A GRAPH-THEORETIC APPROACH TO A CLASS OF INTEGER-PROGRAMMING PROBLEMS

J. F. Desler

Shell Development Company, Houston, Texas

and

S. L. Hakimi

Northwestern University, Evanston, Illinois

(Received July 31, 1968)

This paper presents an efficient algorithm for finding a minimum-weight generalized matching in a weighted bipartite graph. Computational evidence is given that indicates that the time required to find a least-cost assignment of n jobs to n workers goes roughly as n^2 for $10 \leqq n \leqq 50$. It is shown that this algorithm can be used to solve effectively the well known transportation problem of integer programming where the objective function is convex-separable. Finally, the paper gives an algorithm that applies the same concept to a graph that is not necessarily bipartite.

A *GRAPH* G is a collection of two types of entities, a set of m *branches* $B=\{b_1, b_2, \cdots, b_m\}$ and a set of n *vertices* $V=\{v_1, v_2, \cdots, v_n\}$. Associated with each branch $b_k \epsilon B$ are two distinct vertices $v_i \epsilon V$ and $v_j \epsilon V$ called the *ends* of b_k. Such a branch b_k may be represented by the unordered pair (v_i, v_j), and such a pair of vertices is said to be *adjacent*. A branch is said to be *incident* at its ends. A *subgraph* $g \subseteq G$ is defined by a subset of the branches of G, and g has the same vertices as does G. Each branch of g has the same two ends as it has in G. By this definition of a subgraph, the set-theoretic operations on subgraphs are meaningful. In this connection, we define

$$g_1 \oplus g_2 = (g_1 \cup g_2) - (g_1 \cap g_2), \tag{1}$$

where g_1 and g_2 are subgraphs of the graph G. By the *degree* $d(v_i, g)$ of a vertex v_i of a subgraph $g \subseteq G$, we mean the number of branches of g that are incident at v_i. Let g_1 and g_2 be any two subgraphs of a graph G, and let v_i be any arbitrary vertex of G. The reader can convince himself of the following identity:

$$d(v_i, g_1 \oplus g_2) = d(v_i, g_1) + d(v_i, \bar{g}_1 \cap g_2) - d(v_i, g_1 \cap g_2). \tag{2}$$

A graph G with vertices V and branches B will be denoted by the ordered pair $[V, B]$. Given a graph $G=[V, B]$, it is understood that a branch-vertex incidence relation exists. A *weighted graph* is a graph $G=[V, B]$ together with a *weight function* $w: B \to R$ that assigns a real number $w(b_i) \epsilon R$ to each branch $b_i \epsilon B$. If g is a subgraph of the weighted graph G, then the weight of g, $w(g)$, is the sum of the weights of the branches of g. A *bipartite*[1] (or simple[2]) graph is a graph $G=[V, B]$ in which there exist two mutually disjoint subsets of vertices $X \subset V$ and $Y \subset V$ such that $X \cup Y = V$ and such that each branch $b_k \epsilon B$ is incident at a vertex in X and at a vertex in Y. Such a bipartite graph G will be denoted $[(X, Y), B]$.

Let $G=[(X, Y), B]$ be a weighted bipartite graph where $|X|=p$ and $|Y|=q$. Let us assume that we are given two sets of nonnegative integers $\alpha=\{\alpha_1, \alpha_2, \cdots, \alpha_p\}$ and $\beta=\{\beta_1, \beta_2, \cdots, \beta_q\}$ called the *degree constraints* on X and Y respectively. We say that the constraints are *feasible* if there exists a subgraph $g \subseteq G$ such that $d(x_i, g)=\alpha_i$ for all $x_i \epsilon X$ and $d(y_j, g)=\beta_j$ for all $y_j \epsilon Y$. Such a subgraph g is called a *feasible subgraph* of G. If G contains no feasible subgraphs, then the constraints are said to be *infeasible*. We would like to determine whether or not the given degree constraints α and β on X and Y are feasible. If they are, then from among all feasible subgraphs of G we would like to find one of minimum weight.

An important application of the above problem is the least-cost assignment problem. In this problem $p=q$, and we associate the vertices in $X=\{x_1, x_2, \cdots, x_p\}$ with p workers and the vertices in $Y=\{y_1, y_p, \cdots, y_p\}$ with p jobs. A branch $b_k=(x_i, y_j)$ in G signifies the willingness and ability of the worker corresponding to vertex $x_i \epsilon X$ to do the job corresponding to vertex $y_j \epsilon Y$. The weight $w(b_k)$ of branch b_k is the total cost of having worker x_i do job y_j. If we set $\alpha_i=\beta_i=1$ for $i=1, 2, \cdots, p$, then a minimum-weight feasible subgraph $g^* \subseteq G$ corresponds to a least-cost assignment of jobs to workers. In graph-theoretic terms the subgraph g described above (that is, a subgraph in which every vertex has degree one) is called a *matching*.

Graph-theoretic problems of finding optimal matchings and optimal degree-constrained subgraphs have been considered by, among others, Norman and Rabin,[3] Berge,[2] and, more recently, Edmonds[4, 5] and Goldman.[6] Edmonds[5] gave an algorithm for finding a minimum-weight matching in a weighted graph, and some of his work was modified by Witzgall and Zahn.[7] It should be pointed out that the work of Edmonds dealt with the problem of finding optimal matchings in weighted graphs, while our main results apply to the problem of finding optimal degree-constrained subgraphs in a weighted bipartite graph. Each of these problems is a special case of the problem of finding optimal degree-constrained subgraphs in a weighted graph. In Section VI we show how the ideas introduced in this paper apply to this more general problem.

For the case where the graph is bipartite, it can be shown that the problem of finding a minimum-weight feasible subgraph is equivalent to the HITCHCOCK[8] problem of integer programming. Such problems can be solved by the Hungarian method of KUHN.[9, 10] FORD AND FULKERSON[11] and MUNKRES[12] have given algorithms that are based on Kuhn's idea and that can be used to solve the Hitchcock problem. In Section III we show that our algorithm has an upper bound of n^4 on the number of operations necessary to find a least-cost assignment of n jobs to n workers. This same upper bound is reported by Munkres[12] for his algorithm, but his 'scan a row' or 'scan a column' are not single operations, but, rather, require n comparisons each. In Section IV, we present a basic idea leading to a significant improvement in the efficiency of the algorithm. We offer computational evidence using this revised algorithm that indicates the time required to find a least-cost assignment of n jobs to n workers goes roughly as n^2 for $10 \leqq n \leqq 50$. In Section V we show that our method can be used to solve the well known transportation problem of integer programming, where the objective function is a sum of convex functions, each in one variable. Finally, it is shown that the theoretical basis of our method extends to the case of nonbipartite graphs.

I. ALGORITHM

AN *elementary path* between vertices v_i and v_j in a graph G is a subgraph consisting of a sequence of branches represented by $(v_i, v_{k_1})(v_{k_1}, v_{k_2})\cdots(v_{k_{t-1}}, v_j)$, where vertices $v_i, v_{k_1}, v_{k_2}, \cdots, v_{k_{t-1}}, v_j$ are distinct, with the possible exception of v_i and v_j. If $v_i = v_j$, then the elementary path is called a *circuit.* A *directed graph* is a graph, each of whose branches has assigned to it a definite orientation. When a directed graph is given by means of a diagram, it is common practice to place arrowheads on the branches in order to signify the orientations. If b_k is a branch of a directed graph $G=[V, B]$ and the arrowhead on b_k points from $v_i \epsilon V$ to $v_j \epsilon V$ (where v_i and v_j are the ends of b_k), then b_k may be represented by the ordered pair (v_i, v_j). An *elementary directed path* from vertex v_i to vertex v_j in a directed graph is a subgraph consisting of a sequence of branches represented by $(v_i, v_{k_1})(v_{k_1}, v_{k_2})\cdots(v_{k_{t-1}}, v_j)$, where vertices $v_i, v_{k_1}, v_{k_2}, \cdots, v_{k_{t-1}}, v_j$ are distinct, with the possible exception of v_i and v_j. If $v_i = v_j$, then the elementary directed path is called a *directed circuit.*

Let g be a subgraph of a weighted graph $G=[V, B]$. We define a weighted graph $G|g$ by assigning new weights to the branches of G. Since G and $G|g$ have the same structure (that is, they have the same branches, vertices, and branch-vertex incidence relations), there should be no confusion if we say that a subgraph of G is a subgraph of $G|g$ and vice versa. Let $w(\cdot)$ and $w(\cdot|g)$ be the weight functions (their domains being the set of all subgraphs) for G and $G|g$, respectively. If h is any subgraph, then we define

$$w(h|g) = w(h \cap \bar{g}) - w(h \cap g). \tag{3}$$

Since a single branch is a subgraph, the weighted $G|g$ is well defined. We have $w(b_i|g) = w(b_i)$ if $b_i \epsilon \bar{g}$ and $w(b_i|g) = -w(b_i)$ if $b_i \epsilon g$. Let h_1 and h_2 be any two subgraphs of G. The reader can convince himself of the following elementary property:

$$w(h_1 \oplus h_2) = w(h_1) + w(h_2|h_1). \tag{4}$$

If $G = [(X, Y), B]$ is a weighted bipartite graph and $g \subseteq G$, then we define a weighted directed graph $\mathbf{G}|g$ by placing orientations on the branches of $G|g$ as follows: (1) if b_k is a branch of g, then b_k in $\mathbf{G}|g$ is directed from X to Y, and (2) otherwise b_k is directed from Y to X.

Let $G = [(X, Y), B]$ be a given weighted bipartite graph, where $X = \{x_1, x_2, \cdots, x_p\}$ and $Y = \{y_1, y_2, \cdots, y_q\}$. Let $\alpha = \{\alpha_1, \alpha_2, \cdots, \alpha_p\}$ and $\beta = \{\beta_1, \beta_2, \cdots, \beta_q\}$ be given degree constraints on X and Y respectively. Let g be a subgraph of G. We define two sets of vertices with respect to g:

$$U(g) = \{x_i \epsilon X | d(x_i, g) < \alpha_i\} \cup \{y_j \epsilon Y | d(y_j, g) < \beta_j\},$$
$$O(g) = \{x_i \epsilon X | d(x_i, g) > \alpha_i\} \cup \{y_j \epsilon Y | d(y_j, g) > \beta_j\}.$$

Clearly, g is a feasible subgraph if and only if $U(g) = \Phi$ and $O(g) = \Phi$. The sets $U(g)$ and $O(g)$ are, respectively, called the set of *under-exposed vertices* and the set of *over-exposed vertices* with respect to g. We would like either to find a minimum-weight feasible subgraph $g^* \subseteq G$ or to determine that the constraints are infeasible. The following conditions are necessary, but not sufficient, for the feasibility of the constraints:

$$\begin{aligned}
&(1) \quad d(x_i, G) \geqq \alpha_i \quad \text{for all } x_i \epsilon X,\\
&(2) \quad d(y_j, G) \geqq \beta_j \quad \text{for all } y_j \epsilon Y,\\
&(3) \quad \textstyle\sum_{i=1}^{i=p} \alpha_i = \sum_{j=1}^{j=q} \beta_j.
\end{aligned}$$

Throughout the remainder of this paper we will assume that these conditions are satisfied. The algorithm given below will produce the desired subgraph g^*, if the constraints are feasible, and will determine that the constraints are infeasible otherwise.

Algorithm

Step 1. Let g_0 be a subgraph of G consisting of β_j least-weight branches incident at y_j for $j = 1, 2, 3, \cdots, q$. (Therefore g_0 is a minimum-weight subgraph of G that satisfies $d(y_j, g_0) = \beta_j$ for all $y_j \epsilon Y$.)

Step 2. Let g_i be the subgraph of G obtained after $i \geqq 0$ cycles of the algorithm. (We start with $i = 0$.) If $O(g_i) = \Phi$, then $g_i = g^*$, and the algorithm is terminated. Otherwise, choose $x_{s_i} \epsilon O(g_i)$, and go to Step 3.

Step 3. Find the least-weight elementary directed path in $\mathbf{G}|g_i$ that begins at x_{s_i}

and ends at a vertex of $U(g_i)$. If no such path exists, terminate the algorithm, for the constraints are infeasible. Otherwise, let p_i denote the path found, and go to Step 4.

Step 4. $g_{i+1} = g_i \oplus p_i$. Return to Step 2.

In the next section, we prove that the steps in the algorithm are meaningful and that they do not constitute any computational difficulty. When the algorithm is allowed to run until it is terminated, either the desired subgraph is obtained or it is determined that no feasible subgraph exists. Before that, however, we give a simple example to demonstrate the steps involved in the algorithm.

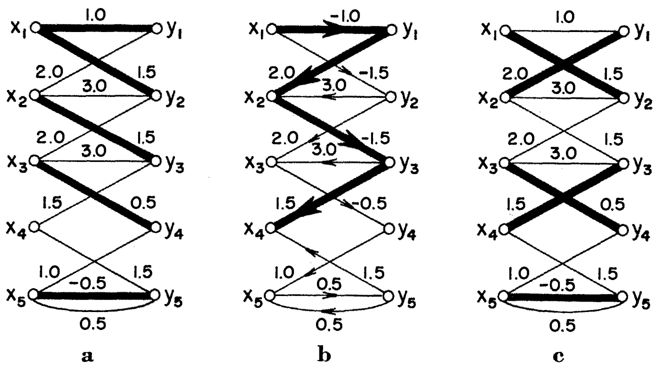

Fig. 1. A simple demonstration of the steps involved in the algorithm.

Example. Consider the weighted bipartite graph shown in Fig. 1a. Assuming $\alpha_i = \beta_i = 1$ for $i = 1, 2, 3, 4, 5$ and applying Step 1, we obtain the subgraph g_0 that is shown to consist of the darkened branches in Fig. 1a. Clearly $O(g_0) = \{x_1\}$ and $U(g_0) = \{x_4\}$. The graph $G|g_0$ is shown in Fig. 1b, and the darkened branches of this graph correspond to the least-weight directed path from $x_1 = x_{s_0}$ to x_4. Step 4 yield the subgraph g_1 indicated by the darkened branches in Fig. 1c. $O(g_1) = \Phi$. Thus, returning to Step 2, we have that $g^* = g_1$.

II. PROOF OF CONVERGENCE OF THE ALGORITHM

LET $G = [V, B]$ be a graph. A subgraph $e \subseteq G$ is called an *Euler graph*[13] if every vertex of e has even degree. We say that e is an *alternating Euler subgraph* with respect to $g \subseteq G$ if $d(v_i, e \cap \bar{g}) = d(v_i, e \cap g)$ for all $v_i \in V$. An alternating Euler subgraph e is an Euler graph, since $d(v_i, e) = d(v_i, e \cap g) + d(v_i, e \cap \bar{g}) = 2d(v_i, e \cap g)$ for all $v_i \in V$. We say that $p \subseteq G$ is a *$g\bar{g}$-alternating path* from v_i to v_j if

(1) $d(v_k, p \cap \bar{g}) = d(v_k, p \cap g)$ for all $v_k \in [V - \{v_i, v_j\}]$,

(2) $d(v_i, p \cap \bar{g}) = d(v_i, p \cap g) - 1$, and

(3) $d(v_j, p \cap \bar{g}) = d(v_j, p \cap g) + 1$.

An alternating Euler subgraph $e \subseteq G = [V, B]$ with respect to $g \subseteq G$ is said to be an *alternating circuit* with respect to g if the only alternating Euler subgraph strictly contained in e is the null subgraph. If a circuit $c \subseteq G$ is an alternating circuit with respect to g, then the branches of c alternate between being in g and in $\bar{g}$. That is, if branch (v_i, v_j) is in $c \cap g$, then the other branch of c incident at v_i is in $\bar{g}$. Conversely, if $(v_i, v_j) \epsilon c \cap \bar{g}$, then the other branch of c incident at v_i is in g. The alternating path was introduced by PETERSEN[14] and is used by both Berge[2] and Ore.[1] A $g\bar{g}$-alternating path $p \subseteq G$ is said to be an *elementary $g\bar{g}$-alternating path* if the only alternating Euler subgraph with respect to g strictly contained in p is the null subgraph.

Let g be a subgraph of a bipartite graph $G = [(X, Y), B]$. If $c \subseteq G$ is an alternating circuit with respect to g, then c is a circuit of G, and if $p \subseteq G$ is an elementary $g\bar{g}$-alternating path from $x_i \epsilon X$ to $x_j \epsilon X$, then p is an elementary path of G. Furthermore, there is a one-to-one correspondence between elementary directed paths in $G|g$ that begin and end at distinct vertices of X and elementary $g\bar{g}$-alternating paths that begin in X.

Now we briefly outline what we must prove. We must prove that when the algorithm is allowed to run until it is terminated, either the desired subgraph is obtained or it is determined that no feasible subgraph exists. Steps 1 and 4 of the algorithm require no proof. In Step 2 we must prove that if $O(g_i) = \Phi$, then g_i is a minimum-weight feasible subgraph of G. In Step 3 we must show that $O(g_i) \neq \Phi$ implies that $U(g_i) \neq \Phi$, and we must show that we can find the least-weight directed path p_i in $G|g_i$. We must prove that, if the algorithm is terminated in Step 3, then the constraints are infeasible. Finally, we must prove that the algorithm is finite.

Let g be a subgraph of $G = [(X, Y), B]$ and let $\alpha = \{\alpha_1, \alpha_2, \cdots, \alpha_p\}$ and $\beta = \{\beta_1, \beta_2, \cdots, \beta_q\}$ be the degree constraints on X and Y, respectively. We define the *index of over-exposure*, $\theta(g)$, as follows.

$$\theta(g) = \textstyle\sum_{x_i \epsilon O(g) \cap X} [d(x_i, g) - \alpha_i] + \sum_{y_j \epsilon O(g) \cap Y} [d(y_j, g) - \beta_j]. \quad (5)$$

LEMMA 1. *Let g_i be the subgraph of $G = [(X, Y), B]$ obtained after $i > 0$ cycles of the algorithm. Then $d(y_j, g_i) = \beta_j$ for all $y_j \epsilon Y$, and $\theta(g_i) = \theta(g_{i-1}) - 1$.*

Proof. $g_i = g_{i-1} \oplus p_{i-1}$, where p_{i-1} is an elementary $g_{i-1}\bar{g}_{i-1}$-alternating path from $x_{s_{i-1}} \epsilon O(g_{i-1})$ to $x_{t_{i-1}} \epsilon U(g_{i-1})$. The lemma follows directly from (2), from the definition of an alternating path, and from the fact that $d(y_j, g_0) = \beta_j$ for all $y_j \epsilon Y$.

A direct consequence of Lemma 1 is that $O(g_i) \subset X$ and $U(g_i) \subset X$.
LEMMA 2. *Let g_i be the subgraph of $G = [(X, Y), B]$ obtained after $i \geq 0$ cycles of the algorithm. Then $O(g_i) \neq \Phi$ implies that $U(g_i) \neq \Phi$.*

Proof. This lemma follows directly from our assumption that $\sum_{k=1}^{k=p} \alpha_k = \sum_{j=1}^{j=q} \beta_j$, from the fact that $\sum_{k=1}^{k=p} d(x_k, g_i) = \sum_{j=1}^{j=q} d(y_j, g_i)$, and from Lemma 1.

THEOREM 1. *Let $G=[V, B]$ be a weighted graph, and let $g \subseteq G$ be any subgraph of G. There exists a subgraph $h \subseteq G$ such that $d(v_i, h) = d(v_i, g)$ for all $v_i \epsilon V$ and such that $w(h) < w(g)$ if and only if there exists an alternating Euler subgraph $e \subseteq G$ with respect to g such that $w(e|g) < 0$.*

Proof. Let $e \subseteq G$ be an alternating Euler subgraph with respect to g such that $w(e|g) < 0$. By (2), $d(v_i, g \oplus e) = d(v_i, g)$ for all $v_i \epsilon V$. By (4), $w(g \oplus e) = w(g) + w(e|g)$. Therefore $w(g \oplus e) < w(g)$. Conversely, let $h \subseteq G$ be a subgraph such that $d_i(v_i, h) = d(v_i, g)$ for all $v_i \epsilon V$ and such that $w(h) < w(g)$. In (2) let $g_1 = g$ and $g_2 = g \oplus h$. Therefore, $d(v_i, \bar{g} \cap [g \oplus h]) = d(v_i, g \cap [g \oplus h])$ for all $v_i \epsilon V$, so that $g \oplus h$ is an alternating Euler subgraph with respect to g. By (4), $w(g \oplus h|g) = w(h) - w(g) < 0$.

THEOREM 2. *Let $G=[V, B]$ be a weighted graph, and let g be a subgraph of G such that no alternating Euler subgraph of G with respect to g has negative weight in $G|g$. Let p be a $g\bar{g}$-alternating path from $v_i \epsilon V$ to $v_j \epsilon V$ that has least weight in $G|g$. Then no alternating Euler subgraph of G with respect to $g \oplus p$ has negative weight in $G|(g \oplus p)$.*

Proof. Let $|V| = n$, and let $|B| = m$. Add a vertex v_{n+1} to G, add a branch b_{m+1} between v_j and v_{n+1}, and add a branch b_{m+2} between v_i and v_{n+1}. Let the weight of b_{m+1} be $w(p|g)$ and let the weight of b_{m+2} be zero. Let $G' = [V \cup \{v_{n+1}\}, B \cup \{b_{m+1}, b_{m+2}\}]$ be the weighted graph thus defined, and let $g' = g \cup \{b_{m+1}\}$. There are no alternating Euler subgraphs of G' with respect to g' that have negative weight in $G'|g'$, for the existence of such a subgraph would contradict the conditions on p. By construction, $e' = p \cup \{b_{m+1}, b_{m+2}\}$ is an alternating Euler subgraph of G' with respect to g', and $w(e'|g') = 0$. Let $g^* = g' \oplus e'$. By (4), $w(g^*) = w(g')$. By Theorem 1, there exist no alternating Euler subgraphs of G' with respect to g^* that have negative weight in $G'|g^*$. Removing branches cannot create new Euler subgraphs, and $g \oplus p = g^* - \{b_{m+2}\}$. Therefore no alternating Euler subgraph of G with respect to $g \oplus p$ has negative weight in $G|(g \oplus p)$.

THEOREM 3. *Let g_i be the subgraph of $G=[(X, Y), B]$ obtained after $i \geqq 0$ cycles of the algorithm. If $O(g_i) = \Phi$, then $g_i = g^*$.*

Proof. Clearly there are no alternating Euler subgraphs of G with respect to g_0 that have negative weight in $G|g_0$. By repeated applications of Theorem 2, there are no alternating Euler subgraphs of G with respect to g_i that have negative weight in $G|g_i$. If $O(g_i) = \Phi$, then g_i is a feasible subgraph of G. By Theorem 1, g_i is a least-weight feasible subgraph of G.

Let g be a subgraph of the bipartite graph $G = [(X, Y), B]$. By construction, an elementary directed path $p \subseteq G|g$ which begins at a vertex $x_s \epsilon X$ and ends at a vertex $x_t \epsilon X$ is an elementary $g\bar{g}$-alternating path in G

from x_s to x_t. Conversely, an elementary $g\bar{g}$-alternating path in G from x_s to x_t is a directed path in $G|g$ that begins at x_s and ends at x_t. Let g_i be the subgraph obtained after $i \geqq 0$ cycles of the algorithm. By Theorem 2, there exist no alternating circuits in G with respect to g_i that have negative weight in $G|g_i$. Therefore, $G|g_i$ contains no negative-weight directed circuits. Ford and Fulkerson[11] have given an efficient algorithm for finding least-weight directed paths in a weighted directed graph. They prove that the algorithm converges if the directed graph contains no negative-weight directed circuits. Using their algorithm, we can find the least-weight directed paths in $G|g_i$, as is required in Step 3 of our algorithm. It should be pointed out that $G|g_i$ $(i \geqq 1)$ can be obtained directly from $G|g_{i-1}$ by reversing the orientation and changing the sign of the weight of all branches in p_{i-1}.

Our algorithm is finite by Lemma 1 and by the fact that $\theta(g_0)$ is finite. It remains to be shown that if the constraints are feasible and if $O(g_i) \neq \Phi$, then there exists a directed path in $G|g_i$ that begins at $x_{s_i} \epsilon O(g_i)$ and ends at a vertex of $U(g_i)$. The proof of convergence of the algorithm is completed with the following theorem.

THEOREM 4. *Let g_i be the subgraph of $G = [(X, Y), B]$ obtained after $i \geqq 0$ cycles of the algorithm, and let $x_{s_i} \epsilon O(g_i)$. If the constraints are feasible, then there exists an elementary $g_i\bar{g}_i$-alternating path in G from x_{s_i} to some vertex of $U(g_i)$.*

Proof. Let g be a feasible subgraph of G, and consider $g \oplus g_i$. Remove from $g \oplus g_i$ any alternating Euler subgraphs with respect to g_i, and call the remaining subgraph h. It follows from (2) that $d(v, h \cap \bar{g}_i) = d(v, h \cap g_i)$ for all $v \epsilon \{[X \cup Y] - [U(g_i) \cup O(g_i)]\}$ and that $d(v, h \cap \bar{g}_i) < d(v, h \cap g_i)$ for all $v \epsilon O(g_i)$. Let $p \subseteq h$ be an elementary $g_i\bar{g}_i$-alternating path that begins at x_{s_i} and contains a maximum number of branches. Let us denote $p = (x_{s_i}, y_{k_1})(y_{k_1}, x_{k_2})(x_{k_2}, y_{k_3}) \cdots (y_{k_n}, x_{k_n})$. Suppose $x_{k_n} \epsilon [X - U(g_i)]$. There exists a branch $(x_{k_n}, y_{k_{n+1}})$ of $h \cap g_i$ incident at x_{k_n}, since $d(x_{k_n}, h \cap g_i) \geqq d(x_{k_n}, h \cap \bar{g}_i)$ and since $(y_{k_n}, x_{k_n}) \epsilon h \cap \bar{g}_i$. Since h contains no alternating Euler subgraphs with respect to g_i, $d(y_{k_{n+1}}, p) = 0$. There exists a branch $(y_{k_{n+1}}, x_{k_{n+1}})$ of $h \cap \bar{g}_i$ incident at $x_{k_{n+1}}$, since $d(y_{k_{n+1}}, h \cap g_i) = d(y_{k_{n+1}}, h \cap \bar{g}_i)$. Since h contains no alternating Euler subgraphs with respect to g_i, $d(x_{k_{n+1}}, p) = 0$. But then $p' = p \cup \{(x_{k_n}, y_{k_{n+1}}), (y_{k_{n+1}}, x_{k_{n+1}})\}$ is an elementary $g_i\bar{g}_i$-alternating path contained in h, beginning at x_{s_i} and containing more branches than p. This contradicts the maximality of p. Thus $x_{k_n} \epsilon U(g_i)$, and the proof is complete.

III. EFFICIENCY OF THE ALGORITHM

WE WANT to obtain an upper bound on the number of operations that would be required if our algorithm were used to find a minimum-weight degree-constrained subgraph of a bipartite graph. Let $G = [(X, Y), B]$

be a given weighted bipartite graph, where $|X|=p$ and $|Y|=q$, and let $\alpha=\{\alpha_1, \alpha_2, \cdots, \alpha_p\}$ and $\beta=\{\beta_1, \beta_2, \cdots, \beta_q\}$ be given degree constraints on X and Y respectively. Without loss of generality we assume that $p=\min(p, q)$. It follows that there exists no elementary directed path in $\boldsymbol{G}|g_i$ that contains more than $2p$ branches, where g_i is the subgraph of G obtained after $i \geqq 0$ cycles of our algorithm. The process of labeling vertices (Ford and Fulkerson's algorithm), which may be used in order to find the least-weight directed paths in $\boldsymbol{G}|g_i$, involves comparing vertex weights. At each step of their algorithm, at least one vertex receives its final label. It can be shown that no more than $p(q-j+1)$ comparisons are required at the $2j$th step, and no more than $q(p-j)$ comparisons are required at the $(2j+1)$th step. The index j runs from 1 through $p-1$, since no directed path contains more than $2p$ branches. Therefore, the number of comparisons M that are required in order to find the least-weight directed path in $\boldsymbol{G}|g_i$ is no more than $\frac{1}{2}\, p(p-1)(3q-p+2)$.

It is elementary to show that the initial index of over-exposure, $\theta(g_0)$, is less than $N=\sum_{i=1}^{i=p} \alpha_i=\sum_{j=1}^{j=q} \beta_j$. Therefore, in order to find a least-weight feasible subgraph of $G=[(X, Y), B]$, Step 3 of our algorithm is used less than N times. Thus, the time required for our algorithm should be, at the very worst, proportional to $N(\frac{1}{2}|X|^2)(3|Y|-|X|)$. If we apply this bound to the job-assignment problem where $|X|=|Y|=N$, we see that the number of comparisons required by our algorithm is at worst N^4.

Munkres[12] gives an algorithm that can be used to solve the job-assignment problem; it has a theoretical upper bound of N^4 to its speed. We have shown that our algorithm has the same theoretical upper bound. Kurtzberg[15] has shown that the computing time for Munkres' algorithm goes roughly as N^3. The computation time for our algorithm will be discussed in the next section.

IV. INCREASING THE EFFICIENCY OF OUR METHOD

The number of operations required for our algorithm to find a minimum-weight feasible subgraph of a weighted bipartite graph is strongly dependent on the method used for finding least-weight elementary directed paths. Our algorithm of Section I does not specify how to find such paths, so we are free to use any means at our disposal. In this section we present results that significantly increase the efficiency of our method. The algorithm of Ford and Fulkerson that can be used to find least-weight directed paths will be referred to as the labeling process. As a result of Theorem 5 below, we can make two changes in the algorithm as it was interpreted in the last section. First of all, we may be able to reduce the index of over-exposure by more than one with a single labeling. Second, subsequent labeling processes can be initiated at an advanced stage.

Let $G=[(X, Y), B]$ be a weighted bipartite graph, where $|X|=p$ and

$|Y|=q$, and let $\alpha=\{\alpha_1, \alpha_2, \cdots, \alpha_p\}$ and $\beta=\{\beta_1, \beta_2, \cdots, \beta_q\}$ be the degree constraints on X and Y respectively.

Revised Algorithm

Step 1. Choose g_0 as in Step 1 of the algorithm of Section I. If $O(g_0)=\Phi$, then $g^*=g_0$, and the algorithm is terminated. Otherwise, initiate the labeling process at each vertex of $O(g_0)$. That is, set the weight of the label at each vertex of $O(g_0)$ equal to zero, and set the weight of the label at each vertex of $X-O(g_0)$ equal to infinity. Go to Step 2.

Step 2. Let g_i be the subgraph of G obtained thus far. (We start with $i=0$.) Set the weight of the label at each vertex of Y equal to infinity. Apply the labeling process to $\mathbf{G}|g_i$. Go to Step 3.

Step 3. Set $k=i$. Now the vertex labels can be used to determine least-weight elementary directed paths from vertices in $O(g_i)$ to vertices in $X-O(g_i)$. Let $p(x_j)$ for $x_j \epsilon X-O(g_i)$ be the elementary directed path from $s(x_j)\epsilon O(g_i)$ to x_j that is indicated by the labels. Let $U(g_i)=\{x_{j_1}, x_{j_2}, \cdots, x_{j_n}\}$. By a slight modification of Theorem 4, the label on x_{j_1} must have a finite weight, or the constraints are infeasible. Let $g_{i+1}=g_i\oplus p(x_{j_1})$. Go to Step 5 if $n=1$, and go to Step 4 if $n>1$.

Step 4. Do this step for $m=2, 3, \cdots, n$. If $p(x_{j_m})\cap[\bigcup_{l=1}^{m-1} p(x_{j_l})]=\Phi$ and $s(x_{j_m})\epsilon O(g_{i+m-1})$, then $g_{i+m}=g_{i+m-1}\oplus p(x_{j_m})$. Otherwise $g_{i+m}=g_{i+m-1}$.

Step 5. If $O(g_{i+n})=\Phi$, then $g^*=g_{i+n}$, and the algorithm is terminated. Otherwise do the following for each $x_m\epsilon X-O(g_i)$. Let $P=\bigcup_{l=1}^{l=n} p(x_{j_l})$. If $p(x_m)\cap P=\Phi$, do not change the label on x_m. Otherwise, set the weight of the label at x_m equal to infinity. Set the weight of the label at each vertex of $O(g_i)-O(g_{i+n})$ equal to infinity. Set $i=k+n$, and go to Step 3.

To prove that this revised algorithm will find the desired subgraph g^*, it is sufficient to prove the following. If p_1 and p_2 are least-weight branch disjoint elementary directed paths in $\mathbf{G}|g_i$ from z_1, $z_2\epsilon O(g_i)$ to x_1, $x_2\epsilon X-O(g_i)$, then p_2 is a least-weight elementary directed path in $\mathbf{G}|g_i\oplus p_1$ from z_2 to x_2.

Let $G=[V, B]$ be a given weighted graph, and let U be a nonempty proper subset of V. We will let $U=\{u_1, u_2, \cdots, u_k\}$ and $V-U=\{v_1, v_2, \cdots, v_n\}$. Let $g\subseteq G$ be a subgraph such that there are no alternating Euler subgraphs of G with respect to g that have negative weight in $G|g$. Now let $p_{ij}\subseteq G$ be a $g\bar{g}$-alternating path from u_i to v_j that has least weight in $G|g$. We assume such paths exist between every $u_i\epsilon U$ and $v_j\epsilon V-U$. If, in fact, there is no $g\bar{g}$-alternating path p_{ij} from u_i to v_j, then $w(p_{ij}|g)=\infty$. We say that v_j is a *$g\bar{g}$ descendant* of u_i with respect to U if $w(p_{ij}|g)\leqq w(p_{mj}|g)$ for $l\leqq m\leqq k$. A $g\bar{g}$-alternating path p from u_i to v_j that satisfies $w(p|g)\leqq w(p_{mj}|g)$ for $l\leqq m\leqq k$ is called a *$g\bar{g}$-descendant path* from u_i to v_j with respect to U.

Theorem 5. *Let $G=[(X, Y), B]$ be a weighted bipartite graph, and let $g\subseteq G$ be a subgraph such that no alternating Euler subgraph of G with respect to g has* *negative weight in $G|g$. Let Z be a nonempty proper subset of X. Let p_1 be a $g\bar{g}$-descendant path from $z_1 \epsilon Z$ to $x_1 \epsilon X-Z$ with respect to Z, and let p_2 be a $g\bar{g}$-descendant path from $z_2 \epsilon Z$ to $x_2 \epsilon X-Z$ with respect to Z. (The vertices z_1 and z_2 need not be distinct, nor do the vertices x_1 and x_2.) If $p_1 \cap p_2 = \Phi$, then p_2 is a $g \oplus p_1, \overline{g \oplus p_1}$ descendant path from z_2 to x_2 with respect to Z.*

Proof. Let $n=|Y|$, and let $m=|B|$. Add two vertices y_{n+1} and y_{n+2} to G, add a branch b_{m+1} between y_{n+1} and x_1, add a branch b_{m+2} between y_{n+1} and z_1, add a branch b_{m+3} between y_{n+2} and x_2, and add a branch b_{m+4} between y_{n+2} and z_2. Let $w(b_{m+1})=w(p_1|g)$, $w(b_{m+2})=0$, $w(b_{m+3})=w(p_2|g)$ and $w(b_{m+4})=0$. Let $G^*=[(X, Y \cup \{y_{n+1}, y_{n+2}\}), B \cup \{b_{m+1}, b_{m+2}, b_{m+3}, b_{m+4}\}]$ be the weighted bipartite graph thus defined, and let $g^*=g \cup \{b_{m+1}, b_{m+3}\}$. It can be shown that there exists no alternating Euler subgraph of G^* with respect to g^* that has negative weight in $G^*|g^*$. Let $e_1=p_1 \cup \{b_{m+1}, b_{m+2}\}$, and let $g'=g^* \oplus e_1$. By construction, e_1 is an alternating Euler subgraph of G^* with respect to g^*, and $w(e_1|g^*)=0$. By Theorem 1, there are no alternating Euler subgraphs of G^* with respect to g' that have negative weight in $G^*|g'$. Suppose p_2 is not a $g'\bar{g}'$-descendant path in G^* from z_2 to x_2 with respect to Z. Then there exists a $g'\bar{g}'$-alternating path from $z_3 \epsilon Z$ to x_2 such that $w(p_3|g')<w(p_2|g')$. If $b_{m+1} \epsilon p_3$, then $p_3 \oplus e_1$ is a $g\bar{g}$-alternating path from z_3 to x_2 such that $w(p_3 \oplus e_1|g)<w(p_2|g)$. This is a contradiction. If $b_{m+1} \notin p_3$, then p_3 is a $g\bar{g}$-alternating path from z_3 to x_2 such that $w(p_3|g)<w(p_2|g)$. This is a contradiction. Therefore p_2 is a $g'\bar{g}'$-descendant path in G^* from z_2 to x_2 with respect to Z. Therefore p_2 is a $g \oplus p_1$, $\overline{g \oplus p_1}$-descendant path in G from z_2 to x_2 with respect to Z.

The experimental data in this part was obtained by considering randomly generated, fully connected, n by n bipartite graphs. For each n and for each degree constraint, 20 problems were generated and solved. The average CDC 6400 computer times given in Table I and in Fig. 2 do not include the computer time required to generate the graphs, nor do these times include input-output times. The random numbers generated were integers between 0 and 999.

From Fig. 2 we see that the computer time needed to find a minimum-weight feasible subgraph varies roughly as n^2, a significant improvement over the Munkres algorithm, as reported by Kurtzberg. The numbers of comparisons made during the labeling process were counted for these same tests. A plot of n versus the number of comparisons was made[16]; and the number of comparisons varies roughly as $n^{2.5}$ for $10 \leqq n \leqq 50$. For very large n, therefore, we would expect the computer time to vary like $n^{2.5}$.

V. THE TRANSPORTATION PROBLEM

IN THIS SECTION, we show that the transportation problem of integer programming can be solved through the use of our algorithm. Two versions of the transportation problem, T1 and T2, are given below. T1 is more general than the classical transportation problem, which is often referred to as the Hitchcock-Koopmans Transportation Problem.[17] Its increased generality is due to the fact that the objective function is convex-separable rather than only linear as in the classical transportation problem. An objective function $z=\sum_{j=1}^{j=n} f_j(x_j)$, where f_j is a convex function, is said to be *convex-separable*.[18]

TABLE I

CDC 6400 Computer Time (in Seconds) Used to Find an Optimal Feasible Subgraph

All degree constraints are equal to γ	$n = 10$		$n = 20$		$n = 30$		$n = 40$		$n = 50$	
	Average	Extremes	Average	Extremes	Average	Extremes	Average	Extremes	Average	Extremes
$\gamma = 1$	0.027	0.020 0.040	0.116	0.90 0.162	0.262	0.188 0.426	0.436	0.312 0.600	0.744	0.566 1.100
$\gamma = 2$	0.036	0.024 0.046	0.154	0.118 0.198	0.350	0.278 0.550	0.620	0.472 0.900	1.028	0.844 1.354
$\gamma = 3$	0.045	0.030 0.056	0.193	0.150 0.250	0.467	0.374 0.654	0.852	0.578 1.124	1.316	0.954 1.914
$\gamma = 4$	0.056	0.040 0.090	0.226	0.164 0.304	0.528	0.388 0.726	1.006	0.832 1.402	1.609	1.306 2.104
$\gamma = 5$	0.061	0.048 0.080	0.250	0.178 0.298	0.585	0.430 0.776	1.150	0.766 1.618	1.892	1.306 2.380
$\gamma = 6$	0.064	0.052 0.086	0.289	0.234 0.354	0.664	0.526 0.802	1.216	0.946 1.754	2.197	1.620 3.110

There are several sets of doubly subscripted numbers or variables in the discussion below. In certain places it will be more convenient to represent these sets by the obvious associated matrix. When we consider these as sets, they will be represented by the lower-case letters, and when we consider these as matrices they will be represented by the corresponding upper-case letters. For example, if $w=\{w_{ij}|1\leqq i\leqq p$ and $1\leqq j\leqq q\}$, then $W=[w_{ij}]$ will denote the corresponding $p\times q$ matrix.

T1. *Find integers t_{ij} for $1\leqq i\leqq n$ and $1\leqq j\leqq m$ that minimize $h(T)=\sum_{i=1}^{i=n}\sum_{j=1}^{j=m} h_{ij}(t_{ij})$ (where the h_{ij} are given convex functions), subject to the con-*

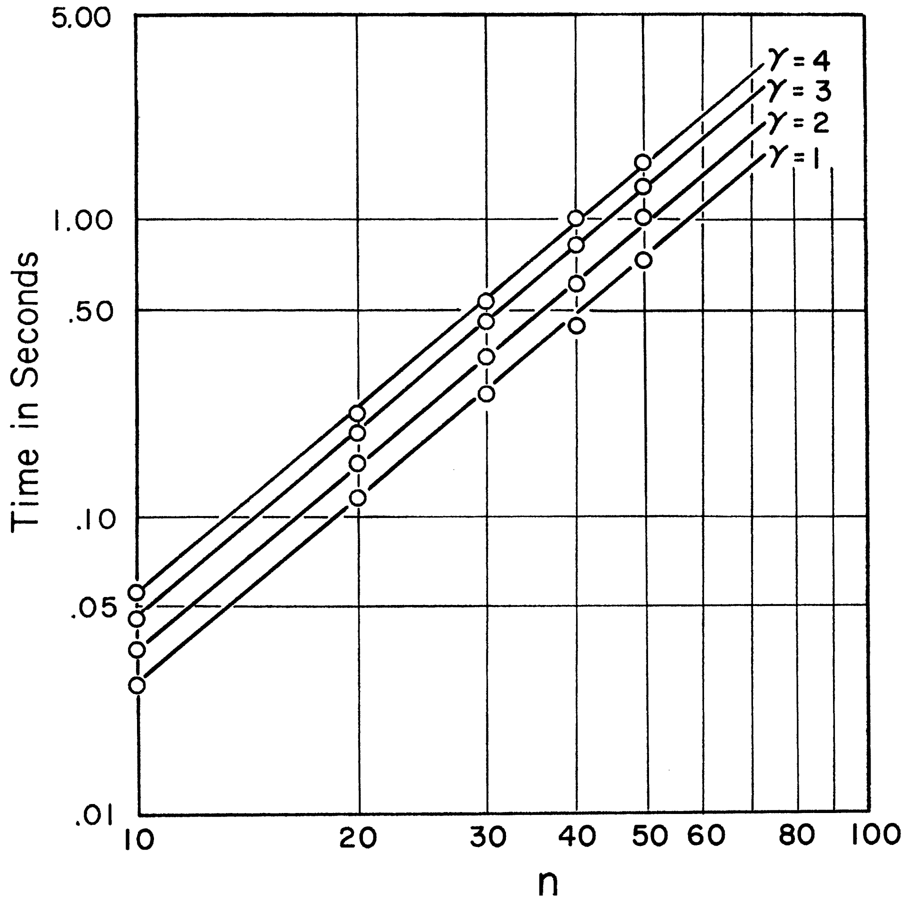

Fig. 2. A plot showing how the computer solution time varies with n for $\gamma=1, 2, 3, 4$.

straints:

(1) $0 \leqq a_i \leqq \sum_{j=1}^{j=m} t_{ij} \leqq b_i$ *for* $1 \leqq i \leqq n$,

(2) $0 \leqq c_j \leqq \sum_{i=1}^{i=n} t_{ij} \leqq d_j$ *for* $1 \leqq j \leqq m$,

(3) $0 \leqq l_{ij} \leqq t_{ij} \leqq k_{ij}$ *for* $1 \leqq i \leqq n$ and $1 \leqq j \leqq m$.

The a_i, b_i, c_j, d_j, l_{ij} *and* k_{ij} *are given nonnegative integers.*

T2. *Find nonnegative integers* s_{ij} *for* $1 \leqq i \leqq p$ *and* $1 \leqq j \leqq q$ that minimize $f(S) = \sum_{i=1}^{i=p} \sum_{j=1}^{i=q} f_{ij}(s_{ij})$ [*where the* f_{ij} *are given convex functions that satisfy* $f_{ij}(0) = 0$], *subject to the constraints:*

(1) $\sum_{j=1}^{j=q} s_{ij}=\alpha_i$ *for* $1\leqq i\leqq p$,

(2) $\sum_{i=1}^{i=p} s_{ij}=\beta_j$ *for* $1\leqq j\leqq q$,

(3) $s_{ij}\leqq\lambda_{ij}$ *for* $1\leqq i\leqq p$ *and* $1\leqq j\leqq q$.

The α_i, β_j, *and* λ_{ij} *are given nonnegative integers.*

Clearly T2 is a special case of T1. Let T1 be given and make the following definitions:

(a) $p=n+1$, $q=m+1$.

(b) $\alpha_i=b_i-\sum_{j=1}^{j=m} l_{ij}$, $\lambda_{iq}=b_i-a_i$ and $f_{iq}=0$ for $1\leqq i\leqq n$.

(c) $\beta_j=d_j-\sum_{i=1}^{i=n} l_{ij}$, $\lambda_{pj}=d_j-c_j$ and $f_{pj}=0$ for $1\leqq j\leqq m$.

(d) $\lambda_{ij}=k_{ij}-l_{ij}$ and $f_{ij}(s_{ij})=h_{ij}(s_{ij}+l_{ij})-h_{ij}(l_{ij})$ for $1\leqq i\leqq n$ and $1\leqq j\leqq m$.

(e) $\alpha_p=\sum_{j=1}^{j=m}(d_j-c_j)$, $\beta_q=\sum_{i=1}^{i=n} b_i-\sum_{j=1}^{j=m} c_j$, $\lambda_{pq}=\sum_{j=1}^{j=m}(d_j-c_j)$ and $f_{pq}=0$.

Problem T2 that we have thus defined will be denoted T2(T1). It can be shown[16] that these two problems are equivalent. That is, given a feasible (optimal feasible) solution $t=\{t_{ij}\}$ of T1, we can find a feasible (optimal feasible) solution $s(t)=\{s_{ij}(t)\}$ of T2(T1); and conversely, given a feasible (optimal feasible) solution $s=\{s_{ij}\}$ of T2(T1), we can find a feasible (optimal feasible) solution $t(s)=\{t_{ij}(s)\}$ of T1.

T2 is the form of the transportation problem that we wish to consider. Let $G=[(X, Y), B]$ be the weighted bipartite graph described below. $X=\{x_1, x_2, \cdots, x_p\}$ and $Y=\{y_1, y_2, \cdots, y_q\}$. Let $x_i\epsilon X$ and let $y_j\epsilon Y$ and define $k_{ij}=\min(\alpha_i, \beta_j, \lambda_{ij})$. In G between x_i and y_j are branches $b_{ij}^1, b_{ij}^2, \cdots, b_{ij}^{k_{ij}}$, and the weight $w(b_{ij}^m)$ of b_{ij}^m is $f_{ij}(m)-f_{ij}(m-1)$ for $1\leqq m\leqq k_{ij}$. If $k_{ij}=0$, there are no branches in G between x_i and y_j. Since f_{ij} is convex and since $f_{ij}(0)=0$, we have that $w(b_{ij}^1)\leqq w(b_{ij}^2)\leqq\cdots\leqq w(b_{ij}^{k_{ij}})$ and that $\sum_{n=1}^{n=m} w(b_{ij}^n)=f_{ij}(m)$ for $1\leqq m\leqq k_{ij}$. The bipartite graph G is thus well defined. Let $\alpha=\{\alpha_1, \alpha_2, \cdots, \alpha_p\}$ and $\beta=\{\beta_1, \beta_2, \cdots, \beta_q\}$ be the degree constraints on X and Y respectively, where the α_i and β_i come from T2.

We want to define a subgraph $g(s)$ of G for each feasible solution s of T2. Let $s=\{s_{ij}\}$ be a feasible solution of T2. Then $g(s)$ is the subgraph of G consisting of branches b_{ij}^m for $1\leqq m\leqq s_{ij}$, and if $s_{ij}=0$, then there is no branch between v_i and v_j, for $1\leqq i\leqq p$ and for $1\leqq j\leqq q$. Then $g(s)$ is a feasible subgraph of G and $w[g(s)]=f(S)$. Conversely, let h be any feasible subgraph of G. We define $s_{ij}(h)$ to be equal to the number of branches of h that are incident between x_i and y_j. If $s(h)=\{s_{ij}(h)\}$, it follows from the feasibility of h that $s(h)$ is a feasible solution of T2 and that $f[S(h)]\leqq w(h)$.

Suppose now that the constraints are feasible and that g^* is the solution subgraph obtained through the use of our algorithm. Then $f[S(g^*)]\leqq w(g^*)$. Suppose s^* is a feasible solution of T2 and that $f(S^*)<f[S(g^*)]$. Then $g(s^*)$ is a feasible subgraph of G, and $w[g(s^*)]<w(g^*)$. This is a con-

tradiction. Therefore $s(g^*)$ is an optimal feasible solution of T2, and we have shown that our algorithm can be used to solve T2.

VI. EXTENSION TO THE NONBIPARTITE CASE

LET H be a weighted graph (not necessarily bipartite) with vertices $V=\{v_1, v_2, \cdots, v_n\}$. Let $\gamma=\{\gamma_1, \gamma_2, \cdots, \gamma_n\}$ be a given set of nonnegative integers called the degree constraints on V. A subgraph h of H is said to be feasible if $d(v_j, h)=\gamma_j$ for all $v_j \epsilon V$. The constraints are said to be feasible if there exists at least one feasible subgraph of H. Otherwise the constraints are infeasible. Clearly two necessary conditions for the feasibility of the constraints are (1) $d(v_j, H) \geqq \gamma_j$ for all $v_j \epsilon V$ and (2) $\sum_{j=1}^{j=n} \gamma_j = 2N$, where N is a nonnegative integer. We would like either to find a minimum-weight feasible subgraph h^* of H or to determine that the constraints are infeasible. Let $h \subseteq H$ be any subgraph of H. As in the case of bipartite graphs, we define:

$$O(h)=\{v_i \epsilon V | d(v_i, h) > \gamma_i\},$$

$$U(h)=\{v_i \epsilon V | d(v_i, h) < \gamma_i\},$$

$$\theta(h)=\sum_{v_i \epsilon O(h)} [d(v_i, h)-\gamma_i].$$

An algorithm that solves this more general problem is now presented. It must be pointed out that we do not have as efficient an algorithm for finding the alternating paths as is necessary in Step 3. By Theorem 2, however, there are no alternating Euler subgraphs with respect to h_i' that have negative weight in $H_i'|h_i'$, where h_i' is the subgraph of H_i' obtained after i cycles of the algorithm. This fact offers us some encouragement as we search for a method of finding alternating paths.

ALGORITHM

Step 1. Let h_0 be a subgraph of H consisting of $N=\frac{1}{2}\sum_{j=1}^{j=n} \gamma_j$ least-weight branches of H. Thus h_0 is a minimum-weight subgraph of H which contains N branches. Add two vertices v_o and v_u. Let W be a large positive number. For each $v_i \epsilon U(h_0)$, add $\gamma_i - d(v_i, h_0)$ branches of weight W between v_i and v_o. For each $v_j \epsilon O(h_0)$, add $d(v_j, h_0)-\gamma_j$ branches of weight $-W$ between v_j and v_u. Let this new graph be denoted H'. Define $\gamma_i'=\gamma_i$ for all $v_i \notin O(h_0)$, and $\gamma_i'=d(v_i, h_0)$ for all $v_i \epsilon O(h_0)$. Finally let $\gamma_o'=0$ and $\gamma_u'=\theta(h_0)$, and let h_0' be h_0 plus all the branches incident at v_o. With these newly defined constraints, $\gamma'=\{\gamma_1', \gamma_2', \cdots, \gamma_n', \gamma_o', \gamma_u'\}$, we have $O(h_0')=\{v_o\}$ and $U(h_0')=\{v_u\}$.

Step 2. Let h_i' be the subgraph of H' obtained after i cycles of the algorithm. (We start with $i=0$.) If $O(h_i')=\Phi$, then h^* is obtained from h_i' by removing all branches incident at v_u, and the algorithm is terminated. Otherwise go to Step 3.

Step 3. (Search for an alternating path.) Find the $h_i'\overline{h_i'}$-alternating path from v_o to v_u that has least weight in $H'|h_i'$. If no such path exists, terminate the algorithm, for the constraints are infeasible. Otherwise, let p_i' denote the path found, and go to Step 4.

Step 4. $h_{i+1}' = h_i' \oplus p_i'$. Return to Step 2.

The proof of this algorithm is similar in nature to the proof of the algorithm in Section I. The main theorems of this proof are Theorems 1 and 2, which have already been proved for the general case.

It was shown in Section V that the transportation problem can be solved by our algorithm of Section I. The class of problems that can be solved by this first algorithm is a subclass of the class of problems that can be solved by this latest algorithm. This larger class of integer programming problems is solved if an efficient method can be found to determine least-weight $g\bar{g}$-alternating paths in a graph that contains no negative-weight alternating Euler subgraphs with respect to g.

VII. CONCLUSIONS

WE HAVE presented an algorithm that will yield a minimum-weight degree-constrained subgraph of a weighted bipartite graph. This algorithm, based entirely on the theory of graphs, can be used to solve the well known transportation problem of integer programming. On the basis of computational experience, this algorithm is more efficient than the algorithm of Munkres. The concepts introduced in the case of bipartite graphs apply to the more general problem of finding a minimum-weight degree-constrained subgraph in a weighted, not necessarily bipartite, graph. In this generalization, the efficiency of our algorithm is poor, because we have no efficient algorithm for finding alternating paths with respect to a given subgraph. The discovery of such an algorithm would result in a method for solving a large class of integer programming problems.

Although it was not explicitly pointed out, the algorithm of Section I applies equally as well to the problem of finding a minimum-weight subgraph g^* of $G = [(X, Y), B]$ such that $d(y_j, g^*) = \beta_j$ for all $y_j \epsilon Y$ and $d(x_i, g^*) \leqq \alpha_i$ for all $x_i \epsilon X$. The application of the algorithm to this problem does not require the addition of 'slack branches' as would be suggested by our discussion in Section VI.

ACKNOWLEDGMENTS

THE AUTHORS with to thank S. G. WASILEW for his Fortran IV coding of the algorithm of Section IV. This work was supported by the Air Force Office of Scientific Research, Office of Aerospace Research, United States Air Force.

REFERENCES

1. O. ORE, *Theory of Graphs*, Amer. Math. Soc. Colloquium Publication, Providence, R. I., 1962.

2. C. Berge, *The Theory of Graphs,* Methuen, London, 1962.
3. R. Z. Norman and M. O. Rabin, "An Algorithm for a Minimum Cover of a Graph," *Proc. Amer. Math. Soc.* **10,** 315–319 (1959).
4. J. Edmonds, "Paths, Trees and Flowers," *Can. Math. J.* **17,** 449–467 (1965).
5. ———, "Maximum Matchings and a Polyhedron with 0–1 Vertices," *J. Res. Nat. Bur. Stand.-B, Math. and Math. Phys.* **69B,** 125–130 (1965).
6. A. J. Goldman, "Optimal Matchings and Degree Constrained Subgraphs," *J. Res. Nat. Bur. Stand.-B, Math. and Math. Phys.* **68B,** 27–29 (1964).
7. C. Witzgall and C. T. Zahn, Jr., "Modification of Edmonds' Maximum Matching Algorithm," *J. Res. Nat. Bur. Stand.-B, Math. and Math. Phys.* **69B,** 91–98 (1965).
8. F. L. Hitchcock, "The Distribution of a Product from Several Sources to Numerous Localities," *J. Math. Phys.* **20,** 224–230 (1941).
9. H. W. Kuhn, "The Hungarian Method for the Assignment Problem," *Naval Res. Log. Quart.* **2,** 83–97 (1955).
10. ———, "Variants of the Hungarian Method for the Assignment Problem," *Naval Res. Log. Quart.* **3,** 253–258 (1956).
11. L. R. Ford, Jr. and D. R. Fulkerson, *Flows in Networks,* Princeton Univ. Press, Princeton, N. J., 1962.
12. J. Munkres, "Algorithms for the Assignment and Transportation Problems," *J. SIAM* **5,** 32–38 (1957).
13. S. Seshu and M. B. Reed, *Linear Graphs and Electrical Networks,* Addison-Wesley, Reading, Mass., 1961.
14. J. Petersen, "Die Theorie der regularen Graphs," *Acta Math.* **15,** 193 (1891).
15. J. M. Kurtzberg, "On Approximation Methods for the Assignment Problem," *JACM* **9,** 419–439 (1962).
16. J. F. Desler, "Degree Constrained Subgraphs, Covers, Codes and K-Graphs," Ph.D. dissertation, Northwestern University, Evanston, Illinois, 1969.
17. G. B. Dantzig, *Linear Programming and Extensions,* Princeton Univ. Press, Princeton, N. J., 1963.
18. A. Charnes and C. E. Lemke, "Minimization of Non-Linear Separable Convex Functionals," *Naval Res. Log. Quart.* **1,** 301–312 (1954).