Metaheuristic Optimization Methods: Tabu Search

Origin of Tabu Search

- Fred Glover 1986: "Future paths for integer programming and links to artificial intelligence"
- Pierre Hansen 1986: "The Steepest Ascent/Mildest Descent Heuristic for Combinatorial Optimization"
- *Tabu* coined by Glover

Basic notions of Tabu Search

- The word tabu (or taboo) comes from Tongan, a language of Polynesia, where it indicates things that cannot be touched because they are sacred
- Now, it also means “a prohibition imposed by social custom”
- In Tabu Search, part of the search place will be designated as "tabu" temporarily

Tabu Search (TS)

- Similar to SA and GLS, TS allows *non-improving* moves
- SA and basic VNS rely on semi-random processes that use sampling, TS is *mostly deterministic*
  - basic TS, always select the *best improvement*
- So how to avoid cycling between solutions?
  - *tabu criteria* to create *tabu status*
  - tabu status → some solutions (and parts of solution space) are prohibited
  - this includes solutions *recently* visited

General Formulation

Tabu Search

1: *current* $\Leftarrow$ a starting solution
2: Initialize tabu memory
3: **while** stopping criterion not met **do**
4: Find a list of candidate moves, a subset of $N(current)$
5: Select the solution, s, in the candidate list that minimizes an extended cost function
6: Update tabu memory and perform the move: $current \Leftarrow s$
7: **end while**

Four dimensions of TS memory

- Short term

  recency-based memory

- Long term

  frequency-based memory

- Quality

  ability to differentiate the merit of solutions

- Influence

  impact on the choices made in search in terms of quality and structure

Recency based memory

- Recency based memory records solution **attributes** that have changed “recently”
- Selected attributes in recently visited solutions become **tabu-active** during their tabu tenures
- Solutions containing these attributes are classified as **tabu**
- Tabu solutions are excluded from $N^*(x)$ and not revisited during the **tabu tenure** (a certain period of time = certain number of moves)

Frequency Based Memory

- *Transition Measure*
  - Number of iterations when an attribute has been changed (e.g., added or deleted from a solution)
  - E.g. Number of times that element i has been moved to an earlier position in the sequence

- *Residence Measure*
  - Number of iterations where an attribute has stayed in a particular position (e.g., belonging to the current solution)
  - E.g. Number of times that element i has occupied position k

Tabu Search

- Cut off the search from parts of the search space (temporarily) through the use of *tabu criteria*
  - Note: the effective neighborhood in TS, $N^*(x)$ is a subset of a the original neighborhood $N(x)$: $\quad N^*(x) \subseteq N(x)$
  - Tabu criteria usage is based on the tabu memory structure

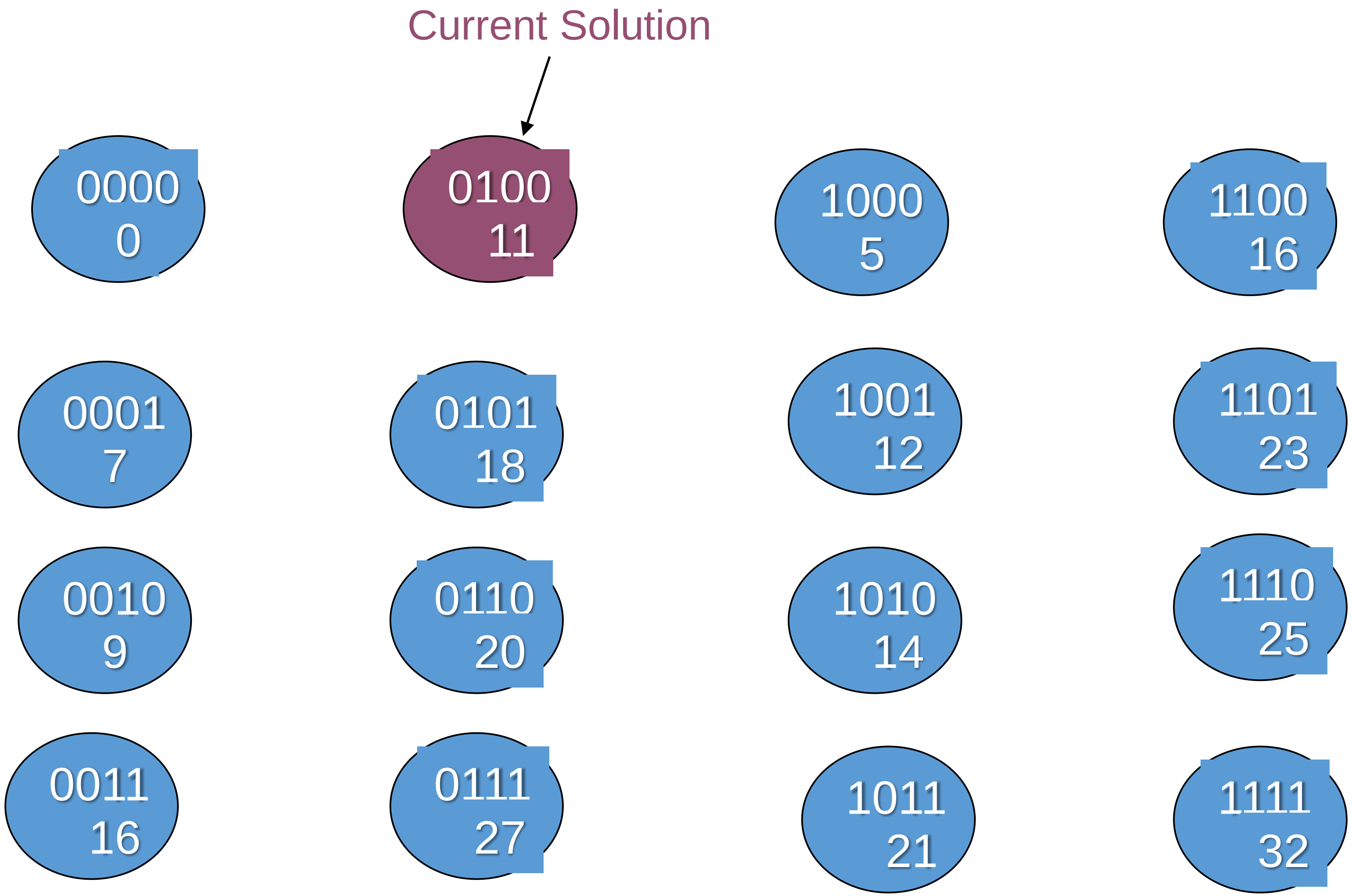
Tabu Criteria

- The *tabu criterion* is defined on selected attributes of a move, (or the resulting solution if the move is selected)
- Attributes may be *tabu active* for a certain time: *tabu tenure*
  - Can have several tabu criteria on different attributes, each with its own tabu tenure
- The tabu criterion usually avoids the immediate move reversal (or repetition)
- It also avoids the other (later) moves containing the tabu attribute. *This cuts off a much larger part of the search space.*

Example: 0/1 Knapsack

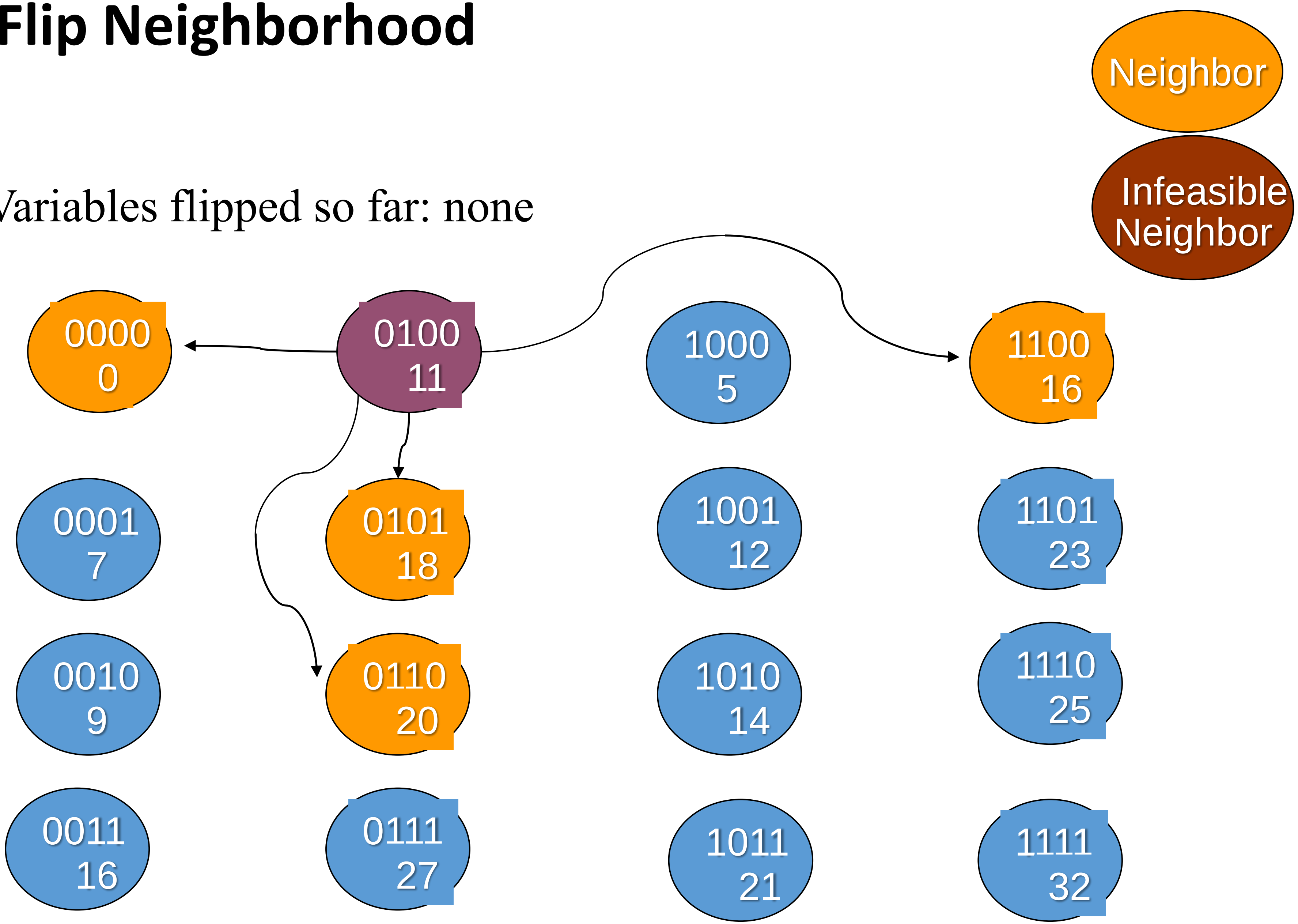
- **0/1** Knapsack; **4** items; max weight **24**
- **Flip-Neighborhood**
- **Example Tabu criteria:**
  - If the move selects an item to include in the solution, then any move trying to remove the same item is tabu for the duration of the tabu tenure
  - Similarly, an item removed is not allowed in for the duration of the tabu tenure iterations

Flip Neighborhood



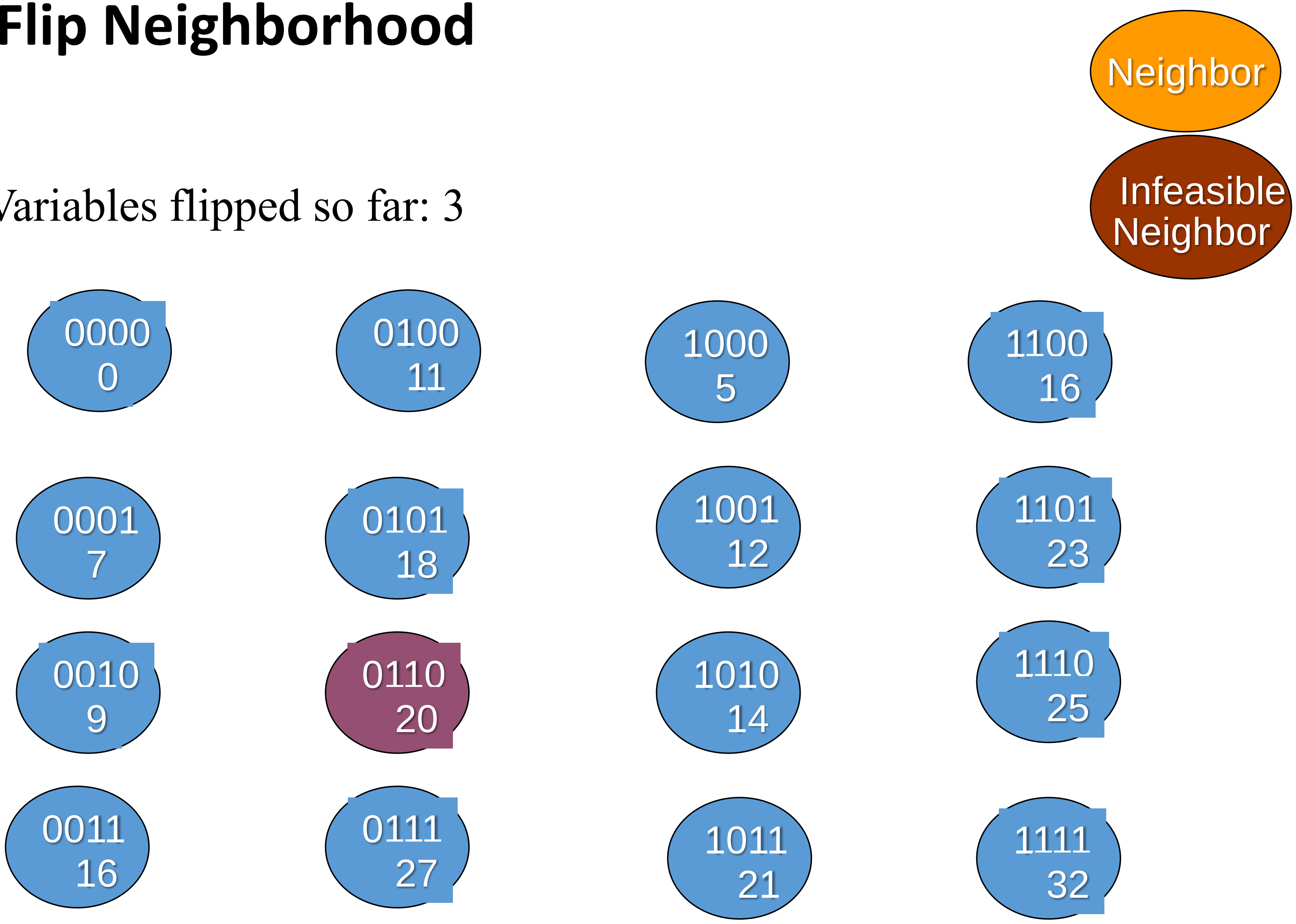
Flip Neighborhood

Variables flipped so far: none



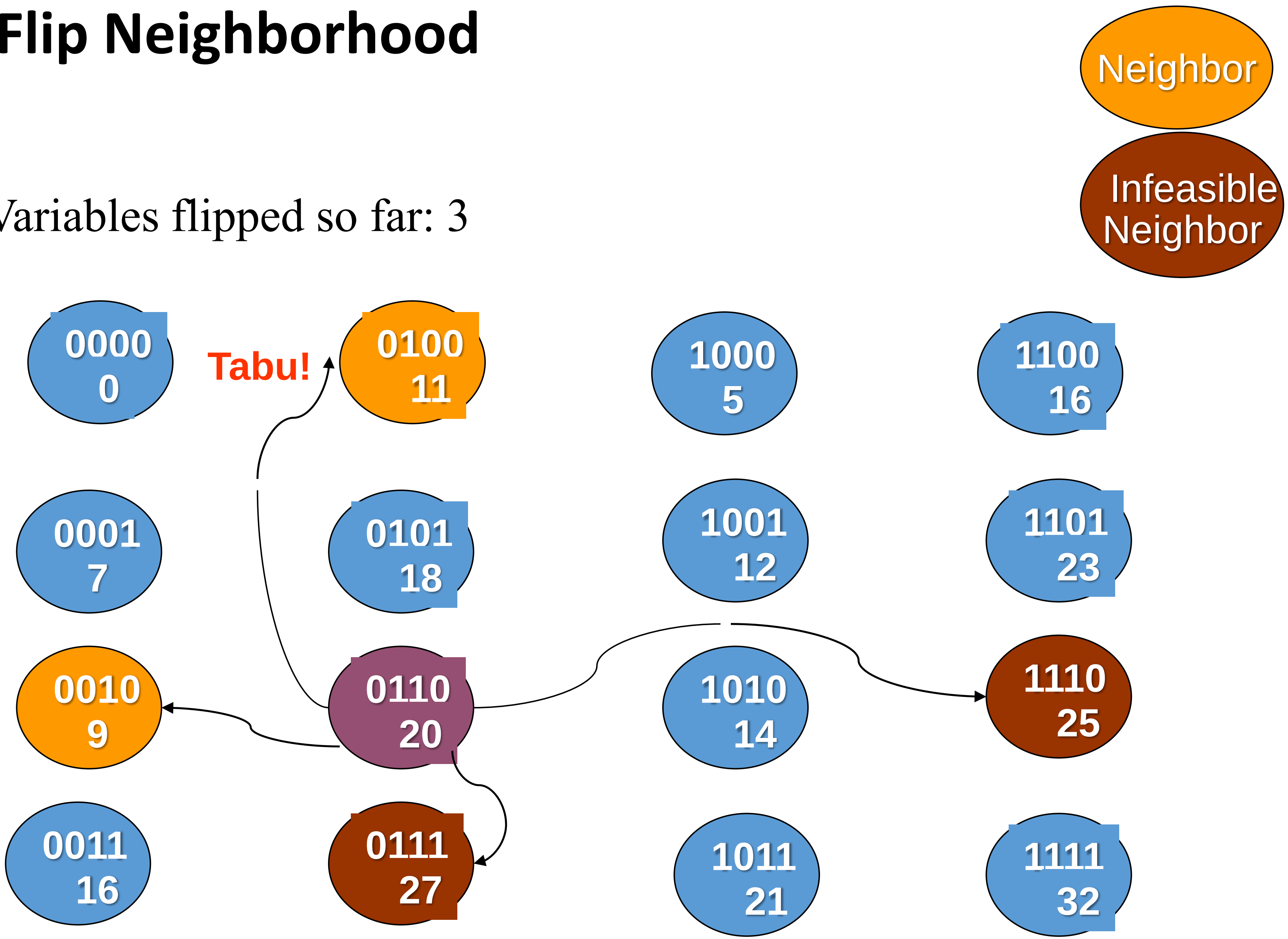
Flip Neighborhood

Variables flipped so far: 3

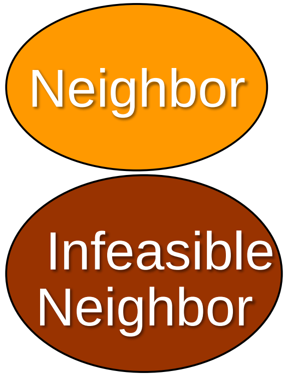


Flip Neighborhood

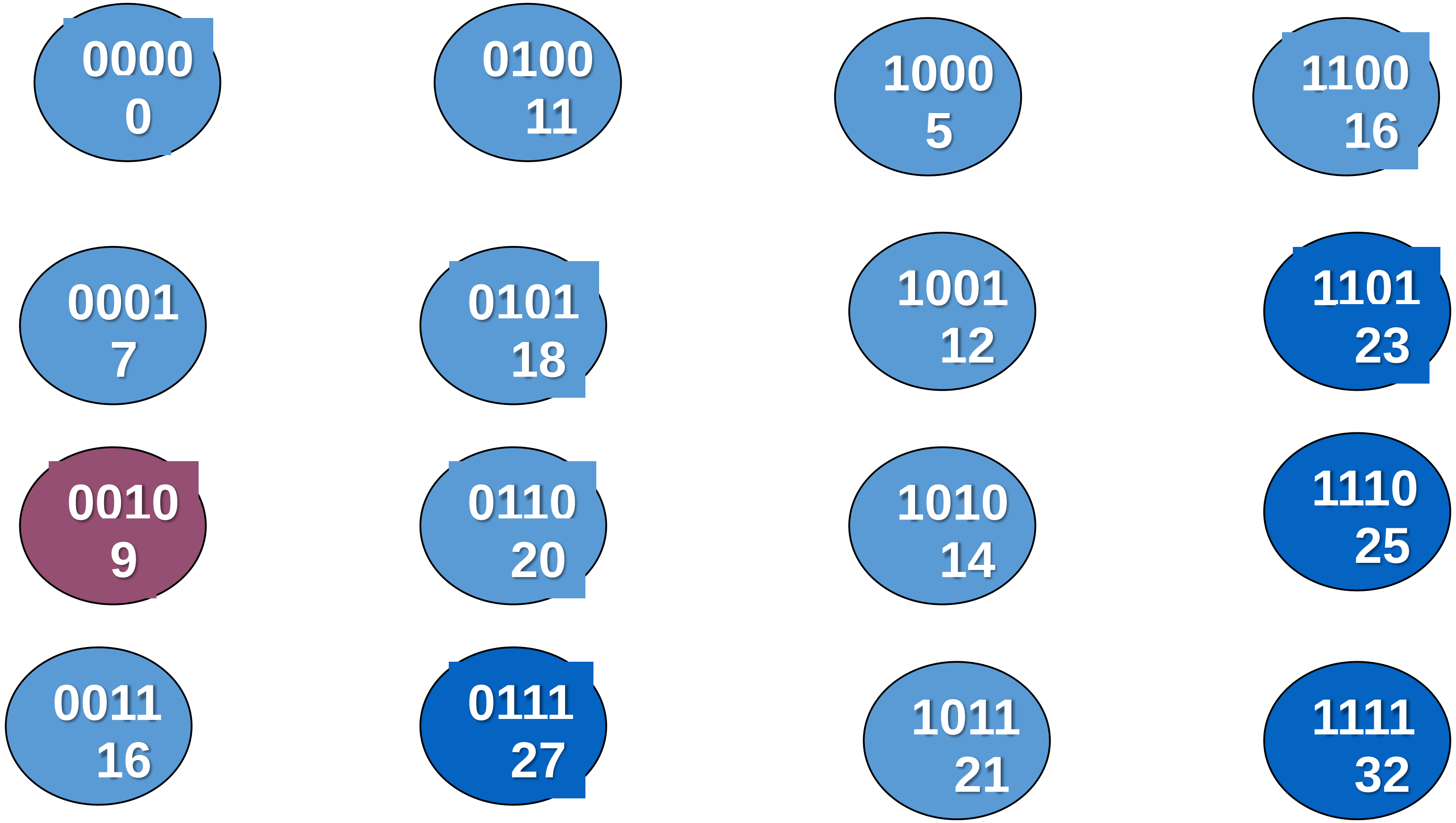
Variables flipped so far: 3



Flip Neighborhood

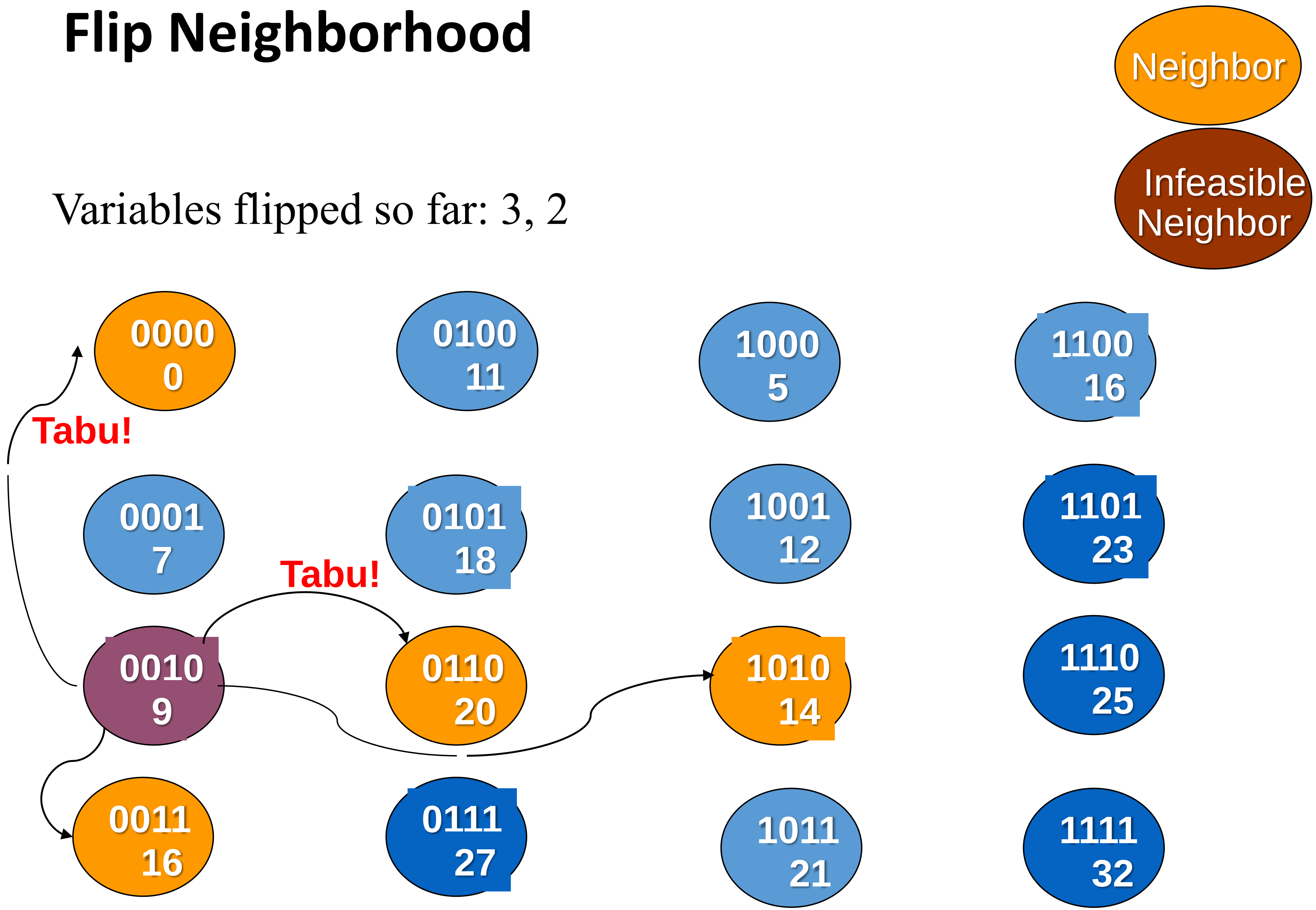


Variables flipped so far: 3, 2



Flip Neighborhood

Variables flipped so far: 3, 2



Tabu Search

- Cut off the search from parts of the search space (temporarily) through the use of *tabu criteria*
  - Note: the effective neighborhood in TS, $N^*(x)$ is a subset of a the original neighborhood $N(x)$: $\quad N^*(x) \subseteq N(x)$
  - Tabu criteria usage is based on the tabu memory structure
- Tabu status of a solution can be overruled for a preferrable alternative (aspiration criteria)

Aspiration criteria

Over-ruling the tabu status of a move

e.g., **Improved-best** or **best solution criterion**:

If a tabu solution encountered at the current iteration is better than the best solution found so far, then its tabu status is overridden

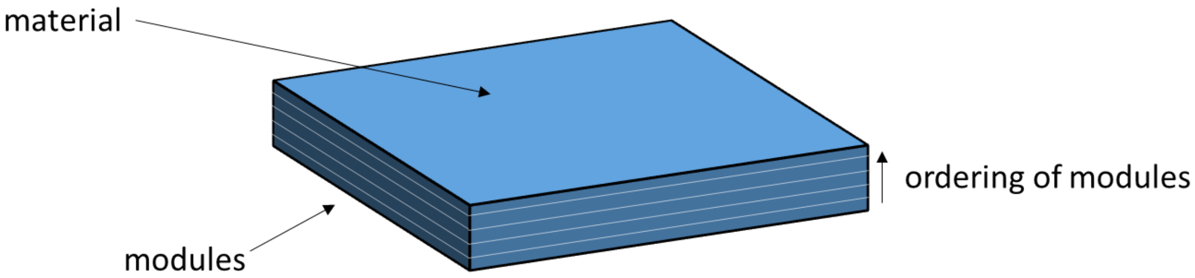
example: ordering of modules

Example of tabu search to demonstrate:

- Tabu criterion
- Short-term tabu memory structure
- Tabu tenure
- Aspiration

example: ordering of modules

Problem definition: Find the ordering of modules (filters) that maximizes the overall insulating property of the composite material



- Representation of a solution for 7 modules:

2	5	7	3	4	6	1

- Neighborhood structure: swapping modules

2	4	7	3	5	6	1

- A solution has 21 neighbors (7 choose 2)

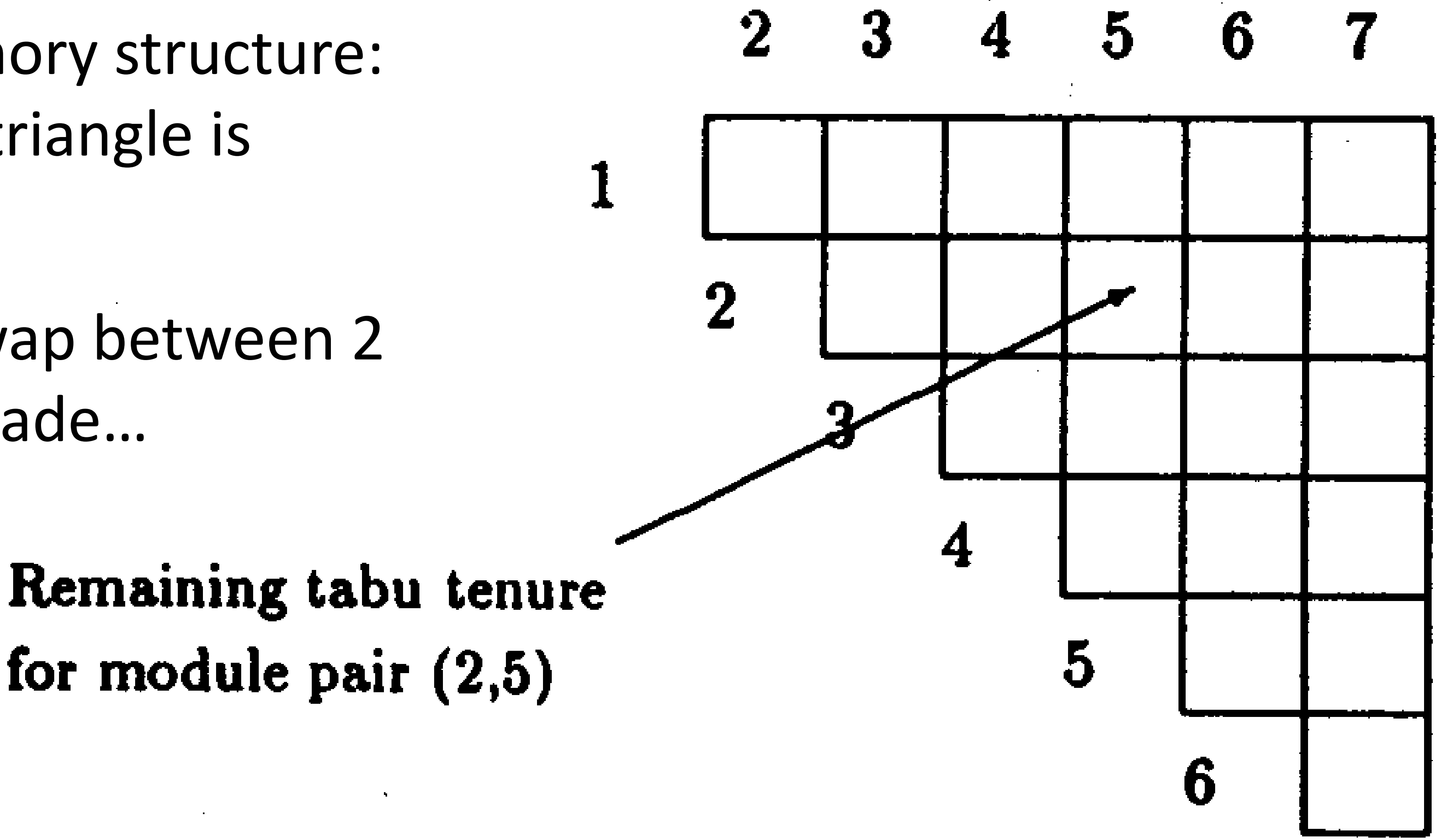
example: recency based memory and tabu classification

- Tabu attributes are selected as most recently made swaps
- Tabu tenure is set as 3 iterations
- Hence, solutions involving the 3 most recent swaps will be classified as tabu
- Aspiration criterion is chosen as best solution

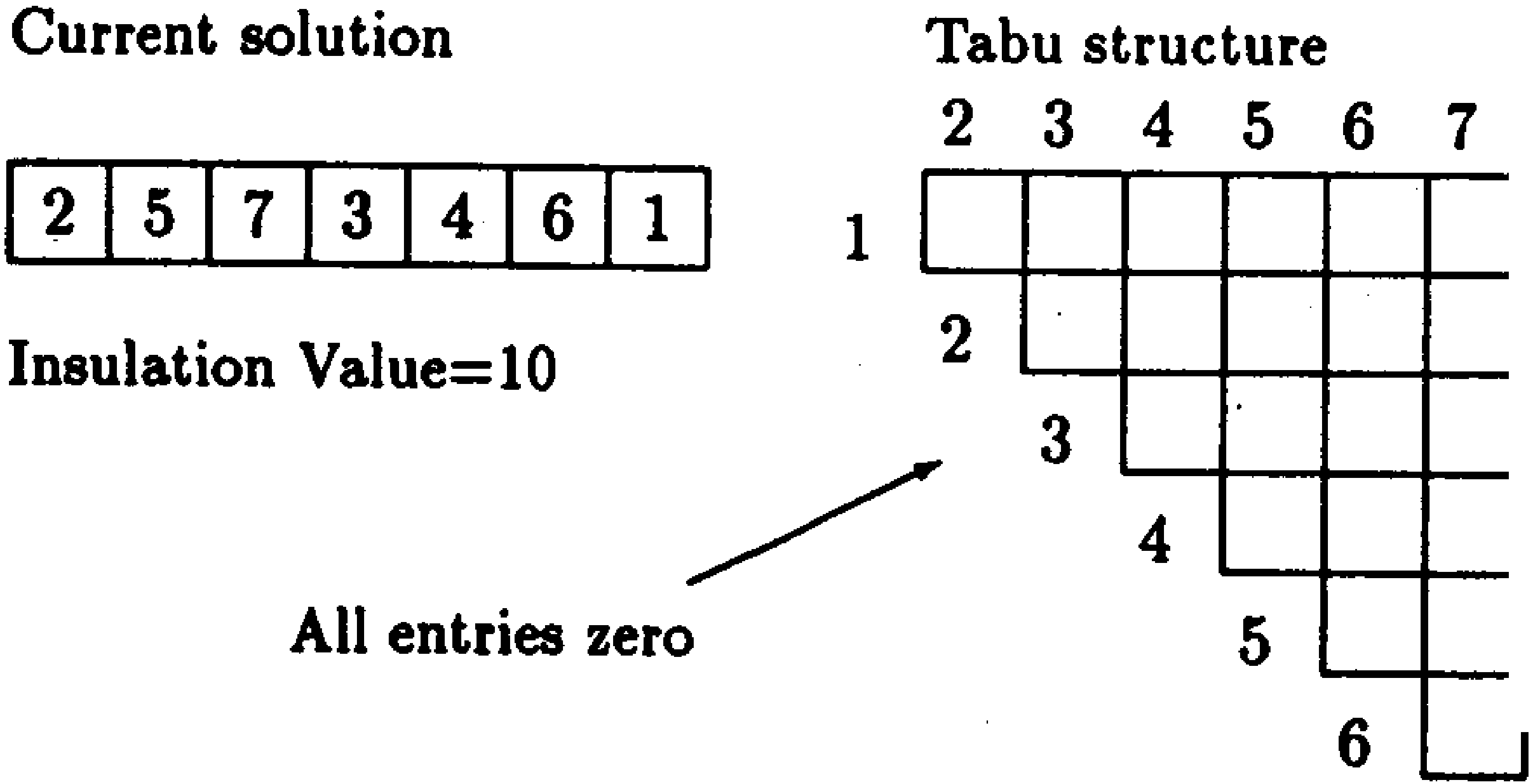
example: initialization of recency based memory

Tabu memory structure: an upper triangle is enough

e.g. if a swap between 2 and 5 is made…



Iteration 0 (*Starting point*)



Top 5 candidates constitute the candidate list

"Value" is the gain of swap

Iteration 1

Current solution

2	4	7	3	5	6	1

Insulation Value=16

Tabu structure

	2	3	4	5	6	7
1						
2						
3						
4				3		
5						
6						

Move (4,5) has now a tabu tenure of 3 iterations

Iteration 2

Current solution

2	4	7	1	5	6	3

Insulation Value=18

Tabu structure

	2	3	4	5	6	7
1		3				
2						
3						
4				2		
5						
6						

Moves (1,3) and (4,5) have respective tabu tenures 3 and 2

No move with a positive gain, hence best (non-tabu) move will be non-improving

Iteration 3

Current solution

4	2	7	1	5	6	3

Insulation Value=14

Tabu structure

	2	3	4	5	6	7
1		2				
2			3			
3						
4				1		
5						
6						

Top 5 candidates

Swap	Value	
4,5	6	T*
5,3	2	
7,1	0	
1,3	-3	T
2,6	-6	

Move (4,5) has a tabu tenure of 1 iteration

But this move results in the best solution so far

Hence its tabu status is overridden

Iteration 4

Current solution

5	2	7	1	4	6	3

Insulation Value=20

Tabu structure

	2	3	4	5	6	7
1		1				
2			2			
3						
4				3		
5						
6						

Top 5 candidates

Swap	Value	
7,1	0	*
4,3	-3	
6,3	-5	
5,4	-6	T
2,6	-8	

Aspiration Criterion

- Simplest: Allow new best solutions, otherwise keep tabu status
- Other criteria based on:
  - Degree of feasibility
  - Degree of change
  - Feasibility level vs. Objective function value
- If all moves are tabu:
  - Choose the best move, or choose randomly (in the candidate list)

Tabu Attribute Selection

- Attribute
  - A property of a solution or a move
- Can be based on any aspect of the solution that are changed by a move
- Attributes are the basis for tabu restrictions
- A move can change more than one attribute
  - e.g. a 2-opt move in TSP involves 4 cities and 4 edges

Example

After a move that changes the value of x_i from 0 to 1, we would like to prevent x_i from taking the value of 0 in the next tabu tenure iterations

Attribute to record: *i*

Tabu activation rule:

move ($x_i \leftarrow 0$) is tabu if *i* is tabu-active

Example

After a move that exchanges the positions of element *i* and *j* in a sequence, we would like to prevent elements *i* and *j* from exchanging positions in the next tabu tenure iterations

<u>Attributes to record: i and j</u>

<u>Tabu activation rule:</u>
<u>move (i $\leftrightarrow$ j) is tabu if both i and j are tabu-active</u>

Example

After a move that drops element *i* from and adds element *j* to the current solution, we would like to prevent element *i* from being added to the solution in the next *tabu add tenure* iterations and prevent element *j* from being dropped from the solution in the next *tabu drop tenure* iterations

Attributes to record: *i* and *j*

Tabu activation rules:

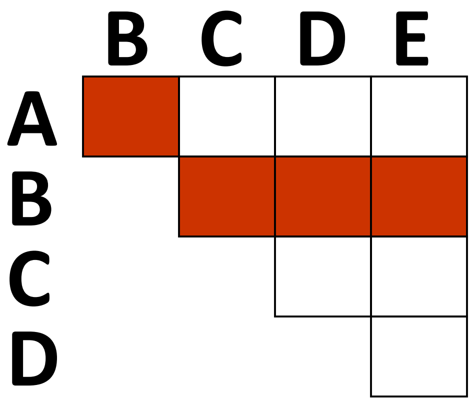
move (Add *i*) is tabu if *i* is tabu-active

move (Drop *j*) is tabu if *j* is tabu-active

Example

Elements	A	D	C	B	E
Positions	1	2	3	4	5

<u>Tabu activation rule</u>: move (B $\leftrightarrow$ *) is tabu



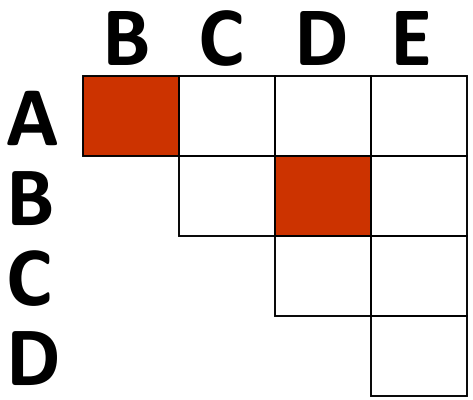
Tabu move

Example

Elements	A	D	C	B	E
Positions	1	2	3	4	5

<u>Tabu activation rule</u>:

move (B ↔ *) is tabu if B moves to 2 or earlier

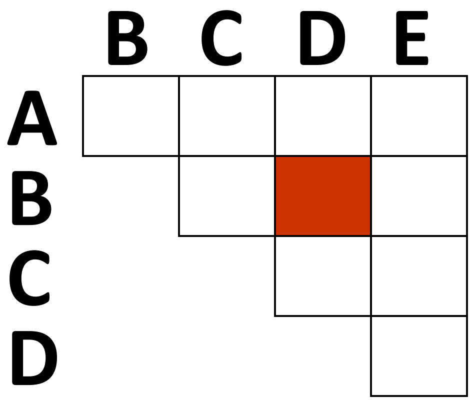


 Tabu move

Example

Elements	A	D	C	B	E
Positions	1	2	3	4	5

<u>Tabu activation rule</u>: move (B $\leftrightarrow$ D) is tabu



Tabu move

Tabu Tenure Management

- **Static Memory**
  - The value of *tabu tenure* is fixed and remains fixed during the entire search
  - All attributes remain tabu-active for the same number of iterations

- Dynamic Memory
  - The value of ***tabu tenure*** is not constant during the search
  - The length of the tabu-active status of attributes varies during the search

➢ Experience shows that dynamic rules are more robust than static rules

➢ Attribute-dependent dynamic rules have proved effective for difficult problems (scheduling and routing)

Tabu Tenure Management

Dynamic tabu tenure

- Change the tabu tenure at certain intervals
- Can use *uniform random selection* in [MinTenure, MaxTenure]

Note: The values of MinTenure and MaxTenure are search parameters

Dynamic tabu tenure allows for:

Reactive Tabu Search Battiti and Tecchiolli (1994)

- Detect stagnation → increase tabu tenure
- When escaped → reduce tabu tenure

Tabu Search

- Cut off the search from parts of the search space (temporarily) through the use of *tabu criteria*
  - Note: the effective neighborhood in TS, $N^*(x)$ is a subset of a the original neighborhood $N(x)$: $\quad N^*(x) \subseteq N(x)$
  - Tabu criteria usage is based on the tabu memory structure
- Tabu status of a solution can be overruled for a preferrable alternative (aspiration criteria)
- Using memory, guide the search towards other parts of the search by using penalties and bonuses (similar to GLS)

TS - Diversification

- Basic Tabu Search often gets stuck in one area of the search space
- Diversification is trying to get to somewhere else
- Historically random restarts have been very popular
- Frequency-based diversification tries to be more clever
    - penalize elements of the solution that have appeared in many other solutions visited

TS - Intensification

- Intensification is to aggressively prioritize good solution attributes in a new solution
- Usually based on frequency
- Can be based on elite solutions

Intensification and Diversification: Modifying Choice Rules based on Frequency

- Move modification is based on penalty functions

  - E.g.: Rule
    - Choose the move with the best move value if at least one admissible improving move exists
    - Otherwise, choose the admissible move with the best modified move value

modified move value = move value + rule parameter * f(frequency)

Carrots and Sticks

Diversification:

- Moves containing attributes with a high frequency count are penalized
- TSP-example: $g(x) = f(x) + w_1 \Sigma \omega_{ij}$

Intensification:

- Moves to solutions containing attributes with a high frequency among the elite solutions are encouraged
- TSP-example: $g(x) = f(x) - w_2 \Sigma \gamma_{ij}$

Strategic Oscillation

- Move until hitting a boundary, e.g. infeasibility
- Instead of stopping there, extend neighborhood definition or modify evaluation criteria to permit crossing the boundary
- Proceed for a specified depth beyond the boundary, then turn around and cross the boundary in reverse direction

Example: Strategic Oscillation

- Knapsack problem:
  - a TS may be designed to allow variables to be set to 1 even after reaching the feasibility boundary
  - After a selected number of steps, the direction is reversed by choosing moves that change variables from 1 to 0
  - “Critical Event Tabu Search for Multidimensional Knapsack Problems” Glover and Kochenberger (1996)

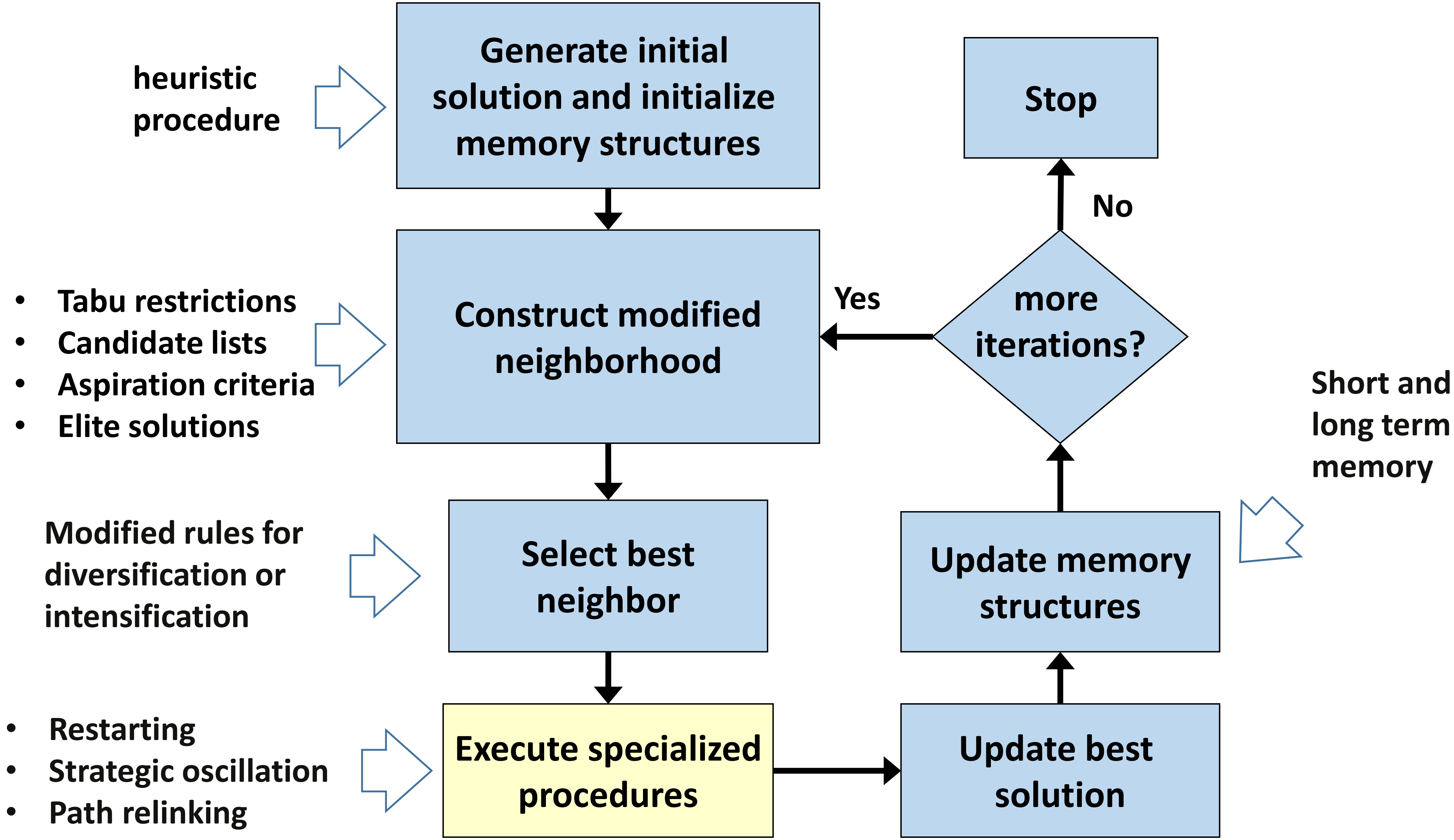
Exploiting Infeasible Solutions

- Why should the search be allowed to visit infeasible solutions?
- The feasible space can be fragmented
  - Given a neighborhood operator: there is no path between feasible solutions S_A and S_B
- There are infeasible solutions for which the optimal solution is a neighbor
  - Knapsack: removing one item can change infeasible to feasible

Table I. Guidance by Strategic Oscillation

Application	Element Controlled	Reference
Quadratic Assignment	Tabu Restrictions	Battiti and Tecchiolli[2]
Transportation	Infeasibility Penalties	Cao and Uebe[3]
Operation Timetables	Penalty Measures	Costa[4]
Multidimensional Knapsack	Infeasibility Depth	Freville and Plateau[11]
Vehicle Routing	Nodes and Infeasibility	Gendreau, Hertz and Laporte[12]
Employee Scheduling	Number of Employees	Glover and McMillan[21]
Network Design	Platform (node) assignment	Glover, Lee and Ryan[20]
Data Integrity	Verification Infeasibility	Kelly, Golden and Assad[32]
Multilevel GAP	Assignments/Infeasibility	Laguna et al.[32]
Classroom Scheduling	Assignments/Infeasibility	Mooney and Rardin[38]
Vehicle Routing	Objective Function	Osman[43]
Capacitated Clustering	Objective Function	Osman and Christofides[44]
Mixed Fleet VRP	Vehicles and Infeasibility	Osman and Salhi[45]
Delivery Systems	Penalty Measures	Rochat and Semet[47]
Graph Partitioning	Partitioned Nodes	Rolland, Pirkul and Glover[48]
Time Deadline VRP	Infeasibility	Thangiah et al.[54]
Graph Design	Objective Function	Verdejo, Cunquero and Sarli[55]
P-Median Problem	Values of P	Voss[56]
Traveling Purchaser Problem	Markets	Voss[57]

Tabu Search Framework



Path Relinking

- Seminal ideas originated in connection with tabu search…
  - Glover, F. and M. Laguna (1993) “Tabu Search,” in Modern Heuristic Techniques for Combinatorial Problems, C. Reeves (ed.) *Blackwell Scientific Publications*, pp. 70-150.
- General version of "Scatter Search"
- Modern versions have been applied as a combination method within scatter search and in the improvement phase of GRASP

Path Relinking

- This approach generates new solutions by exploring trajectories that connect elite solutions

- The exploration starts from an **initiating solution** and *generates a path in the neighborhood space* that leads to a **guiding solution**

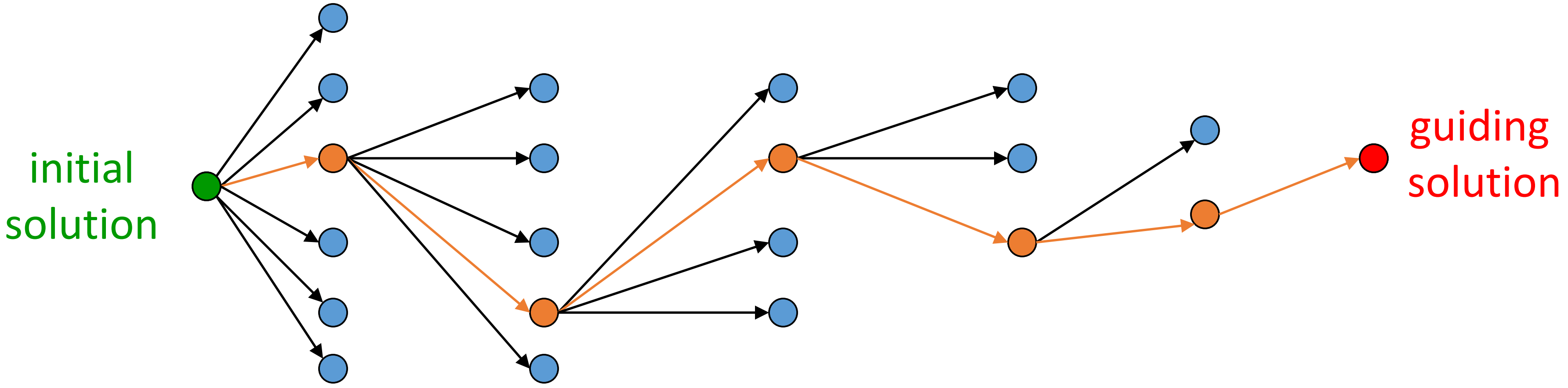
- Choice rules are designed to incorporate attributes contained in the guiding solution

Path-relinking

- Path is generated by selecting moves that introduce in the initial solution attributes of the guiding solution.
- At each step, all moves that incorporate attributes of the guiding solution are evaluated and the best move is selected:

Path-relinking

- Path is generated by selecting moves that introduce in the initial solution attributes of the guiding solution.
- At each step, all moves that incorporate attributes of the guiding solution are evaluated and the best move is selected:



Path-relinking

Goal: Create a path betwen solutions x and y

$\Delta(x,y)$: symmetric difference between x and y

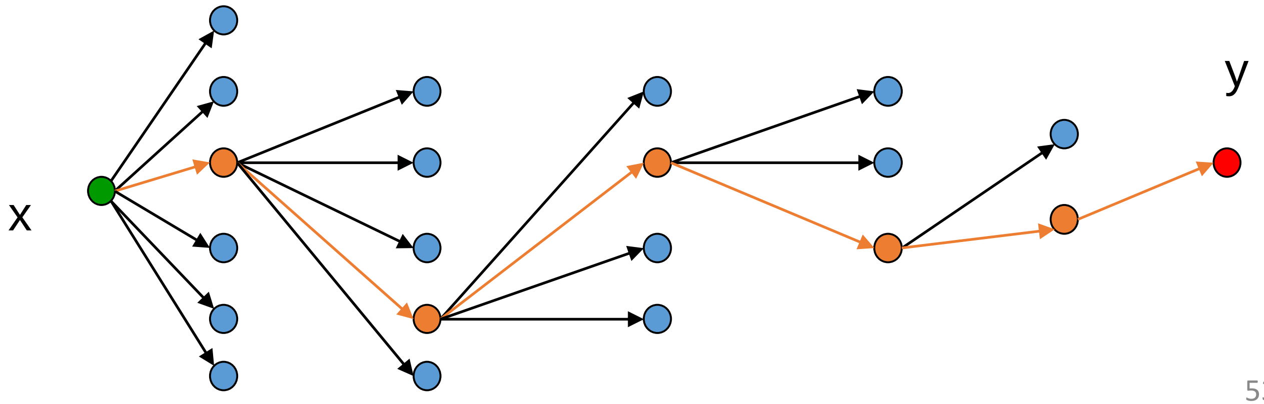
while ($|\Delta(x,y)| > 0$) {

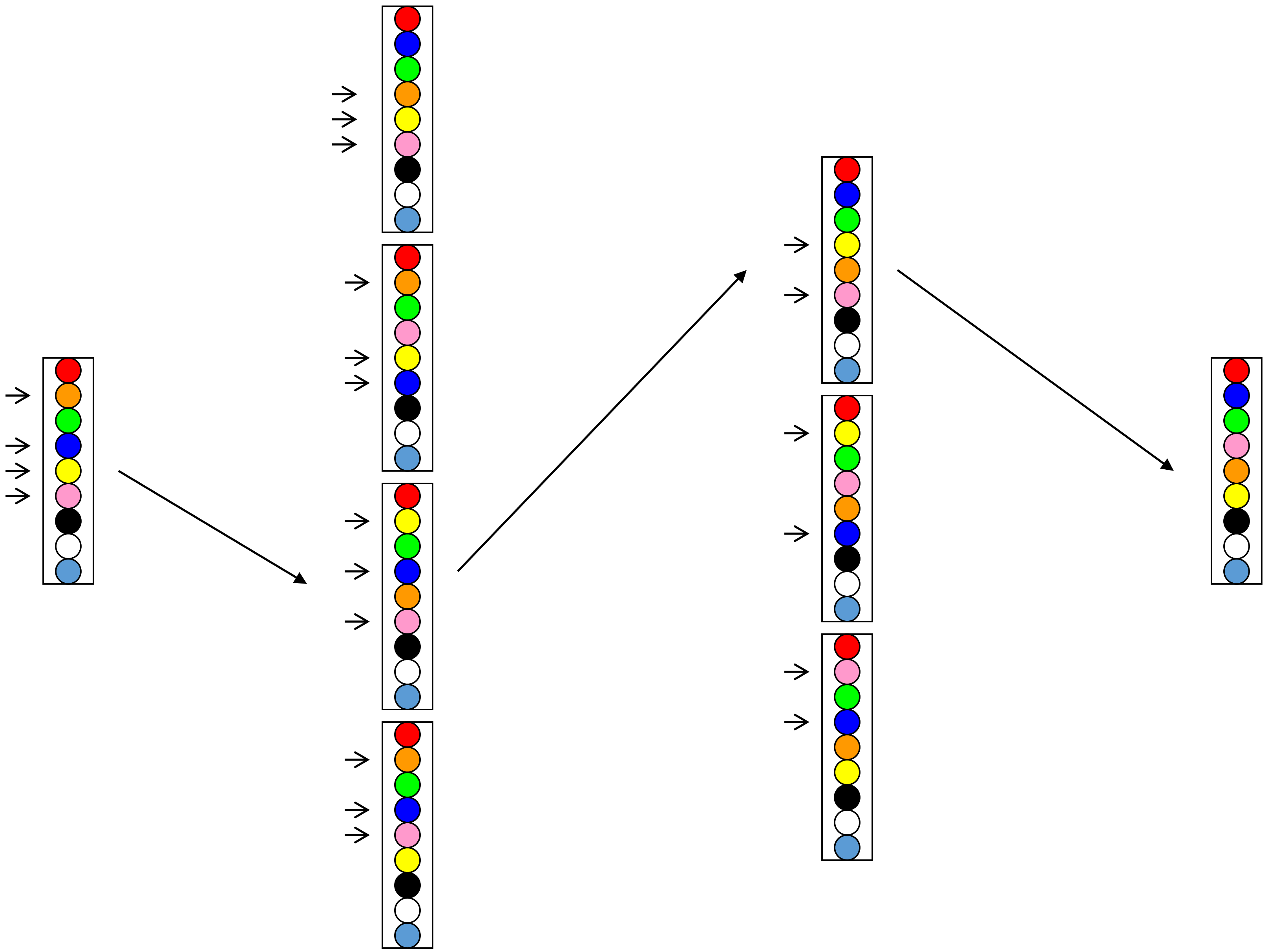
-evaluate moves corresponding in $\Delta(x,y)$

-make best move

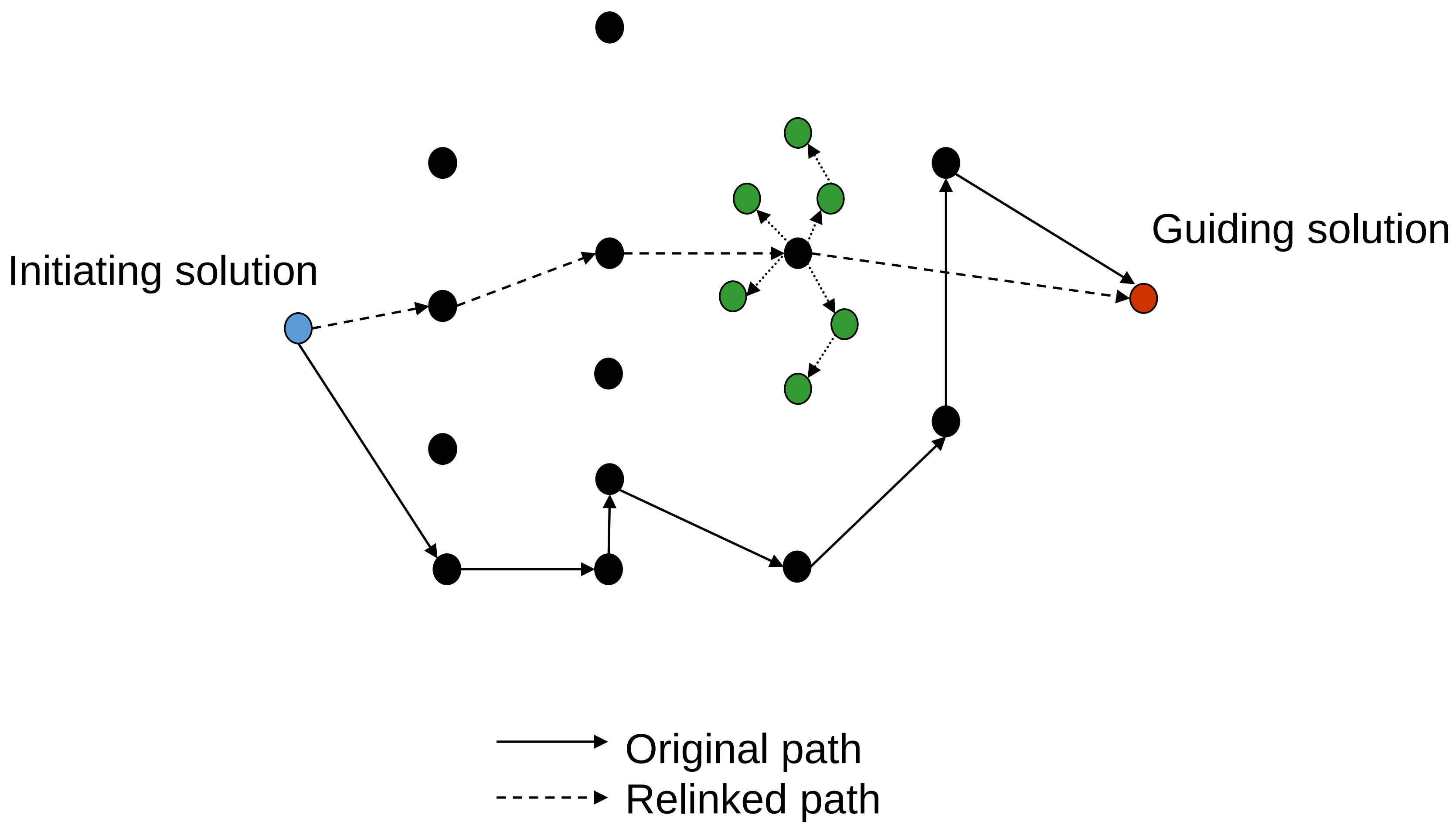
-update $\Delta(x,y)$

}





Relinking Solutions



Relinking Strategies

- Forward relinking: worst solution is the initiating solution
- Backward relinking: best solution is the initiating solution
- Backward and forward relinking: both directions are explored
- Mixed relinking: relinking starts at both ends
- Tunneling: mixed relinking but occasionally allow infeasible moves
- Randomized relinking: stochastic selection of moves
- Truncated relinking: the guiding solution is not reached
- Extrapolated relinking: do not stop when you reach but go beyond it for further diversification