Fixed Charge Network Flow Problem

By Temesgen Tsegaye

ADDIS ABABA UNIVERSITY

NATURAL SCIENCE COLLAGE

DEPARTEMENT OF MATHEMATICS

FACULTY OF COMPUTER AND MATHEMATICAL SCIENCES

"In partial fulfilment of the requirement of the degree of master of science in mathematics"

Stream: Optimization

Advisor: Berhanu Guta(PhD)

Addis Ababa University, Ethiopia

June, 2011

Declaration

"I declare that this project has been composed by me and that no part of the project has formed the basis for the award of any Degree, Diploma, Associate ship, Fellowship or any other similar title to me."

Author's signature

Permission

"This is to certify that this project is compiled by Temesgen Tsegaye in the department of Mathematics, Addis Ababa University, under my supervision. I hereby also confirm that the project can be submitted for evaluation by examiners and eventual defence."

____

Advisor's signature

Acknowledgment

Above all, my best gratitude goes to GOD, who helped me from the beginning to the end. Moreover, I will extend my heartfelt appreciation to my advisor Dr.Berhanu Guta, who helped me by providing his valuable guidance for the successful accomplishment of this project.

Summary of the project

Fixed Charge Network Flow Problem (FCNFP) is a well known NP-hard problem. It has a wide spectrum of applications. Among them are problems in network design, scheduling, production planning, supply chain and transportation science. Many exact methods and heuristic approach have been developed to solve the FCNFP. In this project, I focused on some exact method such as Branch-and-cut method to solve single commodity uncapacitated fixed charge network flow problem, and exact algorithm for fixed charge transportation problem. In addition, there is also one greedy (heuristic) algorithm to solve the uncapacitated facility location problem which is a special case of fixed charge network flow problem.

List of figures

List of tables

Contents

1. Introduction

Many practical planning, transportation and distribution problems such as network design, plant location, communication, production scheduling, and investment and distribution decisions can be modelled as Fixed Charge Network Flow Problems (FCNFP). FCNFP is to choose which arcs should be present so as to minimize total fixed and variable cost while observing flow constraints. Since this is a special subclass of the Minimum Concave cost Network Flow Problem (MCNFP), FCNFP also has the same characteristics as MCNFP [10] does. Moreover, the objective function in FCNFP is discontinuous at the origin.

The only difference between the formulation of the fixed-charge problem and easy linear programming problems is the discontinuous cost function. This deceptively simple detail is sufficient to distinguish the fixed-charge problem as NP-hard [9]. Solving the problem to optimality therefore becomes exponentially more difficult as the problem size increases. The fixed-charge problem has been studied extensively and is of significant practical use. It includes the single and multiple source fixed-charge transportation problem, capacitated and uncapacitated lot-sizing, warehouse location problem, and Steiner tree problem as variants.

The single source uncapacitated case with a fixed number of fixed arc cost is strongly polynomial solvable [14]. Exact solution methods for the fixed charge network flow problem often utilize a binary mixed integer programming formulation together with Branch & Bound [1,2 and 4], Benders decomposition [3],or exploit the fact that the optimum in a concave minimization problem is obtained in a vertex [8]. There are a huge variety of heuristic methods [4 and 7] which are especially favourable for large-scale problems due to the NP-hardness.

The outline of the paper is as follows. Section two contains some basic preliminary concept. In section three we describe the general description of fixed charge problems and section four describe the different special case of the problem. Section five explains about branch and cut method to solve the single commodity uncapacitated fixed charge network flow problems. Section 6 deals the solution method and the last section contains some concluding remarks.

2. Preliminaries

2.1. Linear programming

Linear programming is a very important class of problems, both algorithmically and combinatorially. A linear program is the problem of optimizing a linear objective function in the decision variables, $x_1, x_2, \dots, x_n$, subject to linear equality or inequality constraints on the x_i's. In standard form, it is expressed as:

$$\min \sum_{j=1}^{n} c_j x_j$$

s.t:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \text{ for } i = 1,2,\dots,m \qquad (1)$$

$$x_j \geq 0, j = 1,2,\dots,n$$

where $\{a_{ij}, b_i, c_j\}$ are given.

A linear program is expressed more conveniently using matrices:

$$\min \ c^T x \text{ subject to } \begin{cases} Ax = b \\ x \geq 0 \end{cases}$$

where $x = (x_1, \dots, x_n)^T \in R^{n\times 1}$, $b = (b_1, \dots, b_m)^T \in R^{m\times 1}, c = (c_1, \dots, c_n)^T \in R^{n\times 1}$ and A is $m \times n$ matrix.

2.2. Mixed integer linear program (MILP)

A mixed integer linear programming (MILP) ("mixed" due to the fact that some of the variables are restricted to take only integer values) problem is an optimization problem with a linear objective function and linear constraints

It is one type of integer programming problems in which:

- Some, but not all decision variables must have integer solution
- Non-integer variable can have fractional optimal values

Definition 2.1: A mixed integer linear program (MILP) is an optimization program involving continuous and integer variables, and linear constraints. Any MILP can be written as

$$\text{(MILP)} \qquad \min\{cx + fy : (x, y) \in X\}$$

where the set X is called the set of feasible solutions and is described by m linear constraints, nonnegativity constraints on the x, y variables, and integrality restrictions on the y variables. In matrix notation

$$X = \{(x, y) \in R_+^n \times Z_+^p : Ax + By \geq b\},$$

Where

- x and y denote, respectively, the n -dimensional (column) vector of nonnegative continuous variables and the p -dimensional (column) vector of nonnegative integer variables.
- $c \in R^n$ and $f \in R^p$ are the (row) vectors of objective coefficients.
- $b \in R^m$ is the (column) vector of right hand side coefficients of the m constraints
- A and B are the matrices of constraints with real coefficients of dimensions $(m \times n)$ and $(m \times p)$ respectively.

Definition 2.2: A valid inequality for the feasible set X of MILP,

$$X = \{(x, y) \in R_+^n \times Z_+^p : Ax + By \geq b\},$$

is a constraint or inequality $\alpha x + \beta y \geq r$ (with $\alpha \in R^n, \beta \in R^p$ and $r \in R$) satisfied by all points in X; that is,

$$\alpha x^* + \beta y^* \geq r \text{ for all } (x^*, y^*) \in X$$

Thus a valid inequality for a MILP is an inequality that is satisfied by all feasible solutions.

A basic rounding argument:

If $x \in Z$ and $x \leq f, f \notin Z$, then $x \leq \lfloor f \rfloor$.

Consider an inequality $\alpha x \leq \beta$ such that $\alpha_j \in Z, j = 1,2, \ldots, n$. If $\alpha x \leq \beta$ then $\alpha x \leq \lfloor \beta \rfloor$ is valid as well.

Example 2.1: Consider the inequality $x_1 + x_2 \leq 1.9$

$$x \in Z^2 \text{ such that } x_1 + x_2 \leq 1.9$$

$$\Longrightarrow x_1 + x_2 \leq \lfloor 1.9 \rfloor = 1$$

$$\text{Thus } x_1 + x_2 \leq 1 \text{ is valid}$$

Theorem 2.1: If $x \in Z^n$ satisfies $Ax \leq b$, then the inequality $uAx \leq \lfloor ub \rfloor$ is valid for S for all $u \geq 0$ such that $uA \in Z^m$.

Example 2.2: Consider the polyhedron given by the inequalities

$$x_1 + x_2 \leq 2$$

$$3x_1 + x_2 \leq 5$$

Let $u_1 = u_2 = 0.5$. Then to find valid inequality let us add the two inequalities, we get

$$4x_1 + 2x_2 \leq 7$$

$$\frac{1}{2}(4x_1 + 2x_2 \leq 7) \Longrightarrow 2x_1 + x_2 \leq 3.5$$

and then rounding we obtain $2x_1 + x_2 \leq 3$. Thus $2x_1 + x_2 \leq 3$ is valid inequality.

2.2.1. A simple disjunctive argument

The theory of disjunctive inequalities gives a methodology for generating general cutting planes that can be powerful on certain hard integer programs.

If $x \in R^n, x \geq 0$ and x satisfies both $\sum_{i=1}^{n} a_i^1 x_i \geq 1$ or $\sum_{i=1}^{n} a_i^2 x_i \geq 1$, then x satisfies

$$\sum_{i=1}^{n} \max\{a_i^1, a_i^2\} x_i \geq 1$$

Example 2.3:

If $x \geq 0$ satisfies both $\frac{x_1}{2} + \frac{x_2}{2} \geq 1$ and $\frac{3x_1}{5} + \frac{x_2}{5} \geq 1$.

$\max\{a_1^1, a_1^2\} = \max\{\frac{1}{2}, \frac{3}{5}\} = \frac{3}{5}$

$\max\{a_2^1, a_2^2\} = \max\{\frac{1}{2}, \frac{1}{5}\} = \frac{1}{2}$ .Thus $\frac{3x_1}{5} + \frac{x_2}{2} \geq 1$ is valid.

2.2.2. Mixed-integer-rounding cuts (MIR)

Consider the 2-Variable mixed integer set $\{(x, y) \in Z \times R_+ : x + y \geq b\}$ with $b \in R$

The inequality $\frac{x}{b - \lfloor b \rfloor} + y \geq \lceil b \rceil$ is valid and together with the original inequality defines the convex hull of the mixed integer set.

Mixed integer rounding (MIR) procedure:

Step 1: The inequalities

$$\sum_{j\in N}(ua_j)x_j + \sum_{j\in J}(ug_j)y_j \le ub \text{ are valid for all } u \in R_+^m$$

Step 2: given two valid inequalities

$$\sum_{j\in N}\pi_j^i x_j + \sum_{j\in J}\mu_j^i y_j \le \pi_0^i \text{ for } i = 1,2$$

Construct the third valid inequality

$$\sum_{j\in N}\lfloor\pi_j^2 - \pi_j^1\rfloor x_j + \frac{1}{1-f_0}\left(\sum_{j\in N}\pi_j^1 x_j + \sum_{j\in J}\min(\mu_j^1, \mu_j^2)y_j - \pi_0^1\right) \le \lfloor\pi_0^2 - \pi_0^1\rfloor$$

Where $\pi_0^2 - \pi_0^1 = \lfloor\pi_0^2 - \pi_0^1\rfloor + f_0$

2.3. NP completeness

Definition 2.3: Decision Problem: A problem to which there is a yes or no answer.

Optimization Problem: A problem to which the answer is an optimal feasible solution. An optimization problem can be solved by calling a decision problem in the form “Is there a solution with value less than or equal to k?" (for minimization) a polynomial number of times using binary search. A very important class of decision problems is called NP, which stands for nondeterministic polynomial time. It is an abstract class, not specifically tied to optimization

Definition 2.4: A decision problem is said to be in NP if for all “Yes" instances of it there exists a polynomial length “certificate" that can be used to verify in polynomial time that the answer is indeed Yes.

Definition 2.5. The class of problems for which the certificate exists for all “No” instances is called co-NP.

Definition 2.6. A problem P_1 is said to reduce in polynomial time to problem P_2 if there exists a polynomial-time algorithm A_1 for P_1 that makes calls to a subroutine solving P_2 and each call to a subroutine solving is P_2 counted as a single operation.

Definition 2.7: A decision problem P is said to be NP-complete if:

1. $P \in$ NP
2. All other problems in NP are polynomially reducible to P.

It follows that if any NP-complete problem were to have a polynomial algorithm, they all would. When a decision problem is NP-Complete, we call the optimization problem NP-hard problem.

2.4. Basic graph definitions

- A directed graph or digraph D is an ordered pair $D = (V, \mathcal{A})$, where V is a set whose elements are called vertices or nodes, and $\mathcal{A}$ is a set of ordered pairs of vertices of the form (i, j) called arcs. In an arc (i, j) node i is called the tail of the arc and node j the head of the arc.
- A path (directed path) is an ordered list of vertices$(v_{1,} \dots, v_k)$, so that $(v_i, v_{i+1}) \in \mathcal{A}$ for all $i = 1,2, \dots, k$.The length of the path is $|(v_{1,} \dots, v_k)| = k$.
- A cycle (directed cycle) is an ordered list of vertices $v_0, v_1, \dots, v_k$, so that $(v_i, v_{i+1}) \in \mathcal{A}$ for all $i = 1,2, \dots, n$ and $v_0 = v_k$.The length of a cycle is$|(v_{1,} \dots, v_k)| = k$.
- A digraph (network) is said to be connected if, for every pair of nodes, there is a path starting at one node and ending at the other node. The degree of a vertex is the number of edges incident to the vertex.
- A sub network of a network is a subset of the nodes and arcs of the original network. The arcs in the sub network must connect nodes in the sub network and must not involve nodes that are not in the sub network.
- A tree is a connected sub network containing no cycles. A spanning tree is a tree that includes every node in the network.

2.4.1. Node Selection and Branching Rules

To implement a branch-and-bound algorithm, we need to specify a node selection rule and a branching rule. The node selection rule specifies which linear set from the list L to select at each iteration. This rule has an impact on the order in which the nodes are treated, and therefore on the evolution of the lower and upper bounds during the execution of the algorithm. Some standard rules are:

• Depth-first search in which one selects a child of the preceding node after branching, and backtracks (i.e., moves back in the enumeration tree to select the next node) as few nodes up as possible after pruning a node;

• Breadth-first search in which one selects nodes those are closest to the top of the tree. We use this search in branch and cut algorithm.

• Best-bound search in which one selects the node with the best (lowest) lower bound;

• Combinations of these, such as

  - Breadth-first search for a certain number of nodes followed by depth-first search, or
  - Depth-first search as long as branching is performed, and then best bound after pruning a node.

The choice of the rule is often heuristic. Depth-first search allows one to obtain solutions quickly (because there are more branching constraints deep in the tree, and such bound constraints help to obtain integer solutions), but with the risk that the solution quality suffers if wrong branching decisions are taken. On the other hand breadth-first search avoids this risk by working in parallel on all branches of the tree, but is rather slow to obtain feasible solutions. Best-bound search minimizes the number of nodes treated to prove optimality, and the combinations of rules are attempts to find a good compromise.

3. Problem description

Fixed charge problems were first proposed by Hirsch and Dantzig in 1954. They proved that the optimal solution occurs at one of the extreme points defined by the constraints of the problem.

Consider a production planning involving n products, the cost for product j consists of a variable cost c_{ij} per unit and a set up cost (or fixed charge) $f_{ij} > 0$ independent of the amount produced.

Let x_{ij} be the number of units produced of product j. Then the production function can be written as:

$$c_{ij}\left(x_{ij}\right) = \begin{cases} c_{ij}\,x_{ij} + f_{ij} & \text{if } x_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

To minimize the objective function with non negative fixed charges we introduce new binary variables y_{ij}

where $y_{ij} = \begin{cases} 1 & \text{if } x_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$

This condition can be expressed in the form of a single linear constraint as

$x_{ij} \leq U y_{ij}$, where $U > 0$ is a sufficiently large number (U can be given or in any derived upper bound on the variable of x_{ij} in any feasible solution). Then the objective function in the original fixed charge problems becomes

$$\min \sum_i \sum_j (c_{ij}\,x_{ij} + f_{ij}\,y_{ij})$$

$$s.t \qquad (2)$$

$$0 \leq x_{ij} \leq U y_{ij}\ ,\forall i,j$$

$$y_{ij} \in \{0,1\}, \forall i,j$$

Thus a fixed charge problem is a product selection problem where there are an additional fixed charge to be paid for each product produced.

Fixed charge network flow problems can be defined as on a digraph $D = (V, \mathcal{A})$,a set of vertex V,and arc set $\mathcal{A}$. Let c_{ij} be the unit flow cost and f_{ij} the fixed cost, on arc $(i,j) \in \mathcal{A}$. Let U be the maximum flow capacity of arc (i,j) and b_i the net supply at node $i \in V$. An arc can be present (allowed to carry positive flow) only if it is fixed cost is incurred. The problem is to choose which arcs should be present so as to minimize the total fixed and variable cost while observing flows constraints.

To model the FCNFP we consider the following assumptions:

1. $\sum_{i\in V} b_i = 0$. i.e. the total supply equals the total demand within network
2. $f_{ij} \geq 0$ for all $(i,j) \in \mathcal{A}$ since if $f_{ij} < 0$,we can set $y_{ij} = 1$ and y_{ij} eliminate from the problem.
3. We assume that with respect to the c_{ij} there are no negative cost directed cycles. Thus the objective function is bounded from below.

The fixed charge network flow problem can be formulated as $0-1$ model as follows.

$$\min \sum_{(i,j)\in\mathcal{A}} c_{ij}\, x_{ij} + \sum_{(i,j)\in\mathcal{A}} f_{ij}\, y_{ij} \qquad (3)$$

s.t

$$\sum_{j\in V_i^-} x_{ji} - \sum_{j\in V_i^+} x_{ij} = b_i, \forall i \in V \qquad (4)$$

$$0 \leq x_{ij} \leq U y_{ij}, \ \forall (i,j) \in \mathcal{A} \qquad (5)$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in \mathcal{A} \qquad (6)$$

where $D = (V, \mathcal{A})$ is a digraph , $V_i^+ = \{j \in V: (i,j) \in \mathcal{A}\}$ and $V_i^- = \{j \in V: (j,i) \in \mathcal{A}\}$, b_i is the supply at node i and U is a large positive integer, c_{ij} is the variable cost per unit flow on arc (i,j), f_{ij} is the fixed cost having flow on arc (i,j).

4. Types of Fixed Charge Network Flow Problem

The fixed charge network flow problem can be capacitated or uncapacitated depending on the capacity of each arc $(i, j) \in \mathcal{A}$. That is if the arc capacity is ∞ (unlimited) we call the problem uncapaciteated fixed charge network flow problem; otherwise we say that it is capacitated fixed charge network flow problem.

Besides being an important model in its own right for a variety of network design problems, several special cases of FCNFP are of substantial interest. There are different ways to find the special cases.

We can find special cases using:

a. we can restrict the network structure (e.g. Transportation problem)
b. Capacity constraints - when U is sufficiently large for instance $U \geq \sum_{i \in V} |b_i|$-there is no feasible flow with $x_{ij} > U$.The capacity constraints only serves to force the fixed cost to be included in the objective function when the flow is positive. We call such problem Uncapacitated problems.
c. When $|i \in V : b_i > 0| = 1$ we call this problem single source problem.

There are many problems which arise as a special case of fixed charge network flow problem some of them are:

- Fixed charge transportation problem
- Uncapacitated facility location problem
- Steiner tree problem
- Uncapacitated lot sizing problem, etc…

But in this paper we consider the first two problems to find their solution.

4.1. The uncapacitated facility location problem (UFL)

The facility location problem is a special case of the fixed charge network flow problem. One particularity is that the fixed costs are incurred on opening nodes (locations) rather than arcs. The problem is select a subset of locations from a given candidate set, and place in each of these locations a “facility” that will serve the needs of certain “clients.” There is a 0-1 decision variable associated with selecting any given location for facility placement, at a given cost. Once these variables are chosen, an assignment (or transportation) problem must be solved to optimally match clients with facilities. Facility location problem can be capacitated or uncapacitated but here we consider the uncapacitated facility location problem. Given a set of potential sites, a set of clients, and relevant profit and cost data, the goal is to find a maximum profit plan giving the number of facilities to open, their locations and an allocation of each client to an open facility.

In detail, the input to an uncapacitated facility location problem (UFL) problem consists of

- A set $J = \{1,2, \dots, n\}$ of potential sites for locating facilities,
- A set $I = \{1,2, \dots, m\}$ of clients whose demands need to be served by the facilities,
- A profit c_{ij} for each $i \in I$ and j∈J; this is the profit made by satisfying the demand of client i from a facility located at site j, and
- A fixed non negative cost f_j for each j $\in J$; this is the (one time) cost of opening a facility at site j .

The problem is to select a subset $Q (Q \subseteq J)$ of sites, to open facilities at these sites, and to assign each client to exactly one facility such that the difference of the variable profits and the fixed costs is maximized. The number of facilities to be opened,$|Q|$, is not presented, rather it is determined by an optimal solution. The profits c_{ij} usually depend on several factors such that as the per unit production cost of a facility at site, the per unit transportation cost j to i, and the selling price to client i.

It is an NP-hard problem and is used to model many applications. Some of these applications are: bank account allocation, clustering analysis, lock-box location, location of offshore drilling platforms, economic lot sizing, machine scheduling and inventory management, portfolio management and the design of communication networks.

Let us see the formulation of UFL to use greedy (heuristic) algorithm to solve the problem. We can formulate the problem as maximization problem. For each potential site $j \in J$ we have a zero one variable x_j. The intension is that a facility is opened at site $j \in J$, iff $x_j = 1$. For each client $i \in I$ and site $j \in J$, we have a zero one variable y_{ij} .The intension is that the demand of client i is served by the facility at site j iff $y_{ij} = 1$.Thus the linear programming formulation of the UFL problem:

$$\text{maximize } \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} - \sum_{j \in J} f_j x_j$$

$$s.t \tag{7}$$

$$\sum_{j \in J} y_{ij} = 1 \qquad , \forall i \in I$$

$$y_{ij} \leq x_j \qquad , \forall i \in I, \forall j \in J$$

$$x_j \in \{0,1\} \qquad , \forall j \in J$$

$$y_{ij} \in \{0,1\} \qquad , \forall i \in I, \forall j \in J$$

The strong LP relaxation (SLPR) is obtained by replacing the integrality restrictions on the $x_j{}'$ s by the linear constraints, i.e. the constraint $x_j \in \{0,1\}, \forall j \in J$ are replaced by the constraints $0 \leq x_j \leq 1, \forall j \in J$.

$$\text{(SLPR)} \qquad \text{maximize } \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} - \sum_{j \in J} f_j x_j$$

$$s.t \tag{8}$$

$$\sum_{j \in J} y_{ij} = 1 \qquad , \forall i \in I$$

$$y_{ij} \leq x_j \qquad , \forall i \in I, \forall j \in J$$

$$x_j \leq 1 \qquad , \forall j \in J$$

$$x_j \geq 0 \qquad , \forall j \in J$$

$$y_{ij} \geq 0 \qquad , \forall i \in I, \forall j \in J$$

Now let us see how to write the duality of the strong LP relaxation and two condensed form for the dual of SLPR.

a. Duality

To write the dual of the strong LP relaxation we introduce dual variables, $u_i, i \in I, w_{ij}, i \in I, j \in J$, and $t_j, j \in J$, corresponding to the (SLPR) constraints $\sum_{j \in J} y_{ij} = 1$ $(\forall i \in I)$,

$y_{ij} - x_j \le 0$ $(\forall i \in I, \forall j \in J)$, and $x_j \le 1$ $(\forall j \in J)$, respectively. The dual of LP is as follows

$$\text{(Dual SLPR)} \qquad \text{minimize} \sum_{i \in I} u_i + \sum_{j \in J} t_j$$

$$s.t \tag{9}$$

$$t_j - \sum_{i \in I} w_{ij} \ge -f_j \quad , \forall j \in J$$

$$u_i + w_{ij} \ge c_{ij} \quad , \forall i \in I, \forall j \in J$$

$$u_i \quad \text{free} \quad , \forall i \in I$$

$$w_{ij} \ge 0 \quad , \forall i \in I, \forall J \in J$$

$$t_j \ge 0 \quad , \forall j \in J$$

b. First condensed dual (CD1)

Suppose that all the variables u_i in (Dual SLPR) have fixed values. Then to minimize the objective function, we must assign each w_{ij} the minimum value such that the constraints $u_i + w_{ij} \ge c_{ij}$ $(\forall i \in I, \forall j \in J)$ and $w_{ij} \ge 0 (\forall i \in I, \forall j \in J)$ are satisfied.

This gives

$$w_{ij} = \left(c_{ij} - u_i\right)^+, \forall i \in I, \forall j \in J$$

where for an expression α,$(\alpha)^+$means $\max(\alpha, 0)$.

Now consider the variables t_j, $(\forall j \in J)$. To minimize the objective function, we must assign each t_j the minimum value such that the constraints $t_j - \sum_{i \in I} w_{ij} \ge -f_j$ $(\forall j \in J)$and $t_j \ge 0$ are satisfied.

Let $t_j = \left(\sum_{i \in I} w_{ij} - f_j\right)^+, \forall j \in J$. Now substitute $w_{ij} = \left(c_{ij} - u_i\right)^+$, we get

$$t_j = \left(\sum_{i \in I} \left(c_{ij} - u_i\right)^+ - f_j\right)^+ , \forall j \in J$$

Thus we get the first condensed dual

$$(CD1) \qquad \underset{u_1,\ldots,u_m}{\text{minimize}} \left\{ \sum_{i\in I} u_i + \sum_{j\in J} \left(\sum_{i\in I} (c_{ij} - u_i)^+ - f_j \right)^+ \right\}$$

c. Second condensed dual

In feasible solution of the dual LP (Dual SLPR), suppose that there is a $k \in J$ such that t_k is positive, that is, $(\sum_{i\in I}(c_{ik} - u_i)^+ - f_k)^+$ is positive. Then there exists a $u_l (l \in I)$, such that $c_{lk} - u_l > 0$. If we increase u_l by amount $\epsilon(\epsilon > 0)$, then t_k will decrease by ϵ, therefore the objective value stays the same; also, all dual constraints will continue to hold. It follows that there always exists an optimal solution to the dual LP with $t_j \leq 0, \forall j \in J$.

To see this, use the above procedure repeatedly until each t_j is at most zero. We also add the constraints $u_i \leq \max(c_{ij}) \; \forall i \in I, \forall j \in J$.This gives second condensed dual

$$(CD2) \text{ minimize } w = \sum_{i\in I} u_i$$

$$s.t$$

$$\sum_{i\in I} (c_{ij} - u_i)^+ - f_j \leq 0 \quad , \forall j \in J$$

$$u_i \leq \max(c_{ij}) \; \forall i \in I, \forall j \in J$$

4.2. The fixed charge transportation problem

The fixed-charge transportation problem may be simply stated in terms of a distribution problem in which there are suppliers (warehouses or factories) and n customers (destinations). Each of the m suppliers can ship to any customer at a per unit shipping cost of c_{ij} (unit cost for shipping from supplier i to customer j) plus a fixed cost of f_{ij} assessed for opening this route. Each supplier, $i = 1,2,\ldots,m$ has S_i units of supply and each customer, $j = 1,2,\ldots,n$, demands D_j units. The objective is to determine which routes to open and shipment size, so that the total cost of meeting demand given the supply constraints is minimized.

Fixed charge network flow problem

This problem differs from the standard Hitchcock transportation problem in that a fixed charge (e.g., for route establishment or a franchise) is associated with each route over which goods are shipped. Since the objective is to minimize total cost, it is possible to trade off variable cost (i.e., transportation cost) and fixed cost.

In general the hierarchy of fixed charge problems is given in figure 1

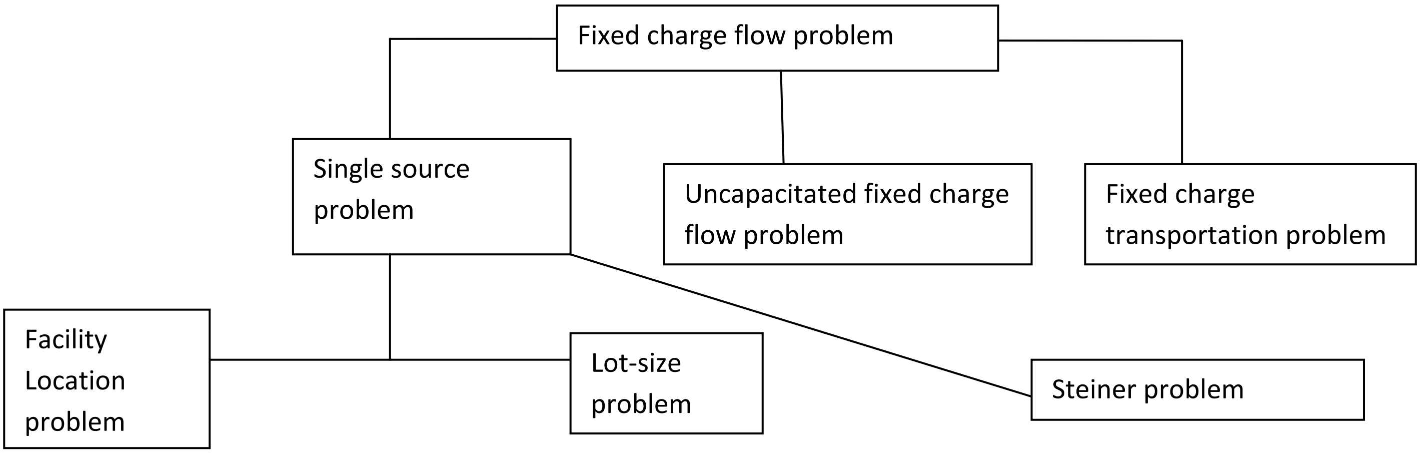

Fig.1

5. Single commodity uncapacitated fixed charge network flow problem

A network flow problem in which it is not necessary to distinguish among the units flowing in the networks is called single commodity or type of units. The formulation for single commodity uncapacitated fixed charge network flow problem (UFCNFP) is given in (3)-(6). The problem is NP-hard, as it generalizes the Steiner tree problem [9]. Let us repeat the formulation of FCNFP here to observe other characteristics and additional notations.

In order to describe UFCNFP as a mixed integer program, define x_{ij} to be the flow on arc (i,j), and $y_{ij} = 1$ if the arc (i,j) is used ($x_{ij} > 0$),and $y_{ij} = 0$ otherwise. A resulting formulation is

$$\min \sum_{(i,j)\in\mathcal{A}} c_{ij}\, x_{ij} + \sum_{(i,j)\in\mathcal{A}} f_{ij}\, y_{ij} \tag{10}$$

s.t

$$\sum_{j\in V_i^-} x_{ji} - \sum_{j\in V_i^+} x_{ij} = b_i, \forall i \in V \tag{11}$$

$$x_{ij} \le U y_{ij} \;, \forall (i,j) \in \mathcal{A} \tag{12}$$

$$x_{ij} \ge 0 \quad \forall (i,j) \in \mathcal{A}, \;\; y_{ij} \in \{0,1\} \;\; \forall (i,j) \in \mathcal{A} \tag{13}$$

where $V_i^+ = \{j \in V: (i,j) \in \mathcal{A}\}$ and $V_i^- = \{j \in V: (j,i) \in \mathcal{A}\}$, and U is a large positive integer.

Constraints (11) are the well known conservation constraints, constraints (12) are the forcing constraints that y_{ij} takes values one whenever x_{ij} is positive. Note that it suffices to take $U = \sum_{i\in V: b_i>0} b_i$.

As additional notation we use $V_s = \{i \in V: b_i < 0\}$ to denote the set of supply nodes, $V_D = \{i \in V: b_i > 0\}$ the set of demand nodes, and $V_0 = \{i \in V: b_i = 0\}$ the set of transhipment nodes. Let X be the set of vectors (x, y) satisfying (11)-(13). The next proposition characterizes the extreme points of conv(X).

Proposition 5.1 Given (x, y) in X, let

$$F(x,y) = \{a \in \mathcal{A} : 0 < x_a < U, y_a = 1\}$$
$$L(x,y) = \{a \in \mathcal{A} : x_a = 0, y_a \in \{0,1\}\}$$
$$U(x,y) = \{a \in \mathcal{A} : x_a = U, y_a = 1\}$$

Then (x, y) is an extreme point of conv(X) if the graph $D_{x,y} = \big(V, F(x,y)\big)$ contain no cycles.

This characterization will be used in section 6.2 to devise branching rule, a pruning criterion, and also to fix variables in the enumeration tree.

Example 5.1: Consider the following network flow problem

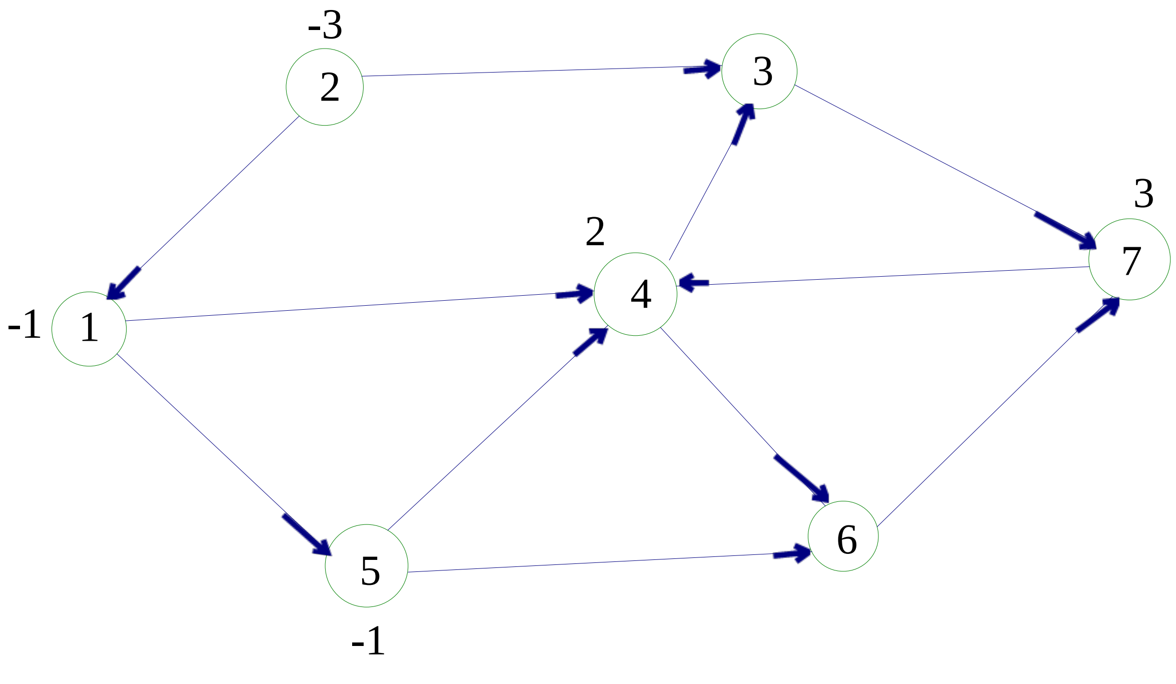

Fig. 2

From the given network flow graph we can verify the flow conservation equation and different inequalities as follows:

At node 1 we have

$$\sum_{j\in V_i^-} x_{ji} - \sum_{j\in V_i^+} x_{ij} = b_i, \forall i \in V \qquad \Rightarrow x_{21} - x_{14} - x_{15} = b_1 = -1$$

Similarly we get other flow conservation constraints on the other nodes

$$-x_{21} - x_{23} = b_2 = -3$$

$$x_{23} + x_{43} - x_{37} = b_3 = 0$$

$$x_{14} + x_{34} + x_{74} - x_{43} - x_{46} = b_4 = 2$$

$$x_{15} - x_{54} - x_{56} = b_5 = -1$$

$$x_{56} + x_{46} - x_{67} = b_6 = 0$$

$$x_{37} + x_{67} - x_{74} = b_7 = 3$$

and $U = \sum_{i \in V: b_i > 0} b_i = b_4 + b_7 = 2 + 3 = 5$

$x_{ij} \leq 5y_{ij}$, $x_{ij} \geq 0$, $y_{ij} \in \{0,1\}$

Consider a proper subset S of nodes for which the net demand is positive, i.e.

$b(S) = \sum_{i \in S} b_i > 0$.The set obtained by summing up the conservation constraints (11) over all nodes in S is called X_S and know as a single node flow problem. Mathematically it takes the form:

$$X_S = \left\{ (x,y) \in R^{|\mathcal{A}|} \times R^{|\mathcal{A}|} : \begin{array}{lr} \sum_{(i,j) \in \delta^-(S)} x_{ij} - \sum_{(i,j) \in \delta^+(S)} x_{ij} = \sum_{i \in S} b_i & (14) \\ x_{ij} \leq U y_{ij} \text{ for } (i,j) \in \mathcal{A} & (15) \\ x_{ij} \geq 0, y_{ij} \in \{0,1\} \text{ for } (i,j) \in \mathcal{A} & (16) \end{array} \right\}$$

where $\delta^+(S) = \{(i,j) \in \mathcal{A}: i \in S, j \notin S\}$ is the set of arcs leaving S and

$\delta^-(S) = \{(i,j) \in \mathcal{A}: i \notin S, j \in S\}$ is the set of arcs entering S.

Example 5.2: Consider the previous example with $S = \{3,5,6,7\}$

To determine the constraints (14) and (15)

$$\delta^+(S) = \{(5,4),(7,4)\}$$

$$\delta^-(S) = \{(2,3),(4,3),(1,5),(4,6)\}$$

$$\sum_{i \in S} b_i = b_3 + b_5 + b_6 + b_7 = 0 - 1 + 0 + 3 = 2$$

Then constraint (14) becomes

$$x_{23} + x_{43} + x_{46} + x_{15} - x_{54} - x_{74} = 2$$

and constraint (15) becomes

$x_{23} \leq 5y_{23}$, $x_{43} \leq 5y_{43}$, $x_{46} \leq 5y_{46}$, $x_{15} \leq 5y_{15}$, $x_{54} \leq 5y_{54,} x_{74} \leq 5y_{74}$.

Because S has a positive supply of 2 units, every feasible flow must contain at least 2 units leaving S. Thus, at least one arc of $\delta^{-}(S) = \{(2,3), (4,3), (1,5), (4,6)\}$, the set of arcs entering S, must be open. This is expressed in the inequality:

$$y_{23} + y_{43} + y_{46} + y_{15} \geq 1$$

Proposition 5.2 For $S \subset V$ with $\sum_{i \in S} b_i > 0$, the basic dicut inequality

$$\sum_{(i,j) \in \delta^{-}(S)} y_{ij} \geq 1$$

is valid for X.

These inequalities have been used to formulate the Steiner tree problem.

Consider again the same subset $S = \{3,5,6,7\}$. A relaxation of the aggregated set X_S is obtained if x_{23} and x_{15} are replaced by their upper bounds, and x_{54} and x_{74} are replaced by their lower bounds for zero. The resulting set is:

$$5y_{23} + x_{43} + x_{46} + 5y_{15} \geq 2, y_{23}, y_{15} \in \{0,1\}, x_{43}, x_{46} \geq 0.$$

Applying mixed integer rounding procedure (MIR) we get the inequality:

$2y_{23} + x_{43} + x_{46} + 2y_{15} \geq 2.$

In general we have the following result.

Proposition 5.3 For $S \subset V$ with $\sum_{i \in S} b_i > 0$, the mixed dicut inequality

$$\sum_{(i,j) \in \delta^-(S) \setminus C} x_{ij} + \sum_{(i,j) \in C} b(S) y_{ij} \geq b(S)$$

is valid for X for all $C \subseteq \delta^-(S)$.

Example 5.3: Let us take $S = \{4,5,6,7\}$ for the previous example

To determine the mixed dicut inequality using the above proposition

We have $S = \{4,5,6,7\}$ and $V = \{1,2,3,4,5,6,7\}$

$$\delta^-(S) = \{(1,4), (1,5), (3,7)\}$$

$$\sum_{i \in S} b_i = b_4 + b_5 + b_6 + b_7 = 2 - 1 + 0 + 3 = 4$$

$$\Longrightarrow b(S) = 4 > 0$$

Let $C = \{(3,7)\}$. And $\delta^-(S) \setminus C = \{(1,4), (1,5)\}$.Then using the above proposition we get the following mixed dicut inequality

$$x_{14} + 4y_{37} + x_{15} \geq 4.$$

Now the flow going from $V \setminus S = \{1,2\}$ to satisfy demand in S passing through the arc (3,7) is at most 3 units, the supply of node 2. This observation allows us to tighten the coefficient associated to y_{37} giving the valid inequality:

$$x_{14} + 3y_{37} + x_{15} \geq 4$$

In general, for $e \in C \subseteq \delta^-(S)$ let $E(S) = \{(i,j) \in \mathcal{A}: i,j \in S\}$,$V_{e,S}^+ = \{i \in S: b_i > 0\}$

and there exists a dipath in $G_S = (S, E(S))$ between the head node of e and the node i}.

$$V_{e,S}^- = \{i \in V \setminus S: b_i < 0 \text{ and there exists a dipath in}$$

$$G_{V \setminus S} = \big(V \setminus S, E(V \setminus S)\big) \text{ between the node i and the tail node of } e\}$$

$$\alpha_e(S) = \min\left\{\sum_{i\in V_{e,S}^+} b_i\,, \sum_{i\in V_{e,S}^-} |b_i|\right\}$$

Proposition 5.4 For $S \subset V$ with $\sum_{i\in S} b_i > 0$, the simple inflow-outflow inequality

$$\sum_{(i,j)\in\delta^-(S)\setminus C} x_{ij} + \sum_{(i,j)\in C} \alpha_{ij}(S) y_{ij} \geq b(S)$$

is valid for X for all $C \subset \delta^-(S)$.

This modification is important when the underlying graph is sparse. For the single item uncapacitated lot-sizing problem all the inequalities required to give a complete description of the convex hull are of this type.

Example 5.4: Consider $S = \{3,4,6,7\}$

To find the mixed dicut inequality we use proposition 5.4

$$V \setminus S = \{1,2,5\}$$

$$b(S) = b_3 + b_4 + b_6 + b_7 = 0 + 2 + 0 + 3 = 5$$

$$\Rightarrow b(S) = 5$$

$$\delta^-(S) = \{(2,3), (1,4), (5,4), (5,6)\}$$

Let $C = \{(5,4), (5,6)\}$ and $(i,j) \in \delta^-(S) \setminus C = \{(2,3), (1,4)\}$

$$\alpha_{ij}(S) = \min\left\{\sum_{i\in V_{e,s}^+} b_i\,, \sum_{i\in V_{e,S}^-} |b_i|\right\}$$

where $V_{e,s}^+ = \{4,7\}$ and $V_{e,S}^- = \{1,2,5\}$

$\Rightarrow \alpha_{ij}(S) = \min\{b_4 + b_7, b_1 + b_2 + b_5\} = \min\{5,5\} = 5$

Then we get the following dicut inequality

$$x_{23} + x_{14} + 5y_{54} + 5y_{56} \geq 5.$$

Now suppose that the contribution of the flow in arc(7,4) to satisfy the demand $b(S)$ is measured separately. In this case, the maximum flow that can pass through the arc $(5,6)$

using the arcs of $E(S)$ except for (7,4) to satisfy the demand $b(S)$ is 3,the demand of node 7. So the inequality

$$x_{23} + x_{14} + 5y_{54} + 5y_{56} + x_{74} \geq 5.$$

is valid.

In general, given $R \subset E(S)$, define $V_{e,S}^{R} = \{i \in S: b_i > 0$ and there exist a dipath in $G_S = (S, E(S) \setminus R)$ between the head of e and the node i} and $\alpha_e^R(S) = \sum_{i \in V_{e,S}^R} b_i$.

Proposition 5.5. For $S \subset V$ with $\sum_{i \in S} b_i > 0$, the simple inflow-outflow inequality

$$\sum_{(i,j) \in (\delta^-(S) \setminus C) \cup R} x_{ij} + \sum_{(i,j) \in C} \alpha_{ij}(S) y_{ij} \geq b(S)$$

is valid for X for all $C \subset \delta^-(S), R \subseteq E(S)$.

Example 5.5: Consider $S = \{3,5,6,7\}$ from example 5.1 another possible relaxation of X_S is given by

$5y_{23} + 5y_{43} + x_{46} + 5y_{15} \geq 2 + x_{74}, x_{74} \leq 5y_{74}$

Letting $\bar{x}_{74} = 5y_{74} - x_{74} \geq 0$ and $\bar{y}_{74} = 1 - y_{74}$ we get

$$5y_{23} + 5y_{43} + x_{46} + 5y_{15} \geq 2 + 5y_{74} - \bar{x}_{74} = 2 + 5(1 - \bar{y}_{74}) - \bar{x}_{74} = 7 - 5\bar{y}_{74} - \bar{x}_{74}$$
$$\Longrightarrow 5y_{23} + 5y_{43} + x_{46} + 5y_{15} + \bar{x}_{74} + 5\bar{y}_{74} \geq 7$$

Now applying the MIR procedure, we obtain

$$2y_{23} + 2y_{43} + x_{46} + 2y_{15} + \bar{x}_{74} + 2\bar{y}_{74} \geq 4$$

and reintroducing the original variables:

$$2y_{23} + 2y_{43} + x_{46} + 2y_{15} + \bar{x}_{74} + 2\bar{y}_{74} \geq 4$$
$$\Longrightarrow 2y_{23} + 2y_{43} + x_{46} + 2y_{15} + 5y_{74} - x_{74} + 2(1 - y_{74}) \geq 4$$
$$\Longrightarrow 2y_{23} + 2y_{43} + x_{46} + 2y_{15} + 3y_{74} - x_{74} + 2 \geq 4$$

Thus we get the following valid inequality:

$$2y_{23} + 2y_{43} + x_{46} + 2y_{15} \geq 2 + (x_{74} - 3y_{74})$$

The general expression is given in the next proposition

Proposition 5.6 For $S \subset V$ with $\sum_{i \in S} b_i > 0$, the mixed dicut with outflow inequality

$$\sum_{(i,j) \in (\delta^-(S) \setminus C^-)} x_{ij} + \sum_{(i,j) \in C^-} b(S) y_{ij} \geq b(S) + \sum_{(i,j) \in C^+} \{x_{ij} - r(S) y_{ij}\}$$

is valid for X for all $C^- \subset \delta^-(S)$, and $C^+ \subseteq \delta^+(S)$ with $r(S) = U - b(S)$

Proof: Consider the following relaxation of X_S:

$$\sum_{(i,j) \in \delta^-(S) \setminus C^-} x_{ij} + U \sum_{(i,j) \in C^-} y_{ij} \geq b(S) + \sum_{(i,j) \in C^+} x_{ij}$$

plus $x_{ij} \leq U y_{ij}$ for all $(i,j) \in \delta^-(S) \cup C^+$ and all y_{ij} binary.

Defining the variable $\bar{x}_{ij} = U y_{ij} - x_{ij}$, and $\bar{y}_{ij} = 1 - y_{ij}$, we can make the substitution $x_{ij} = U - U\bar{y}_{ij} - \bar{x}_{ij}$ for $(i,j) \in C^+$. Now the previous constraint can be written as

$$\sum_{(i,j) \in \delta^-(S) \setminus C^-} x_{ij} + U \sum_{(i,j) \in C^-} y_{ij} + \sum_{(i,j) \in C^+} \bar{x}_{ij} + U \sum_{(i,j) \in C^+} \bar{y}_{ij} \geq b(S) + U|C^+|$$

Applying the MIR procedure, we get:

$$\sum_{(i,j) \in \delta^-(S) \setminus C^-} x_{ij} + \sum_{(i,j) \in C^+} \bar{x}_{ij} \geq b(S) \left(1 + |C^+| - \sum_{(i,j) \in C^-} y_{ij} - \sum_{(i,j) \in C^+} \bar{y}_{ij} \right)$$

Let us introduce the original variable:

$$\sum_{(i,j) \in \delta^-(S) \setminus C^-} x_{ij} + \sum_{(i,j) \in C^+} (U y_{ij} - x_{ij}) \geq b(S) \left(1 + |C^+| - \sum_{(i,j) \in C^-} y_{ij} - \sum_{(i,j) \in C^+} (1 - y_{ij}) \right)$$

$$\Rightarrow \sum_{(i,j) \in \delta^-(S) \setminus C^-} x_{ij} + \sum_{(i,j) \in C^+} U y_{ij} - \sum_{(i,j) \in C^+} x_{ij} \geq d$$

$$\text{where } d = b(S)\left(1 + |C^+| - \sum_{(i,j)\in C^-} y_{ij} - \sum_{(i,j)\in C^+} (1 - y_{ij})\right)$$

$$= b(S)\left(1 + |C^+| - \sum_{(i,j)\in C^-} y_{ij} - \sum_{(i,j)\in C^+} y_{ij} - \sum_{(i,j)\in C^+} 1\right)$$

Then we have

$$\sum_{(i,j)\in\delta^-(S)\setminus C^-} x_{ij} + \sum_{(i,j)\in C^-} b(S)y_{ij} \geq$$

$$b(S) + b(S)|C^+| - b(S)\sum_{(i,j)\in C^+} y_{ij} - b(S)\,|C^+| - \sum_{(i,j)\in C^+} Uy_{ij} + \sum_{(i,j)\in C^+} x_{ij}$$

$$\Rightarrow \sum_{(i,j)\in\delta^-(S)\setminus C^-} x_{ij} + \sum_{(i,j)\in C^-} b(S)y_{ij} \geq b(S) + \sum_{(i,j)\in C^+} \{x_{ij} - Uy_{ij} - b(S)y_{ij}\}$$

Now letting $r(S) = U - b(S)$ we get the required inequality:

$$\sum_{(i,j)\in(\delta^-(S)\setminus C^-)} x_{ij} + \sum_{(i,j)\in C^-} b(S)y_{ij} \geq b(S) + \sum_{(i,j)\in C^+} \{x_{ij} - r(S)y_{ij}\}$$

5.1. Difficulty of the Separation Problem

First we formalize the separation problem for simple dicut inequalities. To find a violated simple dicut inequality, we look for a subset of nodes S such that $b(S) > 0$ and $\sum_{(i,j)\in\delta^+(S)} \bar{y}_{ij} < 1$

This can be seen as a minimum cut problem with an additional constraint to ensure that $b(S) > 0$. For $i \in V$, define variable $z_i = 1$ if i belongs to S, and $z_i = 1$ otherwise. The separation problem reduces to solving the problem:

$$\xi = \min\left\{\sum_{(i,j)\in\mathcal{A}} \bar{y}_{ij}\, z_j\,(1 - z_i) : \sum_{i\in V} b_i z_i > 0 \;, z_i \in \{0,1\} \text{ for all i}\in\text{V}\right\}$$

If $\xi < 1$, the set S defined by $\{i \in V : z_i = 1\}$ leads to a violated inequality

By reduction from the exact partitioning problem the separation problem associated to the simple dicut inequalities can be shown to be NP-complete [9]. However for the single source problem in which $|V_S| = 1$, the imposed constraint can be dropped. The separation problem can be solved in polynomial time. It can be reduced to $|V_D|$ minimum s-t cut problems where s is the source and t varies over the set V_D.

The problem of finding a violated mixed dicut inequality can be stated as follows: given a fractional point$(\bar{x}, \bar{y})$ we look for $S \subset V$ with $b(S) > 0$ and $C \subseteq \delta^-(S)$ such that:

$$\sum_{(i,j)\in\delta^-(S)\setminus C} \bar{x}_{ij} + \sum_{(i,j)\in C} b(S)\bar{y}_{ij} < b(S)$$

For a given S, finding the most violated inequality is trivial. It suffices to set $C = \{ij \in \delta^-(S): \bar{x}_{ij} > b(S)\bar{y}_{ij}\}$ .Therefore, the principal difficulty regarding the separation of dicut inequalities is to find the right set *S*.

6. Solution method

Numerous exact and heuristic approaches were developed to find exact and approximate solution for fixed charge network flow problem. The basic approach to obtaining algorithms for the exact solution of the fixed charge problem is to formulate them as mixed integer linear programs and to solve these programs by decomposing them into a master problem (which is an integer program) and a series of subproblems which are linear program. Here we consider two exact methods and one heuristic method to solve the problem.

6.1. Exact algorithm for fixed charge transportation problem

As we have seen in section 4.2, fixed charge transportation problem can be considered as the special case of fixed charge network flow problem.
The problem can be formulated as:

(Problem 1) minimize $\sum_i \sum_j (c_{ij} x_{ij} + f_{ij} y_{ij})$

$$\sum_j x_{ij} = D_j \ , \quad j = 1,2,3, \dots, n \qquad (17)$$

$$\sum_i x_{ij} = S_i \ , \quad i = 1,2,3, \dots, m \qquad (18)$$

$$u_{ij} y_{ij} - x_{ij} \geq 0 \ , \forall(i,j) \qquad (19)$$

$$x_{ij} \geq 0, \forall(i,j) \ , \forall(i,j)$$

$$y_{ij} = 0 \text{ or } 1 \ , \quad \forall(i,j)$$

where c_{ij} is the unit cost of transporting goods from source i to destination j,f_{ij} is the fixed charge if route ij is open (i.e., if the amount shipped, $x_{ij} > 0$), y_{ij} is a 0-1 variable, D_j is the j^{th} demand, S_i the i^{th} supply, and $u_{ij} = \min(D_j, S_i)$. We assume $f_{ij} \geq 0$.

Hirsch and Dantzig showed that, for any fixed-charge problem, an optimal solution occurs at an extreme point of the constraint set. This important result is basic to the methods of solution, because it restricts the search for an optimum to the set of extreme points.

6.1.1. Decomposition approach

Examination of the constraint set shows that the integer variables occur only in (19) and that each integer variable y_{ij} is associated with only one continuous variable x_{ij} .Thus, constraint sets (17) and (18) express the usual supply and demand requirements. The quantity u_{ij} in constraint set (19) is given by $u_{ij} = \min(D_j, S_i)$ and reperesents the maximum amount of goods that can be shipped from source i to destination j.

As Baliniski [13] showed, an approximate solution to this problem can be obtained by disregarding the individualities of the y_{ij} ,and recognizing that at linear programming optimality constraint set (19) will be satisfied with equality. That is at LP optimality,

$$u_{ij} y_{ij} - x_{ij} = 0 \implies y_{ij} = \frac{x_{ij}}{u_{ij}}$$

Substituting this in the objective function we get

$$\text{minimize} \sum_i \sum_j \left[c_{ij} + \left(\frac{f_{ij}}{u_{ij}} \right) \right] x_{ij} \qquad (20)$$

$$\text{s.t constraint set (17) and (18) and } x_{ij} \geq 0 .$$

Thus (19), can be thought of as providing a weak coupling between the discreet and coupling that leads to decomposition.

Suppose we specify a particular set of values for the variable y_{ij} ; that is, we fix each y_{ij} at either the 0 or 1. Then we have created the following subproblem:

$$\text{(problem 2)} \qquad \text{minimize } Z(x) = \sum_i \sum_j c_{ij} x_{ij}$$

$$\sum_j x_{ij} = D_j \ , \qquad j = 1,2,3, \dots, n \qquad (21)$$

$$\sum_i x_{ij} = S_i \ , \qquad i = 1,2,3, \dots, m \qquad (22)$$

$$u_{ij} y_{ij} - x_{ij} \geq 0 \, , \forall (i,j) \qquad (23)$$

$$x_{ij} \geq 0, \forall (i,j) \, , \forall (i,j)$$

This subproblem involves only continuous variables, and hence can be solved as a linear program. In fact, it has the structure of a transportation problem in which only certain routes are allowed to be open (i.e. route ij is open if $y_{ij} = 1$).

If Problem 2 does not have a feasible solution for this particular set of y_{ij}, then Problem 1 also does not have a feasible solution for these y_{ij}. If, however, an optimal solution $\bar{x}_{ij}$ is found for Problem 2, then the total cost for this case is known, since the fixed charges are specified. We have the following algorithm for solving the fixed charge transportation problem.

1. Find a feasible solution to problem 1: One such solution is obtained by specifying all routes open, in which case problem 2 is just the standard transportation problem. We can use also Balinski's approximation method [13].
2. Enumerate the finite set of vector $y = (y_{11}, \dots, y_{mn})$ where the components $y_{ij} = 0$ or 1. We shall refer to the vector y specifies a sequence of zeros and ones.
3. For each y found in step 2, solve Problem 2. If Problem 2 is infeasible for this value of y, so is Problem 1. If an optimal solution to Problem 2 is found, evaluate

$$\sum_i \sum_j \left(c_{ij}\,\bar{x}_{ij} + f_{ij}\,y_{ij}\right),$$

   setting $y_{ij} = 0$ if $\bar{x}_{ij} = 0$,

4. The optimal is min $\sum_i \sum_j \left\{c_{ij}\,\bar{x}_{ij} + f_{ij}\,y_{ij}\right\}$ for all y.

Although this decomposition procedure is guaranteed to produce an optimal solution to Problem 1, it does not, on the face of it, appear attractive computationally for problems of even moderate size, because the number of LP problems to be solved is 2^{mn}. However, there are many regularities in the transportation problem, and there is a key bound on the fixed charges that can be exploited, thus reducing the number of extreme points that need to be searched.

6.1.2. Upper bound on the total fixed charge

Suppose we have a feasible solution to problem 1. Let T_0 be the value of the objective function for the initial solution which can be used to obtain an upper bound on the total fixed charge in the optimal solution.

This upper bound is obtained by solving problem 1with all fixed charges set equal to 0. In this case, problem 1 reduces to just the ordinary transportation problem. Let us call the optimal value for this case $ZMIN$, since this the lowest value of the variable cost $Z(x)$. If we subtract $ZMIN$ from T_0 we obtain at once an upper $FMAX$ on the fixed charges for the optimal solution to problem1, i.e. $FMAX = T_0 - ZMIN$. Any solution of problem 1 with lower total cost than T_0 must have a total fixed charge less than $FMAX$. As lowest cost solutions,$T_i < T_{i-1}, i = 1,2, ...$ are found, the value $FMAX$ can be reduced to

$FMAX = T_i - ZMIN$.

6.1.3. Lower bound on the fixed charge

A crude lower bound $ZMIN$ on the fixed costs can be found by considering the supply constraints one at a time. Enough routes must be opened from each source to at least permit disposing of the supplies there. Thus, consider the sources one at a time and find the cheapest set of routes out of that source that can absorb the supply.

Formally, solve the problem

$$\text{minimize} \sum f_{ij}\, y_{ij}$$

s.t

$$\sum_j D_j\, y_{ij} \geq S_i \qquad (24)$$

for each source $i(i = 1,2,3, \dots, m)$ and then sum these minima. For small problems, these cheapest sets of routes can be obtained by enumeration.

6.1.4. Branch and Bound

The algorithm for finding eligible 0-1 strings in such a way that this problem need not be solved explicitly, since it is guaranteed that all 0-1 strings generated will satisfy the source constraint of equation (24) and hence will equal or exceed this crude bound, $FMIN$.

Preliminary tests of the algorithm indicated that the range defined by $FMAX$ and $FMIN$ was often large and that it included a large number of feasible extreme points for which transportation problems had to be solved. To obtain tighter bounds on $FMAX$, the basic idea of branch-and-bound techniques was added.

Suppose that the combination of open routes from Source 1 is specified, but open routes from other sources are unspecified. Solving the transportation problem under these conditions specifies the minimum variable cost, given these open routes. Thus, $FMAX = T_0 - ZMIN_1$, where T_0 is the value of the objective function for the approximate integer solution and $ZMIN_1$ is the minimum transportation cost if the allowable open routes from source 1 are specified . Note that $ZMIN_1 \geq ZMIN$ and, hence, $FMAX_1$ may be less than or equal to that obtained when the routes from Source 1 are not specified. Adding this bounding calculation can (and for most route combinations does) reduce the number o f extreme points that need to be examined.

This bounding technique can be applied iteratively; that is, it can be applied with the routes from source 1 specified, or from sources 1 and 2, or source 1, 2 and 3, and so on. It can also be used selectively for example, one specify the open routes from sources 1 and 5.

6.1.5. Generating Vertices

The key part of the problem is to reduce the number of vertices to be examined. Since there are mn routes, there are $(2^n - 1)^m$ potential 0-1 strings, recognizing that each of the m sources must ship to at least one of the n destinations.

The basic idea is to arrange the vertices in a favourable sequence for enumeration. Define the open-route combinations from a given source as a number in a base 2^n number system. Thus, for sources, an m-digit number defines a set of open routes; for example, 6312 denotes the combinations of open routes shown in the table:

source	Destination		
	3	2	1
1	1	1	0
2	0	1	1
3	0	0	1
4	0	1	0

$(1 = \text{open}, 0 = \text{closed})$

A systematic way of enumerating the combinations of open routes is to count in a base 2^n system, starting from 11111... 1 and continuing to $2^{n-1}2^{n-1} \ldots 2^{n-1}$. There are, of course, many regularities in the structure of transportation problems that allow us to throw out many of these numbers. We will make use of the following list:

1. Maximum open routes: since the optimal solution must lie at a vertex we need consider only basic solutions. The maximum number of nonzero elements in a basic solution is $m + n - 1$. Hence, any m −digit number that calls for more than $m + n - 1$ open routes can be discarded.
2. Row feasibility: each source must dispose of its entire supply, i.e. $\sum_j D_j y_{ij} > S_i$.
3. Single source: at least one route must be open to each destination.
4. Column feasibility: the total supplies potentially available at each destination must at least equal the demand, $\sum_i S_i y_{ij} > D_j$.
5. Strong feasibility: we use
   a. For each destination supplied by a single source, reduce the supply at this source by this demand and eliminate the demand from further consideration.
   b. For each source supplying only one demand, decrease the demand at this destination by the source supply.
   c. Work back and forth between steps (a) and (b) until no further improvements are possible or until tests 2 or 4 are failed.

The procedure consists of three parts: initialization, bounding, and iteration. The initialization, which is performed once and arranges the data in a favorable way for the iterations, consists of five steps:

Step 1: Rearrange the sources in decreasing order of the supplies available.

Step 2: For each source, compute the total demand for each combination of open routes. If the supply exceeds the demands from a combination of destinations, the combination is infeasible. Assign an effectively infinite fixed charge M to such combinations.

Step 3: For the remaining combinations, compute the total fixed charge associated with opening the combination.

Step 4: For each source, sequence the combinations of destinations in order of increasing fixed cost.

Step 5: For each row in the fixed-cost table prepared in Step 4, find the largest allowable fixed charge in that row. The largest allowable fixed charge is determined by subtracting the sum of the smallest fixed charge in each of the other rows from $FMAX$. For all combinations with larger fixed charges assign a value M.

The data are now arranged so that the sum of the fixed costs in the first column corresponds to the $FMIN$ defined previously and so that infeasible source-destination combinations are not considered.

The bounding procedure involves two steps. The sources for which bounding is to be performed are specified as input, and the procedure is carried out whenever the combination of open routes is changed. Assume that the bounding is being done for the first k sources. Then:

Step 6: Solve the transportation problem with only the specified routes open for sources 1 to k and with all routes for sources $k+1$ through m open.

Step 7: Determine the value of $FMAX$ to be used in the subsequent iterations.

The iterations proceed through the following five steps:

Step 8: Specify the next combination of the first $m-1$ sources. Assume that the last source (smallest supply) can ship to any destination (all routes from this source open), and perform the column feasibility test $\sum_i S_i y_{ij} \geq D_j$. If this is failed, repeat Step 8, since no feasible solution exists that involves this combination of the first $m-1$ sources. If it is passed, begin examining route combinations. Note that by arranging the supplies in decreasing order, the column feasibility test is iterative. That is, the test can be applied to the last 2,3, ..., sources together by assuming they can ship to all destinations.

Step 9: Check that no more than $m+n-1$ routes are open. Discard combinations that exceed this bound.

Step 10: Perform the destination feasibility tests 3, 4, and 5 described above.

Step 11: If a combination passes the $FMAX$, $n+m-1$, and destination feasibility tests solve the transportation problems with only the routes defined by the m −digit number open.

Step 12: Increment the m digit number by one. If the $FMAX$ test is failed, go to Step 8; otherwise, go to Step 9. Terminate if there are no more numbers to be examined.

Example 6.1: Consider the following fixed charge transportation problem with the following data

$$m = 4, \qquad n = 3$$

$$c_{ij} = \begin{pmatrix} 7.60 & 5.94 & 5.94 \\ 0.71 & 0.64 & 0.69 \\ 2.83 & 1.70 & 0.79 \\ 5.94 & 5.64 & 2.02 \end{pmatrix}$$

$$f_{ij} = \begin{pmatrix} 14 & 14 & 11 \\ 15 & 8 & 7 \\ 12 & 12 & 9 \\ 13 & 11 & 5 \end{pmatrix}$$

$$D = (50 \quad 15 \quad 5) \text{ and } S = (25 \quad 20 \quad 15 \quad 10)$$

Solution: we have

Number of source $(m) = 4$ and number of destination $(n) = 3$

The maximum number of non zero elements in a basic variable is $m+n-1 = 6$

The maximum number of vertices is $(2^n - 1)^m = (2^3 - 1)^4 = 2401$.

Let us express the supplies and demand in the following table 1.

Table 1

Fixed charge table

Source	Demand		
	50	15	5
25	14	14	11
20	15	8	7
15	12	12	9
10	13	11	5

From table 1, we get the following seven destination combinations are infeasible:

From source1 to destination 2,or to 3,or to 2 and 3

From source 2 only to destination 2 or to 3;

From source 3 only to destination 3;

From source 4 only to destination 3;

Computing the total fixed costs and assigning a value of 1000 to the infeasible combinations yields Table 2. The number of combinations that must be considered further has been reduced to 4(5) (6) (6) =720 from the original 2401.

Table 2

Total fixed costs

Source	Destination combination						
	1	2	3	4	5	6	7
1	14	1000	28	1000	25	1000	39
2	15	1000	23	1000	22	15	30
3	12	12	24	1000	21	21	33
4	13	11	24	1000	18	16	29

Sequencing the total costs in increased order for each source and setting up a separate table showing the order of destination combinations yield Tables 3 and 4.

Table 3

Sequenced cost table

source	Fixed cost						
1	14	25	28	39	1000	1000	1000
2	15	15	22	23	30	1000	1000
3	12	12	21	21	24	33	1000
4	11	13	16	18	24	29	1000

Table 4

Combination order

Source	Destination combination						
1	1	5	3	7	2	4	6
2	1	6	5	3	7	2	4
3	1	2	5	6	3	7	4
4	2	1	6	5	3	7	4

One feasible solution involves opening routes $(1,1), (2,1), (3,2), (4,1)$ and $(4,3)$. We can find the fixed cost for this solution:

$$\text{fixed cost } = 14 + 15 + 12 + 13 + 5 = 59$$

For the cost structure used in the test problem, this solution results in an $FMAX$value of 65. Using this value, additional entries can be set to 1000 as is done in Table 5. The reason is that, when these combinations are used, even with the cheapest entry (first column) for all other sources, the total fixed cost exceeds 65.

Table 5

Sequenced cost table (Revised)

Source	Fixed cost						
1	14	25	1000	1000	1000	1000	1000
2	15	15	22	23	1000	1000	1000
3	12	12	21	21	24	1000	1000
4	11	13	16	18	24	1000	1000

From the combination, 189 fail the tests in step 8, 9, and 10 as shown in the following table 6

Table 6

Tests	Number of combination failing tests
First column feasibility check	30
FMAX Test	136
Number of open routes exceeds $n + m - 1$	2
No route open to some destination	8
Test for column feasibility	13
Total	189

Thus total number of combinations has been reduced to 200 from 2401 and the previous 720. From this combinations we get 11 combinations pass all tests require solution of transportation problem.

Using the above algorithm we get the following exact solution:

$$\text{Total cost} = 329$$

$$\text{Transportaion cost} = 270$$

$$\text{Fixed cost} = 60$$

In general, using this algorithm we can find a solution for series of test problems up to 6×8 in size. The algorithm is most efficient when the fixed costs are large compared to the variable costs.

6.2. Branch and cut algorithm for UFCNFP

Mixed integer linear programming can be solved using branch and cut method which is an exact method combining branch and bound and cutting planes. The basic idea is to take a linear programming relaxation of the problem, solve the relaxation, and then either improves the relaxation by adding additional valid constraints, or split the problem into two or more sub problems and repeats the process.

Let us see the general branch and cut algorithm to solve any integer linear programming problem. The set of active nodes in the branch and-bound tree is denoted by $\mathcal{L}$. The value of the best known feasible point for (ILP) is stored as $\bar{z}$, and provides an upper bound on the optimal value of the problem. This point is called the incumbent solution. We use $\underline{z}_\ell$ to denote a lower bound on the optimal value of the current sub problem ℓ under consideration. This lower bound is initialized to the value of the parent node, and is then updated to the value of the LP relaxation of the sub problem.

Algorithm 6.1: A general branch-and-cut algorithm

1. Initialization: Denote the initial problem by ILP^0 and define the set of active nodes to be $\mathcal{L} = \{\mathrm{ILP}^0\}$. Let $\bar{z} = +\infty$ and set $\underline{z}_\ell = -\infty$ for the initial problem $\ell \in \mathcal{L}$.
2. Termination: If $\mathcal{L} = \emptyset$, then STOP. If $\bar{z} = \infty$, then (ILP) is infeasible; else, the solution x^* which yielded the incumbent objective value $\bar{z}$ in step 7(b) or step 5 is optimal.
3. Problem selection: Select and delete a problem ILP^ℓ form $\mathcal{L}$.
4. Relaxation: Solve the LP relaxation of ILP^ℓ. If the relaxation is infeasible, set $\underline{z}_\ell = +\infty$ and go to step 7. If the relaxation is unbounded set $\underline{z}_\ell = -\infty$. If the relaxation has a finite optimal value let $x^{\ell R}$ be optimal solution and set $\underline{z}_\ell = c^T x^{\ell R}$.
5. Heuristic rounding: If $x^{\ell R}$ is not integral, and if desired, use a rounding approach or a heuristic approach to construct a feasible integral solution$x^{\ell H}$.
   Update $\bar{z} = \min\{c^T x^{\ell H}, \bar{z}\}$.
6. Add cutting planes: If desired, search for cutting planes that violated by $x^{\ell H}$; if any are found, add them to the relaxation and return to step 4.
7. Fathoming and Pruning:
   (a) Fathom by bounds or infeasibility: If $\underline{z}_\ell \geq \bar{z}$ go to step 2.
   (b) Fathom by integrality: If $\underline{z}_\ell < \bar{z}$ and $x^{\ell R}$ is integral feasible, update $\bar{z} = \underline{z}_\ell$, delete from $\mathcal{L}$ all problems with $\underline{z}_\ell \geq \bar{z}$, and go to step 2.
8. Partitioning: Let $\{S^{\ell j}\}_{j=1,2,3,\dots,k}$ be a partition of the constraint set S^ℓ of problem ILP^ℓ.Add problems $\{\mathrm{ILP}^{\ell j}\}_{j=1,2,3,\dots,k}$ to $\mathcal{L}$, where $\mathrm{ILP}^{\ell j}$ is ILP^ℓ with feasible region restricted to $S^{\ell j}$, and $\underline{z}_{\ell j} = \underline{z}_\ell$ for $j = 1,2,3,\dots,k$. Return to step 2.

Now let us describe the branch and cut system for UFCNFP. We begin with some basic implementation issues, and then we present the separation heuristic, the primal heuristic, branching rules, and finally pruning and variable fixing criteria.

6.2.1. The Basics

In the implementation of branch and cut algorithm we use branch and bond algorithm with a series of entry points. Using these entry points we can generate cuts and add them to the matrix, apply heuristics and develop branching rules. At the top node we have used those entry points to generate cuts, and to apply a heuristic. In the enumeration tree we have used the entry points to generate cuts, prune a node, to choose branching variables and to implement a primal heuristic.

Because several cuts can be generated from the same set *S*, we have set up a set pool in order to store the candidate sets. The set pool is a dynamic double linked list. Sets are active until an associated frequency parameter falls below a certain value, at which point the set becomes inactive. How this parameter is updated and when a set is declared inactive is described later in the Cut generation step.

6.2.2. Separation

The separation consists of the following steps: cut deletion, shrinking, set generation, cut generation and reoptimization. One realization of all these steps is called a pass. The default number of passes at the top node has been fixed at 30, where as in the enumeration tree it has been fixed at 5. We describe now each step of a pass.

- Cut deletion. Because the number of violated dicuts can be large, keeping all of them in the matrix during all the passes can be too expensive. Therefore, we eliminate the non binding cuts from the matrix at the beginning of each pass. Because some of the deleted cuts may be violated later, we perform cut pool separation at the beginning of the cut generation step. No cuts are deleted from the cut pool.
- Shrinking. In order to reduce the size of the graph on which we search for "interesting" subsets, the graph is shrunk based on the current linear programming solution. Specifically, whenever $\bar{y}_{ij} > 0.99$ and $\bar{x}_{ij} > 10^{-6}$, the two end nodes i, j are

contracted into one super node. The demand of the new super node is the sum of the demands. Only nodes are contracted, so the resulting reduced graph typically contains multiple arcs and loops. This shrinking procedure is heuristic. Arcs with $\bar{y}_{ij} > 0$ but $\bar{x}_{ij} = 0$ are not used for shrinking because the addition of the dicut inequalities often forces $y_{ij} > 0$ artificially in the linear programming relaxation, even when there is never flow in the corresponding arc.

- Subset generation. The dicut inequalities are based on "node subsets". Therefore finding good subsets can reduce the number of iterations of cut generation, and also lead to a better top node reformulation. In our implementation, three greedy procedures are used to generate subsets for a given fractional solution$(\bar{x}, \bar{y})$. They differ in the choice of the initial node and in the quantity used to enlarge the current set. Below, we describe these choices:

➢ Initialize $S = \{i_0\}$ for $i_0 \in V_S \cup V_D$. Enlarge S using $\max\{\bar{y}_{ij} : (i,j) \in \delta^-(S), \bar{y}_{ij} \in (0,1)\}$.

➢ Initialize $S = \{i_0\}$ for $i_0 \in V_S \cup V_D$. Enlarge S using $\max\{\bar{x}_{ij} - b(S)\bar{y}_{ij} : (i,j) \in \delta^-(S), \bar{y}_{ij} \in (0,1)\}$.

➢ Given an arc $a = (i_0 j_0)$ such that $\bar{y}_{i_0 j_0}$is fractional, two candidate sets S are built. In the first case, we start with a set S such that $j_0 \in S, b(S) > 0$ and $a \in \delta^-(S)$, and

We expand S using the criterion $\max\{\bar{x}_{ij} - b(S): (i,j) \in \delta^-(S), \bar{y}_{ij} \in (0,1), b(S) + d_i > 0\}$. In the second case, we start with a set $\bar{S} = V \setminus S$ such that $i_0 \in V \setminus S, b(V \setminus S) < 0$, and $a \in \delta^+(V \setminus S)$ and we expand $V \setminus S$ using the criterion

$$\max\{\bar{x}_{ij} - |b(V \setminus S)\bar{y}_{ij}| : (i,j) \in \delta^-(V \setminus S), \bar{y}_{ij} \in (0,1), b(V \setminus S) + d_j < 0\}$$

The procedure stops either when the maximum number allowed is reached or when a violated inequality can be generated.

The three procedures are called sequentially. The sets generated are stored in the set pool with the status "active", and the frequency parameter initialized at zero.

- Cut generation. We start by performing cut pool separation. Then for each active set S, three dicut inequalities can be generated:

a. Simple dicut

If $\sum_{a\in\delta^-(S)} \bar{y}_a < 1 - 0.015$, then the inequality

$$\sum_{a\in\delta^-(S)} y_a < 1$$

is added to the cut pool.

b. Simple inflow-outflow inequality.

Define $C = \{a \in \delta^-(S): \bar{x}_a > \alpha_a(S)\bar{y}_a\}$. If $\sum_{a\in\delta^-(S)\setminus C} \bar{x}_a + \sum_{a\in C} \alpha_a(S)\bar{y}_a < b(S) - 0.015$, then the inequality $\sum_{a\in\delta^-(S)} x_a + \sum_{a\in C} \alpha_a(S)y_a \geq b(S)$ is added to the cut pool. The coefficient $\alpha_{ij}(S)$ is computed using breadth-first-search to determine the sets $V^+_{ij,S}$ and $V^-_{ij,S}$ defined in proposition 5.4. Then, $\alpha_{ij}(S) = \min\{b(S), \sum_{k\in V^+_{ij}} b_k, \sum_{k\in V^-_{ij}} |b_k|\}$.

c. Mixed dicut with outflow

Define $C^- = \{a \in \delta^-(S): \bar{x}_a > b(S)\bar{y}_a\}, r(S) = U - b(S),\ C^+ = \{a \in \delta^+(S): \bar{x}_a > r(S)\bar{y}_a\}$ and if $\sum_{a\in C^-} b(S)\bar{y}_a < b(S) + \sum_{a\in C^+}\{\bar{x}_a - r(S)\bar{y}_a\} - 0.015$, then the inequality

$$\sum_{a\in\delta^-(S)\setminus C^-} x_a + \sum_{a\in C^-} b(S)y_a \geq b(S) + \sum_{a\in C^+}\{x_a - r(S)y_a\}$$

is added to the cut pool.

If a cut is added to the cut pool the frequency parameter is increased by 1. Otherwise, decrease the frequency parameter by 1 and look at the next set.

Whenever the frequency parameter of a set is less than -3, the set is declared inactive. Once, all the active sets have been visited, the cuts generated are added into the matrix.

- Reoptimization. If violated inequalities have been found, the linear program is reoptimized. If the number of passes is less than the maximum, go to the next pass. Otherwise go to the enumeration phase.

6.2.3. Primal heuristics

There are two heuristic algorithms to find good feasible solutions

- Slope scaling [10]

Dynamic slope scaling procedure:

Step 1: Set $x^0 = 0, k = 1$, and $\pi_j^1 = c_j$, for $j = 1,2, \dots, n$. Go to step 2

Step 2: Let $x^k = \text{argmin}\{x^T\pi^k : Ax = b, x_j \geq 0, \forall j\}$. If $\|x^k - x^{k-1}\| \leq \epsilon$, stop and x^k is an approximate solution to the problem. Otherwise, go to step 3.

Step 3: Set $\pi_j^{k+1} = \begin{cases} c_j + {f_j}/{x_j^k} & \text{if } x_j^k > 0 \\ \pi_j^k & \text{if } x_j^k = 0 \end{cases}$ and $k = k + 1$. Go to Step 1 .

This algorithm is based on the idea that there exists a linear program,

$$\big(p(\bar{c})\big) \min\left\{\sum_{(i,j)\in\mathcal{A}} \bar{c}_{ij}\, x_{ij} : \sum_{j\in V_i^-} x_{ij} - \sum_{j\in V_i^+} x_{ij} = b_i \text{ for all } i \in V, 0 \leq x_{ij} \leq U, \forall (i,j) \in \mathcal{A}\right\}$$

that has the same optimal solution as the original mixed integer problem, or in other words, that there exist $\bar{c}$ such that $v(UFCNFP) = v(p(\bar{c}))$.

To find such a $\bar{c}$, a sequence $\{\bar{c}^k\}_{k=1}^k$ of slope are constructed, such that $\bar{c}^k$ is not far $\bar{c}$. Let x^k be optimal solution of $p(\bar{c}^k)$. The slope at iteration k+1 is computed as follows:

$$\bar{c}_{ij}^{k+1} = \begin{cases} c_{ij} + \dfrac{f_{ij}}{x_{ij}^k} & \text{if } x_{ij}^k > 0 \\ g(x^k, x^{k-1}, \dots, x^1) & \text{otherwise} \end{cases}$$

where,c and f are the original variable and fixed costs, and $g()$is a function that depends on the solutions of the previous iterations. In our implementation, the first objective function is computed from the solution obtained after the cut generation phase. Indeed, let $(\bar{x}, \bar{y})$ be such a solution. The first objective function is given by:

$$\bar{c}_{ij}^l = \begin{cases} c_{ij} + \dfrac{f_{ij}}{\bar{x}_{ij}} & \text{if } \bar{x}_{ij} > 0 \\ c_{ij} + \dfrac{f_{ij}}{U} & \text{otherwise} \end{cases}$$

Then the iteration k+1 we define the cost function as follows:

$$\bar{c}_{ij}^{k+1} = \begin{cases} c_{ij} + \dfrac{f_{ij}}{x_{ij}^{k}} & \text{if } x_{ij}^{k} > 0 \\ \lambda \bar{c}_{ij}^{r} + (1-\lambda)(c_{ij} + \dfrac{f_{ij}}{U}) & \text{otherwise} \end{cases}$$

Where $\lambda \in (0,1)$ and $r \in \{1,2,\dots,k-1\}$ is the last iteration in which $x_{ij}^{r} > 0$ and the cost assigned was $\bar{c}_{ij}^{r}$.

If $x^{k+1} = x^{k}$, then stop. Otherwise, go to the next iteration. If the maximum number of iterations is attained, we stop. If no solution has been found, we apply a rounding heuristic. Let $\mathcal{A}' = \{a \in \mathcal{A}: \bar{y}_a > 0\}$. Find a feasible flow x^* using just the arcs of $\mathcal{A}'$.

Set $y_a^* = 1$ if $x_a^* > 0$. (x^*, y^*) is the heuristic solution.

- Min cost Flow.

Here we solve a minimum cost flow problem on the graph defined by the open arcs of the current fractional solution. In other words, given a fractional solution $(\bar{x}, \bar{y})$, we define the graph $G' = (V, \mathcal{A}')$ where $\mathcal{A}' = \{a \in \mathcal{A}: \bar{y}_a > 0\}$. The objective function is defined as:

$$\bar{c}_{ij}(\bar{x}) = \begin{cases} c_{ij} + \dfrac{f_{ij}}{\bar{x}_{ij}} & \bar{x}_{ij} > 0 \\ c_{ij} + \dfrac{f_{ij}}{u_{ij}} & \text{otherwise} \end{cases} \quad \text{for all } (i,j) \in \mathcal{A}'$$

The solution to this problem is obtained with the network simplex implementation MCF [11]. The basis is stored and reused as initial basis in the next call. The slope scaling procedure is called at the top node, whereas the Min cost flow heuristic is called at 10 consecutive nodes every 100 nodes in the branch-and-bound tree.

6.2.4. Branching rule

Whereas many specialized branch-and-cut codes use simple variable branching rules such as most fractional, most costly, etc, commercial MIP systems use pseudo-costs based on dual variable estimates. As UFCNFP falls somewhat between the general and the special purpose, we have attempted to compare some of the simple branching rules.

Below, we briefly present the variable and constraint branching rules that we have tried.

- Variable branching. The rules we have studied are :
  - ✓ Most fractional. Branch on the variable that maximizes $w_a = \max\{\bar{y}_a, 1 - \bar{y}_a\}$ over the set of arcs $a \in \mathcal{A}$ such that $\bar{y}_a \in (0,1)$.
  - ✓ Least fractional. This criteria is similar to the previous one. Branch on the variable that minimizes $w_a = \max\{\bar{y}_a, 1 - \bar{y}_a\}$.
  - ✓ Maximum fixed charge. Branch on the variable for which the fixed cost fa is maximum among those with $\bar{y}_a$ fractional.
  - ✓ Maximum remaining fixed charge. Branch on the variable that maximizes $w_a = (1 - \bar{y}_a). f_a$ among those $\bar{y}_a$with fractional.
  - ✓ 2-strong branching (2-st). Select the two arcs,a^1, a^2 with the biggest and second biggest fixed charge. Then compute (using five iterations of dual simplex) a lower bound of their possible offsprings, namely, z_0^1, z_1^0 for a^1 and z_0^2, z_1^2 for $a^2$when the variable is fixed to zero and one respectively. Then, compute $w^1 = \min\{z_0^1, z_1^1\}$ and $w^2 = \min\{z_0^2, z_1^2\}$ and branch on the variable that maximizes w^i.
- Constraint Branching. Branching can also be based on linear inequalities. Here we consider the possibility of branching on subtour constraints, motivated by the fact that optimal solutions of uncapacitated problems do not contain cycles.

These inequalities have the following form:

$$\sum_{a\in} y_a \leq |S| - 1 \ \forall S \subset V.$$

Given the solution at the current node, determine a fractional subtour, i.e. a set S for which there exists at least one arc $a \in E(S)$ with $\bar{y}_a$ fractional. Then

- ✓ If the subtour is encountered for the first time, compute its value, i.e $l(S) = \sum_{a\in E(S)} \bar{y}_a$. If this value is less than $|S| - 1$ and greater than or equal to one, then branching constraints chosen are $\sum_{a\in E(S)} y_a \leq \lfloor l(S) \rfloor$ and $\sum_{a\in E(S)} y_a \geq \lceil l(S) \rceil$. The node to be solved in the next iteration is one of the successors of the current node.
- ✓ Otherwise, if the subtour has already been used, select the most fractional variable in $E(S)$ to branch on.

6.2.5. Pruning criteria and variable fixing

When the variable and fixed costs are non negative, there exists an optimal solution that is cycle free. This allow us to prune some nodes of the enumeration tree, and also provides a test allowing us to fix an arc variable y_a to zero, if arc a plus the arcs fixed to one form a cycle. Both these tests are carried out before solving the linear program at each node of the enumeration tree.

6.3. Greedy (heuristic) algorithm for UFL

There are different exact and heuristic methods to solve the UFL problem. Here we will see the well known heuristic approach called greedy algorithm. We start an empty set S of open facilities, and each step we add to S a site $j \in J \setminus S$ that yields the maximum improvement in the objective value. For a set, $S \subseteq J$, of open facilities, the objective value is given by

$$Z(S) = \sum_{i \in I} \max_{j \in S}\{c_{ij}\} - \sum_{j \in S} f_j$$

For a site $j \in J \setminus S$. Let $P_j(S) = Z(S \cup \{j\}) - Z(S)$ denote the change in the objective value when a new facility is opened at j. For the currently open set of facilities S and for each client $i \in I$, define $u_i(S)$ to be $\max_{j \in S}\{c_{ij}\}$; define $u_i(\emptyset) = 0$.That is $u_i(S)$ is the maximum profit obtained from serving client i using only the facilities in S. Then

$$Z(S) = \sum_{i \in I} \max_{j \in S}\{c_{ij}\} - \sum_{j \in S} f_j$$

$$Z(S) = \sum_{i \in I} u_i(S) - \sum_{j \in J} f_j$$

and

$$P_j(S) = Z(S \cup \{j\}) - Z(S) = \sum_{i \in I} \left(c_{ij} - u_i(S)\right)^+ - f_j$$

In each iteration of the greedy heuristic, we compute $P_j(S)$ for each $j \in J \setminus S$. If either $J \setminus S$ is empty or $P_j(S) \leq 0$ for each $j \in J \setminus S$, then we terminate the heuristic. Otherwise, we add to S a $j \in J \setminus S$ whose incremental value $P_j(S)$ is maximum.

Consider the first condensed dual of (SLPR), (CD1), and for each $i \in I$. Let the i^{th} dual variable u_i be assigned the value $u_i(S)$ defined above. Then the dual objective value corresponding to S is

$$w\big(u(S)\big) = \sum_{i \in I} u_i(S) + \sum_{j \in J} \left(\sum_{i \in I} \big(c_{ij} - u_i\big)^+ - f_j \right)^+$$
$$= \sum_{i \in I} u_i(S) + \sum_{j \in S} \big(P_j(S)\big)^+,$$

because each $j \in S$ has $\sum_{i \in I} \big(c_{ij} - u_i\big)^+ - f_j = P_j(S)$.

For each final solution S^G found by the greedy heuristic, each $j \in S \setminus S^G$ has $P_j(S^G) \leq 0$, hence the dual objective value is $w(u(S^G)) = \sum_{i \in I} u_i(S^G)$.

Algorithm 6.3: Greedy algorithm (heuristic) for UFL problem

Initialization: $S_0 = \emptyset, t = 1$

Iteration t:

Step1: Compute the objective value $Z(S) = \sum_{i \in I} u_i(S) - \sum_{j \in S} f_j$

Step 2: Compute the change in the objective value

$$P_j(S) = Z(S \cup \{j\}) - Z(S) = \sum_{i \in I} \big(c_{ij} - u_i(S)\big)^+ - f_j \ , \forall j \in J \setminus S$$

Step 3: If $P_j(S) > 0$ is maximum, set $S_j := \{j\}$

Step 4: If $P_j(S) \leq 0$ for each $j \in J \setminus S$, then stop. Otherwise go to step 1.

The greedy (heuristic) algorithm give a candidate solution to the UFLP problem instance as well as feasible solution for the dual of the strong LP relaxation.

Example 6.1: Solve the uncapacitated facility location problem by greedy algorithm given by the following

$$m = 4\,, n = 6$$
$$f = [\ 3\ \ 2\ \ 2\ \ 2\ \ 3\ \ 3\ ]$$
$$c = \begin{bmatrix} 6 & 6 & 8 & 6 & 0 & 6 \\ 6 & 8 & 6 & 0 & 6 & 6 \\ 5 & 0 & 3 & 6 & 3 & 0 \\ 2 & 3 & 0 & 2 & 4 & 4 \end{bmatrix}$$

Solution:

Iteration 1:

$$S_0 = \emptyset, \qquad Z(S_0) = \sum_{i\in I} u_i(S) - \sum_{j\in S} f_j = \sum_{i\in I} u_i(S_0) - \sum_{j\in S_0} f_j = 0$$
$$u(S) = [\ 0 \quad 0 \quad 0 \quad 0\ ]$$

$$P_1(S_0) = \sum_{i\in I}\left(c_{ij} - u_i(S_0)\right)^+ - f_j = c_{11} + c_{21} + c_{31} + c_{41} - f_1 = 6 + 6 + 5 + 2 - 3 = 16$$
$$P_2(S_0) = c_{12} + c_{22} + c_{32} + c_{42} - f_2 = 6 + 8 + 0 + 3 - 2 = 15$$
$$P_3(S_0) = c_{13} + c_{23} + c_{33} + c_{43} - f_3 = 8 + 6 + 3 + 0 - 2 = 15$$
$$P_4(S_0) = c_{14} + c_{24} + c_{34} + c_{44} - f_4 = 6 + 0 + 6 + 2 - 2 = 12$$
$$P_5(S_0) = c_{15} + c_{25} + c_{35} + c_{45} - f_5 = 0 + 6 + 3 + 4 - 3 = 10$$
$$p_6(S_0) = c_{16} + c_{26} + c_{36} + c_{46} - f_6 = 6 + 6 + 0 + 4 - 3 = 13$$
$$w(S_0) = \sum_{i\in I} u_i(S_0) + \sum_{j\in J}\left(\sum_{i\in I}\left(c_{ij} - u_i(S_0)\right)^+ - f_j\right)^+$$
$$= (0) + (16 + 15 + 15 + 12 + 10 + 13) = 81$$

Set $S_1 \coloneqq \{1\}$, since $P_1(S_0) > 0$ is maximum.

Iteration 2:

$$S_1 = \{1\}, \qquad Z(S_1) = \sum_{i\in I} u_i(S) - \sum_{j\in S} f_j = \sum_{i\in I} u_i(S_1) - \sum_{j\in s_1} f_j$$

$$= u_1(1) + u_2(1) + u_3(1) + u_4(1) - f_1$$

$$= 6 + 6 + 5 + 2 - 3 = 16$$

$$u(S_1) = [\ 6 \quad 6 \quad 5 \quad 2\ ]$$

$$P_2(S_1) = \sum_{i\in I} \left(c_{ij} - u_i(S_1)\right)^+ - f_j$$

$$= c_{12} - u_1(S_1) + c_{22} - u_2(S_1) + c_{32} - u_3(S_1) + c_{42} - u_4(S_1) - f_2$$

$$= (6-6) + (8-6) + (0-5) + (3-2) - 2 = (2+1) - 2 = 1$$

$$P_3(S_1) = c_{13} - u_1(S_1) + c_{23} - u_2(S_1) + c_{33} - u_3(S_1) + c_{43} - u_4(S_1) - f_3$$

$$= (8-6) + (6-6) + (3-5) + (0-2) - 2 = (2) - 2 = 0$$

$$P_4(S_1) = c_{14} - u_1(S_1) + c_{24} - u_2(S_1) + c_{34} - u_3(S_1) + c_{44} - u_4(S_1) - f_4$$

$$= (6-6) + (0-6) + (6-5) + (2-2) - 2 = (1) - 2 = -1$$

$$P_5(S_1) = c_{15} - u_1(S_1) + c_{25} - u_2(S_1) + c_{35} - u_3(S_1) + c_{45} - u_4(S_1) - f_5$$

$$= (0-6) + (6-6) + (3-5) + (4-2) - 3 = (2) - 3 = -1$$

$$P_6(S_1) = c_{16} - u_1(S_1) + c_{26} - u_2(S_1) + c_{36} - u_3(S_1) + c_{46} - u_4(S_1) - f_6$$

$$= (6-6) + (6-6) + (0-5) + (4-2) - 3 = (2) - 3 = -1$$

$$w\big(u(S_1)\big) = (6+6+5+2) + (1) = 20$$

Now set $S_2 \coloneqq \{1,2\}$, since $P_2(S_1) > 0$ is maximum

Iteration 3:

$$S_2 = \{1,2\}\,, Z(S_2) = \sum_{i\in I} u_i(S_2) - \sum_{j\in S_2} f_j$$

$$= u_1(S_2) + u_2(S_2) + u_3(S_2) + u_4(S_2) - (f_1 + f_2)$$

where

$$u_1(S_2) = \max\{c_{11}, c_{12}\} = 6$$
$$u_2(S_2) = \max\{c_{12}, c_{22}\} = 8$$
$$u_3(S_2) = \max\{c_{13}, c_{23}\} = 5$$
$$u_4(S_2) = \max\{c_{14}, c_{24}\} = 3$$

Thus $Z(S_2) = (6 + 8 + 5 + 3) - (3 + 2) = 17$

and

$$u(S_2) = [\ 6 \quad 8 \quad 5 \quad 3\ ]$$

$$P_3(S_2) = c_{13} - u_1(S_2) + c_{23} - u_2(S_2) + c_{33} - u_3(S_2) + c_{43} - u_4(S_2) - f_3$$
$$= (8 - 6) + (6 - 8) + (3 - 5) + (0 - 3) - 2 = (2) - 2 = -1$$
$$P_4(S_2) = c_{14} - u_1(S_2) + c_{24} - u_2(S_2) + c_{34} - u_3(S_2) + c_{44} - u_4(S_2) - f_4$$
$$= (0 - 6) + (6 - 8) + (3 - 5) + (4 - 3) - 2 = (1) - 2 = -1$$

$$P_5(S_2) = c_{15} - u_1(S_2) + c_{25} - u_2(S_2) + c_{35} - u_3(S_2) + c_{45} - u_4(S_2) - f_5$$

$$= (0 - 6) + (6 - 8) + (3 - 5) + (4 - 3) - 3 = (1) - 3 = -2$$

$$P_6(S_2) = c_{16} - u_1(S_2) + c_{26} - u_2(S_2) + c_{36} - u_3(S_2) + c_{46} - u_4(S_2) - f_6$$

$$= (6 - 6) + (6 - 8) + (0 - 5) + (4 - 3) - 3 = (1) - 3 = -2$$

$$w\big(u(S_2)\big) = (6 + 8 + 5 + 3) + (0) = 22$$

Now we have to stop because $P_j(S_2) \leq 0, \forall j \in J \setminus S_2$. The greedy solution is $S^G = \{1,2\}$ with objective value $Z^G = Z(S_2) = 17$. The dual greedy value W^G is the best upper bound computed on the optimal value $w^G = w\big(u(S_1)\big) = 20$. Thus opening facilities at sites 1 and 2 gives an objective value of 19. i.e. assign client 1 to site 2, client 2 to site 1, client 3 to site 1, and client 4 to site 2 to obtain an optimal value 0f 17.

Example 6.2: Solve the following uncapacitated facility location problem with the given data

$$m = 6\,, \qquad n = 5$$

$$c = c_{ij} = \begin{pmatrix} 12 & 13 & 6 & 0 & 1 \\ 8 & 4 & 9 & 1 & 2 \\ 2 & 6 & 6 & 0 & 1 \\ 3 & 5 & 2 & 10 & 8 \\ 8 & 0 & 5 & 10 & 8 \\ 2 & 0 & 3 & 4 & 1 \end{pmatrix}$$

$$f = f_j = (4 \quad 3 \quad 4 \quad 4 \quad 7)$$

Solution:

Iteration1:

$$S_0 = \emptyset\,, Z(S_0) = 0\,, u(S_0) = [\,0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\,]$$

$$P_1(S_0) = 31\,, P_2(S_0) = 25\,, P_3(S_0) = 27\,, P_4(S_0) = 21\,, P_5(S_0) = 14$$

$$w(S_0) = (0) + (31 + 25 + 27 + 21 + 14) = 118$$

Now we set $S_1 := \{1\}$, since $P_1(S_0) = 31 > 0$ is maximum

Iteration 2:

$$S_1 = \{1\}, Z(S_1) = 31\,, u(S_1) = [\,12 \quad 8 \quad 2 \quad 3 \quad 8 \quad 2\,]$$

$$P_2(S_1) = 4\,, P_3(S_1) = 2, P_4(S_1) = 7, P_5(S_1) = -2$$

$$w\big(u(S_1)\big) = (12 + 8 + 12 + 3 + 8 + 2) + (4 + 2 + 7) = 48$$

Now we set $S_4 := \{1,4\}$ because $P_4(S_1) > 0$ is maximum.

Iteration 3:

$$S_4 = \{1,4\}, \qquad Z(S_4) = 38, \qquad u(S_4) = [\,12 \quad 8 \quad 2 \quad 10 \quad 10 \quad 4\,]$$

$$P_2(S_4) = 2, P_3(S_4) = 1, P_5(S_4) = -7$$

We set $S_2 := \{1,2,4\}$, since $P_2(S_4) > 0$ is maximum.

Iteration 4:

$$S_2 = \{1,2,4\}, \qquad Z(S_2) = 40\,, \qquad u(S_2) = [\,13 \quad 8 \quad 6 \quad 10 \quad 10 \quad 4\,]$$

$$P_3(S_2) = -2, P_5(S_2) = -7$$

we have to stop, because $P_j(S_2) \leq 0, \forall j \in J \setminus S_2$.

The greedy solution is $S^G = \{1,2,4\}$ with objective value:

$$Z^G = z(S_2) = (13 + 8 + 6 + 10 + 10 + 4) - (4 + 3 + 4) = 40$$

Thus opening facilities at sites 1,2, and 4 gives an objective value of 40.

7. Concluding remarks

Fixed charge network flow problem is to find a minimum cost arc combination that provides flows from the supply nodes to the demand nodes. Associated with all arcs are two costs: the fixed charge of using the arc and the variable cost depending on the amount of flow the arc actually carries.

The fixed charge network flow problem can be capacitated or uncapacitated depending on the capacity of each arc $(i, j) \in \mathcal{A}$. That is if the arc capacity is ∞ (unlimited) we call the problem uncapacitated fixed charge network flow problem; otherwise we say that it is capacitated fixed charge network flow problem.

There are many problems which arise as a special case of fixed charge network flow problem some of them are:

- Fixed charge transportation problem
- Uncapacitated facility location problem
- Steiner tree problem
- Uncapacitated lot-sizing problem, etc…

To solve the fixed charge network flow problem, there are two methods. These are exact method and heuristic approach. Branch and cut algorithm solve the single commodity UFCNFP which includes the Steiner tree problem, uncapacitated lot-sizing problems, and the fixed charge transportation problem as special cases.

Uncapacitated facility location problem is a special case of the fixed charge network flow problem. We can use greedy algorithm to find the heuristic solution of this problem. The algorithm is used to find a candidate solution for UFLP problem instance as well as a feasible solution for every feasible solution for the dual of the strong LP relaxation.

References:

[1] A.V. Cabot, S.S.Erenguc,*Some branch-and-bound procedures for fixed-cost transportation problems*, Nav. Res.Logistics Q. 31, 145-154,1984.

[2] Barr R.S., Glover F., and Klingman D, *A new optimization method for large scale fixed charge transportation problems*, Operations Research, 29(3):448–463, 1981.

[3] Costa, A.M., *A survey on benders decomposition applied to fixed-charge network design problems*,Comput, Oper. Res. 32(6), 1429–1450 ,2005

[4] Eksioglu, B., Duni-Eksioglu, S., Pardalos, P.M, Equilibrium problems and Variational models. In:Maugeri, A., Giannesi, F., Pardalos, P.M. (eds.) ,*Solving large-scale fixed charge network flow problems*. Kluwer, Dordrecht ,2001.

[5] G.L. Nemhauser and L.A. *Wolsey, Integer and Combinatorial Optimization*, John Wiley & sons, Chichester, 1988, Wiley Interscience Series in Discrete Mathematics and Optimization.

[6] Francisco Ortega and Laurence Wolsey CORE and INMA, *A branch-and-cut algorithm for the single commodity uncapacitated fixed charge network flow problem*, Universit´e Catholique de Louvain,2000.

[7] George L. Nemhauser, *Combining exact and heuristic approaches for the capacitated fixed charge network flow problem*.

[8] K.G. Murty, *Solving the fixed-charge problem by ranking the extreme points*, Oper. Res. 16, 268-279, 1968

[9] K.G. Murty, *Linear and combinatorial programming*, Wiley,Newyork,1976

[10] Kim D. and Pardalos P., *A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure*, Operations Research Letters, 24:195–203,1998.

[11] L¨obel A. MCF: *a network simplex implementation*. Konrad-Zuse-Zentrum f¨ur Informationstechnik Berlin, http://www.zib.de/Optimization/Software/Mcf/, 2000.

[12] Mehlhorn K., N¨aher S., Seel M., and Uhrig C,The LEDA User Manual, Version 4.1. http://www.mpisb.mpg.de/LEDA/MANUAL/MANUAL.html, 2000.

[14] M. L. Balinski, *Fixed cost transportation problem*, Naval Res. Log. Quart. 8, 41-54 ,1961.

[13] Tuy, H., *Strong polynomial time solvability of minimum concave cost network flow problem*. ACTA, Math. Vietnam. 25(2), 209–218, 2000.