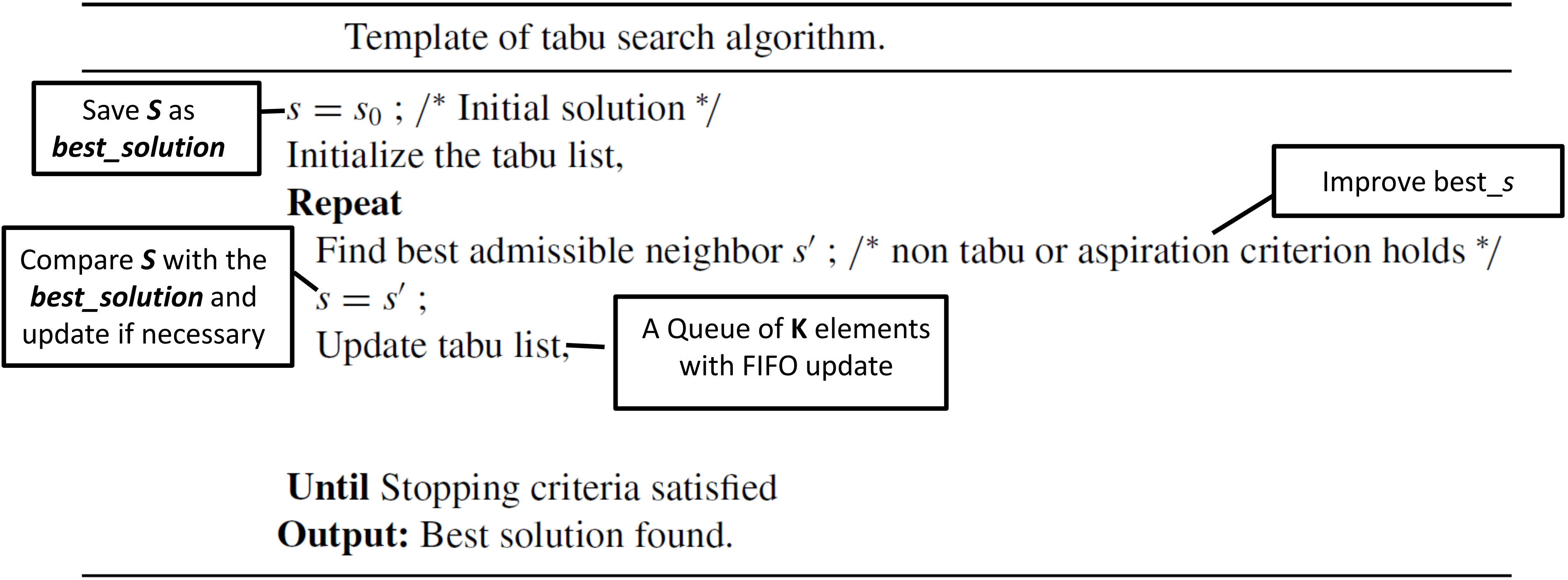
2.3 S-metaheuristics
Tabu Search (TS)

S-metaheuristics – Tabu Search Algorithm (TS)

- Main characteristics
  - Use of memory to store information related to the search process
  - Accept non improving solutions to escape from local optima

S-metaheuristics – Tabu Search Algorithm (TS)



Template of tabu search algorithm.

```
$s = s_0$ ; /* Initial solution */
Initialize the tabu list,
Repeat
  Find best admissible neighbor $s'$ ; /* non tabu or aspiration criterion holds */
  $s = s'$ ;
  Update tabu list,

Until Stopping criteria satisfied
Output: Best solution found.
```

Save ***S*** as ***best_solution***

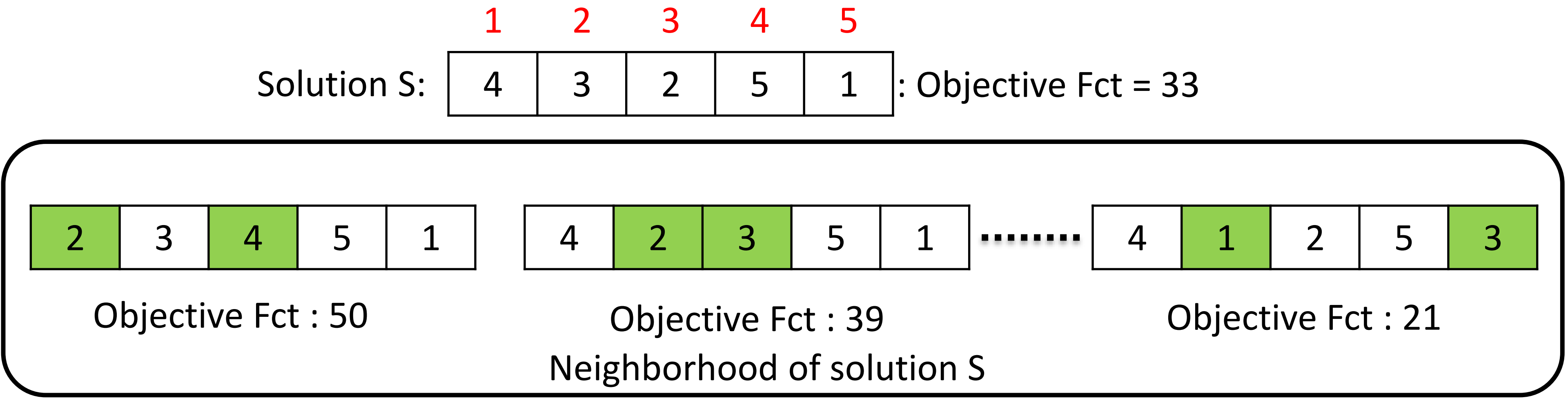
Improve best_*s*

Compare ***S*** with the ***best_solution*** and update if necessary

A Queue of **K** elements with FIFO update

S-metaheuristics – Tabu Search Algorithm (TS)

- Best admissible solution (TSP prob): Scenario 1



Solution S:

1	2	3	4	5
4	3	2	5	1

: Objective Fct = 33

🟩 2	3	🟩 4	5	1

Objective Fct : 50

4	🟩 2	🟩 3	5	1

Objective Fct : 39

……

4	🟩 1	2	5	🟩 3

Objective Fct : 21

Neighborhood of solution S

Tabu list empty

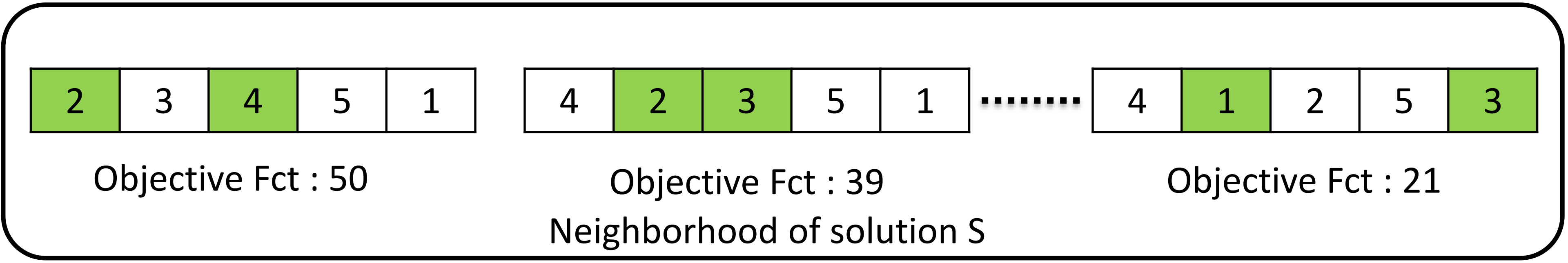
S-metaheuristics – Tabu Search Algorithm (TS)

- Best admissible solution (TSP prob): Scenario 1

Solution S:

1	2	3	4	5
4	3	2	5	1

: Objective Fct = 33



2	3	4	5	1

Objective Fct : 50

4	2	3	5	1

Objective Fct : 39

……

4	1	2	5	3

Objective Fct : 21

Neighborhood of solution S

Tabu list empty

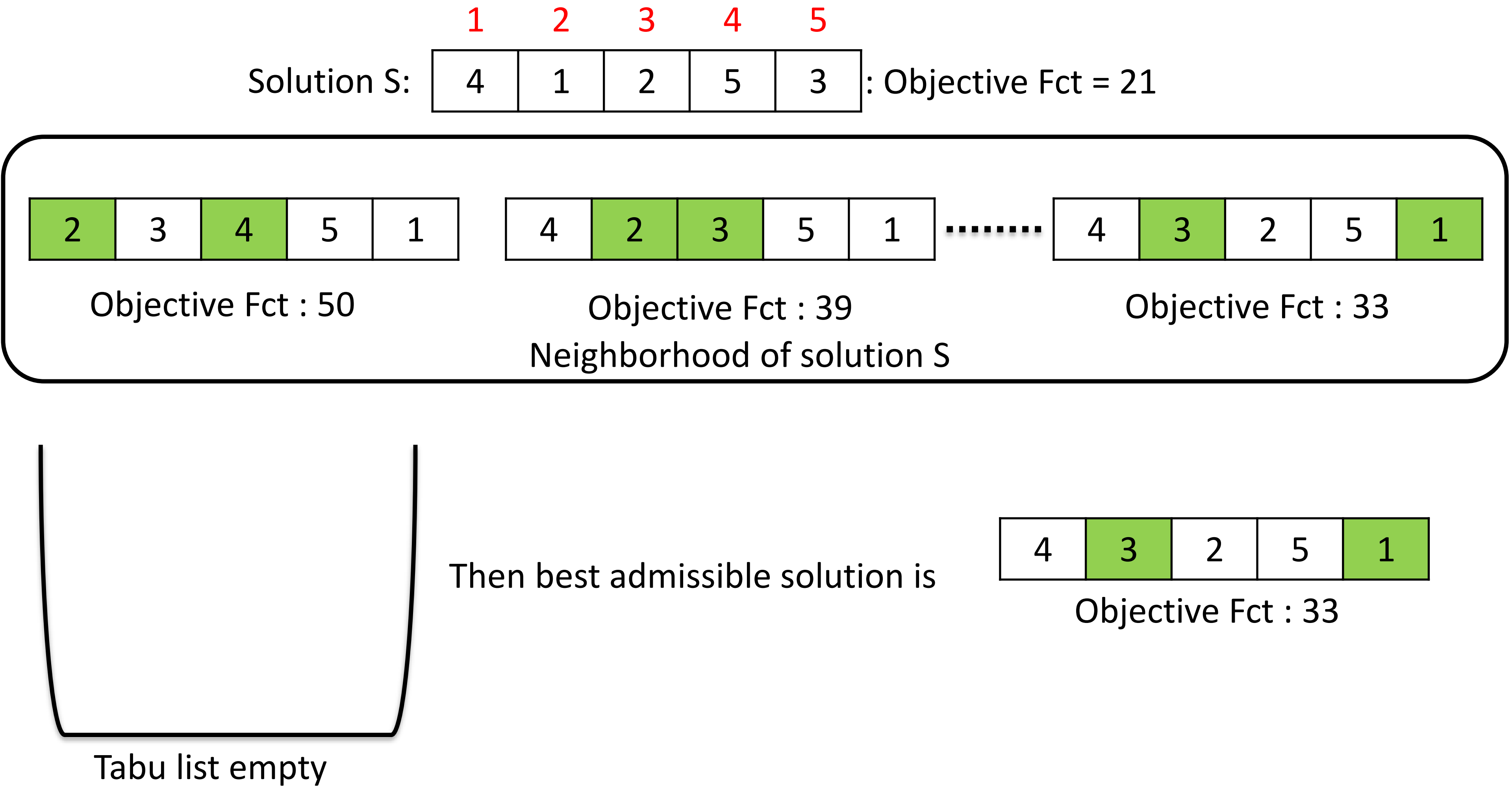
Then best admissible solution is

4	1	2	5	3

Objective Fct : 21

S-metaheuristics – Tabu Search Algorithm (TS)

- Best admissible solution (TSP prob): Scenario 2



	1	2	3	4	5	
Solution S:	4	1	2	5	3	: Objective Fct = 21

🟩 2	3	🟩 4	5	1

Objective Fct : 50

4	🟩 2	🟩 3	5	1

Objective Fct : 39

……..

4	🟩 3	2	5	🟩 1

Objective Fct : 33

Neighborhood of solution S

Then best admissible solution is

4	🟩 3	2	5	🟩 1

Objective Fct : 33

Tabu list empty

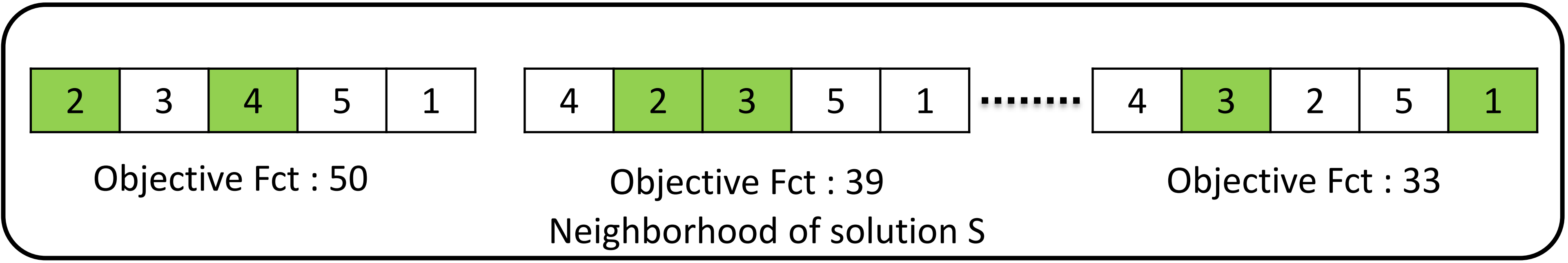
S-metaheuristics – Tabu Search Algorithm (TS)

- Best admissible solution (TSP prob): Scenario 3

Solution S:

1	2	3	4	5
4	1	2	5	3

: Objective Fct = 21



🟩 2	3	🟩 4	5	1

Objective Fct : 50

4	🟩 2	🟩 3	5	1

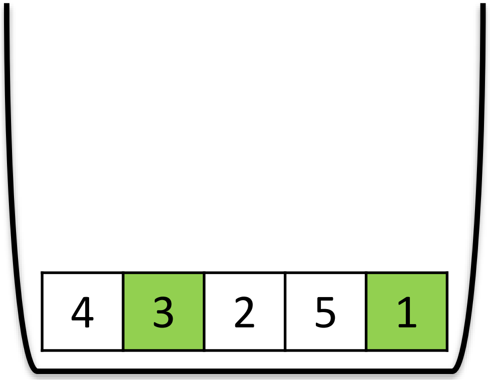
Objective Fct : 39

……..

4	🟩 3	2	5	🟩 1

Objective Fct : 33

Neighborhood of solution S



4	🟩 3	2	5	🟩 1

Tabu list empty

Then best admissible solution is

4	🟩 2	🟩 3	5	1

Objective Fct : 39

S-metaheuristics – Tabu Search Algorithm (TS)

- Main characteristics
  - TS behaves like a steepest LS algorithm, but it accepts non improving solutions to escape from local optima when all neighbors are non improving solutions.
  - Usually, the whole neighborhood is explored in a deterministic manner
  - Dynamic neighborhood
    - Managing a short memory of visited neighbors called *tabu list*
    - Neighborhood of x may change according to the history of the search

S-metaheuristics – Tabu Search Algorithm (TS)

- Design issues
  - In addition to standard S-metaheuristics issues
    - Neighborhood definition
    - Initial solution generation
  - Tabu list definition
    - Which term for the memory
      - Short, medium, long (time consuming)
  - Aspiration criterion
    - Accept a tabu solution if it satisfies some “conditions”

    ➔ tabu solution better than the visited solutions

S-metaheuristics – Tabu Search Algorithm (TS)

- Additional search mechanisms
  - Intensification (medium-term memory)
    - Save best solutions (elite) during search then focus search around its neighborhood
  - Diversification (long-term memory)
    - Save visited solutions along search process and explore unvisited areas

S-metaheuristics – Tabu Search Algorithm (TS)

Template of tabu search algorithm.

```
$s = s_0$ ; /* Initial solution */
Initialize the tabu list, medium-term
Repeat
    Find best admissible neighbor $s'$ ; /* non tabu or aspiration criterion holds */
    $s = s'$ ;
    Update tabu list, aspiration conditions, medium and long term memories ;
    If intensification_criterion holds  Then intensification ;

Until Stopping criteria satisfied
Output: Best solution found.
```

Save ***S*** as ***best_solution***

Improve best_***S***

Compare ***S*** with the ***best_solution*** and update if necessary

S-metaheuristics – Tabu Search Algorithm (TS)

- Intensification (medium term memory)
  - Exploit best solutions for guiding search in promising regions
  - Recency memory technic (problem dependent)
    - Start search from best found solution while keeping some of its components unchangeable
    - The ones that are present for a successive set of iterations
    - Example for TSP (Travelling Salesman problem)
      - Create a matrix *R* where *r(i,j)* corresponding to number of successive iterations where city *i* is positioned at *j*
      - Intensification
        - Start search from best found solution *S*
        - Use largest values of *R* to freeze components of S
        - Focus search on other components

S-metaheuristics – Tabu Search Algorithm (TS)

- Intensification (Recency memory – TSP 5 cities)

POSITIONS

CITIES	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

R MATRIX

SOLUTION S

1	2	3	4	5
3	5	1	2	4

ITERATION 1 →

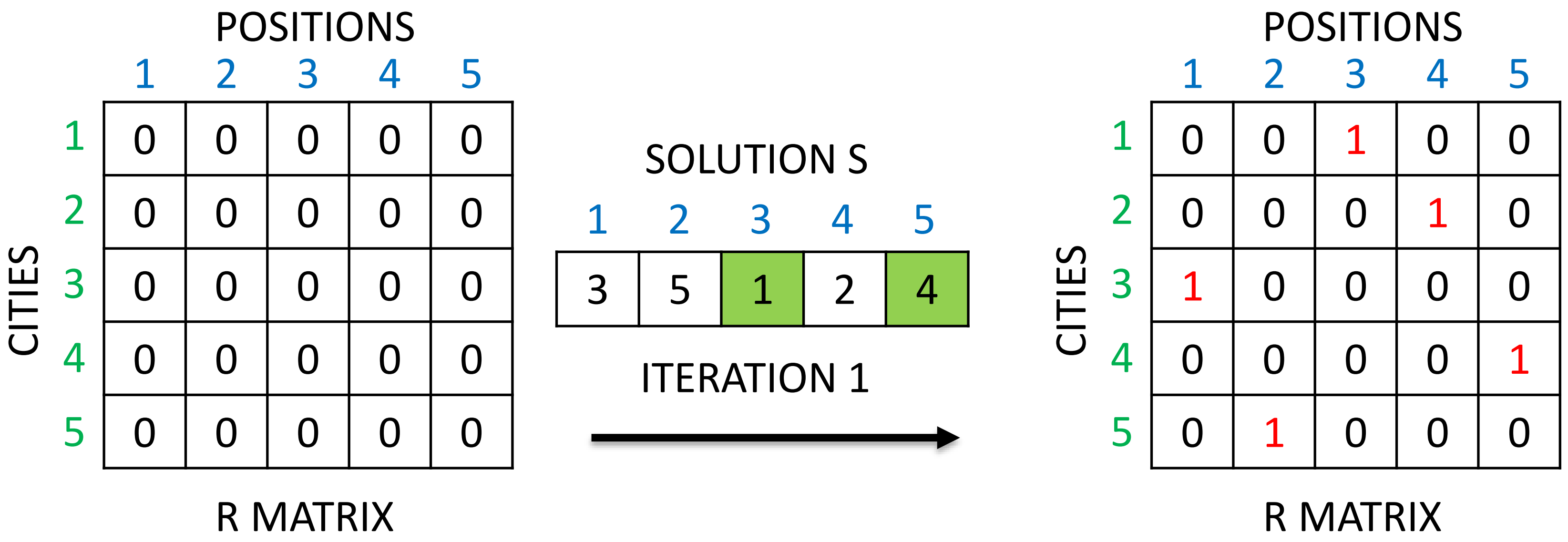
POSITIONS

CITIES	1	2	3	4	5
1	0	0	1	0	0
2	0	0	0	1	0
3	1	0	0	0	0
4	0	0	0	0	1
5	0	1	0	0	0

R MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Intensification (Recency memory – TSP 5 cities)



POSITIONS

CITIES	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

R MATRIX

SOLUTION S

1	2	3	4	5
3	5	1	2	4

ITERATION 1

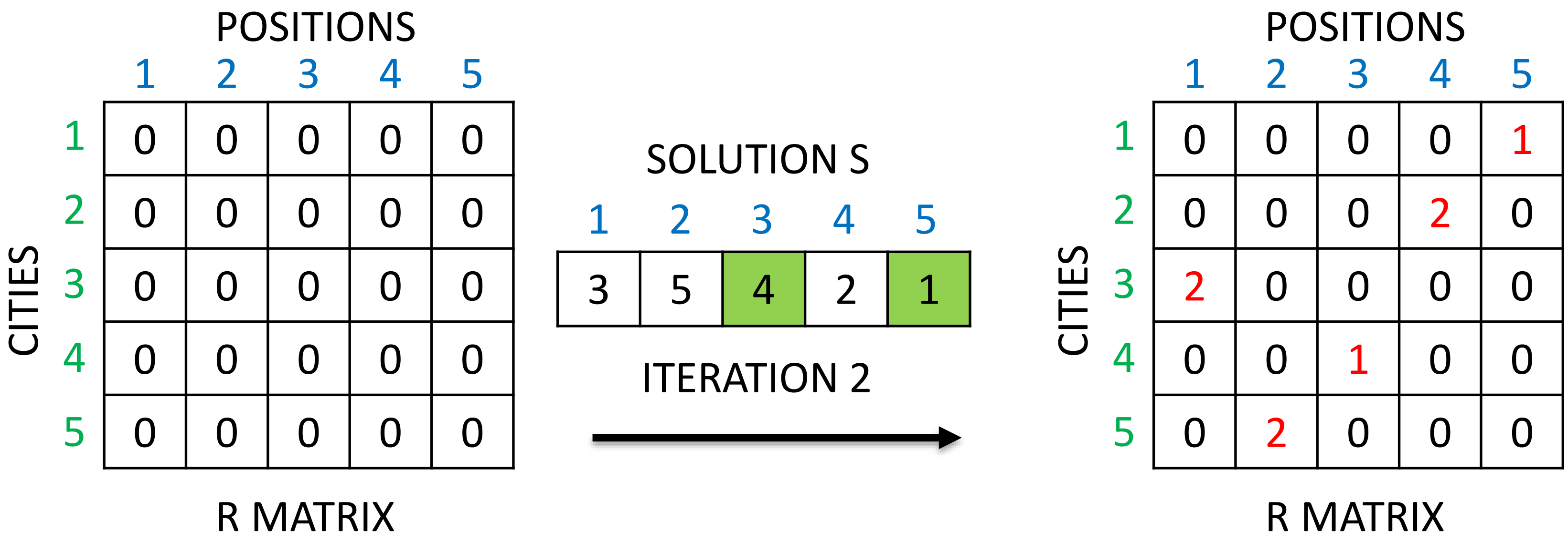
POSITIONS

CITIES	1	2	3	4	5
1	0	0	1	0	0
2	0	0	0	1	0
3	1	0	0	0	0
4	0	0	0	0	1
5	0	1	0	0	0

R MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Intensification (Recency memory – TSP 5 cities)



POSITIONS

CITIES

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

R MATRIX

SOLUTION S

1	2	3	4	5
3	5	4	2	1

ITERATION 2

POSITIONS

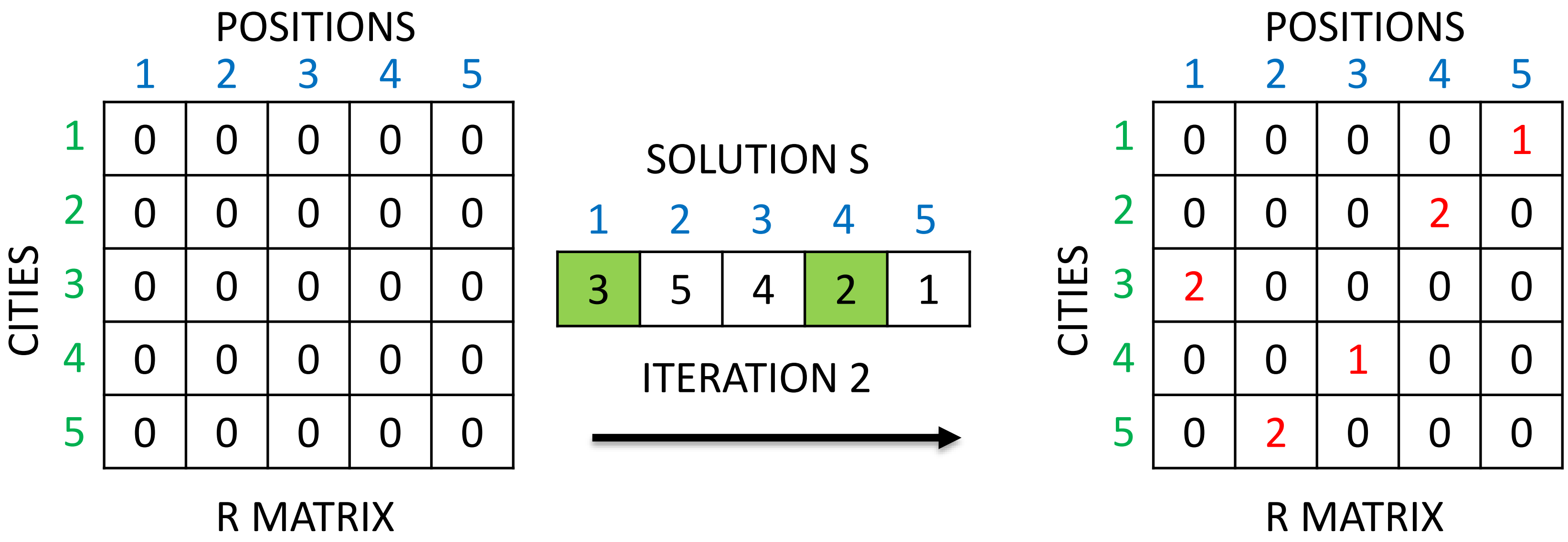
CITIES

	1	2	3	4	5
1	0	0	0	0	1
2	0	0	0	2	0
3	2	0	0	0	0
4	0	0	1	0	0
5	0	2	0	0	0

R MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Intensification (Recency memory – TSP 5 cities)



POSITIONS

CITIES

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

R MATRIX

SOLUTION S

1	2	3	4	5
3	5	4	2	1

ITERATION 2

POSITIONS

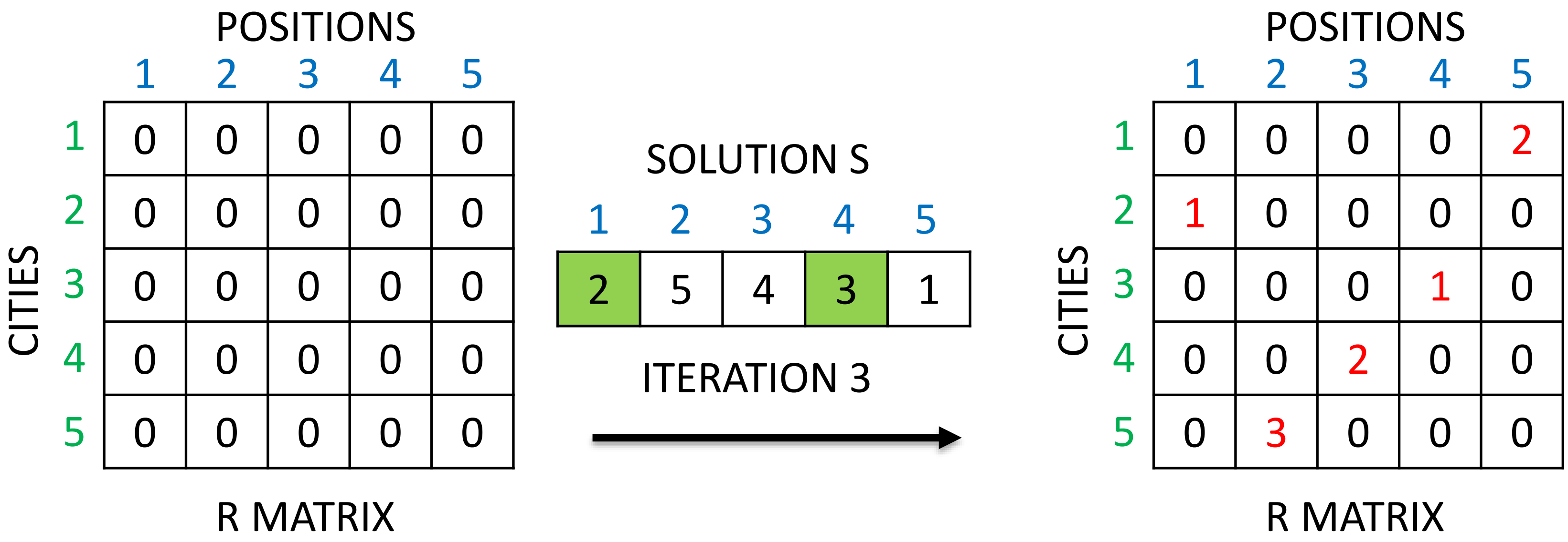
CITIES

	1	2	3	4	5
1	0	0	0	0	1
2	0	0	0	2	0
3	2	0	0	0	0
4	0	0	1	0	0
5	0	2	0	0	0

R MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Intensification (Recency memory – TSP 5 cities)



POSITIONS

CITIES

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

R MATRIX

SOLUTION S

1	2	3	4	5
2	5	4	3	1

ITERATION 3

POSITIONS

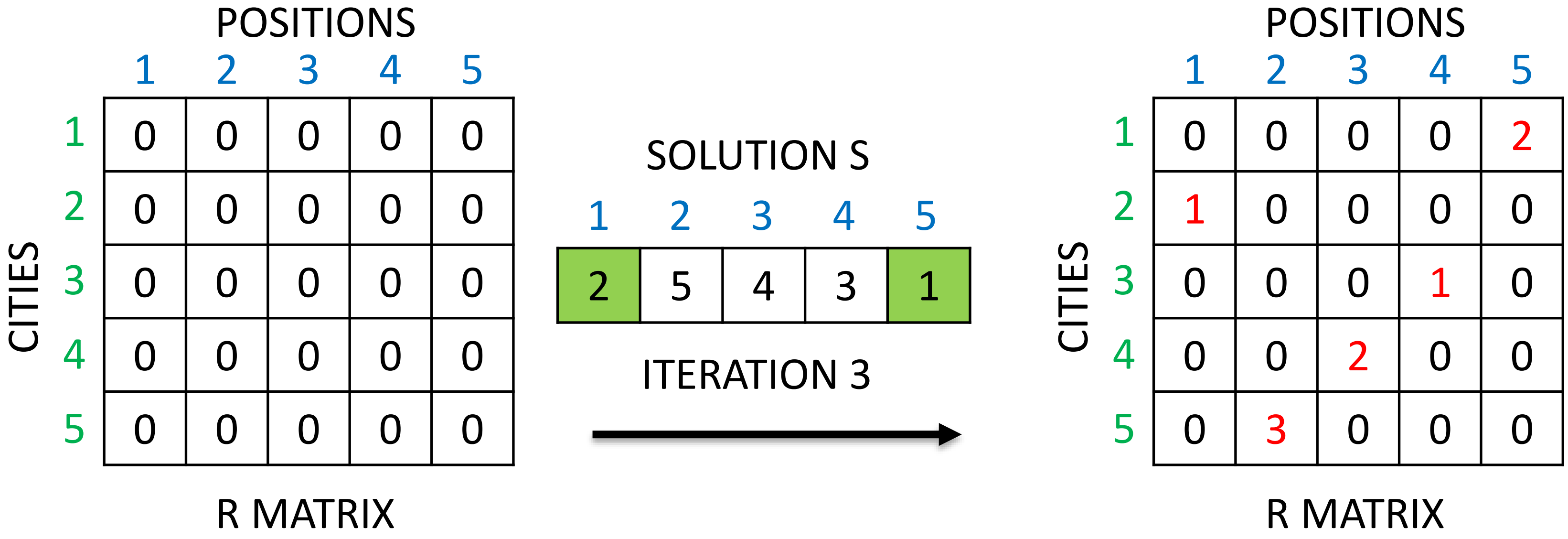
CITIES

	1	2	3	4	5
1	0	0	0	0	2
2	1	0	0	0	0
3	0	0	0	1	0
4	0	0	2	0	0
5	0	3	0	0	0

R MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Intensification (Recency memory – TSP 5 cities)



POSITIONS

CITIES

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

R MATRIX

SOLUTION S

1	2	3	4	5
2	5	4	3	1

ITERATION 3

POSITIONS

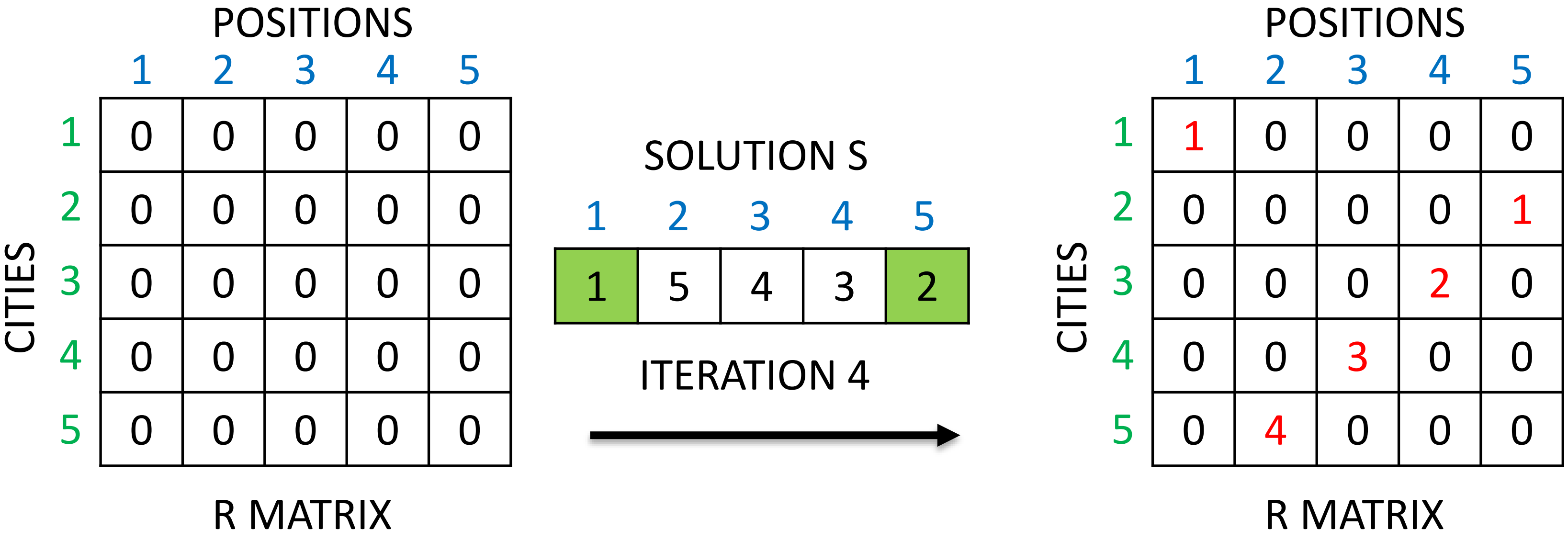
CITIES

	1	2	3	4	5
1	0	0	0	0	2
2	1	0	0	0	0
3	0	0	0	1	0
4	0	0	2	0	0
5	0	3	0	0	0

R MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Intensification (Recency memory – TSP 5 cities)



POSITIONS

CITIES	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

R MATRIX

SOLUTION S

1	2	3	4	5
1	5	4	3	2

ITERATION 4

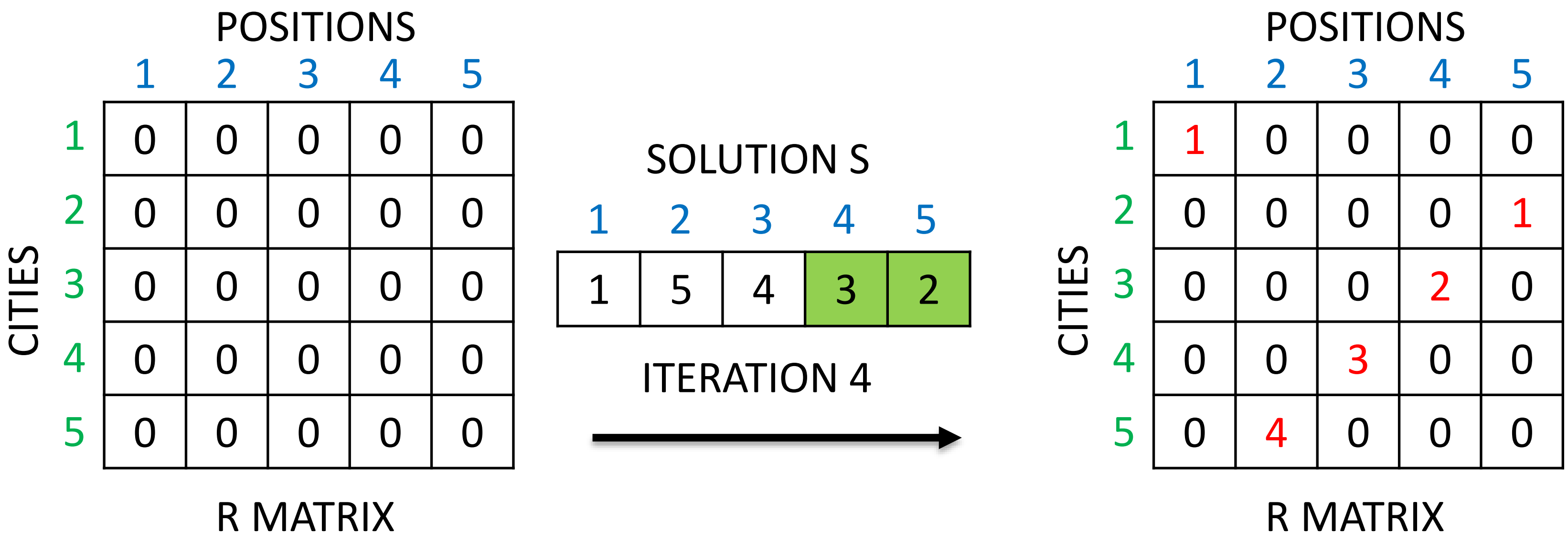
POSITIONS

CITIES	1	2	3	4	5
1	1	0	0	0	0
2	0	0	0	0	1
3	0	0	0	2	0
4	0	0	3	0	0
5	0	4	0	0	0

R MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Intensification (Recency memory – TSP 5 cities)



POSITIONS

CITIES

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

R MATRIX

SOLUTION S

1	2	3	4	5
1	5	4	3	2

ITERATION 4

POSITIONS

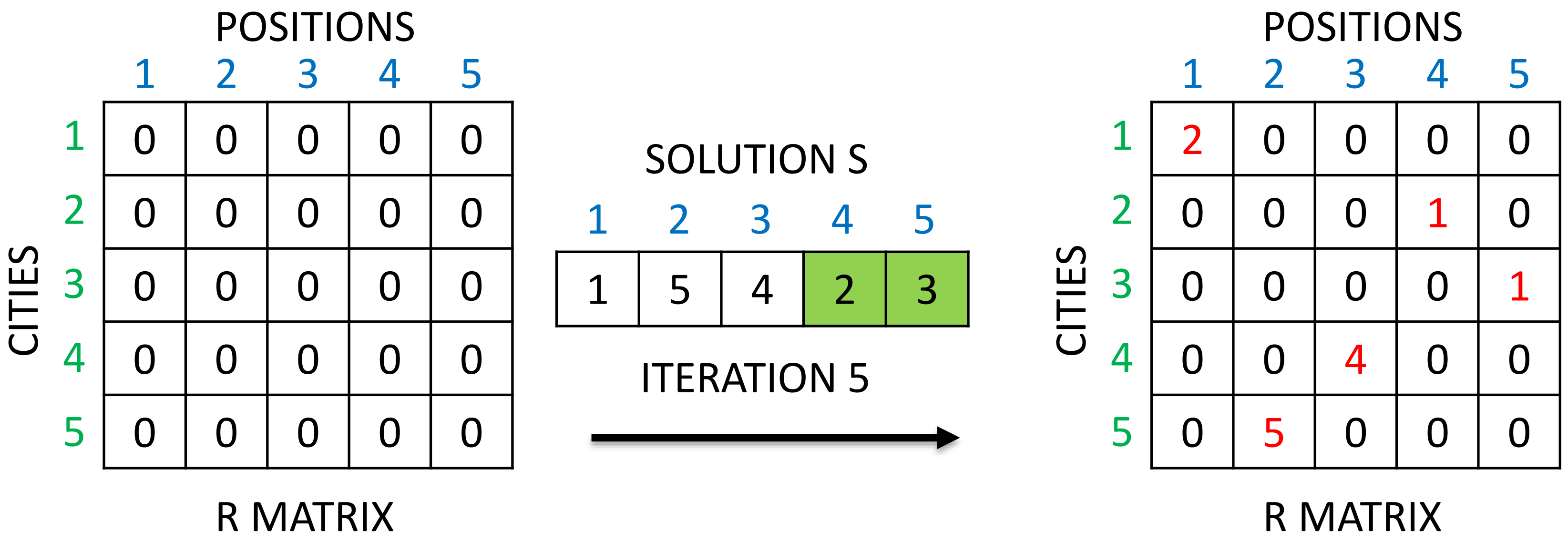
CITIES

	1	2	3	4	5
1	1	0	0	0	0
2	0	0	0	0	1
3	0	0	0	2	0
4	0	0	3	0	0
5	0	4	0	0	0

R MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Intensification (Recency memory – TSP 5 cities)



POSITIONS

CITIES	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

R MATRIX

SOLUTION S

1	2	3	4	5
1	5	4	2	3

ITERATION 5

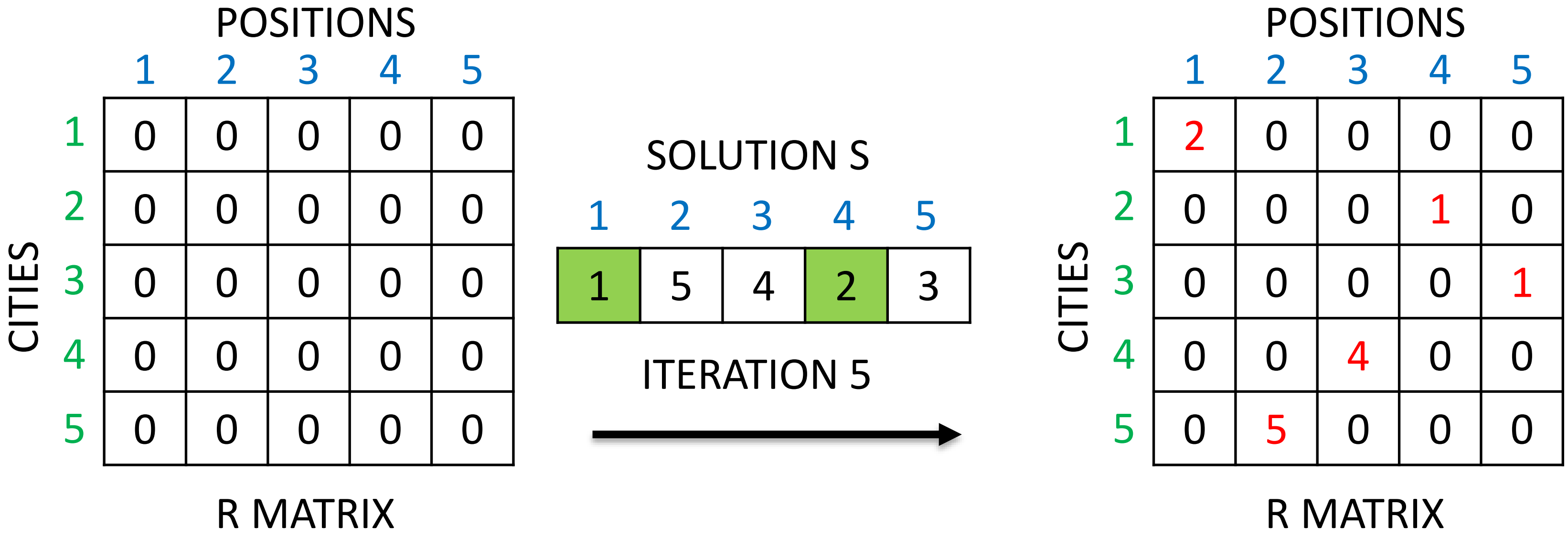
POSITIONS

CITIES	1	2	3	4	5
1	2	0	0	0	0
2	0	0	0	1	0
3	0	0	0	0	1
4	0	0	4	0	0
5	0	5	0	0	0

R MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Intensification (Recency memory – TSP 5 cities)



POSITIONS

CITIES

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

R MATRIX

SOLUTION S

1	2	3	4	5
1	5	4	2	3

ITERATION 5

POSITIONS

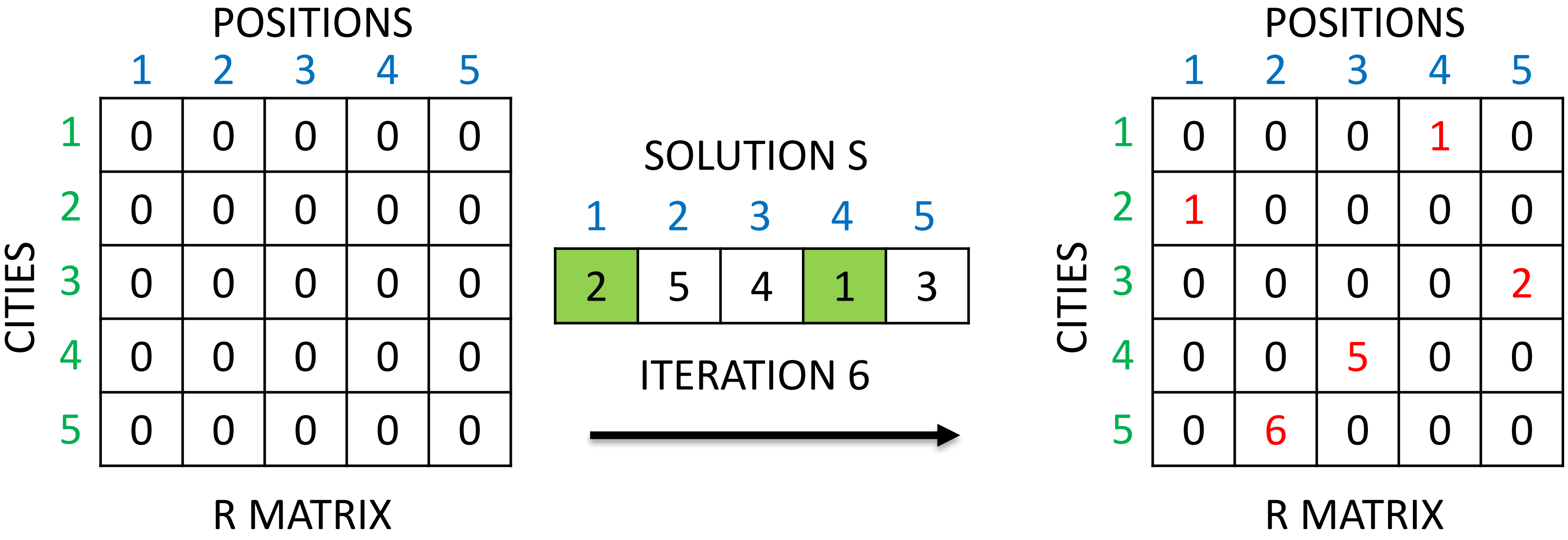
CITIES

	1	2	3	4	5
1	2	0	0	0	0
2	0	0	0	1	0
3	0	0	0	0	1
4	0	0	4	0	0
5	0	5	0	0	0

R MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Intensification (Recency memory – TSP 5 cities)



POSITIONS

CITIES	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

R MATRIX

SOLUTION S

1	2	3	4	5
2	5	4	1	3

ITERATION 6

POSITIONS

CITIES	1	2	3	4	5
1	0	0	0	1	0
2	1	0	0	0	0
3	0	0	0	0	2
4	0	0	5	0	0
5	0	6	0	0	0

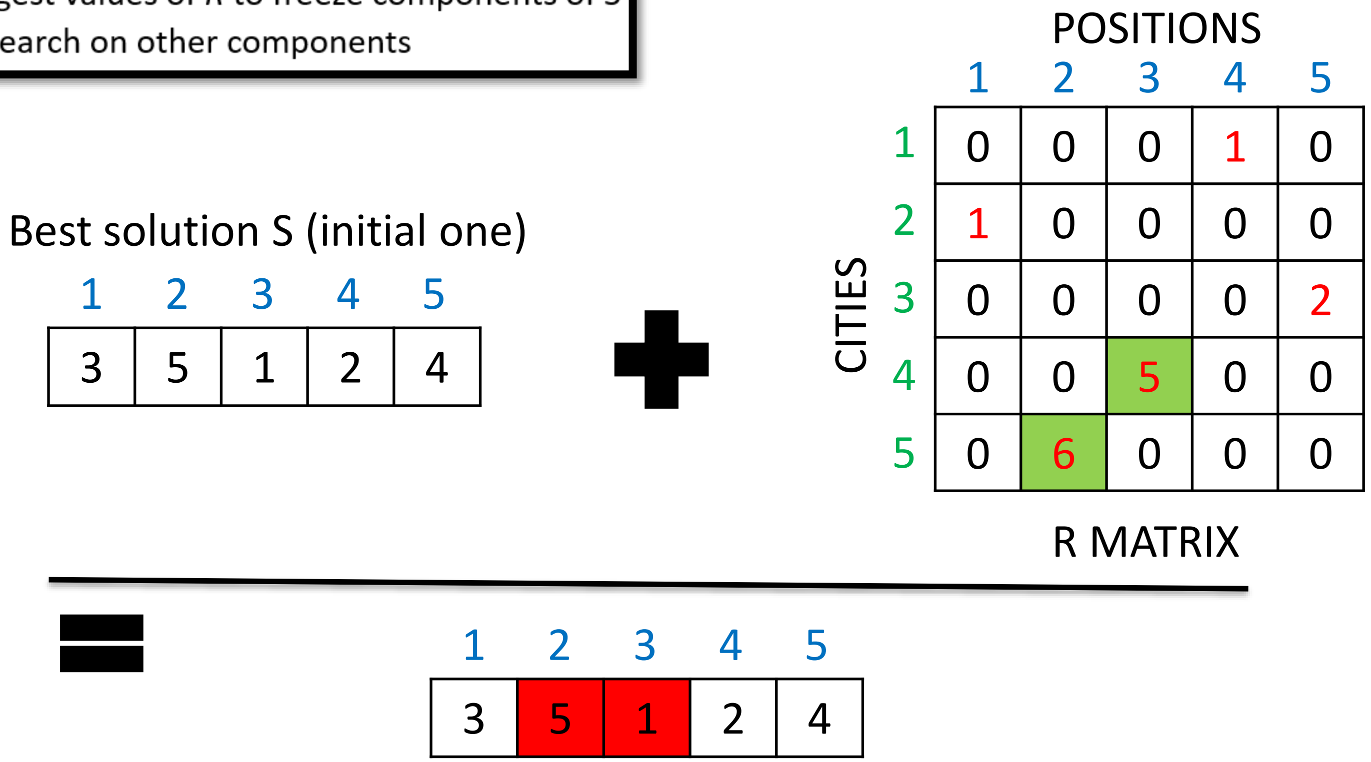
R MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Intensification (Recency memory – TSP 5 cities)

- Intensification
  - Start search from best found solution *S*
  - Use largest values of *R* to freeze components of S
  - Focus search on other components

After a given number of iterations, start intensification



Best solution S (initial one)

1	2	3	4	5
3	5	1	2	4

POSITIONS

CITIES	1	2	3	4	5
1	0	0	0	1	0
2	1	0	0	0	0
3	0	0	0	0	2
4	0	0	5	0	0
5	0	6	0	0	0

R MATRIX

1	2	3	4	5
3	5	1	2	4

S-metaheuristics – Tabu Search Algorithm (TS)

- Intensification (Recency memory – TSP 5 cities)

- Intensification
  - Start search from best found solution *S*
  - Use largest values of *R* to freeze components of S
  - Focus search on other components

After a given number of iterations, start intensification

POSITIONS

CITIES

	1	2	3	4	5
1	0	0	0	1	0
2	1	0	0	0	0
3	0	0	0	0	2
4	0	0	5	0	0
5	0	6	0	0	0

R MATRIX

Best solution S (initial one)

1	2	3	4	5
3	5	1	2	4

+

=

1	2	3	4	5
3	5	1	2	4

S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (long term memory)
  - Forcing search in unexplored regions
  - Frequency memory technic (problem dependent)
    - Focus search on less changed components in the search history
    - Example for TSP (Travelling Salesman problem)
      - Create a matrix *F* where *f(i,j)* corresponding to number of iterations where city *i* is positioned at *j* from the starting of algorithm
      - Diversification
        - Start search from a new initial solution *S* generated as follow
          - Use smallest values of *F* to replace components of *S*
          - Pursue search

S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)

POSITIONS

CITIES	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

F MATRIX

SOLUTION S

1	2	3	4	5
3	5	1	2	4

ITERATION 1 →

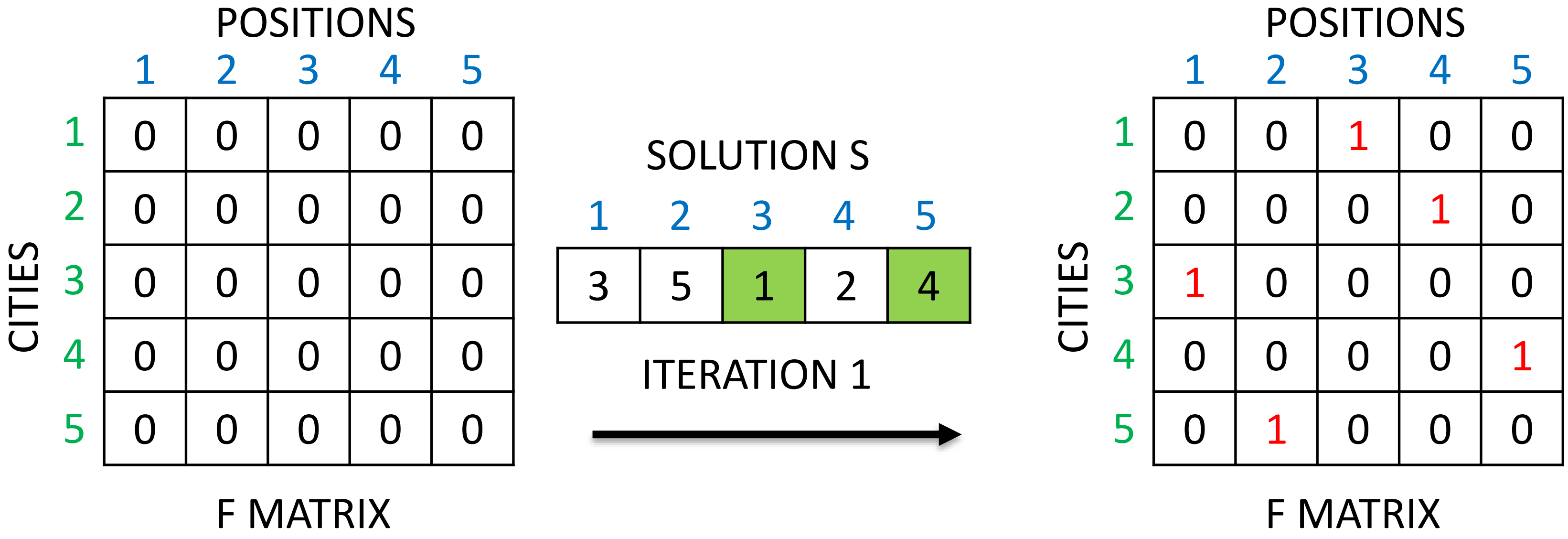
POSITIONS

CITIES	1	2	3	4	5
1	0	0	1	0	0
2	0	0	0	1	0
3	1	0	0	0	0
4	0	0	0	0	1
5	0	1	0	0	0

F MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)



POSITIONS

CITIES

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

F MATRIX

SOLUTION S

1	2	3	4	5
3	5	1	2	4

ITERATION 1

POSITIONS

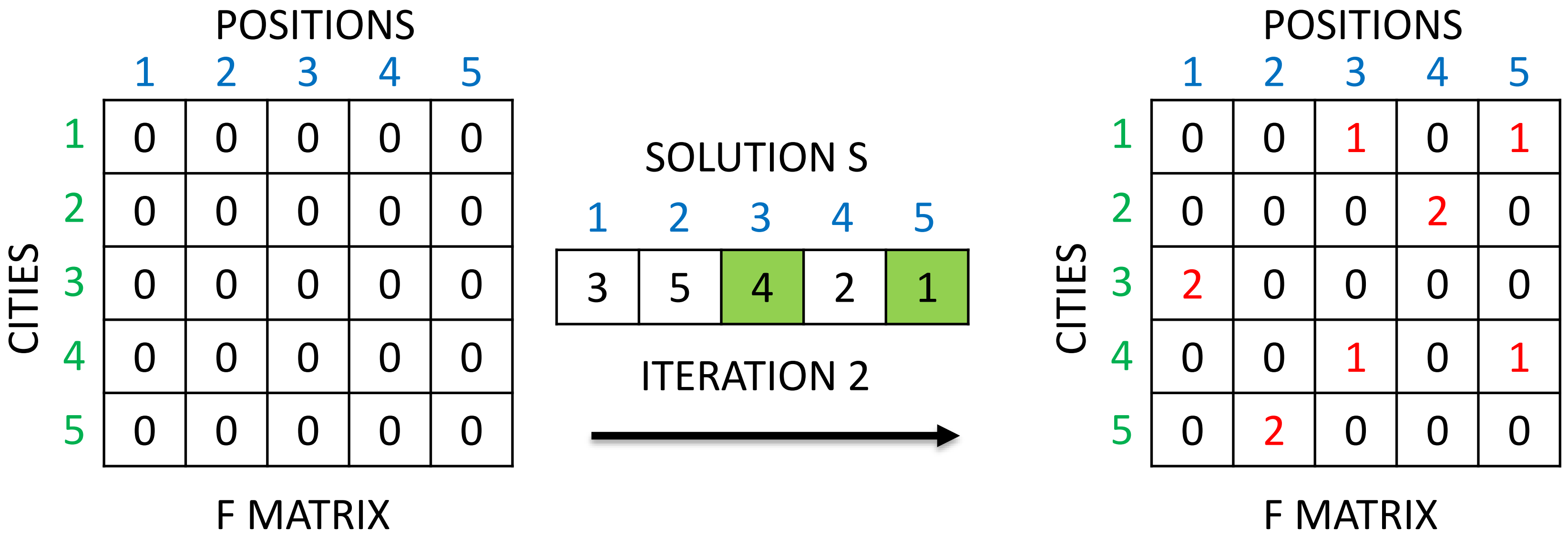
CITIES

	1	2	3	4	5
1	0	0	1	0	0
2	0	0	0	1	0
3	1	0	0	0	0
4	0	0	0	0	1
5	0	1	0	0	0

F MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)



POSITIONS

CITIES

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

F MATRIX

SOLUTION S

1	2	3	4	5
3	5	4	2	1

ITERATION 2

POSITIONS

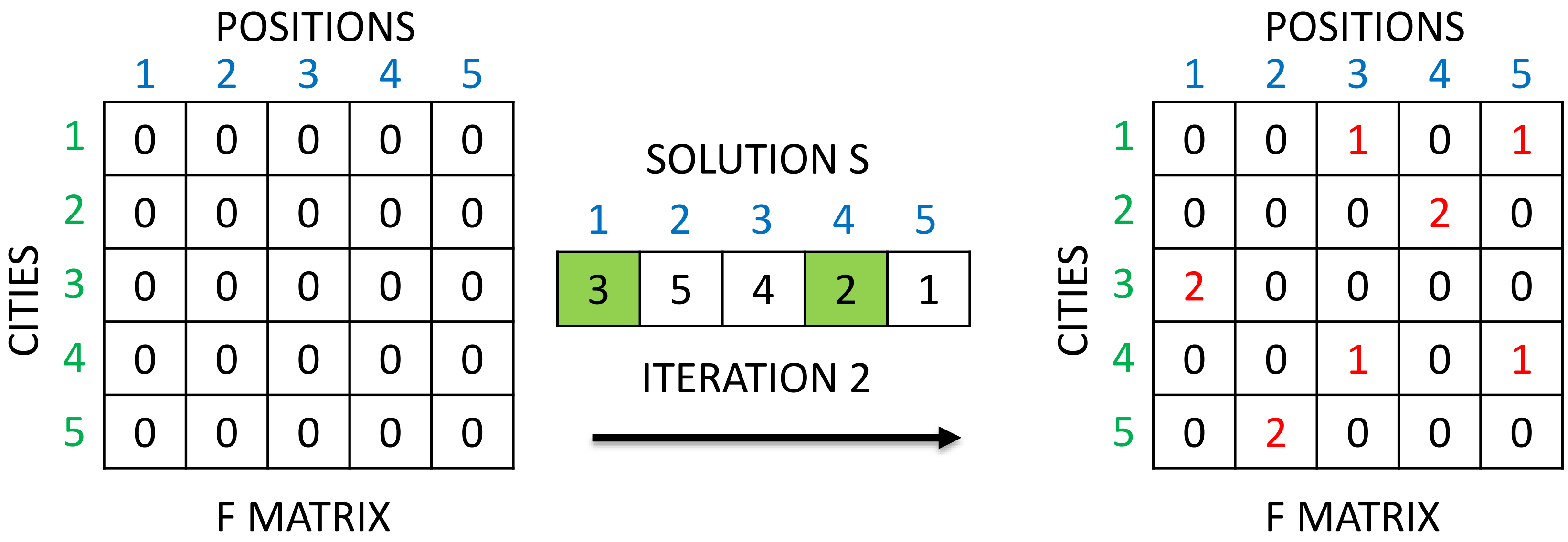
CITIES

	1	2	3	4	5
1	0	0	1	0	1
2	0	0	0	2	0
3	2	0	0	0	0
4	0	0	1	0	1
5	0	2	0	0	0

F MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)



POSITIONS

CITIES

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

F MATRIX

SOLUTION S

1	2	3	4	5
3	5	4	2	1

ITERATION 2

POSITIONS

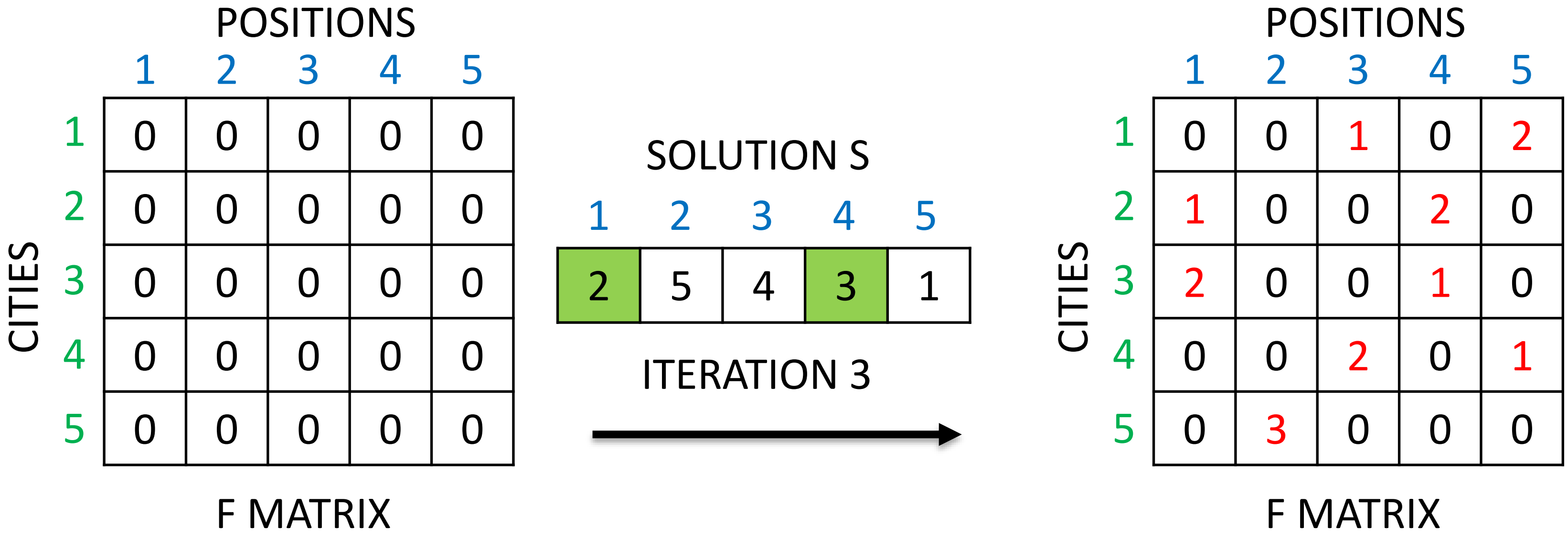
CITIES

	1	2	3	4	5
1	0	0	1	0	1
2	0	0	0	2	0
3	2	0	0	0	0
4	0	0	1	0	1
5	0	2	0	0	0

F MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)



POSITIONS

CITIES

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

F MATRIX

SOLUTION S

1	2	3	4	5
2	5	4	3	1

ITERATION 3

POSITIONS

CITIES

	1	2	3	4	5
1	0	0	1	0	2
2	1	0	0	2	0
3	2	0	0	1	0
4	0	0	2	0	1
5	0	3	0	0	0

F MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)

POSITIONS

CITIES

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

F MATRIX

SOLUTION S

1	2	3	4	5
2	5	4	3	1

ITERATION 3

→

POSITIONS

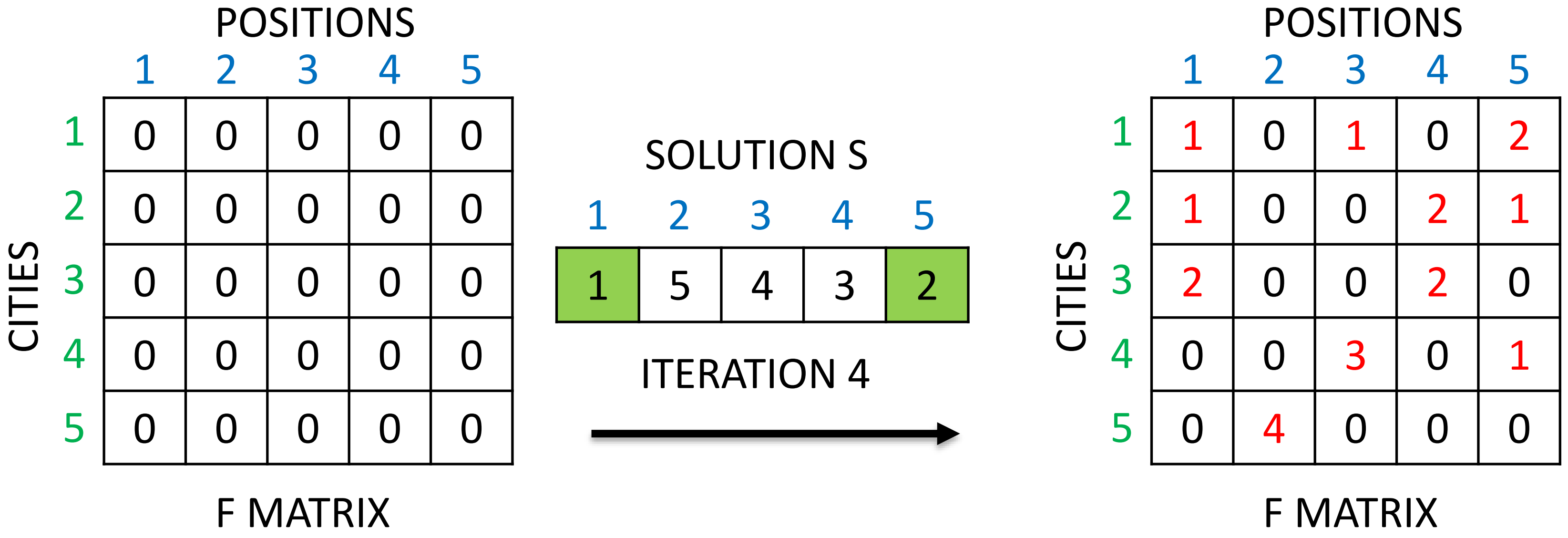
CITIES

	1	2	3	4	5
1	0	0	1	0	2
2	1	0	0	2	0
3	2	0	0	1	0
4	0	0	2	0	1
5	0	3	0	0	0

F MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)



POSITIONS

CITIES

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

F MATRIX

SOLUTION S

1	2	3	4	5
1	5	4	3	2

ITERATION 4

POSITIONS

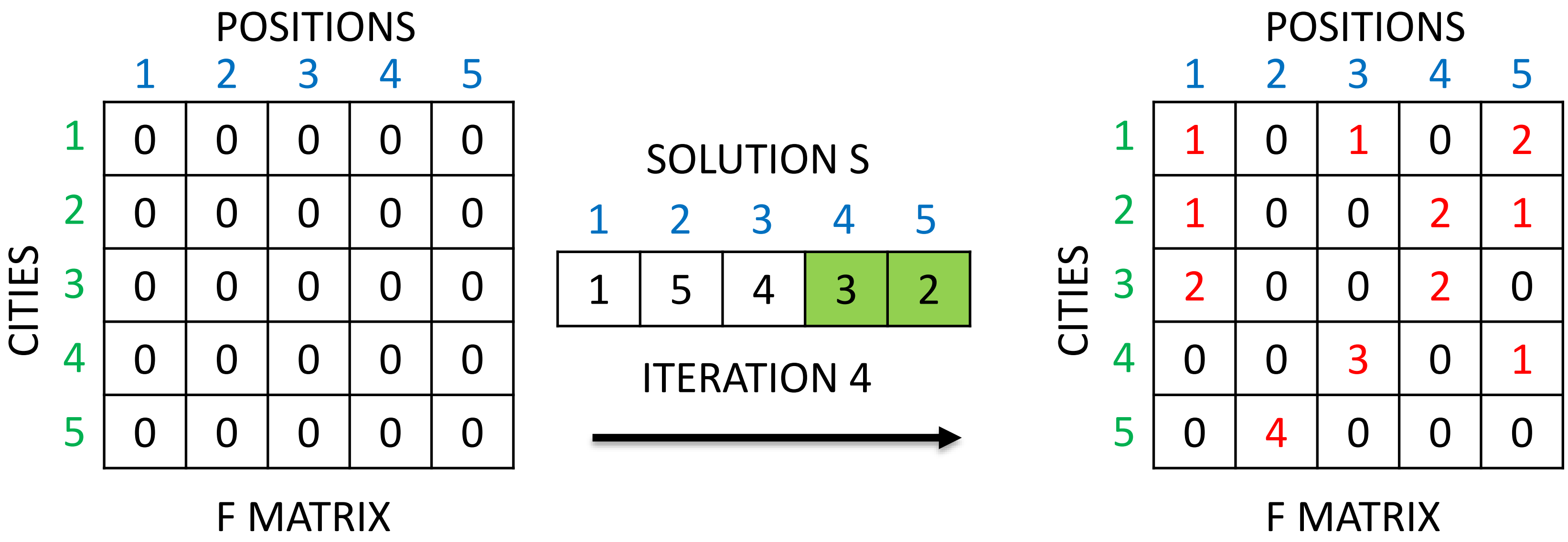
CITIES

	1	2	3	4	5
1	1	0	1	0	2
2	1	0	0	2	1
3	2	0	0	2	0
4	0	0	3	0	1
5	0	4	0	0	0

F MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)



POSITIONS

CITIES	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

F MATRIX

SOLUTION S

1	2	3	4	5
1	5	4	3	2

ITERATION 4

POSITIONS

CITIES	1	2	3	4	5
1	1	0	1	0	2
2	1	0	0	2	1
3	2	0	0	2	0
4	0	0	3	0	1
5	0	4	0	0	0

F MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)

POSITIONS

CITIES	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

F MATRIX

SOLUTION S

1	2	3	4	5
1	5	4	2	3

ITERATION 5

→

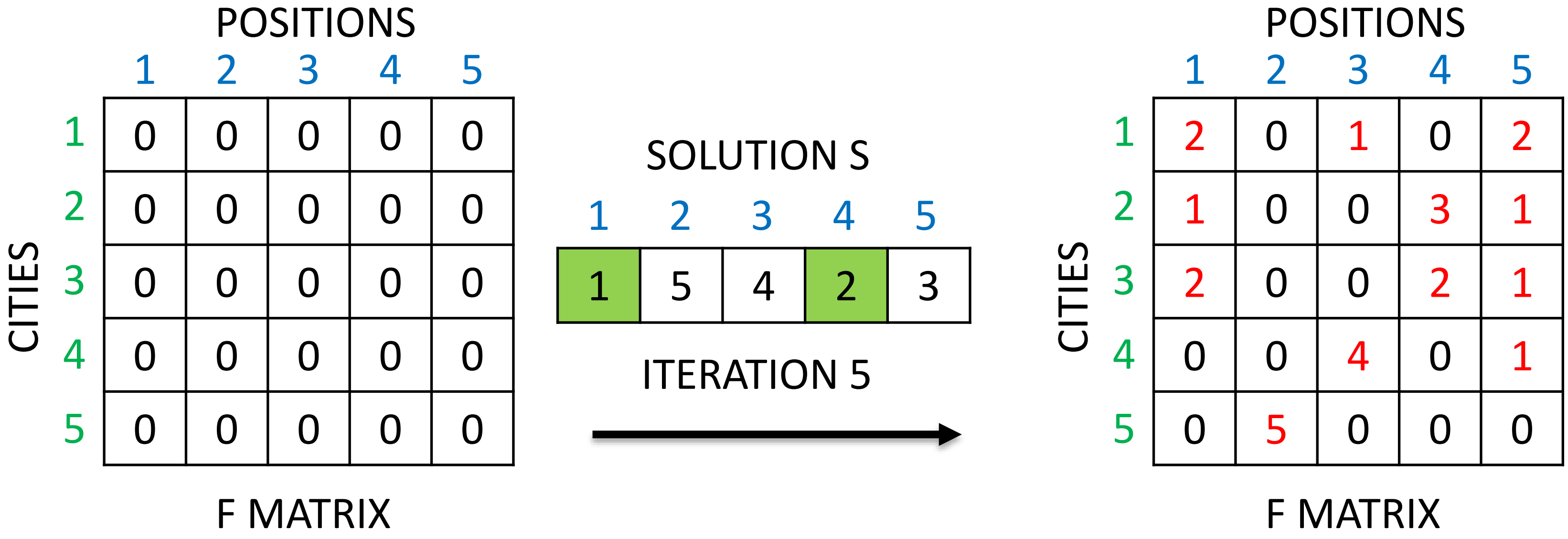
POSITIONS

CITIES	1	2	3	4	5
1	2	0	1	0	2
2	1	0	0	3	1
3	2	0	0	2	1
4	0	0	4	0	1
5	0	5	0	0	0

F MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)



POSITIONS

CITIES	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

F MATRIX

SOLUTION S

1	2	3	4	5
1	5	4	2	3

ITERATION 5

POSITIONS

CITIES	1	2	3	4	5
1	2	0	1	0	2
2	1	0	0	3	1
3	2	0	0	2	1
4	0	0	4	0	1
5	0	5	0	0	0

F MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)

POSITIONS

CITIES

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

F MATRIX

SOLUTION S

1	2	3	4	5
2	5	4	1	3

ITERATION 6

→

POSITIONS

CITIES

	1	2	3	4	5
1	2	0	1	1	2
2	2	0	0	3	1
3	2	0	0	2	2
4	0	0	5	0	1
5	0	6	0	0	0

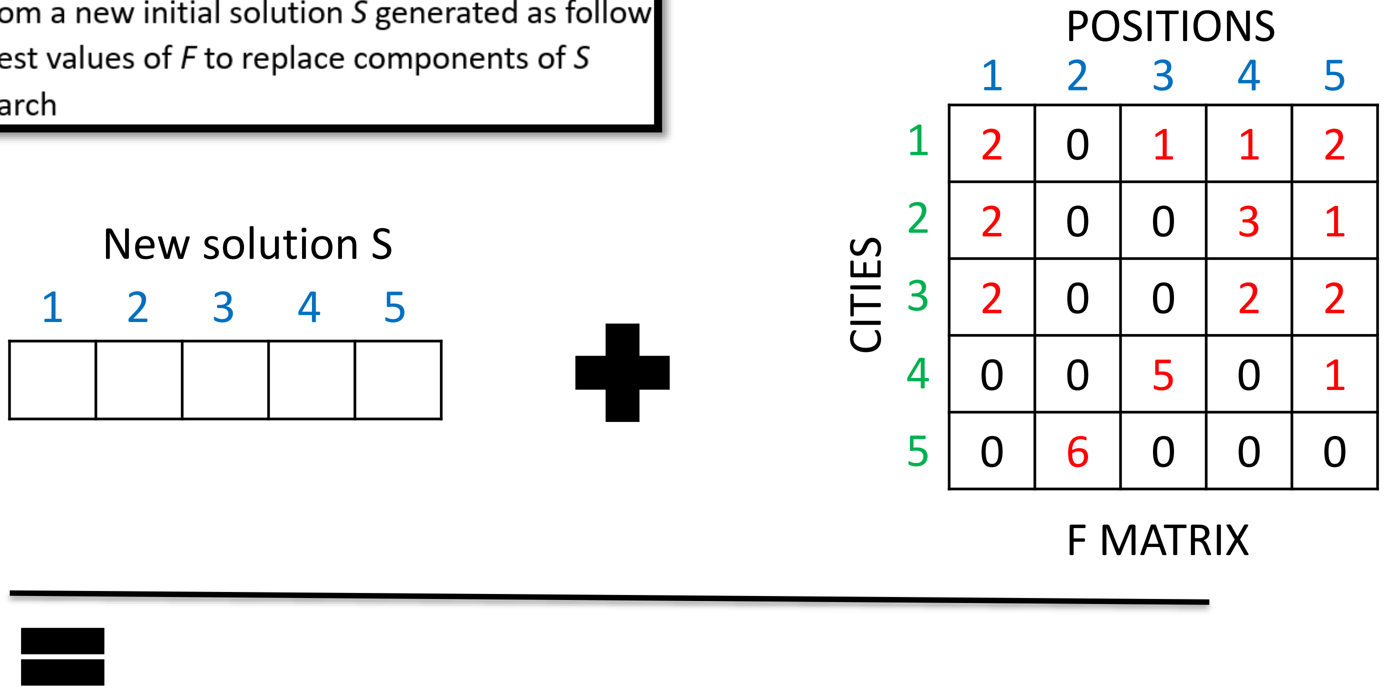
F MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)

- Diversification
  - Start search from a new initial solution *S* generated as follow
    - Use smallest values of *F* to replace components of *S*
    - Pursue search

After a given number of iterations, start diversification



New solution S

1	2	3	4	5

POSITIONS

CITIES	1	2	3	4	5
1	2	0	1	1	2
2	2	0	0	3	1
3	2	0	0	2	2
4	0	0	5	0	1
5	0	6	0	0	0

F MATRIX

S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)

After a given number of iterations, start diversification

- Diversification
  - Start search from a new initial solution *S* generated as follow
    - Use smallest values of *F* to replace components of *S*
    - Pursue search

New solution S

1	2	3	4	5
5				

+

POSITIONS

CITIES

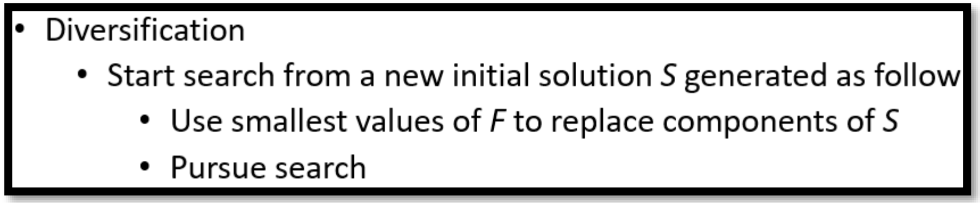
	1	2	3	4	5
1	2	0	1	1	2
2	2	0	0	3	1
3	2	0	0	2	2
4	0	0	5	0	1
5	0	6	0	0	0

F MATRIX

=

S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)



- Diversification
  - Start search from a new initial solution *S* generated as follow
    - Use smallest values of *F* to replace components of *S*
    - Pursue search

After a given number of iterations, start diversification

New solution S

1	2	3	4	5
5				

+

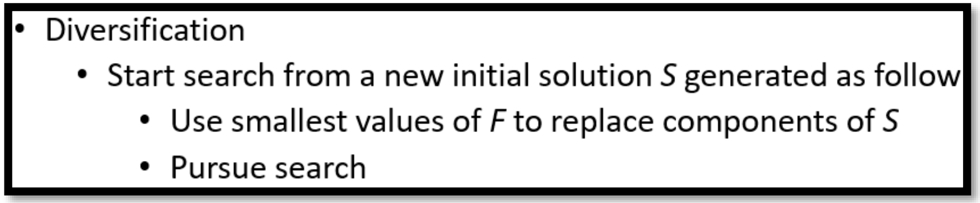
CITIES \ POSITIONS	1	2	3	4	5
1	■	0	1	1	2
2	■	0	0	3	1
3	■	0	0	2	2
4	■	0	5	0	1
5	■	■	■	■	■

F MATRIX

=

S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)



- Diversification
  - Start search from a new initial solution *S* generated as follow
    - Use smallest values of *F* to replace components of *S*
    - Pursue search

After a given number of iterations, start diversification

New solution S

1	2	3	4	5
5	1			

+

POSITIONS

CITIES	1	2	3	4	5
1	■	0	1	1	2
2	■	0	0	3	1
3	■	0	0	2	2
4	■	0	5	0	1
5	■	■	■	■	■

F MATRIX

=

S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)

After a given number of iterations, start diversification

- Diversification
  - Start search from a new initial solution *S* generated as follow
    - Use smallest values of *F* to replace components of *S*
    - Pursue search

New solution S

1	2	3	4	5
5	1			

+

POSITIONS

CITIES	1	2	3	4	5
1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
2	[redacted]	[redacted]	0	3	1
3	[redacted]	[redacted]	0	2	2
4	[redacted]	[redacted]	5	0	1
5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

F MATRIX

=

S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)

- Diversification
  - Start search from a new initial solution *S* generated as follow
    - Use smallest values of *F* to replace components of *S*
    - Pursue search

After a given number of iterations, start diversification

New solution S

1	2	3	4	5
5	1	3		

+

POSITIONS

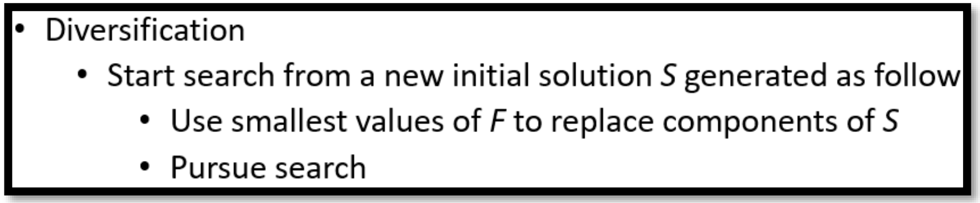
CITIES	1	2	3	4	5
1					
2			0	3	1
3			0	2	2
4			5	0	1
5					

F MATRIX

=

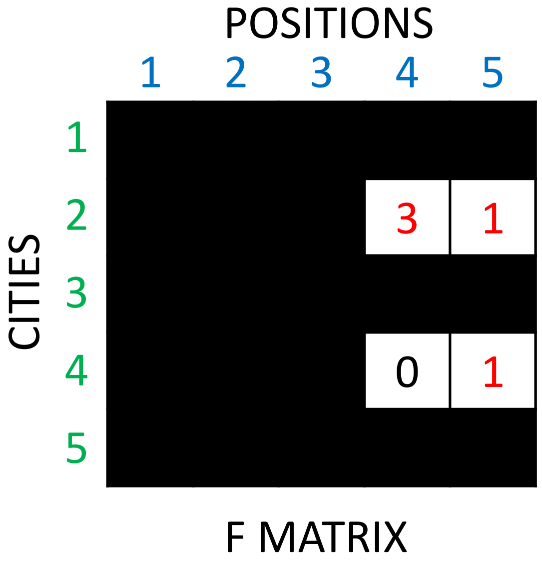
S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)



- Diversification
  - Start search from a new initial solution *S* generated as follow
    - Use smallest values of *F* to replace components of *S*
    - Pursue search

After a given number of iterations, start diversification



New solution S

1	2	3	4	5
5	1	3		

+

POSITIONS

CITIES	1	2	3	4	5
1	■	■	■	■	■
2	■	■	■	3	1
3	■	■	■	■	■
4	■	■	■	0	1
5	■	■	■	■	■

F MATRIX

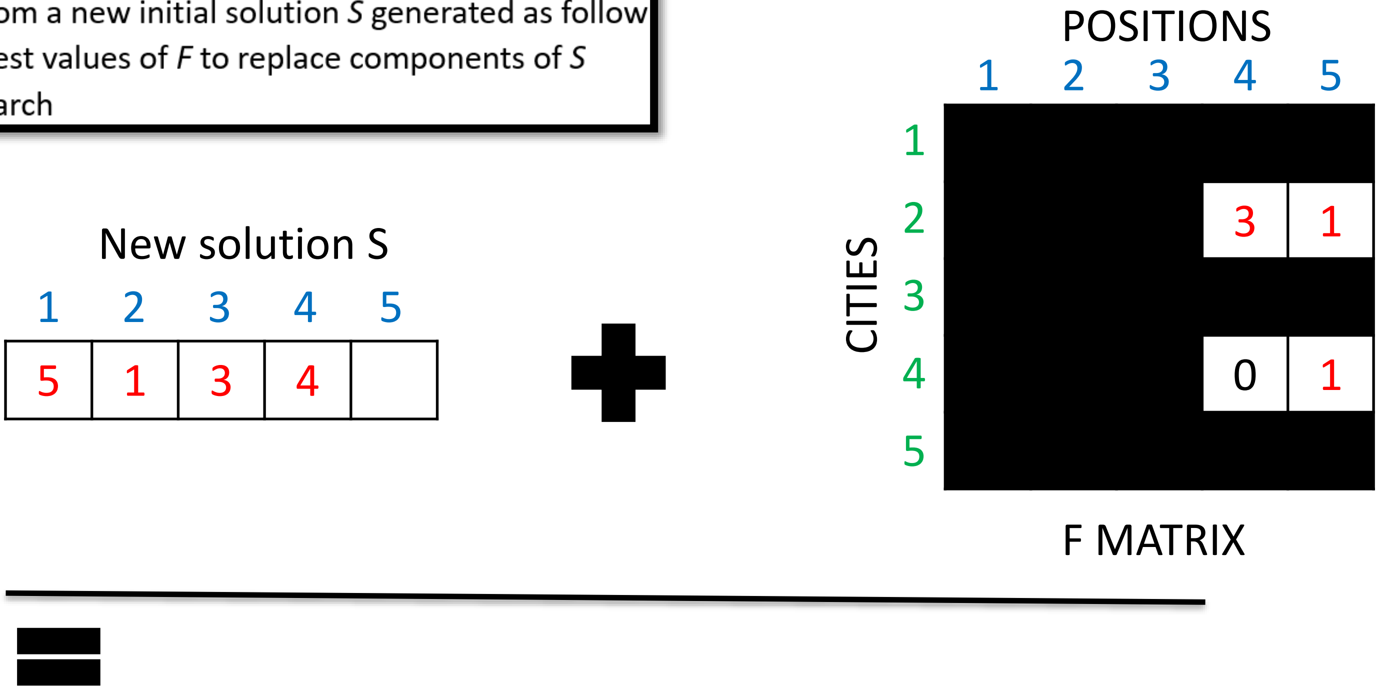
=

S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)

- Diversification
  - Start search from a new initial solution *S* generated as follow
    - Use smallest values of *F* to replace components of *S*
    - Pursue search

After a given number of iterations, start diversification



New solution S

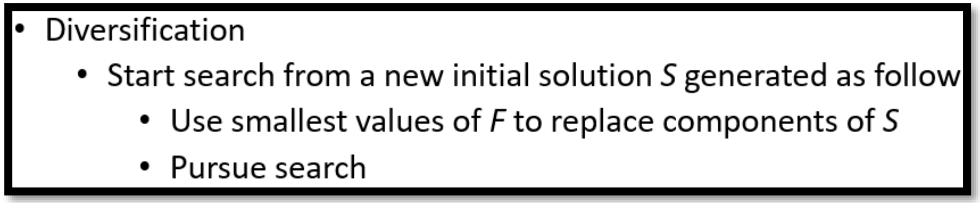
1	2	3	4	5
5	1	3	4	

	1	2	3	4	5
1	■	■	■	■	■
2	■	■	■	3	1
3	■	■	■	■	■
4	■	■	■	0	1
5	■	■	■	■	■

F MATRIX

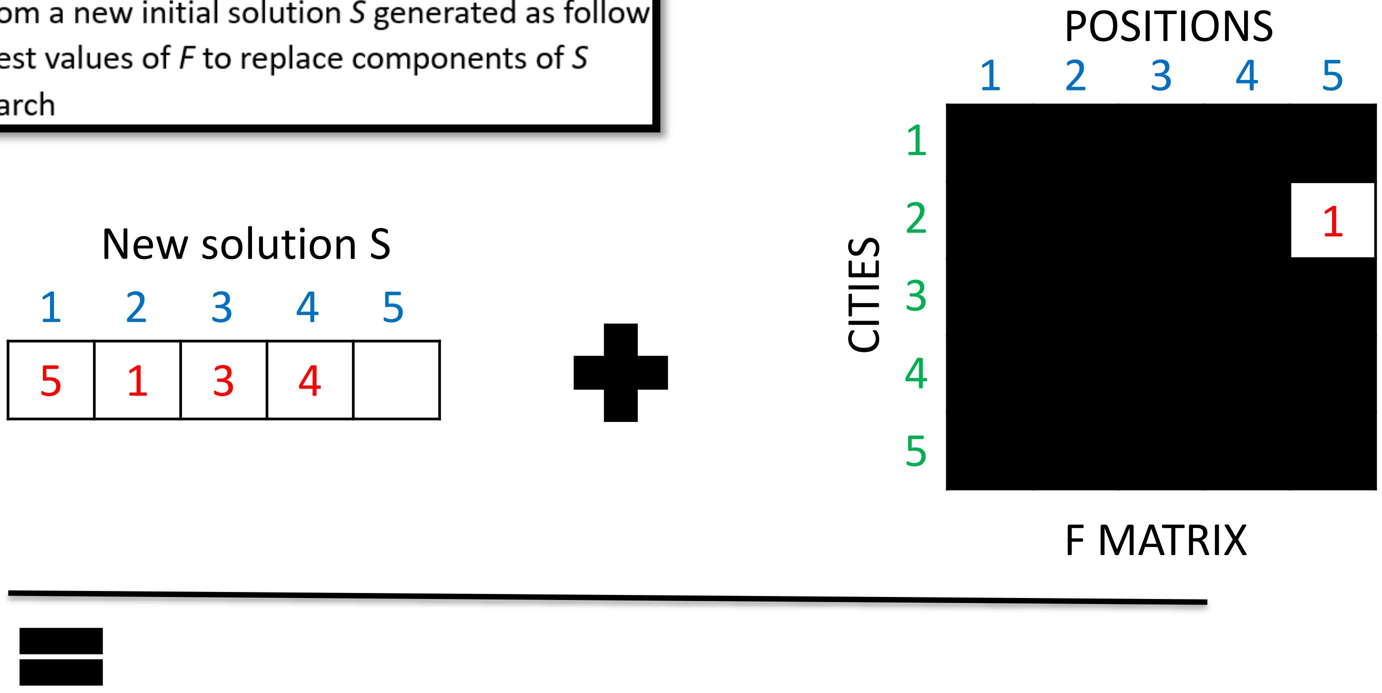
S-metaheuristics – Tabu Search Algorithm (TS)

- Diversification (Frequency memory – TSP 5 cities)



- Diversification
  - Start search from a new initial solution *S* generated as follow
    - Use smallest values of *F* to replace components of *S*
    - Pursue search

After a given number of iterations, start diversification



S-metaheuristics – Tabu Search Algorithm (TS)

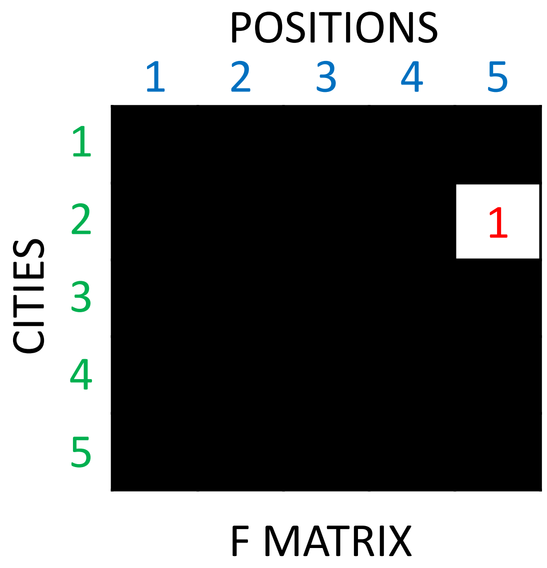
- Diversification (Frequency memory – TSP 5 cities)

- Diversification
  - Start search from a new initial solution *S* generated as follow
    - Use smallest values of *F* to replace components of *S*
    - Pursue search

After a given number of iterations, start diversification

New solution S

1	2	3	4	5
5	1	3	4	2



1	2	3	4	5
5	1	3	4	2

S-metaheuristics – Tabu Search Algorithm (TS)

Template of tabu search algorithm.

```
$s = s_0$ ; /* Initial solution */
Initialize the tabu list, medium-term and long-term memories ;
Repeat
  Find best admissible neighbor $s'$ ; /* non tabu or aspiration criterion holds */
  $s = s'$ ;
  Update tabu list, aspiration conditions, medium and long term memories ;
  If intensification_criterion holds  Then intensification ;
  If diversification_criterion holds  Then diversification ;
Until Stopping criteria satisfied
Output: Best solution found.
```

Save ***S*** as ***best_solution***

Improve ***S***

Compare ***S*** with the ***best_solution*** and update if necessary

S-metaheuristics – Algorithms review

Local search

- Selection strategies of the best neighbor
  - Best improvement (steepest descent)
  - First improvement
  - Random improvement

High probability to fall into local optima

Simulated annealing

- Accepting the degradation of a solution under some conditions
  - High temperatures promote accepting bad solutions
  - Static temperatures prevent accepting very bad solutions

Tabu search

- Accepting the degradation of a solution if and only if
  - Non tabu solution
- Memory usage to optimize the search process
  - Short term → tabu list
  - Medium term → recency for intensification
  - Long term → frequency for diversification

Strategies to escape from local optima

S-metaheuristics – Tabu Search Algorithm (TS)

- Lab session – first part

Implement your third and last S-metaheuristic algorithm - the Tabu Search algorithm (TS) -

- Version 1, only Tabu list (recommended before next session)
- Version 2, add intensification process
- Version 3, add diversification process

Apply the 2 versions to

- The TSP problem – Data available on the campus
- Show the best solution and associated trajectory curve for each version