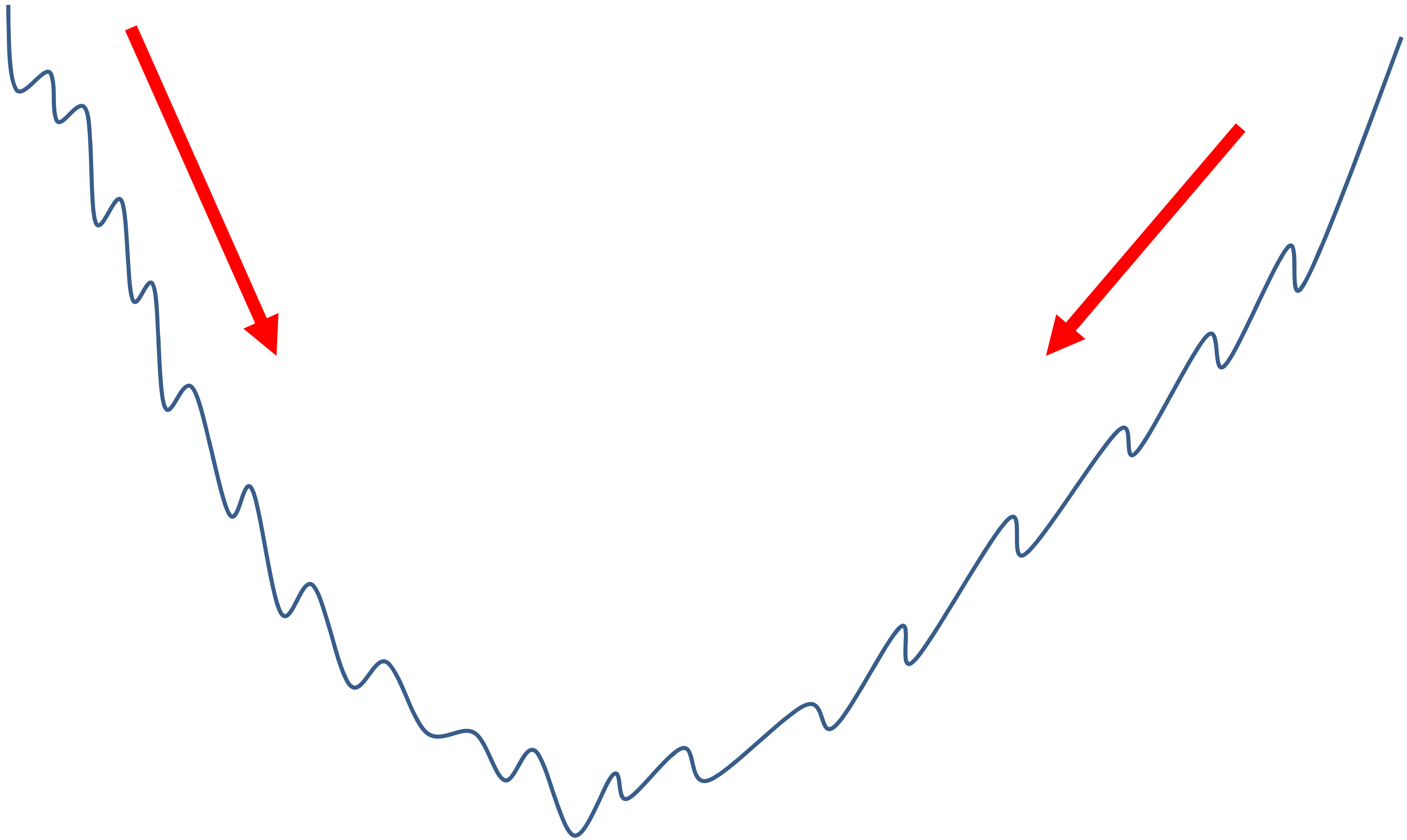
- Choice of an initial solution is not important.
- It is difficult to efficiently utilize a good initial solution.
- Some initial steps look useless (since they are almost random)
- Many final steps look useless (since they cannot move from a local solution)

Some Simple Modifications:

(i) Use a fixed temperature (to escape from a local solution).

(ii) Randomly select K neighbors of the current solution $\boldsymbol{x}$, choose the best neighbor $\boldsymbol{y}$ among the K neighbors, and apply the acceptance rule to $\boldsymbol{y}$ (to utilize a good initial solution).

Explanation of Search by Simulated Annealing



Simulated Annealing

- Choice of an initial solution is not important.
- It is difficult to efficiently utilize a good initial solution.
- Some initial steps look useless (since they are almost random)
- Many final steps look useless (since they cannot move from a local solution)

Some Simple Modifications:

(i) Use a fixed temperature (to escape from a local solution).

(ii) Randomly select K neighbors of the current solution $\boldsymbol{x}$, choose the best neighbor $\boldsymbol{y}$ among the K neighbors, and apply the acceptance rule to $\boldsymbol{y}$ (to utilize a good initial solution).

===> Useful in Anytime Optimization Framework

Anytime Algorithms

Requirements:

Interruptibility:

The algorithm can be stopped at any time and provide some answer.

Preemptability:

The algorithm can be stopped at any time and restarted again with minimal overhead.

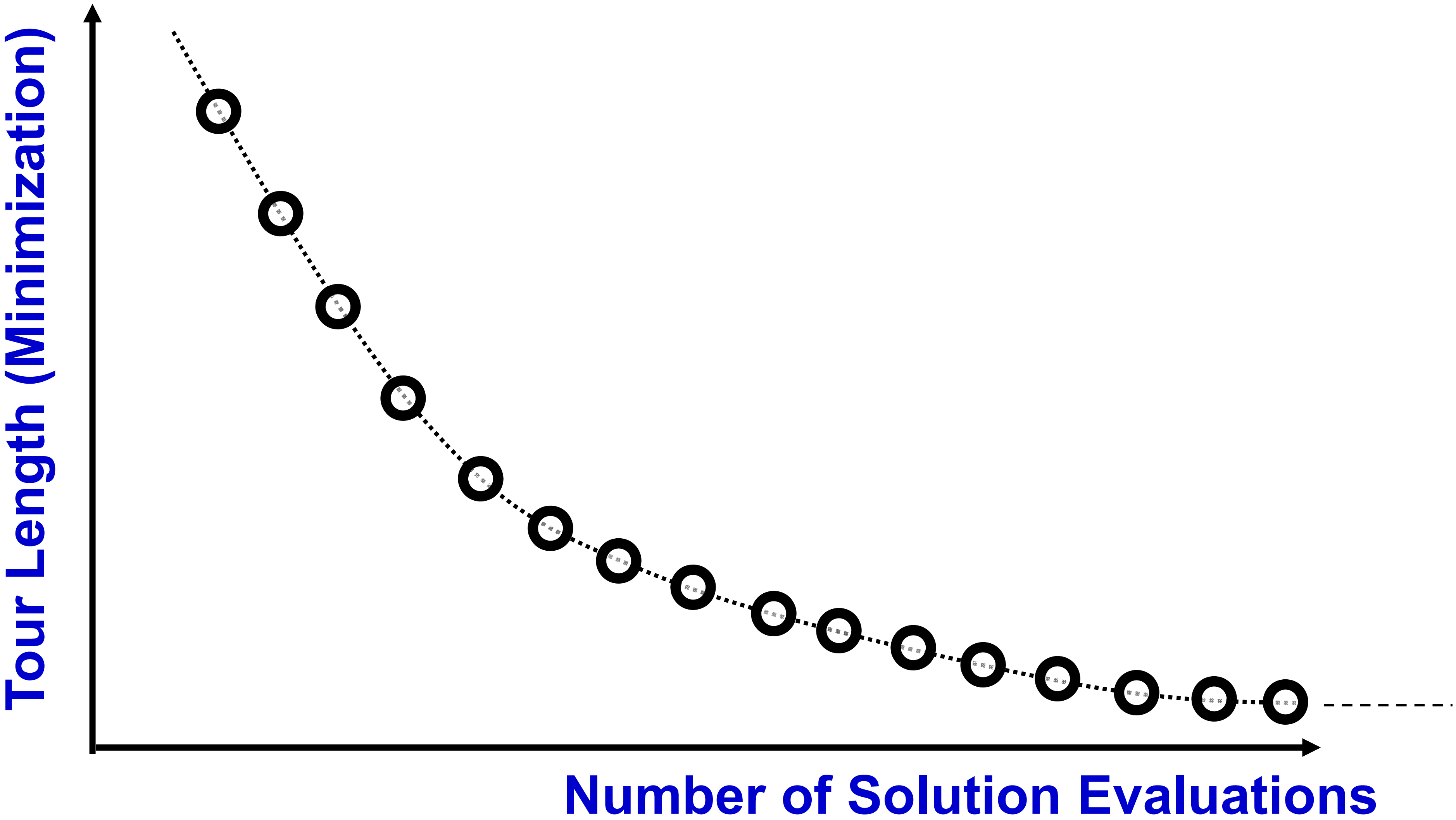
Anytime Algorithms

Two Algorithm Categories

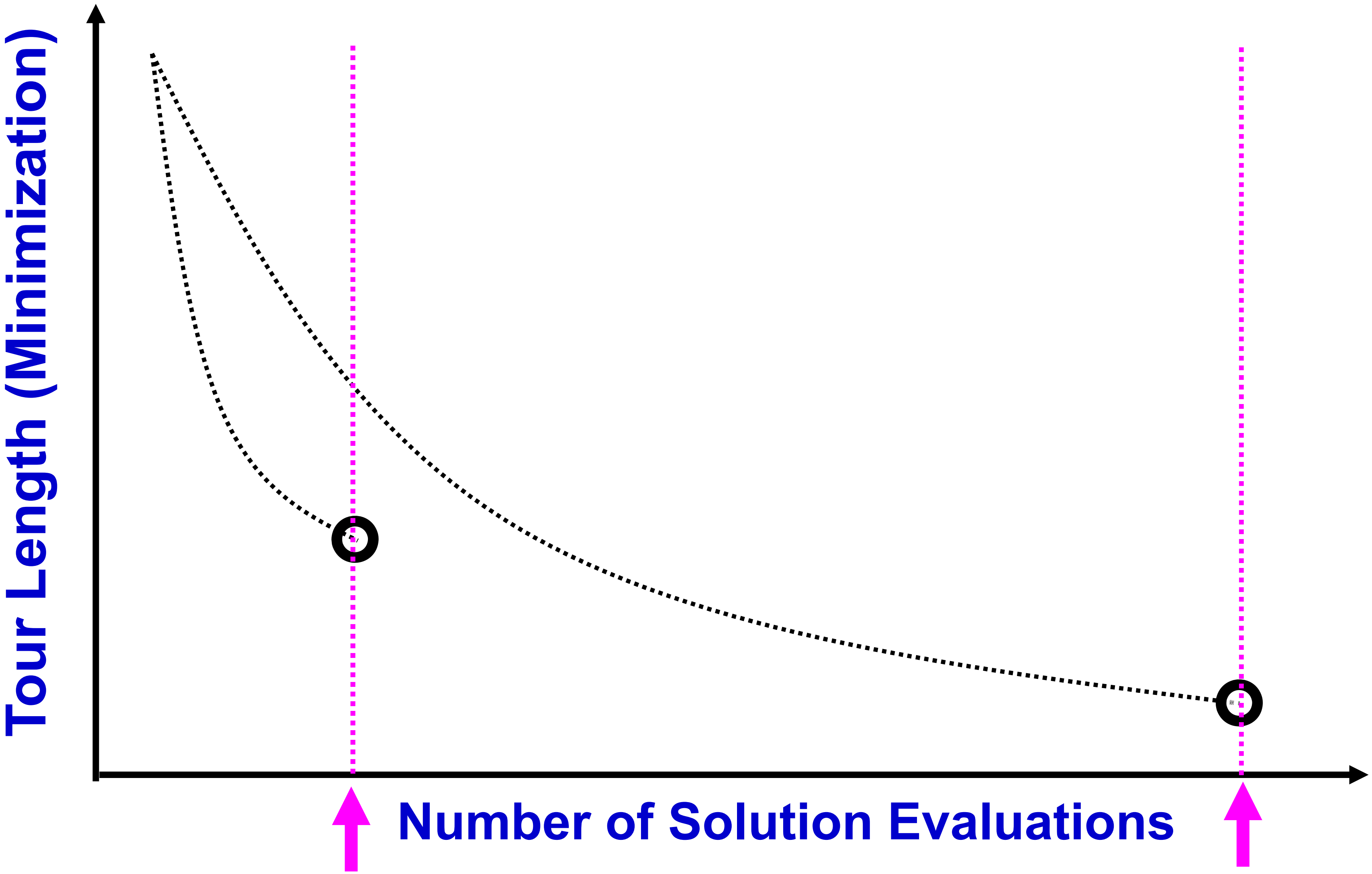
Interruptible Algorithms:
The algorithm continues to generate solutions over time. (The total available time is not given in advance.)

Contract algorithms:
When the total available time is given in advance, the algorithm adjusts its search behavior based on the given total available time. If the algorithm is terminated before the given total available time, the solution quality can be very poor.

Interruptible Algorithm

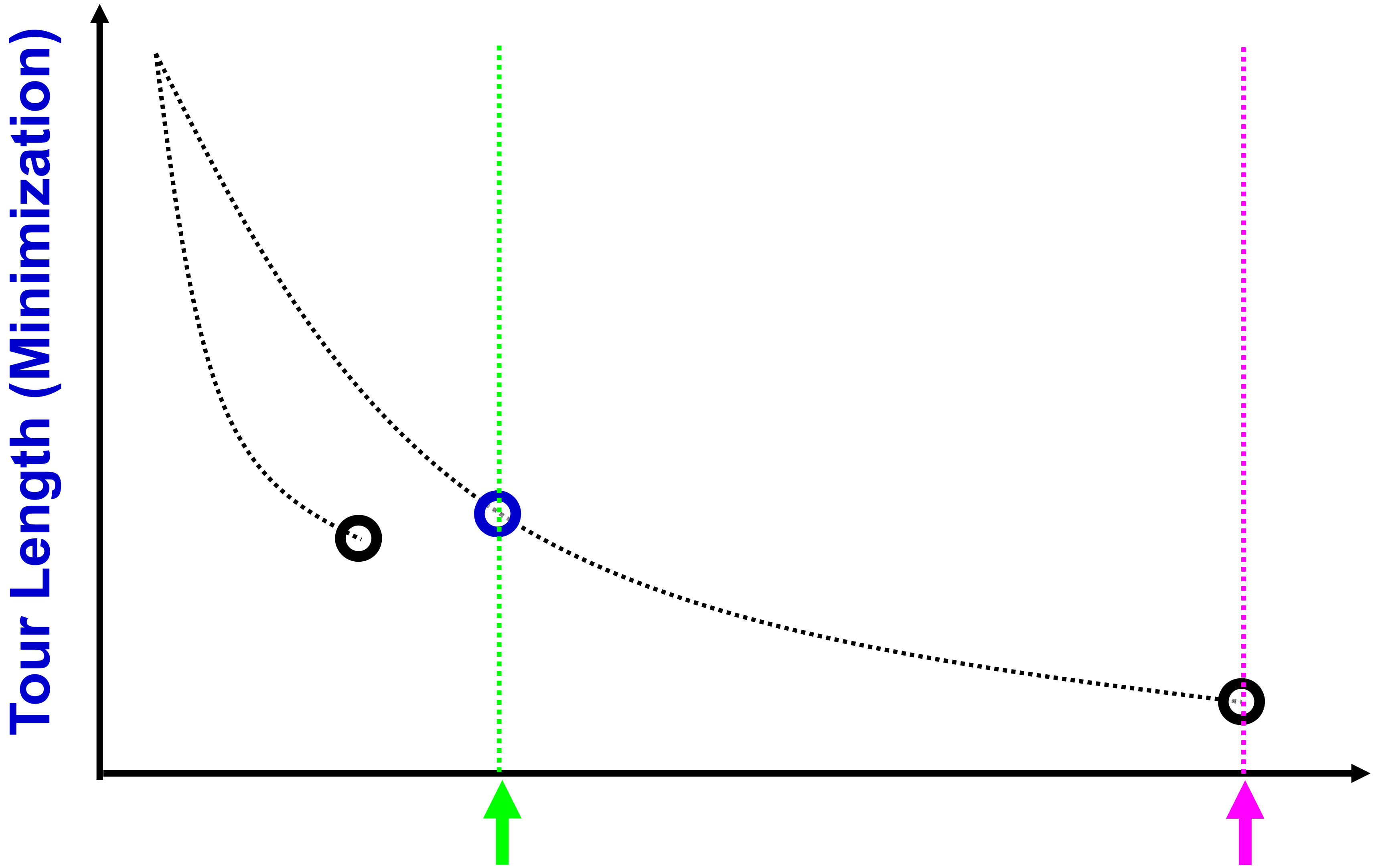


Contract algorithm



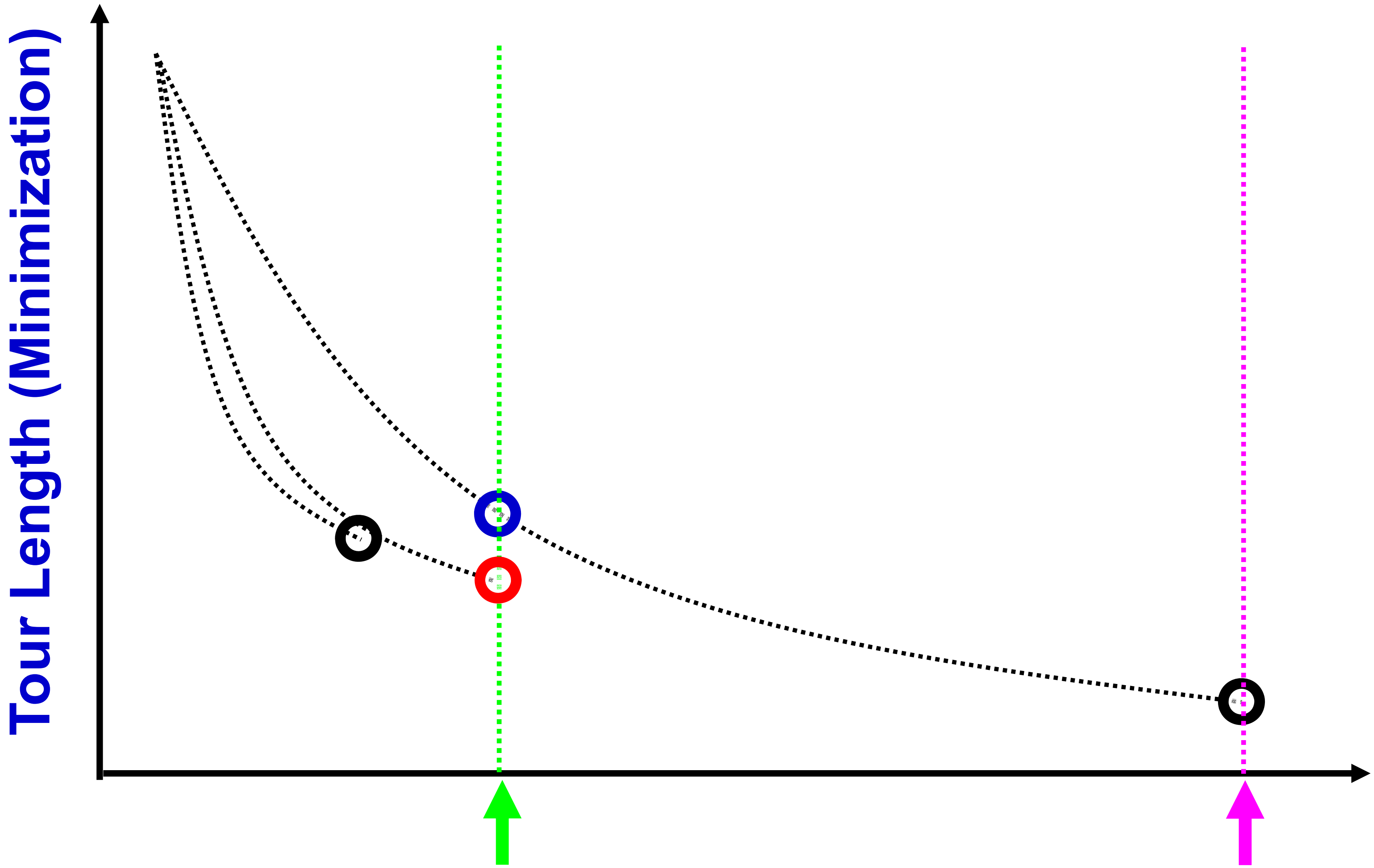
The available computation time is given in advance.

Contract algorithm

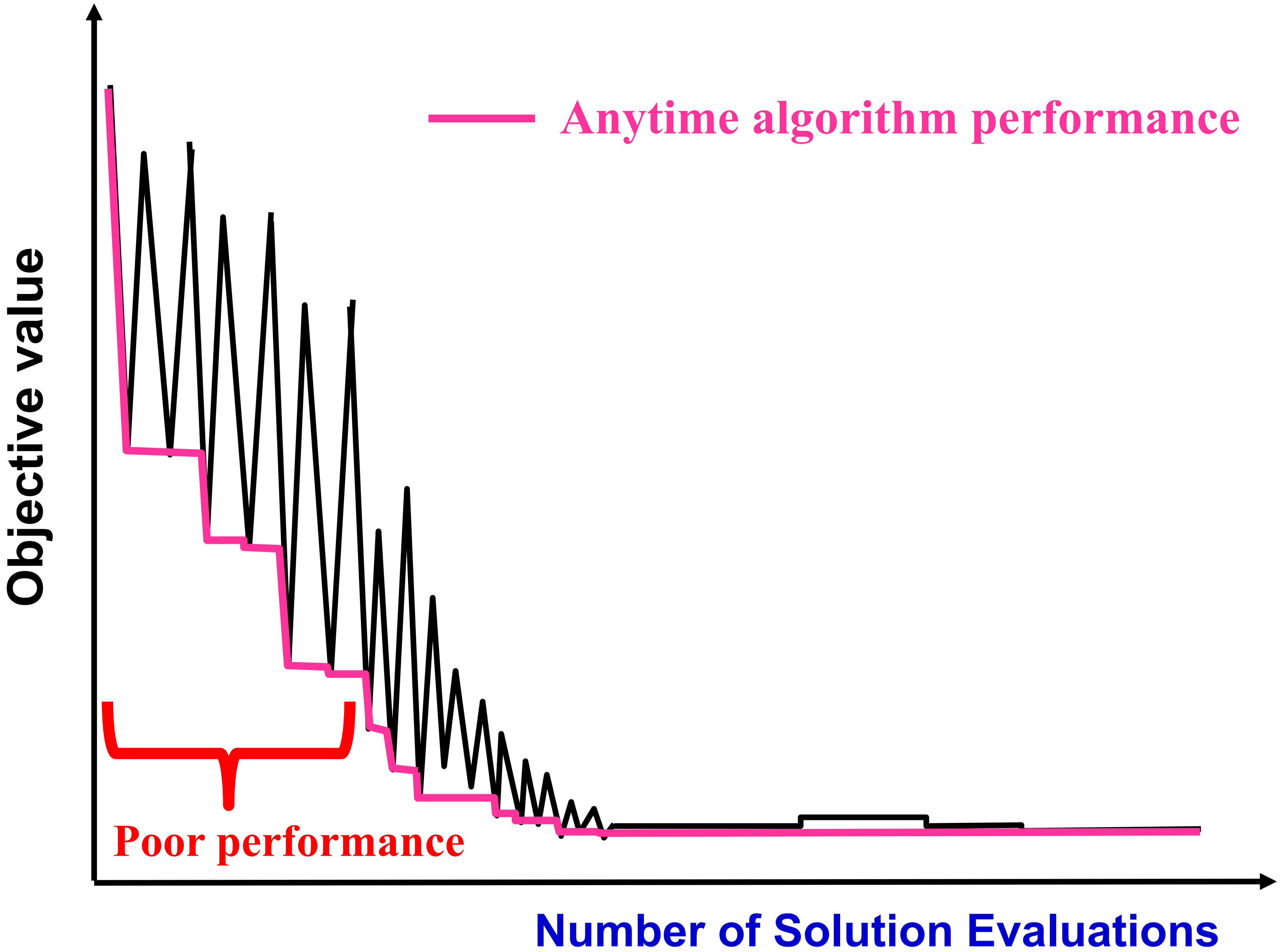


If the algorithm is terminated earlier, its performance can be poor.

Contract algorithm



If the algorithm is terminated earlier, its performance can be poor.


Anytime algorithm performance
Objective value
Poor performance
Number of Solution Evaluations

Main Advantage of Simulated Annealing

- **General Purpose Algorithm**
  (which is applicable to various optimization problems).
- **High Performance**
  (better than LS on problems with many local solutions).
- **Easy Implementation**
  If we have a local search algorithm, we can implement its SA version by specifying a cooling schedule. We do not have to worry about
  - Specification of an initial solution.
  - Termination of the current local search run.

Main Advantage of Simulated Annealing

- **General Purpose Algorithm ==> Wide Applications** (which is applicable to various optimization problems).
- **High Performance ==> Not Always** (better than LS on problems with many local solutions).
- **Easy Implementation ==> Easy to Modify (e.g., Parallel)** If we have a local search algorithm, we can implement its SA version by specifying a cooling schedule. We do not have to worry about
  - Specification of an initial solution.
  - Termination of the current local search run.

Lab Session Task 1:

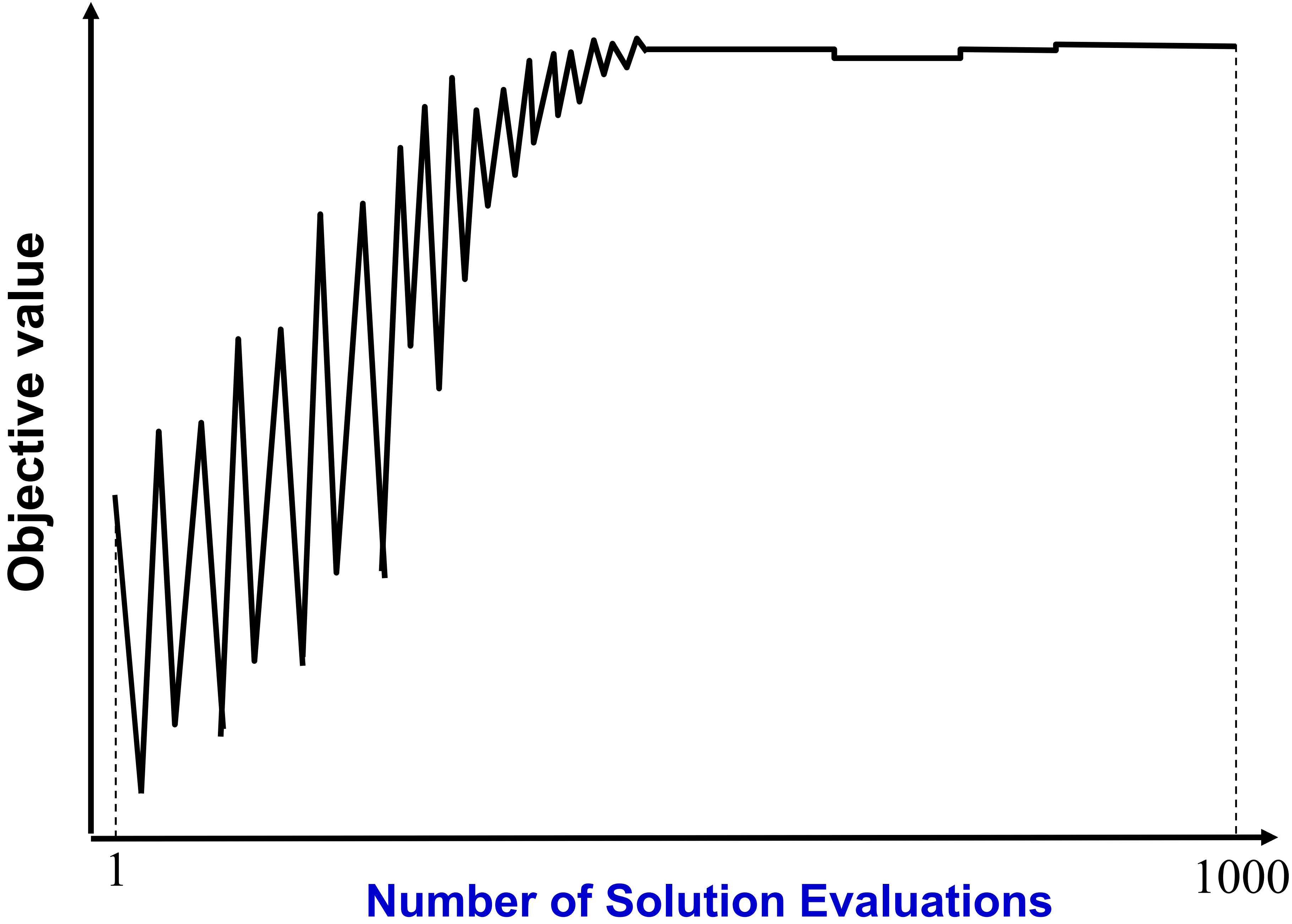
You have a 100-item knapsack problem with 10 constraint conditions.

$$\text{Maximize } f(\boldsymbol{x}) = \sum_{i=1}^{100} v_i x_i \quad \text{Subject to } \sum_{i=1}^{100} w_{ij} x_i \le W_j,\ j = 1, 2, \ldots, 10,$$

$$\text{and } x_i \in \{0, 1\},\ i = 1, 2, \ldots, 100$$

First we create an initial feasible solution as follows: Include items one by one in the knapsack in the order of item 1, item 2, ..., item 100. If the (k+1)-th item cannot be included in the knapsack, the initial feasible solution is created by including the1st, 2nd, ..., kth items. We define the neighbors of the current solution as follows: "Neighbors are feasible solutions with the Hamming distance D from the current solution". Three values of D are examined: D = 1, 2, 3. A termination condition is specified as 1,000 solution evaluations. Under these conditions, apply a simulated annealing algorithm to this knapsack problem with various cooling schedules (e.g., constant high temperature, constant low temperature, constant medium temperature, quick decrease of the temperature, slow decrease of the temperature).

The results are compared by drawing the following figure:

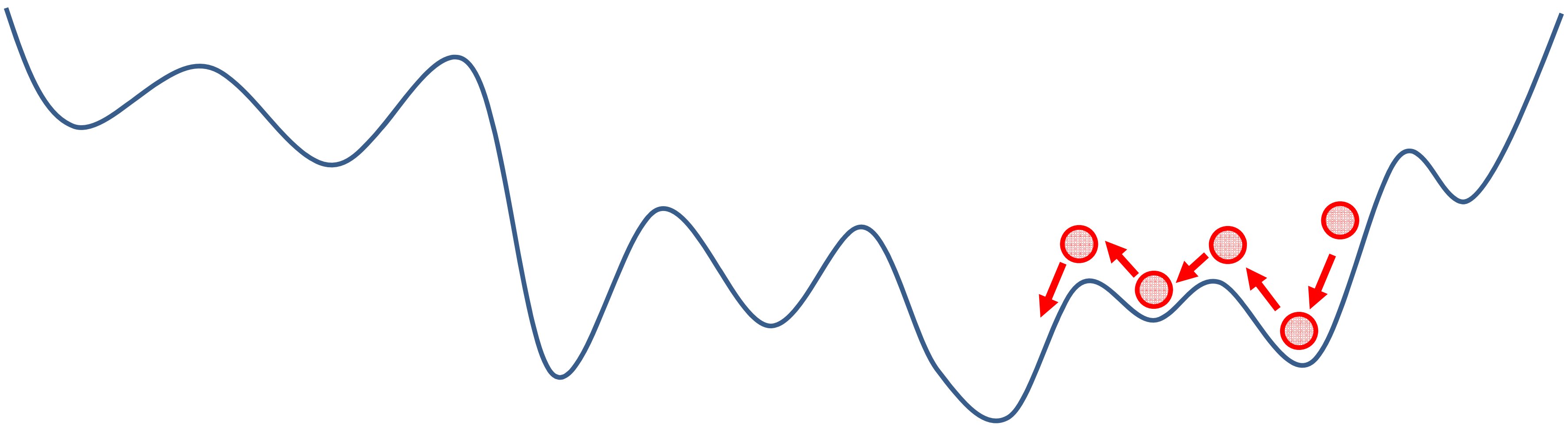


Move to a Better Solution

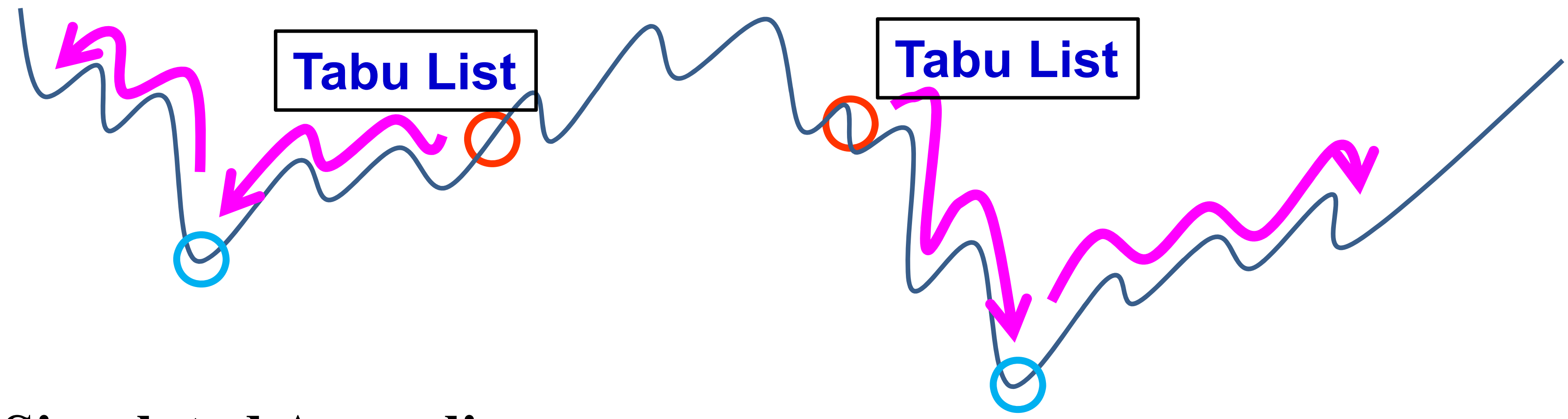
- Local Search (LS)
- Iterated Local Search (ILS)
- Variable Neighborhood Search (VNS)

Allow the Move to a Worse Solution

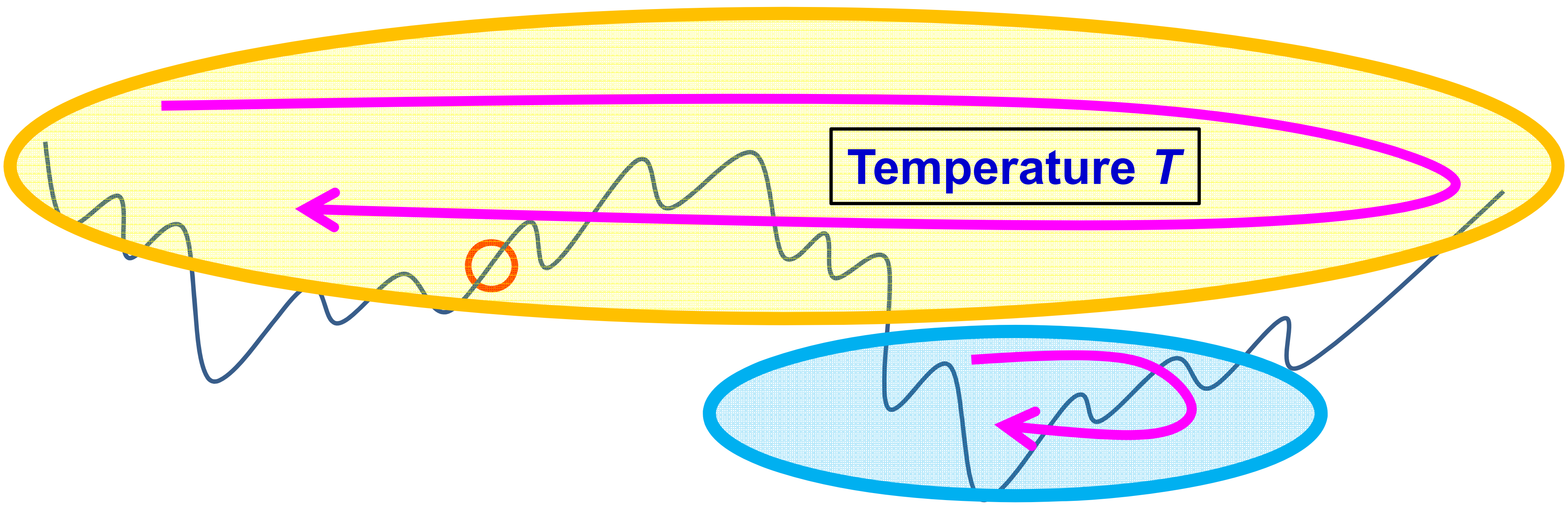
- Simulated Annealing (SA)
- **Tabu Search (TS)**



Tabu Search



Simulated Annealing



Tabu Search (Taboo Search)

Google Scholar

Fred Glover

FOLLOW

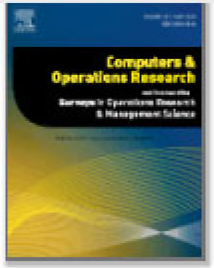
Distinguished Professor, University of Colorado & Chief Scientific Officer, Entanglement, Inc.
Verified email at entanglement.ai - Homepage

Optimization computational intelligence quantum computing

TITLE	CITED BY	YEAR
Tabu search F Glover, M Laguna Handbook of combinatorial optimization: Volume1–3, 2093-2229	10296	1998
Tabu search—part I F Glover ORSA Journal on computing 1 (3), 190-206	10247	1989
Future paths for integer programming and links to artificial intelligence F Glover Computers operations research 13 (5), 533-549	6853	1986
Tabu search—part II F Glover ORSA Journal on computing 2 (1), 4-32	6212	1990

Computers & Operations Research

Volume 13, Issue 5, 1986, Pages 533-549



Future paths for integer programming and links to artificial intelligence

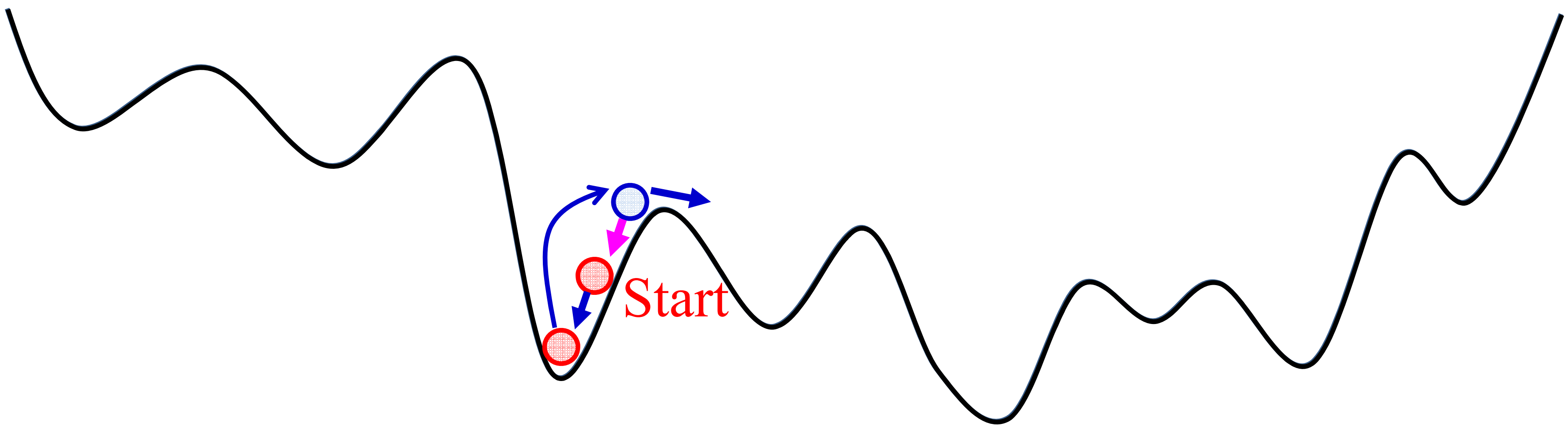
Fred Glover *

⊞ **Show more**

https://doi.org/10.1016/0305-0548(86)90048-1

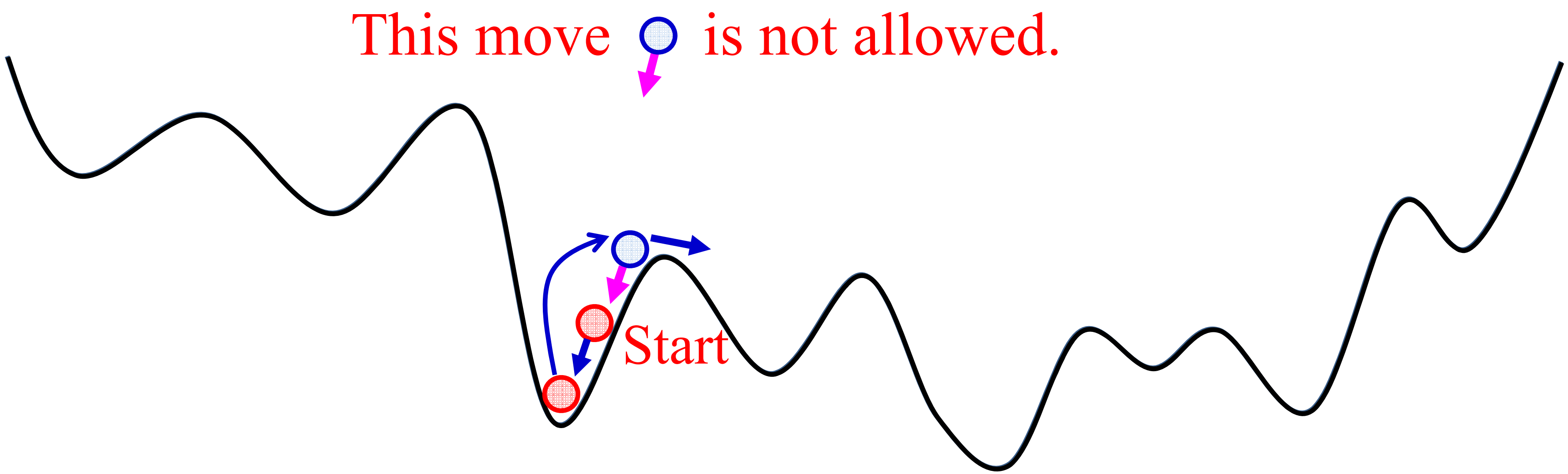
Tabu Search

Basic Idea: To prevent the local search from going back to the same local solution by using a list of tabu moves (a list of banned moves).



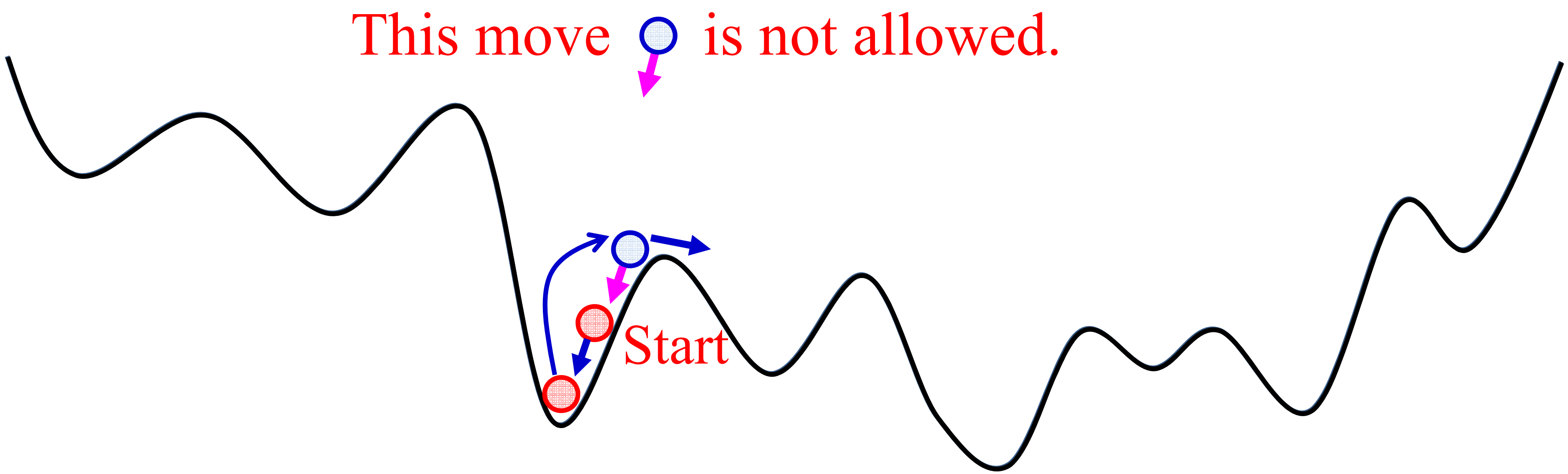
Tabu Search

Basic Idea: To prevent the local search from going back to the same local solution by using a list of tabu moves (a list of banned moves).



Tabu Search

Basic Idea: To prevent the local search from going back to the same local solution by using a list of tabu moves (a list of banned moves).

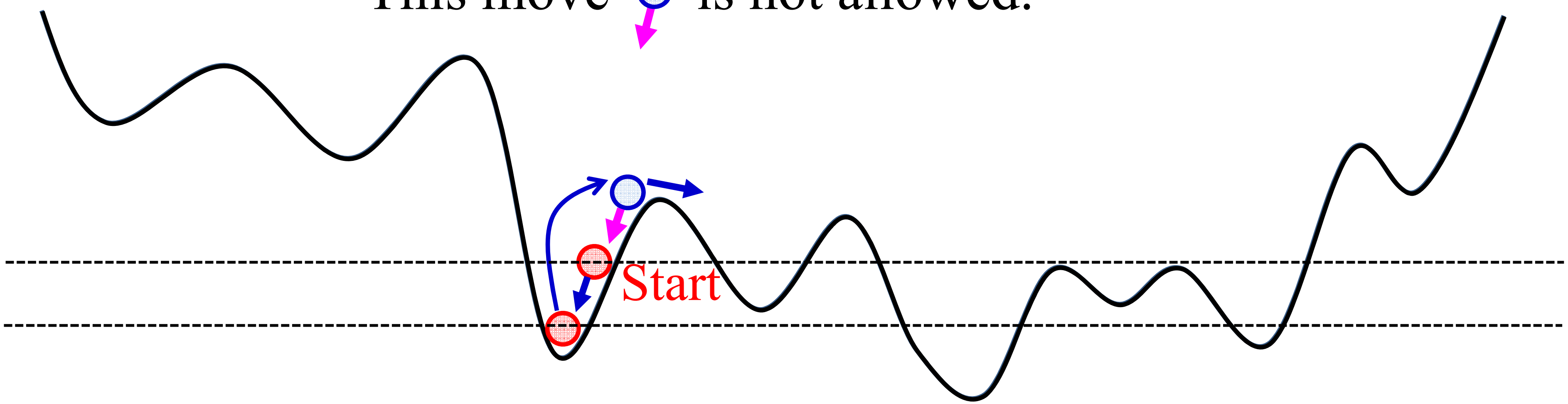


Point: How to create and update a Tabu list.

Tabu Search

Basic Idea: To prevent the local search from going back to the same local solution by using a list of tabu moves (a list of banned moves).

This move is not allowed.



Point: How to create and update a Tabu list.

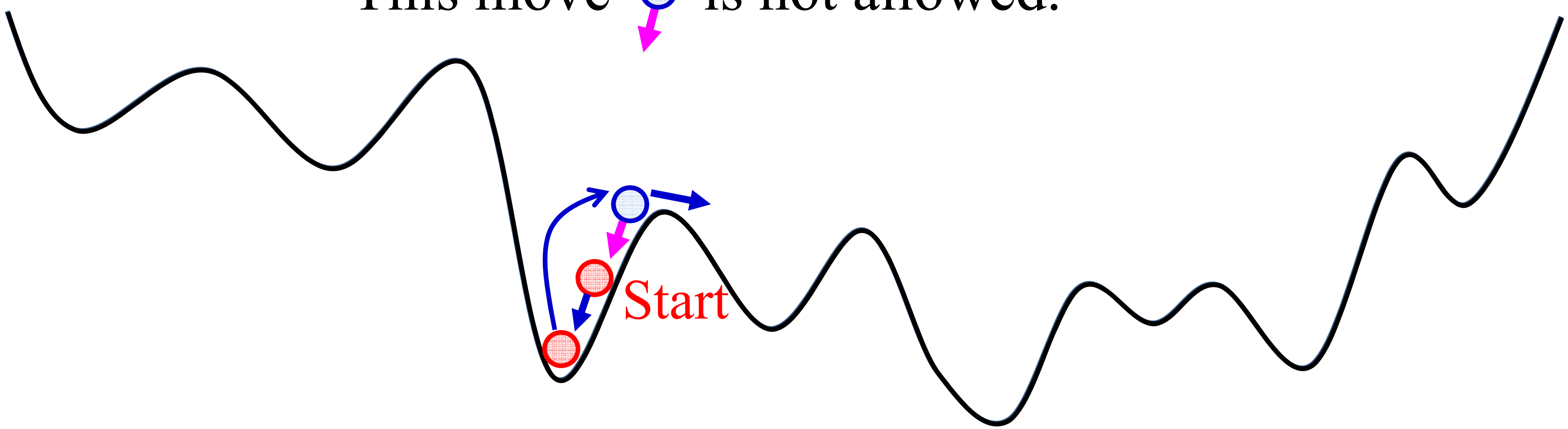
Elements of a tabu list:

(1) Objective function values

Tabu Search

Basic Idea: To prevent the local search from going back to the same local solution by using a list of tabu moves (a list of banned moves).

This move is not allowed.



Point: How to create and update a Tabu list.

Elements of a tabu list:

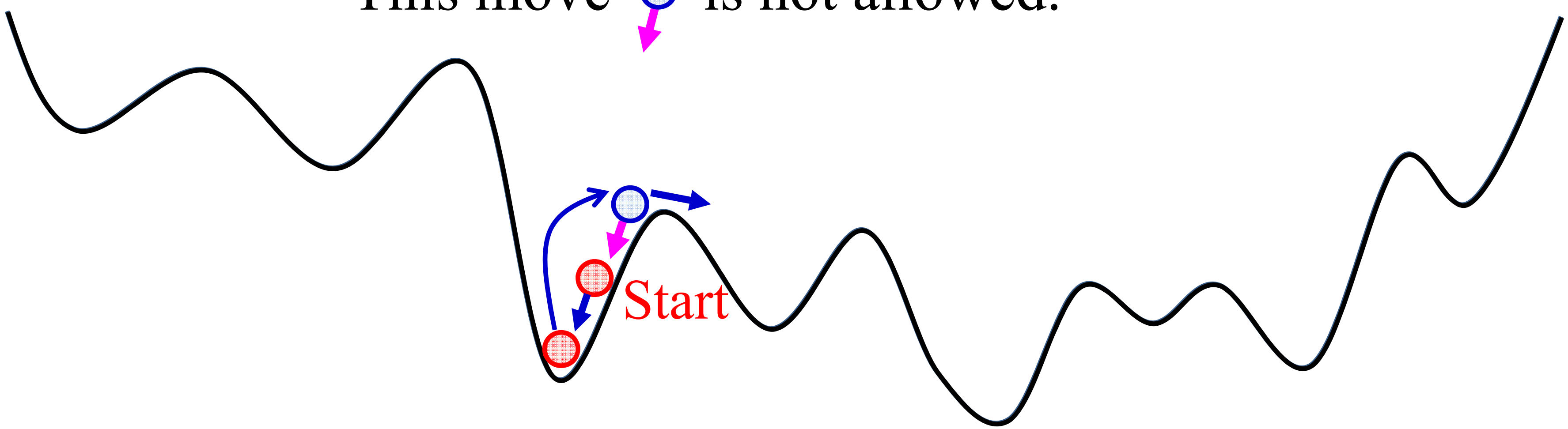
(1) Objective function values

(2) Moves

Tabu Search

Basic Idea: To prevent the local search from going back to the same local solution by using a list of tabu moves (a list of banned moves).

This move is not allowed.



Point: How to create and update a Tabu list.

Elements of a tabu list:

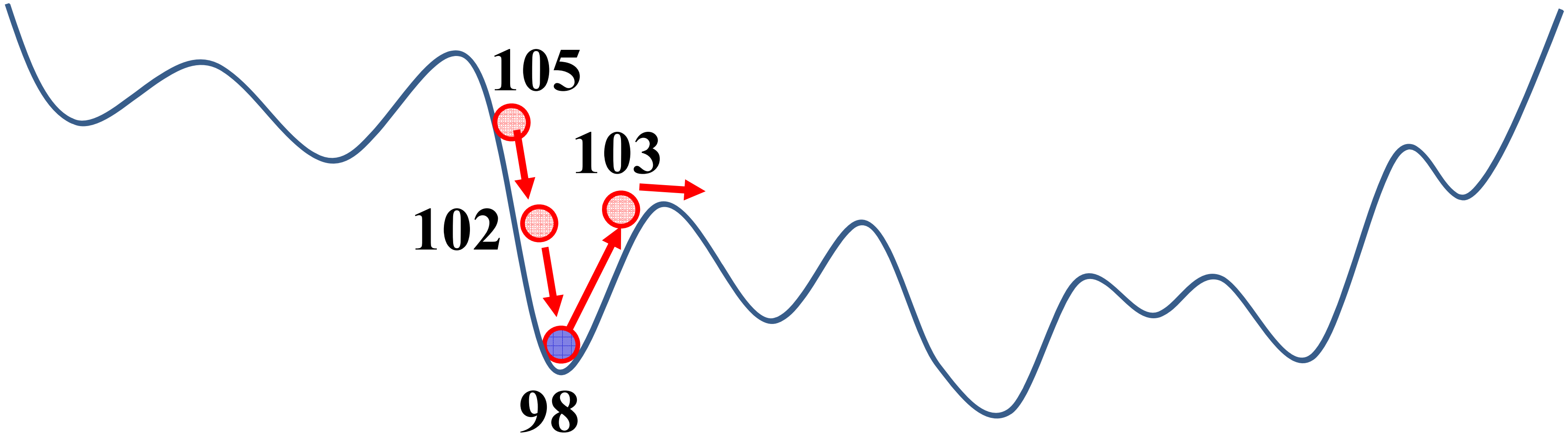
(1) Objective function values

(2) Moves

(3) Solutions

Example of Tabu List

Objective Function Values: For example, when the recent two moves by local search are 105 ==> 102 ==> 98, the two objective values {105, 102} are included in the tabu list for the next move from the current solution with the objective value 98 (if the tabu list length is 2). Neighbors with those objective values in the tabu list are not selected as the next solution. Let us assume that the objective value of the next solution is 103. Then the tabu list is update as {102, 98} for the next move from the current solution with the objective value 103 (when the tabu list length is 2).



Example of Tabu List

Local moves: Pairs of exchanged positions (Tabu list length 3)

1	2	3	4	5	6	7	8	9	{ }
1	2	3	4	5	8	7	6	9	{ (6, 8) }
1	2	7	4	5	8	3	6	9	{ (6, 8), (3, 7) }
3	2	7	4	5	8	1	6	9	{ **(?, ?), (?, ?), (?, ?)** }

Example of Tabu List

Local moves: Pairs of exchanged positions (Tabu list length 3)

1	2	3	4	5	**6**	7	**8**	9	{ }
1	2	**3**	4	5	8	**7**	6	9	{ (6, 8) }
1	2	7	4	5	8	**3**	6	9	{ (6, 8), (3, 7) }
3	2	7	**4**	5	8	1	**6**	9	{ (6, 8), (3, 7), (1, 7) }
3	2	7	6	5	8	1	4	9	{ **(?, ?), (?, ?), (?, ?)** }

Example of Tabu List

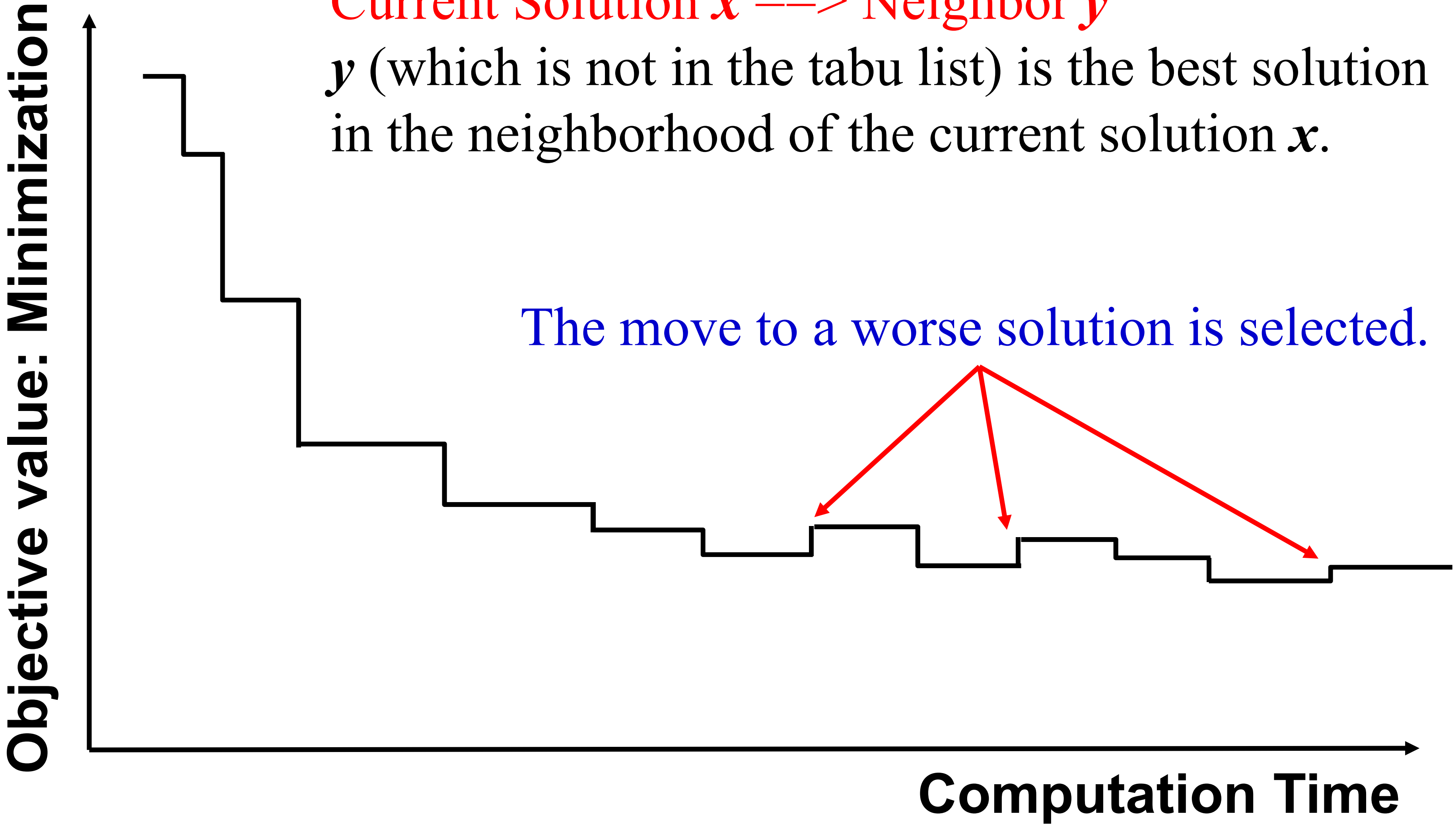
Local moves: Pairs of exchanged positions (Tabu list length 3)

1	2	3	4	5	6	7	8	9	{ }
1	2	3	4	5	8	7	6	9	{ (6, 8) }
1	2	7	4	5	8	3	6	9	{ (6, 8), (3, 7) }
3	2	7	4	5	8	1	6	9	{ (6, 8), (3, 7), (1, 7) }
3	2	7	6	5	8	1	4	9	{ (3, 7), (1, 7), (4, 8) }

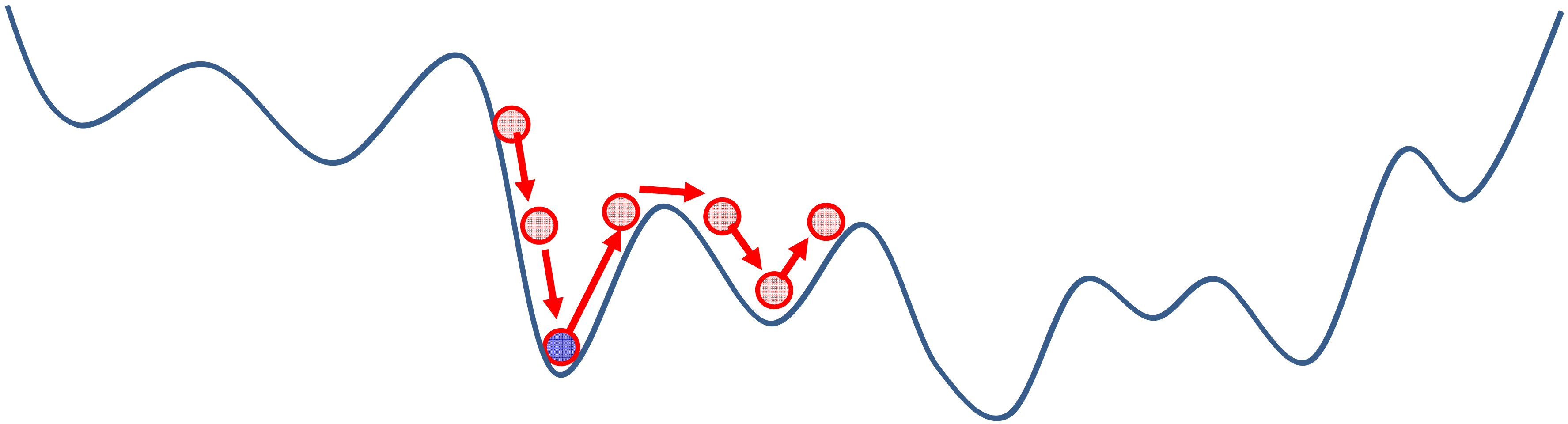
Explanation of Search behavior of Tabu Search

Current Solution ***x*** ==> Neighbor ***y***

y (which is not in the tabu list) is the best solution in the neighborhood of the current solution ***x***.



Explanation of Search behavior of Tabu Search

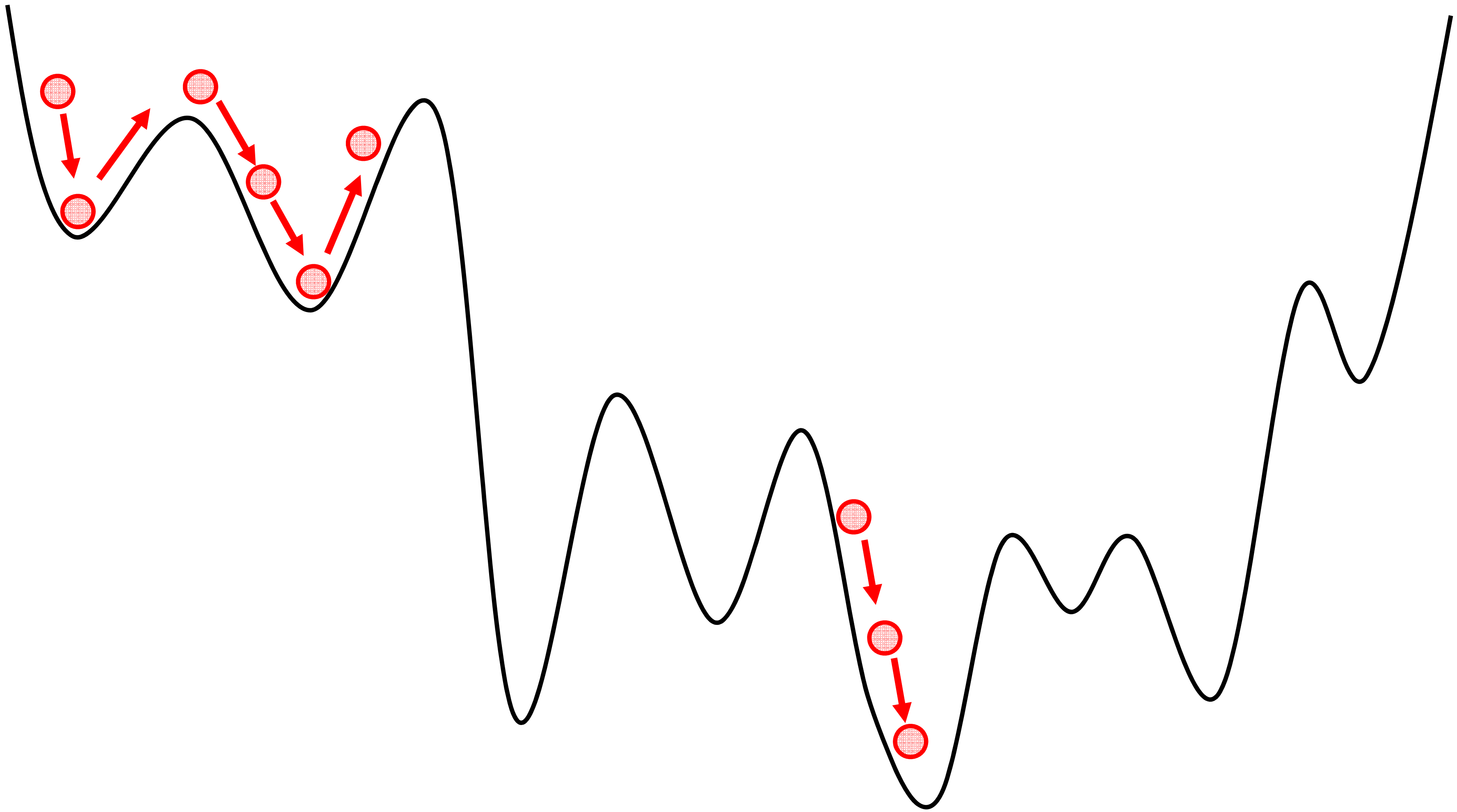


Tabu Search

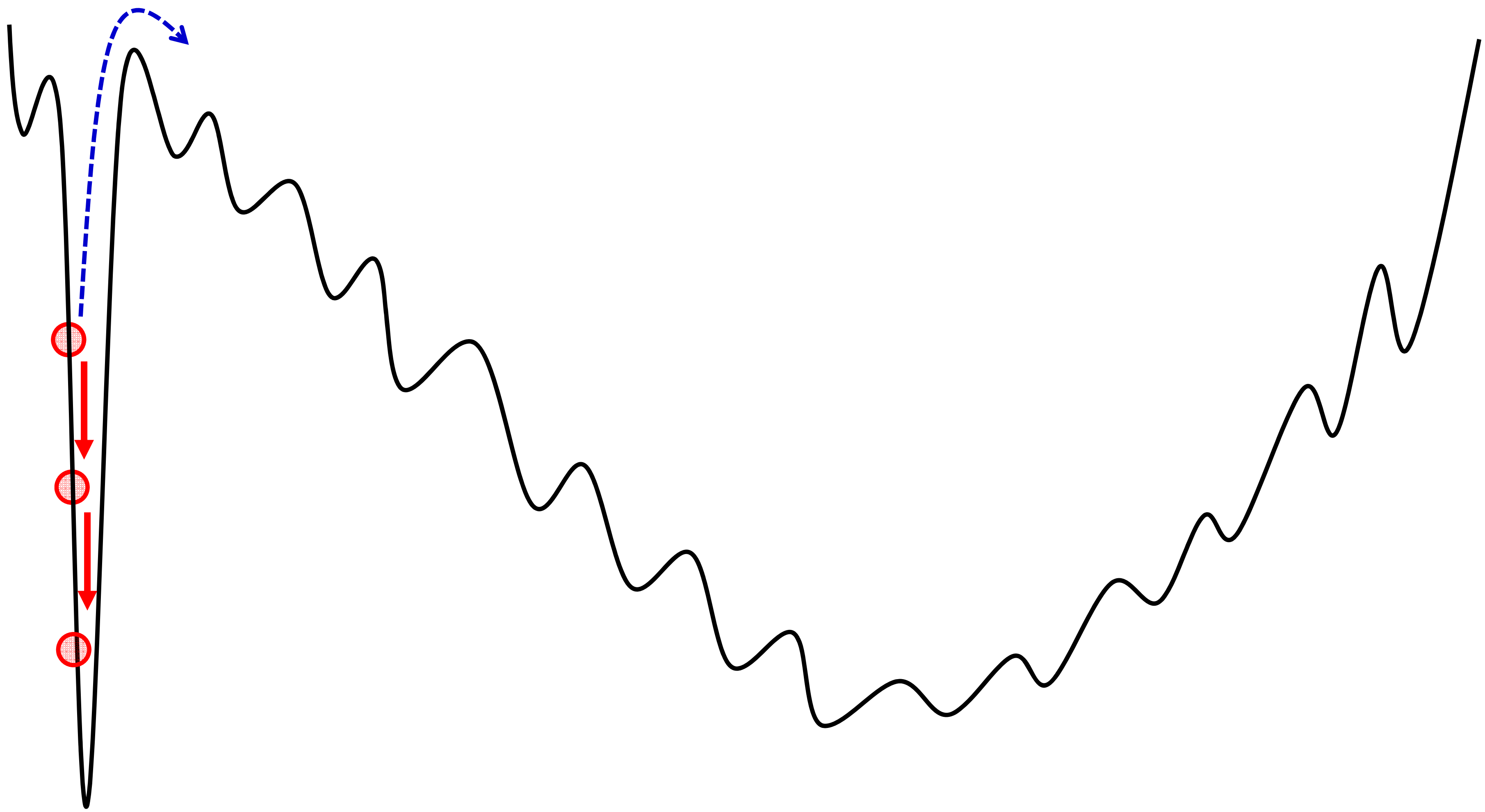
- Choice of an initial solution is important.
- It is easy to efficiently utilize a good initial solution.

- Choice of an initial solution is important.

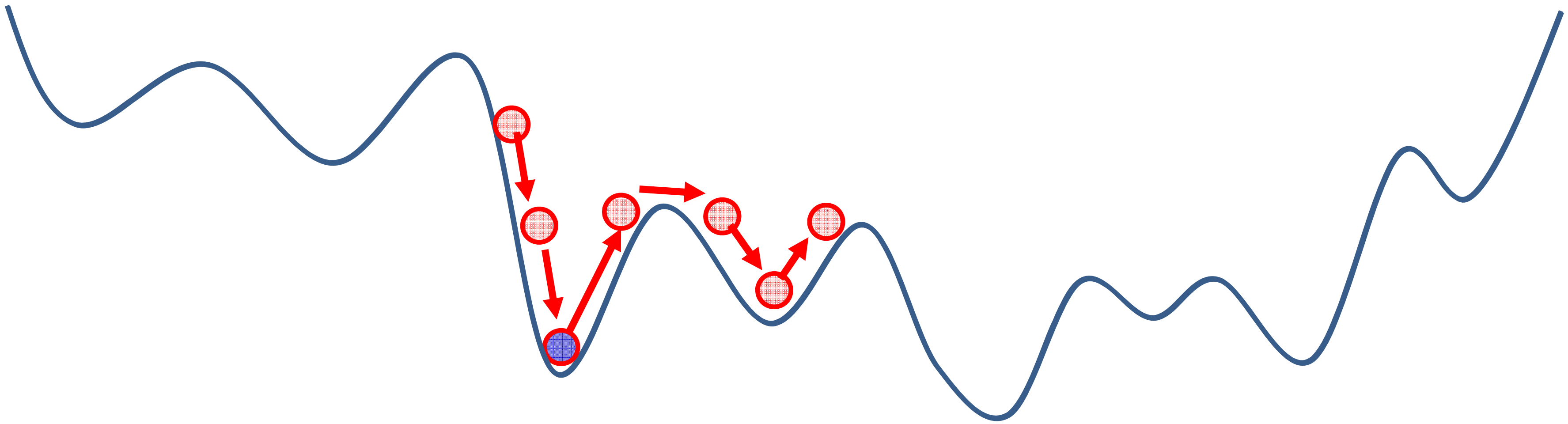
- It is easy to efficiently utilize a good initial solution.



- **Choice of an initial solution is important.**
- **It is easy to efficiently utilize a good initial solution.**



Explanation of Search behavior of Tabu Search



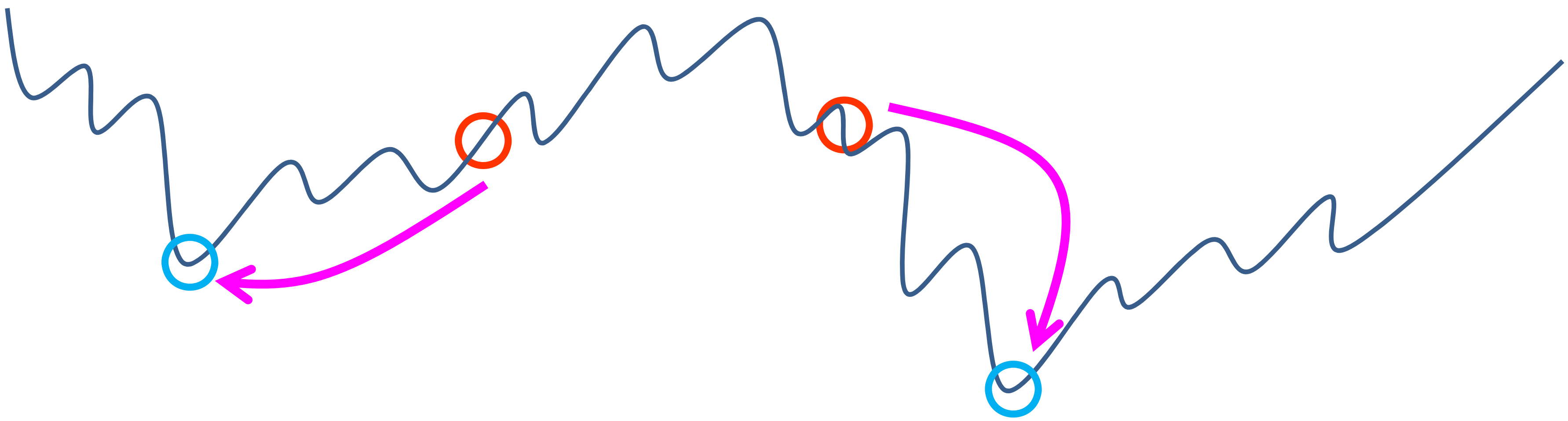
Tabu Search

- Choice of an initial solution is important.
- It is easy to efficiently utilize a good initial solution.
- Deep domain knowledge is usually needed to appropriately specify a tabu list

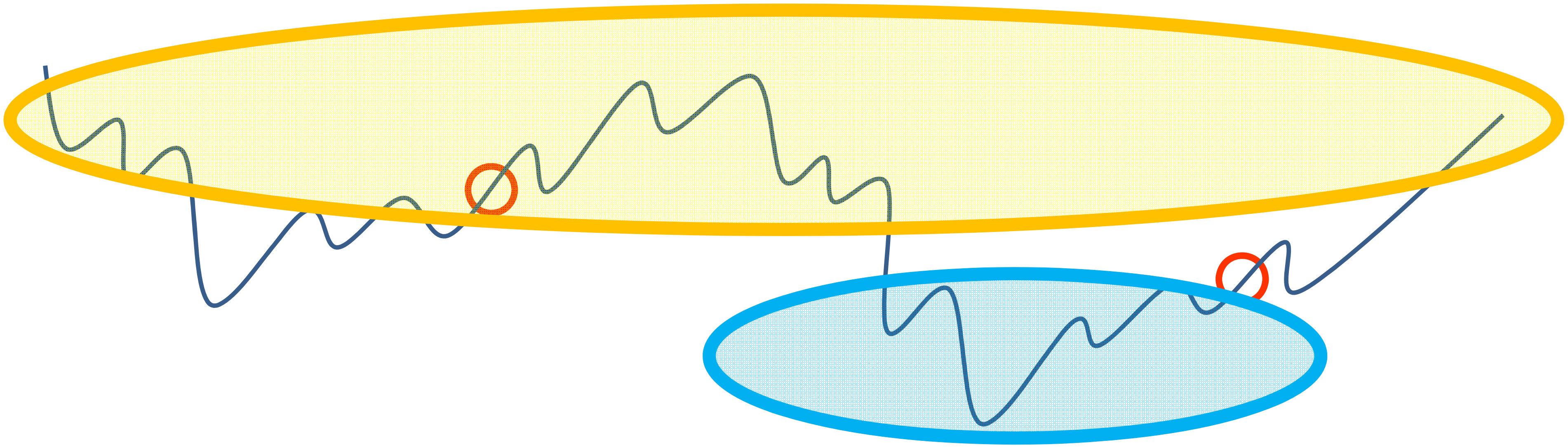
Main Advantage of Tabu Search

- **General Purpose Algorithm**
  (which is applicable to various optimization problems).
- **High Performance**
  (better than LS on problems with many local solutions).
- **Easy Implementation ???**
  Deep understanding about the optimization problem is often needed to appropriately specify a tabu list. The specification of an initial solution is also important.

Tabu Search

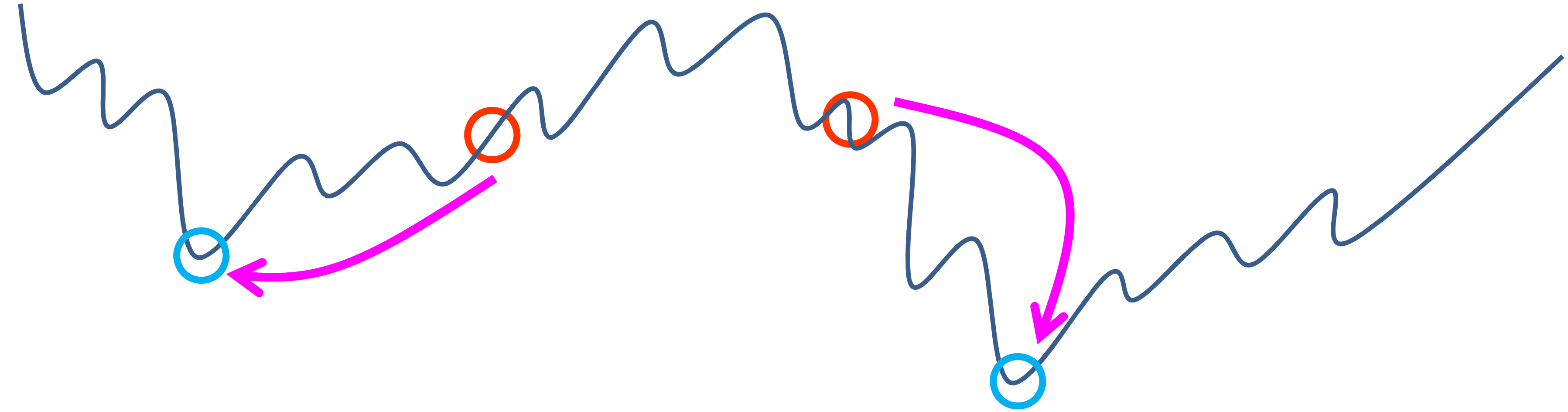


Simulated Annealing



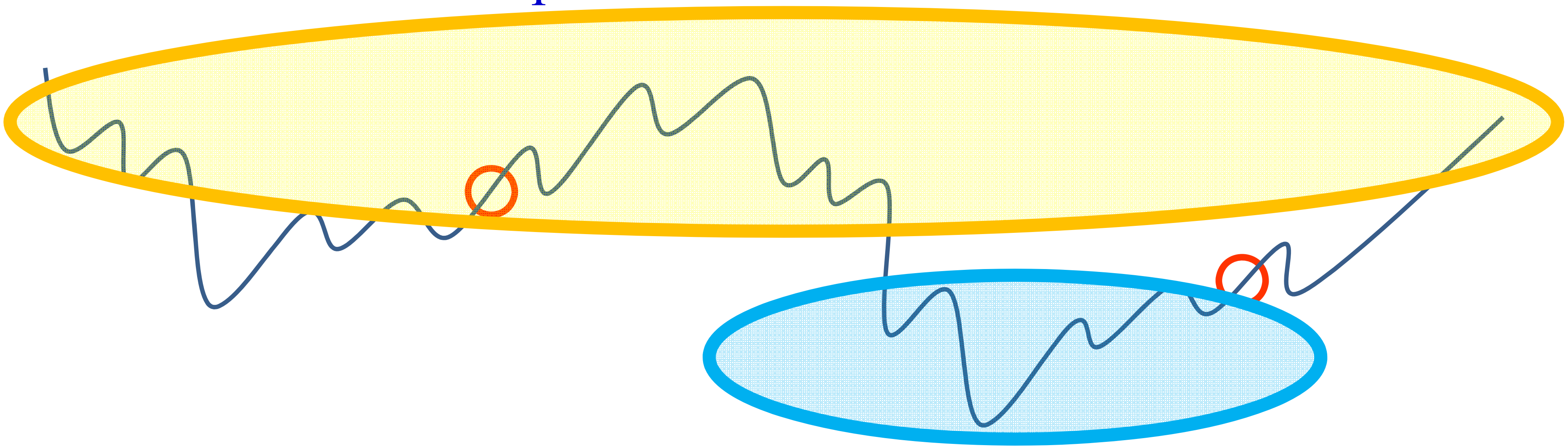
Tabu Search:

Known available computation time is not needed in advance.



Simulated Annealing:

Known available computation time is needed in advance.



Main Advantage of Tabu Search

- **General Purpose Algorithm**
  (which is applicable to various optimization problems).
- **High Performance**
  (better than LS on problems with many local solutions).
- **Easy Implementation ???**
  Deep understanding about the optimization problem is often needed to appropriately specify a tabu list. The specification of an initial solution is also important.

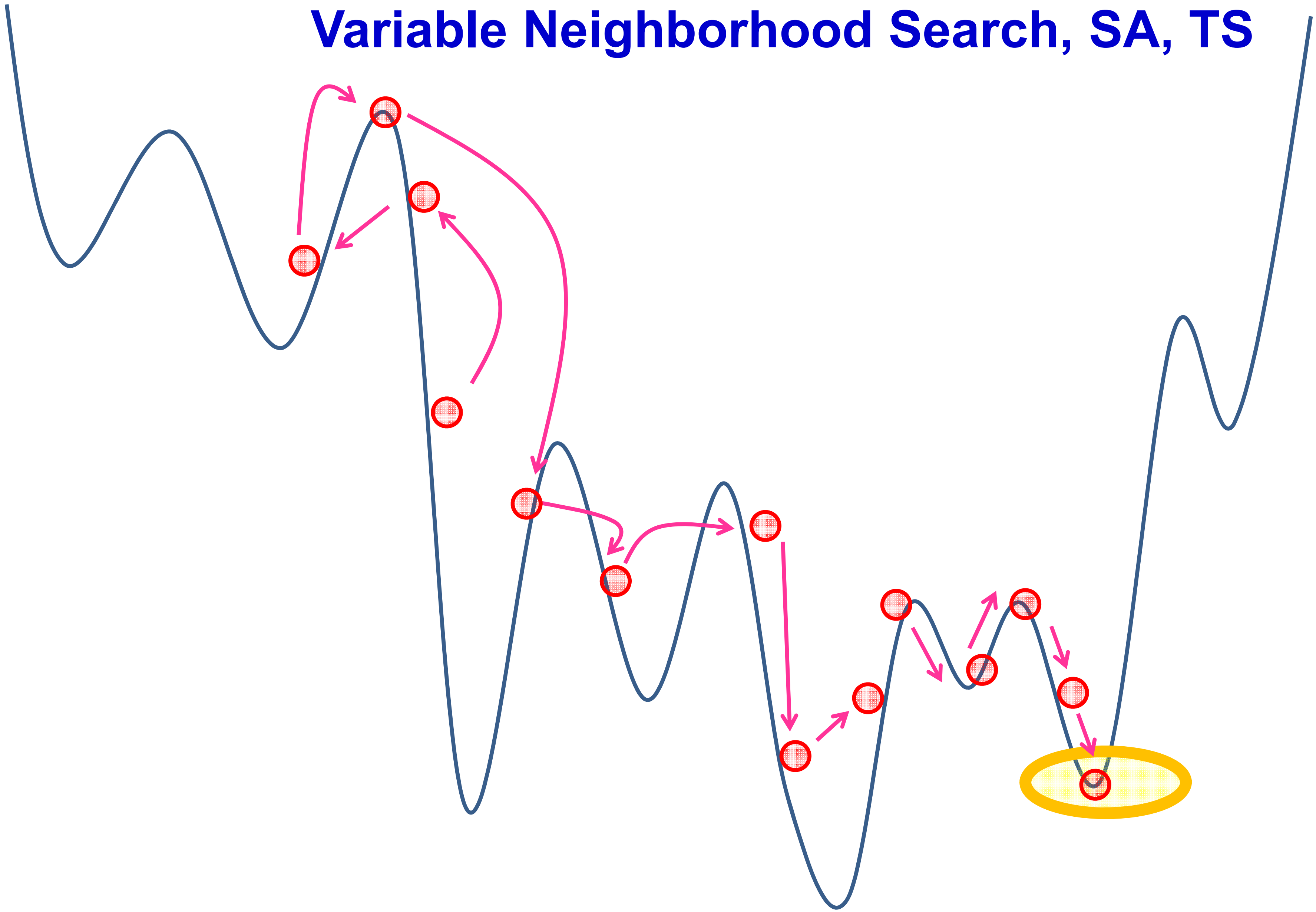
Tabu Search Paper: **About 10,000 citations**

Simulated Annealing Paper: **About 50,000 citations**

Point-based algorithms: Almost all algorithms

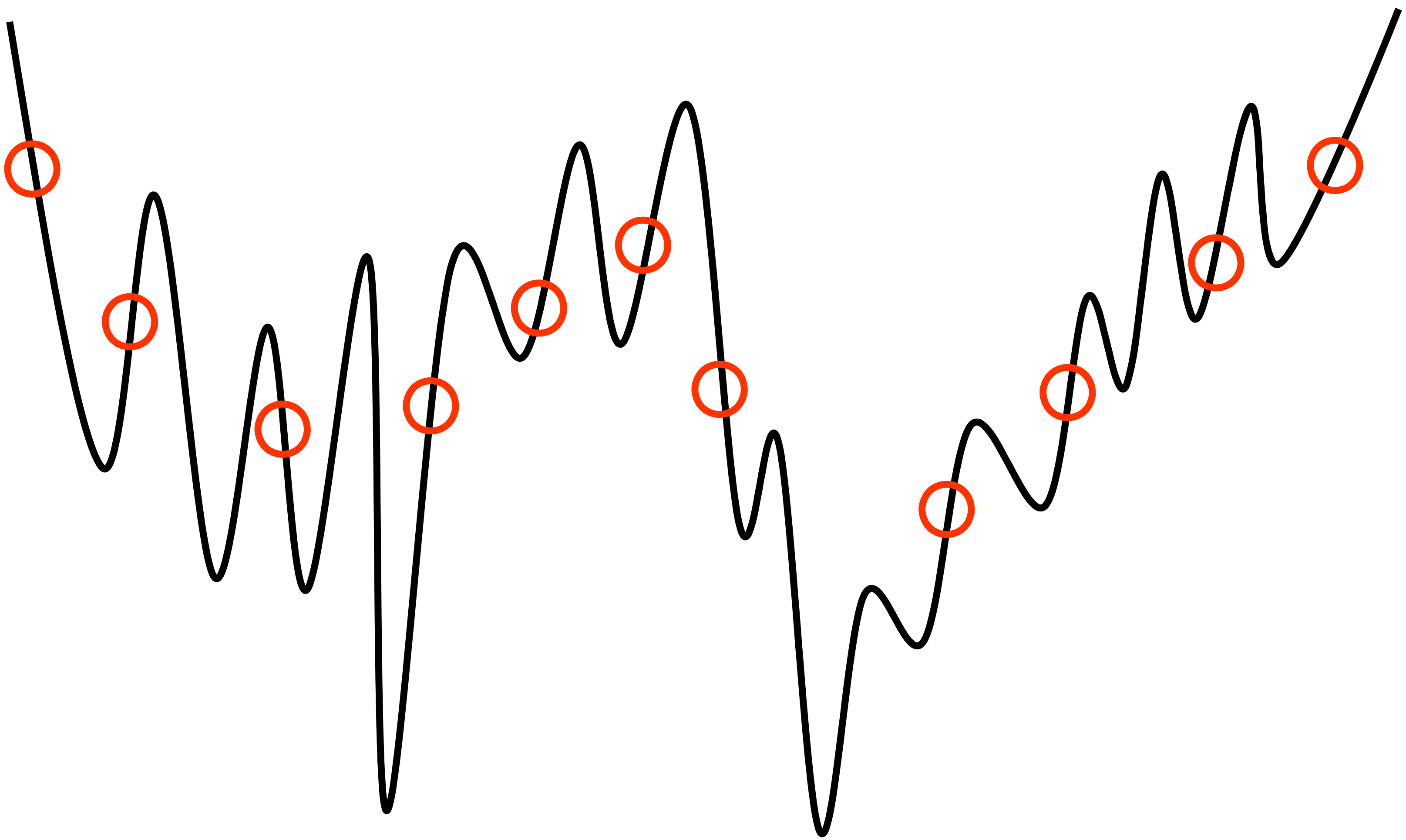
LS (Local Search), Iterated LS,

Variable Neighborhood Search, SA, TS



Genetic Algorithms: Population-based search

(multi-point search)



Lab Session Task 1:

You have a 100-item knapsack problem with 10 constraint conditions.

$$\text{Maximize } f(\boldsymbol{x}) = \sum_{i=1}^{100} v_i x_i \quad \text{Subject to } \sum_{i=1}^{100} w_{ij} x_i \leq W_j,\ j = 1, 2, ..., 10,$$

$$\text{and } x_i \in \{0, 1\},\ i = 1, 2, ..., 100$$

First we create an initial feasible solution as follows: Include items one by one in the knapsack in the order of item 1, item 2, ..., item 100. If the (k+1)-th item cannot be included in the knapsack, the initial feasible solution is created by including the1st, 2nd, ..., kth items. We define the neighbors of the current solution as follows: "Neighbors are feasible solutions with the Hamming distance D from the current solution". Three values of D are examined: D = 1, 2, 3. A termination condition is specified as 1,000 solution evaluations. Under these conditions, apply a simulated annealing algorithm to this knapsack problem with various cooling schedules (e.g., constant high temperature, constant low temperature, constant medium temperature, quick decrease of the temperature, slow decrease of the temperature).

The results are compared by drawing the following figure: