CHAPTER III

THEORETICAL FRAMEWORK

Mathematical formulation of the problem

This chapter starts describing a model for the capacitated p-median problem, followed by a theoretical framework. The formulation used is this work is the same one that was used in (Lorena & Senne, 2004). This formulation considers that a node demand cannot be split and that the capacities are hard, as explained in (Li, 2014) this means that each vertex can only have one facility and thus these cannot be duplicated in order to augment the capacity of that vertex. Let $G = (N, E)$ be a complete graph with $N = \{1 \ldots n\}$ a set of demand points and $E = \{(i, j) | i, j \in N, i <> j\}$ a set of edges. For each edge d_{ij} denotes the distance between demand points i and j. Let q_i denote the demand of demand point i, and Q the capacity of each median. The capacitated p-median problem is to select p medians among the set of demand points, and allocate each demand point to one of these medians in order to minimize the total distance between the medians and the demand points allocated to it without surpassing the capacity of the medians. Then the CPMP is modeled as:

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

Subject to

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

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

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

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

The objective function (1) is the total sum of the distances between the nodes and the medians they are assigned to. Constraints (2) state that each node must be assigned to exactly one median. The constraint (3) enforces that there will be exactly p medians. Constraints (4) are the ones that

ensure that no median may surpass its capacity. Finally, constraints (5) indicate that variables are binary.

Relaxation of an optimization problem

In this section a relaxation of an optimization problem is defined, followed by an explanation of Lagrangean relaxation. We could regard a relaxation of a problem as another problem whose feasible region contains the feasible region of the original problem. By having a bigger feasible region, it not only becomes easier to solve but it also gets better objective values. A mathematical definition is provided in (Guignard, 2003) and it goes as follows and stands for a minimization problem, such as the one is being presented in this work:

A problem (RP_{min}): $min\{g(x)|x \in W\}$ is a relaxation of problem (P_{min}): $min\{f(x)|x \in V\}$, with the same decision variable x, if and only if:

- The feasible set of (RP_{min}) contains that of (P_{min}), this is V is a subset of W
- $X \in V$, where X is the set of feasible solutions to RP_{min}
- The objective function of (RP_{min}) is better than that of (P_{min}), this is $g(x) \leq f(x)$

As mentioned before relaxations are used to provide dual bounds for the optimal value of difficult problems, these bounds are frequently used as starting points for many algorithms. While the solutions to the relaxed problem are most often infeasible to the original problem, they can be used to generate solutions to the original problem. Another advantage is that they can be much easier to solve and thus having to use reduced computational effort when solving them.

Relaxations are obtained when a constraint is ignored, dropped or transformed into an easier constraint. An integer programming problem, such as the CPMP, is one whose variables are integer; a common relaxation among this type of problems is the linear or continuous relaxation which ignores the integrality restriction and then the variables are allowed to be real numbers. (Guignard, 2003)

In this work a Lagrangean relaxation is used. In these relaxations, some of the constraints are dualized and added to the cost function. Many hard-to-solve problems could be seen as easy-to-solve problems that contain constraints that make them hard. Dualizing the complicating constraints and adding them to the cost function produces a relaxed problem, a Lagrangean problem, which is easier to solve and provides dual bounds for the original problem. (Fisher, 2004)

Let (P) be an integer programming problem which value is denoted with Z. We desire to minimize said value, also the set of constraints have been partitioned into two sets. Also let c be a cost vector with dimension of $1 \times n$ and x a vector with dimension of $n \times 1$. A is a coefficient matrix of $m \times n$ and b is a vector of $m \times 1$. D is another coefficient matrix of $k \times n$, and finally e is a vector of $k \times 1$. The problem (P) is then stated as follows:

$$Z = min\ cx \tag{6}$$

$Subject\ to$

$$Ax = b \tag{7}$$

$$Dx \leq e \tag{8}$$

$$x \geq 0\ and\ integral \tag{9}$$

For this formulation the set of constraints $Ax = b$ are considered to be the complicating constraints, while $Dx \leq e$ are those considered being easy to solve. So, in the Lagrangean problem the complicating constraints are dualized, this is, they are added to the objective function with multipliers (weights u), these multipliers are called Lagrange multipliers. The resulting formulation is a Lagrangean Relaxation (LR_u) of the original problem:

$$Z_D(u) = min\ cx + u(Ax - b) \tag{10}$$

$Subject\ to$

$$Dx \leq e \tag{11}$$

$$x \geq 0\ and\ integral \tag{12}$$

$u = (u_1, \dots, u_m)$ is the vector of Lagrange multipliers, the logic behind these is that if the multipliers are big enough, the objective function will not violate the complicating constraints. (Fisher, 2004) Something worth noting is the fact that a bound obtained by a Lagrangean relaxation is always equal or better than the bound obtained by the linear relaxation, never worse. (Guignard, 2003)

The subgradient method

By what has been stated it becomes apparent that finding better multipliers for the (LR_u) problem will yield better lower bounds to the (P) problem, it is even possible to find its optimum if the solution is feasible and also satisfies the complementary slack conditions which means that $u[Ax(u) - b] = 0$. (Fisher, 2004)

It is only natural then to try to find the correct set of multipliers. This can be achieved by maximizing the objective value of the (LR_u) problem; this problem is referred as a Lagrangean Dual (D) and is stated with the following formulation:

$$Z_D = Max_u Z_D(u) \tag{13}$$

The problem (D) is in the space of u as opposed to (LR_u) which is in the space of x.

To find the optimum for such problem many approaches have been developed: the subgradient optimization, the constraint generation method, the two-phase hybrid method, the primal relaxation method, etc. (Guignard, 2003). The subgradient optimization approach is the most widely studied and is relatively easy to implement. One of the characteristics of the $Z_D(u)$ function is that it is not differentiable at optimal points; this means that a gradient cannot be found in these points, but subgradients could be found at any point of the function. It is assumed that there is a set $X = \{x | D_x \leq e, x \geq 0 \; and \; integral\}$ and it is a finite set of feasible solutions for the problem (LR_u). By being finite it can be represented as follows $X = \{x^t, t = 1, \dots, T\}$ (Fisher, 2004).

The vector $(Ax^t - b)$ will be a subgradient at any u for which (LR_u) is solved by x^t. The subgradient method is iterative and let k be the current iteration. It starts with a value $u^0$and then a sequence $\{u^k\}$ is then generated with the following formula:

$$u^{k+1} = u^k + t_k(Ax^k - b) \tag{14}$$

Where x^k is an optimal solution to the problem (LR_{u^k}) and $t_k$is a positive scalar which is called step size. The logic is that a step has to be taken so the multiplier vector will be getting closer to the optimal value at each iteration. There are various ways to calculate the step size, but the most common one is the following:

$$t_k = \frac{\rho_k(Z^* - Z_D(u^k))}{||Ax^k - b||^2} \tag{15}$$

.

The value of the scalar ρ is usually $0<\rho \leq 2$, and $\rho_0 = 2$ it is then is halved when the value of $Z_D(u)$ has not improved in a fixed number of iterations. Unless the $Z_D(u^k)$ has a value of a known feasible solution, the only other way other way of proving that u^* is optimal for problem (D) is if the subgradient of $Z_D(u)$ at u^* is 0, so the subgradient method stops after a certain number of iterations. The number of iterations is chosen in an arbitrary way. (Fisher, 2004)

There are several considerations to this; firstly a subgradient will not necessarily be an improvement from the last level as it can go in several directions. Also the formula uses Z^*which is the optimal value of Z_D and it is currently unknown. (Guignard, 2003) One approach to generate that value is to solve the (P) problem with an heuristic, this will result in an upper bound of the problem that can be used as the value of Z^* (Fisher, 2004)

Decomposition

The decomposition principle is a systematic procedure for solving large scale linear programs or linear programs that contain constraints of special structure. In simple words decomposition is to partition a problem (called master problem) into smaller and easier to solve problems (called sub problems). The base of decomposition procedure is to partition the original problem on two separate problems. The Lagrangean Relaxation, along with the Dantzing-Wolfe decomposition and the Bender's Reformulation, make use of the principle of decomposition. (Bazaraa, Jarvis, & Sherali, 2010) Many problems that involve capacity constraints can be decomposed into smaller knapsack problems.

The Knapsack Problem

The knapsack problem is to select a subset of set of objects such that it maximizes the total value of the objects contained in the knapsack. The knapsack has a limited capacity and each object has a weight and an associated value. Let J be the set of objects, p_j and $w_j$the value and weight of object j respectively and c the knapsack capacity. All these parameters are considered to be positive integers. Let x_j be the decision variable that takes the value of 1 if the *j* th item is selected and 0 otherwise. The formulation is the following:

$$Max \sum_{j \epsilon J} p_j x_j \tag{16}$$

$Subject\ to$

$$\sum_{j \epsilon J} w_j x_j \leq c \tag{17}$$

$$x_j \epsilon \{0,1\}, j \epsilon J \quad (18)$$

The objective function (16) states the sum of the value of the selected objects; the restriction (17) ensures that the capacity of the knapsack is not surpassed; finally, restrictions (18) makes the variable to be binary. The 0-1 knapsack problem is NP-hard, but among different NP-hard problems this one is quite easy to solve. (Martello & Toth, 1990).

The Generalized Assignment Problem

This problem, also referred as GAP, consists of assigning a set of jobs $(j \in J)$ to machines $(i \in I)$ with the smallest total cost. Each job must be assigned to just one machine. Assigning a job to a machine has a related cost c_{ij}. Let a_{ij} be the amount of resources consumed by machine i to perform the job *j;* then $b_j$is the resource capacity of the machine *I* and x_{ij} is the decision variable which will be 1 if the job j is assigned to the machine i, and 0 otherwise.

The model described in (Posta, Ferland, & Michelon, 2012) is the following:

$$\min \sum_{i \epsilon I} \sum_{j \epsilon J} c_{ij} x_{ij} \quad (19)$$

$Subject\ to$

$$\sum_{j \epsilon J} a_{ij} x_{ij} \leq b_i \quad \forall i \epsilon I \quad (20)$$

$$\sum_{i \epsilon I} x_{ij} = 1 \quad \forall j \epsilon J \quad (21)$$

$$x_{ij} \epsilon \{0,1\} \forall i \in I, j \in J \quad (22)$$

Where:

The objective function (19) is the total cost, which is desired to be minimized; the first set of constraints (20) ensures that the capacity of each machine is not surpassed; the following set of constraints (21) enforces that each job is assigned to exactly just one machine; the last set of constraints (22) makes the variable binary.