

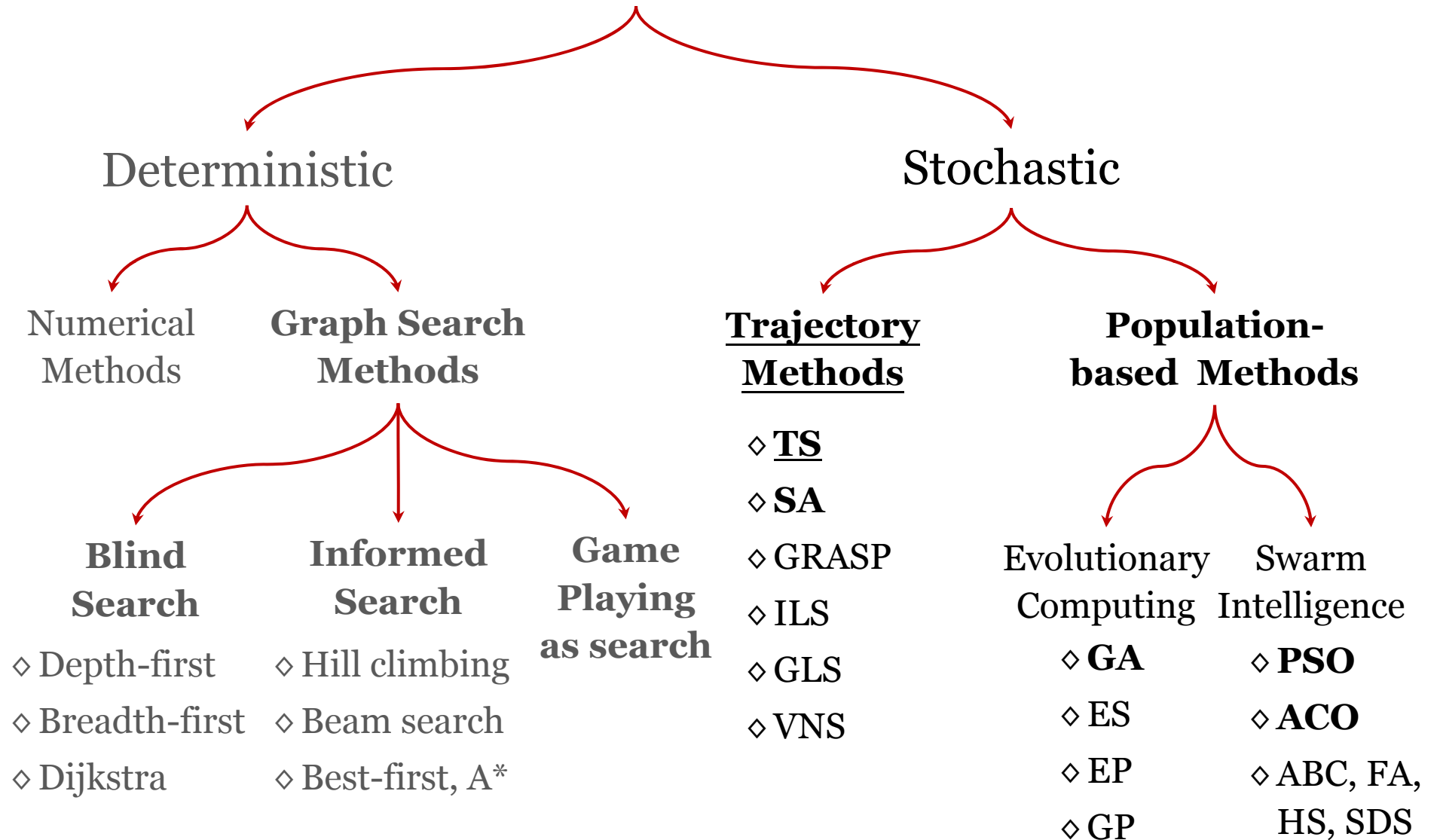
Trajectory-based Metaheuristics:

Tabu Search - I

Lecture 5 – Thursday May 22, 2014

Outline

Search Methods



Outline

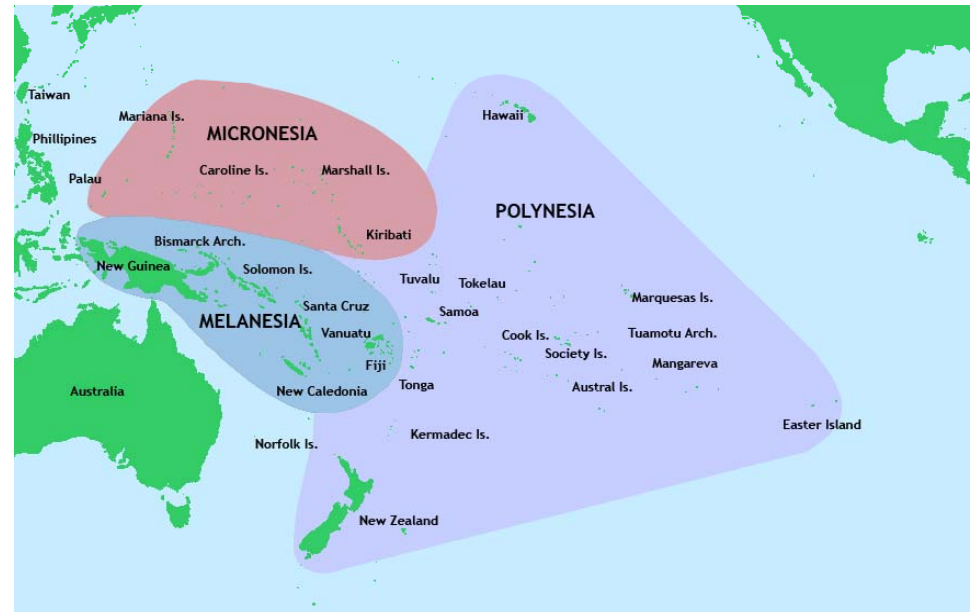
- Tabu Search
- Solving Permutation Optimization
- Solving Graph Problems
- Solving Constraint Satisfaction Problems
- Vehicle Routing Problem (VRP)
- Assignment Problem
- Knapsack Problem
- Adaptation
- Cooperation
- Summary

Outline

- **Tabu Search**
- Solving Permutation Optimization
- Solving Graph Problems
- Solving Constraint Satisfaction Problems
- Vehicle Routing Problem (VRP)
- Assignment Problem
- Knapsack Problem
- Adaptation
- Cooperation
- Summary

Tabu Search

- The actual first usage of metaheuristic is probably due to **Fred Glover's Tabu search (TS)** in 1986, though his seminal book on Tabu search was published later in 1997 [1].
- The word **tabu (or taboo)** comes from Tongan, a language of Polynesia, where it indicates things that cannot be touched because they are sacred.
- It also means “**a prohibition imposed by social custom**” or “**forbidden**”.



Tabu Search

• Applications

Tabu Search Applications	
Scheduling	Design
Flow-Time Cell Manufacturing Heterogeneous Processor Scheduling Workforce Planning Classroom Scheduling Machine Scheduling Flow Shop Scheduling Job Shop Scheduling Sequencing and Batching	Computer-Aided Design Fault Tolerant Networks Transport Network Design Architectural Space Planning Diagram Coherency Fixed Charge Network Design Irregular Cutting Problems
Location and Allocation	Routing
Multicommodity Location/Allocation Quadratic Assignment Quadratic Semi-Assignment Multilevel Generalized Assignment Lay-Out Planning Off-Shore Oil Exploration	Vehicle Routing Capacitated Routing Time Window Routing Multi-Mode Routing Mixed Fleet Routing Traveling Salesman Traveling Purchaser

[1]

Tabu Search

• Applications (cont'd)

Tabu Search Applications	
Production, Inventory and Investment	Telecommunications
Flexible Manufacturing Just-in-Time Production Capacitated MRP Part Selection Multi-item Inventory Planning Volume Discount Acquisition Fixed Mix Investment	Call Routing Bandwidth Packing Hub Facility Location Path Assignment Network Design for Services Customer Discount Planning Failure Immune Architecture Synchronous Optical Networks
Logic and Artificial Intelligence	Technology
Maximum Satisfiability Probabilistic Logic Clustering Pattern Recognition/Classification Data Integrity Neural Network Training and Design	Seismic Inversion Electrical Power Distribution Engineering Structural Design Minimum Volume Ellipsoids Space Station Construction Circuit Cell Placement

Tabu Search

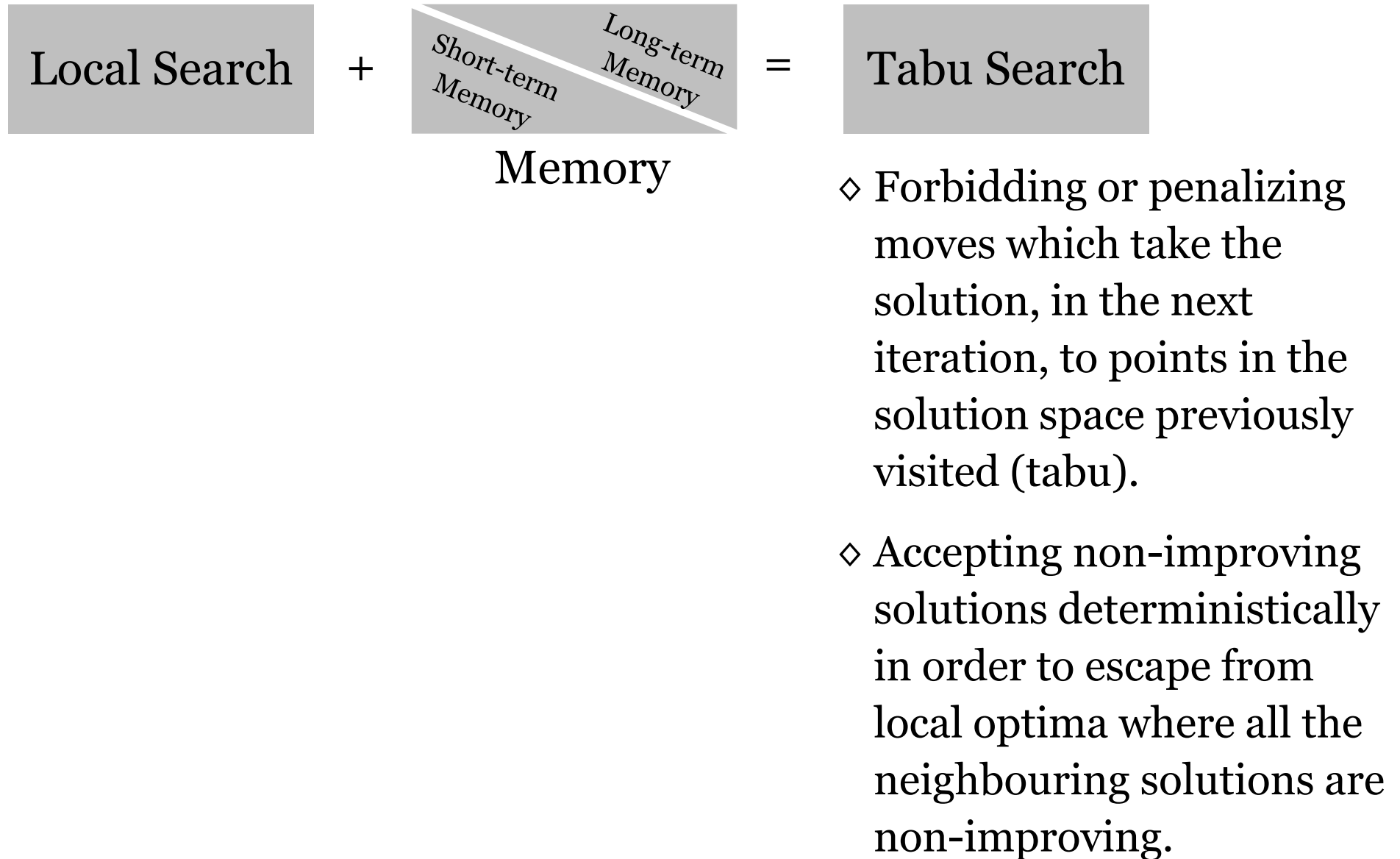
• Applications (cont'd)

Tabu Search Applications	
Graph Optimization	General Combinational Optimization
Graph Partitioning Graph Coloring Clique Partitioning Maximum Clique Problems Maximum Planner Graphs P-Median Problems	Zero-One Programming Fixed Charge Optimization Nonconvex Nonlinear Programming All-or-None Networks Bilevel Programming General Mixed Integer Optimization

Tabu Search

- TS is a powerful search optimization, which can be considered as the combination of local search (LS) and memory structures.
- TS is originally proposed to allow Local search (LS) to overcome local optima.
- TS uses memory via a **Tabu list** to avoid revisiting recently visited neighborhood. This may substantially increase the efficiency for solving some problems.
- In TS, when finding the next solution to visit, some solution elements (or moves) are regarded as tabu, i.e. they cannot be used in constructing next solution.

Tabu Search



Tabu Search

- **Local Search**

Start with an initial feasible solution,

While termination criterion not met

Generate a neighboring solution by applying a
 series of local modifications (or moves),

If the new solution is better

Replace the old one,

End

Tabu Search

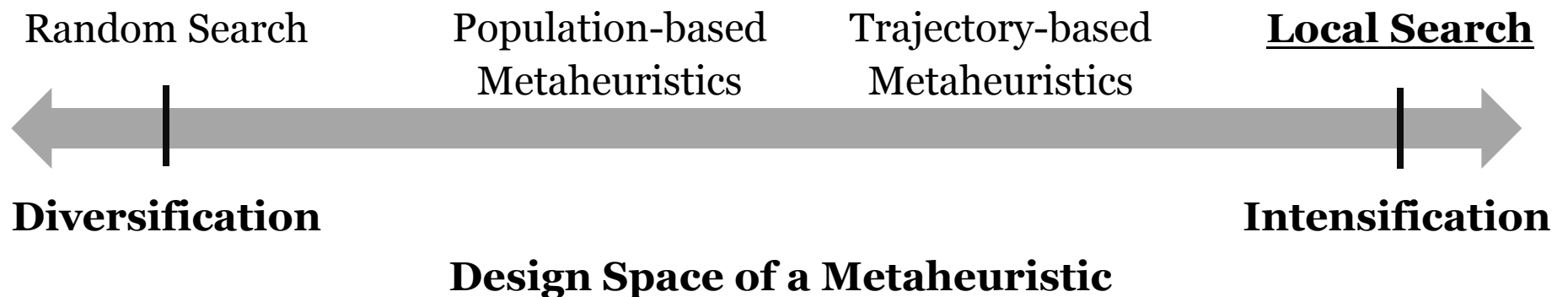
- **Local Search: Example of moves**

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

Tabu Search

• Local Search: Neighborhood

- ◇ One extreme is 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

- **Local Search: Local Optimum**

- ◇ LS starts from some initial solution and moves from neighbor to neighbor as long as possible while decreasing the objective function value.
- ◇ LS terminates when it hits a local optimum.
- ◇ Such local optima are usually an average solutions, unless the search is extremely lucky.
- ◇ The obtained solution quality and the computation time is usually dependent on the chosen local moves.
- ◇ TS was originally proposed to allow LS to overcome local optima.

Tabu Search

• Tabu Search Concepts

- ◇ Tabu search (TS) is an **iterative neighborhood search algorithm**, where the neighborhood changes dynamically.
- ◇ TS enhances local search by actively avoiding points in the search space already visited.
- ◇ By avoiding already visited points, loops in search trajectories are avoided and local optima can be escaped.

Tabu Search

• Tabu Search Concepts

- ◇ TS can be considered as the combination of local search (LS) and memory structures.
- ◇ The main feature of TS is the use of an explicit memory.
- ◇ Uses of memory in two ways:
 - Prevent the search from revisiting previously visited solutions.
 - Explore the unvisited areas of the solution space.

Tabu Search

- **Tabu Search Concepts**

- ◇ A simple TS usually implements two forms of memory:

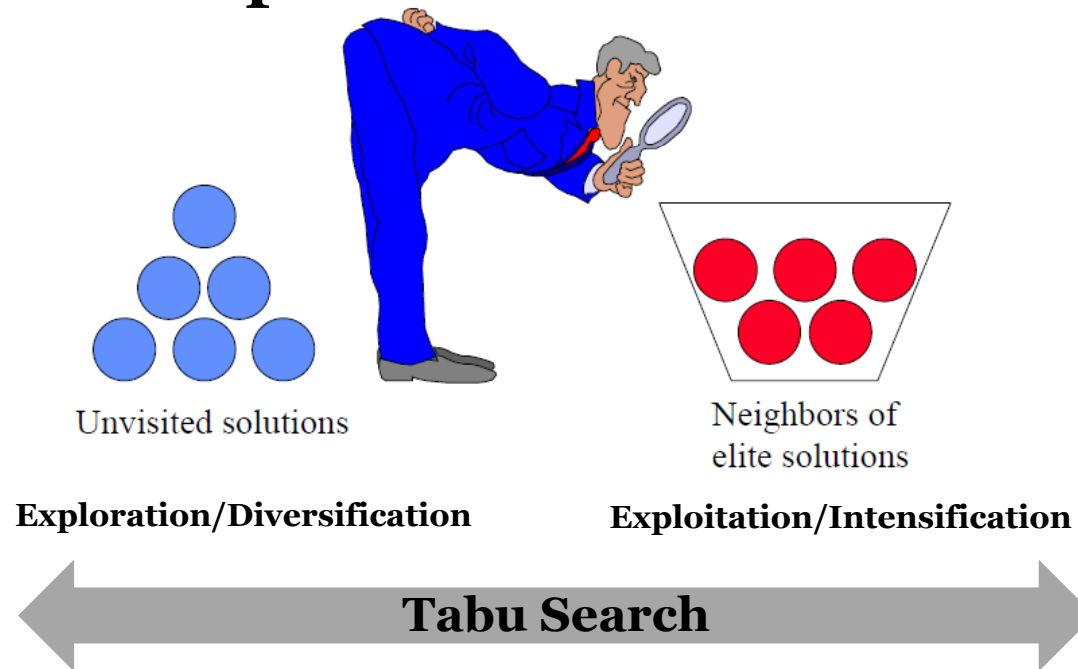
- **A recency-based memory or short-term memory or tabu list:** which maintains information about **how recently** a search point has been visited (or how recently a move has been made). Recency is based on the iteration at which the event occurred.

- **A frequency-based memory or long-term memory:** which maintains information about **how often** a search point has been visited (or how often a move has been made) during a specified time interval.

[3]

Tabu Search

• Tabu Search Concepts



Frequency-based memory or long-term memory

Long term Memory is used to **diversify** the search and explore unvisited areas of the solution space by avoiding frequently visited components

Recency-based memory or short-term memory

Recency memory can be used to keep track of good components to return to in order to **intensify** the search

Tabu Search

- **Tabu Search Concepts: 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.

Tabu Search

- **Tabu Search Concepts: 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**).
- ◇ Choosing Tabu Tenure (length) T :
 - **Static:** choose T to be a constant or as a guideline where n =problem size.
 - **Dynamic:** choose T to vary randomly between T_{\min} and T_{\max} . Can make the threshold T_{\min} and T_{\max} vary for attributes.

Tabu Search

• Tabu Search Concepts: Neighborhood

- ◇ Like other search method we need a neighborhood structure $N(s)$

Definition For a finite set of candidate solutions S , a neighborhood structure is a function $N: S \rightarrow 2^S$ that assigned to every $s \in S$ a set of neighbors $N(s) \subseteq S$.

- ◇ 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.

Tabu Search

• Tabu Search Concepts: Stagnation

- ◇ When a single attribute is marked as tabu, this typically results in more than **one solution being tabu**. Some of these solutions that must now be avoided could be of excellent quality and might not have been visited. This problem is known as **stagnation**.
- ◇ To **prevent stagnation**, sometimes it's useful to allow a certain move even if it's in the tabu list.
- ◇ Approaches used to cancel the tabus are referred to as **aspiration criteria**.
- ◇ A commonly used aspiration criterion is to allow solutions which are **better than** the currently-known best solution.

Tabu Search

• Tabu Search 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 Tabu list $T(s)$ and Aspiration criteria $A(s)$

End

Tabu Search

- **Tabu Search Algorithm**

- ◇ The following may be used to terminate TS:

- The **neighborhood is empty**, i.e. all possible neighboring points have already been visited, or
 - When the **number of iterations** since the last improvement is larger than a **specified threshold**.
 - Evidence shows that an **optimum solution** has been obtained.

Tabu Search

Pros

- Allows non-improving solution to be accepted in order to escape from a local optimum.
- Can be applied to both discrete and continuous solution spaces.
- For larger and more difficult problems (scheduling, quadratic assignment and vehicle routing), tabu search obtains solutions that often surpass the best solutions previously found by other approaches .

Cons

- Too many parameters to be determined.
- Number of iterations could be very large.
- Global optimum may not be found, depends on parameter Settings.

Outline

- Tabu Search
- **Solving Permutation Optimization**
- Solving Graph Problems
- Solving Constraint Satisfaction Problems
- Vehicle Routing Problem (VRP)
- Assignment Problem
- Knapsack Problem
- Adaptation
- Cooperation
- Summary

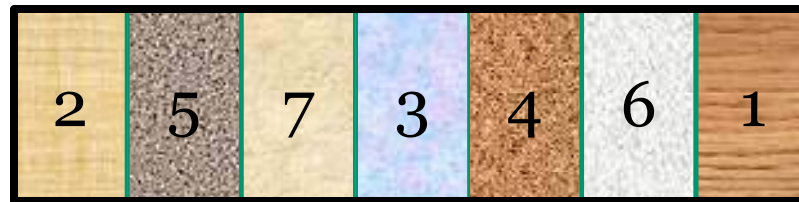
Solving Permutation Optimization

- **Permutation**

Permutation is a sequence or an ordered combination containing each element from a finite set once, and only once.

- **Problem Definition**

Find the ordering of modules (filters) that maximizes the overall insulating property of a composite material.



Representation of a solution for 7 modules of Composite Material

Solving Permutation Optimization

- **Representation**

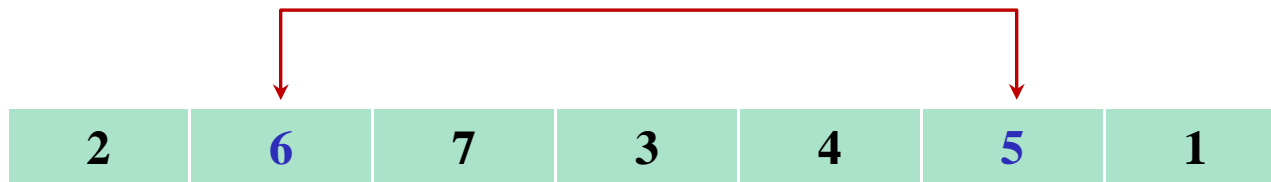
- ◇ Representation of a solution for 7 modules:

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

- ◇ Number of permutations (Permutations without Repetition)

$$P(n, n) = n! = 7! = 5,040$$

- ◇ Neighborhood structure: swapping modules



Solving Permutation Optimization

- **Representation (cont'd)**

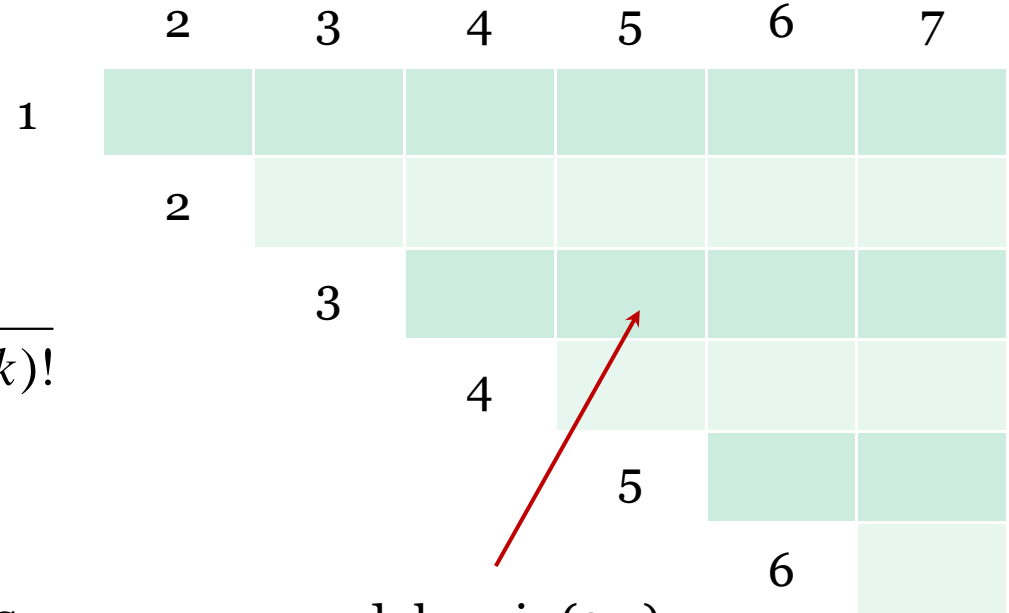
◇ **Neighborhood:** is defined as any other solution that is obtained by a **pair wise exchange** of any two nodes in the solution.

Let n =number of nodes=7, K =pair wise exchange=2

of neighbors is # of combination without repetition $C(7,2)$:

$$C(n,k) = n\text{-choose-}k = \binom{n}{k} = \frac{n!}{k!(n-k)!}$$
$$= \frac{7!}{2!5!} = 21$$

A solution has 21 neighbors



module pair (3,5)

Solving Permutation Optimization

- **Tabu Search Attributes**

- ◇ 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.

Solving Permutation Optimization

• Iteration 0 (Starting Point)

◇ Current Solution

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

Insulation Value=10

◇ Tabu Structure

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

All entries zero

◇ Top 5 candidates

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

*

“Value” is the gain of swap

Solving Permutation Optimization

• 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						

Tabu move

◇ Top 5 candidates

Swap	Value
3,1	2
2,3	1
3,5	-1
7,1	-2
6,1	-4

*

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

Solving Permutation Optimization

• 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.

◇ Top 5 candidates

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

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

Solving Permutation Optimization

• 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						

Move (4,5) has a tabu tenure of

6

1 iteration but this move results in the best solution so far Hence its tabu status is overridden

◇ Top 5 candidates

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

“Aspiration criteria “

Solving Permutation Optimization

• 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	
3,6	-8	

Best move is (1,7)

Solving Permutation Optimization

• Diversification using Frequency-based Memory

◇ Iteration 26

◇ Current Solution **1 3 6 2 7 5 4** Insulation Value=12

◇ Tabu Structure

(Recency)							
	1	2	3	4	5	6	7
1				3			
2							
3	3					2	
4	1	5					1
5		4		4			
6			1		2		
7	2			3			

(Frequency)

Counts of moves over past iterations

◇ Top 5 candidates

Swap	Value	Penalized Value	Total
1,4	3	3	6
2,4	-1	-6	-7
3,7	-3	3	0
1,6	-5	-5	-10
6,5	-4	-6	-10

T

Diversify when no admissible improving moves exist. Penalize non-improving moves by assigning larger penalty to more frequent swaps, choose (3,7) using penalized value.

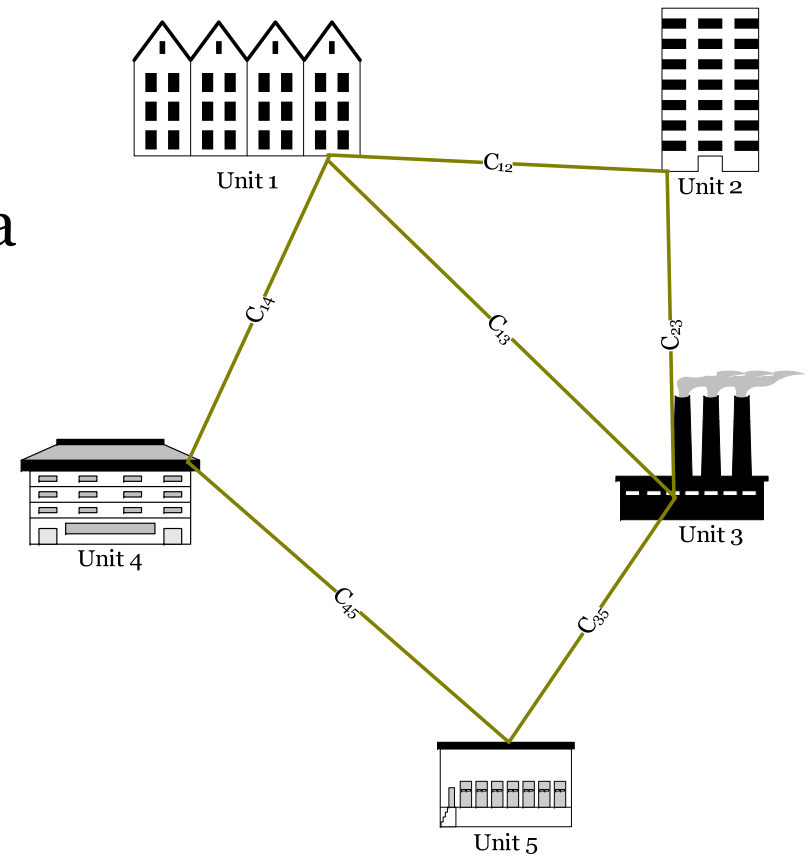
Outline

- Tabu Search
- Solving Permutation Optimization
- **Solving Graph Problems**
- Solving Constraint Satisfaction Problems
- Vehicle Routing Problem (VRP)
- Assignment Problem
- Knapsack Problem
- Adaptation
- Cooperation
- Summary

Solving Graph Problems

• Problem Definition

An industrial communication company is planning to lay cables to a new factory allowing all the factory's units to be linked for the interchange of data. If it is constrained to bury the cable only along certain paths, then there would be an undirected graph representing which points are connected by those paths.



Some of those paths might be more expensive, because they are longer, or require the cable to be buried deeper; these paths would be represented by edges with larger weights.

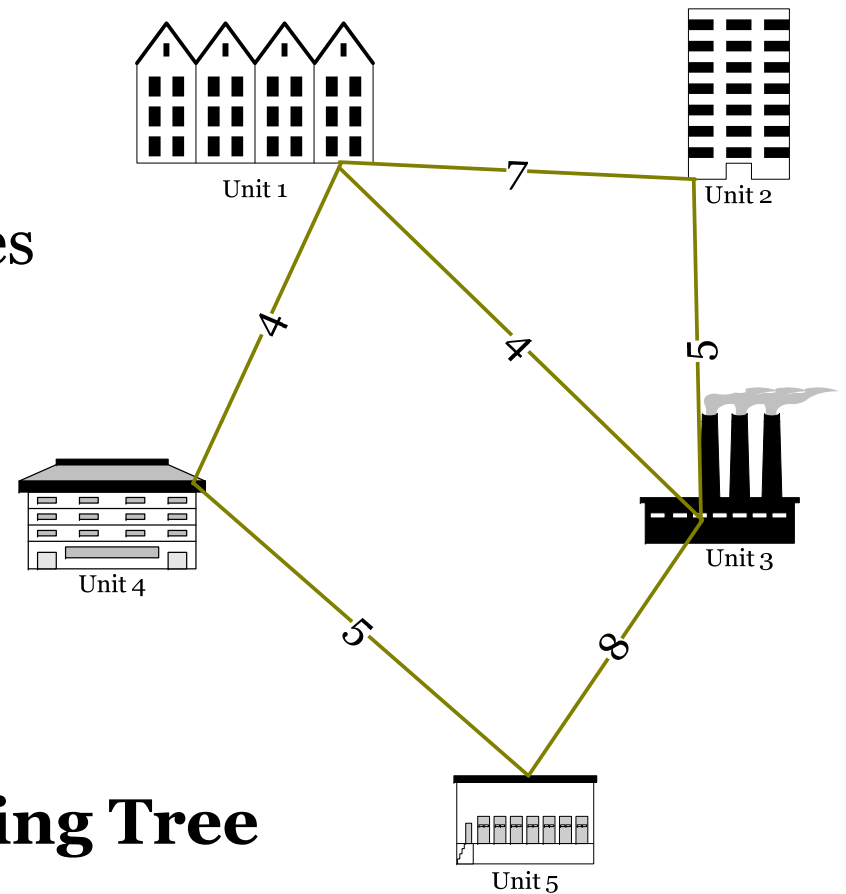
Solving Graph Problems

- **Problem Definition**

Given: 5 factory's units and the possible paths along which the cables can be buried.

Required: Find the most cost-effective paths to connect all the units together.

This is a typical **Minimum Spanning Tree (MST)** problem



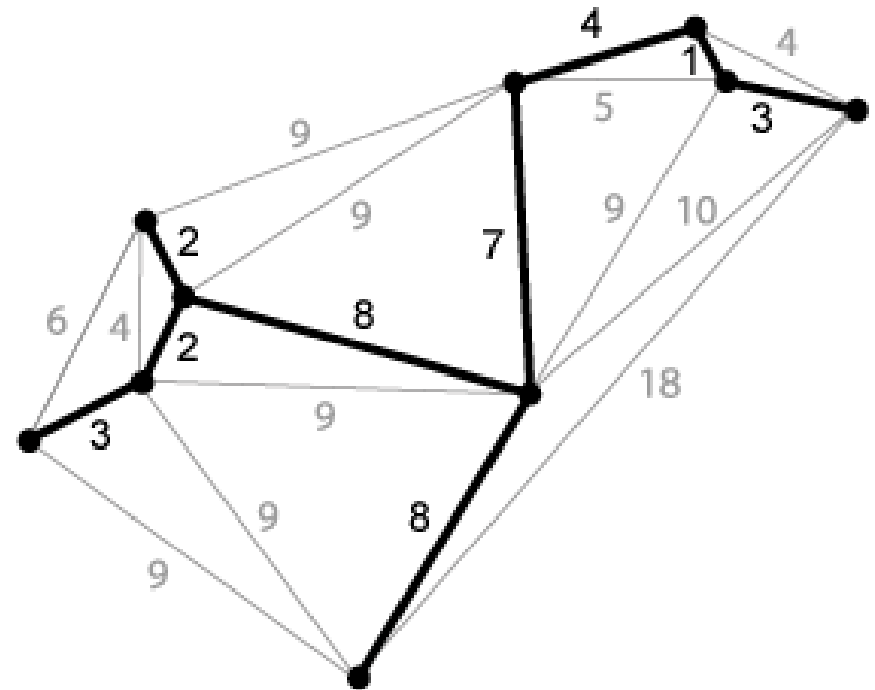
Solving Graph Problems

- **Minimum Spanning Tree (MST)**

Given a connected, undirected graph, a **spanning tree** of that graph is a subgraph that is a tree and connects all the vertices together.

A single graph can have many different spanning trees.

A **minimum spanning tree (MST)** or minimum weight spanning tree is a spanning tree with weight less than or equal to the weight of every other spanning tree.



Source: Wikipedia

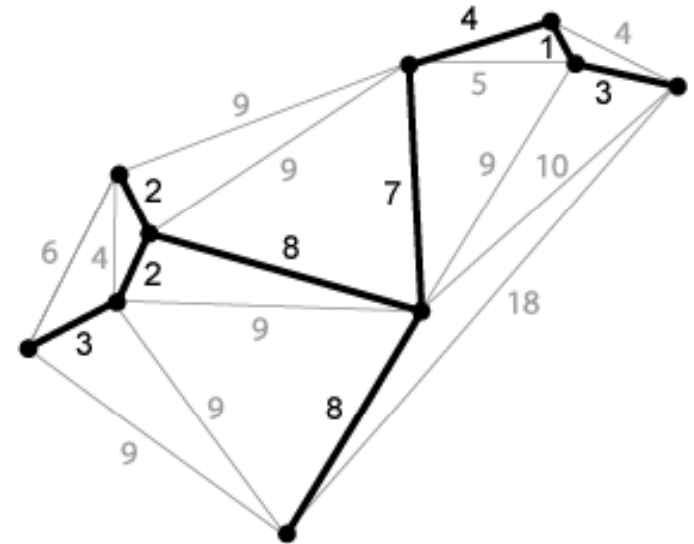
Solving Graph Problems

• Kruskal's Algorithm

Kruskal's algorithm is a greedy algorithm that finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized.

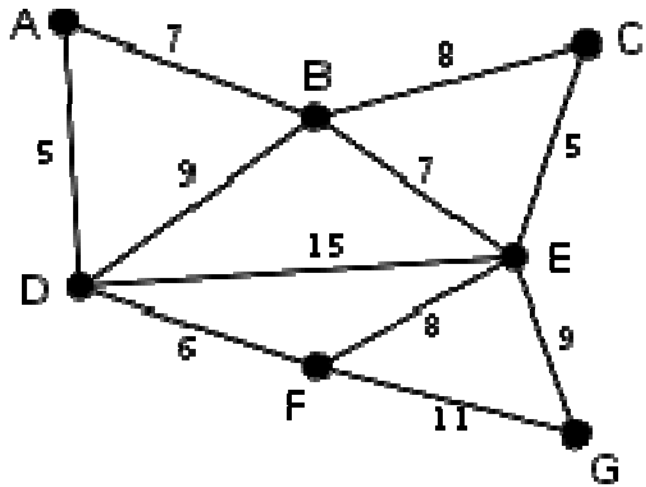
If the graph is not connected, then it finds a **minimum spanning forest** (a minimum spanning tree for each connected component).

Source: Wikipedia

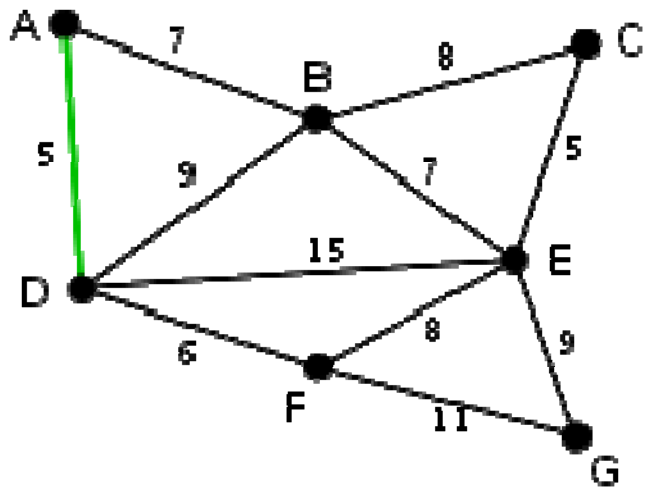


Solving Graph Problems

• Kruskal's Algorithm



- This is our original graph. The numbers near the arcs indicate their weight. None of the arcs are highlighted.

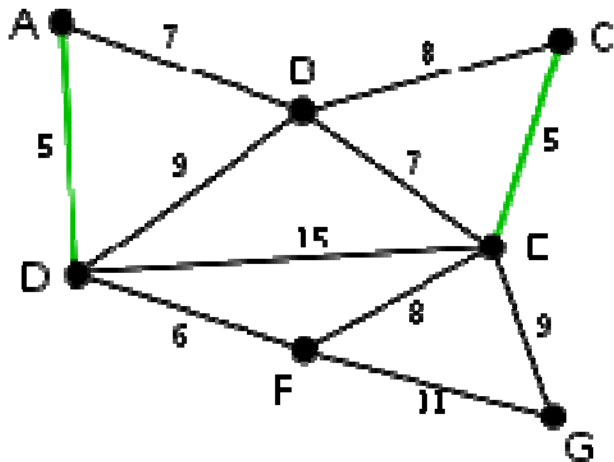


- **AD** and **CE** are the shortest arcs, with length 5, and **AD** has been arbitrarily chosen, so it is highlighted.

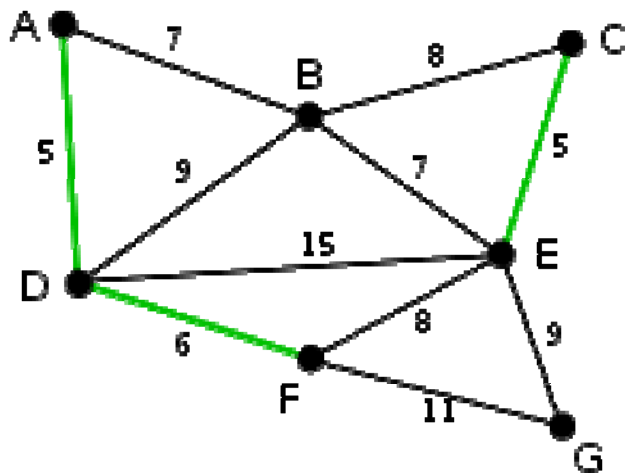
Source: Wikipedia

Solving Graph Problems

• Kruskal's Algorithm (cont'd)



- **CE** is now the shortest arc that does not form a cycle, with length 5, so it is highlighted as the second arc.

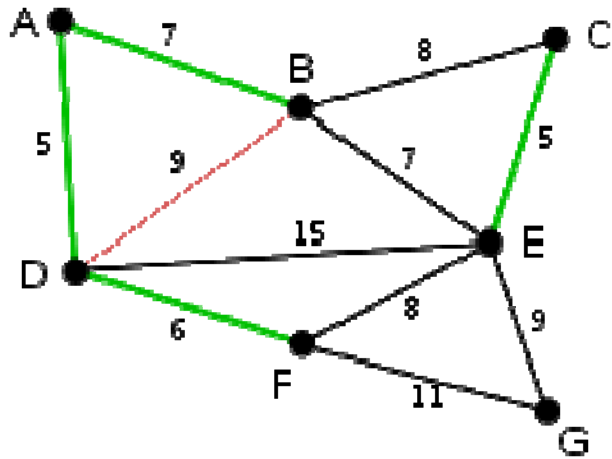


- The next arc, **DF** with length 6, is highlighted using much the same method.

Source: Wikipedia

Solving Graph Problems

• Kruskal's Algorithm (cont'd)

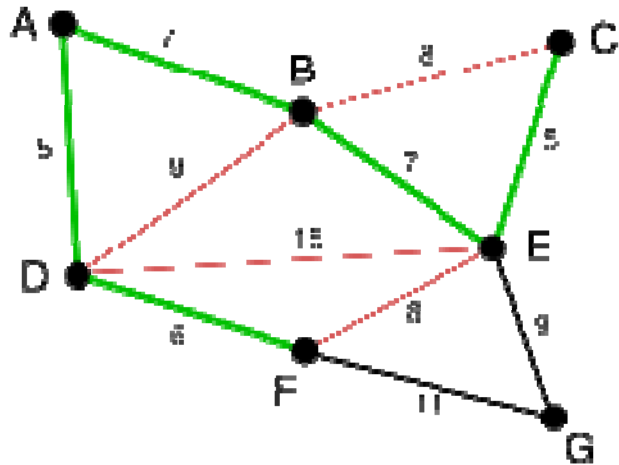


- The next-shortest arcs are **AB** and **BE**, both with length 7. **AB** is chosen arbitrarily, and is highlighted. The arc **BD** has been highlighted in red, because there already exists a path (in green) between **B** and **D**, so it would form a cycle (**ABD**) if it were chosen.

Source: Wikipedia

Solving Graph Problems

• Kruskal's Algorithm (cont'd)

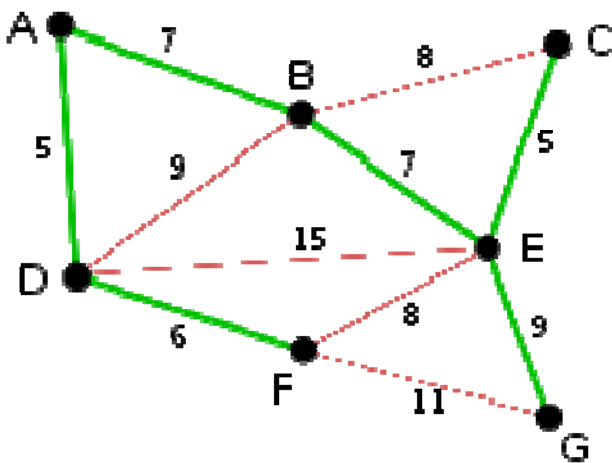
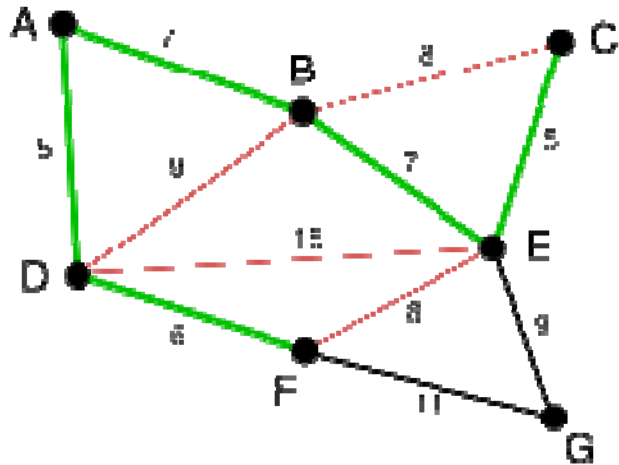


- The process continues to highlight the next-smallest arc, **BE** with length 7. Many more arcs are highlighted in red at this stage: **BC** because it would form the loop **BCE**, **DE** because it would form the loop **DEBA**, and **FE** because it would form **FEBAD**.

Source: Wikipedia

Solving Graph Problems

• Kruskal's Algorithm (cont'd)



- The process continues to highlight the next-smallest arc, **BE** with length 7. Many more arcs are highlighted in red at this stage: **BC** because it would form the loop **BCE**, **DE** because it would form the loop **DEBA**, and **FE** because it would form **FEBAD**.
- Finally, the process finishes with the arc **EG** of length 9, and the minimum spanning tree is found.

Source: Wikipedia

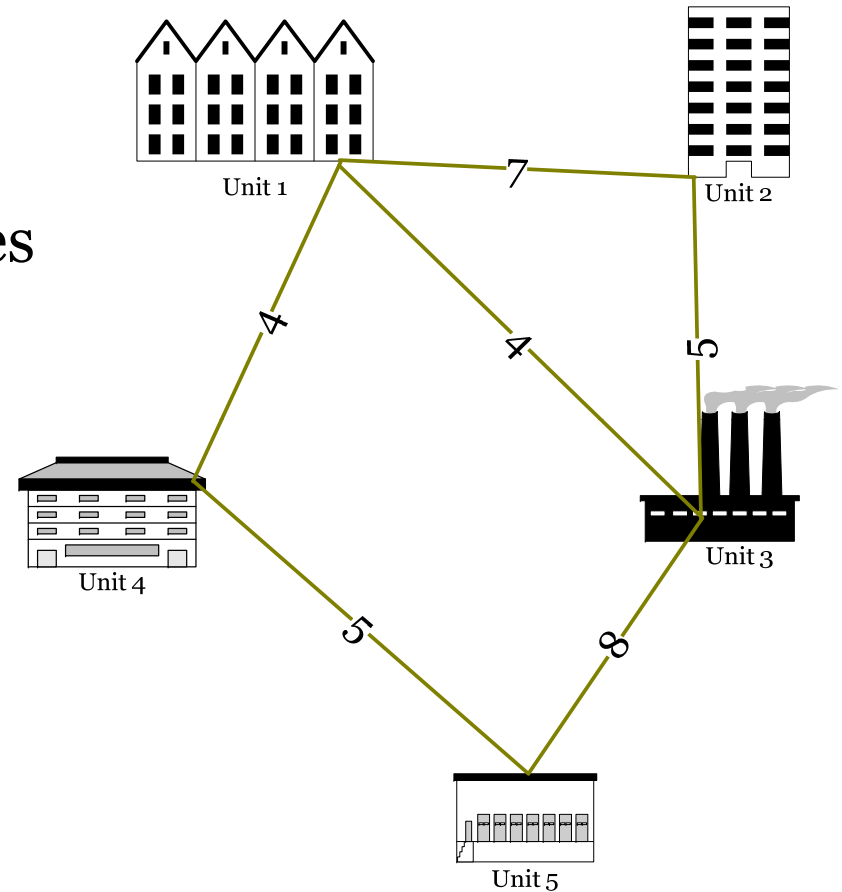
Solving Graph Problems

• Tab Search

Given: 5 factory's units and the possible paths along which the cables can be buried.

Required: Find the most cost-effective paths to connect all the units together using Tabu search.

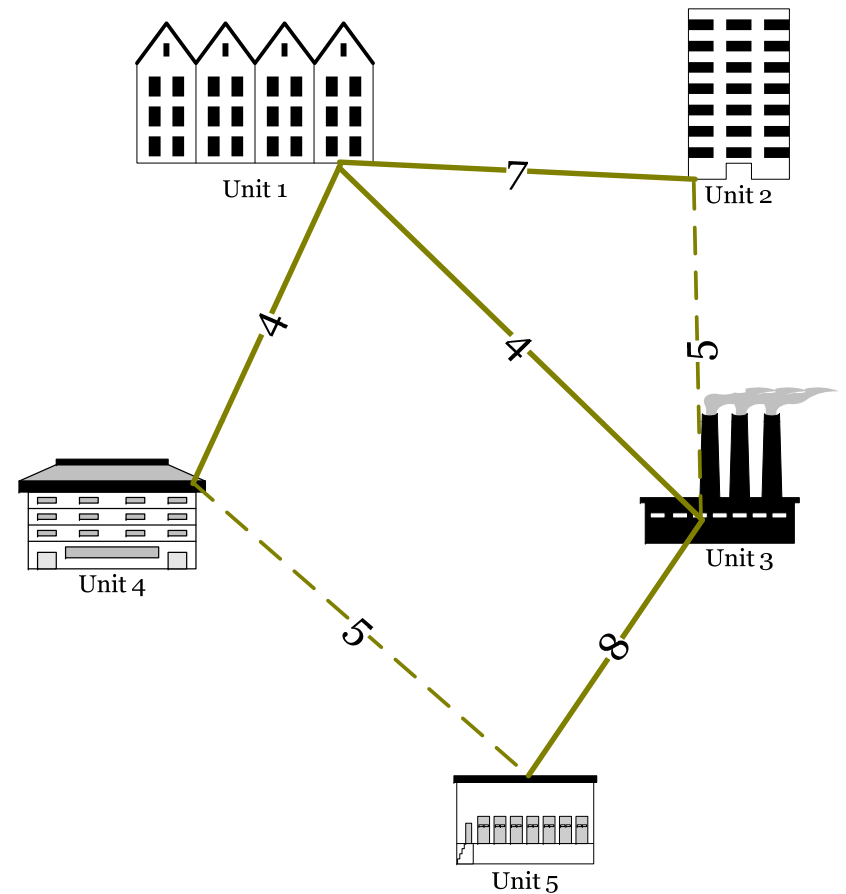
Use Tabu length=2



Solving Graph Problems

- Iteration 0 (Initial Solution)

$$\text{Cost} = 7 + 4 + 4 + 8 = 23$$



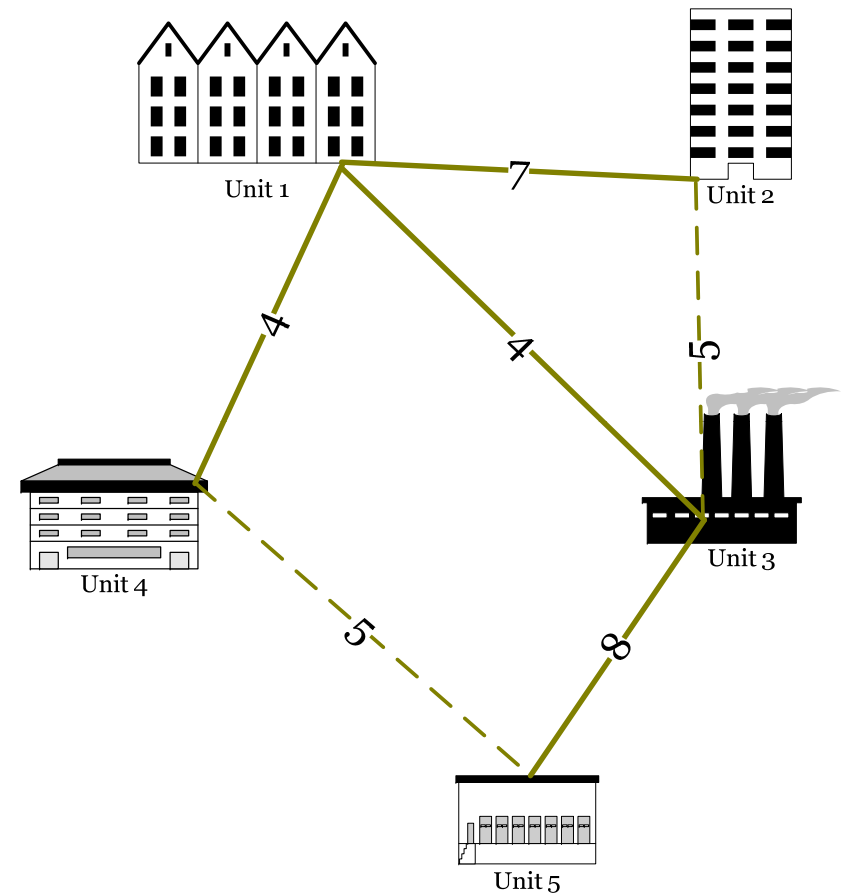
Solving Graph Problems

• Iteration 1

$$\text{Cost} = 7 + 4 + 4 + 8 = 23$$

Tabu

		Swap		Value
		Remove	Add	
E12				
E13				
E14		E12	E23	21
E23		E13	E23	24
E35		E13	E45	24
E45		E14	E45	24
		E35	E45	20



*

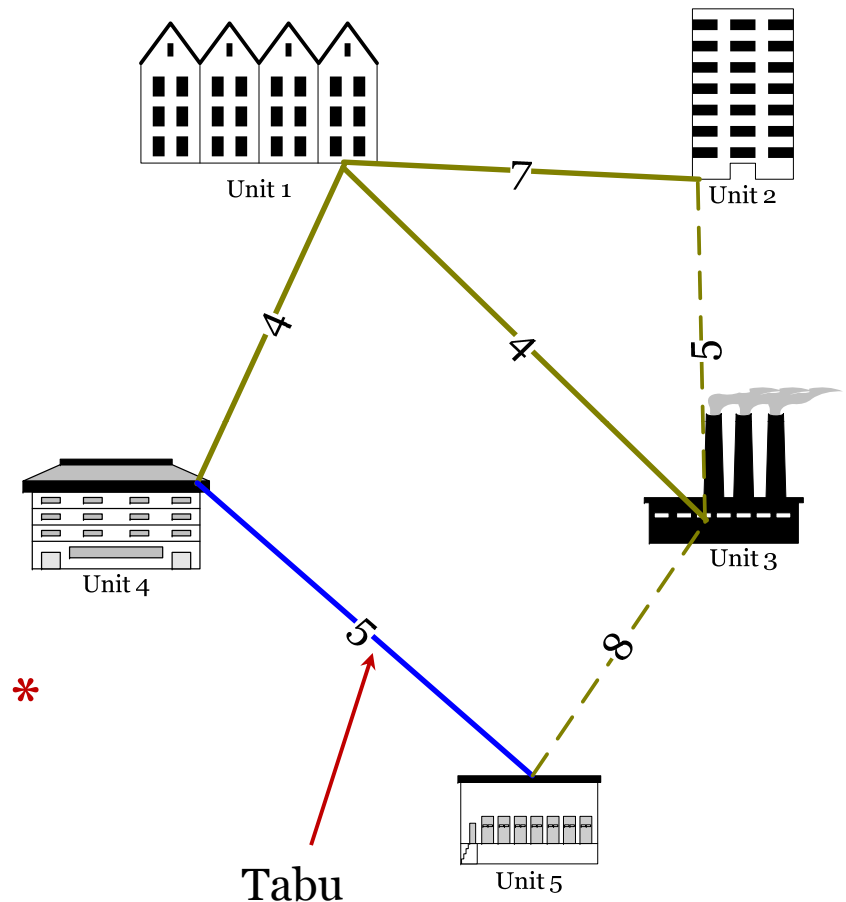
Solving Graph Problems

• Iteration 2

$$\text{Cost} = 7 + 4 + 4 + 5 = 20$$

Tabu

		Swap		Value
		Remove	Add	
E12				
E13				
E14		E12	E23	18
E23		E13	E23	21
E35		E13	E35	24
E35		E14	E35	24
E45	2	E45	E35	23



T

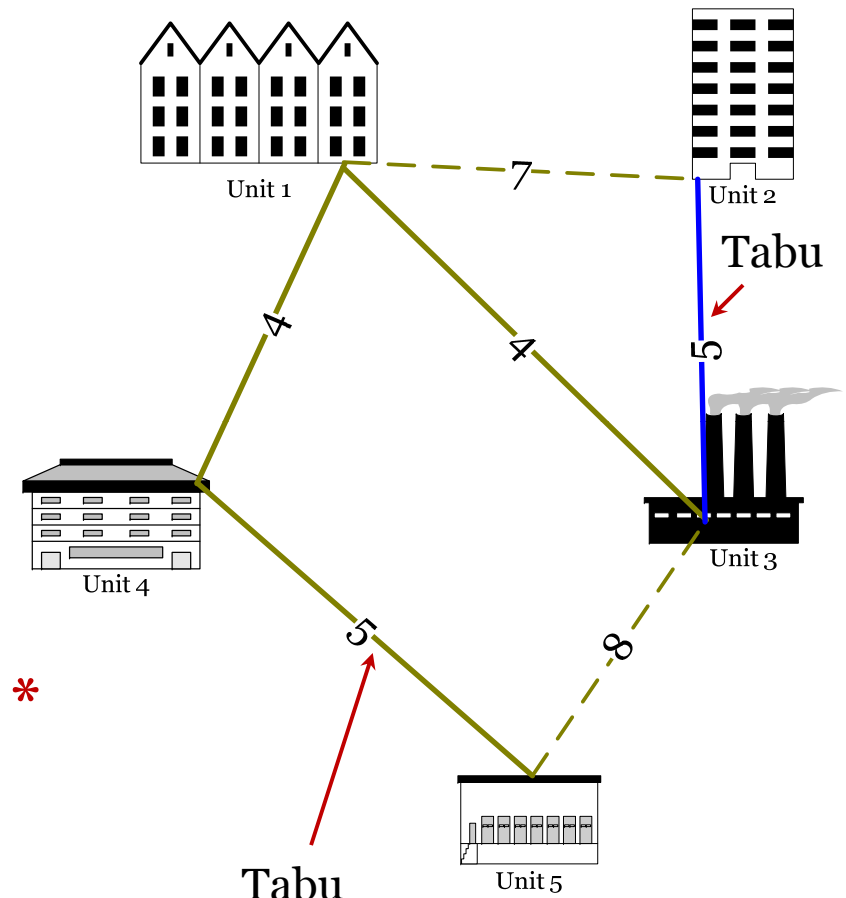
Solving Graph Problems

• Iteration 3

$$\text{Cost} = 4 + 4 + 5 + 5 = 18$$

Tabu

		Swap		Value
		Remove	Add	
E12				
E13				
E14		E13	E12	21
E23	2	E13	E35	22
E35		E14	E35	22
E45		E23	E12	20
	1	E45	E35	21



T
T

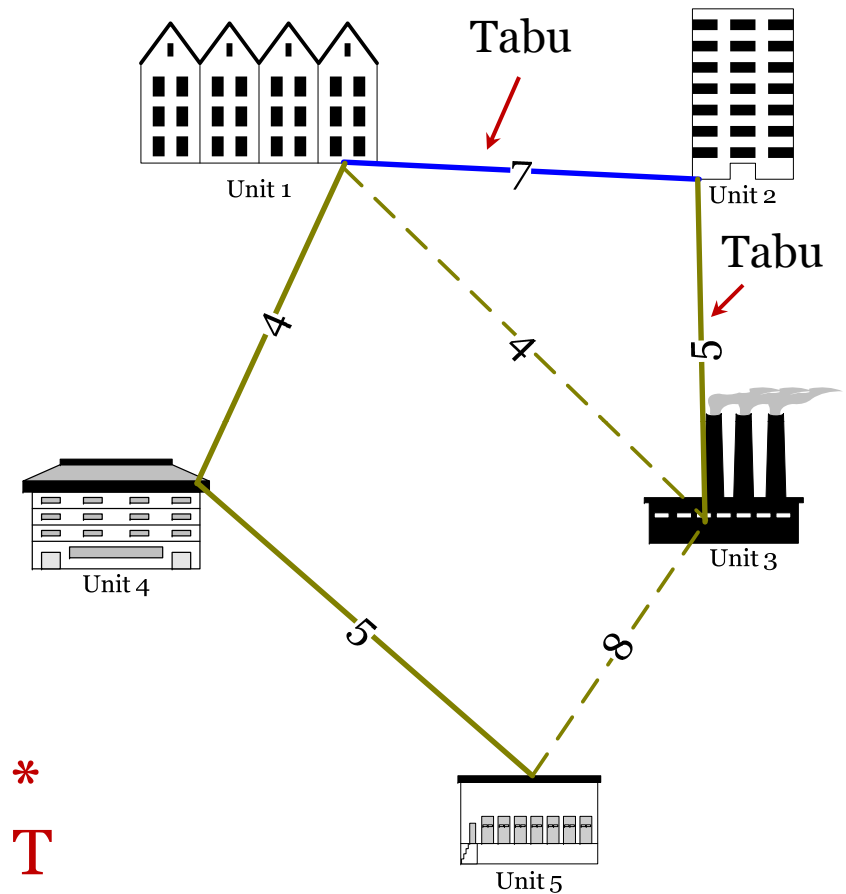
Solving Graph Problems

• Iteration 4

$$\text{Cost} = 7 + 4 + 5 + 5 = 21$$

Tabu

		Swap		Value
		Remove	Add	
E12	2			
E13				
E14				
E23	1			
E35				
E45				
		E14	E35	25
		E45	E35	24
		E23	E13	20
		E12	E13	18
		E23	E35	24



*
T
T
T

Solving Graph Problems

• Assignment-2

- Use the given functions to find the most cost-effective paths for the graph given in **Units100.mat** and report the total cost of laying the cables.
- Run the code with different tabu list lengths (5, 10, 15, 25, 50) and report the cost obtained in each case. What do you observe?
- Modify **GetBestNeighbourST.m** and other necessary functions to include the use of an **aspiration criterion**. The aspiration criterion in this problem could be to allow solutions that are better than the currently known best solution. Repeat (a) and (b) with the modified code.
- Suppose that splitters need be used at the factory's units if the number of cables going through the unit exceeds 2. The cost of the **splitter** increases as the number of cables connected to the splitter ***n*** increases and it can be calculated as

$$C_s = \frac{10}{1 + e^{-n/10}}$$

Modify **GetBestNeighbourST.m** and other necessary functions to consider adding the cost of the splitters. Repeat (a) and (b) with the modified code

- Compare between Tabu Search and the greedy algorithm of **Kruskal** as an example for greedy algorithms.

Code Developer: A. Farahat, University of Waterloo

Outline

- Tabu Search
- Solving Permutation Optimization
- Solving Graph Problems
- **Solving Constraint Satisfaction Problems**
- Vehicle Routing Problem (VRP)
- Assignment Problem
- Knapsack Problem
- Adaptation
- Cooperation
- Summary

Solving Constraint Satisfaction Problems

- A constraint satisfaction problem (or CSP) is a special kind of problem that satisfies some additional structural properties beyond the basic requirements for problems in general.
- In a CSP, the states are defined by the values of a set of variables and the goal test specifies **a set of constraints** that the values have to obey.

Solving Constraint Satisfaction Problems

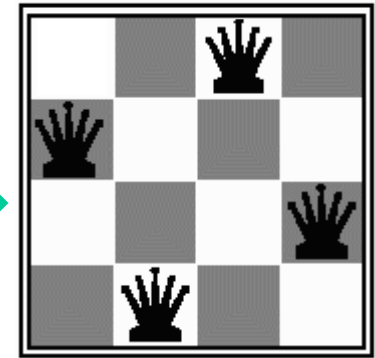
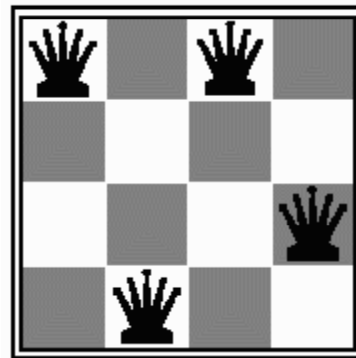
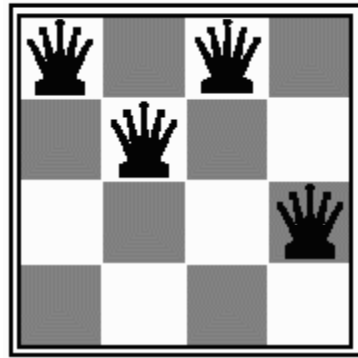
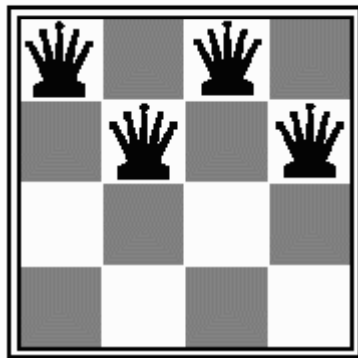
- **State components:**
 - ◇ Variables
 - ◇ Domains (possible values for the variables)
 - ◇ (Binary) constraints between variables
- **Goal:** to find a state (a complete assignment of values to variables), which satisfies the constraints
- **Examples:**
 - ◇ map coloring
 - ◇ crossword puzzles
 - ◇ n-queens
 - ◇ resource assignment/distribution/location

[5]

Solving Constraint Satisfaction Problems

- **N-queens Problem**

Put n queens on an $n \times n$ board with no two queens on the same row, column, or diagonal



Move a queen to reduce number of conflicts

Goal configuration

no queen is
attacking
another queen

Solving Constraint Satisfaction Problems

- **N-queens Problem**

- ◇ Variables: $Q_1 \dots Q_n$ (queens)
- ◇ Domains: $[1 \dots n]$ for each C_i (columns)
- ◇ Constraints: Q_i does not attack Q_j
- ◇ No of possible configurations = $C_N^{N \times N}$
- ◇ By restricting each queen to a row we can reduce this to N^N .
Representing the problem as permutation of N column locations of the queens the number becomes $N!$
- ◇ There is currently no known formula for the exact number of solutions.

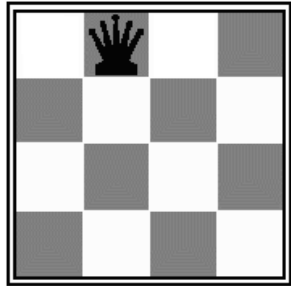
Solving Constraint Satisfaction Problems

- 4-queens Problem**

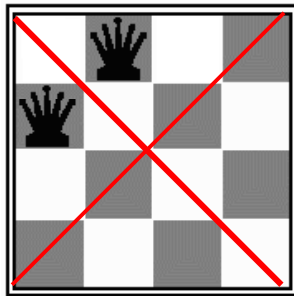
each queen is represented as a row: Q1, Q2, Q3, and Q4

An assignment consists of assigning a column to a queen

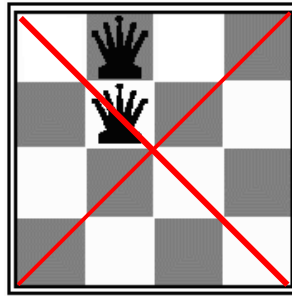
$\{ Q1 = 2 \} \rightarrow$ places a queen in row 1, column 2



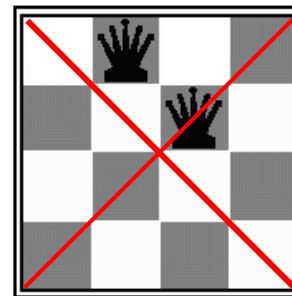
$\{ Q1 = 2 \}$



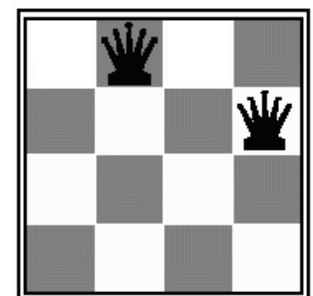
$\{ Q2 = 1 \}$



$\{ Q2 = 2 \}$



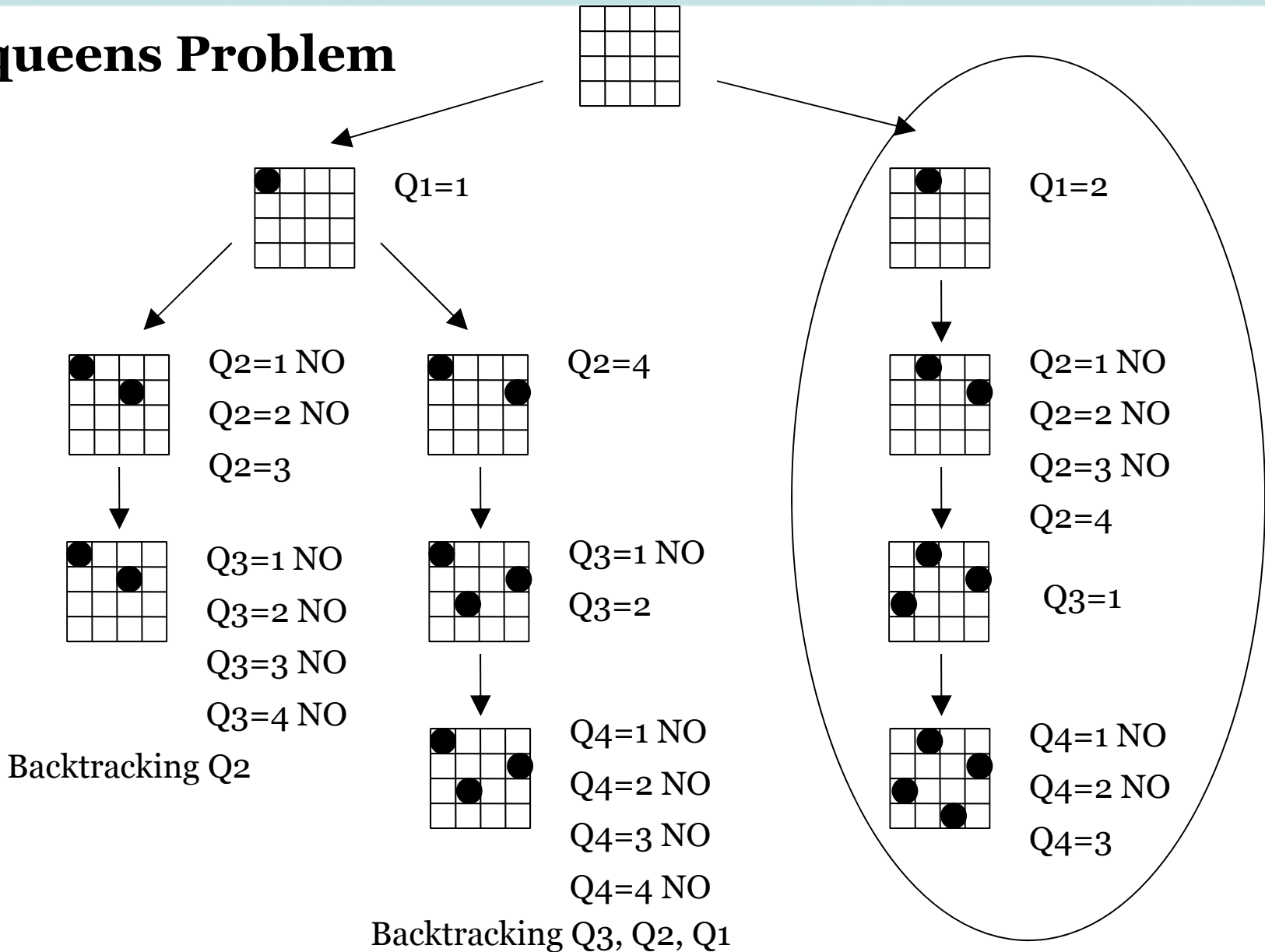
$\{ Q2 = 3 \}$



$\{ Q2 = 4 \}$

Solving Constraint Satisfaction Problems

- 4-queens Problem

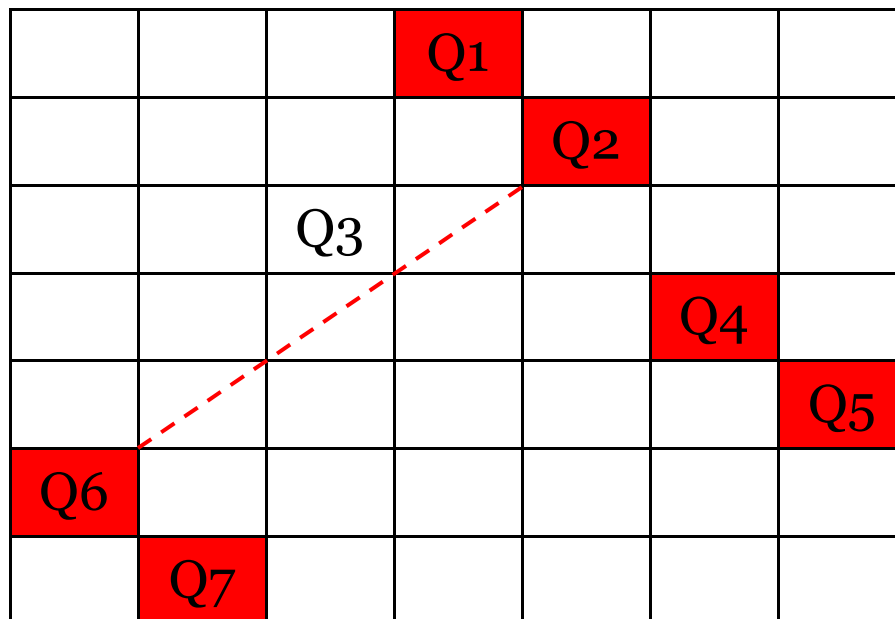


Solving Constraint Satisfaction Problems

- **7-queens Problem**

The queen is able to attack any square on:

- ◇ the same row,
- ◇ any square on the same column,
- ◇ and also any square on either of the diagonals



A 7x7 chessboard illustrating a partial solution to the 7-queens problem. The queens are placed at the following positions (row, column): Q1 at (1,4), Q2 at (2,5), Q3 at (3,3), Q4 at (4,6), Q5 at (5,7), Q6 at (6,1), and Q7 at (7,2). A red dashed line connects the squares (3,3) and (6,1), indicating a diagonal collision between Q3 and Q6. There are four collisions in total: (Q1, Q3), (Q1, Q5), (Q2, Q4), and (Q3, Q6).

			Q1			
				Q2		
		Q3				
					Q4	
						Q5
Q6						
	Q7					

Collisions=4

Solving Constraint Satisfaction Problems

- 7-queens Problem

			Q1			
				Q2		
		Q3				
					Q4	
						Q5
Q6						
	Q7					

Collisions=4

Q	1	2	3	4	5	6	7
C	4	5	3	6	7	1	2

Representation as ordering

[1]

Solving Constraint Satisfaction Problems

- Solving 7-queens Problem using TS

Queen Swap	Δ Value
1,2	1
1,3	1
1,4	1
1,5	-1
1,6	-1
1,7	-2
2,3	1

Queen Swap	Δ Value
2,4	-2
2,5	0
2,6	-2
2,7	-1
3,4	3
3,5	-1
3,6	-1

Queen Swap	Δ Value
3,7	1
4,5	2
4,6	1
4,7	-1
5,6	-2
5,7	1
6,7	-1

Q	1	2	3	4	5	6	7
C	4	5	3	6	7	1	2

21 possible moves

collisions = 4

Solving Constraint Satisfaction Problems

- Solving 7-queens Problem using TS: Iteration-0

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

Queen Swap	Δ Value
1,7	-2
2,4	-2
2,6	-2
5,6	-2
1,5	-1

*

Q	1	2	3	4	5	6	7
C	4	5	3	6	7	1	2

collisions = 4

Select (1,7) as a move

Solving Constraint Satisfaction Problems

- Solving 7-queens Problem using TS: Iteration-1

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

Q	1	2	3	4	5	6	7
C	2	5	3	6	7	1	4

collisions = 2

Queen Swap	Δ Value
2,4	-1
1,6	0
2,5	0
1,2	1
1,3	1

*

Select (2,4) as a move

Solving Constraint Satisfaction Problems

- Solving 7-queens Problem using TS: Iteration-2

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

Q	1	2	3	4	5	6	7
C	2	6	3	5	7	1	4

collisions = 1

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

Select (1,3) as a move

Solving Constraint Satisfaction Problems

- Solving 7-queens Problem using TS: Iteration-3

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

Q	1	2	3	4	5	6	7
C	3	6	2	5	7	1	4

collisions = 1

Queen Swap	Δ Value
1,3	0
1,7	0
5,7	1
6,7	1
1,2	2

T
T

Select (5,7) as a move

Solving Constraint Satisfaction Problems

- Solving 7-queens Problem using TS: After Iteration-3

		Q1				
					Q2	
	Q3					
				Q4		
			Q5			
Q6						
						Q7

Collisions=**2**

Q	1	2	3	4	5	6	7
C	3	6	2	5	4	1	7

Current
solution

Solving Constraint Satisfaction Problems

- Solving 7-queens Problem using TS: Iteration-4

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

Q	1	2	3	4	5	6	7
C	3	6	2	5	4	1	7

collisions = 2

Queen Swap	Δ Value	
4,7	-1	*
5,7	-1	T
1,5	0	
2,5	0	
2,4	2	T

Select (4,7) as a move

Solving Constraint Satisfaction Problems

- Solving 7-queens Problem using TS: Iteration-5

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

Q	1	2	3	4	5	6	7
C	3	6	2	7	4	1	5

collisions = 1

Queen Swap	Δ Value
1,3	-1
5,6	0
5,7	0
1,6	0
1,7	2

* T

T

Select (1,3) as a move

Using aspiration

Solving Constraint Satisfaction Problems

- Solving 7-queens Problem using TS: After Iteration-5**

	Q1					
					Q2	
		Q3				
						Q4
			Q5			
Q6						
				Q7		

Selecting (1,3)

Collisions=0

Desired solution

Q	1	2	3	4	5	6	7
C	2	6	3	7	4	1	5

One of 40 possible

Solutions (6 fundamental)

References

1. F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Boston, 1997.
2. M. Kamel. ECE493T8: Cooperative and Adaptive Algorithms. University of Waterloo, 2009-2010.
3. Andries P. Engelbrecht. *Computational Intelligence: An Introduction*. 2nd Edition, John Wiley & Sons Ltd, 2007.
4. Glover and Laguna, Chapter 3 in Reeves, *Modern Heuristic Techniques for Combinatorial Problems*, Wiley, 1993.
5. S. Russel and P. Norving. *AI: A modern approach*. Prentice Hall, 2003.