Large Scale Optimization

Emmanouil Zachariadis, Assistant Professor
Department of Management Science and Technology, Athens University of Economics and Business
E: ezach@aueb.gr
A: Evelpidon 47A and Lefkados 33 street, 9th floor, Room 906, 11362, Athens, Greece
T: +30-210-8203 674

Course Objective

- Objective of the course: Prepare students on developing and implementing computational methods for tackling well-defined management science applications
- Computational methods are the main tool for solving a wide range of Management Science Problems
- Emphasis will be given on the design, scheduling organization

  of tasks/resources towards cost minimization, profit maximization, customer service maximization for companies and organizations
- Data = Problem statement enabler
  - Let us formulate optimization problems, so that we can solve them
- Computational Methods
  - The tool to solve optimization problems

Course Objective

In other words,

We are going to explore strategies for managing organizations by solving their problems in an effective (high quality solutions) and efficient (short CPU time) manner

Importance of Computational Methods

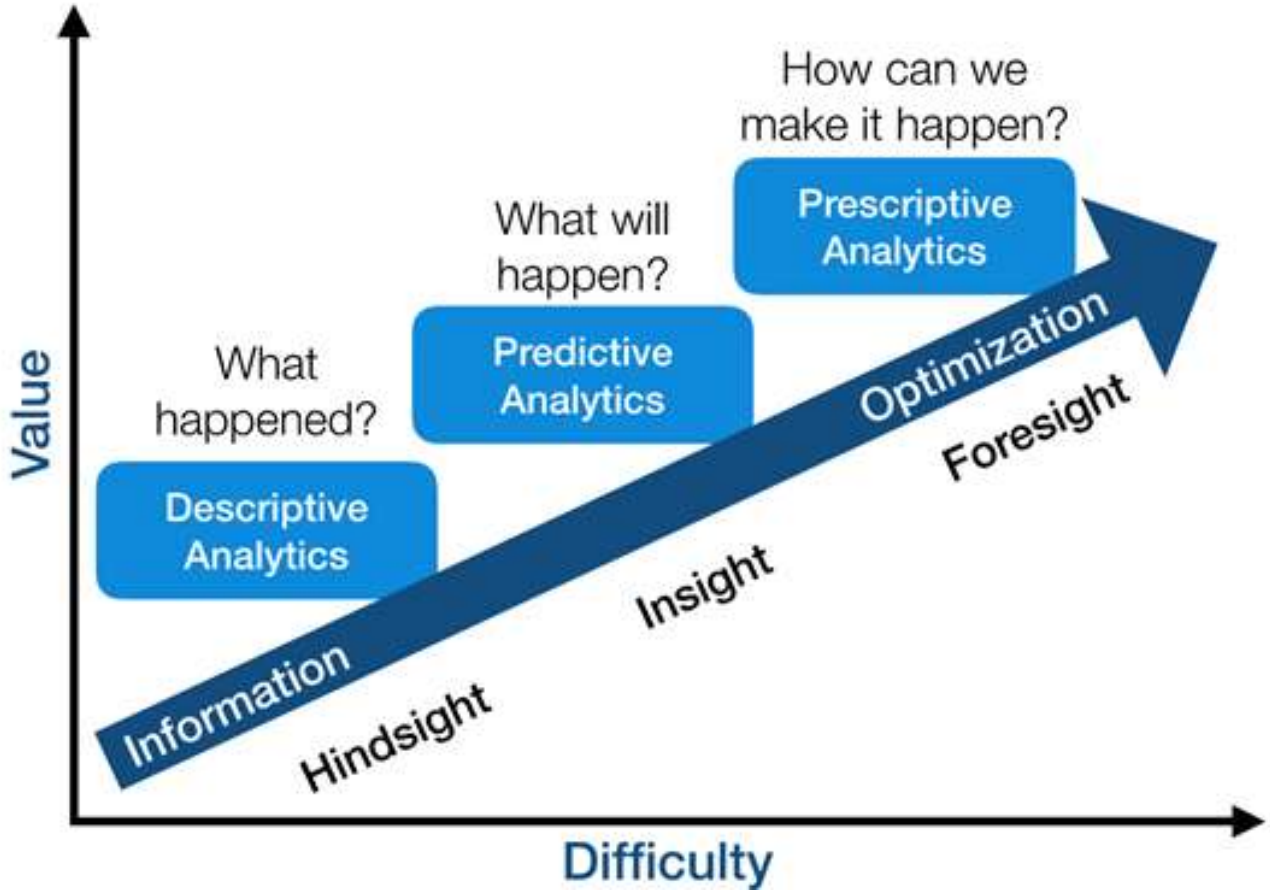
- "Want Less-Biased Decisions? Use Algorithms.", Harvard Business Review, 7/26/2018
- "Organizations are increasingly favoring algorithms in an effort to make organizational decision making and judgment more rigorous" Harvard Business Review (2016)
- A research Accenture study involving 254 large sale companies in the US shows that 60% of managerial decisions are drawn via (some kind of) computational methods

Importance of Computational Methods

- Academic study involving 16 top management teams of US companies reported that the use of computational methods on making managerial decisions is very and increasingly important

- IBM has recently reorganized their advisory activities to form a new structure of 4000 employees working on computational methods for management science problems

Prescriptive Analytics – Optimizing Business



Tactical Decision Making in Management Science

Tactical Decisions: Decisions that impact day-to-day life of a company/organization

Main Characteristic of a Tactical Decision:

Short term impact
(Tactical Decisions are taken frequently)

Tactical Decision Making in Management Science

Tactical Decision Examples:

- Vehicle Routing Decisions
- Management and Scheduling of Resources
- Management of Queues

Strategic Decision Making in Management Science

Strategic Decisions: Decisions that have a major impact on the general operation and course of a company/organization

Main Characteristic of a Tactical Decision:

- Long term impact

(Strategic Decisions are taken less frequently)

Strategic Decision Making in Management Science

Strategic Decision Examples:

- Location of facilities (Production plant, warehouses, retail spots)
- Assignment of suppliers to production sites, production Sites to Distribution centers, customers to distribution centers etc.
- Facility Layout (Assignment of units to locations – Departments in a university campus,)
- Vehicle Fleet Composition (Private/Leased, Capacities)

Basic features of well-defined problems

Most problems of management science problems have the following common characteristic**:**

The "shape" of their solution has a combinatorial nature

(represented by combinatorial analysis structures: permutations, combinations, etc)

- Solution Shape
  - Everything that must be defined so that the cost (or profit) of a solution can be uniquely calculated
  - Everything that must be defined for the problem to be considered totally solved
- Combinatorial Nature:
  - Solution characteristics which form the shape of a given solution can be recombined in different ways to produce different solutions

Example – What is an optimal solution?

- Suppose you are organizing a trip with your friends
- Starting from Athens you will visit 15 major cities and then come back to Athens
- Each of the 15 cities will be visited just once
- The distance from any starting point to any other target point is available to you
  - Google, Bing Maps, etc.

Example – What is an optimal solution?

Try answering to the following questions:

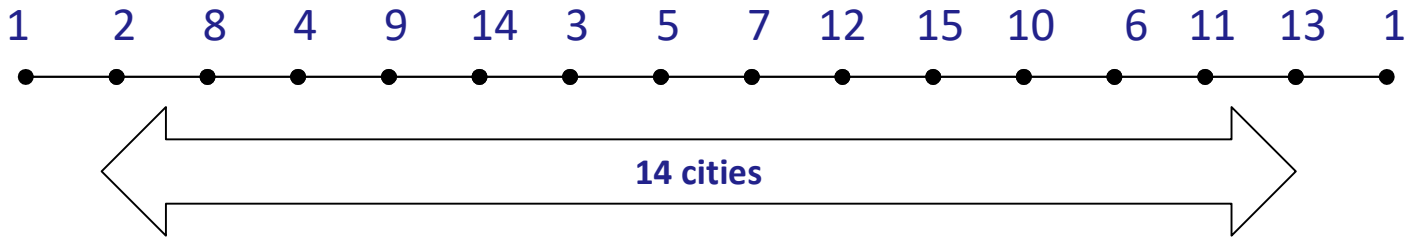
- Having a simple method which examines every possible solution (brute-force algorithm), how many solutions must be examined till the best (optimal) solution is identified?
- Consider that you are in possession of a PC able to examine 1M solutions per second
- How much time is needed by the brute-force methodology to generate the optimal solution

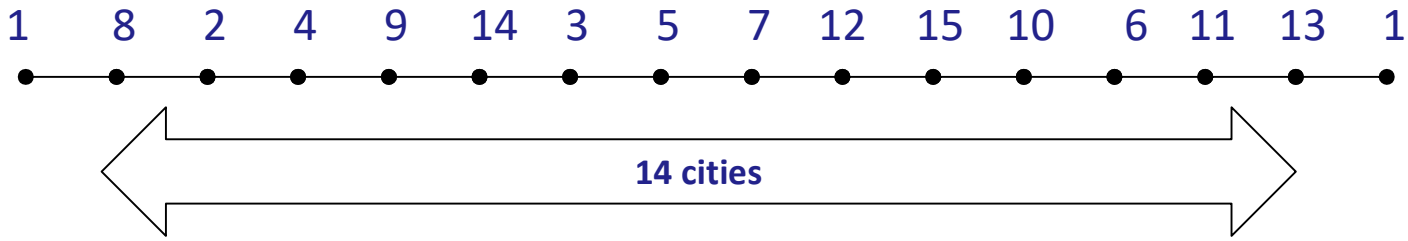
Can an optimal solution be achieved in acceptable CPU time?

- Estimation of the brute force method CPU time
- $(15-1)! = 14! = 8.8 \cdot 10^{10} = 88$ billion solutions (approx.)

$$\frac{8.8 \cdot 10^{10}\ sols}{10^{6}\ sols/sec} = 8.8 \cdot 10^{4}\ sec \Longrightarrow \frac{8.8 \cdot 10^{4}\ sec}{3600\ sec/hr} = 24.44\ hr$$

- Optimal Solutions:
  - Brute Force: Huge CPU times
  - State-of-The Art Integer Programming methods: Hard to develop, Very sensitive to the operation realities (the operational scenario that is being solved)
- In practice, we are interested in a (very) good quality solution which can be obtained in short CPU time
  - Computational Methods for Optimization
  - Simplest Computational Methods: Greedy Optimization Methods





Two of the $8.8 \cdot 10^{10}$ total feasible solutions (routes, permutations) of the problem

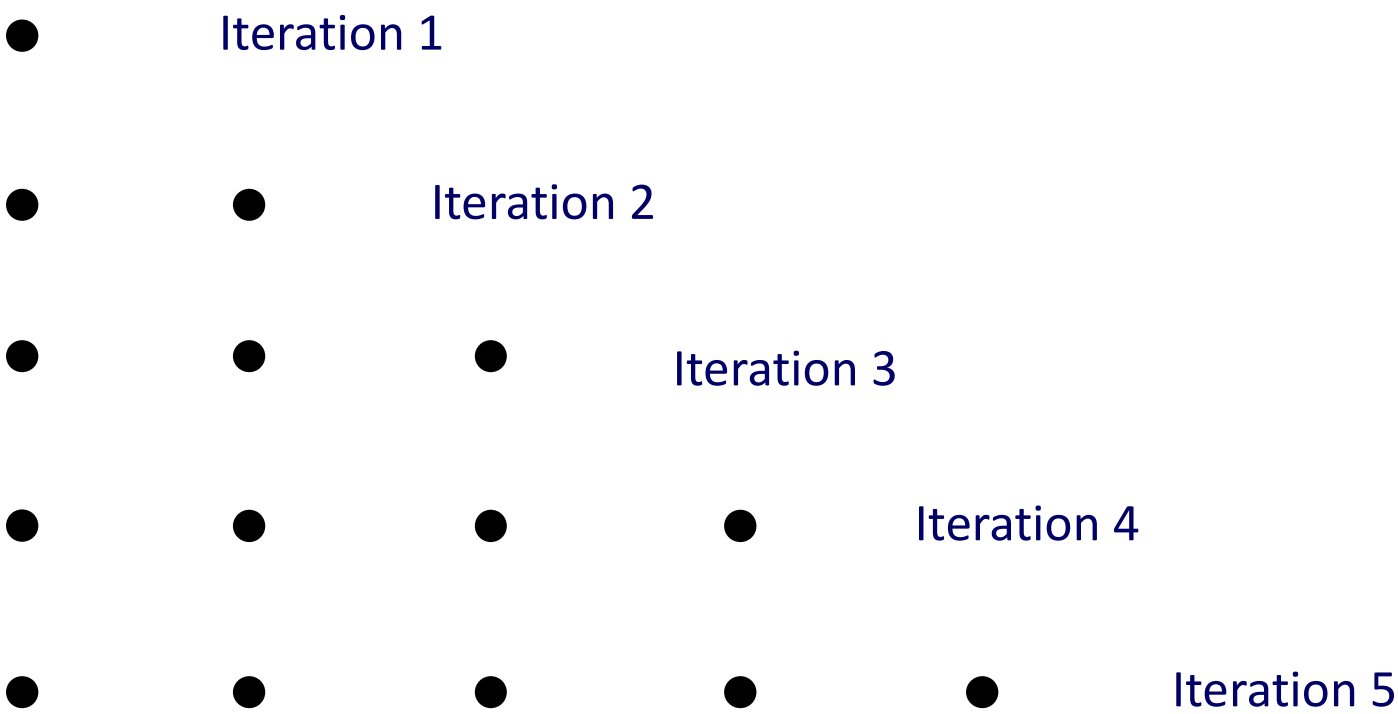
Importance of Greedy Methodology

- A Greedy Methodology is an iterative procedure (Algorithm) that iteratively generates a complete solution for the problem at hand
- The basic structure of the method is the following (although variations exist):
- Initially, *solution* S is empty (does not contain any solution feature)
- **A Greedy Methodology constructs a complete solution by iteratively inserting a solution feature to the partial solution S, until a complete solution is generated**
- **The solution feature to be inserted to the solution are selected according to two basic criteria**
- Solution Features are selected according to a selection criterion
  - Feasibility

    The inserted solution features must satisfy all the examined constraints The inserted features must lead to feasible solutions
  - Quality

    The inserted features must lead to the minimal additional cost change of the partial solution (or the maximum profit change, if a maximization problem is tackled)

Solution construction by iteratively adding points

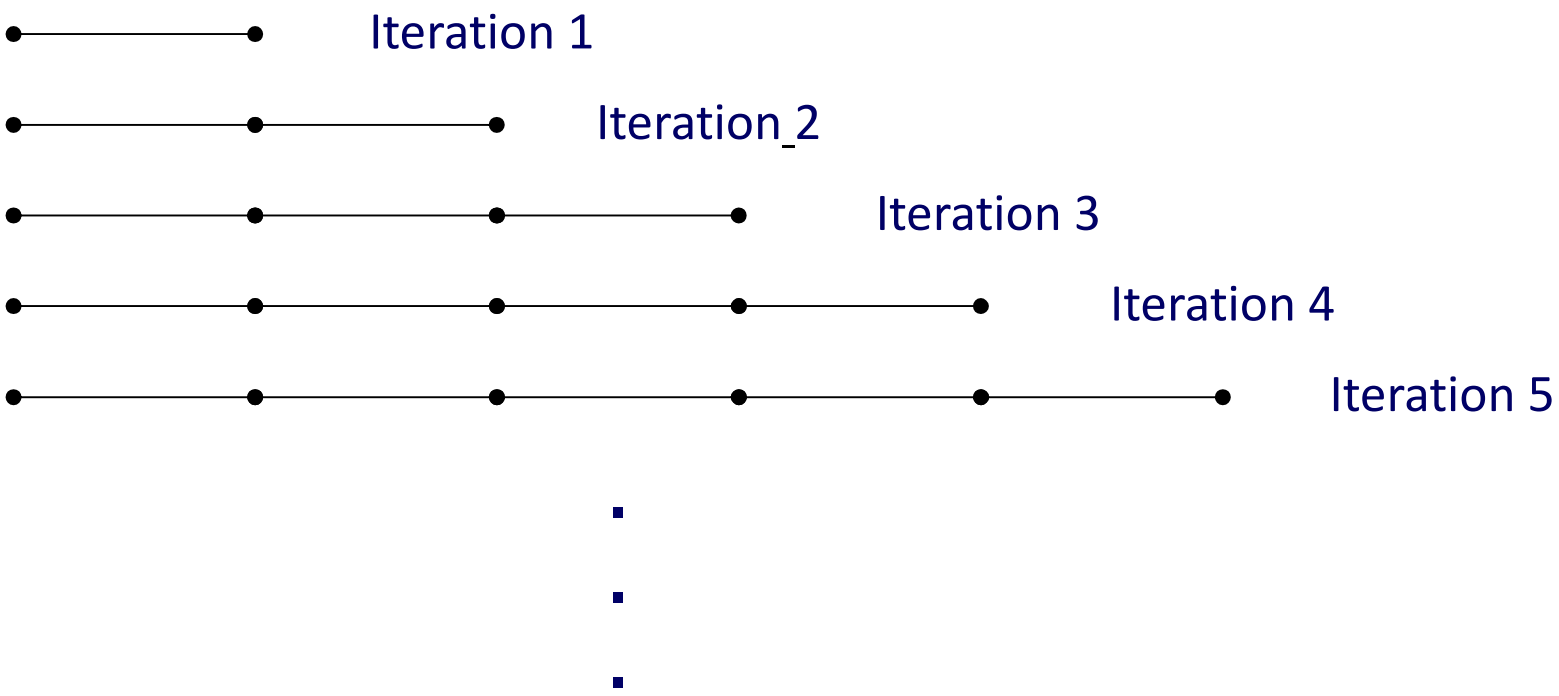


For the tourist trip problem discussed

Solution Shape:
A permutation of 16 cities

Solution Feature: A city (node)

Solution construction by iteratively adding edges/connections



For the tourist trip problem discussed	Solution Shape: A permutation of 15 links	Solution Feature: A link between two cities (Arc)

Importance of the (solution element) selection criterion

- As mentioned, solution features are selected according to a selection criterion that guarantees
  - All problem constraints are satisfied
  - The additional profit is maximized (Maximization problems), or the additional cost is minimized (Minimization Problems).
- The more effective the selection criterion, the higher the quality of the final solution
- If the selection criterion is not appropriately defined, then the solution characteristics incorporated in the solution are low quality
- Thus, a “bad” selection criterion design lead to a low-quality final solution
- In other words, the selection criterion is the “core” of a greedy method and defines the effectiveness of the method.

Solution Quality – Objective Function

- The quality of a solution can be measured (quantified)
- **Objective Function** responsible for quantifying the quality of a solution
- Objective Function
  - associates a complete solution with the total cost (or profit) that this solution brings
  - It is a function z: $S* \rightarrow \mathbb{R}$,
    where $S*$: The set of all feasible solutions of the problem
  - Minimization of the total kilometers travelled:
    $$z(S) = 311\ km$$
  - Minimization of the total time required for producing an order:
    $$z(S) = 132\ hr$$
  - Minimization Problem:
    Solution s_1 is better than $s_2 \iff z(S_1) < z(S_2)$
  - Maximization Problem:
    Solution s_1 is better than $s_2 \iff z(S_1) > z(S_2)$
- The selection criterion of a Greedy method must be compatible with the objective function of the problem

Example: Objective function selection

- It is 7.00 AM, and the water providing company has received 14 calls for technical help / repairs in the areal of Athens
- The service vehicle starts off from the water company headquarters and provides service to the 14 locations. Then it returns to the company headquarters.
- Suggest an objective function which is suitable for the above described application.

Example: Objective function selection

- Goal of the management is to route the company vehicle, so that the total travel time is minimized
- Note: In the beginning, no estimation on the time required for each service can be made.
- By minimizing the total duration of the complete vehicle tour the company manages to
  - Provide service to all customers quickly
  - The servicemen will be again available for provide additional work as soon as possible
  - Minimize fuel costs
  - Save money from the maintenance of the vehicle

Example: Objective function selection

- How should the objective of the problem be modified if there was information on the severity (priority level) of each repair request?
- For instance three urgency levels:
  - 1: High Priority
  - 2: Normal Priority
  - 3: Low Priority

Example: Objective function selection

- Requests: 1, 4, 10 – Priority Level 1
- Requests : 2, 3, 5, 11 – Priority Level 2
- Requests : 6, 7, 8, 9, 10, 12, 13, 14 - Priority Level 3
- Let $t_1(S)$: The time when the last PL 1 request is served in solution S
- Let $t_2(S)$: The time when the last PL 2 request is served in solution S
- Let $t_3(S)$: The time when the last PL 3 request is served in solution S
- Let three positive values M_1, M_2, M_3 for which:

$$M_1 \gg M_2 \gg M_3$$

- Objective for the problem:

$$z(S) = M_1 \cdot t_1(S) + M_2 \cdot t_2(S) + M_3 \cdot t_3(S)$$

Example: Objective function selection

- This objective function promotes solutions that
  - Firstly, provide service to requests of PL1
  - Then, provide service to requests of PL2
  - Finally, provide service to requests of PL3

- At the same time, this objective makes sure that each priority level solution segment (part of the solution that contains requests of the same PL) requires the minimum travel time.

Objective function selection

- We could leave the objective function intact and still satisfy the urgency level requirements
- Objective Function: Minimization of the Travel Cost

- Modification of the Problem Data: Cost Matrix (!)

- Any Suggestions/Ideas?

Objective function selection

- We do not want to travel any
  - PL3 -> PL1 arcs
  - PL3 -> PL2 arcs
  - PL2 -> PL1 arcs
- Thus, for these arcs, set: $c'_{i,j} \leftarrow c_{i,j} + M$
  where M is a very large positive value
- Thus, the optimal solution will try to avoid any 'penalized' arcs

Structure of A Greedy Optimization Methodology

To apply a Greedy Optimization Methodology for solving a Management Science problem, the following must be determined

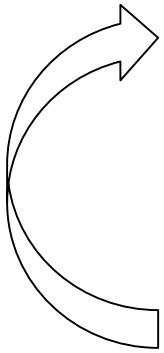
Problem Level

- Shape of the solution for the problem
- Objective function of the problem

Methodology Level

- Solution Feature: The basic building block of a solution (The features which jointly form a complete solution)
- Selection Criterion: The criterion according to each solution features are iteratively selected and incorporated in the partial solution

Structure of A Greedy Optimization Methodology



while (solution S is not complete)

Identify the solution feature s which according to the employed selection criterion

Insert the selected solution feature s into the partial solution S

Evaluate the quality (objective function) of the generated, complete solution S (calculate $z(S)$)

Importance of Greedy Algorithms

- Mathematical methods of Operations Research (Linear Programming, Integer Programming) which fail to solve problem instances of realistic sizes
  - Medium and Large-Scale applications
  - Huge Solution Space
- Greedy Algorithms can provide a high-quality solution within seconds of CPU time
- Greedy algorithms generate a single solution (ignoring other solutions in the solution space) and this allows them to be fast
- Applicable to huge problems
  - For the trip problem, imagine a trip of plenty of thousands of cities

Importance of Greedy Algorithms

Note that the aforementioned comments on the exact methods limitations refer to brute-force approaches. For some emblematic problems (TSP), after 70 years of technological and research advances:

> Given a collection of points, the TSP asks for the shortest route to visit them all. Simple enough. But even a whisper of the problem strikes fear in the heart of the computing world. Last year, a Washington Post article reported it would take "1,000 years to compute the most efficient route between 22 cities." This claim, however, ignores 70 years of intense study in the OR community. A 22-city TSP can be handled in a snap with modern algorithms, even on an iPhone. Going larger, we describe techniques that have been used to solve to precise optimality examples with nearly 50,000 points and Google Map walking distances. And if you need to visit the nearest 2,079,471 stars, there is a route, ready to go, that is guaranteed to be no more than 1.00002 times longer than a shortest possible tour.

Recap – Greedy Algorithms

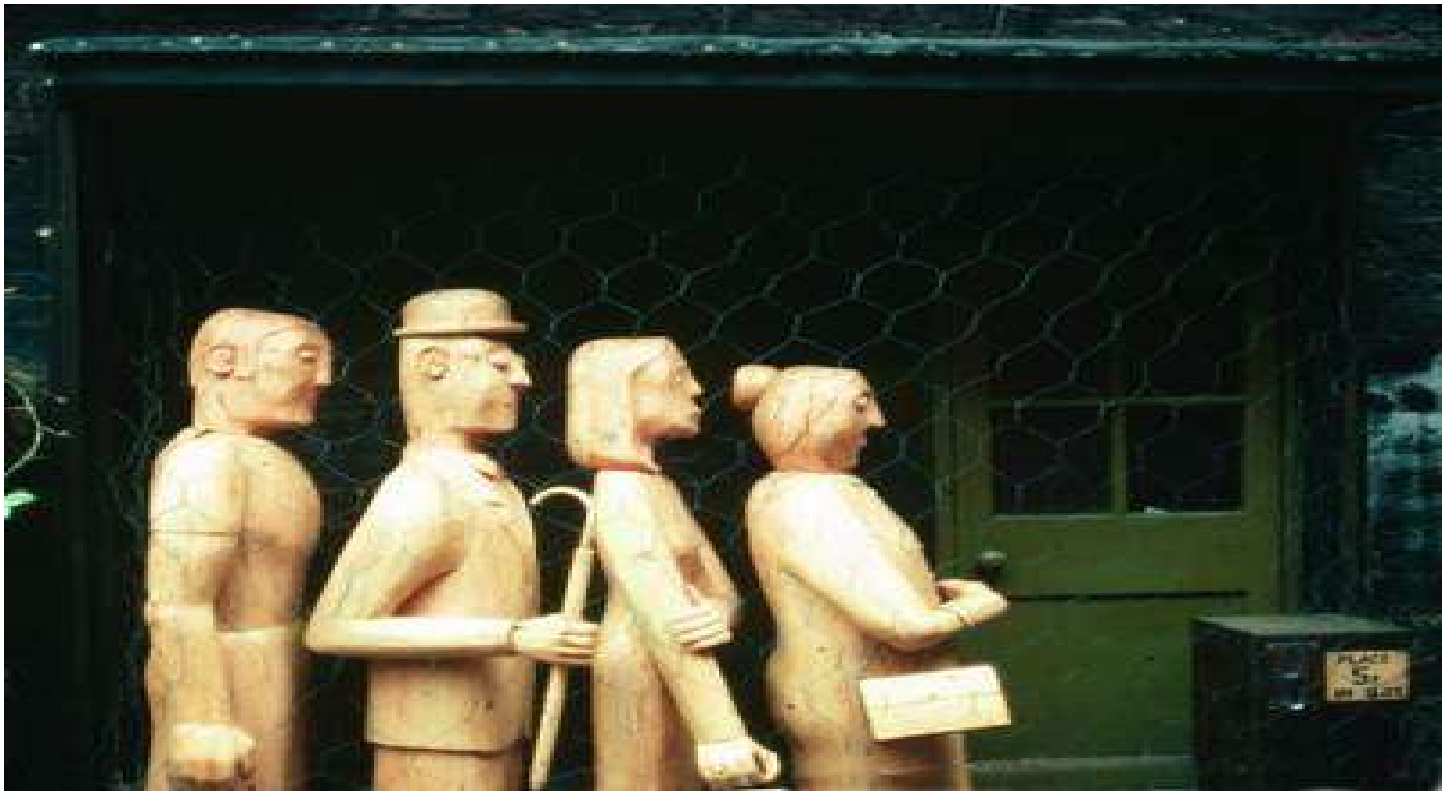
- Greedy Algorithms solve problems, the solutions of which have a combinatorial shape (structure)
- Greedy Algorithms are iterative procedures of strictly defined steps
- Greedy Algorithms gradually (iteratively) construct a single solution for the examined problem
- Greedy Algorithms are aimed at constructing a high-quality solution (not necessarily the optimal one)
- Greedy Algorithms can solve huge problem instances in short CPU time

Combinatorial Optimization Problems

- Three main categories of combinatorial optimization problems
  - Ordering Problems
  - Assignment Problems
  - Selection Problems

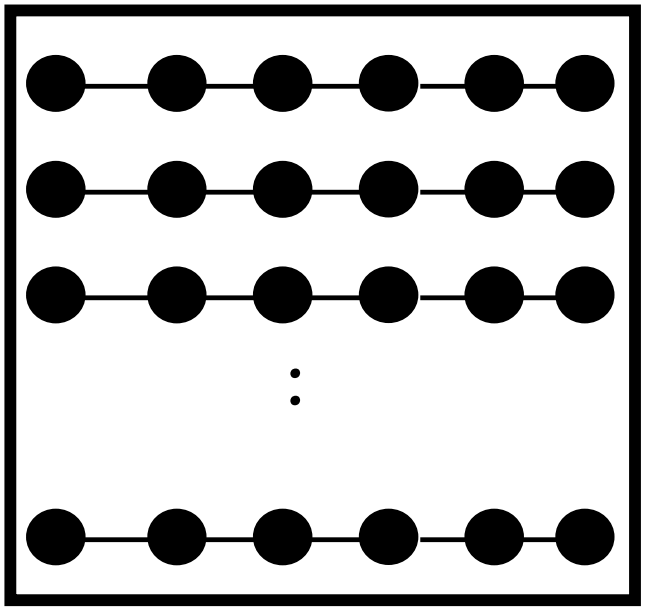
  Depending on the solution shape (structure)
- Note: This is not a strict categorization.
  - Problems can share characteristics of multiple categories
  - Each category can be translated to another category
- Recall, that the term combinatorial refers to the fact that the solution of these problems can be encoded using tools taken from Combinatorial Analysis (Permutations, Sets, etc…)
- Thus, an ordering problem is a problem, the solution of which is formed by a permutation of the solution features
  - Solution = Sequence of the points = (Athens, Belgrade, Vienna, …, Athens)

Permutation Problems



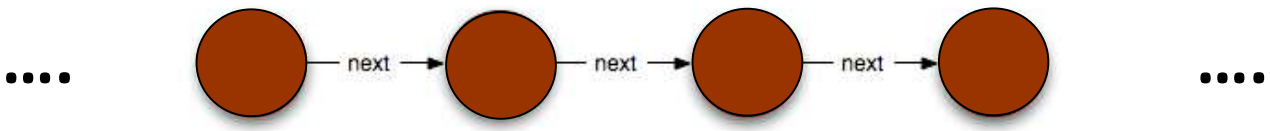
Permutation Problems

- A combinatorial optimization problem is a permutation problem when the solution structure of this problem is a permutation (or a sequence of permutations)



1st Combinatorial Structure: Permutation

- A permutation of objects is not a set of objects
- In a permutation, the ordering of the object matters
  - Sets are unordered
  - For instance, sets {A, B, C} and {C, A, B} are the same sets
  - Permutations (A, B, C) and (C, A, B) are not the same!
- Different Permutations correspond to different solutions
- The objective of permutation problems requires the definition of a cost matrix which contains the cost of the links connecting the objects of the permutation



Management Science & Ordering Problems

- Aim of permutation problems:

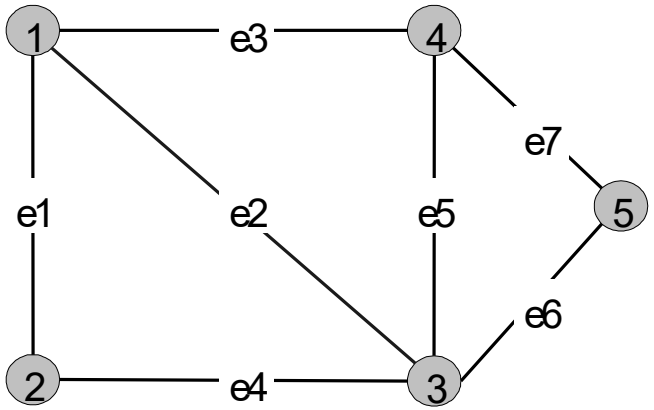
  Generation of a single, or multiple permutations (sequence) of solution features

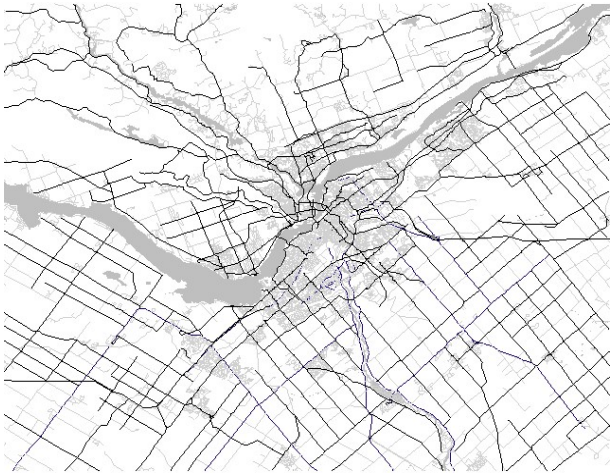
- Two emblematic ordering/permutation problems which correspond to multiple operational scenarios faced by companies and organizations
  - Traveling Salesman Problem - TSP
  - Vehicle Routing Problem - VRP

Management Science & Ordering Problems

- Permutation problems are “naturally” represented by graphs
- Graphs provide a formal way of describing an ordering problem:
  - Nodes: $V = \{1, 2, \ldots, m\}$, these are locations, cities, service points, etc.
  - Arcs: $A = \{e_1, e_2, \ldots, e_n\}$, these are the links connecting two nodes
  - Note: If the direction of a link matters (the link is called arc)

    If the direction of the link does not matter (fully symmetric problems), the links are called edges





Management Science & Ordering Problems

- A graph can represent a road network in which a transportation operation takes place (goods distribution, people distribution)
  - Nodes: Customers, Vehicle Station
  - Arcs: Represent the direct transition between a pair of nodes

- Each arc is associated with a cost (weight):
  - This weight can represent the distance, time, money, fuel etc. that must be paid for implementing the transition between the corresponding nodes

Transportation Logistics Management – Problem Solving by Greedy Algorithms

- Company distributes oil products in the region of Attica
- The oil truck of the company needs to satisfy the orders of 4 major industrial customers
- One single visit must take place to each customer
- We do not want to be accused of discriminating between our customers
  - Our aim is to minimize the time difference between the service time of customers
- The travel times between every customer pair are available
- Propose an objective function for the problem

From \ To	A	B	C	D
A	0	3	14	17
B	3	0	12	16
C	13	12	0	4
D	14	15	2	0

Transportation Logistics Management – Problem Solving by Greedy Algorithms

- To minimize the difference of service times between our customers, the best thing we can do is to:

Minimize the travel time required for travelling from the customer served first to the customer served last

- In other words,

Minimize the travel time of the open path connecting the four customers

Transportation Logistics Management – Problem Solving by Greedy Algorithms

- Design a Greedy Method to solve the examined problem

To

From

	A	B	C	D
A	0	3	14	17
B	3	0	12	16
C	13	12	0	4
D	14	15	2	0

Example solution

- To design a Greedy Algorithm for the examined problem, we must firstly understand the basic characteristics of the problem
  - What is the solution structure?
  - What are we going to define as solution feature
  - What is going to be the selection criterion to be incorporated in the Greedy Algorithm

Example solution: Nearest Neighbor Selection Criterion

- Solution Structure:
  - Searching for the solution minimizing the service time between the last and the first visit
  - We are looking for a permutation of the four nodes
  - This permutation defines the visit order: (A, C, B, D)
- Objective Function:
  - Minimization of the open path connecting the four customers
  - Minimization of the total cost required for travelling along the three arcs connecting the four nodes
  - $S = (A, C, B, D), z(S) = t_{AC} + t_{CB} + t_{BD}$
- Solution Feature:
  - The arc connecting a pair of customers

Example solution: Nearest Neighbor Selection Criterion

- Selection Criterion:
  - If the solution is not empty:

  Select the arc

  (a) starting from the last customer of the partial solution,

  (b) going to a customer not in the partial solution and

  (c) minimizing the travel time
  - If the solution is empty:

  Select the arc minimizing the travel time

Example solution: Nearest Neighbor Selection Criterion

- Iteration 1. Select the arc minimizing the travel time: DC, t_{DC}

  Therefore s: $\{(D, C)\}$

- Iteration 2. Select the arc (a) starting from the last customer of the partial solution, (b) going to a customer not in the partial solution and (c) minimizing the travel time: CB, t_{CB}

  Therefore, s: $\{(D, C), (C, B)\}$

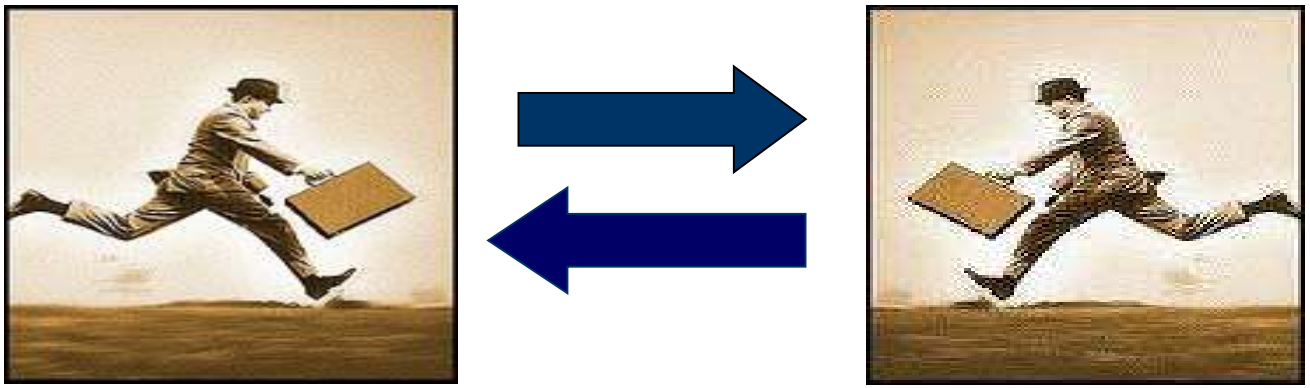
- Iteration 3. Select the arc (a) starting from the last customer of the partial solution, (b) going to a customer not in the partial solution and (c) minimizing the travel time: BA, t_{BA}

  The final solution is s: $\{(D, C)(C, B)(B, A)\}$

  The objective of the final solution is $z(s) = 17$

The Travelling Salesman Problem - TSP

- Given the costs required for travelling between every node pair, the standard Traveling Salesman Problem (TSP) is aimed at generating the route (node sequence) that:
  - The total cost of the route is minimized
  - The route is a cycle (the route begins from the central location and returns back to the central station)
  - The route contains all nodes just once



TSP – DFJ Mathematical Formulation

Dantzig-Fulkerson-Johnson formulation

Label the cities with the numbers 1, …, *n* and define:

$$x_{ij} = \begin{cases} 1 & \text{the path goes from city } i \text{ to city } j \\ 0 & \text{otherwise} \end{cases}$$

Take c_{ij} to be the distance from city *i* to city *j*. Then TSP can be written as the following integer linear programming problem:

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

$$0 \leq x_{ij} \leq 1 \qquad i, j = 1, \ldots, n;$$

$$\sum_{i=1, i \neq j}^{n} x_{ij} = 1 \qquad j = 1, \ldots, n;$$

$$\sum_{j=1, j \neq i}^{n} x_{ij} = 1 \qquad i = 1, \ldots, n;$$

$$\sum_{i \in Q} \sum_{j \in Q} x_{ij} \leq |Q| - 1 \qquad \forall Q \subsetneq \{1, \ldots, n\}, |Q| \geq 2$$

TSP – MTZ Mathematical Formulation

Miller-Tucker-Zemlin formulation

Label the cities with the numbers 1, ..., *n* and define:

$$x_{ij} = \begin{cases} 1 & \text{the path goes from city } i \text{ to city } j \\ 0 & \text{otherwise} \end{cases}$$

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

$$x_{ij} \in \{0, 1\} \qquad i, j = 1, \ldots, n;$$

$$u_i \in \mathbf{Z} \qquad i = 1, \ldots, n;$$

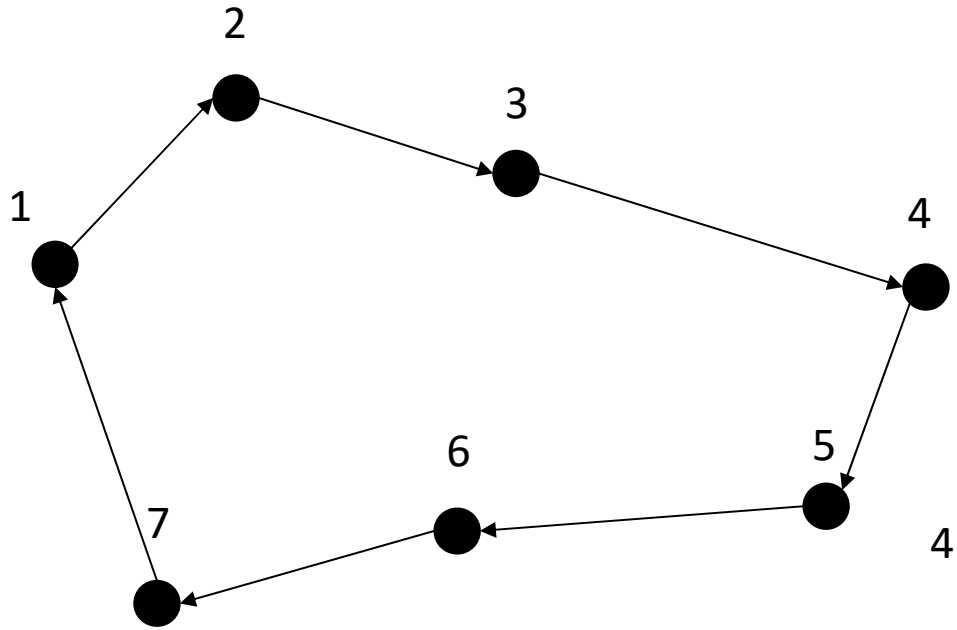
$$\sum_{i=1, i \neq j}^{n} x_{ij} = 1 \qquad j = 1, \ldots, n;$$

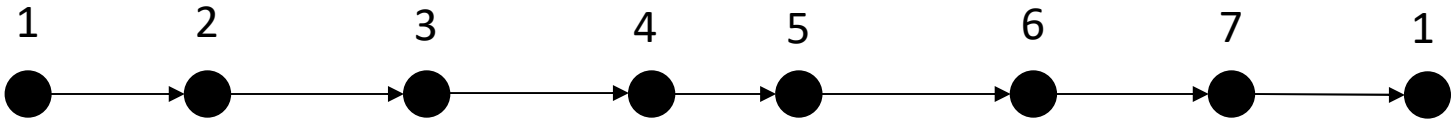
$$\sum_{j=1, j \neq i}^{n} x_{ij} = 1 \qquad i = 1, \ldots, n;$$

$$u_i - u_j + n x_{ij} \leq n - 1 \qquad 2 \leq i \neq j \leq n;$$

$$0 \leq u_i \leq n - 1 \qquad 2 \leq i \leq n.$$

Illustration of a TSP solution





TSP Example

- We are located in the central warehouse (depot) of a production site
- We must satisfy the product requests received by 5 customers
- Our company vehicle is going to be routed to start from the depot, visit the five customers and then return to the depot
- The cost for moving between every node pair is given in the following symmetric cost matrix:

	0	**1**	**2**	**3**	**4**	**5**
0	0	52	37	46	47	28
1	52	0	84	90	97	41
2	37	84	0	11	15	47
3	46	90	11	0	20	50
4	47	97	15	20	0	61
5	28	41	47	50	61	0

TSP Example

- Solve the problem with two greedy methods
  - 1st Greedy Method (Nearest Neighbor):

  Starting from the depot (0), iteratively insert after the last node in the partial route, the node which minimizes the additional distance travelled

  - 2nd Greedy Method (Minimum Insertions):

  Starting from the partial solution (0,0), iteratively insert the unrouted customer into the insertion point which leads to the minimum objective function change

  Note:

  All unrouted customers must be checked

  All insertion positions must be checked

TSP Example

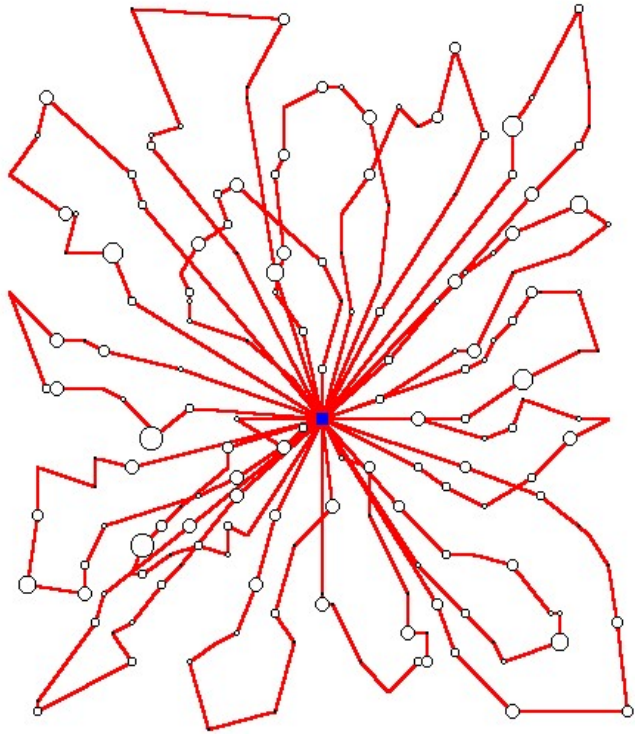
- Greedy Method 1 – Nearest Neighbor
  - Initial Solution: $s = \{0\}$
  - Solution after the 1st insertion: $s = \{0, 5\}$
  - Solution after the 2nd insertion: $s = \{0, 5, 1\}$
  - Solution after the 3rd insertion: $s = \{0, 5, 1, 2\}$
  - Solution after the 4th insertion: $s = \{0, 5, 1, 2, 3\}$
  - Solution after the 5th insertion: $s = \{0, 5, 1, 2, 3, 4\}$
  - Solution after the insertion of the depot: $s = \{0, 5, 1, 2, 3, 4, 0\}$
  - $z(s) = 231$

TSP Example

- Greedy Method 2 – Minimum Cost Insertions
  - Initial Solution: $s = \{0,0\}$
  - Solution after the 1st insertion (Customer 5, Position 1): $s = \{0,5,0\}$
  - Solution after the 2nd insertion (Customer 2, Position 2): $s = \{0,2,5,0\}$
  - Solution after the 3rd insertion (Customer 3, Position 2): $s = \{0,2,3,5,0\}$
  - Solution after the 4th insertion (Customer 4, Position 2): $s = \{0,2,4,3,5,0\}$
  - Solution after the 5th insertion (Customer 1, Position 5): $s = \{0,2,4,3,5,1,0\}$

  - $z(s) = 215$

VEHICLE ROUTING PROBLEM - VRP



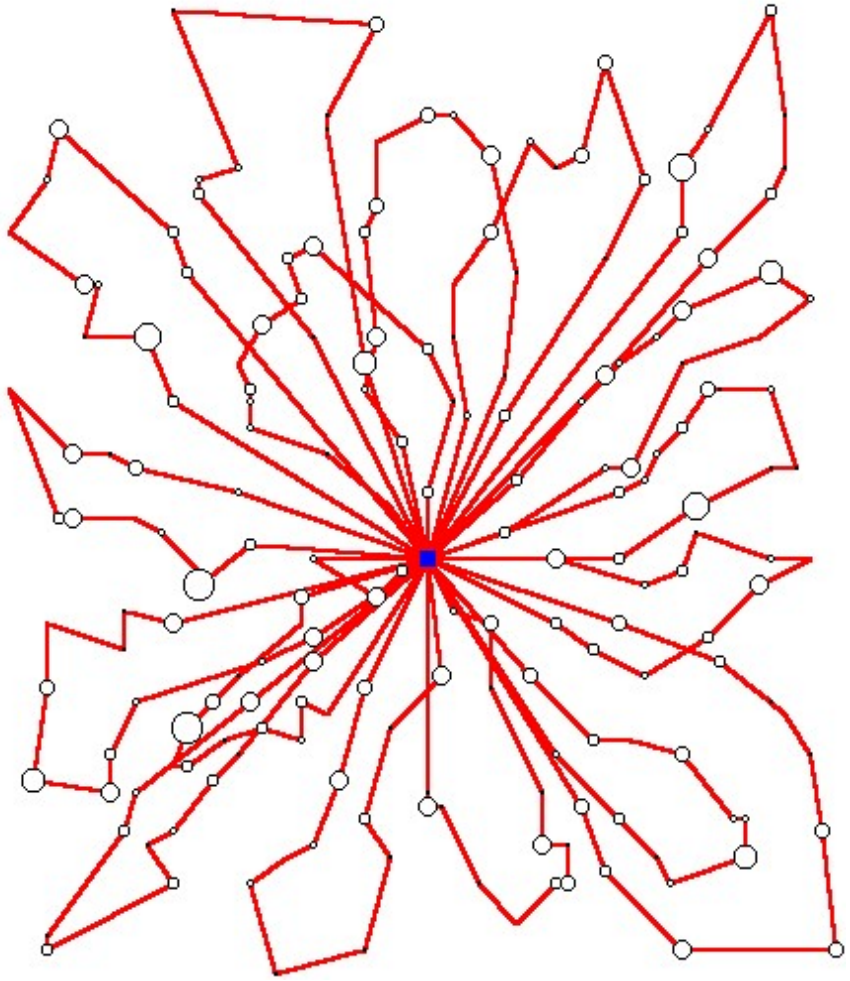
The Vehicle Routing Problem - VRP

- Let V be a set of nodes
  - N: customers
  - 0: depot
- With each customer in N, is associated a predetermined demand quantity
- Let K be a set of vehicles, each with capacity Q

The Vehicle Routing Problem - VRP

- The aim of the VRP is to generate a set of routes (one route for every vehicle), such that:
  - The total cost of all routes is minimized
  - Each customer is served once by exactly one vehicle (no splitting of the orders is allowed)
  - The total demand of the customers visited by a route does not exceed the capacity of the vehicle Q
  - Each route is a closed cycle (Vehicle leaves the depot, serves customers, returns to the depot)

Illustration of a VRP Solution



A VRP Analytic Solution: A set of routes

– 199 customers, 17 vehicles

0	6	85	61	16	141	44	119	191	91	193	98	92	151	0
0	26	149	195	179	110	198	72	74	171	73	180	53	0	
0	5	84	173	17	113	86	140	38	14	192	100	37	97	117	0
0	18	114	8	174	45	125	199	83	60	118	166	0		
0	69	101	70	30	160	131	32	181	63	126	90	108	10	189	0
0	54	130	165	55	25	170	67	39	187	139	4	155	0		
0	1	122	128	20	188	66	9	120	81	33	157	50	0		
0	27	167	127	190	31	162	132	176	111	28	0				
0	152	58	2	178	57	15	43	142	42	172	144	87	13	0	
0	51	103	161	71	65	136	35	135	164	34	78	169	129	0	
0	82	46	124	168	47	36	143	49	64	11	17	107	19	123	0
0	105	40	21	197	56	186	23	75	133	22	115	145	115	137	0
0	102	185	79	158	3	77	116	196	76	184	138	0			
0	156	112	0												
0	183	94	95	59	93	99	104	96	147	89	0				
0	146	88	148	159	62	182	48	7	194	106	153	52	0		
0	109	177	134	163	24	29	121	68	80	150	12	154	0		

Total distance = 1311.48

Distribution Optimization – Greedy Algorithm

- An oil distribution company (node 0) owns a private fleet of two oil trucks
- Each truck has a maximum carrying capacity of 21.000 lt
- The company ais mto generate the optimal routing of the two trucks to satisfy the orders raised by a set of 9 customers (nodes 1-9)
- Operational Constraints:
  - Vehicles start from the depot and return to the depot.
  - All customers must be served
  - Each customer is visited once by a single vehicle.
  - The total demand of the customers assigned to a vehicle is less than 21.000 lt.

Transportation Management – Solving via Greedy Algorithm

In the following, the cost matrix and the customer orders

	0	1	2	3	4	5	5	7	8	9
0	0	21	56	89	56	25	64	41	48	55
1	21	0	29	34	52	36	12	55	35	67
2	56	29	0	32	72	56	64	81	89	26
3	89	34	32	0	36	64	81	95	99	18
4	56	52	72	36	0	29	55	47	65	39
5	25	36	56	64	29	0	40	56	62	46
6	64	12	64	81	55	40	0	29	31	43
7	41	55	81	95	47	56	29	0	11	46
8	48	35	89	99	65	62	31	11	0	36
9	55	67	26	18	39	46	43	46	36	0

Transportation Management – Solving via Greedy Algorithm

Customer demands (thousands of lt)

Customer	1	2	3	4	5	6	7	8	9
Demand	2	4	5	1	7	3	5	4	4

Transportation Management – Solving via Greedy Algorithm

- Solution Shape: Two node permutations each starting and ending at the depot 0.
- Solution Feature: An arc connecting two nodes
- Solution Initialization: Depot insertion for both permutations
- Selection Criterion:

At each iteration, insert the arc which

1. Starts from the last nodes of the given routes*
2. Ends at a customer not violating any problem constraint
3. Minimizes the corresponding cost

(Variation of NN, considering the last customer of every route)

*Two starting nodes for our problem

Transportation Management – Solving via Greedy Algorithm

- The candidate arcs to be inserted must not lead to any problem constraint:

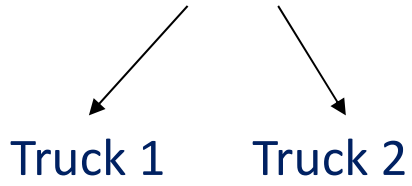
a) The inserted arcs must not end at customers already present in the solution

b) The inserted arcs must not end at customers which violate the 21.000 lt capacity of the oil trucks

Transportation Management – Solving via Greedy Algorithm

- Initialization of the partial solution:

$$s_0 = \{\{0\}, \{0\}\}$$



Transportation Management – Solving via Greedy Algorithm

- From the candidate feasible arcs, select the one with the minimum cost
- Iteration 1. $s_1 = \{\{\mathbf{0}, \mathbf{1}\}, \{0\}\}$
- Iteration 2. $s_2 = \{\{0, \mathbf{1}, \mathbf{6}\}, \{0\}\}$
- Iteration 3. $s_3 = \{\{0, 1, 6\}, \{\mathbf{0}, \mathbf{5}\}\}$
- Iteration 4. $s_4 = \{\{0, 1, 6\}, \{0, \mathbf{5}, \mathbf{4}\}\}$
- Iteration 5. $s_5 = \{\{0, 1, 6, \mathbf{7}\}, \{0, 5, 4\}\}$
- Iteration 6. $s_6 = \{\{0, 1, 6, \mathbf{7}, \mathbf{8}\}, \{0, 5, 4\}\}$

Transportation Management – Solving via Greedy Algorithm

- Iteration 7. $s_7 = \{\{0, 1, 6, 7, 8\}, \{0, 5, \mathbf{4}, \mathbf{3}\}\}$
- Iteration 8. $s_8 = \{\{0, 1, 6, 7, 8\}, \{0, 5, \mathbf{4}, \mathbf{3}, \mathbf{9}\}\}$
- Iteration 9. $s_9 = \{\{0, 1, 6, 7, 8\}, \{0, 5, \mathbf{4}, \mathbf{3}, \mathbf{9}, \mathbf{2}\}\}$
- Iteration 10. $S = s_{10} = \{\{0, 1, 6, 7, \mathbf{8}, \mathbf{0}\}, \{0, 5, 4, 3, 9, \mathbf{2}, \mathbf{0}\}\}$
  (Return to the central depot for both trucks)

Objective Function for the final solution: $\boldsymbol{z(S) = 311\ km}$

Transportation Management – Solving via Greedy Algorithm

- Greedy Methods for the VRP:
  - Clarke & Wright heuristic
  - Sweep method

Clarke & Wright heuristic

- Given a set of n nodes
  - Depot: 0
  - Customers: $\{1, 2, \dots, n\}$
  - Cost Matrix $c_{ij},\ i = 0, \dots, n,\ j = 0, \dots, n$
- For each arc i, j calculate the metric (Saving):

$$s_{ij} = c_{i0} + c_{0j} - c_{ij}$$

- Arrange the obtained savings in decreasing order
- Initialization: Construct one route for each customer
  - {0, 1, 0}
  - {0, 2, 0}
  - {0, n, 0}

Clarke & Wright heuristic

- In every greedy algorithm iteration
  - Select the highest saving in the descending list of savings with customer edges: (i, j) (for problems with symmetric distance matrix only use $(i < j)$)
  - Merge the routes that contain customers i and j with edge (i, j) iff
    - Node i and Node j belong to different routes
    - Node i is the first or last of the corresponding route
    - Node j is the first or last of the corresponding route
    - Merging the routes containing i and j does not violate the vehicle capacity
  - Stop if no other route merges are allowed due to capacity restriction or all possible savings are checked

Clarke & Wright heuristic Example

- Distance/Cost matrix

c_{ij}	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	-	12	11	7	10	10	9	8	6	12
1	12	-	8	5	9	12	14	16	17	22
2	11	8	-	9	15	17	8	18	14	22
3	7	5	9	-	7	9	11	12	12	17
4	10	9	15	7	-	3	17	7	15	18
5	10	12	17	9	3	-	18	6	15	15
6	9	14	8	11	17	18	-	16	8	16
7	8	16	18	12	7	6	16	-	11	11
8	6	17	14	12	15	15	8	11	-	10
9	12	22	22	17	18	15	16	11	10	-

Clarke & Wright heuristic Example

- Customer demands

i	1	2	3	4	5	6	7	8	9
d_i	10	15	18	17	3	5	9	4	6

- Vehicle capacity: $Q = 40$

Clarke & Wright heuristic Example

- Savings calculation matrix

s_{ij}	1	2	3	4	5	6	7	8	9
1		15	14	13	10	7	4	1	2
2			9	6	4	12	1	3	1
3				10	8	5	3	1	2
4					17	2	11	1	4
5						1	12	1	7
6							1	7	5
7								3	9
8									8

Clarke & Wright heuristic Example

- Savings in descending order:

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

Clarke & Wright heuristic Example

- Steps synopsis

Edge [4,5]: Join cycles 0-4-0 and 0-5-0: result **0-4-5-0**, load $d_4 + d_5 = 20 < K$.
Edge [1,2]: Join 0-1-0 and 0-2-0: result **0-1-2-0**, load $d_1 + d_2 = 25 < K$.
Edge [1,3]: Capacity limit: $d_1+d_2+d_3 = 43 > K$.
Edge [1,4]: Capacity limit: $d_1+d_2+d_4 = 42 > K$.
Edge [2,6]: Join cycles 0-1-2-0 and 0-6-0: result **0-1-2-6-0**, load $d_1+d_2+d_6 = 30 < K$.
Edge [5,7]: Join cycles 0-4-5-0 and 0-7-0: result **0-4-5-7-0**, load $d_4+d_5+d_7 = 29 < K$.
Edge [4,7]: Condition 4(i) doesn't hold: nodes belong to same cycle.
Edge [1,5]: Condition 4(iii) doesn't hold: node 5 is an interior node of its route.
Edge [3,4]: Capacity limit: $d_3+d_4+d_5+d_7 = 47 > K$.
Edge [2,3]: Condition 4(iii) doesn't hold: node 2 is an interior node.
Edge [7,9]: Join cycles 0-4-5-7-0 and 0-9-0: result **0-4-5-7-9-0**,
load $d_4+d_5+d_7+d_9 = 35 < K$.
Edge [3,5]: Condition 4(iii) doesn't hold: node 5 is an interior node.
Edge [8,9]: Join cycles 0-4-5-7-9-0 and 0-8-0: result **0-4-5-7-9-8-0**,
load $d_4+d_5+d_7+d_9+d_8 = 39 < K$.
Edge [1,6]: Condition 4(i) doesn't hold: nodes belong to same cycle.
Edge [5,9]: Condition 4(i) doesn't hold: nodes belong to same cycle.
Edge [6,8]: Capacity limit: $(d_1+d_2+d_6)+(d_4+d_5+d_7+d_9+d_8) = 69 > K$.

Clarke & Wright heuristic Example

- Final solution $s =$

Route:	Load:	Cost:
0-3-0	18	14
0-1-2-6-0	30	37
0-4-5-7-9-8-0	39	46

- Objective function value:

$$z(s) = 97$$

Example

- Your position is at the HR dept of a company
- For a new opening, four interviews will take place (each for every candidate)
- Depending on the candidate expertise, the interview panel will consist of different company employees (your colleagues)
- Interviews will take place at the following timeslots [9:00-10:00], [10:00:11.00], [11:00:12.00] και [12:00-13.00] (each interview lasts an hour)
- Your objective is to minimize the times that your colleagues will participate in an interview for two consecutive hours
- Suggest: Solution Shape, Objective Function
- If you have a TSP solver, What cost matrix should be fed to the TSP solver for constructing the timetable

	Cand A	Cand B	Cand C	Cand D
Helen	*			*
Peter		*	*	
Mary	*	*	*	*
Emma	*		*	
James		*	*	*