COOPERATIVE AND ADAPTIVE ALGORITHMS

WEEK 4

Otman A. Al-Basir, Spring 2016

META-HEURISTICS

Meta-heuristics are algorithms that combine heuristics in a higher level framework aimed at efficiently and effectively exploring the search space,

Types:

- Trajectory methods, which handles one solution at a time,
- Population-based, which handles multiple solutions.

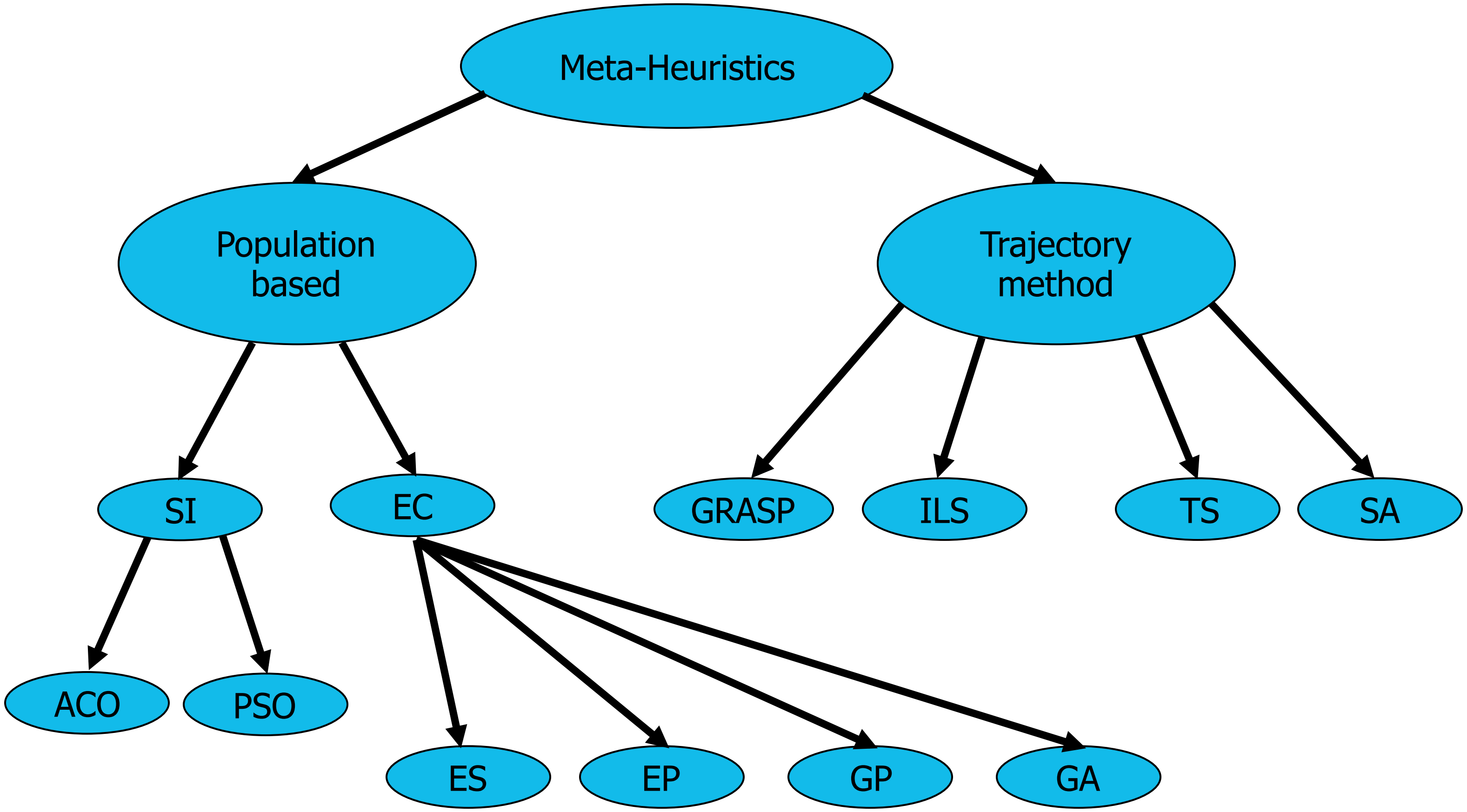
META-HEURISTICS

Meta-heuristics are algorithms that combine heuristics in a higher level framework aimed at efficiently and effectively exploring the search space,

Types:

- Trajectory methods, which handles one solution at a time,
- Population-based, which handles multiple solutions.

META-HEURISTICS



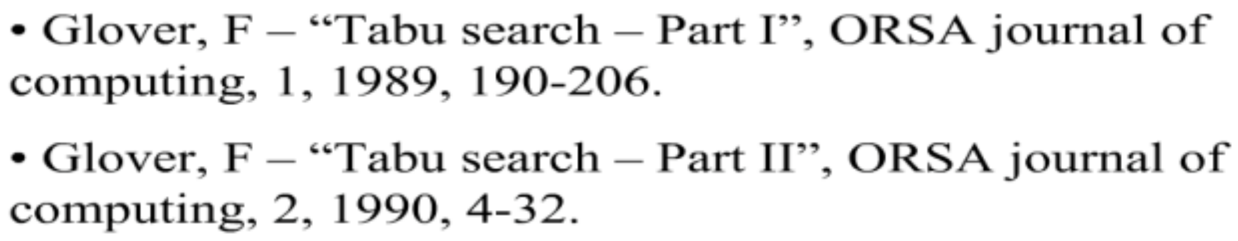
TABU SEARCH

Developed by Glover in 1989 [2,3],

Among the most cited meta-heuristics used for optimization,

It uses memory structures to:

- Escape local minima,
- Implement an explorative strategy. Avoid revisiting visited nodes.



- Glover, F – "Tabu search – Part I", ORSA journal of computing, 1, 1989, 190-206.
- Glover, F – "Tabu search – Part II", ORSA journal of computing, 2, 1990, 4-32.

LOCAL SEARCH STRATEGY

Local Search:

- Start with an initial feasible solution,
- While termination criterion not met
  - Generate a neighbouring solution by applying a series of local modifications (or moves),
  - If the new solution is better
    - Replace the old one,

end

LS CHALLENGES

One extreme is to consider all the possible neighbors of the current solutions (can be computationally demanding)

Another extreme is to only consider one neighbor (very limited horizon)

How can we consider a subset of the search space

How can we escape from local optimum

TABU SEARCH

TS can be regarded as the combination of LS and memory structures.

It is a meta-heuristic, a general strategy for guiding and controlling inner heuristics.

TS BASIC IDEA

Forbid or penalize moves which take the solution, in the next iteration, to points in the solution space previously visited (*tabu*).

Accepting non-improving solutions deterministically in order to escape from local optima where all the neighbouring solutions are non-improving.

TS- USE OF MEMORY

Uses of memory Structures:

- Short term memory based on recency of occurrence to prevent the search from revisiting previously visited solutions;
- Recency memory can also be used to keep track of good components to return to in order to intensify the search

TS-USE OF MEMORY

- Long term Memory based of frequency on occurrence of solution components from the start of the iterations.
- It is used to diversify the search and explore unvisited areas of the solution space by avoiding frequently visited components

TS – THE SHORT TERM MEMORY

The short-term memory is known as the *Tabu list*,

This memory usually holds a fixed and limited amount of information,

- Complete solutions, rarely used because of the space requirement,
- Recent moves applied to the current solutions, in order to prevent reverse moves.

TS- SHORT TERM MEMORY

Tabu lists length refer to the number of iterations *T* for which we keep a certain move (or its attributes) in the list (*tabu tenure*).

The list may record tabu active attributes

TS- NEIGHBORHOOD

- Like other search method we need a neighborhood structure N(s)
- In selecting a new state we consider neighbours that are not on the Tabu list N(s) - T(s)
- N(s) can be reduced and modified based on history and knowledge

TS- CANDIDATE LIST

Candidate lists are used to reduce the number of solutions examined on a given iterations

They isolate regions of the neighborhood containing moves with desirable features

EXAMPLE OF MOVES

- Toggle variable between 0 and 1
- Swap nodes in a routing tour
- Insert/delete edge in a graph
- Interchange variables

Usually the attributes are recorded in the memory and moves involving the attributes become tabu

TS – ASPIRATION

Sometimes it’s useful to allow a certain move even if it is tabu,

This could be done to prevent *stagnation*,

Approaches used to cancel the tabus are referred to as *aspiration criteria*,

TS – BASIC ALGORITHM

```
Generate an initial solution
While termination criterion not met
    ▪ Choose the best:
          $s' \in N(s) = \{N(s) - T(s)\} + A(s)$
    ▪ Memorize s' if it improves the best known solution
    ▪ s = s',
    ▪ Update T(s) and A(s)
end
```

Tabu list   Aspiration list

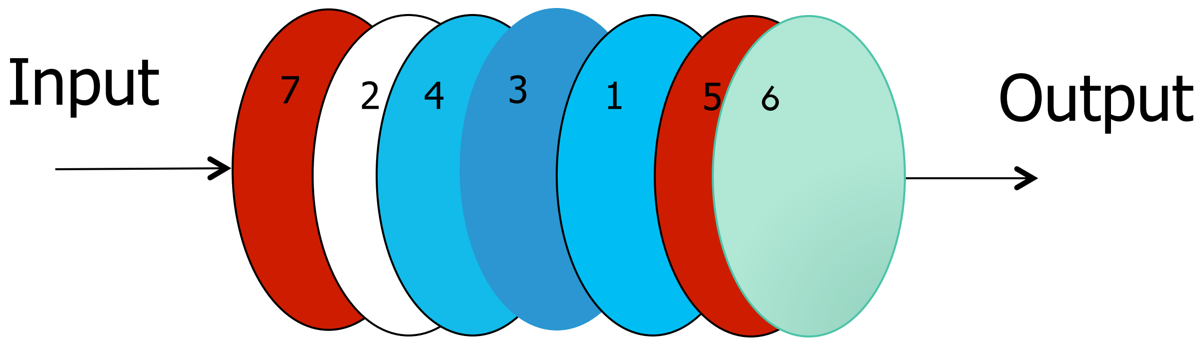
TERMINATION CONDITIONS

Some stopping conditions could be the following [8]:

1. no feasible solution in the neighborhood of current solution
2. reached the maximum number of iterations allowed.
3. The number of iterations since the last improvement is larger than a specified number.
4. Evidence shows that an optimum solution has been obtained.

TS EXAMPLES

INSULATING MODULES PROBLEM



Filtering Sequence

Ordering of filters determines the overall insulating performance

INSULATING MODULES PROBLEM

- Problem definition: Given the type of filters and a function that determines the insulating value of a given ordering, find the ordering of modules (filters) that maximizes the overall insulation
- Representation of a solution for 7 modules:

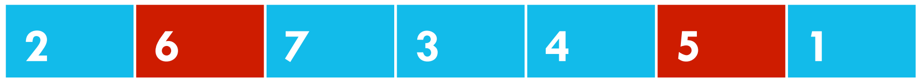
x_0 :

2	5	7	3	4	6	1

F(x_0)=10 units

NEIGHBORHOOD SIZE

- To consider all permutation will be *n!* for *n=7* it will be *5040*
- Neighborhood structure: swapping 2 modules

x_1 :


- A solution has 21 neighbors: 7 choose 2 (C_2^7)

F(x_1)= 8

NEIGHBORHOOD SIZE

As the number of allowed swaps increase, the neighbourhood size increases as well,

It is a computationally expensive task to generate the whole neighbourhood,

Sometimes only a reasonable part of the neighbourhood is generated.

DESIGNING TABU LIST DATA STRUCTURE

An important issue is how to design the tabu list without consuming a lot of memory,

A reasonable choice is to define the tabu list as an *nxn* matrix with the value of the *(i,j)* element denoting if the swap between filters *i* and *j* is permitted.

DESIGNING TABU LIST

If a swap is performed, the corresponding element in the matrix is set to *T*, the tabu list length,

After every iteration, all the non-zero elements are decreased by 1,

Hence this swap is considered tabu for *T* consecutive iterations.

SELECTION OF PARAMETERS

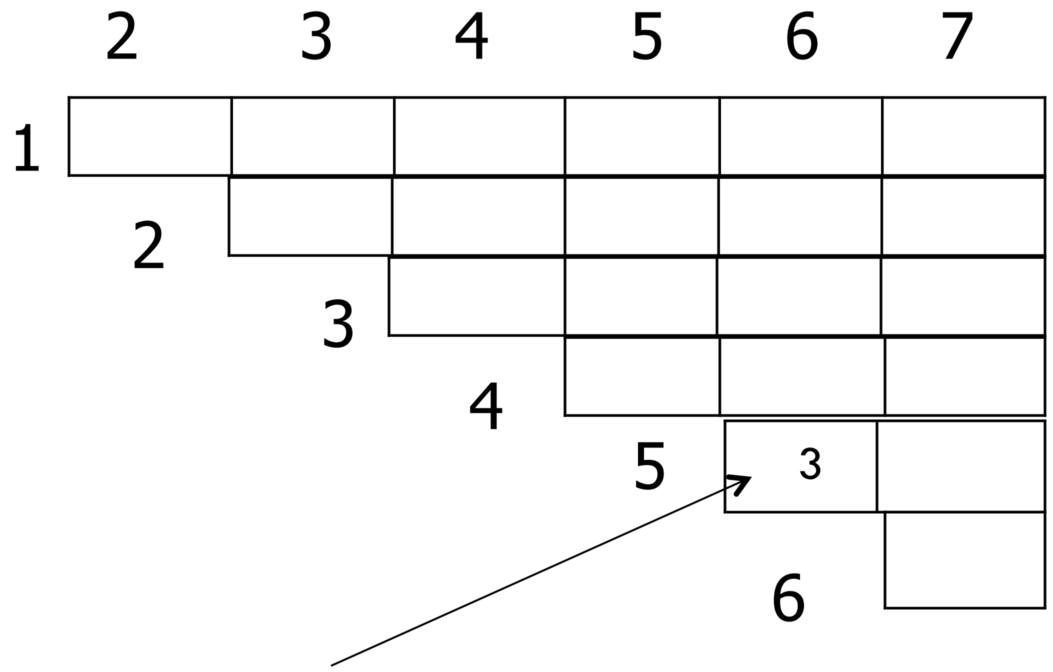
Tabu attributes are selected as most recently made swaps

Tabu tenure is set as 3 iterations

Hence, solutions involving 3 most recent swaps will be classified as tabu

Aspiration criterion is chosen as best solution

TABU LIST DATA STRUCTURE



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

Store tabu tenure for the pair (5,6)

ITERATION 0 (INITIAL SOLUTION)

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

Top 5 candidates

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

2	5	7	3	4	6	1

Select (5,4) as a move

Insulation value=10

ITERATION 1

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

Swap	Δvalue	
3,1	2	*
2,3	1	
3,6	-1	
7,1	-2	
6,1	-4	

(4,5) is tabu for 3 iterations

2	4	7	3	5	6	1

Value=16

Select (3,1)

ITERATION 2

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

Reduced by 1

Swap	Δvalue	
1,3	-2	T
2,4	-4	*
7,6	-6	
4,5	-7	T
5,3	-9	

2	4	7	1	5	6	3

Value = 18

ITERATION 3

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

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

4	2	7	1	5	6	3

Value= 14
Best so far was 18

(4,5) is tabu but will achieve best solution
Select using aspiration

ITERATION 4

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

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

5	2	7	1	4	6	3

Value=20

Select (7,1)

EXAMPLES FOR TABU RESTRICTIONS, TABU TENURE (LENGTH) AND ASPIRATION

Restrictions:

- A move that involves the same exchange of positions of a tabu move
- A move that involves any of the positions that were involved in a tabu move
- Same for adding and deleting rather than exchanging.

TABU TENURE (LENGTH)

Static: choose *T* to be a constant or as a guideline *(problem size)* $\sqrt{n}$

Dynamic: choose *T* to vary randomly between *Tmin* and *Tmax*

Can make the threshold *Tmin and Tmax* vary for attributes.

TABU LENGTH

Fixed length tabu lists cannot always prevent cycling,

There could exist some cycles with lengths longer than the tabu tenure,

Some researchers proposed the use of tabu lists with varying lengths during the search,

ASPIRATION CRITERIA

By Default

- A tabu move becomes admissible if it yields a solution that is better than any obtained solution so far

By Objective

- A tabu move becomes admissible if it yields a solution that is better than an aspiration value

By Search Direction

- A tabu move becomes admissible if the direction of the search (improving or non-improving) does not change

TS – INTENSIFICATION AND DIVERSIFICATION

Two important concepts in TS are:

- *Intensification*, which refers to the process of exploiting a small portion of the search space, or penalizing solutions that are far from current solution.
- *Diversification*, which refers to forcing the search into previously unexplored areas of the search space, or penalizing solutions that are close to current.

TS – INTENSIFICATION AND DIVERSIFICATION

Intensification is not always necessary as sometimes the search process is thorough enough,

It can be based on *recency memory*,

The memory holds the number of consecutive iterations in which certain components always appear in the current solution without interruption.

TS – INTENSIFICATION AND DIVERSIFICATION

After a while, one could stop the current search process and move into an intensification phase,

This phase tries to locally optimize the best known solution while keeping those components intact.

TS – INTENSIFICATION AND DIVERSIFICATION

Despite the use of tabu lists, TS may tend to be too local sometimes,

This problem causes TS to miss some good solutions in unexplored search space areas,

Diversification aims to overcome this problem.

TS – INTENSIFICATION AND DIVERSIFICATION

Having a proper diversification strategy is very critical to the success of the TS,

It is based on a long-term memory referred to as the *frequency memory*,

The memory holds the total number of iterations (since the beginning of the search) in which certain components always appear in the current solution or are involved in the selected moves,

TS – INTENSIFICATION AND DIVERSIFICATION

Two major approaches exist for applying the diversification process:

- *Restart diversification*, which forces components that rarely appear in the current solution and restart the search from these points,
- *Continuous diversification*, which bias the evaluation of possible moves by adding to the objective function a term related to component frequency. Thus incorporating diversification into the regular search process.
- One can use the approach of penalizing solutions that are close to current solution

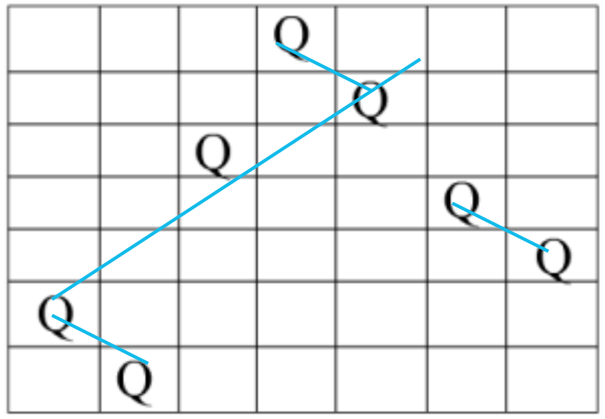
N-QUEENS PROBLEM

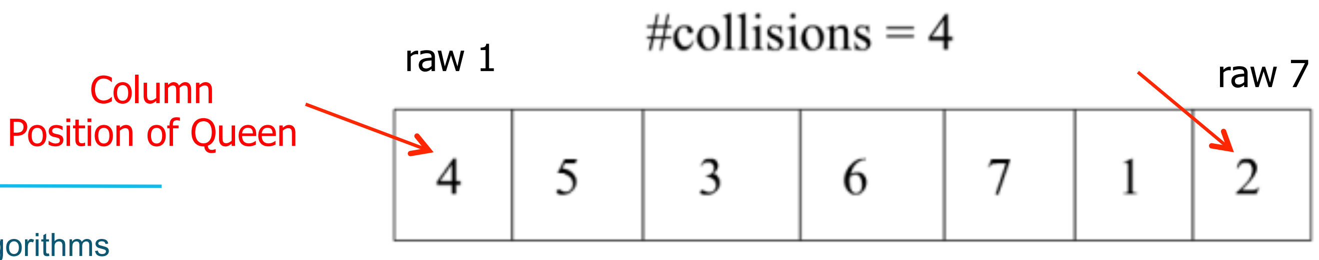
The n-queens problem consists of placing n queens on a n x n chessboard in such a way that no two queens capture each other.

In order for this to happen,

- no two queens should be placed
  on the same row,
  on the same column, or
  on the same diagonal,
- If two queens are placed such that they are able to capture each other,

it is said that a "collision" has occurred.





4	5	3	6	7	1	2

Iteration 0 (Starting Point)

Current solution

4

5	3	6	7	1	2

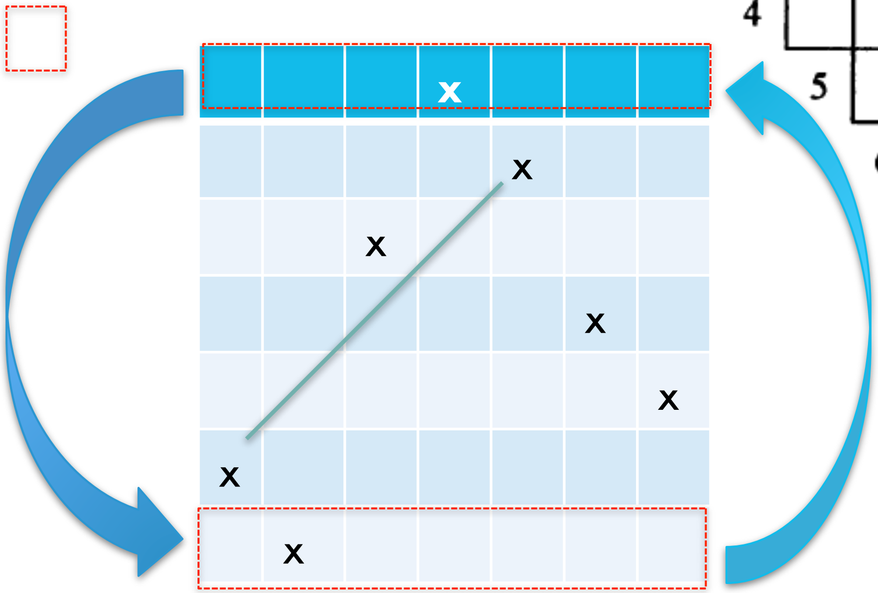
Number of collisions = 4

Tabu structure

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

Top 5 candidates

Swap		Value	
1	7	-2	*
2	4	-2	
2	6	-2	
5	6	-2	
1	5	-1	

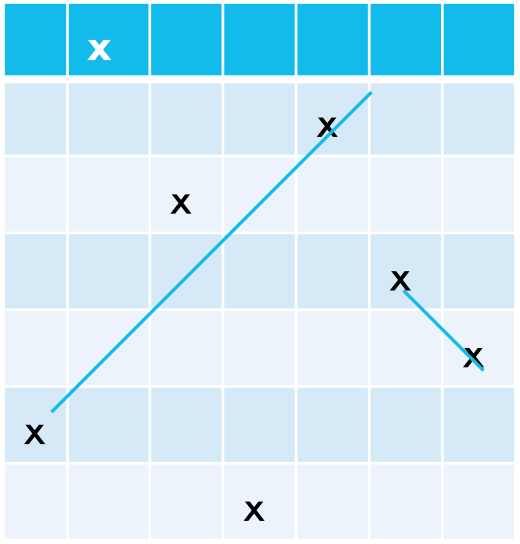


Iteration 1

Current solution

2	5	3	6	7	1	4

Number of collisions = 2

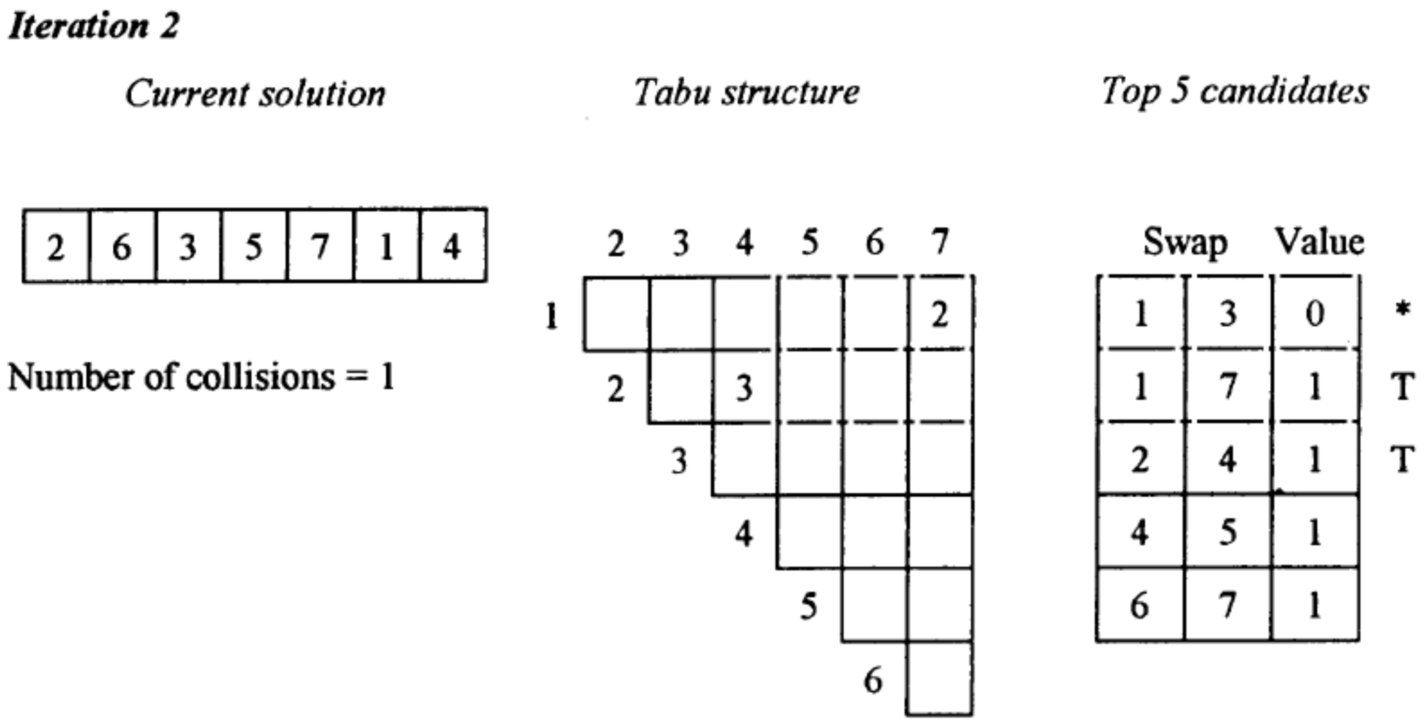


Tabu structure

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

Top 5 candidates

Swap		Value	
2	4	-1	*
1	6	0	
2	5	0	
1	2	1	
1	3	1	



Iteration 2

Current solution

2	6	3	5	7	1	4

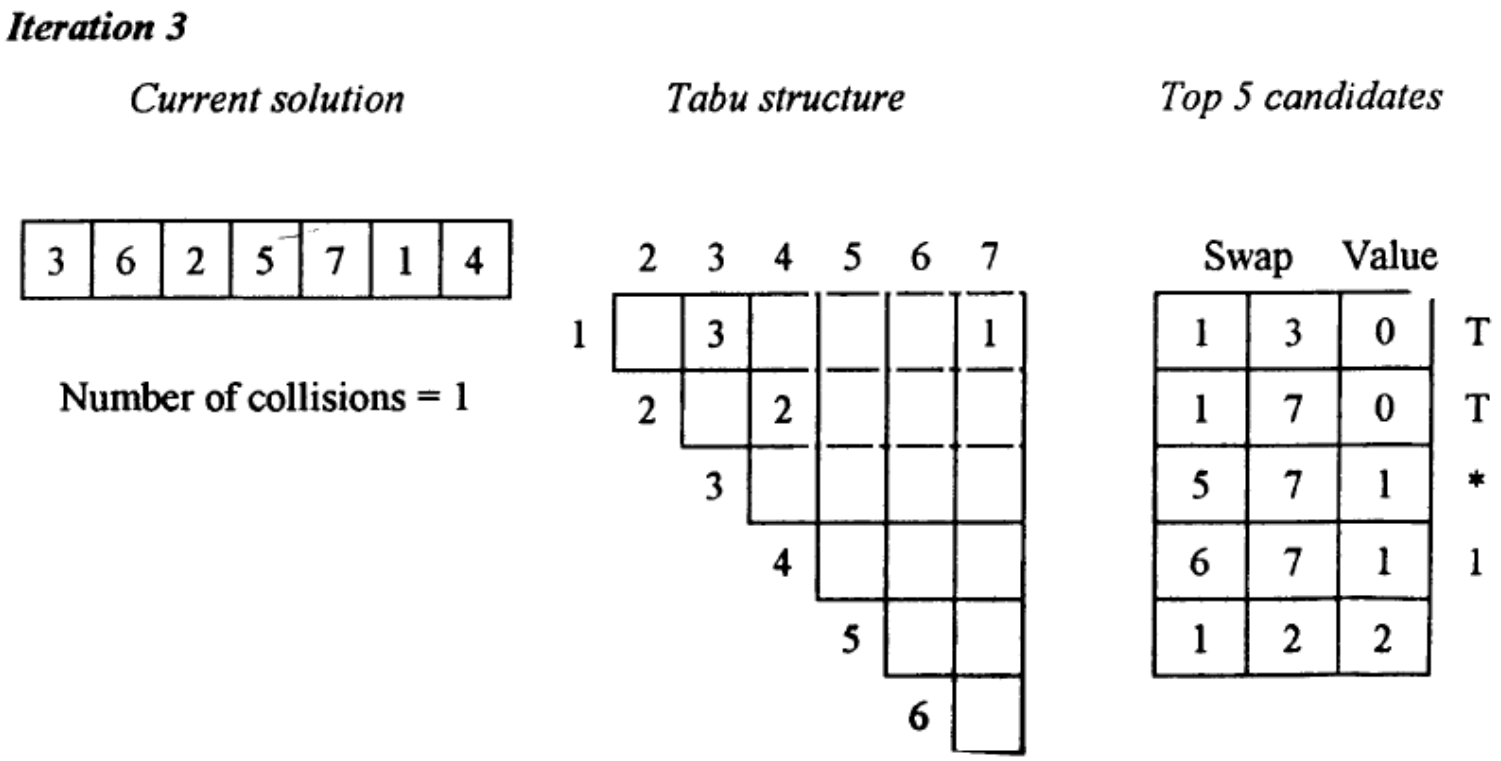
Number of collisions = 1

Tabu structure

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

Top 5 candidates

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



Iteration 3

Current solution

3	6	2	5	7	1	4

Number of collisions = 1

Tabu structure

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

Top 5 candidates

Swap		Value	
1	3	0	T
1	7	0	T
5	7	1	*
6	7	1	1
1	2	2	

Iteration 4

Current solution

3	6	2	5	4	1	7

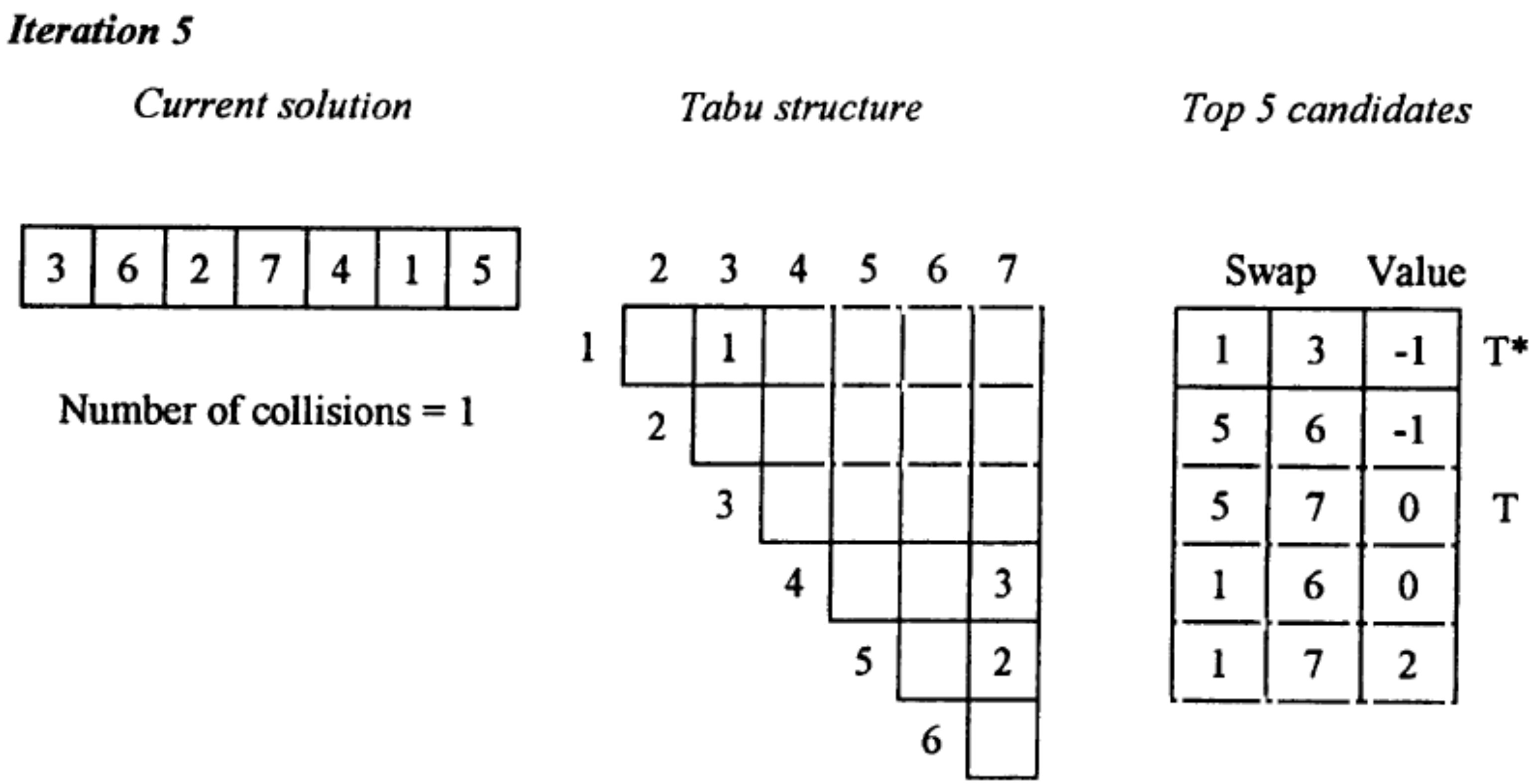
Number of collisions = 2

Tabu structure

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

Top 5 candidates

Swap		Value	
4	7	-1	*
5	7	-1	T
1	5	0	
2	5	0	
2	4	2	T



Iteration 5

Current solution

3	6	2	7	4	1	5

Number of collisions = 1

Tabu structure

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

Top 5 candidates

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

Iteration 26

Current solution

1	3	6	2	7	5	4

Number of collisions = 1

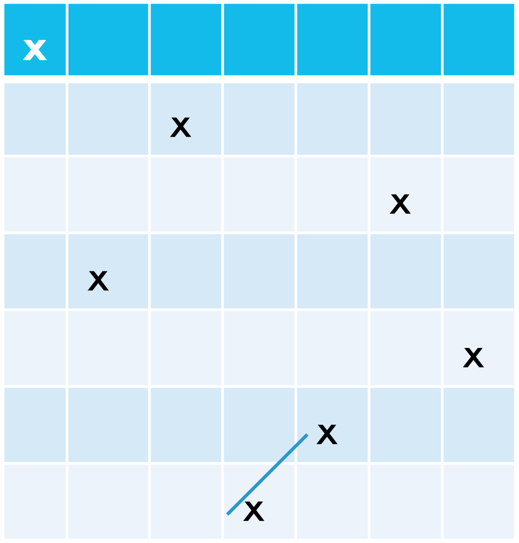
Tabu structure

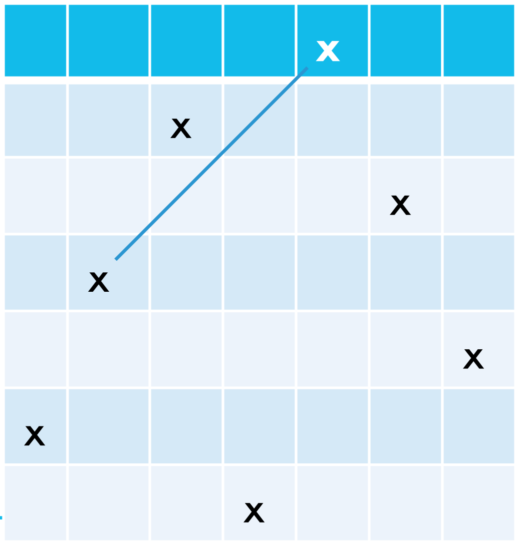
	1	2	3	4	5	6	7
1	▨					3	
2		▨					
3	5		▨			2	
4		3		▨			1
5	2			4	▨		
6	1				2	▨	
7		4	3	1			▨

Frequency

Top 5 candidates

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



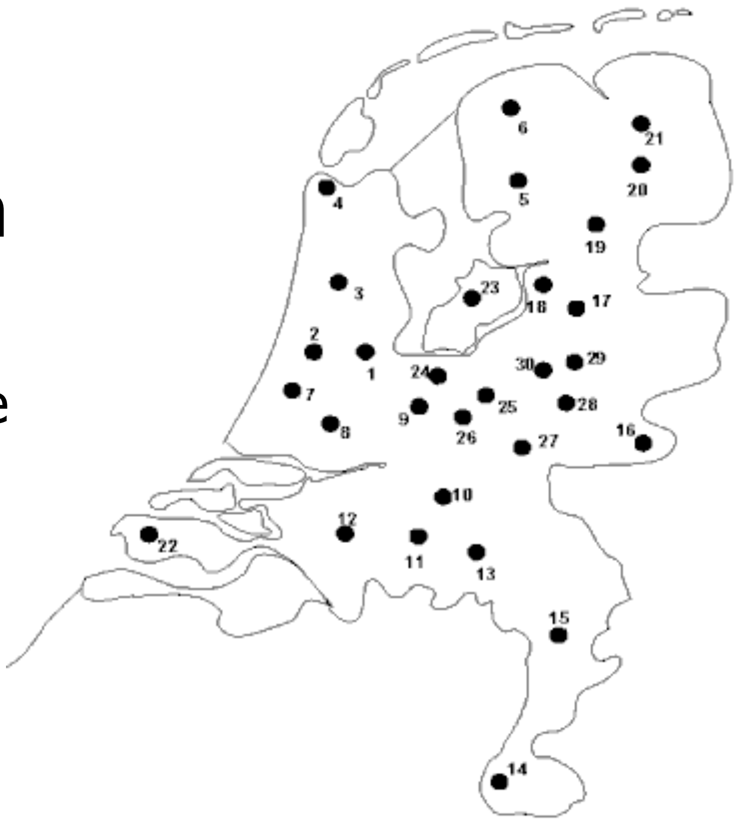


TS FOR TSP

Problem:

- Given n cities,
- Find a complete tour with minimal length.

Search space is BIG: for 30 cities there are 30! 10^{32} possible tours.



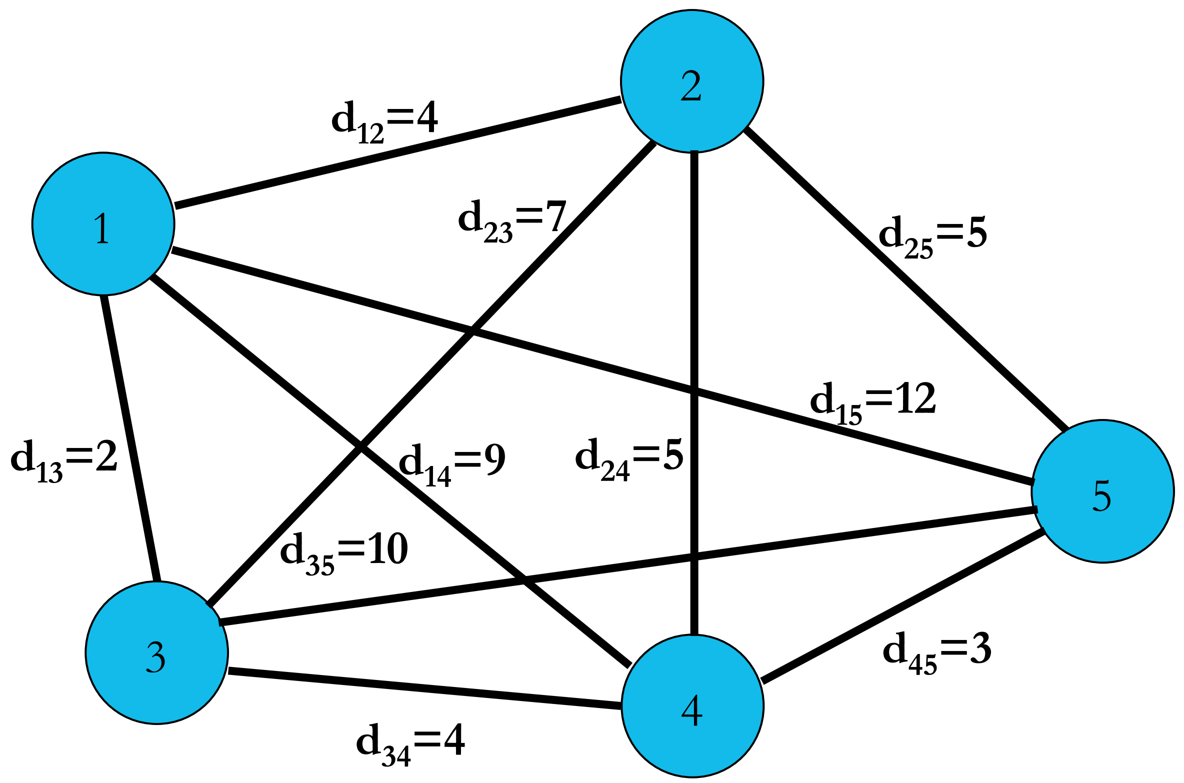
TS FOR TSP

The solution is a permutation of n cities:

Several choices exist for generating the required neighbourhood.

$$Sol = \left[\ 1,\ 2,\ 3,\ ..,\ n\ \right]$$

TS FOR TSP - EXAMPLE



TS FOR TSP - EXAMPLE

Start with a random solution :

$$Sol = [\ 1,\ 3,\ 4,\ 2,\ 5\ ]$$

For a closed tour

The total distance would be:

$$total\_dist = \sum_{i=1}^{n-1} d_{Sol_i Sol_{i+1}} + \boxed{d_{Sol_n Sol_1}}$$

TS FOR TSP - EXAMPLE

$$Sol = [\ 1,\ 3,\ 4,\ 2,\ 5\ ]$$

The tour length of the initial solution is:

$$total\_dist = 2 + 4 + 5 + 5 + 12 = 28$$

With swap of two cities , the neighbourhood size of each solution is 10
{1-3,1-4,1-2,1-5,3-4,3-2,3-5,4-2,4-5,2-5}

Considering these 10 candidates, the best is to swap city 1 and 4

The tour length of the candidate solution is

$$Sol = [\ 4,\ 3,\ 1,\ 2,\ 5\ ]$$

$$total\_dist = 4 + 2 + 4 + 5 + 3 = 18$$

TS FOR TSP

If we define the neighbourhood by performing a predetermined number of swaps on the current solution:

Number of cities	**Neighbourhood size (one swap)**
14	91
29	406
52	1326

TS FOR TSP

The TS was applied for the Ulysses16 (16 city) TSP instance by:

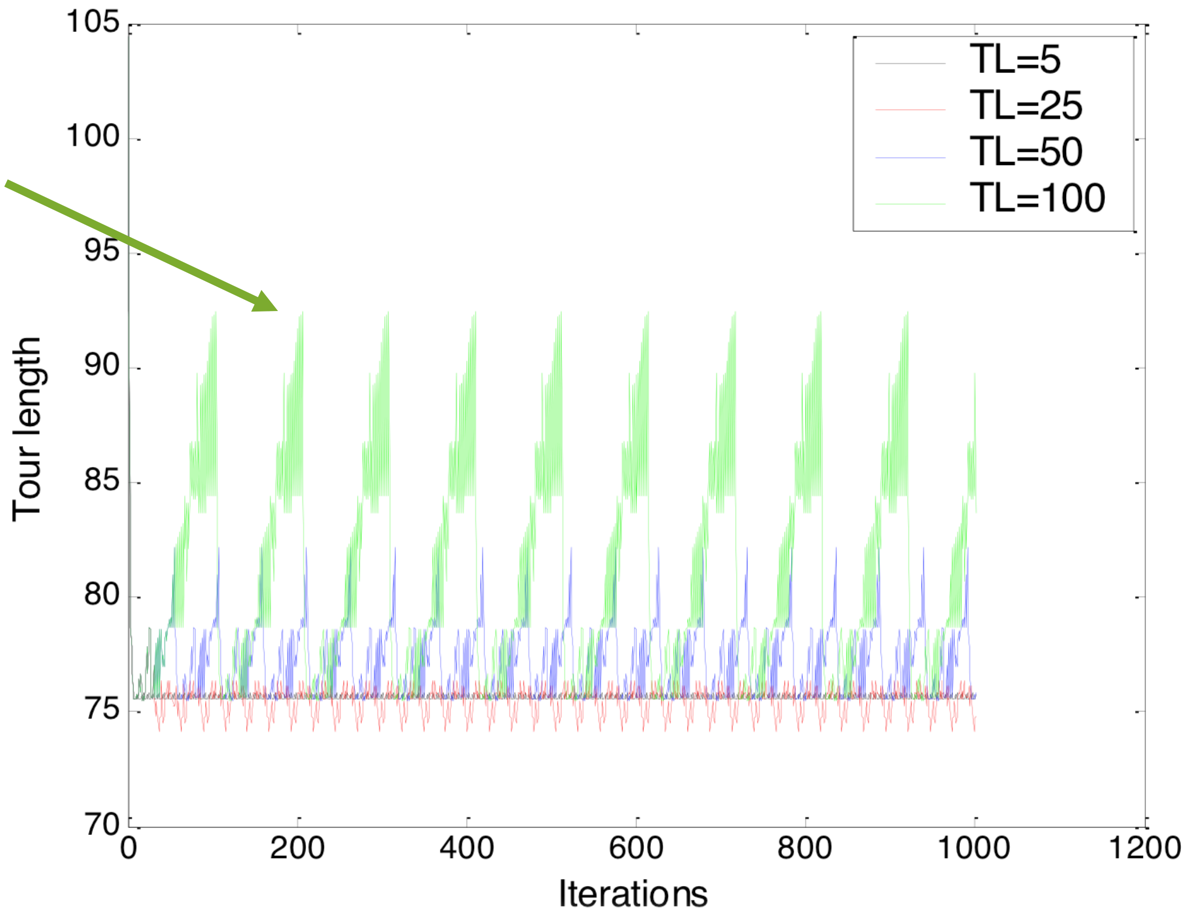
- Exploring the complete neighbourhood of the current solution,
- Performing 1000 iterations,
- Using different tabu list lengths (5, 25, 50 and 100),
- No aspiration criteria used,

TS FOR TSP

Tabu list length	**Tour length**
5	75.5624
25	**74.1445**
50	75.4771
100	75.4771

TS FOR TSP

Longer cycles are detected when using longer tabu lists

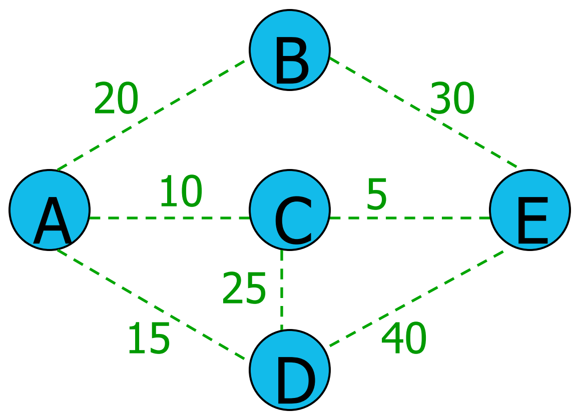


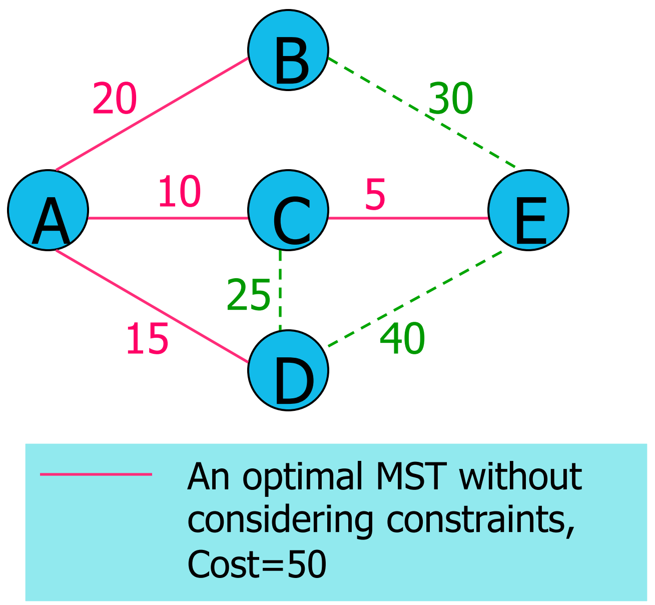
MIN SPANNING TREE WITH CONSTRAINTS

Finding a spanning tree that also satisfies extra constraints (on edges or nodes or both)

Example constraints: certain edges (or nodes) are inclusive (have to occur together); edges (or nodes) are exclusive only some of them need to be in the tree.

EXAMPLE MST WITH CONSTRAINTS [3]





Constraints 1: Link AD can be included only if link DE also is included. (penalty:100)
Constraints 2: At most one of the three links – AD, CD, and AB – can be included. (Penalty of 100 if selected two of the three, 200 if all three are selected.)

TS MODEL

$N(x)$ is defined by edge exchange moves

TL: the inserted edge

Tabu tenure = 2

Aspiration criterion: Best Objective

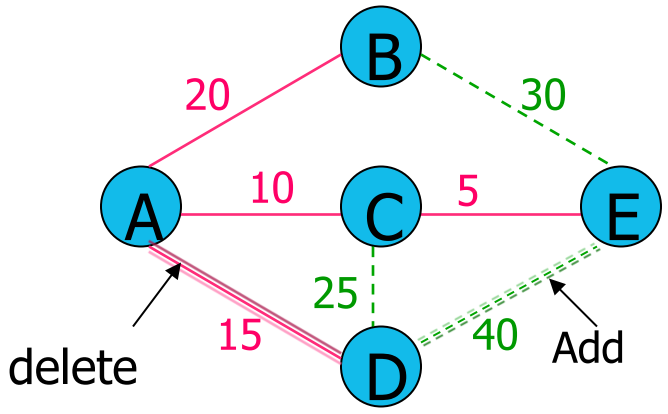
Use MST without constraints as initial solution

ITERATION 0

- Cost of initial solution=50+200 (constraints penalties)

C1: AD if DE

- C2 one of AB, AD, CD



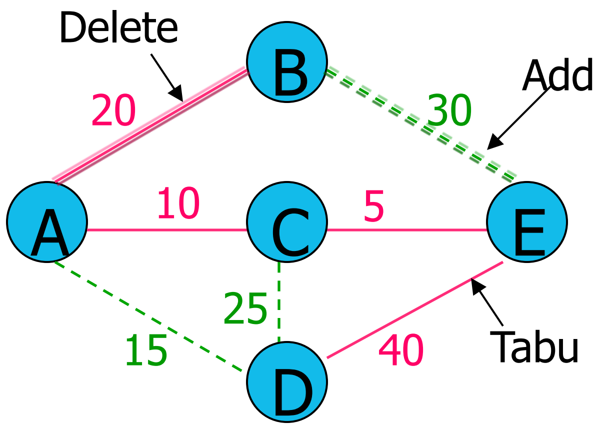
Add	Delete	Cost
BE	CE	75+200=275
BE	AC	70+200=270
BE	AB	60+100=160
CD	AD	60+100=160
CD	AC	65+300=365
DE	CE	85+100=185
DE	AC	80+100=180
DE	AD	**75+0=75**

Select adding DE and deleting AD
Put DE to Tabu list
Iteration 1 starts with MST of cost 75

ITERATION1

Tabu list: DE, T=2

Cost=75



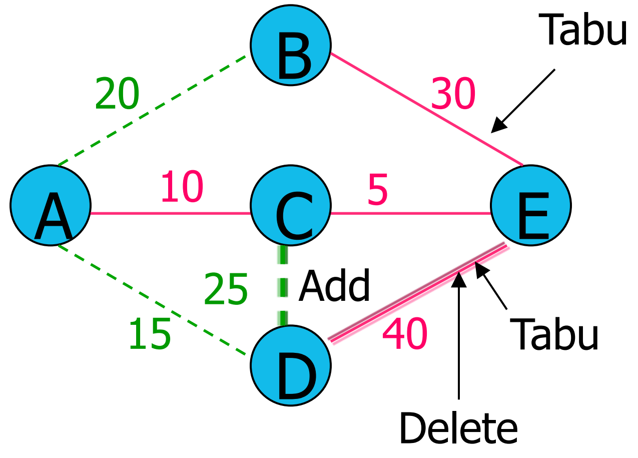
Add	Delete	Cost
AD	DE*	Tabu move
AD	CE	85+100=185
AD	AC	80+100=180
BE	CE	100+0=100
BE	AC	95+0=95
BE	AB	**85+0=85**
CD	DE*	60+100=160
CD	CE	95+100=195

- C1: AD if DE
- C2 one of AB, AD, CD

- ***A tabu move***
- ***Select adding BE and deleting AB***
- Iteration 2 starts with MST of cost = 85
- Escape local optimum

ITERATION 2

Tabu list: DE , T=1, BE, T=2
Iteration 3 Cost=85



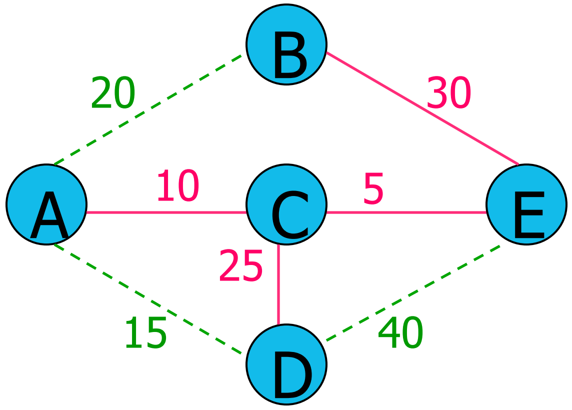
Add	Delete	Cost
AB	BE*	Tabu move
AB	CE	100+0=100
AB	AC	95+0=95
AD	DE*	60+100=160
AD	CE	95+0=95
AD	AC	90+0=90
CD	DE*	**70+0=70**
CD	CE	105+0=105

**** A tabu move will be considered only if it would result in a better solution than the best trial solution found previously (Aspiration Condition)***

- C1: AD if DE
- C2 one of AB, AD, CD

Iteration 3 new cost = **70** Override tabu status

ITERATION 3



Optimal Solution

Cost = 70

Additional iterations only find inferior solutions

ASSIGNMENT PROBLEM

Linear Assignment Problem formulated by Hanan and Kurtzberg [1972],

involves the assignment or *n* objects to *n* sites.

For each assignment, there is a related cost, , of assigning object *i* to site *j*.

The objective is to assign each object to one c_{ij} and only one site in such a manner that minimizes the sum of each assignment cost, i.e., the total cost.

- Mathematically, the above problem can be formulated as follows:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

over all permutations of {1, 2, . . . , n},

Subject to

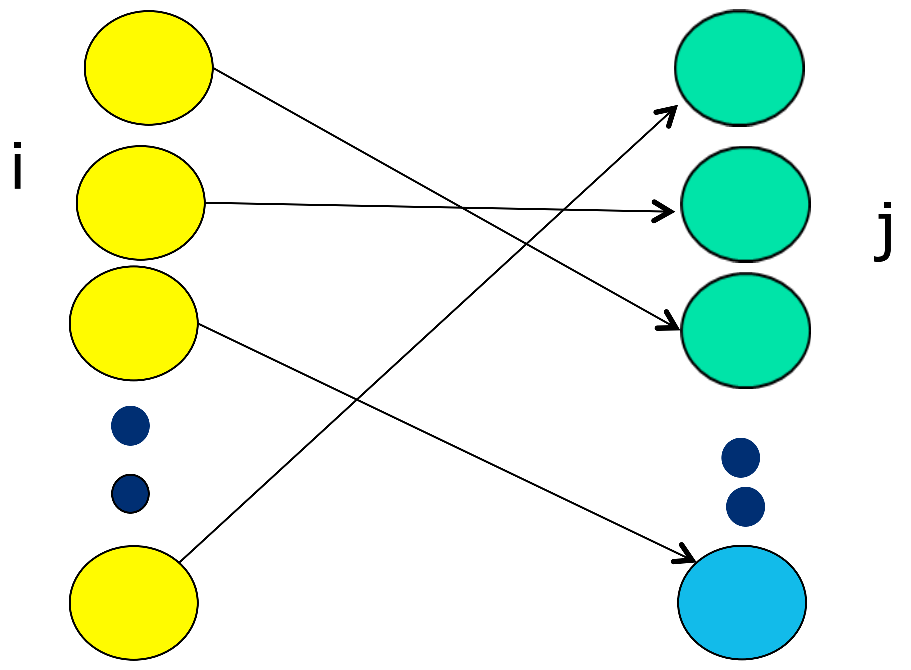
$$\sum_{i=1}^{n} x_{ij}, = 1, j = 1,..n, \sum_{j=1}^{n} x_{ij} = 1, i = 1,..n$$

$$x_{ij} = 0 or 1$$

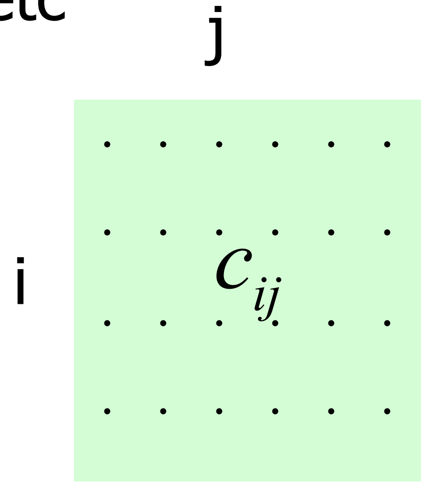
Notice that each set of assignments is a permutation of a set on n integers; hence, there are n! distinct permutations from which to choose the optimal assignment

LINEAR ASSIGNMENT

Set of Objects (persons, facility, etc)



Set of sites Jobs, locations, etc



Bipartite graph

APPLICATIONS

assignment of offices or services in buildings (e.g. university campus, hospital, etc.),

assignment of the departure gates to airplanes in an airport, the placement of logical modules in electronic circuits,

distribution of the files in a database

placement of the keys of keyboards of typewriters.

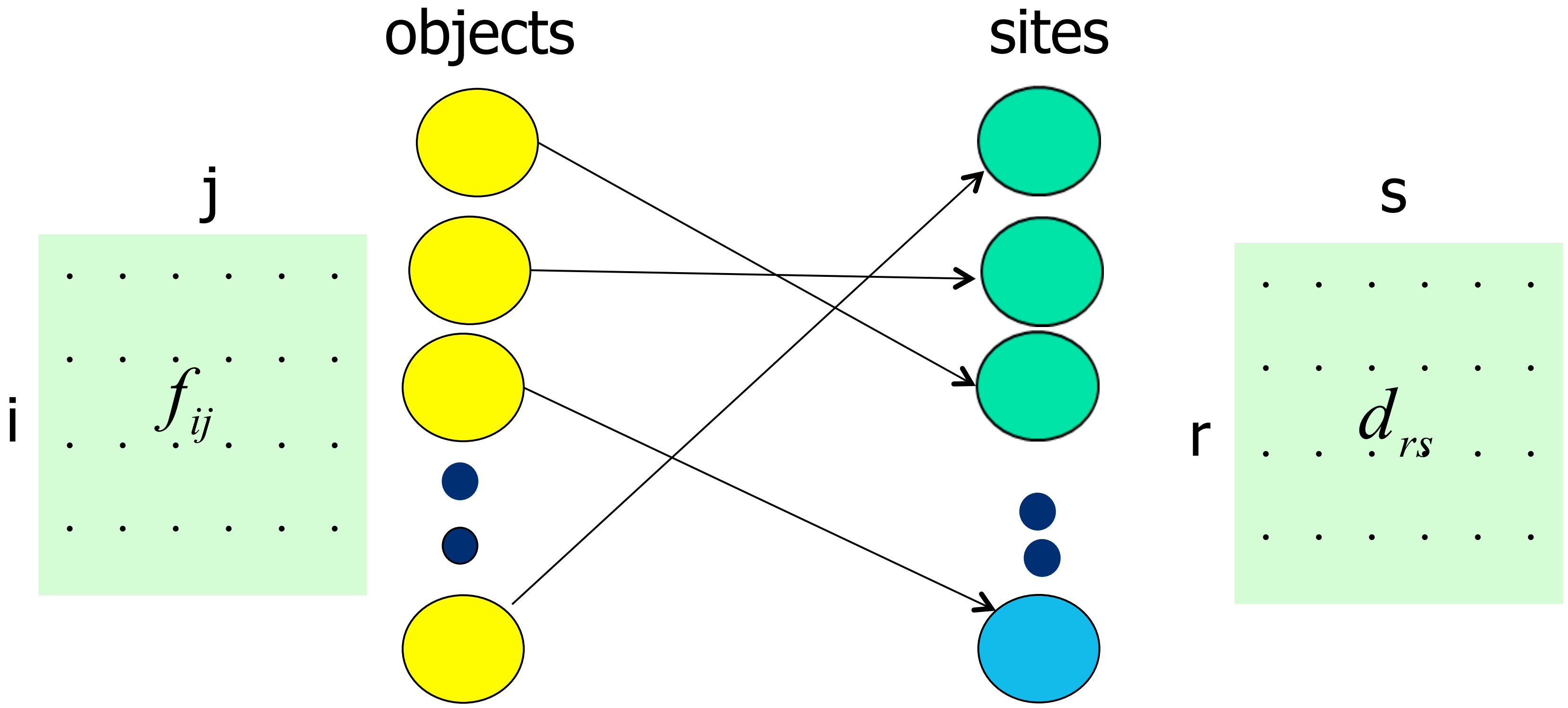
QAP
QUADRATIC ASSIGNMENT PROBLEM

Given *n* objects and the flows f_{ij} between object *i* and object *j* (*i*, *j* = 1. . . *n*), and given *n* sites with distance d_{rs}

between the sites *r* and *s* (*r*, *s* = 1. . . *n*), the problem deals with placing the *n* objects on the *n* sites so as to minimize the sum of the products, flows × distances. Mathematically, this is equivalent to finding a permutation **p**, whose p_i *ith* component denotes the object assigned to site *i*, which minimizes

$$\sum_{i=1}^{n} \sum_{j=1}^{n} f_{p_i p_j} d_{ij}$$

QAP



Bipartite graph

QAP: EXAMPLE

Let us consider the small 5×5 instance of quadratic assignment problem,

known in literature as NUG5, with flows matrix F and distances matrix D :

F

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

D

0	1	1	2	3
1	0	2	1	2
1	2	0	1	2
2	1	1	0	1
3	2	2	1	0

ITERATION 0

On the basis of the initial solution $\mathbf{p} = (2, 4, 1, 5, 3)$ which means object 2 is assigned to site 1, object 4 to site 2,… etc. of cost 72, the following evolution of tabu search will take place. The search can be started

by initializing a tabu matrix $T = \mathbf{0}$

HOW THE COST CALCULATED

Take the permutation and apply it to the matrix F (rows and columns) then multiply corresponding rows of permuted F and D

P= (2, 4, 1, 5, 3

F	PF	PFP
0 5 2 4 1	5 0 3 0 2	0 0 5 2 3
5 0 3 0 2	4 0 0 0 5	0 0 4 5 0
2 3 0 0 0	0 5 2 4 1	5 4 0 1 2
4 0 0 0 5	1 2 0 5 0	2 5 1 0 0
1 2 0 5 0	2 3 0 0 0	3 0 2 0 0

$$\sum_{i=1}^{n}\sum_{j=1}^{n} f_{p_i p_j} d_{ij}$$

2 to site 1, 4 to site 2, 1 to site 3,.
f24*d12+f21*d13+…

- Take the permutation and apply it to the matrix F (rows and columns) then multiply corresponding rows of permuted F and D

PFP	D	PFP*D
0 0 5 2 3	0 1 1 2 3	5+4+9 =18
0 0 4 5 0	1 0 2 1 2	8+5 =13
5 4 0 1 2	1 2 0 1 2	5+8+1+4=18
2 5 1 0 0	2 1 1 0 1	4+5+1 =10
3 0 2 0 0	3 2 2 1 0	9+4 =13

----

cost= 72

ITERATION P= (2, 4, 1, 5, 3)

calculate value $\Delta(\mathbf{p}, (i, j))$ is calculated for each transposition (i, j):

move	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(2, 4)	(2, 5)	(3, 4)	(3, 5)	(4, 5)
Cost	2	−12	−12	2	0	−10	−12	4	8	6

It can be realized that three moves produce a maximum profit of 12: to exchange the objects on the sites (1, 3), (1, 4) and (2, 5).

P= (2, 4, 1, 5, 3)

One can assume that it is the first of these moves, (1, 3), which is retained. If it is prohibited during *t* = 6 iterations to put element 2 on site 1 and element 1 on site 3 simultaneously, the following matrix is obtained:

TABU LIST: P= (2, 4, 1, 5, 3)
SITE 1 3

$$T \quad \begin{array}{cccccc} & & & \text{sites} & & \\ \text{elements} & 0 & 0 & 6 & 0 & 0 \\ & 6 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 \end{array}$$

ITERATION 2.

The move chosen during iteration 1 leads to the solution **p** = (1, 4, 2, 5, 3) of cost 60. Calculating the value of each transposition one can obtain:

move	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(2, 4)	(2, 5)	(3, 4)	(3, 5)	(4, 5)
cost	14	12	−8	10	0	10	8	12	12	6
tabu		*yes*								

For this iteration, it should be noticed that the reverse of the preceding move is now prohibited.

P = (1, 4, 2, 5, 3)

- The authorized move (1, 4), giving minimum cost, is retained, for a profit of 8.
- tabu duration of the reverse move

is $t = 6$, the matrix T becomes:

P = (1, 4, 2, 5, 3), (1,4)

T=

$$\begin{matrix} 6 & 0 & 5 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 0 \end{matrix}$$

ITERATION 3.

The solution **p** = (5, 4, 2, 1, 3) of cost 52 is reached which is a local optimum. Indeed, at the beginning of iteration 3, no move has a negative cost:

move	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(2, 4)	(2, 5)	(3, 4)	(3, 5)	(4, 5)
cost	10	24	8	10	0	22	20	8	8	14
tabu			*yes*							

P = (5, 4, 2, 1, 3)

The selected move (2, 3) in this iteration has a zero cost. It should be noticed here that the move (1, 3), which was prohibited during iteration 2 is again authorized, since the element 5 was never in third position.

tabu duration results in $t = 6$, the following matrix is obtained:

P = (5, 4, 2, 1, 3), (2,3)

T=

$$\begin{matrix} 5 & 0 & 4 & 0 & 0 \\ 4 & 0 & 6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 \end{matrix}$$

ITERATION 4.

One can then obtain a solution **p** = (5, 2, 4, 1, 3) of cost 52 and the data structures situation will be as follows:

move	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(2, 4)	(2, 5)	(3, 4)	(3, 5)	(4, 5)
cost	24	10	8	10	0	8	8	22	20	14
tabu			*yes*		*yes*					

P = (5, 2, 4, 1, 3)

However, it is not possible any more to choose the move (2, 3) corresponding to the minimum cost, which could bring us back to the preceding solution,

Similar situation arises for the move (1, 4) which would exchange element 5 in position 1 and element 1 in fourth position, although this exchange leads to a solution not yet visited.

P = (5, 2, 4, 1, 3)

Hence we are forced to choose an unfavorable move (2, 4), that increases the cost of the solution by 8. With a selected tabu duration of $t = 6$, one can obtain:

P = (5, 2, 4, 1, 3), (2,4)

T=

4 0 3 6 0

3 6 5 0 0

0 0 0 0 0

0 5 0 0 0

0 0 0 4 0

ITERATION 5.

The solution at the beginning of this iteration is: **p** = (5, 1, 4, 2, 3) with cost 60

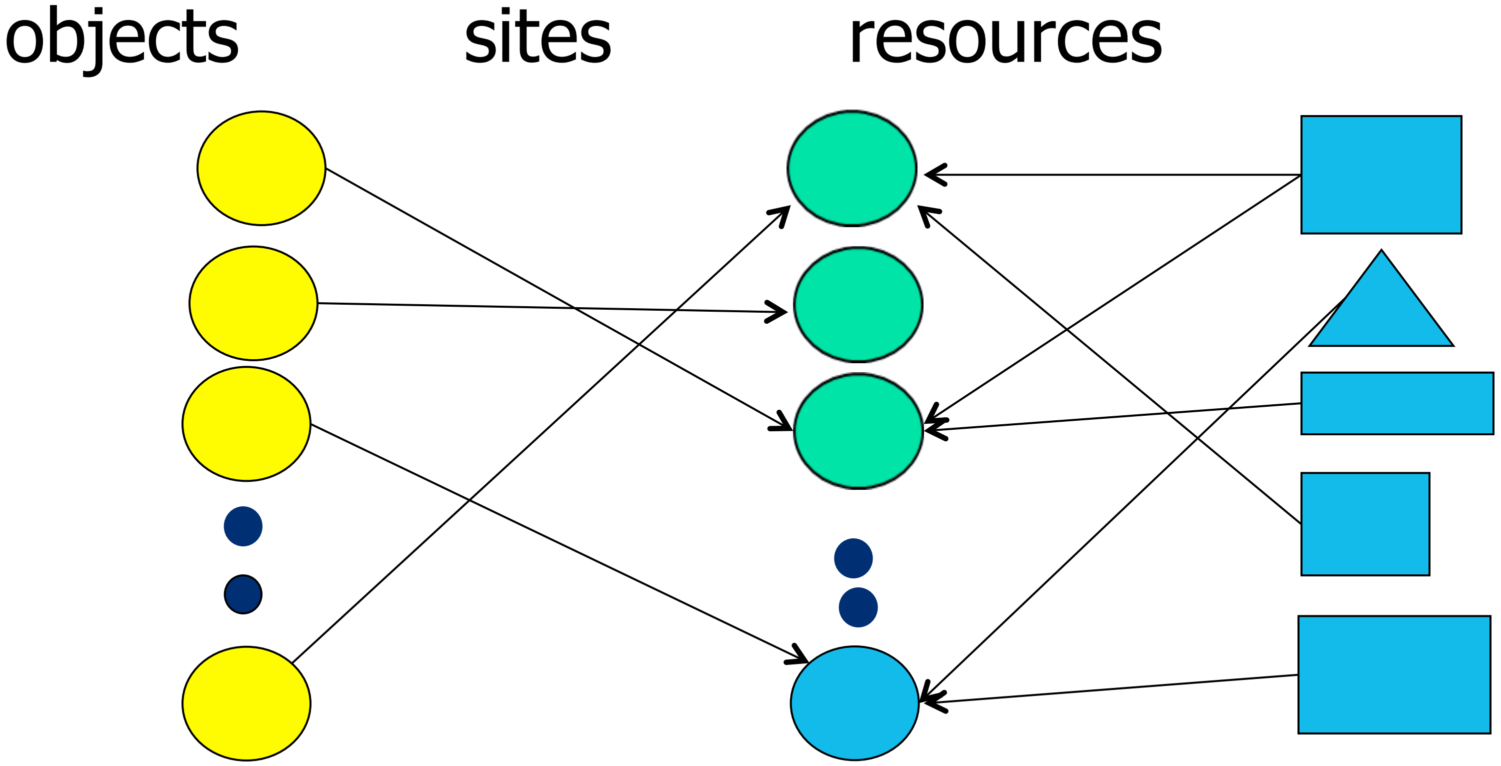
The calculation of the cost of the moves gives:

move	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(2, 4)	(2, 5)	(3, 4)	(3, 5)	(4, 5)
cost	12	−10	12	10	0	−8	4	14	20	10
tabu			*yes*		*yes*	*yes*				

Select (1,3)

It is noticed that the move degrading the quality of the solution at the preceding iteration was beneficial, because it now facilitates to arrive at an optimal solution **p** = (4, 1, 5, 2, 3) of cost 50, by choosing the move (1, 3).

GAP, GQAP



REFERENCES

1. S. Russel and P. Norving."AI: A modern approach", Prentice Hall, 2003
2. Glover, F. and Laguna, M. 1997. Tabu Search. Norwell, MA: Kluwer Academic Publishers.
3. Hillier, F.S. and Lieberman, G.J. 2005. Introduction to Operations Research. New York, NY: McGraw-Hill. 8th Ed

ADAPTATION

One of the most important parameters of the TS is the length of the tabu list,

If the length is too small, the a search can be trapped into cycles,

If the length is too large, too many moves could be prevented at each iteration.

ADAPTATION

One of the adaptive approaches incorporated into TS is to allow the length of the short term memory (tabu list) to vary dynamically,

These approaches were introduced by Taillard [3] and Dell' Amico and Trubian [4].

ADAPTATION

In [3], the adaptive approach was as follows:

- A range for the tabu list length in computed in advance according to the problem size,
- A new list length is randomly selected from this range every predetermined number of iterations.

ADAPTATION

In [4], the adaptive approach was as follows:

- The tabu list length is restricted between L_{min} and L_{max},
- If the current solution is better than the best-so-far, the tabu list length is set to 1,
- If in an improving phase (the solution has improved over the last iteration) the tabu list length is decreased by 1,

ADAPTATION

- If not in an improving phase (the solution has deteriorated over the last iteration) the tabu list length is increased by 1,
- The values of L_{min} and L_{max} are randomly changed every A iterations.

ADAPTATION

The idea behind the approach is that if the current solution has improved the tabu list length is decreased in order to focus the search in a region of potential improvement,

On the other hand, if the current solution has deteriorated the tabu list length is increased in order to guide the search away from an apparently bad region.

COOPERATION

In Crainic et al. [5], the authors studied the idea of having *p* independent tabu searches working concurrently and exchanging information every predetermined number of iterations (*synchronous communication*),

This approach was referred to as *coordinated searches*.

COOPERATION

The different search processes may use different:

- Initial solutions,
- Search strategies (parameter settings),
- Ways of handling the incoming solution:
  - Replacing its own best-so-far solution (forced diversification), referred to as *simple import*,
  - Replacing its own best-so-far solution only if the incoming solution is better), referred to as *conditional import*.

COOPERATION

Another implementation is found in Malek et al. [6], where the tabu lists are reset after any synchronization step.

COOPERATION

General conclusions found were:

- For a fixed number of iterations, increasing the number of processors improved the obtained solution up to a certain point,
- Reducing the number of iterations between synchronization steps increases the computational times because of the increased message passing overhead,
- Conditional import always produces better or similar results to simple import.

COOPERATION

An *asynchronous communication* approach was proposed in [10],

Instead of exchanging information every predetermined number of iterations, each search process only broadcasts information when its best-so-far solution is updated,

A *central memory* is used to handle the information exchange.

COOPERATION

The mechanism used is as follows:

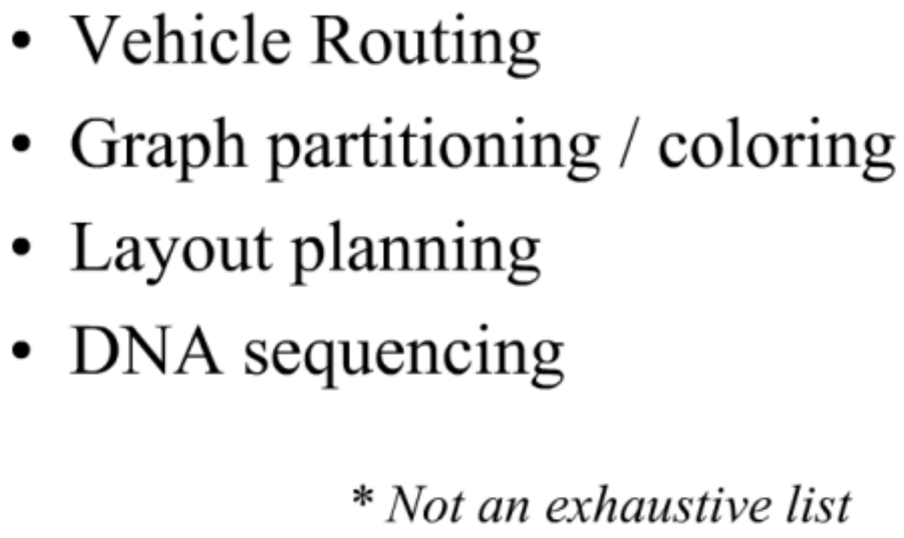
- Each search process sends its best-so-far solution to the central memory when it gets updated,
- If the sent solution is worse than the solution available in the central memory, the search process uses the stored solution,
- If its best-so-far solution is not improved for a specified number of iteration, the search process requests the solution stored in the central memory.

COOPERATION

Another approach is to implement the central memory as a pool of the best *s* found solutions,

When a search process requests a solution from the central memory, it gets a randomly selected solution from the pool rather than always getting the best one.

OTHER EXAMPLES



- Vehicle Routing
- Graph partitioning / coloring
- Layout planning
- DNA sequencing

* *Not an exhaustive list*

REFERENCES

1. F. Glover. ”TabunSearch Part I”. ORSA Journal on Computing, vol. 1, no. 3, pp. 190-206, 1989.
2. F. Glover. “TabunSearch Part II”. ORSA Journal on Computing, vol. 2, no. 1, pp. 4-32, 1989.
3. E. Taillard. “Parallel Taboo Search Techniques for the Job Shop Scheduling Problem”. *ORSA Journal on Computing*, vol. 6, no. 2, 108-117, 1994.
4. M. Dell’Amico and M. Trubian. “Applying Tabu Search to the Job Shop Scheduling Problem”. Annals of Operation Research, vol. 41, no. 3, pp. 231-252, 1993.
5. T. G. Crainic, M. Toulouse and M Gendreau. “Synchronous Tabu Search Parallelization Strategies for Multicommodity Location-Allocation with Balancing Requirements”. Centre de recherche sur les transports, Universite de Montreal, Tech. Rep. 934, 1993.

REFERENCES

6. M. Malek, M. Guruswamy, M. Pandya and H. Owens. “Serial and Parallel Simulated Annealing and Tabu Search Algorithms for the Travelling Salesman Problem”. Annals of Operation Research, vol. 21, pp. 59-84, 1989.

7. P. Hansen, The steepest ascent mildest descent heuristic for combinatorial programming, Congress on Numerical Methods in Combinatorial Optimization, Italy, 1986.

8. Hertz, A., Taillard, E. and Werra, D. A Tutorial on Tabu Search. http://www.cs.colostate.edu/~whitley/CS640/hertz92tutorial.pdf

9. Revees, C., (ed) Modern Heuristic Techniques for Cominatorial Problems. Ch 3 by Glover and Languna.

10. T. G. Crainic, M. Toulouse and M Gendreau. “Parallel Asynchronous Tabu Search for Multicommodity Location-Allocation with Balancing Requirements”. Centre de recherche sur les transports, Universite de Montreal, 1994.