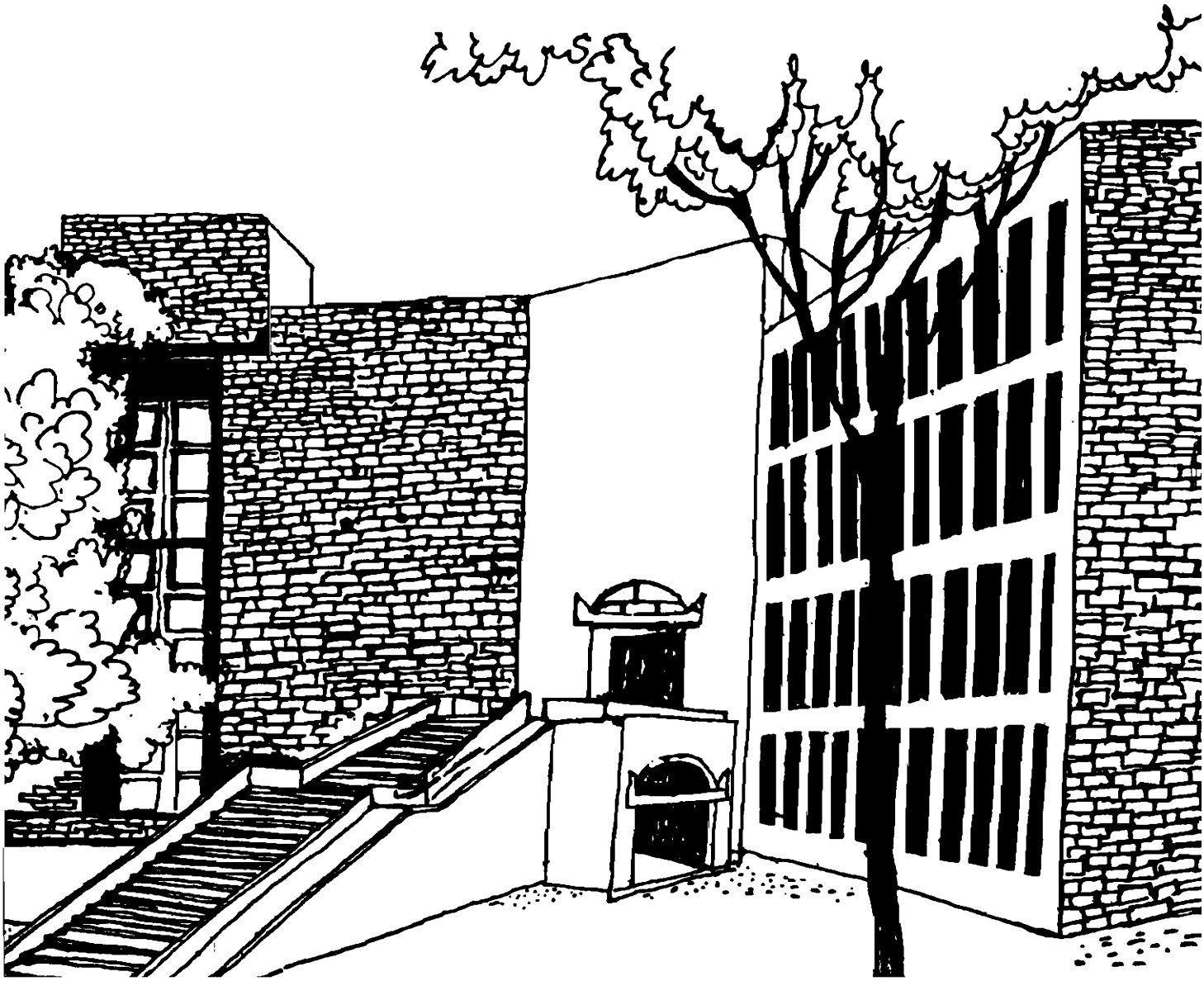
विद्याविनियोगाद्विकासः
IIM
AHMEDABAD

Working Paper



ONE AND TWO FACILITY NETWORK DESIGN REVISITED

By

Trilochan Sastry

W.P. No. 98-08-05
August 1998 1466

INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD - 380 015
INDIA

Abstract

The one facility one commodity network design problem $OFOC$ with flow costs considers the problem of sending d units of flow from a source to a destination where capacity is purchased in batches of C units. The two commodity problem $TFOC$ is similar, but capacity can be purchased either in batches of C units or one unit. Flow costs are zero. These problems are known to be NP-hard. We describe an exact $O(n^3 3^n)$ algorithm for these problems based on the repeated use of a bipartite matching algorithm. We also present a better lower bound of $\Omega(n^{2k^*})$ for an earlier $\Omega(n^{2k})$ algorithm described in the literature where $k = \lfloor d/C \rfloor$ and $k^* = \min\{k, \lfloor (n-2)/2 \rfloor\}$. The matching algorithm is faster than this one for $k \geq \lfloor (n-2)/2 \rfloor$. We next consider an extended formulation of the problem, describe an efficient heuristic based on this formulation, and use it to show that for problems with up to five nodes, the formulation guarantees integer optimal solutions. We also give an example of a six node graph for which the extended formulation has a fractional solution. Finally, we provide another reformulation of the problem that is *quasi integral*, i.e., every edge of the integer polytope is an edge of the polytope associated with the linear programming relaxation of the reformulation. This property could be useful in designing a modified version of the simplex method to solve the problem using a sequence of pivots with integer extreme solutions, referred to as the *integral simplex method* in the literature.

Key words and phrases: network design, exact algorithm, integer solutions, quasi-integral.

1 Introduction

In this paper we consider the one-facility, one-commodity ($OFOC$) and the two facility, one commodity ($TFOC$) network design problems. The problem OFOC can be stated as follows. Consider a directed graph $G = (V, A)$ with a source s and a destination t. Capacity on each arc can be purchased in integer multiples of C units with each batch of C units costing $w_a \geq 0$ on arc a. There is a flow cost of $p_a \geq 0$ per unit of flow on arc a. The total cost of the flow is the flow cost plus the cost of purchasing capacity. The objective is to design a minimum cost network to send d units of flow from s to t. The two-facility one-commodity network design problem ($TFOC$) is similar to $OFOC$ where we assume that capacity can be purchased either in batches of size 1 at a cost of $w_a^1 \geq 0$ or size C at a cost of $w_a^2 \geq 0$, and that flow costs are zero.

A more general form of these problems with several sources and sinks arises in the telecommunications and transportation industry. The problem $TFOC$ has been studied by Magnanti and Mirchandani(1993). They give an inequality description and report computational results for the problem. The multi commodity problem has been considered by Magnanti, Mirchandani, and Vachani(1995) and Chopra et al. (1995). Chopra, Gilboa and Sastry (1997) show the problems $OFOC$ and $TFOC$ to be NP-hard. They use results characterizing optimal solutions to $OFOC$ and $TFOC$ to describe an exact algorithm to solve $OFOC$ and $TFOC$ based on finding a shortest path on an auxiliary graph with $O(n^{2\lfloor d/C \rfloor})$ nodes. They also obtain extended formulations in each case and report computational results based on these formulations.

We describe an integer programming formulation for $OFOC$. For each arc (i, j), let f_{ij} be the flow and y_{ij} the batches of capacity installed (each batch provides C units of capacity). Let $d = kC + r$ for some integer k and $0 < r \leq C$, where we define $r = C$ if d is a multiple of C. $OFOC$ can be formulated as follows:

$$\text{Min} \sum_{(i,j)\in A} w_{ij} y_{ij} + \sum_{(i,j)\in A} p_{ij} f_{ij}$$

s.t.

$$\sum_j f_{ji} - \sum_j f_{ij} = \begin{cases} -kC - r & \text{for i = s} \\ kC + r & \text{for i = t} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$Cy_{ij} - f_{ij} \geq 0 \quad (2)$$

$$y, f \geq 0; y \text{ integer.}$$

The problem $TFOC$ has a similar description (see also Magnanti and Mirchandani (1993)). Let y_{ij}^1 and y_{ij}^2 denote the units of facility 1 and 2 respectively installed on arc (i, j).

$$\text{Min} \sum_{(i,j)\in A} (w_{ij}^1 y_{ij}^1 + w_{ij}^2 y_{ij}^2)$$

s.t.

$$\sum_j f_{ji} - \sum_j f_{ij} = \begin{cases} -kC - r & \text{for i = s} \\ kC + r & \text{for i = t} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

$$y_{ij}^1 + Cy_{ij}^2 - f_{ij} \geq 0 \quad (4)$$

$$y, f \geq 0; y \text{ integer.}$$

The assumptions that $w_{ij} \geq 0$ for $OFOC$ and $w_{ij}^1 \geq 0$, $w_{ij}^2 \geq 0$ for $TFOC$ for all arcs (i, j) are not restrictive. Otherwise, the problem is unbounded since we can set y_{ij} or y_{ij}^1, y_{ij}^2 to an arbitrarily large value. In the next section, we describe an exact $O(n^3 3^n)$ algorithm for the problem which performs better than the algorithm described by Chopra, Gilboa and Sastry (1997) for large k. We also show that we can bound the quantity of flow that goes on a shortest path from source to destination, and use it to derive a better lower bound on the algorithm described by them. In Section 3 we use an extended formulation of the problems to describe a dual based efficient heuristic. We then derive conditions under which the dual heuristic and the linear programming relaxation of the extended formulation guarantee integer optimal solutions. We also prove that for problems with at most five nodes, the formulation guarantees integer optimal solutions and give an example of a six node problem with a fractional solution. In Section 4 we describe a *quasi integral* polytope for $OFOC$ and $TFOC$ such that every edge of the integer polytope is an edge of the quasi integral polytope.

2 An Exact Algorithm

We use a result of Chopra, Gilboa and Sastry (1997) characterising optimal solutions of $OFOC$ and $TFOC$ to derive an algorithm for these problems. Consider an optimal solution vector (y^*, f^*) where y_a^* is the capacity installed on arc a and f_a^* is the flow through arc a. Note that since $w_a \geq 0$ and $p_a \geq 0$ for $OFOC$, given the flow vector f^* in any optimal solution, the optimal capacity installed can be assumed to be given by $y_a^* = \lceil f_a^*/C \rceil$. Similarly, since $w_a^1 \geq 0$, $w_a^2 \geq 0$ for $TFOC$, we can assume that $y_a^{1*} + Cy_a^{2*} = f_a^*$ if $y_a^{1*} > 0$ and $y_a^{2*} = \lceil f_a^*/C \rceil$ if $y_a^{1*} = 0$. Given a solution (y^*, f^*), define an arc a with $f_a^* < Cy_a^*$ to be a *free arc*, and any connected set of free arcs (ignoring direction) without cycles as a *free path.*

Theorem 1 (Chopra, Gilboa and Sastry (1997)) *There exists an optimal solution to $OFOC$ and $TFOC$ with exactly one free path from source s to sink t, such that free arcs directed forward along the path have a flow from $\{lC + r\}_{l=0}^{k}$, and those directed backward have a flow from $\{lC - r\}_{l=1}^{k}$. Further, all other flows equal ρC for some integer $\rho \geq 0$.*

Hereafter, we only consider optimal solutions with one free path as described by Theorem 1. Any node i on the free path with a free forward and reverse arc leaving it has at least C units leaving it, and hence, must have at least C units entering it. Label node i as an *in turn* node. Similarly, any node i with a forward and a reverse arc entering it has at least C units leaving it, and is an *out turn* node. In addition, we define s to be an out turn node, and t to be an in turn node. Notice that there are an equal number of in turn and out turn nodes. Other than flows on the free path, there are k units of *full flow* where each unit of full flow corresponds to C units. Thus we can consider the optimal solution to consist of k units of full flow leaving node s, and each such full flow either enters an in turn node $i \neq t$, or directly flows to node t. Similarly, k units of full flow enter node t, where each unit either starts at some out turn node $i \neq s$, or directly starts at node s. Since we count the full flows separately, we consider the free path to have exactly r units on forward arcs and $C - r$ units on reverse arcs. Since arc costs are nonnegative and since there is one free path with r units leaving node s and entering node t, we assume that no reverse arcs enter or leave nodes s and t.

Example 1.

Consider Figure 1 which has 6 nodes. If $s-1-2-3-4-t$ is the free path, then $s, 1, 3$ are out turn nodes and $2, 4, t$ are in turn nodes. If $k = 3$, then 3 units of full flow start at node s and end at each of the nodes $2, 4, t$. Further, 1 unit of full flow starts at each of the nodes $1, 3$, and ends at node t. Forward arcs $(s,1),(2,3),(4,t)$ have a flow of r units each, and reverse arcs $(2,1),(4,3)$ have a flow of $C-r$ units each.

□

INSERT FIGURE 1 HERE

Using the structure of the optimal solution, and the number of turn nodes, we show that we can bound the amount of full flow that flows directly from node s to node t. Let $k^* = \min\{k, \lfloor (n-2)/2 \rfloor\}$, and let $a(i,j)$, $b(i,j)$ and $c(i,j)$ be the length of the shortest distance between nodes i and j using $w_{uv} + Cp_{uv}$, $w_{uv} + rp_{uv}$ and $w_{uv} + (C-r)p_{uv}$ as arc costs for $OFOC$, and w^2_{uv}, min $\{rw^1_{uv},\ w^2_{uv}\}$, and min $\{(C-r)w^1_{uv},\ w^2_{uv}\}$ as arc costs for $TFOC$. Let the shortest paths between nodes i and j of length $a(i,j), b(i,j), c(i,j)$ be the α, β and δ paths. Assume that there is a path from s to every node j and from every node j to node t. Otherwise node j can be deleted from the graph. We also assume that $k \geq 1$: otherwise, the problem reduces to finding the shortest β path of length $b(s,t)$ without any turn nodes.

Lemma 1 *There exists an optimal solution to $OFOC$ and $TFOC$ such that at least $(k-k^*)C$ units flow directly from s to t on the shortest α path of length $a(s,t)$.*

Proof

Restrict attention to optimal solutions with exactly one free path. The number of in turn nodes and out turn nodes are equal, and hence, excluding nodes s and t, there are at most $\lfloor (n-2)/2 \rfloor$ in turn nodes on the free path. Therefore, at most min $\{k,\ \lfloor (n-2)/2 \rfloor\}$ units of full flow leaving s enter these in turn nodes, at which they split into two flows of r and $C-r$. The same number of full flows leave the out turn nodes and enter node t. The remaining units of full flow can therefore go directly from node s to node t on the shortest path.

□

Thus, if $k \geq 2$ in Figure 1, then $k-2$ units of full flow can be sent directly from node s to node t. Chopra, Gilboa and Sastry describe an exact algorithm for the problem which is a shortest path algorithm on an auxiliary graph with n^{2k+1} nodes, and hence takes $\Omega(n^{2k})$ time. We call this the *multiple path* algorithm.

Corollary 1 *The multiple path algorithm can be modified to solve OFOC and TFOC in $\Omega(n^{2k^*})$ iterations on an auxiliary graph with n^{2k^*+1} nodes.*

Proof
Lemma 1 implies that $(k-k^*)C$ units of flow can be sent directly from source to sink on the shortest path of length $a(s,t)$. The problem reduces to finding a solution for $k^*C + r$ units of flow, and hence the auxiliary graph needs to have only n^{2k^*} nodes.

□

We now show that if the free path is fixed, then the full flows can be obtained by solving a bipartite matching problem on an auxiliary graph. For any solution with exactly one free path, let I and O denote the set of in turn and out turn nodes excluding s and t, and let $k_I = \min\{k, |I|\}$. Create k_I copies each of nodes s and t, denoted by $s(1), \ldots, s(k_I)$ and $t(1), \ldots, t(k_I)$. Let all in turn nodes $I \cup \{t(1), \ldots t(k_I)\}$ form one partition of nodes, and let all out turn nodes $O \cup \{s(1), \ldots, s(k_I)\}$ form the other partition. There is a directed arc from each out turn node i to each in turn node j of cost $a(i,j)$. Let $H_{IO}(K)$ denote the bipartite graph with nodes $O \cup \{s(1), \ldots, s(k_I)\}$ and $I \cup \{t(1), \ldots, t(k_I)\}$, and arcs as defined. For the given set of in turn nodes $I \subset N$, and out turn nodes $O \subset N$, the *full matching* problem is to find the minimum cost perfect matching in the bipartite graph $H_{IO}(K)$. Let $K(I,O)$ be the cost of the optimal matching plus $(k-k_I) * a(s,t)$, which is the cost of sending the remaining $(k-k_I)$ units of full flow from s to t. Assume that if $(i,j) \notin A$, then $w_{ij} = \infty$ for $OFOC$ and $w^1_{ij} = w^2_{ij} = \infty$ for $TFOC$. Each arc (i,j) in any full matching in $H_{IO}(K)$ corresponds to a *full segment* of arcs on the original graph G, which connect out turn node i to in turn node j. We say that a matched arc (i,j) in $H_{IO}(K)$ *passes through* arc (u,v) in G if (u,v) is on the full segment corresponding to the matched arc. The total full flow on arc (u,v) equals the number of matched arcs in $H_{IO}(K)$ passing through it.

Lemma 2 *For a given free path, the optimal solution to $OFOC$ and $TFOC$ can be obtained by solving the full matching problem.*

Proof
Let $I \cup \{t\}$, $O \cup \{s\}$ denote the set of in turn and out turn nodes on the free path. Each arc $(i,j) \in H_{IO}(K)$ in the full matching corresponds to one unit of full flow leaving out turn node i and entering in turn node j in the original graph G. It is easy to verify that we get a feasible solution.

Consider any minimum cost solution to $OFOC$ or $TFOC$ for the given free path. Each in turn node $j \in I$ has one unit of full flow entering it which splits into two free flows of r and $C-r$ units after leaving j. Similarly, each out turn i node has one unit of full flow leaving it which is formed from two free flows of r and $C-r$ units entering it. If we consider nodes s and t to be split into nodes $s(1),\ldots,s(k_I)$ and $t(1),\ldots,t(k_I)$, then there is a one to one correspondence between nodes in the set $O \cup \{s(1),\ldots,s(k_I)\}$ and nodes on the set $I \cup \{t(1),\ldots,t(k_I)\}$ with one unit of full flow leaving a node in the first set and entering a node in the second set. Clearly, any full flow between an out turn node i and an in turn node j is on the shortest α path between i and j of cost $a(i,j)$. Therefore, these k_I units of full flow correspond to a full matching in $H_{IO}(K)$. The total cost $K(I,O)$ of the optimal full matching in $H_{IO}(K)$ therefore equals the cost of the full flows in G.

□

Next, we need to identify the optimal free path. Given a set of in turn nodes $I \cup \{s\}$, and out turn nodes $O \cup \{t\}$, we can number them in a natural order as we go from s to t along the free path. Forward arcs between an in turn node and the next out turn node form a *forward segment*, and reverse arcs between an out turn node and the next in turn node form a *reverse segment.* In addition, forward arcs between s and the first out turn node in O, and forward arcs between the last in turn node in I and t also form forward segments. Thus in Figure 1, if $s-1-2-3-4-t$ is the free path, then $(s,1),(2,3),(4,t)$ are forward segments and $(2,1),(4,3)$ are reverse segments. In this case the segments consist of one arc, but in general, each segment consists of several arcs. Consider bipartite graph $H_{IO}(F)$ induced by nodes $I \cup \{s\}$ and $O \cup \{t\}$, arcs between each in turn node and each out turn node, and arcs between s and each out turn node $j \neq s$, and between each in turn node $i \neq t$ and t. The length of each arc (i,j) in $H_{IO}(F)$ equals $b(i,j)$. The

forward matching problem is to find the minimum cost perfect matching on graph $H_{IO}(F)$. Let $F(I,O)$ denote the cost of the optimal forward matching. Notice that each arc (i,j) of length $b(i,j)$ in a forward matching in $H_{IO}(F)$ corresponds to a forward segment of arcs between out turn node i and in turn node j in graph G of the same length. The *reverse matching* problem is defined similarly on bipartite graph $H_{IO}(R)$ with an arc between each in turn node $i \neq t$ and out turn node $j \neq s$ of length $c(i,j)$. Let $R(I,O)$ denote the cost of the optimal reverse matching. Here again, each arc in any reverse matching in $H_{IO}(R)$ corresponds to a reverse segment in G.

The optimal forward and reverse matchings may not induce one free path, but may have a free path with a set of cycles as in Figure 2, which shows only the free arcs. These *free cycles* have alternate forward and reverse segments. The forward matching consists of $[s,1],[2,3],[4,5],[6,t]$ each carrying r units of flow, and the reverse matching consists of $[2,5],[4,3]$ and $[6,1]$, each carrying $C-r$ units of flow. Thus, there is a free cycle $2-5-4-3-2$, and a free path $s-1-6-t$ where arc $(6,1)$ is a reverse free arc. The solution to the full matching problem with $I=\{2,4,6\}$ and $O=\{1,3,5\}$ would however give a feasible solution since nodes $2,4$ and $3,5$ would each have C units of flow entering and leaving. If the reverse matching was $[2,1],[4,3],[6,5]$, we would have a free path with no cycles.

INSERT FIGURE 2 HERE

Lemma 3 *For a given set of in turn nodes I and out turn nodes O such that $|I|=|O|$ and $I \cap O = \Phi$, solving the full, forward and reverse matching problems gives a feasible solution to $OFOC$ or $TFOC$, with exactly one free path and perhaps some free cycles.*

Proof

Suppose we first solve the forward and reverse matching problems. We show that there is exactly one free path. Each node $i \in I$ has one free forward arc and one free reverse arc leaving it. Similarly, each node $i \in O$ has one free forward arc and one free reverse arc entering it. However, node s has exactly one free forward arc leaving it, and node t has exactly one free forward arc entering it. If we proceed from node s along free arcs, ignoring arc directions, we must eventually reach node t since each node on the path has degree two. If the free path does not visit all turn nodes, then the turn nodes not on the

free path form free cycles. If we now solve the full matching problem, it is easy to establish that we obtain a feasible solution.

□

Let the full, forward and reverse matchings comprise the *three matching* solution $(f,y)_{IO}$. Let $M(I,O) = K(I,O) + F(I,O) + R(I,O)$ denote the cost of the solution obtained by solving the three matching problems. Clearly, $M(I,O)$ equals the minimum cost for the given nodes I, O if the forward and reverse matchings form a connected path between s and t without any cycles. Otherwise, let $\hat{I} \subset I$, $\hat{O} \subset O$ denote the in turn and out turn nodes on the free path, and $I\backslash\hat{I}$, $O\backslash\hat{O}$ the nodes which form free cycles. The free cycles formed by nodes in $I\backslash\hat{I}$ and $O\backslash\hat{O}$ correspond to a total of $\rho \leq |I\backslash\hat{I}|$ units of full flow between s and t, where ρ is an integer equal to the total amount of full flow between s and in turn nodes $I\backslash\hat{I}$. For instance, in Figure 2, one unit of full flow may enter each of the nodes 2 and 4 from node s, and one unit of full flow leave each of the nodes 3 and 5 and enter node t. This corresponds to sending 2 units of full flow from s to t. Thus, these ρ units of full flow can be sent on the shortest path between s and t of length $a(s,t)$ without increasing cost. The remaining $|I\backslash\hat{I}| - \rho$ in turn nodes and $|O\backslash\hat{O}| - \rho$ out turn nodes correspond to feasible flows that are separate from the free path defined by $\hat{I},\hat{O}$. For instance, in Figure 2, if $\rho = 0$, then one unit of full flow may leave each of the nodes 3 and 5, and enter nodes 2 and 4. These feasible flows have nonnegative cost, and hence, $M(\hat{I},\hat{O}) \leq M(I,O)$. Therefore, if we find $M(I,O)$ for all possible disjoint subsets I,O of nodes such that $|I| = |O|$ and $I \cap O = \phi$, we would also find the solution for $\hat{I},\hat{O}$, and obtain the optimal solution. Let $(f,y)_{\phi,\phi}$ denote the optimal solution if there are no reverse segments in the feasible solution, i.e., if we send kC units along the shortest α path and r units along the shortest β path. An exact algorithm can be described as follows.

The Three Match Algorithm
begin
find $a(i,j)$, $b(i,j)$, $c(i,j)$ for all pairs of nodes
$I^* = O^* = \Phi$, $M = k * a(s,t) + b(s,t)$
$(f,y) \longleftarrow (f,y)_{\phi,\phi}$
for all $I,O \subset N : |I| = |O|$, $I \cap O = \phi$ **do**
begin

solve the matching problems on $H_{IO}(F), H_{IO}(R), H_{IO}(K)$ and obtain $(f,y)_{IO}$
if $M(I,O) < M$ **then do**
    begin
    $M = M(I,O)$
    $(f,y) \longleftarrow (f,y)_{IO}$
    $I^* = I$, $O^* = O$
    end
end
end

It takes $O(n^3)$ iterations to find $a(i,j), b(i,j), c(i,j)$ for all $i,j \in N$. For each subset $I \cup O \subset N$ of cardinality $2q \leq n$, there are ${}^{2q}C_q$ ways of splitting the $2q$ nodes into two sets I and O of equal cardinality, where ${}^nC_q$ is the number of ways of choosing q items from n. There are ${}^nC_{2q}$ subsets $I \cup O$ of cardinality $2q$. For each pair of subsets I, O of cardinality q each, we solve three bipartite matching problems which takes $p(n,m)$ time where $p(n,m)$ is a polynomial in n and m. The compexity of the algorithm is therefore

$$\begin{aligned}
\sum_{q=1}^{0.5n} ({}^nC_{2q}\ {}^{2q}C_q)p(n,m) &\leq p(n,m)\sum_{q=1}^{0.5n}({}^nC_{2q})2^{2q} \\
&\leq p(n,m)\sum_{q=1}^{n}(1+2)^n \\
&= O(3^n p(n,m))
\end{aligned}$$

Bipartite matching can be solved by the Hungarian method in $O(n^3)$ time. The running time, $O(n^3 3^n)$ of algorithm three match is better than the $\Omega(n^{2k})$ multiple path algorithm described by Chopra, Gilboa and Sastry if k is large. However, Lemma 1 implies that the multiple path algorithm can be improved to $\Omega(n^{2k^*})$. Even in this case, algorithm three match is better than the multiple path algorithm for $k > (n-2)/2$.

The three match algorithm can be speeded up in practice as follows, although the worst case performance does not improve. Suppose that for a given I, O, the free path visits only nodes $\hat{I} \subset I$ and $\hat{O} \subset O$. Then since $M(\hat{I},\hat{O}) \leq M(I,O)$, we need not check any subsets $Q_I \subset I$, $Q_O \subset O$ such that $\hat{I} \subset Q_I$, $\hat{O} \subset Q_O$ since $M(\hat{I},\hat{O}) \leq M(Q_I,Q_O)$. Therefore, the algorithm can start with $|I| = |O| = \lfloor (n-2)/2 \rfloor$ and then decrease the cardinality of I and O. If at any stage, $|\hat{I}| < |I|$, we can ignore subsets Q_I of I such that $|\hat{I} \subset Q_I$.

3 Integer Optimal Solutions

Chopra, Gilboa and Sastry describe an extended reformulation of $OFOC$, and show that it represents a valid formulation for the problem. We describe a slightly modified version EF of the formulation which is valid for $OFOC$ and $TFOC$ where we eliminate the fixed charge variables y_{ij}, y^1_{ij} and y^2_{ij}. Define $w_{ij}(h) = w_{ij} + Cp_{ij}$, $w_{ij}(e) = w_{ij} + rp_{ij}$, $w_{ij}(g) = w_{ij} + (C-r)p_{ij}$ for $OFOC$ and $w_{ij}(h) = \min\{w^2_{ij},\ Cw^1_{ij}\}$, $w_{ij}(e) = \min\{w^2_{ij},\ rw^1_{ij}\}$, and $w_{ij}(g) = \min\{w^2_{ij},\ (C-r)w^1_{ij}\}$ for $TFOC$.

$$\textbf{Min} \sum_{(ij)\in A} [w_{ij}(h)h_{ij} + w_{ij}(e)e_{ij} + w_{ij}(g)g_{ij})]$$

s.t.

$$\sum_j (e_{ji} - g_{ji} - e_{ij} + g_{ij}) = \begin{cases} -1 & \text{for i = s} \\ 1 & \text{for i = t} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$\sum_j (h_{ji} + g_{ji} - h_{ij} - g_{ij}) = \begin{cases} -k & \text{for i = s} \\ k & \text{for i = t} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$e_{ij}, h_{ij}, g_{ij} \geq 0, \text{ integer}$$

In this formulation, variables e_{ij}, g_{ij}, h_{ij} denote the forward, reverse and full flows in a feasible solution. Define the polytopes,

$$EP = \{(e,g,h) \geq 0 | (e,g,h) \text{ satisfies } (5),(6)\},$$

$$EIP = \{(e,g,h) \in EP | e,g \in \{0,1\}, h \text{ integer}\}.$$

We show that EP always gives integer optimal solutions for graphs with at most 5 nodes. We also give an example of a 6 node graph with a fractional solution. Hence, EP guarantees integer optimal solutions for graphs with at most 5 nodes. To establish the result, we describe a dual based algorithm *Free Tree* that provides a heuristic solution for $OFOC$. The algorithm identifies a tree with a free path connecting s to every node. Assume that if $(i,j) \notin A$ for any node pair i,j, then $w_{ij}(h) = w_{ij}(e) = w_{ij}(g) = \infty$. Since no reverse arcs enter or leave nodes s and t, assume that no g flow enters or leaves nodes

s and t, and hence $g_{sj} \equiv g_{js} \equiv 0$, $g_{jt} \equiv g_{tj} \equiv 0$ for all nodes j. The dual **DEF** of the LP-relaxation of EF is for $OFOC$ is

$$\mathbf{Max}(\alpha_t - \alpha_s)k + \beta_t - \beta_s$$

subject to:

$$\begin{aligned} \alpha_j - \alpha_i &\leq w_{ij}(h) \\ \beta_j - \beta_i &\leq w_{ij}(e) \\ \alpha_j - \alpha_i - \beta_j + \beta_i &\leq w_{ij}(g) \\ \alpha_i, \beta_i &\geq 0. \end{aligned}$$

The algorithm first fixes a set of α_j values satisfying the dual constraints $\alpha_j - \alpha_i \leq w_{ij}(h)$. It then finds a free path based on these α_j values by using a shortest path label correcting type of procedure to ensure that the dual constraints $\beta_j - \beta_i \leq w_{ij}(e)$ and $\alpha_j - \alpha_i - \beta_j + \beta_i \leq w_{ij}(g)$ are satisfied. It then adjusts the values of α_j and finds a possibly better free path, and iterates until dual feasibility is attained. The number of iterations can be arbitrarily fixed at some number, say μ. The free path obtained from the label correcting procedure is called the *standard free path.*

The α_j values are adjusted at the end of each iteration based on the full matching associated with the standard free path having turn nodes I, O. However, unlike the exact algorithm which always maintains primal feasibility, we assume that each of the nodes s and t is split into $|I|$ while solving the full matching. Let $H^*_{IO}(K)$ denote the corresponding graph. Suppose nodes $[i_1, j_1]$, $[i_2, j_2], \ldots, [i_q, j_q]$, $[i, j]$ are matched in the full matching, where $\{i_1, \ldots, i_q, i\} \subset O \cup \{s\}$ and $\{j_1, \ldots, j_q, j\} \subset I \cup \{t\}$. An *alternating path* between s and any out turn node i matched to node j is $s - j_1 - i_1 - j_2 - i_2 - \ldots - j_q - i_q - j - i$ where full segments $s - j_1$, $i_1 - j_2, \ldots, i_q - j$ are traversed in the forward direction and full segments $i_1 - j_1, i_2 - j_2, \ldots, i_q - j_q$, $i - j$ in the reverse direction. Full segments traversed in the forward direction correspond to unmatched nodes in $H_{IO}(K)$, and full segments traversed in the reverse direction correspond to matched nodes in $H_{IO}(K)$. The cost of this alternating path is defined as $a(s, j_1) - a(i_1, j_1) + a(i_1, j_2) - \ldots + a(i_q, j) - a(i, j)$. Let $m(i)$ be the minimum cost of all alternating paths to out turn node i.

There are at least two possible rules for adjusting these α_i values. First, for each out turn node i, we set $\alpha_i = a(s, t) - a(i, t)$. Another possible rule is to

set $\alpha_i = m(i)$ for all out turn nodes. However it requires more work to set α_i values using this rule. For all other nodes we set $\alpha_j = \min\{\alpha_i + a(i,j) : i \in O \cup \{s\}\}$. Let LB denote the value of the lower bound on the optimal cost equal to the dual optimal solution $k * (\alpha_t - \alpha_s) + (\beta_t - \beta_s)$.

The Free Tree Algorithm
Initialize.
for all $i \in V$ let $\alpha_i = a(s,i)$, $\beta_i = b(s,i)$, $pred(i) \longleftarrow s$.
$I^* = O^* = \Phi$, $count = 0$, $LB = 0$
while $count \leq \mu$ **do**
begin
call *free path*
if $LB < k * a(s,t) + \beta_t$, and $count \leq \mu - 1$ **then**
begin
$LB = k * a(s,t) + \beta_t$, $(I^*, O^*) = (I, O)$
solve full matching on $H^*_{IO}(K)$
for $i \in O$, $\alpha_i = m(i)$
for $j \notin O$, $\alpha_j = \min\{\alpha_i + a(i,j) : i \in O,\ a(s,j)\}$
end
$count = count + 1$
end$\{while\}$

procedure *free path*
$\beta_j = b(s,j)$ for all $j \in N$
while some arc (i,j) satisfies $\beta_j > \beta_i + w_{ij}(e)$ or
$\beta_i > \beta_j + w_{ij}(g) + \alpha_i - \alpha_j$ **do**
begin
if $\beta_j > \beta_i + w_{ij}(e)$ **then** * (i,j) is a possible forward arc*\
    begin
        $\beta_j = \beta_i + w_{ij}(e)$
        $pred(j) \longleftarrow i$
    end
if $\beta_i > \beta_j + w_{ij}(g) + \alpha_i - \alpha_j$ **then** * (i,j) is a possible reverse arc*\
    begin
        $\beta_i = \beta_j + w_{ij}(g) + \alpha_i - \alpha_j$
        $pred(i) \longleftarrow j$
    end

end
using $pred(i)$ values trace free path and find I, O
return β_t, I, O
end$\{free\ path\}$

It is easy to obtain an upper bound on the optimal cost. Solve the full matching problem on $H_{IO}(K)$ assuming $k_I = \min\{|I|,\ k\}$ where I, O are the in turn and out turn nodes on the standard free path obtained from the algorithm, and let the cost be $M(I,O)$. However, $M(I,O)$ may equal ∞ if a full matching is not feasible as in Figure 1 with $k = 1$ and a free path $s-1-2-3-4-t$. If $M(I,O)$ is finite, set $e_{ij} = 1$ for all forward arcs and $g_{ij} = 1$ for all reverse arcs on the standard free path. Let the cost of the free path be $F(I,O) + R(I,O)$, and let $HEUR = \min\{k * a(s,t) + b(s,t),\ M(I,O) + F(I,O) + R(I,O)\}$. We thus obtain an upper bound on the optimal solution, whereas free tree gives a lower bound $LB = k * \alpha_t + \beta_t$. We have not tested the heuristic since the computational results of Chopra, Gilboa and Sastry show that the extended formulation EP gives optimal integer solutions for $OFOC$ and $TFOC$ in all instances tested. However, the heuristic is used here to derive conditions on the cost under which we get integer solutions, and to prove that for problems with up to five nodes, EP always guarantees integer solutions.

Example 2

We use problem $OFOC$ and Figure 1 to illustrate algorithm free tree which shows a graph with 6 nodes where fixed costs w_{ij} are shown above the arcs and the variable costs p_{ij} are shown below the arcs. If a particular cost on an arc is not shown, then it is zero. Let $C = 10$, $r = 1$, and $k = 1$ i.e., $d = 10k + 1 = 11$.

At initialisation, $\alpha_i = a(s,i)$ and $\beta_i = b(s,i)$ for all nodes, and $\alpha_t = 25$. After the first call to procedure *free path*, the free path is $s-1-2-3-4-t$ and the β_i values are

$$\beta_s = 0, \beta_1 = 1, \beta_2 = 1, \beta_3 = 2, \beta_4 = 2, \beta_t = 3.$$

This gives a lower bound $LB = 25 * k + 3$. The full matching is $[s,2]$, $[s,4]$, $[1,t]$ and $[3,t]$, and the α_i values are adjusted to

$$\alpha_s = 0, \alpha_1 = 5, \alpha_2 = 10, \alpha_3 = 15, \alpha_4 = 20, \alpha_t = 25$$

whether we use the rule $\alpha_i = m(i)$ or $\alpha_i = a(s,t) - a(i,t)$ for all out turn nodes. After the next call to procedure *free path*, the free path remains unchanged and the β_i values are

$$\beta_s = 0, \beta_1 = 1, \beta_2 = 6, \beta_3 = 7, \beta_4 = 12, \beta_t = 13.$$

The algorithm does not provide any further improvement in the lower bound and terminates with a lower bound of $LB = 38$ and an upper bound of $HEUR = 41 = a(s,t) + b(s,t)$ if $k = 1$, which in this case equals the cost of the optimal solution.

Lemma 4 *Algorithm free tree gives a dual feasible solution.*

Proof

We first show that for any two out turn nodes $i, u \in O$, $\alpha_i + a(i,u) \geq \alpha_u$, and hence, the re-set α values for out turn nodes satisfy the dual constraint $\alpha_u - \alpha_i \leq a(i,u) \leq w_{iu}(h)$. If we use the rule $\alpha_i = m(i)$, and node u is matched to node v, then $\alpha_i + a(i,u) = m(i) + a(i,u) \geq m(i) + a(i,v) - a(u,v) \geq m(u)$ where the last inequality follows from the fact that $m(i) + a(i,v) - a(u,v)$ is an upper bound on the cost of an alternating path to node u that passes through node i. Notice therefore that $\alpha_i + a(i,s) \geq a(s,s) = 0$ for every node $i \in O$ when we re-set the values after the first iteration. Hence, $\alpha_s = \min\{\alpha_s + a(i,s),\ a(s,s)\} = 0$. This holds true after every iteration, and hence, α_s always equals zero. Similarly, $\alpha_t = a(s,t)$ after every iteration.

If we use the rule $\alpha_i = a(s,t) - a(i,t)$, then $\alpha_i + a(i,u) = a(s,t) - a(i,t) + a(i,u) \geq a(s,t) - a(u,t) = \alpha_u$. Arguments similar to those used in the previous case establish that $\alpha_s = 0$ and $\alpha_t = a(s,t)$ after every iteration.

Next, we show that we do not have any negative cost free cycles at the start of any iteration in procedure free path. This ensures that the procedure terminates, and hence algorithm free tree terminates. Consider any cycle Φ. Suppose we go around the cycle in a clockwise direction. Let Φ^f (Φ^b) denote the set of arcs in the forward (reverse) direction. Let $w_f(e)$ ($w_b(g)$) denote the total cost of arcs in Φ^f (Φ^b). Let $w_f(h)$ ($w_b(h)$) denote the total cost of arcs in Φ^f (Φ^b) if C units are sent on these arcs. Procedure free path measures the cost of any reverse arc (j,i) as $\beta_j - \beta_i = \alpha_j - \alpha_i + w_{ji}(g)$. Hence the total cost K around the cycle for procedure free path is

$$K = w_f(e) + w_b(g) + \sum_{kl \in \Phi^b} (\alpha_k - \alpha_l).$$

Hence $K \geq 0$ if and only if

$$\sum_{kl \in \Phi^b} (\alpha_l - \alpha_k) \leq w_f(e) + w_b(g).$$

Since the α_j labels are updated at the end of every iteration and always satisfy the constraints $\alpha_j - \alpha_i \leq w_{ij}(h)$,

$$\sum_{kl \in \Phi^b} (\alpha_l - \alpha_k) \leq w_b(h), \text{and}$$

$$\sum_{kl \in \Phi^f} (\alpha_l - \alpha_k) \leq w_f(h).$$

Since

$$\sum_{kl \in \Phi^b} (\alpha_l - \alpha_k) = \sum_{kl \in \Phi^f} (\alpha_l - \alpha_k),$$

$$\sum_{kl \in \Phi^b} (\alpha_l - \alpha_k) \leq \min \{w_f(h),\ w_b(h)\}.$$

Taking a convex combination of $w_f(h)$ and $w_b(h)$ with weights r/C and $(C-r)/C$ shows that $\min \{w_f(h),\ w_b(h)\} \leq rw_f(h)/C + (C-r)w_b(h)/C$. In problem $OFOC$, $rw_{ij}(h)/C = r(w_{ij} + Cp_{ij})/C \leq w_{ij} + rp_{ij} = w_{ij}(e)$. In problem $TFOC$, assume $w^2_{ij} < Cw^1_{ij}$: otherwise, we need not consider facility 2 while finding an optimal solution. Therefore, $w_{ij}(e) = \min \{rw^1_{ij},\ w^2_{ij}\} \geq rw^2_{ij}/C \geq rw_{ij}(h)/C$. Similarly, $(C-r)w_{ij}(h)/C \leq w_{ij}(g)$ for both $OFOC$ and $TFOC$. Therefore, $\sum_{kl \in \Phi^b}(\alpha_l - \alpha_k) \leq w_f(e) + w_b(g)$. It follows that $K \geq 0$. The same arguments hold if we go around the cycle in a counter clockwise direction.

Since we do not have any negative cost cycles, the procedure free path will terminate. For all arcs, the inequalities $\beta_j - \beta_i \leq w_{ij}(e)$ and $\alpha_j - \alpha_i - \beta_j + \beta_i \leq w_{ij}(g)$ are therefore satisfied since otherwise the procedure will not terminate. The labels α_j are changed at the end of every iteration, while ensuring that the constraints $\alpha_j - \alpha_j \leq a(i,j) \leq w_{ij}(h)$ are satisfied. We therefore have a dual feasible solution.

□

We say that algorithm free tree satisfies the α *matching condition* if $\alpha_j = \alpha_i + a(i,j)$ for every matched pair of nodes i and j in the optimal full matching in $H_{IO}(K)$ such that $i \in O \cup \{s\}$ and $j \in I \cup \{t\}$.

Lemma 5 *If a standard free tree satisfies the α matching condition then algorithm free tree solves $OFOC$ and $TFCO$.*

Proof.
Construct a primal solution as follows. Set $e_{ij} = 1$ for forward arcs and $g_{ij} = 1$ for reverse arcs on the standard free path. Set h_{ij} equal to the number of full matchings passing through arc (i, j) in G. It is easy to verify that we obtain a feasible solution to EF.

We show that this solution satisfies complementary slackness conditions. If $e_{ij} = 1$, then arc (i, j) is a forward arc on the standard free path and hence $\beta_j - \beta_i = w_{ij}(e)$. If $g_{ij} = 1$, then arc (i, j) is a reverse arc on the free path. Hence $\alpha_j - \alpha_i - \beta_j + \beta_i = w_{ij}(g)$. If arc (i, j) carries h flow from an out turn node to an in turn node, then it is on a full segment corresponding to a matched arc (i^*, j^*) in $H_{IO}(K)$. Clearly, this full segment is a shortest α path between nodes i^* and j^* in G of length $a(i^*, j^*)$. Therefore, $\alpha_j - \alpha_i = w_{ij}(h)$ for all arcs (i, j) through which the matched arc (i^*, j^*) passes. Therefore complementary slackness conditions are satisfied, and hence we get an optimal integer solution.

□

For instance, in Figure 1 and Example 2, the α matching condition is satisfied if $k = 2$. Algorithm free tree gives the same dual solution as in Example 2, which now costs 63. The optimal primal solution with free path $s - 1 - 2 - 3 - 4 - t$ and full matching $[s, 2]$, $[s, 4]$, $[1, t]$ and $[3, t]$ also costs 63. We now establish further conditions for obtaining integer optimal solutions from free tree. Algorithm free tree finds a free path with an associated set of turn nodes I, O after a call to procedure free path. Immediately after this, the algorithm re-sets the α_i values for all nodes based on the turn nodes I and O. It then calls procedure free path again and finds another free path based on the revised α_i values. Let I^*, O^* denote the revised set of turn nodes. If the α_i values are re-set based on the minimum cost alternating distance $m(i)$, and $I^* = I$, $O^* = O$, then the free path *fits* the turn nodes I, O.

Lemma 6 *If a free path fits a set of turn nodes I and O, then algorithm free tree gives an integer optimal solution.*

Proof.
We show that the α matching condition is satisfied. If $\alpha_j = \alpha_i + a(i, j)$

for every in turn node $j \in I \cup \{t\}$ matched to out turn node $i \in O \cup \{s\}$, then we are done. Suppose on the contrary, $\alpha_j < \alpha_i + a(i,j)$ for some in turn node j matched to out turn node i. Since $\alpha_s = 0$, it follows that $\alpha_j = \min \{\alpha_u + a(u,j) : u \in O \cup \{s\}\}$. If $\alpha_j = \alpha_u + a(u,j)$ for some $u \neq i$, $u \in O \cup \{s\}$, then $\alpha_u = m(u)$, $\alpha_i = m(i)$ and $m(u) + a(u,j) - a(i,j) < m(i)$. Hence, we can find an alternating path to node i of lower cost via node u. However, this is a contradiction since $m(i)$ is the cost of the minimum alternating path.

□

Theorem 2 *If graph G has at most 5 nodes, then EF always gives integer optimal solutions for $OFOC$ and $TFOC$.*

Proof

If G has at most 5 nodes, then it has at most one in turn node other than node t. We use a modified version of free tree where we initially set $\alpha_u = a(s,u)$ for every node and then re-set α_i for any out turn node $i \neq s$ on the free path to

$$\begin{aligned} \alpha_i \;=\; & \max \{b(s,i) + a(s,j) + c(j,i) - b(s,j),\ a(s,j) + c(j,i) + b(j,t) - b(i,t), \\ & a(s,t) - a(i,t),\ a(s,j) - a(i,j)\} \end{aligned}$$

after every call to procedure free path. The value of α_u for other nodes u is re-set using the relation $\alpha_u = \min \{\alpha_i + a(i,u),\ \alpha_u\}$. We show that we get an optimal integer solution in at most three calls to procedure free path. Suppose there are no turn nodes on the standard free path after a call to free path. Then the α matching condition is trivially satisfied. Hence, Lemma 5 implies that free tree solves $OFOC$ and $TFOC$.

We now consider cases where $I = \{j\}$, $O = \{i\}$, $I \cap O = \Phi$, $i \neq s,t$; $j \neq s,t$ after the first call to procedure free path, and there are turn nodes other than s and t on the free path after the second and third calls. Let s, i, j, u, t be the 5 nodes and let α_i, α_u, α_j, denote the values when we re-set them after the first call, and α_i^2, α_u^2, α_j^2 the values when we re-set them after the second call. Notice that for any in turn node j, $\alpha_i + a(i,j) \geq a(s,j)$ and hence, $\alpha_j = a(s,j)$. Let I^*, O^* denote the in turn and out turn nodes after the second call to free path. Let $b(i,j)$ and $\alpha_j - \alpha_i + c(j,i)$ denote the cost of a forward arc (i,j) and a reverse arc (j,i) respectively since procedure free

path sets $\beta_j - \beta_i$ equal to these quantities for any arc on the free path. For any free path $i_1 - i_2 - \ldots - i_m$, let $q_l(i_1 - i_2 - \ldots - i_m)$ denote the cost of the path after the lth call to procedure free path for $l = 1, 2, 3$. We say that path $i_1 - i_2 - \ldots - i_m$ has no turn nodes if none of the intermediate nodes $i_2 - \ldots - i_{m-1}$ is an in turn or out turn node. There are six cases to consider.

Case 1. $I = I^* = \{j\}$ *and* $O = O^* = \{i\}$.

In this case the free path remains the same before and after the second call to procedure free path. If $\alpha_i = a(s,t) - a(i,t)$, then $a(s,t) - a(i,t) \geq a(s,j) - a(i,j)$ and hence the optimal full matching is $[s,j]$, $[i,t]$ since its cost $a(s,j) + a(i,t)$ is at most equal to the cost of the other full matching $[s,t]$, $[i,j]$. Since $\alpha_s = 0$, $\alpha_j = a(s,j)$ and $\alpha_t = a(s,t)$, the α matching condition is satisfied. Hence we get an integer optimal solution. A similar argument establishes that if $\alpha_i = a(s,j) - a(i,j)$, then the optimal full matching is $[s,t]$, $[i,j]$, and hence, the α matching condition is satisfied.

If $\alpha_i = b(s,i) + a(s,j) + c(j,i) - b(s,j)$, then $b(s,i) + a(s,j) - \alpha_i + c(j,i) = q_2(s - i - j) = b(s,j)$ where i is an out turn node. Hence we have a minimum cost free path $s - j - t$ without any turn nodes since path $s - i - j$ with out turn node i costs as much as the path $s - j$ without any turn node. Hence, we are done. A similar argument establishes that if $\alpha_i = a(s,j) + c(j,i) + b(j,t) - b(i,t)$, then $s - i - t$ is a mimnimum cost path without any turn nodes.

Case 2. $I^* = \{u\}$ *and* $O^* = O$.

Since $s - i - j - t$ is the intitial free path, $q_1(i - j - t) \leq q_1(i - u - t)$. There is a change in the cost of paths $i - j - t$ and $i - u - t$ before and after the second call if and only if the α value at node i changes. The change in the cost in both cases equals $a(s,i) - \alpha_i$, and hence $q_2(i - j - t) - q_1(i - j - t) = q_2(i - u - t) - q_1(i - u - t)$. Adding the inequality $q_1(i - j - t) \leq q_1(i - u - t)$ to this equation establishes that $q_2(i - j - t) \leq q_2(i - u - t)$. But since $s - i - u - t$ is the free path after the second call, $q_2(i - u - t) \leq q_2(i - j - t)$, and hence both paths $s - i - j - t$ and $s - i - u - t$ have equal cost after the second call. This reduces to Case 1.

Case 3. $I^* = \{u\}$ *and* $O^* = \{j\}$.

The free path after the first call is $s - i - j - t$ which changes to $s - j - u - t$ after the second call. Therefore, $b(s,j) \leq q_2(s - i - j) = b(s,i) + a(s,j) - \alpha_i + c(j,i)$. The definition of α_i implies that $\alpha_i \geq b(s,i) + a(s,j) + c(j,i) - b(s,j)$. Hence,

$b(s,j) = q_2(s-i-j)$. Similarly, since $\alpha_u \leq a(s,u)$ and $\alpha_j = a(s,j)$, it follows that $q_1(j-u-t) \leq q_2(j-u-t)$. Since $s-i-j-t$ is the free path after the first call, $b(j,t) \leq q_1(j-u-t)$, and since $s-j-u-t$ is the free path after the second call, $q_2(j-u-t) \leq b(j,t)$. Hence, $q_1(j-u-t) = q_2(j-u-t) = b(j,t)$. Since $b(s,j) = q_2(s-i-j)$ and $q_2(j-u-t) = b(j,t)$, it follows that $s-i-j-t$ remains a minimum cost free path. This reduces to Case 1.

Case 4. $I^* = \{i\}$ *and* $O^* = \{j\}$.

The revised free path after the second call to procedure free path is $s-j-i-t$. If $\alpha_i > a(s,j) + c(j,i) + b(j,t) - b(i,t)$ after re-setting then $b(i,t) > a(s,j) - \alpha_i + c(j,i) + b(j,t)$, i.e., the path $i-j-t$ with out turn node i costs less than the path $i-t$, with no turn nodes. However, in that case, $s-j-i-t$ cannot be the revised free path. Therefore, $\alpha_i = a(s,j) + c(j,i) + b(j,t) - b(i,t)$. A similar argument establishes that $\alpha_i = b(s,i) + a(s,j) + c(j,i) - b(s,j)$. Therefore, $b(s,i) + b(i,t) = b(s,j) + b(j,t)$, and any free path with turn nodes costs at least as much as free paths $s-i-t$ and $s-j-t$ with no turn nodes. Hence, $s-i-t$ without any turn nodes is a free path after the second call.

Case 5. $I^* = \{j\}$ *and* $O^* = \{u\}$.

The free path after the second call is $s-u-j-t$. In this case, we re-set the α values after the second call, and run procedure free path again for the third time. Let I^2, O^2 denote the in turn and out turn nodes after the third call. If $I^2 = \{j\}$, $O^2 = \{u\}$, then it reduces to Case 1 where the turn nodes do not change after a call. If $I^2 = \{u\}$, $O^2 = \{j\}$, then it reduces to Case 4 where the in turn and out turn node are interchanged after a call to procedure free path.

Case 5a. Suppose $I^2 = \{i\}$, $O^2 = \{u\}$.

The three free paths are $s-i-j-t$, $s-u-j-t$ and $s-u-i-t$ after the three calls to procedure free path. If $\alpha_i^2 = \alpha_i$ then $q_3(u-j-t) - q_2(u-j-t) = q_3(u-i-t) - q_2(u-i-t)$. Adding the inequality $q_2(u-j-t) \leq q_2(u-i-t)$ to this equation establishes that $q_3(u-j-t) \leq q_3(u-i-t)$. But then $s-u-i-t$ is the minimum cost free path after the third call, and hence $s-u-j-t$ is also a minimum cost free path. Therefore, $s-u-j-t$ remains a minimum cost free path before and after the third call. This reduces to Case 1. If $\alpha_i^2 < \alpha_i$, then $\alpha_i^2 = \alpha_u^2 + a(u,i)$. Then free path $s-u-i-t$ after the third call with full matching $[s,t]$, $[u,i]$ satisfies the α matching condition.

Case 5b. $I^2 = \{j\}$, $O^2 = \{i\}$.

Consider four cases. First, if $\alpha_i = a(s,t) - a(i,t)$, then $\alpha_u^2 + a(u,i) \geq a(s,t) - a(u,t) + a(u,i) \geq a(s,t) - a(i,t)$. Hence $\alpha_u^2 + a(u,i) \geq \alpha_i$, and therefore, $\alpha_i^2 = \alpha_i$. Second, if $\alpha_i = a(s,j) - a(i,j)$, a similar argument establishes that α_i does not change, i.e., $\alpha_i^2 = \alpha_i$. Therefore, these two cases reduce to Case 1 where the out turn and in turn nodes i and j do not change after a call to free path. Third, suppose $\alpha_i = b(s,i) + a(s,j) + c(j,i) - b(s,j)$. Then $b(s,j) = b(s,i) + a(s,j) - \alpha_i + c(j,i) = q_2(s-i-j) \leq q_3(s-i-j) \leq q_3(s-u-j)$ where the first inequality follows from the fact that $\alpha_i^2 \leq \alpha_i$ and $\alpha_j = \alpha_j^2 = a(s,j)$, and the second inequality from the fact that $s-i-j$ is the shortest cost path after the third call. Further, it follows from the fact that $\alpha_u^2 \geq a(s,j) + c(j,u) + b(s,u) - b(s,j)$ that $q_3(s-u-j) \leq b(s,j)$. Therefore, $b(s,j) = q_2(s-i-j) = q_3(s-i-j) = q_3(s-u-j)$. Hence path $s-j-t$ without any turn nodes is a minimum cost free path after the third call. Similarly, if $\alpha_i = a(s,j) + c(j,i) + b(j,t) - b(i,t)$, then $b(i,t) = q_2(i-j-t)$. Since $\alpha_i^2 \leq \alpha_i$, $q_2(i-j-t) \leq q_3(i-j-t)$. But $q_3(i-j-t) \leq b(i,t)$ since $i-j-t$ is a free path after the third call. Therefore, $b(i,t) = q_3(i-j-t)$ and hence, free path $s-i-t$ without any turn nodes is a minimum cost free path after the third call.

Case 5c. Suppose $I^2 = \{u\}$, $O^2 = \{i\}$.

The three free paths are $s-i-j-t$, $s-u-j-t$ and $s-i-u-t$ after the three calls to procedure free path. Since $\alpha_u^2 \geq a(s,j) + c(j,u) + b(j,t) - b(u,t)$, and $b(u,t) \leq q_3(u-j-t)$, it follows that $b(u,t) = q_3(u-j-t)$, and hence path $s-i-u-j-t$ with out turn node i and in turn node j is a free path after the third call. Hence $q_3(s-i-u-j-t) \leq q_3(s-i-j-t)$ where the righthand side is the cost of the path with out turn node i and in turn node j. This implies that $c(j,u) + c(u,i) \leq c(j,i)$. Since all costs are nonnegative, this implies that $c(j,u) + c(u,i) = c(j,i)$ and hence $s-i-j-t$ is a shortest path after the third call. This reduces to Case 5b.

Case 5d. $I^2 = \{i\}$ *and* $O^2 = \{j\}$.

From the definition of α_u^2 it follows that $q_3(s-u-j) \leq b(s,j)$ where path $s-u-j$ has out turn node u. Since $s-j-i-t$ is a minimum cost path after the third call, $b(s,j) \leq q_3(s-u-j)$. Hence, $b(s,j) = q_3(s-u-j)$. Therefore, path $s-u-j-i-t$ is a minimum cost free path with out turn node u and in turn node i after the third call. This reduces to Case 5a.

Case 6. $I^* = \{i\}$ *and* $O^* = \{u\}$.

The intitial free path is $s-i-j-t$, and after the second call, it is $s-u-i-t$. Therefore, $b(i,t) \leq q_2(i-j-t) = a(s,j) - \alpha_i + c(j,i) + b(j,t)$. It follows from the definition of α_i that $b(i,t) = q_2(i-j-t)$. Hence, after the second call, we can consider path $s-u-i-j-t$ as the free path with u as an out turn node and j as an in turn node. This reduces to Case 5.

□

The next example shows that the linear programming relaxation of EF for a 6 node graph gives a fractional solution.

Example 3

Consider $OFOC$ in Figure 1. If $k = 1$, then Example 2 shows that algorithm free tree does not give the optimal solution. The linear programming relaxation EP gives an optimal solution with an objective function value of 39.5. The variables have the following values:
$e_{s1} = g_{21} = e_{23} = g_{43} = e_{4t} = 0.5$, $e_{st} = 0.5$, $h_{s2} = h_{s4} = h_{1t} = h_{3t} = 0.5$.
The integer optimal solution has no turn nodes, whereas free tree has two out turn and in turn nodes other than s and t.

□

We have thus shown that the polytope EP in the extended formulation guarantees integer optimal solutions for graphs with at most 5 nodes. It does not give the convex hull for graphs with 6 nodes or more as shown in Example 3. Lemma 2, Lemma 5 and Lemma 6 also provide some insight into the reason why Chopra, Gilboa and Sastry (1997) report such excellent computational results for the extended formulation EF: at any stage of the heuristic, if a free path is repeated, or if we get a free path without any turn nodes, we get an optimal solution. Example 3 does have a gap between the linear programming solution and the integer solution. But this gap equals only 1.5 while the problem has a cost structure unlikely to be encountered in practice since nearly all arcs either have a fixed cost or a flow cost, but not both.

4 A Quasi Integral Formulation

We describe another extended formulation for $OFOC$ and $TFOC$ and show that the polyhedron associated with the linear programming relaxation of this formulation is *quasi-integral*, i.e., every edge of the convex hull of feasible integer solutions is also an edge of the polyhedron. This property could be useful in designing a modified version of the simplex method to solve the problem using a sequence of pivots with integer extreme solutions. This method is referred to as the *integral simplex method* by Yemelichev, Kovalev and Kravtsov (1984). Hellstrand, Larsson and Migdalas (1992) have shown that the polyhedron associated with the uncapacitated network design problem is quasi-integral. They also provide references to other research on quasi-integral polytopes.

Consider the extended formulation EF. Define an *e-cycle* (*g-cycle*, *full cycle*) to be a cycle of arcs (ignoring direction) with $e_{ij} > 0$ ($g_{ij} > 0$, $h_{ij} > 0$) on each arc (i,j) in the cycle. The next result follows from Theorem 1.

Theorem 3 *If (e,g,h) is an extreme point of EP then*
(i) e, $g \in \{0,1\}$, h integer
(ii) there are no e,g or full cycles.

Index $k^* = \min\{k, \lfloor (n-2)/2 \rfloor\}$ units of full flow from 1 through k^*, and for $m = 1, \ldots, k^*$, let

$$h_{ij}^m = \begin{cases} 1 \text{ if one unit of full flow numbered } m \text{ is on arc } (i,j) \\ 0 \text{ otherwise} \end{cases}$$

A full flow numbered m ending in in-turn node $i \neq t$ splits into a g-flow numbered m and an e-flow, both leaving node i. Similarly, a full flow numbered m starting in out turn node $i \neq s$ is formed from a g-flow numbered m and an e-flow, both entering node i. Define g_{ij}^m for each arc (i,j) and $m = 1, \ldots, k^*$. The remaining $k - k^*$ units of full flow is denoted by h_{ij}^0. For simplicity we assume that $k > k^*$. Otherwise we can drop the variables h_{ij}^0. By Theorem 3 an extreme point of EP has no full cycles. Hence any of the k^* units of full flow starting in node s and ending in node t and the remaining $k - k^*$ units flow from s to t on one directed path (see Figure 3).

INSERT FIGURE 3 HERE

In this figure, the total demand $d = kC + r$ for some $k > 3$, and $0 < r < C$. Since there are 8 nodes, $k^* = 3$. The flow values are shown on the arcs, and represent an extreme point. Consider the following extended reformulation $EF(k)$.

$$\textbf{Min} \sum_{(ij)\in A} [e_{ij}w_{ij}(e) + \sum_{m=1}^{k^*}\{h_{ij}^m w_{ij}(h) + g_{ij}^m w_{ij}(g)\} + (k-k^*)h_{ij}^0 w_{ij}(h)]$$

s.t.

$$\sum_j (e_{ji} + \sum_{m=1}^{k^*} h_{ji}^m - e_{ij} - \sum_{m=1}^{k^*} h_{ij}^m) = \begin{cases} -(k^*+1) & \text{for } i = s \\ k^*+1 & \text{for } i = t \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

$$\sum_j (h_{ji}^m + g_{ji}^m - h_{ij}^m - g_{ij}^m) = \begin{cases} -1 & \text{for } i = s,\ m = 1,\ldots,k^* \\ 1 & \text{for } i = t \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

$$\sum_j (h_{ji}^0 - h_{ij}^0) = \begin{cases} -1 & \text{for } i = s \\ 1 & \text{for } i = t \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

$$e,\ g,\ h \in \{0,1\}.$$

For technical reasons we introduce the additional constraints

$$\sum_j e_{ij} \leq 1 \quad (10)$$

$$\sum_j h_{ij}^0 \leq 1 \quad (11)$$

$$\sum_j h_{ij}^m \leq 1 \quad (12)$$

$$\sum_j g_{ij}^m \leq 1 \quad (13)$$

If the total cost of sending one unit of e, g or full flow around a cycle is non-negative, there is always an optimal solution that satisfies these conditions. Define the polytopes,-

$$EP(k) = \{(e,g,h) \geq 0 | (e,g,h) \text{ satisfies } (7),\ldots,(13)\},$$

$$EIP(k) = \{(e,g,h) \in EPO(k) | e,g,h \in \{0,1\}\}.$$

This formulation has $O(mn)$ constraints and $O(mn)$ 0-1 variables. The next result is easy to establish and implies that $EIP(k)$ is a valid extended formulation for EIP, and hence, for $OFOC$ and $TFOC$.

Theorem 4 *Any vector* $(e^*, g^*, h^*) \in EIP$ *if and only if there exists a vector* $(e, g, h) \in EIP(k)$ *such that* $e^*_{ij} = e_{ij}$, $g^*_{ij} = \sum_m g^m_{ij}$ *and* $h^*_{ij} = \sum_m h^m_{ij}$ *for every arc* $(i, j) \in A$.

We show that polytope $EP(k)$ is quasi-integral.

Definition 1 . *A graph is* **simple** *if either the out-degree or the in-degree of each node is less than or equal to one.*

The polyhedron $EP(k)$ can be transformed into an equivalent simple graph as follows. Split each node i into $4k^* + 4$ nodes. Flow h^m_{ji} (g^m_{ji}) can enter node $i^m(h)$ ($i^m(g)$) for $m = 1, \ldots, k^*$. Each of these nodes is split into two and is connected by variables $x^m_{ii}(h)$ ($x^m_{ii}(g)$). Arc $u^m_{ii}(hg)$ ($u^m_{ii}(gh)$) can send flow from node $i^m(h)$ to $i^m(g)$ ($i^m(g)$ to $i^m(h)$). Arc $u^m_{ii}(he)$ ($u^m_{ii}(eh)$) can send flow from node $i^m(h)$ to $i(e)$ ($i(e)$ to $i^m(h)$). Flow h^0_{ji} can enter node $i^0(h)$, and flow e_{ji} can enter node $i(e)$. (See Figure 4).

INSERT FIGURE 4 HERE

The constraints of $EP(k)$ correspond to the following constraints in the

equivalent simple graph formulation.

$$\begin{aligned}
\sum_j h^0_{ji} - x^0_{ii}(h) &= \begin{cases} 1 & \text{for } i = t \\ 0 & \text{otherwise} \end{cases} \\
x^0_{ii}(h) - \sum_j h^0_{ij} &= \begin{cases} -1 & \text{for } i = s \\ 0 & \text{otherwise} \end{cases} \\
\sum_j e_{ji} + \sum_{m=1}^{k^*} u^m_{ii}(he) - x_{ii}(e) &= \begin{cases} 1 & \text{for } i = t \\ 0 & \text{otherwise} \end{cases} \\
x_{ii}(e) - \sum_j e_{ij} - \sum_{m=1}^{k^*} u^m_{ii}(eh) &= \begin{cases} -1 & \text{for } i = s \\ 0 & \text{otherwise} \end{cases} \\
\sum_j h^m_{ji} + u^m_{ii}(gh) - x^m_{ii}(h) &= \begin{cases} 1 & \text{for } i = t \\ 0 & \text{otherwise} \end{cases} \\
x^m_{ii}(h) - \sum_j h^m_{ij} - u^m_{ii}(hg) &= \begin{cases} -1 & \text{for } i = s \\ 0 & \text{otherwise} \end{cases} \\
\sum_j h^m_{ji} + u^m_{ii}(eh) - x^m_{ii}(h) &= \begin{cases} 1 & \text{for } i = t \\ 0 & \text{otherwise} \end{cases} \\
x^m_{ii}(h) - \sum_j h^m_{ij} - u^m_{ii}(he) &= \begin{cases} -1 & \text{for } i = s \\ 0 & \text{otherwise} \end{cases} \\
\sum_j g^m_{ji} + u^m_{ii}(hg) - x^m_{ii}(g) &= 0 \\
x^m_{ii}(g) - \sum_j g^m_{ij} - u^m_{ii}(gh) &= 0
\end{aligned}$$

These constraints together with the following constraints define the polytope $SG(k)$.

$$\begin{aligned}
x_{ii}(e) &\leq 1 \\
x^0_{ii}(h) &\leq 1 \\
x^m_{ii}(h) &\leq 1 \\
x^m_{ii}(g) &\leq 1
\end{aligned}$$

These constraints ensure that there are no cycles of flow and therefore flow on all arcs is either 0 or 1. Notice that these constraints imply contraints 10, 11, 12 and 13. Given any point $(e, g, h) \in EP(k)$, there is a unique corresponding point $(e, g, h, x, u) \in SG(k)$ and vice versa. This establishes the next result.

Lemma 7 *There is a unique correspondence between the vertices of* $EP(k)$ *and* $SG(k)$.

We show that the polytope $SG(k)$ is quasi integral.

Definition 2 *(Yemelichev, Kovalev and Kravtsov (1984)). Let Q be a polytope, and Q_z the set of its integer points. The polytope Q is said to be* quasi-integral *if every edge of the convex hull of Q_z is also an edge of Q.*

A sufficient condition for a polytope to be quasi-integral is given in the following theorem. An *integral face* is a face with only integer vertices.

Theorem 5 *(Yemelichev, Kovalev and Kravtsov (1984)). Suppose Q_z belongs to the set of vertices of Q. Then, given any two integer vertices of the polytope Q, if there is an integral face containing them, Q is quasi-integral.*

Based on this result we show that $EP(k)$ is quasi-integral.

Theorem 6 *The polytope $SG(k)$ is quasi-integral.*

Proof

Since $SG(k)$ is contained in a 0-1 hypercube, $SG_z(k)$, the set of integer points, belongs to the set of vertices of $SG(k)$. Let $z^1 = (e^1, g^1, h^1, x^1, u^1)$ and $z^2 = (e^2, g^2, h^2, x^2, u^2)$ denote any two integer vertices of $SG(k)$. Define the index sets

$N_0 = \{l : z_l^1 = z_l^2 = 0\}$, $N_1 = \{l : z_l^1 = z_l^2 = 1\}$, and $N_2 = \{l : z_l^1 \neq z_l^2\}$.

Consider the family of hyperplanes $z_l = 0$ for all $l \in N_0$ and $z_l = 1$ for all $l \in N_1$, each of which supports the polytope $SG(k)$ and the set

$F = \{z \in SG(k) : \; z_l = 0, \; l \in N_0; \; z_l = 1, \; l \in N_1\}$

which constitutes a face of $SG(k)$. Let B denote the constraint matrix of the simple graph polytope $SG(k)$. It suffices to show that the matrix $B_2 = (b_{.l})$, $l \in N_2$ is totally unimodular. This is the matrix obtained when all columns corresponding to variables sharing a common value in both z^1 and z^2 have been deleted. There are three cases to consider.

Case 1. A node is not used by either z^1 or z^2, or both use the same set of incoming and outgoing arcs. The corresponding rows of B_2 are zero rows.

Case 2. Flow enters (leaves) node $i^m(h)$ $(i^m(g), i(e), i^0(h))$ for z^1 and z^2 on different arcs. Since the graph is simple, the outgoing (incoming) arc is common for z^1 and z^2. Therefore the corresponding row in B_2 has two non-zero elements, both $+1$ (-1).

Case 3. Flow enters (leaves) node $i^m(h)$ $(i^m(g), i(e), i^0(h))$ for either z^1 or z^2 but not for both. Then, the corresponding row of B_2 has two elements, one

$+1$ and the other -1.
The case where $i^m(h)$ $(i(e), i^0(h))$ equals $s^m(h)$ $(s(e), s^0(h))$ or $t^m(h)$ $(t(e), t^0(h))$ is a special instance of case 1 or 2. The rows of B_2 therefore have 0, 1 or 2 elements. Delete all rows with zero elements. Consider the following partition of the columns of B_2.
$D_1 = \{$ columns of B_2 corresponding to $z_l^1 = 1$ and $z_l^2 = 0\}$
$D_2 = \{$ columns of B_2 corresponding to $z_l^1 = 0$ and $z_l^2 = 1\}$
Each row contains at most two non-zero elements, and each row of D_1 and D_2 has either one non-zero element or two with opposite signs. Therefore, F is an integer face and hence, $SG(k)$ is quasi-integral.

□

Theorem 7 *The polytope $EP(k)$ is quasi-integral.*

Proof
Let $\hat{z}^1 = (\hat{e}^1, \hat{g}^1, \hat{h}^1)$ and $\hat{z}^2 = (\hat{e}^2, \hat{g}^2, \hat{h}^2)$ be two arbitrary integral vertices of $EP(k)$. Construct the face $\widehat{F}$ as in Theorem 7. The unique corresponding vertices of $SG(k)$ are $z^1 = (e^1, g^1, h^1, x^1, u^1)$ and $z^2 = (e^2, g^2, h^2, x^2, u^2)$. If $\widehat{F}$ is not an integer face, then there is some fractional vertex on this face. Then by Lemma 7, there is a fractional vertex on F corresponding to it. But this is a contradiction since F is an integer face.

5 Conclusions and Further Research

We describe a bipartite matching based exact algorithm for solving $OFOC$ and $TFOC$. This type of algorithm could solve other problems as well. For instance, consider the uncapacitated two commodity network design problem with fixed and flow costs, where each commodity flow alternates between segments of arcs which either share or do not share flow with the other commodity. It is not known whether the problem is easy or hard, but a similar matching type algorithm solves the problem. We also describe an efficient heuristic and use it to show that for problems with at most five nodes, an extended formulation guarantees integer optimal solutions. Finally, we describe a quasi integral formulation for $OFOC$ and $TFOC$. The literature suggests that the quasi integrality property could in principle be used to design a simplex based algorithm where pivots are restricted to integer vertices.

6 References

1. J.Hellstrand, T.Larsson and A.Migdalas, "A characterization of the uncapacitated network design polytope," *OR Letters* **12**, 159-163 (1992).

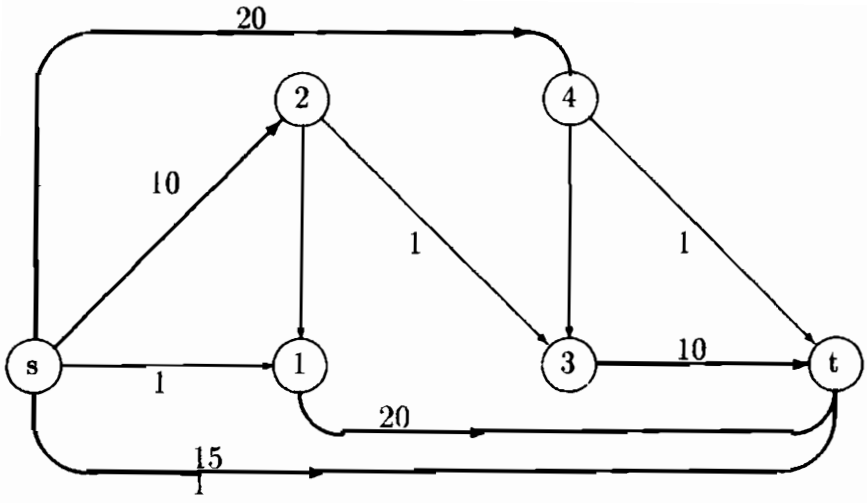
2. S.Chopra, D.Bienstock, O.Gunluck, C.Y.Tsai, "Minimum cost capacity installation for multicommodity networks," Research Report, Northwestern University, January 1995.

3. S.Chopra, I.Gilboa and S.T.Sastry, "Source sink flows with capacity installation in batches," to appear in *Discrete Applied Math.* (1997).

4. T.L.Magnanti and P.Mirchandani, "Shortest Paths, single origin-destination network design and associated polyhedra," *Networks*, vol. 23, No.2 (1993) 103-121.

5. T.L.Magnanti, P.Mirchandani and R.Vachani, "Modeling and solving the two facility capacitated network loading problem," *Oper. Res.* **43** 142-157 (1995).

6. V.A.Yemelichev, M.M.Kovalev and M.K.Kravtsov, "Polytopes, Graphs and Optimization," translated from Russian by G.H.Lawden, Cambridge University Press, Cambridge 1984.



$C = 10,\ r = 1$

FIGURE 1

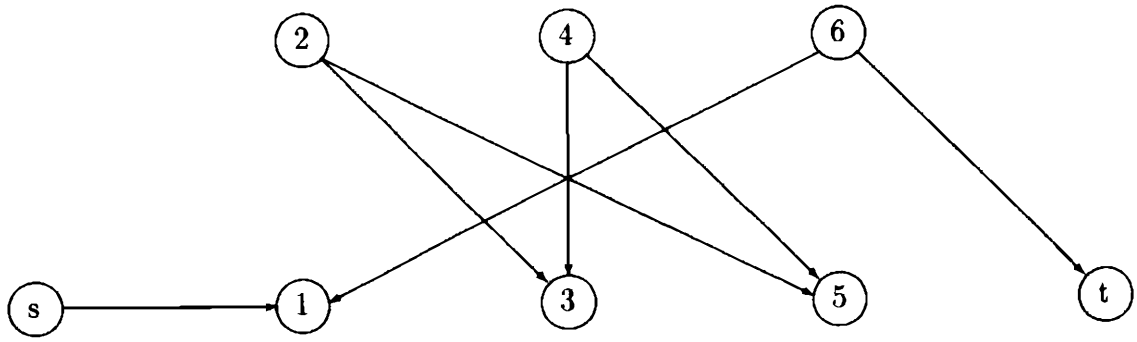


FIGURE 2

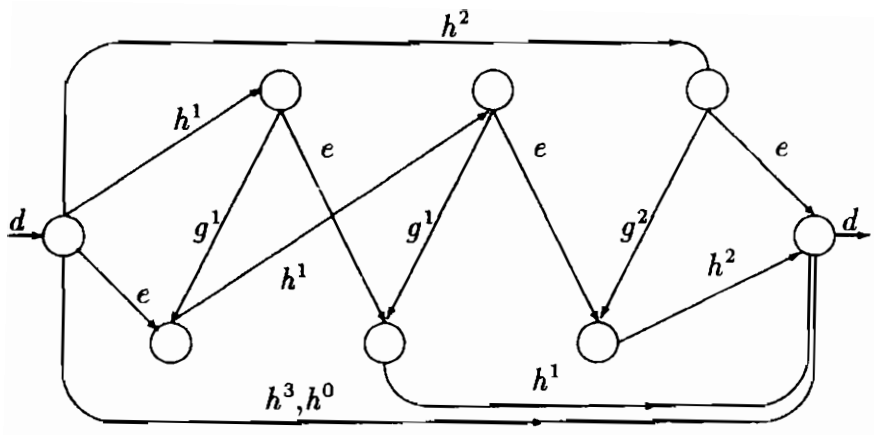


FIGURE 3

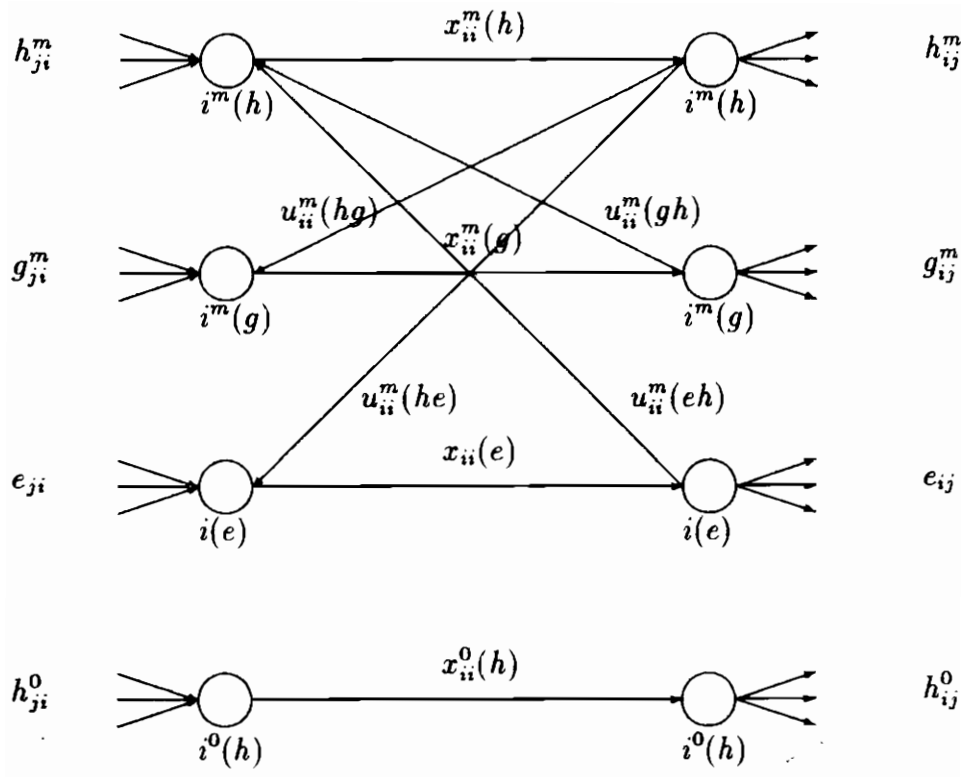

h^m_{ji}
$x^m_{ii}(h)$
h^m_{ij}
$i^m(h)$
$i^m(h)$
$u^m_{ii}(hg)$
$u^m_{ii}(gh)$
$x^m_{ii}(g)$
g^m_{ji}
g^m_{ij}
$i^m(g)$
$i^m(g)$
$u^m_{ii}(he)$
$u^m_{ii}(eh)$
$x_{ii}(e)$
e_{ji}
e_{ij}
$i(e)$
$i(e)$
h^0_{ji}
$x^0_{ii}(h)$
h^0_{ij}
$i^0(h)$
$i^0(h)$

FIGURE 4