

BEBR

**FACULTY WORKING
PAPER NO. 90-1654**

M-median and M-center Problems with
Mutual Communication: Solvable Special Cases

*Dilip Chhajed
Timothy J. Lowe*

The Library of the
JUN 5 1990
University of Illinois
of Urbana-Champaign



College of Commerce and Business Administration
Bureau of Economic and Business Research
University of Illinois Urbana-Champaign

BEBR

FACULTY WORKING PAPER NO. 90-1654

College of Commerce and Business Administration

University of Illinois at Urbana-Champaign

May 1990

M-median and M-Center Problems with Mutual Communication:
Solvable Special Cases*

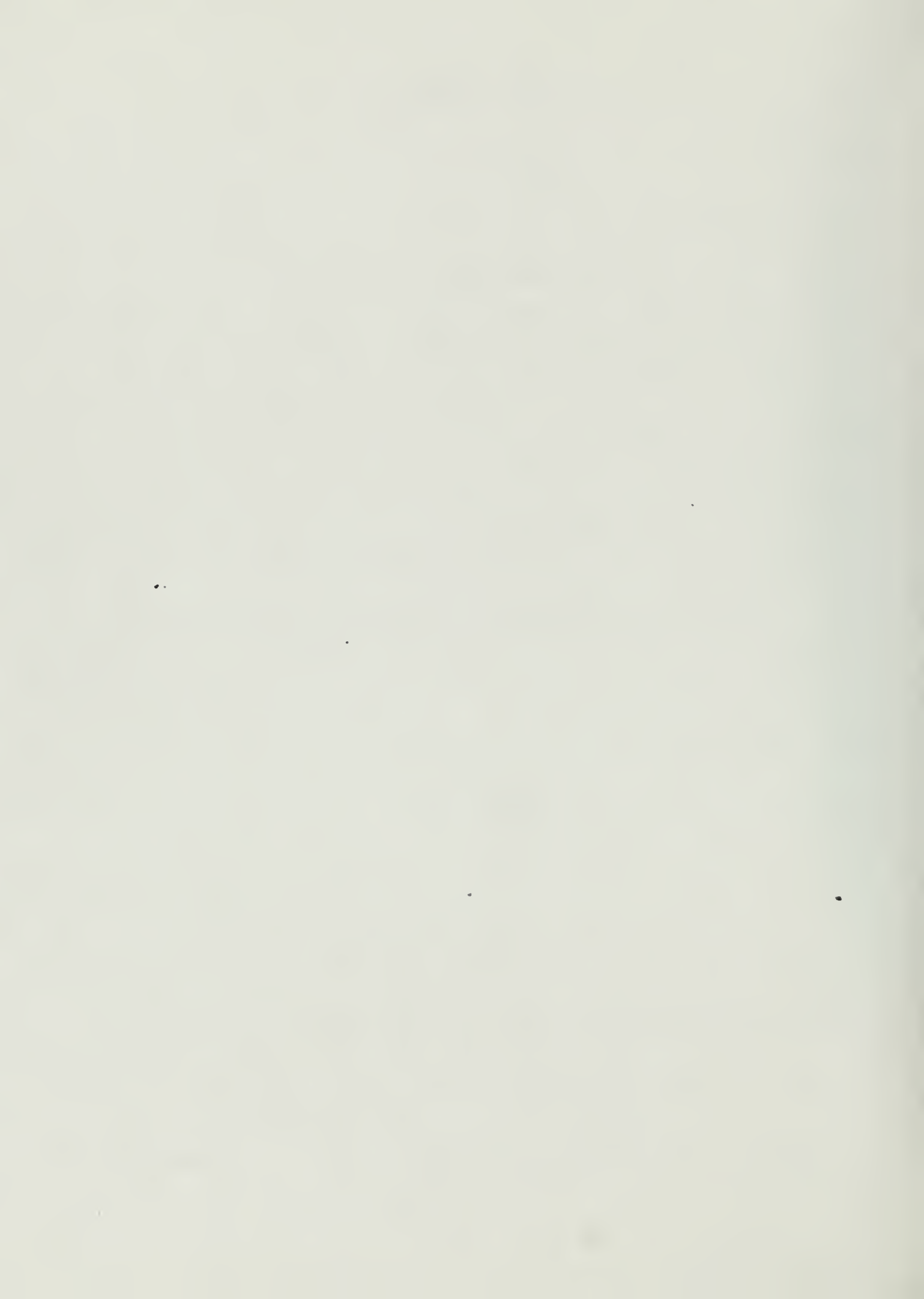
Dilip Chhajed¹

Timothy J. Lowe²

¹Department of Business Administration, University of Illinois at Urbana-Champaign, 1206 South Sixth Street, Champaign, IL 61820

²Department of Management Science, University of Iowa, Iowa City, IA 52242


*We would like to acknowledge the helpful comments of Charles Blair, George Monahan, and Rich Wong.



M-median and M-center Problems with Mutual Communication: Solvable Special Cases

Abstract

In this paper we consider the network version of the m-median problem with mutual communication (MMMC). We reformulate this problem as a graph theoretic node selection problem defined on a special graph. We give a polynomial time algorithm to solve the node selection problem when the flow graph (graph denoting the interaction between pairs of new facilities in MMMC) has a special structure. We also show that with some modification in the algorithm for MMMC, the m-center problem with mutual communication can also be solved when the flow graph has special structure.



Digitized by the Internet Archive
in 2011 with funding from
University of Illinois Urbana-Champaign

1. Introduction

The network version of the m -median problem with mutual communication (MMMC) is to find the location of m new facilities on a network such that the sum of a.) the fixed location cost of each new facility, b.) the cost of interaction between the new facilities and n existing facilities on the network, and c.) cost of interaction between pairs of new facilities is minimized. The existing facilities are located at nodes of the network and the interaction cost between a pair of facilities is a function of the network distance between the facilities. We call the network on which the new facilities are to be located the *transport network*, τ .

An application of MMMC is the location of several new machine centers in a production area. Material movements are made on a transport network (e.g. network of aisles). Each new machine center will send and/or receive material to/from one or more existing machine centers whose locations on the transport network are known. In addition, each new machine will have material flow interaction with some subset of the other new machines. We assume that the existing machines are located at nodes of the transport network. There is no loss of generality here, since as long as each existing machine is on the network, its location can be declared as a node. We consider problems where the set of possible locations on the network for each new facility is finite. We can also declare these locations as nodes of the network. We also allow for the possibility that the fixed cost (cost term a.) above) of locating a new facility is dependent upon its location.

In the above machine location example of MMMC (as well as other examples) it is most likely the case that the cost of interaction between certain pairs of facilities will not depend upon the network distance between their locations. This would occur in the above example if there was no material flow between a pair of facilities. In what follows, we say a pair of facilities *interacts* only if the cost of interaction is a function of the network distance between the facilities.

Two graphs can be defined which represent the interaction structure. The *demand graph*, D , is a bipartite graph with node partition $\{M, N\}$ where set M consists of m nodes

corresponding to the new facilities and set N consists of n nodes corresponding to the existing facilities, and with $u \in M$ and $i \in N$, nodes u and i in D are adjacent if and only if new facility u and existing facility i interact. The *flow graph*, G , consists of m nodes, one corresponding to each new facility. Two nodes in a flow graph are adjacent if and only if the corresponding new facilities interact.

We note that if the demand graph D has no arcs, then (MMMC) is solved by co-locating the n new facilities at a single point on τ which minimizes the sum of the fixed location costs for the new facilities. On the other hand if the flow graph, G , has no edges, then (MMMC) decomposes into n single facility location problems on τ . The interesting cases of (MMMC) occur when both D and G are non-trivial.

Most of the literature associated with (MMMC) deals with the case where there are no fixed location cost (a.) above), and where the interaction costs are linear in network distances. Kolen (1982) has shown that the problem is NP-hard, when τ is a general network, but is polynomially solvable when τ is a tree. Picard and Ratliff (1978) also give a polynomial time algorithm for the problem when τ is a tree. Dearing, Francis, and Lowe (1976) have shown that the problem is a convex optimization problem for all data choices if and only if τ is a tree. Erkut, Francis, Lowe, and Tamir (1989) consider a constrained version of the problem and make use of separation conditions (Francis, Lowe, and Ratliff, 1978) to obtain a mathematical program. The mathematical program is equivalent to the original problem if τ is tree; otherwise the solution to the mathematical program provides a lower bound. A computational study of the lower bound vis-a-vis the original problem is given in Erkut, Francis, and Lowe (1988).

Xu, Francis, and Lowe (1988) consider the version of (MMMC) where there are no fixed location costs, and the transport network, τ , is not necessarily a tree, but τ does contain two or more blocks (maximal, nonseparable subgraphs of τ). They show that by solving a related problem on a "blocking graph" (which is a tree), information can be obtained which localizes each optimal new facility to some vertex or block of τ . The problem then decomposes into a collection of independent problems, one for each localizing block of τ .

In this paper we give a polynomial time algorithm for a special class of network

MMMC problems in which the transport network, τ , and the demand graph, D , are general; the interaction costs are general functions of network distances as long as these cost functions are such that node optimality conditions hold, i.e. there is at least one optimal solution in which each new facility is located at a node of the transport network; and the flow graph, G , is series-parallel. If the node optimality property is not valid then our model is useful when a node restricted solution is sought.

We also consider the m -center problem with mutual communication (MCMC) and show that the algorithm presented for MMMC can be modified to solve special cases of MCMC.

The plan of the rest of the paper is as follows. In Section 2 we present a formulation of the problem. Then we introduce a graph theoretic Node Selection Problem (NSP) and show how our formulation of MMMC can be transformed to an NSP. In Section 4 we define a series-parallel graph and give an algorithm to solve the NSP when the flow graph is series-parallel. In Section 5 we give a detailed example showing an application of the algorithm. Section 6 extends our analysis to the m -center problem with mutual communication. We close with some concluding remarks.

2. Formulation

For clarity of presentation, we will present the formulation where the interaction costs are linear in network distances. However, our complete methodology is applicable for general interaction cost functions of network distances as long as the node optimality property is valid. In addition, for presentation purposes, in this Section we assume that each new facility could feasibly be located at any node of τ . The following notation is used in the formulation.

Notation:

- a_{ui} : interaction cost per unit distance between new facility u and existing facility i
- b_{uv} : interaction cost per unit distance between new facilities u and v

$d(k,r)$:	distance between existing facilities k and r computed on the transport network
D :	demand graph
f_{uk} :	fixed cost of locating facility u at node k
G :	flow graph $(V(G), E(G))$. $V(G) = \{1, \dots, m\}$, $E(G) = \{(u,v) : \text{new facilities } u \text{ and } v \text{ interact, i.e } b_{uv} > 0\}$
M :	set of new facilities
m :	number of new facilities
N :	set of existing facilities
n :	number of existing facilities $= V(\tau) $
p, k, r, i :	indices of existing facilities
τ :	transport network
u, v :	indices of new facilities
x_{uk} :	$\{0,1\}$ variable which takes value 1 if and only if new facility u is located at existing facility k

As mentioned in the introduction, we assume the problem is such that the node optimality condition holds or the solution sought is node restricted. Thus each new facility has to be located at one of the nodes of τ . To avoid notational difficulties, we assume without loss of generality that each node of τ is the site of an existing facility. If in an application of (MMMC) there is no existing facility at some node, we assume the data consists of a dummy existing facility at the node which has no interaction with any of the new facilities.

The condition that an arbitrary new facility u must be located at a node of τ (site of an existing facility) can be represented by the constraint

$$\sum_{k \in N} x_{uk} = 1 \quad \forall u \in M. \quad (1)$$

Given that (1) holds, and the x_{uk} 's are 0,1 variables, (2), (3), and (4) below are valid. The interaction cost between new facility u and existing facility i is given by

$$a_{ui} \sum_{k \in N} d(i,k) x_{uk}. \quad (2)$$

The cost of interaction between two new facilities, u and v , is given by

$$b_{uv} \sum_{r \in N} \sum_{k \in N} x_{uk} x_{vr} d(k,r), \quad (3)$$

and finally, the fixed cost of locating new facility u is

$$\sum_{k \in N} f_{uk} x_{uk}. \quad (4)$$

Summing (2) over all pairs of new and existing facilities, (3) over all pairs of existing facilities, and (4) over all new facilities we get the total cost:

$$\begin{aligned} & \sum_{u \in M} \sum_{i \in N} a_{ui} \sum_{k \in N} d(i,k) x_{uk} + \sum_{u \in M} \sum_{v(>u) \in M} b_{uv} \sum_{r \in N} \sum_{k \in N} x_{uk} x_{vr} d(k,r) \\ & + \sum_{u \in M} \sum_{k \in N} f_{uk} x_{uk}. \end{aligned} \quad (5)$$

The first and the last term of (5) can be combined to give

$$\begin{aligned} & \sum_{u \in M} \sum_{k \in N} \{ [\sum_{i \in N} a_{ui} d(i,k)] + f_{uk} \} x_{uk} \\ & = \sum_{u \in M} \sum_{k \in N} F_{uk} x_{uk}, \text{ where } F_{uk} = [\sum_{i \in N} a_{ui} d(i,k)] + f_{uk}. \end{aligned}$$

Thus, F_{uk} is the sum of the fixed cost of locating new facility u at k and the cost of satisfying the customer demand for new facility u from this location. If we denote $C'_{ukvr} = b_{uv}d(k,r)$ then our integer programming formulation of MMMC is:

$$(P) \quad \min \sum_{u \in M} \sum_{v(>u) \in M} \sum_{r \in N} \sum_{k \in N} C'_{ukvr} x_{uk} x_{vr} + \sum_{u \in M} \sum_{k \in N} F_{uk} x_{uk} \quad (6)$$

Subject to: (1),

$$x_{uk} = \{0,1\} \text{ for all } u,k.$$

The objective function of (P) consists of a quadratic term and a linear term in variables x . The problem can be reformulated in which there is only a quadratic term in the objective function. To do so, first we select a new facility u° for each new facility u such that there is an interaction between new facilities u and u° (We are assuming, without loss of generality, that the flow graph is connected). Then we define

$$C_{ukvr} = F_{uk} + C'_{ukvr}, \quad \forall r \text{ if } v = u^\circ, \text{ and } C_{ukvr} = C'_{ukvr} \text{ otherwise.} \quad (7)$$

With these costs, consider the following problem, which we call the *Quadratic Location Problem* (QLP):

$$\text{Min } \sum_u \sum_{v>u} \sum_k \sum_r C_{ukvr} x_{uk} x_{vr} \quad (8)$$

Subject to: (1),

$$x_{uk} = \{0,1\} \text{ for all } u,k.$$

We now show that problem (P) can be converted to a QLP.

Lemma 1: (P) can be formulated as a QLP.

Proof: (8) is same as: $\sum_u \sum_{v>u, v \neq u} \sum_k \sum_r C_{ukvr} x_{uk} x_{vr} + \sum_u \sum_k \sum_r C_{uku^o r} x_{uk} x_{u^o r}$

$$\begin{aligned} \text{Using (7):} \quad &= \sum_u \sum_{v>u, v \neq u} \sum_k \sum_r C'_{ukvr} x_{uk} x_{vr} + \sum_u \sum_k \sum_r (C'_{uku^o r} + F_{uk}) x_{uk} x_{u^o r} \\ &= \sum_u \sum_{v>u} \sum_k \sum_r C'_{ukvr} x_{uk} x_{vr} + \sum_u \sum_k \sum_r F_{uk} x_{uk} x_{u^o r} \\ &= \sum_u \sum_{v>u} \sum_k \sum_r C'_{ukvr} x_{uk} x_{vr} + \sum_u \sum_k F_{uk} x_{uk} \sum_r x_{u^o r} \end{aligned}$$

$$\text{By (1):} \quad = \sum_u \sum_{v>u} \sum_k \sum_r C'_{ukvr} x_{uk} x_{vr} + \sum_u \sum_k F_{uk} x_{uk} = (6). \llcorner$$

Hence an MMMC can be reformulated as a QLP by redefining the costs as in (7). Note that the QLP is a relaxation of the Quadratic Assignment Problem (QAP). In the QLP, every facility has to be assigned to one site, but at any site multiple facilities (unlike QAP) may be assigned. Thus, in the QAP there is an additional set of constraints, $\sum_u x_{uk} = 1, \forall k$, which is not in QLP.

Kolen has proven that MMMC is NP-hard. Since (as outlined above) MMMC can be reformulated as a QLP, it follows that QLP is also NP-hard. We provide an alternate proof of the complexity of the QLP by reducing a Simple Max Cut problem to QLP. We note that the equivalence between quadratic 0-1 optimization and the (weighted) max cut problem has been demonstrated in Barahona (1986).

Problem SIMPLE MAX CUT (SMC): Given a graph G with nodes $V(G)$ and arcs $E(G)$, find a partition of $V(G)$ into two disjoint sets V_1 and V_2 such that the number of arcs from $E(G)$ that have one endpoint in V_1 and one endpoint in V_2 is maximum.

Theorem 1: QLP is NP-hard.

Proof: As mentioned above, we establish this by showing that (SMC) can be reduced to QLP. Given an (SMC) problem on a graph G , we construct a QLP by defining variables:

$$x_{u1} = 1 \text{ if node } u \text{ is in } V_1, 0 \text{ otherwise, for all } u \in V(G), \text{ and}$$

$$x_{u2} = 1 \text{ if node } u \text{ is in } V_2, 0 \text{ otherwise, for all } u \in V(G).$$

We also set $C_{ukvr} = 1$ for $k=r=1$ or 2 , and arc (u,v) is in $E(G)$; and $C_{ukvr} = 0$ otherwise.

Writing the QLP for this, we get:

$$(QLP(G)) \quad \text{Min } \sum_{u,v \in M} \sum_{k=1,2} \sum_{r=1,2} C_{ukvr} x_{uk} x_{vr} \quad (T1)$$

$$\text{Subject to: } \sum_{k=1,2} x_{uk} = 1, \forall u \in V(G) \quad (T2)$$

$$x_{uk} \in \{0,1\}. \quad (T3)$$

Given a graph G and a feasible solution $\{\hat{x}_{uk}\}$ to $(QLP(G))$ with value $Z(\hat{x}_{uk})$ we can construct a feasible solution to (SMC) problem with value $|E(G)| - Z(\hat{x}_{uk})$ as follows: Form sets of nodes $V_1 = \{u : \hat{x}_{u1} = 1\}$ and $V_2 = \{u : \hat{x}_{u2} = 1\}$. Note that because $\{\hat{x}_{uk}\}$ satisfies $(T2)$, only one of \hat{x}_{u1} or \hat{x}_{u2} will be equal to one and so $V_1 \cap V_2 = \emptyset$. Also, $V_1 \cup V_2 = V(G)$.

For a fixed u° and v° , the term $C_{u^\circ k v^\circ r} \hat{x}_{u^\circ k} \hat{x}_{v^\circ r}$ is equal to one if and only if $k = r$, $\hat{x}_{u^\circ k} = 1$, $\hat{x}_{v^\circ r} = 1$, and $(u^\circ, v^\circ) \in E(G)$. If $C_{u^\circ k v^\circ r} \hat{x}_{u^\circ k} \hat{x}_{v^\circ r} = 1$ then by definition, both u° and v° will be either in V_1 or in V_2 . Thus the number of arcs of G with both endpoints either in V_1 or in V_2 is $Z(\hat{x}_{uk})$. Hence the number of arcs with one node in V_1 and other in V_2 is $|E(G)| - Z(\hat{x}_{uk})$. Minimizing $(QLP(G))$ is same as minimizing the number of arcs with their endpoints in same sets V_1 or V_2 , and hence same as maximizing the number of arcs which have their endpoints in different sets, i.e. maximizing the cut. «»

In formulation (P) of $MMMC$, we have assumed that each new facility could be located at any node of τ . Suppose the location of a new facility u must be at a member of some subset of the nodes of τ , denoted by N_u . In formulation (P) , we could make use of an artificial cost by setting the costs f_{uk} for each node $k \notin N_u$ to be arbitrarily large. The artificial costs would prohibit new facility u from being located at a node outside N_u in an optimal solution to $MMMC$.

In the next Section, we reformulate $MMMC$ as a *node selection problem*, NSP . We assume that N_u is known for each new facility u , however, in our reformulation we will not need to make use of the artificial fixed location costs mentioned above.

3. Node Selection Problem

We now define a special graph, which we call a G -partite graph. We also define an optimization problem, the *node selection problem*, on the G -partite graph. Subsequently, in this Section, we show that the QLP can be posed as a node selection problem on a suitably defined G -partite graph.

Definition: Given a graph G with an integer $|N_v|$ associated with each node $v \in V(G)$, a G -Partite Graph, G° , is formed as follows: Corresponding to each node $v \in V(G)$ we create a *node-family*, σ_v , in G° consisting of $|N_v|$ nodes $\{v_k : k = 1, \dots, |N_v|\}$ (Figure 1).

Two nodes u_k and v_r ($v \neq u$) are *adjacent* in G° if and only if $\text{arc}(v, u) \in E(G)$. Arc (u_k, v_r) in G° is assigned weight ω_{kr}^{uv} . Thus if (v, u) exists in G , nodes of node-family σ_u and node-family σ_v form a complete bi-partite subgraph of G° . Node-families σ_u and σ_v are said to be *adjacent* if and only if every node in σ_u is adjacent with every node in σ_v . By *joining* two node-families σ_u and σ_v we mean adding arcs between all pairs of nodes $(\{u_k, v_r\} : u_k \in \sigma_u, v_r \in \sigma_v)$. We will use the notation (σ_u, σ_v) to denote all arcs between nodes in σ_u and nodes in σ_v .

A G -partite graph is a generalization of a complete bi-partite graph. A complete bi-partite graph is a G -partite graph corresponding to a graph G which is a single arc. Given a G -partite graph G° , let $S(G^\circ)$ be an induced subgraph of G° with one node of each node-family and let $Z(S(G^\circ))$ be the sum of the weights on the arcs in $S(G^\circ)$. We now pose the following problem on G° :

(NSP) Node Selection Problem: Given a graph G and the corresponding G -partite graph, G° , with arc weights ω , find an induced subgraph consisting of exactly one node of each node-family; such that the sum of all the arc weights on the arcs in the induced subgraph is minimum. We will denote the node selection problem on G° by $\text{NSP}(G^\circ)$ and an optimal solution by $S^*(G^\circ)$.

Lemma 2: Problem MMMC is reducible to NSP.

Proof: Given an MMMC problem with n existing facilities and m new facilities, we first

formulate it as a QLP. Then we create a G -partite graph, G° , as follows. Create a node family σ_u with $|N_u|$ nodes for every new facility u . Each of the $|N_u|$ nodes in σ_u corresponds to a node of τ where new facility u can be located. Join two node-families, σ_u and σ_v if and only if $(u,v) \in G$. Recall that joining two node-families means forming a complete bi-partite graph between nodes of σ_u and σ_v . Assign the weight C_{ukvr} , on the arc $(u_k, v_r) \in E(G^\circ)$.

Given the G -partite graph constructed as above, we note that a feasible solution to the NSP corresponds to a feasible solution to the QLP. Given a feasible solution $S(G^\circ)$ to the NSP, construct a feasible solution to QLP, $X(S(G^\circ))$, as follows: set $X(S(G^\circ)) = \{x_{uk} = 1 \text{ if node } k \text{ of node-family } \sigma_u \text{ is chosen, } 0 \text{ otherwise}\}$. Since one node of each node-family is in $S(G^\circ)$, constraint (1) is satisfied. Similarly if a feasible solution to the QLP is given we can construct a feasible solution to NSP by choosing node $u_k \in \sigma_u$ if and only if $x_{uk} = 1$.

A choice of nodes for department u and v contributes $C_{ukvr} = \omega_{kr}^{uv}$ to the objective function of QLP, which is the weight on arc (u_k, v_r) in $S(G^\circ)$. Hence the value of $S(G^\circ)$ is the same as the value of solution $X(S(G^\circ))$. Thus by solving appropriate NSP we will get a solution to the QLP.«»

The above reduction shows that NSP is also NP-hard. Note that an NSP is more general than QLP and many other problems can be modeled as an NSP (Section 7).

We have shown that, given an MMMC, it can be modeled as a QLP which in turn can be reformulated as an NSP. Thus, an MMMC can be modeled as an NSP. The G -partite graph for the MMMC will correspond to the flow graph G of MMMC. This is because, if an arc $(u,v) \notin E(G)$ then $b_{uv} = 0$ and so $C_{ukvr} = 0$ for all $k,r \in N$. And by (7) $C_{ukvr} = 0$ for all $k,r \in N$. Thus node-families σ_u and σ_v will not be adjacent.

4. Polynomially Solvable Special Cases

In this Section we give a rich class of problems for which the node selection problem is polynomially solvable. We have shown that we can use the node selection

problem, NSP, to model MMMC. As we have shown, in general this problem is NP-complete. The special cases we consider are characterized by the structure of the flow graph, G . For the node selection problem, we are given G and the G -partite graph G° . We associate, with each node and each arc of G° a label in the form of a set. Initially we set the label of each arc e , $L_a(e) = \{\}$, where $\{\}$ denotes the empty set, and the label of each node u_k , $L_n(u_k) = \{u_k\}$. We will represent the label of an arc e defined by two nodes p and q by $L_a(p,q)$ rather than $L_a((p,q))$.

Later on, in the course of solving the Node Selection Problem for the special cases, arcs and nodes of graphs G and G° will be deleted, in some cases (new) arcs will be added, and labels of the remaining arcs, as well as the arc weights, will be modified to reflect the change. The labels are used, basically, to carry pertinent information about the deleted portion of the graph. In modifying the labels, we will typically add two labels, where addition of labels is defined as the set union operation on the sets corresponding to the two labels. In what follows, we assume that the flow graph is connected. If this is not the case, e.g. the flow graph consists of more than one component, then MMMC can be solved by solving (independently) an MMMC problem corresponding to each component. We begin with some definitions and results from graph theory:

Definition: A graph is *series-parallel* (Richey, 1989) if it can be reduced to an arc by repeated application of the following operations:

(G1) *Series Reduction:* Replace any degree-2 node q , and the incident arcs (u,q) and (v,q) , $u \neq v$, by a new arc, $a'(u,v)$, incident to u and v .

(G2) *Cut Reduction:* If q is a pendant node (a node of degree one) adjacent to node u , find a node $v \neq q$ adjacent to u , delete node q and add a new arc $a'(u,v)$.

(G3) *Parallel Reduction:* Replace two arcs e and f which are both incident to nodes u and v , by a new arc, g , incident to u and v .

The new arcs that are added to the graph in the above operations are named pseudo-arcs in (Richey, 1989). Richey describes an operation similar to operation (G2), calling it a Jackknife reduction, but does not add the new arc $a'(u,v)$. If we perform parallel reduction on (u,v) immediately after the cut reduction, we get Richey's Jackknife reduction. Thus,

although there is a minor difference in the definition of one operator, which we need for our algorithm, the above definition of a series-parallel graph is identical to that of Richey.

To obtain insight into the structural nature of series-parallel graphs it is useful to introduce the concept of a *2-tree*. A 2-tree (Wald and Colbourn, 1983; Rardin, Parker, and Richey, 1982) is defined recursively as follows: A triangle is a 2-tree. Given any arc (x,y) of a 2-tree, by appending a node z and adding edges (x,y) and (y,z) , the resulting graph is also a 2-tree. The relationship between series-parallel graphs and 2-trees is that (Wald and Colbourn, 1983) a series-parallel graph, without loops and parallel edges, is a subgraph of some 2-tree. Thus, series-parallel graphs are sometimes called *partial 2-trees*.

Several authors have exploited the properties of series-parallel graphs. Takamizawa, Nishizeki, and Saito (1982) have solved several combinatorial problems on series-parallel graphs. Barahona (1986) has solved the 0-1 quadratic programming problem when the graph representing the positive coefficients of the problem, i.e., the flow graph, is series-parallel. Rendl (1986) solved a special case of the QAP by exploiting a series-parallel digraph. Rardin, Parker, and Richey (1982) and Wald and Colbourn (1983) solved the Steiner tree problem on a graph which is series-parallel, and Richey (1989) has solved several location problems on transport networks which are series-parallel.

It is shown in Rardin, Parker, and Richey (1982) that series-parallel graphs subsume circuits, outerplanar graphs, cactus graphs, and trees. Thus series-parallel graphs generalize a number of graph types; however, they still form a subset of planar graphs.

We will soon define graph operations on a G -partite graph which are similar to the operations (G1), (G2), and (G3) discussed above. The outcome of two of these operations will result in parallel arcs in the graph. We emphasize here that if there are parallel arcs between two given nodes of a G -partite graph G° , and if this pair of nodes is in a feasible solution $S(G^\circ)$ to NSP, then the parallel arcs are also in $S(G^\circ)$, so that the arc weights of both arcs contribute to $Z(S(G^\circ))$.

Let ψ be an arbitrary subset of nodes of a G -partite graph G° such that there is at most one node of each node-family in ψ . Let $\text{NSP}(G^\circ, \psi)$ denote the constrained version of NSP on G° with the set of nodes ψ fixed, and let $S^*(G^\circ, \psi)$ denote an optimal solution to $\text{NSP}(G^\circ, \psi)$ with objective function value $Z(S^*(G^\circ, \psi))$. Thus with $\{\}$ denoting the empty

set, $S^*(G^\circ, \{\})$ is a solution to $NSP(G^\circ, \{\}) = NSP(G^\circ)$.

We now define three elementary operations on a G-partite graph, G° . These operations are mirror images of similar operations performed on G so that the new G-partite graph corresponds to G after the elementary operation. Although we give the same name to these operations as in the case of G , the context will make it clear which procedure we are referring to. We present these operations as procedures after describing the essence of what they accomplish. These procedures will be repeatedly used by the main algorithm SP which we will describe later. In Section 5 we give an example which uses each of these procedures to solve the NSP using algorithm SP. The reader may refer to Section 5 while going through these procedures to clarify the steps in each procedure.

(GP1): Series-Reduction: In this process a node-family σ_q such that node q is adjacent to exactly two distinct nodes u and v of G , where $q \neq v \neq u$, is eliminated in G° . The reduced graph has one less node-family. All the information pertinent to optimal node selection in σ_q is retained, as is shown in the following process. For an example of this procedure see Iteration 3 in Section 5.

PROCEDURE SR(σ_q)

Step 1: Let u and v be the two nodes adjacent to node q in G , where $q \neq u \neq v$.

Step 2 For each pair of nodes $u_k \in \sigma_u, v_r \in \sigma_v$, find q_{p° giving

$$\omega_{kp^\circ}^{uq} + \omega_{rp^\circ}^{vq} = \min_{q_p \in \sigma_q} \{ \omega_{kp}^{uq} + \omega_{rp}^{vq} \} \text{ (ties can be arbitrarily broken).}$$

Add an arc $a'(u_k, v_r)$ with weight equal to $\omega_{kp^\circ}^{uq} + \omega_{rp^\circ}^{vq}$ and

let the label of this new arc be $L_{a'}(v_r, u_k) \leftarrow L_a(v_r, q_{p^\circ}) \cup L_a(u_k, q_{p^\circ}) \cup L_n(q_{p^\circ})$.

Step 3: Delete node-family σ_q . Return (to the calling algorithm).

Lemma 3: For a G-Partite graph G° with a node-family σ_q such that q has degree two in G , let \underline{G}° and \underline{G} be the results of series-reducing node-family σ_q and node q in G° and G , respectively. Given an optimal solution $S^*(\underline{G}^\circ)$ to $NSP(\underline{G}^\circ)$, an optimal solution to $NSP(G^\circ)$ can be constructed using the nodes and arc labels of $S^*(\underline{G}^\circ)$. Furthermore, the complexity of procedure SR(.) is $O(|N_u| * |N_v| * |N_q|)$ where u and v are the nodes of G

adjacent to node q .

Proof: Let u and v be the two nodes adjacent to node q in G and let A' denote the set of arcs added to \underline{G}° in procedure $SR(\cdot)$. For any choice $u_k \in \sigma_u$ and $v_r \in \sigma_v$, we note that with $\psi = \{u_k, v_r\}$, a) $S^*(\underline{G}^\circ \setminus A', \psi)$, which is an optimal solution to $NSP(\underline{G}^\circ \setminus A', \psi)$, is also an optimal solution to $NSP(\underline{G}^\circ \setminus A', \psi)$, and b) $S^*(\underline{G}^\circ \setminus A', \psi)$ has the same objective function value when evaluated in graphs $G^\circ \setminus \sigma_q$ and $\underline{G}^\circ \setminus A'$. Observations a) and b) follow since the graphs $G^\circ \setminus \sigma_q$ and $\underline{G}^\circ \setminus A'$ are the same.

Considering $NSP(G^\circ)$, we note that for fixed $u_k \in \sigma_u$ and $v_r \in \sigma_v$, the choice of an optimal node in σ_q is independent of the optimal choices in the graph $G^\circ \setminus \{\sigma_u \cup \sigma_v \cup \sigma_q\}$. Thus, one way to solve $NSP(G^\circ)$ is to find

$$\min_{u_k \in \sigma_u, v_r \in \sigma_v} \{ Z(S^*(G^\circ \setminus \sigma_q, \{u_k, v_r\})) + \min_{q_p \in \sigma_q} \{ \omega_{kp}^{uq} + \omega_{rp}^{vq} \} \}.$$

In graph \underline{G}° , for $u_k \in \sigma_u$, $v_r \in \sigma_v$, the arc $a'(u_k, v_r)$ carries the essential information (labels and value) of the inner minimization problem above. Letting $w(a'(u_k, v_r))$ denote the weight on arc $a'(u_k, v_r)$ in \underline{G}° , we note that $NSP(\underline{G}^\circ)$ can be solved by solving

$$\min_{u_k \in \sigma_u, v_r \in \sigma_v} \{ Z(S^*(G^\circ \setminus \sigma_q, \{u_k, v_r\})) + w(a'(u_k, v_r)) \}.$$

The first part of the Lemma now follows.

The complexity of finding node q_p in Step 2 of $SR(\cdot)$ is $O(|N_q|)$. This Step is repeated $|N_u| * |N_v|$ times, hence the complexity of the procedure is $O(|N_u| * |N_v| * |N_q|)$. «»

(GP2) Cut Reduction: Given two node-families σ_q and σ_u such that node q is a pendant node in G , $(q, u) \in E(G)$, and (q, u) is not the only arc of G , we delete node-family σ_q and add parallel arcs to the G -partite graph. The procedure which follows sketches the steps. The example in Section 5 uses $CR(\cdot)$ during Iteration 1.

PROCEDURE $CR(\sigma_q, \sigma_u)$

Step 1: With q a pendant node of G adjacent to u , select an arc (u, v) of G , $v \neq q$. (Such an arc exists because we have assumed that (u, q) is not the only arc of G and throughout we assume that G is connected.)

Step 2: For each node u_k of σ_u ,

Find a node q_p of node-family σ_q such that $\omega_{kp}^{uq} = \min_{q_p \in \sigma_q} \{ \omega_{kp}^{uq} \}$ (ties

can be arbitrarily broken).

In G° add new arcs $a'(u_k, v_r)$ for all $v_r \in \sigma_v$ with weight ω_{kp}^{uq} and set the label of these new arcs $L_{a'}(u_k, v_r) \leftarrow L_n(q_p^\circ) \cup L_a(q_p^\circ, u_k)$.

Step 3: Delete all the nodes of node-family σ_v in G° , i.e. delete σ_v .

Step 4: Return.

Lemma 4: For a G-Partite graph G° with a node-families σ_u and σ_q such that $(u, q) \in E(G)$, node q is a pendant node in G , and u and q are not the only nodes of G , let \underline{G}° and \underline{G} be the results of cut-reducing node-families σ_u and σ_q in G° and nodes u and q in G . Given an optimal solution $S^*(\underline{G}^\circ)$ to $\text{NSP}(\underline{G}^\circ)$, an optimal solution to $\text{NSP}(G^\circ)$ can be constructed using nodes and arc labels of $S^*(\underline{G}^\circ)$. Also, reducing G° to \underline{G}° can be done in $O(|N_u| * (|N_v| + |N_q|))$ time where v is the node selected in Step 1.

Proof: Let A' denote the set of new arcs added between nodes in σ_u and σ_v . Consider an arbitrary node $u_k \in \sigma_u$ in G° . Since graphs $G^\circ \setminus \sigma_q$ and $\underline{G}^\circ \setminus A'$ are the same with $\psi = \{u_k\}$, it follows that $S^*(\underline{G}^\circ \setminus A', \psi)$, an optimal solution to $\text{NSP}(\underline{G}^\circ \setminus A', \psi)$, is also an optimal solution to $\text{NSP}(G^\circ \setminus \sigma_q, \psi)$ and $S^*(\underline{G}^\circ \setminus A', \psi)$ has the same objective function value when evaluated in graphs $G^\circ \setminus \sigma_q$ and $\underline{G}^\circ \setminus A'$.

Also, due to the structure of G° , a way of solving $\text{NSP}(G^\circ)$ is to find $\min_{u_k \in \sigma_u} \{Z(S^*(G^\circ \setminus \sigma_q, \{u_k\})) + \min_{q_p \in \sigma_q} \{\omega_{kp}^{uq}\}\}$. In graph \underline{G}° with $u_k \in \sigma_u$ fixed, there is an added arc $a'(u_k, v_r)$ for every $v_r \in \sigma_v$ and each such arc carries the essential information (labels and value) of the inner minimization problem above. Since any feasible solution to $\text{NSP}(G^\circ)$ and any feasible solution to $\text{NSP}(\underline{G}^\circ)$ must contain one node of σ_v , the first part of the Lemma follows.

In Step 2 of $\text{CR}(\cdot)$, node q_p° can be found in $O(|N_q|)$ time and arcs $a'(u_k, v_r) \forall v_r \in \sigma_v$, can be updated in $O(|N_v|)$ time. This process is repeated for each node in σ_u , i.e. $|N_u|$ times. Thus the total complexity of this Step is $O(|N_u| * (|N_v| + |N_q|))$. «»

(GP3) **Parallel Reduction:** Initially in G there are no parallel arcs. However, as the main algorithm SP (described later) proceeds, parallel arcs may be created in G and G° . In particular, parallel arcs in G (G°) may be created during series-reduction of node q (σ_q) with adjacent nodes u and v (σ_u and σ_v). Series-reduction would add an arc between u and

v in G (add arcs (σ_u, σ_v) between σ_u and σ_v in G°) and if u and v (σ_u and σ_v) were adjacent before the series-reduction then there would be parallel arcs between nodes u and v (σ_u and σ_v) after the series-reduction. Parallel arcs are always created during cut-reduction. For an illustration of this procedure, see Iteration 2 of the example in the next Section.

Given two node-families σ_u and σ_v such that there are two arcs between every node $u_k \in \sigma_u$ and $v_r \in \sigma_v$, we replace the parallel arcs by a single arc. The weights and the labels associated with the two parallel arcs are added and they form the weight and label, respectively of the new arc. Procedure $PR(\cdot)$ describes this process.

PROCEDURE $PR(\sigma_u, \sigma_v)$

Step 1: Let nodes $u_k \in \sigma_u$ and $v_r \in \sigma_v$ be such that there are two arcs between them.

Delete one of these arcs and add its weight to the weight of the other arc. Also add the label of this deleted arc to the label of the second arc.

Step 2: Continue Step 1 until no parallel arcs between nodes of σ_u and σ_v remain.

Step 3: Return.

Lemma 5: For a G -Partite graph G° with node-families σ_u and σ_v such that there are two arcs between every node $u_k \in \sigma_u$ and $v_r \in \sigma_v$, let \underline{G}° and \underline{G} be the results of parallel-reducing node-families σ_u and σ_v of G° and nodes u and v in G . Given an optimal solution $S^*(\underline{G}^\circ)$ to $NSP(\underline{G}^\circ)$, an optimal solution to $NSP(G^\circ)$ can be constructed using the nodes and arc labels of $S^*(\underline{G}^\circ)$. Furthermore, $PR(\cdot)$ can be performed in $O(|N_u|*|N_v|)$ time.

Proof: The first part of the Lemma easily follows from the fact that for each $u_k^\circ \in \sigma_u$ and $v_r^\circ \in \sigma_v$, the weight and label on arc $(u_k^\circ, v_r^\circ) \in \underline{G}^\circ$ is equal to the sum of the weights and the union of the labels on the parallel arcs $(u_k^\circ, v_r^\circ) \in G^\circ$, and the fact that graphs $\underline{G}^\circ \setminus (\sigma_u, \sigma_v)$ and $G^\circ \setminus (\sigma_u, \sigma_v)$ are the same.

Since there are $|N_u|$ nodes in σ_u and $|N_v|$ nodes in σ_v , there will be $|N_u|*|N_v|$ repetitions of Step 1. Each repetition of Step 1 takes constant time and so the complexity of the procedure is as stated. «»

We now present an algorithm which correctly solves NSP in polynomial time when

the flow graph G is series-parallel. Without loss of generality we assume that G is connected. In the algorithm, during Iteration 1, graphs G and G° are denoted as G_1 and G°_1 , respectively. During each iteration these graphs will be changed and at a general Iteration k , these graphs will be denoted by G_k and G°_k . Algorithm SP proceeds by forming a set, denoted as $2D$, of degree two nodes in G_1 . PA is the set of node pairs having parallel arcs in the current G_k . Initially PA is empty. First we cut-reduce the pendant nodes in G_k and G°_k followed by parallel-reduction of the parallel arcs formed during the cut-reduction. Then a series-reduction is performed (if $2D \neq \{\}$) followed by a parallel-reduction, if parallel arcs are formed due to the series-reduction. If these reductions create a new pendant node, that node is eliminated by cut-reduction and the process is continued until a single arc is left in G_k . The best arc in G°_k at this stage is chosen, the weight of which gives the solution value. The solution to NSP in G° is generated by the end nodes of this arc and the nodes contained in the label of this arc (Step 6).

ALGORITHM SP

Step 0: Set $k \leftarrow 1$, $G^\circ_k \leftarrow G^\circ$, $G_k \leftarrow G$.

Let $2D$ denote the list of nodes in G_k with degree 2. PA is the list of node pairs having parallel arcs in G_k . For our problem, initially PA will be empty as there are no parallel arcs in the flow graph G .

Step 1: If G_k is a single arc then go to Step 6 else find a node $q \in V(G_k)$ with degree one and go to Step 2. If there exists no such node then go to Step 3.

Step 2: Let $(q,u) \in E(G_k)$ be the arc connecting q to another node u .

Cut Reduce node-families σ_u and σ_q in G°_k by calling procedure $CR(\sigma_q, \sigma_u)$.

Cut Reduce nodes u and q in G_k .

Let node v be such that it is adjacent to u in G_k and is used in $CR(\cdot)$. Add (u,v) to PA .

Set $G_{k+1} \leftarrow G_k$ (after cut-reduction)

$G^\circ_{k+1} \leftarrow G^\circ_k$ (after cut-reduction)

$k \leftarrow k+1$.

Go to Step 5.

Step 3: If $|2D| = 0$ then go to Step 5; else choose a node $q \in 2D$. Let nodes u and v be adjacent to q in G_k .

Step 4: If $(u, v) \in E(G_k)$ then add (u, v) to PA.

Series-reduce node-family σ_q by calling procedure $SR(\sigma_q)$.

Series-reduce node q .

Set $G_{k+1} \leftarrow G_k$ (after series-reduction)

$G_{k+1}^\circ \leftarrow G_k^\circ$ (after series-reduction)

$k \leftarrow k+1$

Delete q from $2D$ and go to Step 5.

Step 5: If $|PA| = 0$ then go to Step 1; else let $(u, v) \in PA$.

Parallel-reduce arcs between node-families σ_u and σ_v by calling procedure $PR(\sigma_u, \sigma_v)$.

Parallel-reduce arcs between nodes u and v .

Set $G_{k+1} \leftarrow G_k$ (after parallel-reduction)

$G_{k+1}^\circ \leftarrow G_k^\circ$ (after parallel-reduction)

$k \leftarrow k+1$

If u (or v) has degree 2 in G_k , add it to $2D$.

Go to Step 1.

Step 6: At this stage G is a single arc (u, v) . Find

$\omega_{k^\circ r^\circ}^{uv} = \min_{k \in \sigma_u, r \in \sigma_v} \{ \omega_{kr}^{uv} \}$. This is the value of the optimal solution and the solution can be constructed by $L_a(u_{k^\circ}, v_{r^\circ}) \cup L_n(u_{k^\circ}) \cup L_n(v_{r^\circ})$. Stop.

Theorem 2: Algorithm SP correctly solves NSP in polynomial time when the flow graph G is series-parallel.

Proof: Algorithm SP first cut-reduces all of the pendant nodes of G with each cut-reduction followed by a parallel-reduction. It then performs a series-reduction and a parallel-reduction, if needed. This is followed by a cut-reduction if necessary. This process, if repeated, must reduce G to a single arc if G is series-parallel. The graph obtained after each reduction remains series-parallel. We also perform the corresponding operations on G° . Each of these reductions, by Lemmas 3, 4, and 5, preserves the solution to NSP on the original graph G° . Finally, when only a single arc is left in G , the node

selection problem can be trivially solved as in Step 6 of SP. The end nodes of the single arc selected in Step 6 provide the two nodes to be selected from the node-families that remain in Step 6. The nodes to be selected from other node-families, which were in the original graph G° , are given by the labels on the arc selected in Step 6.

To show the polynomial complexity we note that every cut-reduction and every series-reduction eliminates one node of the flow graph. Furthermore, each such reduction adds at most one parallel arc to the flow graph. If G has β nodes, there cannot be more than β series and cut-reductions and at most β parallel-reductions. Since the complexity of series-reduction dominates the complexity of other two reductions, the complexity of algorithm SP is bounded above by the effort for β series-reductions. If $\lambda = \max_{u \in V(G)} |N_u|$, then this bound is given by $O(\beta * \lambda^3)$. «»

Corollary: The complexity of solving an MMMC problem with m new facilities and n nodes when the flow graph is series-parallel is $O(m * n^3)$. «»

5. An Example of NSP When the Flow Graph is Series-Parallel

Consider the flow graph G and the G -partite graph, G° in Figure 1. Figure 2 shows the weights on the arcs in G° which are presented in the form of matrices. The problem data in Figures 1 and 2 were possibly derived from an instance of MMMC by using equations (2) through (7). We do not illustrate the reduction of MMMC to NSP as doing this is fairly straightforward. Note that in this problem, new facilities a and c are each restricted to one of two locations, and new facilities b and d each have three potential location sites. The entry in row 1 and column 3 of the first of the matrices in Figure 2 is the weight of the arc joining node 1 of σ_c and node 3 of σ_d . In G° , the label on the arc joining nodes u_k and v_r is $L_a(u_k, v_r) = \{ \}$ and the label on each node u_k is $L_n(u_k) = \{ u_k \}$. Now we apply algorithm SP to solve NSP on G° .

Iteration 1:

Step 0: $k \leftarrow 1$, $G^\circ_1 \leftarrow G^\circ$, $G_1 \leftarrow G$.

$2D = \{ a, b \}$.

Step 1: Vertex d in G_1 has degree one. Go to Step 2.

Step 2: d is adjacent to c and (d,c) is not the only arc of G_1 . Call procedure $CR(\sigma_d, \sigma_c)$.

Procedure $CR(\sigma_d, \sigma_c)$

Step 1: Select arc (c,a) of G_1

Step 2: For node c_1 of σ_c

min of $\{5,4,8\}$ is 4 and occurs for node d_2

In G°_1 add arcs $a'(c_1, a_1)$ and $a'(c_1, a_2)$, each with cost 4 and label $\{d_2\}$.

For node c_2 of σ_c ,

min of $\{7,9,6\}$ is 6 and occurs for node d_3 in σ_d .

In G°_1 add arcs $a'(c_2, a_1)$ and $a'(c_2, a_2)$, each with cost 6 and label $\{d_3\}$.

Step 3: Delete σ_c

Step 4: Return

Set this graph to G°_2

Cut reduce node d in G_1 and set this graph to G_2 (Figure 3).

Set k to 2

{In Figure 3b, the first matrix corresponds to the weights on the new arcs that are added in Iteration 1 }

Add (c,a) to PA and go to Step 5.

Iteration 2:

Step 5: We choose $(c,a) \in PA$ and call procedure $PR(\sigma_c, \sigma_a)$.

Procedure $PR(\sigma_c, \sigma_a)$

Step 1: Pick nodes $a_1 \in \sigma_a, c_1 \in \sigma_c$

Delete one of the two parallel arcs. Add the weight and label of the deleted arc to the weight and label of the other arc (new weight = $5+4 = 9$; new label = $\{\} \cup \{d_2\} = \{d_2\}$).

Step 2: We continue Step 1 until no parallel arcs remain between σ_c and σ_a .

Step 3: Return.

Parallel reduce arcs between nodes a and c in G_2 .

Update G_3, G°_3 . Set k to 3. Since c has degree 2 add c to 2D and go to Step 1 (see

Figure 4).

Iteration 3:

Step 1: Since we do not have a pendant node, we go to Step 3.

Step 3: We choose node $c \in 2D$ with b and a the adjacent nodes to c in G_3 .

Step 4: Since (b,a) is in G_3 , we add (b,a) to PA .

Call procedure $SR(\sigma_c)$.

Procedure $SR(\sigma_c)$

Step 1: b and a are two nodes adjacent to node c in G_2

Step 2: Consider nodes $b_1 \in \sigma_b$ and $a_1 \in \sigma_a$

Find min of $\{\omega_{11}^{bc} + \omega_{11}^{ac}, \omega_{12}^{bc} + \omega_{12}^{ac}\}$
or min of $\{9+9, 19+13\} = 18$

Minimum occurs for node $c_1 \in \sigma_c$

Add arc $a'(b_1, a_1)$ with weight 18

Set the label $L_{a'}(b_1, a_1) \leftarrow L_a(b_1, c_1) \cup L_a(a_1, c_1) \cup L_n(c_1)$

or $L_{a'}(b_1, a_1) = \{\} \cup \{d_2\} \cup \{c_1\} = \{c_1, d_2\}$

Consider nodes $b_1 \in \sigma_b$ and $a_2 \in \sigma_a$

Find min of $\{\omega_{11}^{bc} + \omega_{21}^{ac}, \omega_{12}^{bc} + \omega_{22}^{ac}\}$
or min of $\{9+11, 19+16\} = 20$ for node c_1

Add arc $a'(b_1, a_2)$ with weight 20

Set the label $L_{a'}(b_1, a_2) \leftarrow L_a(b_1, c_1) \cup L_a(a_2, c_1) \cup L_n(c_1)$

or $L_{a'}(b_1, a_2) = \{\} \cup \{d_2\} \cup \{c_1\} = \{c_1, d_2\}$

Consider nodes $b_2 \in \sigma_b$ and $a_1 \in \sigma_a$

Find min of $\{\omega_{21}^{bc} + \omega_{11}^{ac}, \omega_{22}^{bc} + \omega_{12}^{ac}\}$
or min of $\{11+9, 6+13\} = 19$ for node c_2

Add arc $a'(b_2, a_1)$ with weight 19

Set the label $L_{a'}(b_2, a_1) \leftarrow L_a(b_2, c_2) \cup L_a(a_1, c_2) \cup L_n(c_2)$

or $L_{a'}(b_2, a_1) = \{\} \cup \{d_3\} \cup \{c_2\} = \{c_2, d_3\}$

Consider nodes $b_2 \in \sigma_b$ and $a_2 \in \sigma_a$

Find min of $\{\omega_{21}^{bc} + \omega_{21}^{ac}, \omega_{22}^{bc} + \omega_{22}^{ac}\}$
or min of $\{11+11, 6+16\} = 22$ for node c_2

Add arc $a'(b_2, a_2)$ with weight 22

Set the label $L_{a'}(b_2, a_2) \leftarrow L_a(b_2, c_2) \cup L_a(a_2, c_2) \cup L_n(c_2)$
 or $L_{a'}(b_2, a_2) = \{ \} \cup \{d_3\} \cup \{c_2\} = \{c_2, d_3\}$

Consider nodes $b_3 \in \sigma_b$ and $a_1 \in \sigma_a$

Find min of $\{\omega_{31}^{bc} + \omega_{11}^{ac}, \omega_{32}^{bc} + \omega_{12}^{ac}\}$
 or min of $\{14+9, 12+13\} = 23$ for node c_1

Add arc $a'(b_3, a_1)$ with weight 23

Set the label $L_{a'}(b_3, a_1) \leftarrow L_a(b_3, c_1) \cup L_a(a_1, c_1) \cup L_n(c_1)$
 or $L_{a'}(b_3, a_1) = \{ \} \cup \{d_2\} \cup \{c_1\} = \{c_1, d_2\}$

Consider nodes $b_3 \in \sigma_b$ and $a_2 \in \sigma_a$

Find min of $\{\omega_{31}^{bc} + \omega_{12}^{ac}, \omega_{32}^{bc} + \omega_{22}^{ac}\}$
 or min of $\{14+11, 12+16\} = 25$ for node c_1

Add arc (b_3, a_2) with weight 25

Set the label $L_{a'}(b_3, a_2) \leftarrow L_a(b_3, c_1) \cup L_a(a_2, c_1) \cup L_n(c_1)$
 or $L_{a'}(b_3, a_2) = \{ \} \cup \{d_2\} \cup \{c_1\} = \{c_1, d_2\}$

Step 3: Delete σ_c . Return. {See Figure 5 for the weights and labels on parallel arcs between σ_a and σ_b .

Series reduce c in G_3 .

Update G_3 and G^o_3 , set k to 4 and go to Step 5.

Iteration 4:

Step 5: Edge $(b, a) \in PA$. Call $PR(\sigma_b, \sigma_a)$.

Procedure $PR(\sigma_b, \sigma_a)$

Step 1: Choose nodes $a_1 \in \sigma_a$ and $b_1 \in \sigma_b$,

Delete one of the two parallel arcs.

Add the weight of the deleted arc to the other arc giving the weight as $18+7=25$.

Add the labels on the two parallel arcs

$= \{c_1, d_2\} \cup \{ \} = \{c_1, d_2\}$

Step 2: We continue Step 1 until no parallel arcs remain (Figure 6 give the weights and labels, before and after this reduction).

Step 3: Return.

Parallel reduce arcs between nodes b and c in G_4 .

Update G_4, G°_4 .

Set k to 5 and go to Step 1.

Iteration 5:

Step 1: G_4 is a single arc, so we go to Step 6.

Step 6: We find $\omega_{k^{\circ}r^{\circ}}^{ba} = \min_{k \in \sigma_b, r \in \sigma_a} \{\omega_{kr}^{ba}\}$.

$= \min \{25, 24, 23, 27, 31, 29\} = 23$ and occurs for $a_{r^{\circ}} = a_1$ and $b_{k^{\circ}} = b_2$

This is the value of the optimal solution.

The solution can be constructed by $L_a(b_2, a_1) \cup L_n(b_2) \cup L_n(a_1) = \{c_2, d_3, b_2, a_1\}$

Thus choose node $c_2 \in \sigma_c, d_3 \in \sigma_d, b_2 \in \sigma_b$, and $a_1 \in \sigma_a$. Stop. {The dark edges along with the nodes incident by the dark edges in Figure 7 depicts the solution.}

6. M-Center Problem With Mutual Communication

In the m -center problem with mutual communication (as defined by Kolen (1982)), we want to find the location of the m new facilities to minimize the maximum of the interaction costs of new and existing facilities, interaction costs of pairs of new facilities, and the fixed cost of locating a new facility (this term is not included in (Kolen, 1982)). In what follows, we consider the node-restricted version of the problem. There is no loss of generality here since we can find (see Hooker, Garfinkel, and Chen, 1988) a finite number of points on the transport network such that in an optimal solution to the problem each new facility will be at one of the points. Thus we can declare these points to be nodes of τ . In our version of the problem, denoted by MCMC, we assume each existing facility is located at some node of τ . Thus we again use the convention that nodes of τ (as well as locations of existing facilities) are indexed by k, r , and i . Similar to (MMMC), if there is a node of τ with no existing facility, we simply declare it to be an existing facility which has zero interaction with all existing facilities. In addition, we will consider the fixed location costs of the new facilities.

We now describe a G -partite graph G° and a *bottleneck* version of NSP, denoted by B-NSP, on G° , which if solved, will solve (MCMC). For each new facility, u , we again

define a node-family σ_u , consisting of nodes $\{u_k, k \in N_u\}$ where N_u is the set of feasible locations (nodes of τ) for new facility u . If new facility u interacts with new facility v , i.e., $b_{uv} > 0$, there is an arc in G° between every member of σ_u and σ_v . Consider $u_k \in \sigma_u$ and $v_r \in \sigma_v$. A choice of these nodes corresponds to new facility u being located at node k of τ and new facility v being located at node r of τ . The cost of arc (u_k, v_r) is defined to be maximum of the following terms: a) f_{uk} , b) f_{vr} , c) $b_{uv}d(k,r)$, d) $\max_i \{a_{vi}d(r,i)\}$, and e) $\max_i \{a_{ui}d(k,i)\}$. Terms a) and b) reflect the fixed cost of locating new facilities u and v at nodes k and r , respectively. Term c) is the cost of interaction between new facilities u and v , given their locations. Term d) (e)) represents the maximum cost of interaction between new facility v (u) and any existing facility.

With this definition of the cost of each arc in G° , the B-NSP on G° to solve (MCMC) can be stated as follows: find an induced subgraph, $S(G^\circ)$, of m nodes of G° , such that $S(G^\circ)$ contains one node of each node-family and where the maximum of the arc costs of the arcs of $S(G^\circ)$ is minimum.

We now point out the changes required in the procedures described in Section 4 to solve B-NSP. Algorithm SP remains identical except that instead of calling procedures CR, PR, and SR, it will call procedures CR (as before) as well as SR' and PR', described below. To distinguish these procedures from the previous procedures we name them b-series-reduction and b-parallel-reduction, respectively.

PROCEDURE SR'(σ_q)

Step 2 For each pair of nodes $u_k \in \sigma_u, v_r \in \sigma_v$, find q_{p° giving

$$\max\{\omega_{kp^\circ}^{uq}, \omega_{rp^\circ}^{vq}\} = \min_{q_p \in \sigma_q} \{\max\{\omega_{kp}^{uq}, \omega_{rp}^{vq}\}\}.$$

Add an arc $a'(u_k, v_r)$ with weight equal to $\max\{\omega_{kp^\circ}^{uq}, \omega_{rp^\circ}^{vq}\}$ and

let the label of this new arc be $L_{a'}(v_r, u_k) \leftarrow L_a(v_r, q_{p^\circ}) \cup L_a(u_k, q_{p^\circ}) \cup L_n(q_{p^\circ})$.

Lemma 3': For a G-Partite graph G° with a node-family σ_q such that q has degree two in G , let \underline{G}° and \underline{G} be the results of b-series-reducing node-family σ_q and node q in G° and G , respectively. An optimal solution to (B-NSP) on G° can be constructed using the nodes and arc labels of an optimal solution to (B-NSP) on \underline{G}° . Furthermore, the complexity of

procedure $SR'(\cdot)$ is $O(|N_u|*|N_v|*|N_q|)$ where u and v are the nodes of G adjacent to node q .

Proof: In the objective function of NSP the sum of arc weights is computed whereas in B-NSP, the maximum of arc weights is taken. Thus, while performing the b-series-reduction, instead of looking at the sum of weights on arcs (u_k, q_p) and (v_r, q_p) we focus on the maximum of the weights on these two arcs. The proof follows exactly the same arguments as in the proof of Lemma 3, except that the '+' operator is replaced by the 'max' operator.

«»

PROCEDURE $PR'(\sigma_u, \sigma_v)$

Step 1: For any two nodes $u_k \in \sigma_u$ and $v_r \in \sigma_v$ such that there are parallel arcs between them, replace the parallel arcs by a new arc with weight equal to the maximum of arc weights on the arcs just deleted and label equal to the union of the labels on the deleted arcs.

Lemma 5': For a G -Partite graph G° with node-families σ_u and σ_v such that there are two arcs between every node $u_k \in \sigma_u$ and $v_r \in \sigma_v$, let \underline{G}° and \underline{G} be the results of b-parallel-reducing node-families σ_u and σ_v of G° and nodes u and v in G , then an optimal solution to (B-NSP) on G° can be constructed using the nodes and arc labels of an optimal solution to (B-NSP) on \underline{G}° . Furthermore, $PR'(\cdot)$ can be performed in $O(|N_u|*|N_v|)$ time.

Proof: The proof of this Lemma is similar to the proof of Lemma 5, but again the 'max' operator is replaces the '+' operator. «»

Note that no modifications are required in PROCEDURE $CR(\cdot)$. Following the notation as in CR , for any node $u_k \in \sigma_u$, the node in σ_q for B-NSP will also be selected by finding q_p° which minimizes ω_{kp}^{uq} . If we add additional arcs A' , and delete σ_q , the solutions on the two graphs for B-NSP remain equivalent (see proof of Lemma 4).

7. Conclusions

In this paper we have provided polynomial time algorithms for special cases of the m -median and m -center problems with mutual communication. The special case is characterized by the structure of the flow graph. First, we reformulated the m -median problem with mutual communication as a quadratic location problem which was then formulated as a node selection problem posed on a G -partite graph. Then we presented an algorithm (Algorithm SP) which solves NSP when the flow graph is series-parallel. The m -center problem with mutual communication was formulated as the bottleneck version of the node selection problem (B-NSP). We presented the modifications required in Algorithm SP to solve B-NSP.

Algorithm SP is a general algorithm to solve any problem which can be modeled as an NSP. As an example, consider the 0-1 quadratic programming problem (Barahona, 1986):

$$(QP): \min \frac{1}{2}x^t Q x + c x = \sum_{i=1..n} \sum_{j=i+1..n} q_{ij} x_i x_j + \sum_{i=1..n} c_i x_i, x \in \{1,0\}.$$

To model QP as an NSP we create a G -partite graph, G° , with a node-family for each x_i which has two nodes, n_{i0} and n_{i1} . Node n_{i0} (n_{i1}) corresponds to the variable x_i taking the value 0 (1). Join two node-families if and only if $q_{ij} > 0$. The weight on arc (n_{i1}, n_{j1}) is initially set equal to q_{ij} and all other arcs between σ_i and σ_j are initially given weight 0. To account for the linear costs c_i , we select a j such that $q_{ij} > 0$, and add c_i to the weight on arcs (n_{i1}, n_{j0}) and (n_{i1}, n_{j1}) . It is easy to see that $NSP(G^\circ)$ is a reformulation of QP. Thus when the graph defined by q_{ij} 's is series-parallel, QP can be solved in $O(n^2)$ time. As we noted earlier, Barahona (1986) also has a linear time algorithm for this case of QP but he solves it by converting it to a weighted max-cut problem on a graph H which has one additional node adjacent to all nodes of the flow graph defined by the q_{ij} 's.

We now discuss certain generalization of series-parallel flow graphs for which NSP can be solved in polynomial time. To give an example of this generalization, consider a series-parallel graph G' and a graph B_1 which has two special nodes, u' and v' , denoted as *terminals* (Figure 8). If we *replace* arc (u,v) in G by the graph B_1 we get a graph G which is no longer series-parallel (Figure 9). Here we define replacement of an arc (u,v) of a

graph G' by a graph B_1 (with two terminals u' and v') as: delete arc (u,v) in G' and attach graph B_1 to G' such that node u' coincides with node u and node v' coincides with node v .

In order to solve an NSP on a G -partite graph, G° , corresponding to the flow graph G defined above, we proceed as follows: for a choice of node pairs $u_{k^\circ} \in \sigma_u$ and $v_{r^\circ} \in \sigma_v$, find the optimal nodes in node-families σ_{n1} and σ_{n2} and let these two nodes be denoted by the set $R(k^\circ, r^\circ)$. Note that once we choose nodes $u_{k^\circ} \in \sigma_u$ and $v_{r^\circ} \in \sigma_v$, the optimal choice of nodes in σ_{n1} and σ_{n2} is independent of the solution on the remainder of G . Also, note that due to the structure of B_1 , finding the optimal nodes on the remainder of B_1 , i.e., finding $R(k^\circ, r^\circ)$, with u_{k° and v_{r° fixed can be done in polynomial time. Let $w(k^\circ, r^\circ)$ denote the sum of the weights on the arcs in the graph induced by nodes $\{u_{k^\circ}, v_{r^\circ}\} \cup R(k^\circ, r^\circ)$. Insert arc $(u_{k^\circ}, v_{r^\circ})$ with weight $w(k^\circ, r^\circ)$ in G° . Repeat this process for every choice of node pairs $u_k \in \sigma_u$ and $v_r \in \sigma_v$, then delete node families σ_{n1} and σ_{n2} from the G -partite graph G° . The new G -partite graph corresponds to the series-parallel flow graph G' . It is easy to see that given an optimal solution to NSP on the new G -partite graph we can construct an optimal solution to NSP on the original G -partite graph. Thus, we can solve NSP with flow graph G in Figure 9, which is not series-parallel, in polynomial time

To generalize the above, we define a family of graphs, π . Graph B_i is a member of π if there are two terminal u and v of B_i , such that for arbitrary fixed nodes, $u_{k^\circ} \in \sigma_u$ and $v_{r^\circ} \in \sigma_v$, $S^*(G^\circ(B_i), \psi)$ can be computed in polynomial time, where $G^\circ(B_i)$ is the G -partite graph corresponding to B_i and with $\psi = \{u_{k^\circ}, v_{r^\circ}\}$. If we obtain a flow graph G by replacing some arcs of a series-parallel graph by a member of π , we can still solve NSP with flow graph G in polynomial time. Campbell and Rardin (1987) have defined a similar class of generalized series-parallel graphs, which they called series-parallel block graphs, and they give an algorithm to recognize the series-parallel block graphs in polynomial time. The algorithm in (Campbell and Rardin, 1987) when applied to a graph G , will give a set of graphs with two terminals. If all the graphs in this set are member of the family π , then NSP on G can be solved in polynomial time.

References

- Barahona, F., "A Solvable Case of Quadratic 0-1 Programming," Discrete Applied Mathematics, Vol. 13, 1986, pp. 23-26.
- Campbell, B. A. and R. L. Rardin, "Steiner Tree Problem on Series-Parallel Block Graphs I: Polynomial Recognition and Solution," Tech. Report CC-87-17, School of Industrial Engineering, Purdue University.
- Dearing, P. M, R. L. Francis, and T. J. Lowe, "Convex Location Problems on Tree Networks," Operations Research, Vol. 24, 1976, pp. 628-642.
- Erkut, E., R. L. Francis, and T. J. Lowe, "A Multimedial Problem with Interdistance Constraints," Environment and Planning B: Planning and Design, Vol. 15, 1988, pp. 181-190.
- Erkut, E., R. L. Francis, T. J. Lowe, and A. Tamir, "Equivalent Mathematical Programming Formulations of Monotonic Tree Network Location Problems," Operations Research, Vol. 37(3), 1989, pp. 447-461.
- Francis, R. L., T. J. Lowe, and H. D. Ratliff, "Distance Constraints for Tree Network Multifacility Location Problems," Operations Research, Vol. 26, 1978, pp. 570-596.
- Hooker, J. N., R. S. Garfinkel, and C. K. Chen, "Finite Dominating Sets for Network Location Problem," Working Paper 19-87-88, Carnegie Mellon University, March 1988
- Kolen, A., "Location Problems on Trees and in the Rectilinear Plane," Stichting Mathematisch centrum, Kruislaan 413, 1098 SJ, Amsterdam, The Netherlands, 1982.
- Picard, J-C and H. D. Ratliff, "A Cut Approach to the Rectilinear Distance Facility Location Problem," Operations Research, Vol. 28, 1978, pp. 422-433.
- Rardin, R. L., R. G. Parker, and M. B. Richey, "A Polynomial Algorithm for a Class of Steiner Tree Problems on Graphs," ISyE Report Series J-82-5, Georgia Tech., August, 1982.
- Rendl, F., "Quadratic Assignment Problems on Series-Parallel Digraphs," Zeitschrift für Operations Research, Vol. 30, 1986, pp. A161-A173.
- Richey, M. B., "Optimal Location of a Path or Tree on a Network with Cycles," Working Paper, George Mason University, 1989.
- Takamizawa, K., T. Nishizeki, and S. Saito, "Linear-Time Computability of Combinatorial Problems on Series-Parallel Graphs," J. of the ACM, Vol. 29, 1982, pp. 623-641.
- Wald, J. A. and C. J. Colbourn, "Steiner Trees, Partial 2-Trees, and Minimum IFI Networks," Networks, Vol. 13, 1983, pp. 159-167.
- Xu, Y., R. L. Francis, and T. J. Lowe, "The Multimedial Problem on a Network: Exploiting Block Structure," Working Paper, Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida, 1988.

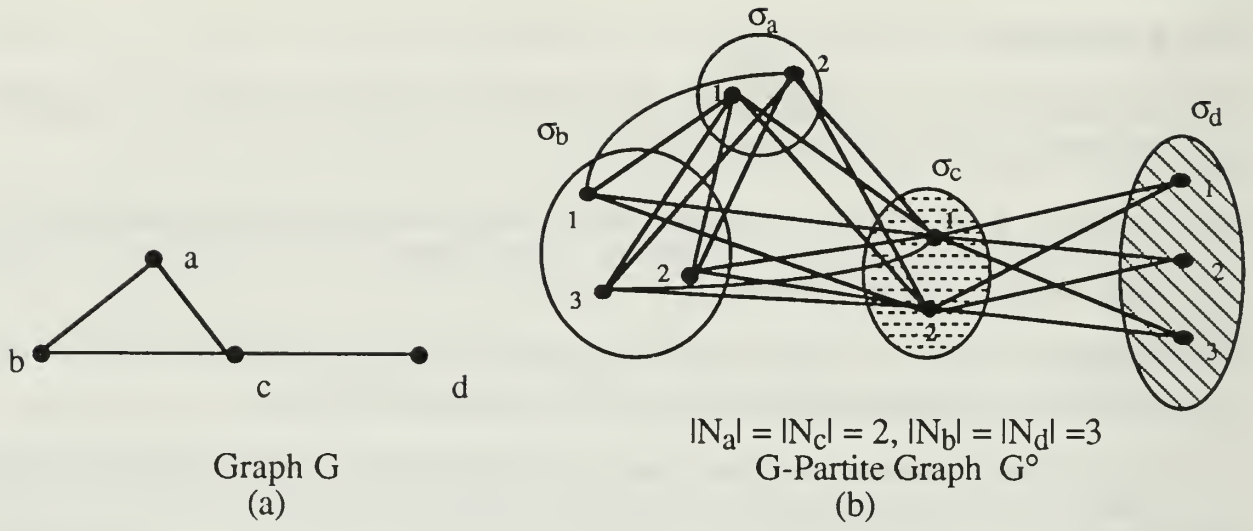


Figure 1. Graphs G and G°

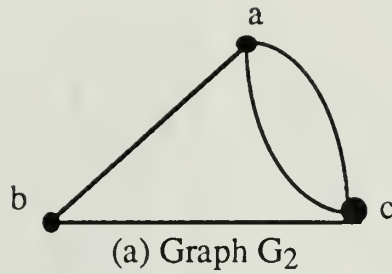
Weights			
c \ d	1	2	3
1	5	4	8
2	7	9	6

Weights		
c \ a	1	2
1	5	7
2	7	10

Weights			
c \ b	1	2	3
1	9	11	14
2	19	6	12

Weights			
a \ b	1	2	3
1	7	4	8
2	4	5	4

Figure 2. Weights on G°



New arcs

Weights		
a \ c	1	2
1	4	4
2	6	6

Weights		
c \ a	1	2
1	5	7
2	7	10

Weights			
c \ b	1	2	3
1	9	11	14
2	19	6	12

Weights			
a \ b	1	2	3
1	7	4	8
2	4	5	4

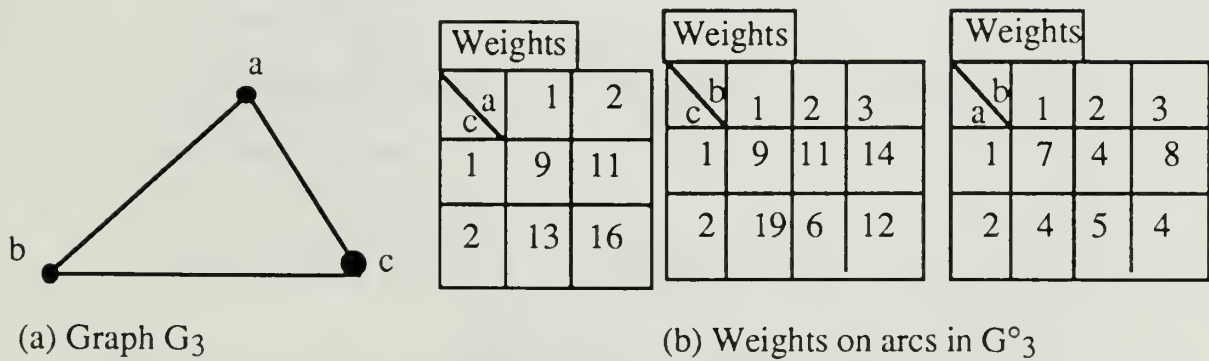
(b) Weights on arcs in G°_2

$$L_a(c_1, a_1) = \{d_2\} \quad L_a(c_1, a_2) = \{d_2\}$$

$$L_a(c_2, a_1) = \{d_3\} \quad L_a(c_2, a_2) = \{d_3\}$$

(c) New Labels

Figure 3. End of iteration 1

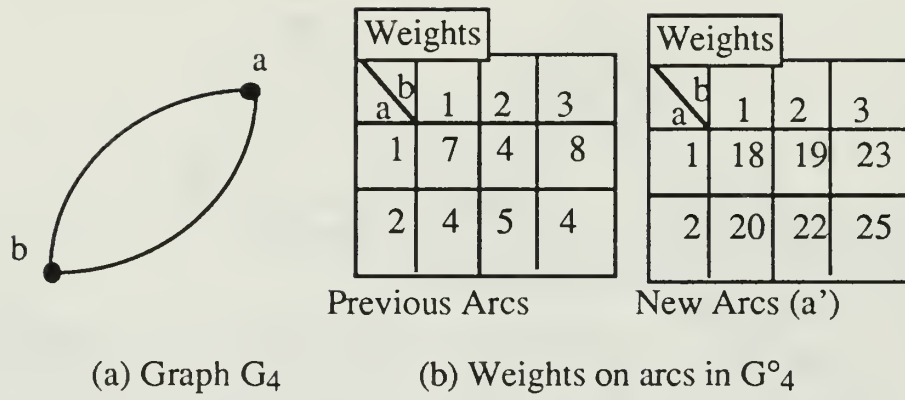


$$L_a(c_1, a_1) = \{d_2\} \quad L_a(c_1, a_2) = \{d_2\}$$

$$L_a(c_2, a_1) = \{d_3\} \quad L_a(c_2, a_2) = \{d_3\}$$

(c) New Labels

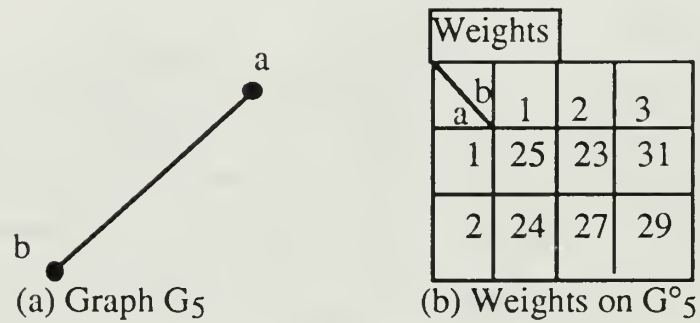
Figure 4. End of Iteration 2



Labels: New Arcs (a')

$$\begin{array}{lll}
 L_{a'}(a_1, b_1) = \{c_1, d_2\} & L_{a'}(a_1, b_2) = \{c_2, d_3\} & L_{a'}(a_1, b_3) = \{c_1, d_2\} \\
 L_{a'}(a_2, b_1) = \{c_1, d_2\} & L_{a'}(a_2, b_2) = \{c_2, d_3\} & L_{a'}(a_2, b_3) = \{c_1, d_2\}
 \end{array}$$

Figure 5. End of Iteration 3



Labels same as the labels on the New Arcs a' in Iteration 3

Figure 6. End of Iteration 4

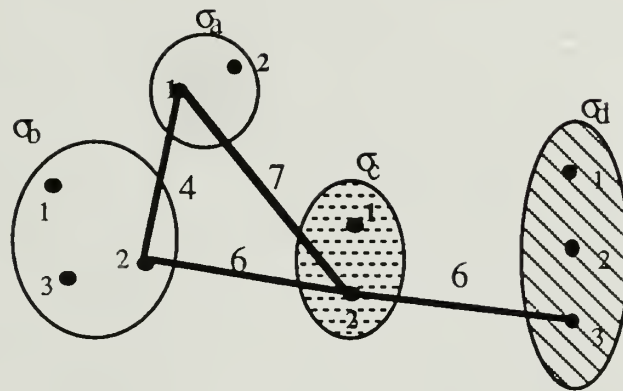


Figure 7. Algorithm SP: The Solution

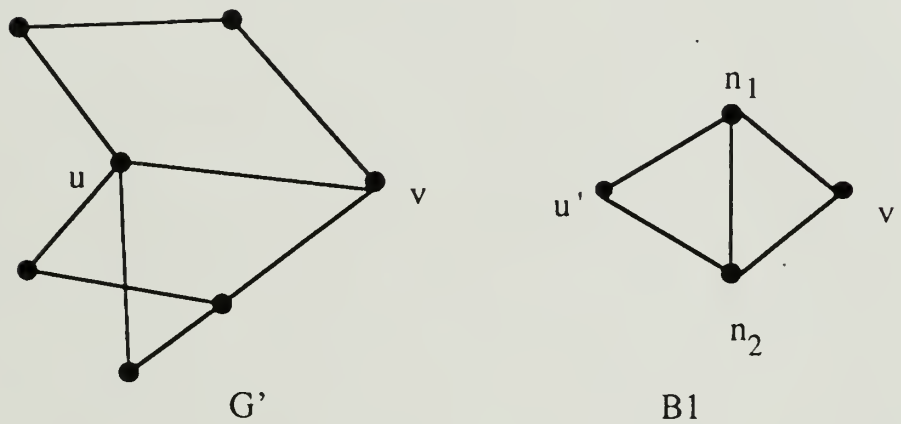


Figure 8. Series-Parallel Graph G' with a Block $B1$

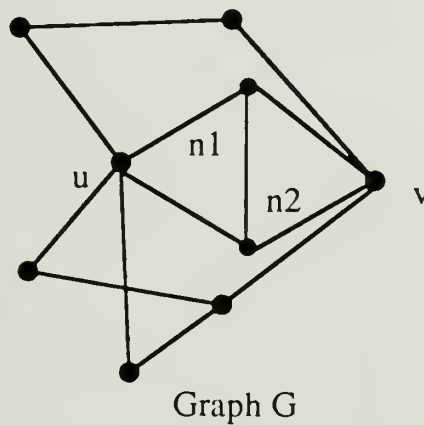


Figure 9. Graph G Obtained by Replacing Edge (u,v) in G' by $B1$

UNIVERSITY OF ILLINOIS-URBANA



3 0112 060295950