

Recent Research Results spring 2003 - Homework

28th February 2003

This is the homework corresponding to the lectures on the 28th of february.

Task 1 - Derive integer programming models for some VRP problems

The simple VRP problem presented at the lectures can be expressed in an integer linear problem formulation (ILP). We are given a directed graph $G = (V, E)$. The nodes $V = \{1, \dots, N\}$ corresponds to customers and the depot (where the depot is node 1). The number of vehicles is given by M . Furthermore we are given an $N \times N$ matrix (c_{ij}) where c_{ij} gives the cost of edge (i, j) . The demand of node i (customer i) is given by q_i (the demand of the depot q_1 is equal to zero). Each vehicle has a capacity Q . We have to find a $N \times N \times M$ matrix $X = (x_{ijk})$, where x_{ijk} is interpreted as follows:

$$x_{ijk} = \begin{cases} 1, & \text{if vehicle } k \text{ goes directly from node } i \text{ to node } j \\ 0, & \text{otherwise} \end{cases}$$

Such that we minimize:

$$\sum_{k=1}^M \sum_{i=1}^N \sum_{j=1}^N c_{ij} x_{ijk} \quad (1)$$

subject to

$$\sum_{k=1}^M \sum_{j=1}^N x_{ijk} = 1, \quad 2 \leq i \leq N \quad (2)$$

$$\sum_{k=1}^M \sum_{j=1}^N x_{1jk} = M \quad (3)$$

$$\sum_{i=1}^N \sum_{j=1}^N q_i x_{ijk} \leq Q, \quad 1 \leq k \leq M \quad (4)$$

$$\sum_{j=1}^N x_{ijk} - \sum_{j=1}^N x_{jik} = 0, \quad 1 \leq i \leq M, 1 \leq k \leq M \quad (5)$$

$$\sum_{i,j \in S} x_{jik} \leq |S| - 1, \quad \forall S \in 2^{V \setminus \{1\}}, 1 \leq k \leq M \quad (6)$$

$$x_{ijk} \in \{0, 1\}, \quad 1 \leq i \leq N, 1 \leq j \leq N, 1 \leq k \leq M \quad (7)$$

Constraint 5 along with 2 ensures that each node is visited once and that it is the same vehicle that enters and exits a node, constraint 3 says that M vehicles leave the

depot. Constraint 4 ensures that the used capacity is below the maximum capacity on all routes. Constraint 6 eliminates subtours (notice that $2^{V \setminus \{1\}}$ is all subsets that can be formed out of the set $V \setminus \{1\}$). Constraint 7 says that all decision variables are binary.

Problem 1.a

If $x_{11k} = 1$ for some k then vehicle k is not used (its route goes directly “from the depot and back to the depot”). How can you force the model to minimize the number used vehicles as the primary objective while minimizing the travel cost as secondary objective?

Hint: Your solution should not modify the model itself, but only modify the cost matrix c_{ij} .

Problem 1.b

Extend the model (1)-(7) such that it can solve problems where vehicles have different capacities and the cost of going from i to j is depends on the vehicle.

Problem 1.c

Extend the model (1)-(7) (not the model developed in 1.b) such that it can handle problems with time windows. The time needed for traveling between node i and j is given by t_{ij} . The time window at node i is given by a_i and b_i , where a_i is the earliest arrival time and b_i is the latest departure time. A vehicle is allowed to arrive at a node i before a_i , but the vehicle must wait until time a_i before it can proceed. A vehicle is not allowed to arrive at a node after b_i . Assume that the unloading at node i can be done in zero time, that is as soon as a vehicle arrives at a node it can leave again (unless it arrives before the start of the time window). The time window at node 1 determines when the vehicle starts and when it must return to the depot. a_1 tells when the vehicles leave the depot and b_1 is the latest arrival time at the depot.

Example: Consider the following route 1 - 3 - 2 - 4 - 1. We are given the following time windows:

Node	a_i	b_i
1	0	100
2	25	35
3	20	30
4	40	50

And the travel times are given as follows:

i	j	t_{ij}
1	3	10
3	2	11
2	4	29
4	1	15

Now we can compute the arrival and departure times:

Node	arrival at node	departure at node
1		0
3	10	20
2	31	31
4	50	50
1	65	

Notice that the vehicle has to wait at node 3 and that it reaches node 4 just in time. If $t_{2,4} = 30$ then the route would be invalid.

Task 2 - Find time complexity of a simple LNS heuristic

In this task we want to find the time complexity of a simple LNS heuristic for the basic VRP problem (as presented in task 1).

Problem 2.a

Consider the algorithm shown in figure 1 in *Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems*, Shaw 1998. Derive the time complexity of this algorithm. The number of visits (customers) in the problem is given by n and the number of visits to remove is given by $k = |toRemove|$.

Problem 2.b

In problem 2.a we saw how to remove visits from the current solution. Now we are inserting these visits again. This is done by algorithm 1. This algorithm takes two parameters, S the current solution and, V the visits we should insert. Some parts of this algorithm needs a little comment. In line 8 we uses the construction “find best insertion of v into r ”. By this I mean the insertion that increases the cost of the solution S the least. If no such insertion exists I assume that some kind of dummy move is returned. In line 9 I use the function “cost(S , $currentIns$)”. This function returns the cost of performing the insertion $currentIns$ on solution S . If the insertion is a dummy insertion then ∞ is returned.

Algorithm 1 Simple Insertion

function greedyInsertion(solution S , visits V)

```
done = false;
while (not done) and ( $V \neq \emptyset$ ) do
    bestCost =  $\infty$ ;
    bestInsertion =  $\emptyset$ ;
    for each visit  $v$  in  $V$  do
        for each route  $r$  in  $S$  do
            currentIns = find best insertion of  $v$  into  $r$ ;
            if cost( $S$ , $currentIns$ ) < bestCost then

                bestCost = cost of insertion;
                bestInsertion =  $currentIns$ ;
    if bestInsertion  $\neq \emptyset$  then
        perform bestInsertion on  $S$ ;
         $V = V \setminus \{inserted\ visit\}$ ;
    else
        done = true;
end while;
return  $S$ ;
```

Find the time complexity of the function `greedyInsertion` given that the number of visits in the problem is n , the number of routes in the solution is given by $m < n$ and the number of visits to insert is given by $k = |V|$. Assume that the cost of inserting a visit can be computed in constant time and that a visit can be inserted in constant time.

Problem 2.c

Using the two algorithms presented in problem 2.a and 2.b we can construct a simple very large-scale neighborhood search heuristic for VRP. This VLSN heuristic is shown in algorithm 2. The function `RemoveVisits` is the one defined on figure 1 in the paper by Shaw (1998), it is assumed that it actually removes the visits from the solution (this is not clear from the pseudocode in Shaws paper). The parameters to the function `SimpleVLNS` is: an initial solution, an integer k that determines how many visits we must remove in each iteration, an integer D that gives the dererminism of Shaws removal algorithm and `maxIter` that controls how many iterations we should perform.

Algorithm 2 Simple VLNS

```
function SimpleVLNS(solution  $S$ , integer  $k$ , integer  $D$ , in-
integer maxIter)

    for  $i=1$  to maxIter
         $S' = S$ ;
         $V = \text{RemoveVisits}(S', k, D)$ ;
         $S' = \text{greedyInsertion}(S', V)$ ;
        if  $\text{cost}(S') < \text{cost}(S)$  then
             $S = S'$ ;
    return  $S$ ;
```

Determine the time complexity of the simple VLNS heuristic. Assume that a solution can be copied in $O(n)$ (we use this in line 3 and 7).