Homework Problems

Problems

- Knapsack Problem (KSP)
- Traveling Salesman Problem (TSP) * homework
- Graph Coloring Problem (GCP) * homework
- Vehicle Routing Problem (VRP)
- Warehouse Location Problem (WLP) * homework

Recommended Approach

- First, implement a very simple approach, e.g., a simple greedy algorithm.
- Then, inspect the solution you obtain.
- Based on your insights and the feedback you receive from the test system, look for ways to improve your results.
- Hybrid approaches may give best results.
- Remember that you can use different approaches even for the different instances of the same problem.

Greedy Approaches

- Constructing a solution by determining the value of one decision variable at a time
- At each step of the solution construction, pick the value which gives the best result for the objective
- Greedy approaches usually provide poor quality solutions, especially for large instances
- Greedy approaches are usually hybridized with other techniques, e.g., local search techniques

Disclaimer

- The formulations and representations we provide in the course handouts may not always be the best choices for achieving the highest quality solutions.
- Always remember to consider other formulations and representations for the problems.

5

Knapsack Problem (KSP)

- Given a knapsack with capacity C and a set of items I, where each item $i \in I$ has a resource requirement r_i and a profit p_i, the objective is to find a subset of items of I which has a maximum value and does not exceed the knapsack capacity C.
- (There are many variants of the KSP. This is the definition of the 1-Dimensional KSP.)

6

Knapsack Problem

maximize $\sum_{i=1 \ldots n} p_i * x_i$

subject to $\sum_{i=1 \ldots n} r_i * x_i \leq C$

where

- n is the number of items,
- $x_i \in \{0,1\}$ are the decision variables,
- p_i are the profits of the items, and
- r_i are the resource requirements of the items.

Knapsack Problem

- Possible solution candidates:

  (0 0 ... 0) (0 0 ... 1) ... (1 1 ... 1)

- Not all solution candidates are feasible, i.e., those whose total weights exceed the knapsack capacity are infeasible.

Knapsack Problem

Simple Greedy Approach for the Knapsack Problem

- Step 1: Choose the first item from the set of items.
- Step 2: Check whether adding the item causes the total resource requirement of all the selected items to exceed the capacity *C*.
- Step 3: If not, add item to subset.
- Step 4: Choose next item from the set of items.
- Step 5: Repeat steps 2-4 until all items have been considered.
- Step 6: Return solution, i.e., a subset of items.

Knapsack Problem

- Example:

n=6 C=6

Items	Profits	Resource Req.
1	10	2
2	12	3
3	6	3
4	2	1
5	16	2
6	12	4

- Solution from greedy approach: 1 1 0 1 0 0
- Total profit = 10+12+2=24
- Total resource req. = 2+3+1=6 (OK)

- Another solution: 0 1 0 1 1 0
- Total profit = 12+2+16=30
- Total resource req. = 3+1+2=6 (OK)

- Feasible but unacceptable solution examples:
- 0 0 0 0 0 0
- 0 0 0 0 1 0

Traveling Salesman Problem

A salesman must visit every city in his territory exactly once and return to his starting city. Given the cost of travel between all cities, how should he plan his itinerary to minimize the total cost of his tour?

- Objective: Given a set of points (vertices/nodes), find the shortest tour. A tour is a permutation of vertices.
- Constraint 1: All vertices have to be visited.
- Constraint 2: Each vertex must be visited only once.

In graphs, this corresponds to finding a Hamiltonian cycle, i.e., a cyclic path that visits every vertex exactly once (except the start/end point)

If the points are given as (x,y) coordinates →Euclidean TSP

Euclidean TSP

- Given: a point list with (x_i, y_i) pairs.
- Assume there are *n* points (vertices/nodes) and the *0th* node is the start/end node.
- v_i is the visitation order of node *i*.
- d_{ij} is the Euclidean distance between points *i* and *j*.

  minimize $\sum_{i=0 \ldots n-2} d_{v_i v_{i+1}} + d_{v_{n-1} v_0}$
- The constraints dictate that v_i is a permutation of all the nodes.

Traveling Salesman Problem (TSP)

A Simple Greedy Approach for the TSP: Nearest Neighbor Heuristic (NN)

- Step 1: Start at the start node (node 0 in this case).
- Step 2: At each step, choose the closest node (which has not been visited before) to current location and add it to the solution.
- Step 3: Update current location.
- Step 4: Repeat steps 2 and 3 until all nodes have been visited.
- Step 5: Return solution, i.e., a permutation of nodes.

Traveling Salesman Problem (TSP)

Example:

n=6 (start and end at node 0)

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

- NN solution: 0 2 4 1 5 3 0
- Total distance = 8

- Another solution: 0 1 5 3 4 2 0
- Total distance = 8

- **Please note!** In this problem representation, all permutations of the vertices are feasible.

Graph Coloring Problem (GCP)

- Given a graph $G=(V,E)$, color all the vertices $v_i \in V$ of the graph using a minimum number of colors, in such a way that no two neighbor vertices have the same color.
- Here, V is the set of vertices of the graph and E is the set of edges.
- Assume n is the number of vertices and c_i is the color of node i.

minimize $\quad max_{i=0\ldots n-1} c_i$

subject to $\quad c_i \neq c_j$ if ($< i, j > \in E$)

Graph Coloring Problem (GCP)

<u>A Simple Greedy Approach for the GCP</u>

- Step 1: Start with the first node; assign the first color to first node.
- Step 2: At each step, choose the next node.
  - Assign an existing color to node, if there are no conflicts with its neighbors.
  - If no possible existing color assignments, then assign a new color to node.
- Step 3: Repeat step 2 until all nodes have been colored.
- Step 4: Return solution, i.e., a color assignment to nodes.

Graph Coloring Problem (GCP)

Example:

n=6

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

- Solution from greedy approach:

  solution uses 3 colors

Node	Color
1	C1
2	C2
3	C2
4	C3
5	C3
6	C2

- **Please note!** A solution which uses a different color for each vertex (6 colors) is feasible but unacceptable.

Vehicle Routing Problem (VRP)

A delivery company needs to deliver goods to many different customers. The deliveries are made by dispatching a fleet of vehicles from a centralized warehouse.

The goal of this problem is to design a route for each vehicle (similar to traveling salesman tours) so that

- all of the customers are served by exactly one vehicle and
- the travel distance of the vehicles is minimized.

The vehicles have a fixed storage capacity and the customers have different demands.

Vehicle Routing Problem (VRP)

The problem is mathematically formulated as follows:

- We are given a list of locations $N = 0, ..., n-1$, where, by convention, location 0 is the location of the warehouse.
- All vehicles start and end their routes at the warehouse.
- The remaining locations are customers.
- Each location is characterized by three values (d_i, x_i, y_i) (where, $i \in N$), i.e., a demand d_i and a point (x_i, y_i).

Vehicle Routing Problem (VRP)

- The fleet of vehicles $V = 0, ..., v-1$ is fixed, and each vehicle has a limited capacity c.
- All of the demands assigned to a vehicle cannot exceed its capacity.
- For each vehicle $i \in V$, let
  - T_i be the sequence of customer deliveries made by that vehicle, and
  - $dist(m_1, m_2)$ be the Euclidean distance between two customers.
- Assume that the vehicles can travel in straight lines between each pair of locations.

Vehicle Routing Problem (VRP)

- Then, the vehicle routing problem is formalized as the following optimization problem:

minimize: $\sum_{i \in V} \left(dist(0, T_{i,0}) + \sum_{\langle j,k \rangle \in T_i} dist(j,k) + dist(T_{i,|T_i|-1}, 0) \right)$

subject to: $\sum_{j \in T_i} d_j \leq c \quad (i \in V)$

$\sum_{i \in V} (j \in T_i) = 1 \quad (j \in N \backslash 0)$

$$dist(j,k) = \sqrt{(x_j - x_k)^2 + (y_j - y_k)^2}$$

Vehicle Routing Problem (VRP)

<u>A Simple Greedy Approach for the VRP</u>

- Step 1: Start each vehicle at a different node.
- Step 2: At each step, for each vehicle choose the closest node (which has not been visited before) to current location and add it to the solution.
- Step 3: Update current location.
- Step 4: Repeat steps 2 and 3 until all nodes have been visited.
- Step 5: Return solution, i.e., a permutation of nodes for each vehicle.

Warehouse Location Problem (WLP)

- A distribution company uses warehouses to provide goods to many different customers.
- The goal of this problem is to determine which warehouses will be the most cost effective for serving the customers.
- The complexity of the problem comes from the fact that each warehouse has different costs and storage capabilities.

Warehouse Location Problem (WLP)

- There are $N = 0, \ldots, n-1$ warehouses to choose from and $M = 0, \ldots, m-1$ customers that need to be served.
- Each warehouse, $w \in N$ has a capacity cap_w and a setup cost s_w. Each customer, $c \in M$, has a demand d_c and travel cost t_{cw} based on which warehouse $w \in N$ serves it.
- Lastly, all customers must be served by exactly 1 warehouse.

Warehouse Location Problem (WLP)

- Let a_w be a set variable denoting the customers assigned to warehouse *w*.
- Then, the warehouse location problem is formalized as the following optimization problem:

minimize: $\sum_{w \in N} \left((|a_w| > 0) s_w + \sum_{c \in a_w} t_{cw} \right)$

subject to:

$$\sum_{c \in a_w} d_c \leq cap_w \quad (w \in N)$$

$$\sum_{w \in N} (c \in a_w) = 1 \quad (c \in M)$$

25

Warehouse Location Problem (WLP)

- Assume 3 warehouses and 4 customers.
- A sample solution candidate: 1 1 0 2

- This represents the assignment of customers to warehouses, a_0 = {2}, a_1 = {0, 1}, a_2 = {3}.

- That is, customers 0 and 1 are assigned to warehouse 1, customer 2 is assigned to warehouse 0, and customer 3 is assigned to warehouse 2.

26