CHAPTER IV

METHODOLOGY

Now that several important concepts have already been explained the methodology used in this work is explained. This chapter presents both methods used to find primal and dual bounds for the CPMP.

Lagrangean Relaxation for the CPMP

As it was presented before, the formulation for the CPMP as an integer programming problem (IP) is the following:

$$Z_{IP} = Min \sum_{i \in N} \sum_{j \in N} d_{ij} x_{ij} \tag{1}$$

Subject to

$$\sum_{j \in N} x_{ij} = 1 \quad \forall i \in N \tag{2}$$

$$\sum_{j \in N} x_{jj} = P \tag{3}$$

$$\sum_{j \in N} q_i x_{ij} = Q x_{jj} \quad \forall j \in N \tag{4}$$

$$x_{ij} \in \{0,1\}; i \in N, j \in N \tag{5}$$

The Lagrangean relaxation is applied to this formulation and the set of restrictions (2) will be dualized, in practice this allows the nodes to be assigned to more than one median. This same relaxation can be seen in (Lorena & Senne, 2004), the only difference is that this work does not present a Lagrangean/surrogate method but instead a regular Lagrangean heuristic and as such, the formulation presented in here does not have a second multiplier. The formulation of the relaxed problem (LR) is the following:

$$Z_{LR} = Min \sum_{i \in N} \sum_{j \in N} (d_{ij} - \lambda_i) x_{ij} + \sum_{i \epsilon N} \lambda_i \tag{23}$$

Subject to constraints (3), (4) and (5).

Now, the structure of the problem allows it to be further simplified, if we take away the assignment constraint (3), the resulting problem is a knapsack problem, an easy problem to solve. But instead of just taking the constraint away, which will be yet another relaxation, a decomposition will be performed, this means that instead of solving (LR) directly it will be partitioned into N knapsack sub problems, one for each node as if that given node would be a median. Let $j \in N$, the objective function for the knapsack problem $(KP_j)$is the following:

$$Z_{KP_j} = Min \sum_{i \epsilon N} (d_{ij} - \lambda_i) x_{ij} \tag{24}$$

Which is subject to constraints (4) and (5). It is important to note that opposed to the knapsack problem described by (Martello & Toth, 1990) this problem is one of minimization.

The value of the(LR) problem (23) is given by selecting the p smallest knapsacks (forming the set P) and can be reformulated as:

$$Z_{LR} = \sum_{j \epsilon P} Z_{KP_j} + \sum_{i \epsilon N} \lambda_i \tag{25}$$

First Approach: Lagrangean Heuristic for the Capacitated *p*-Median Problem

The first methodology for solving the CPMP presented in this work is a Lagrangean heuristic. This method uses a subgradient method for maximizing the lower bound given by the Lagrangean relaxation and a generalized assignment problem to obtain upper bounds.

As shown in the figure 1, in order to start the algorithm the data of the amount of nodes *N*, the amount of medians *p*, the distance matrix *d*, the size of demand of each node *q* and the capacity *Q* must be read. Then initialize parameters. Step size multiplier ρ; vector of Lagrangean multipliers λ; Z_{LD} which is the best value found for the problem Z_{LR} so far and the lower bound; $UBound$ which is the best upper bound found by the GAP until now; the number T of iterations before halving the multiplier ρ; the total number of iterations K and the number ε which will be used to stop the execution in case the step size gets lower than it.

This is an iterative process, in each iteration the Lagrangean relaxation problem (LR) needs to be solved, this is done by decomposing it into N knapsack sub problems and solving them to optimality. Each of their objective values are saved. After that the p smallest knapsacks are selected and a binary vector Y that denotes what nodes are selected as medias is created along with the matrix x that contains the information of what nodes where allocated to which medians. The sum of the value of the p smallest knapsack problems plus the sum of the Lagrangean multiplier vector λ is the value of the Lagrangean relaxation Z_{LR}, as in the formulation (25). This will provide a lower bound.

Read N, p, d, q, Q.
Initialize parameters ρ, vector $\lambda, Z_{LD}, UBound, T, K$ and ε.
Repeat
  Solve the N knapsack problems
  Order the N values of Z_{KP}
  Select the p smallest
  Obtain a vector Y with the p selected medians and a matrix with the x assigned to them
  Obtain a lower bound by solving Z_{LR}
  If $Z_{LR} > Z_{LD}$ then
  $Z_{LD} \rightarrow Z_{LR}$
    GAP is solved using for the vector Y as medians
    If $Z_{GAP} < UBOund$ then
      $UBound \rightarrow Z_{GAP}$
    end-if
  end-if
  Calculate the subgradient vector using the x matrix
  If there has not been a new Z_{LD} for T iterations then
    $\rho \rightarrow \rho/2$
  end-if
  Calculate Step size
  Actualize the λ vector
Until the number of iterations has reached K or the current step size is lesser than ε.

Figure 1 The Lagrangean heuristic pseudo code.

Now the value of the Lagrangean relaxation of the current iteration Z_{LR} is compared with the best value found so far Z_{LD}. If the value is better than the current value, Z_{LR} becomes the new Z_{LD}.

If there is a new Z_{LD} an upper bound must be obtained for the original integer problem (IP) so a Generalized Assignment Problem is solved: when a CPMP has already fixed locations for the

medians, it results in a GAP. The medians used for the GAP are the ones given by the p smaller knapsack sub problems, which are saved in the vector Y. If the value of the current assignment problem Z_{GAP} is better than the best value $UBound$ obtained so far, the Z_{GAP} becomes the new $UBound$.

After that the subgradient vector for the current solution is calculated adapting the formula $(Ax^t - b)$ to the CPMP, which results in the following formulation:

$$s_i = 1 - \sum_{j \in N} x_{ij} \quad \forall i \epsilon N \tag{26}$$

Then if there has not been an improvement of Z_{LD} in T iterations the parameter ρ is divided by two. After that it is needed to calculate the step size, for that we adapt the formula (15) and get the following one:

$$Step\ Size = \rho * (Z_{GAP} - Z_{LD}) / \left|\left|s_i\right|\right|^2 \tag{27}$$

As it can be seen, that formula uses the current upper and lower bound along with the subgradient vector. Now the step size is used for another formula to update the values of the Lagrangean multipliers λ, which is based in the formula (14):

$$\lambda^{k+1} = \lambda^k + s_i * Step\ Size \tag{28}$$

This algorithm will iterate a K number of times (K is arbitrarily chosen) or until the step size is smaller than a number ε. The upper bound will be the best found by the assignment problem and the lower bound will be the best found by the subgradient method.

As noted in (Fisher, 2004) if the value of every s_i in the vector is equal to zero, then the optimum solution of the IP problem has been found.

Second Approach: Cluster Median Improvement Heuristic

In order to improve upon the Lagrangean heuristic just described the cluster median improvement heuristic or CMI is proposed to improve the quality of the upper bounds. Sometimes the process of finding good upper bounds could be slow. In the previous approach, the upper bound depends

solely on the medians given by the p smallest knapsack problems, but as the Lagrangean multipliers are usually initialized with a value of zero, the knapsacks do not yield good medians until several iterations are performed. In order to accelerate the convergence this heuristic will take each cluster and find better medians for each one of them, after that another assignment is performed, if this results in new clusters the process is repeated until no better medians are found or no improvements are made within the clusters. A cluster is a median and the nodes assigned to it. The formulation of the problem creates independent clusters; this means that no median is allowed to be assigned to any other median than itself and the heuristic functions correctly with this type of formulation.

The procedure of this heuristic is fairly simple and is performed instead of the simple GAP solving step, so here is a pseudo code for the Cluster Median Improvement heuristic (CMI). Let $MinDist$ be the smallest sum of distances from a node to the other nodes within a cluster; $NewLBound$ will be the value of the assignment problem calculated by the heuristic and $BMedian$ is the best median node within a cluster.

```
Repeat
  GAP is solved using the vector Y as medians
  Create a set of medians using the vector Y
  Initialize NewLBound
  For each p cluster
      Initialize MinDist
       For each node within the current cluster
         Calculate the sum of distances to the other nodes within the cluster
         If the sum of distances < MinDist then
           MinDist → sum of distances
           BMedian→ current node being evaluated
         end-if
      NewLBound→ NewLBound + MinDist
       If BMedian is not in the set of medians then
          BMedian is added to the set and the median of the current cluster is taken out
       end-if
  Update the vector Y with the set of medians
Until the set of medians does no change or the objective value of the GAP is the same as
NewLBound
```

Figure 2 Pseudo code for the Cluster Median Improvement heuristic.

The figure 2 depicts the process of the CMI heuristic. First a generalized assignment problem must be solved in order to get an upper bound. Then using the Y vector (which is obtained previously from the knapsack solving step) a set of the medians is created. Now the $NewLBound$ is initialized with a value of zero and it represents the value of the objective function when the new nodes are selected as medians for each cluster.

Now, the next steps will be repeated for each cluster, meaning that it will be performed p times:

$MinDist$ is initialized with a value of infinite. Now within each cluster the sum of distances of a given node to all the other nodes within the same cluster will be calculated, if this sum of distances is lower than the current value of $MinDist$, then $MinDist$ will take the value of the sum of distances and the node will be saved in $BMedian$. Then the value of $NewLBound$ will be updated by adding the value of $MinDist$ to it. This will be performed for each node within the current cluster. At the end the best median node for the cluster will be found and if it is not part of the initial set of medians it will be added to it and the previous median for the same cluster will be removed from the set.

After performing that for each cluster, we will obtain the best set of medians for the current clusters. With this set we now update the vector Y. This process is repeated until there are no more changes in the set of medians, meaning that solving the GAP did not yield different clusters and the best ones are found. A second stop condition is added, this one states that the heuristic will be exited if the objective value obtained by solving the GAP did not improve after applying the CMI heuristic. This is done in order to prevent the heuristic from exploring alternate solutions indefinitely.

The following figure 3 shows the new pseudo code indicating the position in the algorithm where the CMI is added.

Read N, p, d, q, Q.
Initialize parameters ρ, vector $\lambda, Z_{LD}, UBound, T, K$ and ε.
Repeat
  Solve the N knapsack problems
  Order the N values of Z_{KP}
  Select the p smallest
  Obtain a vector Y with the p selected medians and a matrix with the x assigned to them
  Obtain a lower bound by solving Z_{LR}
  If $Z_{LR} > Z_{LD}$ then
  $Z_{LD} \rightarrow Z_{LR}$
  Repeat
    GAP is solved using for the vector Y as medians
    Apply the CMI heuristic
  Until either one of the stop conditions are met
    If $Z_{GAP} < UBOund$ then
      $UBound \rightarrow Z_{GAP}$
    end-if
  end-if
  Calculate the subgradient vector using the x matrix
  If there has not been a new Z_{LD} for T iterations then
    $\rho \rightarrow \rho/2$
  end-if
  Calculate Step size
  Actualize the λ vector
Until the number of iterations has reached K or the current step size is lesser than ε.

Figure 3 The updated pseudo code adding the CMI heuristic.