A Linear Programming Approach for the Weighted Graph Matching Problem

H. A. Almohamad and S. O. Duffuaa

Abstract—**A linear programming (LP) approach is proposed for the weighted graph matching problem. A linear program is obtained by formulating the graph matching problem in L_1 norm and then transforming the resulting quadratic optimization problem to a linear one. The linear program is solved using a Simplex-based algorithm. Then, approximate 0–1 integer solutions are obtained by applying the Hungarian method on the real solutions of the linear program. The complexity of the proposed algorithm is polynomial time, and it is $O(n^6L)$ for matching graphs of size n. The developed algorithm is compared to two other algorithms. One is based on an eigendecomposition approach and the other on a symmetric polynomial transform. Experimental results showed that the LP approach is superior in matching graphs than both other methods.**

Index Terms—**Graph matching, Hungarian method, linear programming, optimization, recognition, structural pattern.**

I. Introduction

In pattern recognition, structured data of an object can be partitioned into a weighted graph [1], [2] defined by a set of vertices (nodes) and edges (weighted arcs). The problem of matching two objects represented by weighted graphs can be formulated as finding an optimum permutation matrix that minimizes a distance measure between both graphs. This problem is known as the weighted graph matching problem (WGMP) that includes the isomorphism problem, which is proved neither to be NP complete nor to have an efficient algorithm [3], [4].

Many approaches to solve the WGMP have been proposed. You [5] and Tsai [1] employed tree search techniques for finding isomorphisms between graphs that include both symbolic and numerical labels. The above methods always give the true optimum matching, but because of their combinatorial nature, they are impractical when analyzing large structures. Kitchen [2], [6] used a relaxation method to solve the matching problem in both qualitative and quantitative cases.

Recently, Umeyama [3] proposed a polynomial time method based on the eigendecomposition of the adjancency matrix of a graph. With this technique, real optimum solutions that minimize the euclidean distance between a pair of graphs are obtained when the graphs are sufficiently close to each other. In [7], an approximate method based on a symmetric polynomial transform (SPT), which is invariant under permutation, was proposed for matching pairs of weighted graphs.

In this paper, the WGMP is formulated as a linear programming (LP) problem, which is solved by using a simplex-based algorithm. It is shown that the complexity of the proposed algorithm is polynomial. The proposed approach showed superiority over the methods in [3] and [7] in matching weighted graphs. The rest of the paper is organized as follows: Section II presents the statement of the problem followed by reduction of the problem to a linear model in Section III. Section IV provides the linear programming formulation of the graph matching problem. Section V presents computational results and comparisons with the methods [3], [7]. Section VI concludes the paper.

Manuscript received December 27, 1990; revised October 4, 1991. Recommended for acceptance by Associate Editor M. Nagao.

The authors are with the Systems Engineering Department, King Fahd University of Petroleum and Minerals, College of Computer Science and Engineering, Dhahran, Saudi Arabia.

IEEE Log Number 9205814.

II. Statement of the Problem

A weighted graph G is an ordered pair (v, x), where v is a set of n vertices in the graph, and x is a weighting function, which gives a real nonnegative value $x(v_i, v_j)$ to each pair of vertices (v_i, v_j). An undirected graph is a graph in which the weights $x(v_i, v_j) = x(v_j, v_i)$. For a weighted directed graph, $x(v_i, v_j) \neq x(v_j, v_i)$. The adjacency matrix A_G of a weighted graph $G = (v, x)$ without self-loops is given by $A_G = \{g_{ij}\}$, where

$$\begin{aligned} g_{ij} &= x(v_i, v_j) \quad i \neq j \\ g_{ij} &= 0 \qquad\quad\ \ i = j \end{aligned} \tag{1}$$

and where $i, j = 1, 2, \cdots, n$, and n is the number of nodes in the graph. The problem of matching two weighted graphs $G = (v, x)$ and $H = (w, y)$ of the same number of vertices n consists of finding a one-to-one correspondence between $v = (v_1, v_2, \cdots, v_n)$ and $w = (w_1, w_2, \cdots, w_n)$, which makes G and H as close as possible with respect to a certain norm. Using an $n \times n$ permutation matrix $P = \{P_{ij}\}$, the graph matching problem can be formulated in L_1 norm as

$$\min_P \|A_G - PA_HP^T\|_1 \tag{2}$$

where $\|.\|_1$ denotes the L_1 norm, that is, if A is an $n \times n$ matrix, then $\|A\|_1 = \sum_i \sum_j |a_{ij}|$ for $i, j = 1, 2, \cdots, n$.

At present, there is no polynonial time algorithm that can directly solve the minimization problem in (2) with 0–1 integral solutions. However, algorithms based on brute-force enemuration such as branch-and-bound and search enumerative techniques give integral solutions, but these algorithms have exponential complexity [11].

III. Reduction to Linear Model

Let R_1 be a residual $n \times n$ matrix such that

$$R_1 = A_G - PA_HP^T \tag{3}$$

where P is an orthogonal matrix having the property $PP^T = P^TP = I$, and I is the identity matrix. Multiplying both sides of (3) by P and substituting P^TP by I, we have

$$R_1P = A_GP - PA_H. \tag{4}$$

Since P is a permutation matrix, the L_1 norm of (4) is

$$\|R_1P\|_1 = \|R_1\|_1 = \|A_GP - PA_H\|_1 \tag{5}$$

which implies that the minimization problem given in (3) is equivalent to

$$\min_P \|R_1\|_1 = \min_P \|A_GP - PA_H\|_1. \tag{6}$$

Let us consider a residual $n \times n$ matrix R such that

$$R = A_GP - PA_H \tag{7}$$

and let the matrices $R = \{r_{ij}\}$ and $P = \{p_{ij}\}$ be partitioned by columns:

$$\begin{aligned} VEC(R) = \{&r_{11}, r_{21}, \cdots, r_{n1}, r_{12}, r_{22}, \cdots, \\ &r_{n2}, \cdots, r_{1n}, r_{2n}, \cdots, r_{nn}\}^T \end{aligned} \tag{8}$$

$$\begin{aligned} VEC(P) = \{&p_{11}, p_{21}, \cdots, p_{n1}, p_{12}, p_{22}, \cdots, \\ &p_{n2}, \cdots, P_{1n}, P_{2n}, \cdots, P_{nn}\}^T. \end{aligned} \tag{9}$$

Then, (7) can be written in the form

$$VEC(R) = A_{GH}\ VEC(P) \tag{10}$$

where A_{GH} is an $n^2 \times n^2$ constant matrix derived from the weights of graphs G and H. It is clear from the above transformation that the problem of minimizing $\|A_G P - P A_H\|_1$ is equivalent to the problem of minimizing $\|A_{GH}\ VEC(P)\|_1$. Thus, the WGMP in (6) becomes

$$\min_P \|VEC(R)\|_1 = \min_P \|A_{GH}\ VEC(P)\|_1. \tag{11}$$

A consequent advantage of (11) is that the WGMP given in (2) is transformed from a nonlinear to a linear optimization problem in L_1 norm. As mentioned in Section II, it is not easy to find direct integer solutions of (11), but a nearly optimum solution between 0 and 1 can be obtained by extending the domain of P to the set of real matrices whose sum of elements in any row or column is 1. The method of solving (11) with the above extension is developed in Section IV.

IV. Linear Programming Formulation

The problem in (11) can be formulated as a linear program by introducing goal variables S_i, T_i. Any value of $VEC(P)$ that solves the linear system $A_{GH}\ VEC(P) = 0$ provides the minimum for problem (12). If such a solution exists, the linear programming solution will obtain it; otherwise, the solution driving (12) as close as possible to zero will be obtained. In this section, the method of goal variables is used to find an approximate real optimal solution for $A_{GH}\ VEC(P) = 0$.

In order to simplify the notation, let $p = VEC(P) = \{p_k\}, k = 1, 2, \cdots, n^2$ denote in order the unknown variables $\{P_{ij}\}, i, j = 1, 2, \cdots, n$, as defined in $VEC(P)$. In the first instance, the problem consists of finding an optimal real basic solution p that minimizes the following

$$\|A_{GH}\ p\|_1 \qquad p \geq 0 \tag{12}$$

where A_{GH} is an $n^2 \times n^2$ constant matrix, and p is an $n^2 \times 1$ unknown vector. Since $A_{GH}\ VEC(P) = 0$ may not have a feasible solution, two sets of real positive goal variables $S = \{S_i\}$ and $T = \{T_i\}, i = 1, 2, \cdots, n^2$ are introduced in (13) such that

$$A_{GH}\ p + S - T = 0. \tag{13}$$

These goal variables can be interpreted as residuals of the equation $A_{GH}p = 0$. Then, the linear minimization problem of $\|A_{GH}p\|_1$ becomes

$$\begin{aligned} \min_{p,S,T} \quad & \sum_{i=1}^{n^2} S_i + T_i \\ s.t. \quad & A_{GH}p + S - T = 0 \\ & p \geq 0, S \geq 0, T \geq 0. \end{aligned} \tag{14}$$

Furthermore, some additional constraints on the solution p should be included in (14). These constraints should reflect the fact that in a permutation matrix, the sum of elements in any row or column is 1. Assume $P = \{P_{ij}\}, i, j = 1, 2, \cdots, n$ is a permutation matrix; then, there are $2n$ linear constraints that are formulated as follows:

$$\begin{aligned} \sum_i P_{ij} &= 1 \quad \text{for } i, j = 1, 2, \cdots, n \\ \sum_j P_{ij} &= 1 \quad \text{for } i, j = 1, 2, \cdots, n. \end{aligned} \tag{15}$$

The above constraints can be written in the matrix form

$$B\ p = e \tag{16}$$

where $B = \{b_{ij}\}$ is a $2n \times n^2$ unimodular matrix defined by

$$\begin{aligned} b_{ij} &= 1 \quad \text{for } i = 1, 2, \cdots, n \text{ and,} \\ & \qquad j = i, i+n, i+2n, \cdots, i+n(n-1), \text{ and} \\ b_{ij} &= 0 \quad \text{otherwise} \end{aligned} \tag{17}$$

p is an $n^2 \times 1$ unknown vector defined in (9), and e is a $2n \times 1$ vector of ones. With the above constraints and since the basic solution p must be positive, the linear problem in (14) can be written as follows

$$\begin{aligned} \min_{p,S,T} \quad & \sum_{m=1}^{n^2} S_m + T_m \\ S.T. \quad & A_{GH}\ p + S - T = 0 \\ & B\ p = e \\ & p \geq 0, S \geq 0, T \geq 0. \end{aligned} \tag{19}$$

Because of the constraint $Bp = e$, the linear problem in (19) always has a basic optimal solution $0 \leq p \leq 1$, and the objective function may have an optimum value zero, which is attained for $S_m = 0$ and $T_m = 0$ for all $m = 1, 2, \cdots, n^2$.

An upper and lower bound for the goal variables S_m and T_m can also be found in order to restrict the feasible region of the above problem to the domain of real solutions X (including S_m and T_m variables) in the range [0, 1].

Assume now that G and H have weights between 0 and 1; then, the linear problem given in (18) has certain properties regarding the bounds of S_m and T_m. Let T_m^* and S_m^* denote the optimum of T_m and S_m, respectively. Then, we have the following properties:

Property 1: At optimality either T_m^* or S_m^* is greater than zero, which can be written as

$$T_m^* > 0 \qquad \text{implies} \qquad S_m^* = 0$$

and

$$S_m^* > 0 \qquad \text{implies} \qquad T_m^* = 0.$$

This is known from the theory of linear programming [9].

Property 2: The absolute difference between goal variables S_m and T_m is given by

$$|S_m - T_m| \leq 1 \qquad m = 1, 2, \cdots, n^2.$$

This can be shown as follows: the mth constraints that have T_m and S_m as goal variables can be written as

$$\sum_{k=1}^{n} g_{ik} P_{kj} - \sum_{l=1}^{n} h_{lj} P_{il} = S_m - T_m$$

where $m = (i-1)n + j$. Since the sum of the elements P_{kj} on the kth row is equal to 1 (P_{ij} are not necessarily 0 or 1) and $0 \leq g_{ik} \leq 1$, we have

$$0 \leq g_{\min} \leq \sum_{k=1}^{n} g_{ik} P_{kj} \leq g_{\max} \leq 1$$

where $g_{\min}$ and $g_{\max}$ are the minimum and the maximum elements in the matrix A_G, respectively. Similarly, let $h_{\min}$ and $h_{\max}$ denote the minimum and maximum elements in the matrix A_H respectively; then, we have

$$0 \leq h_{\min} \leq \sum_{l=1}^{n} h_{lj} P_{il} \leq h_{\max} \leq 1.$$

Since $g_{\min}$ and $g_{\max}$ and $h_{\min}$ and $h_{\max}$ are between 0 and 1, then it follows that

$$|S_m - T_m| \leq 1$$

i.e., the second property is shown.

Theorem 1: At optimality, $0 \leq T_m^* \leq 1$, and $0 \leq S_m^* \leq 1$.

Proof: From Property 2, we have

$$|S_m - T_m| \leq 1$$

and from Property 1, if $T_m^* > 0$, then $S_m^* = 0$. By using Property 2, we get $|0 - T_m^*| \leq 1$, which implies

$$0 \leq T_m^* \leq 1.$$

Similarly, if $S_m^* > 0$, then $T_m^* = 0$ by Property 1. Using Property 2, we get $|S_m^* - 0| \leq 1$, which implies

$$0 \leq S_m^* \leq 1. \qquad \square$$

Theorem 2: The optimal objective function of problem (20) cannot exceed n^2.

Proof: Let Z denote the objective function of (20) and Z^* the optimal value of Z; then, from Property 1 and Theorem 1, we have

$$Z^* = \sum_{i=1}^{n^2} (S_i^* + T_i^*) \leq \sum_{i=1}^{n^2} 1 = n^2.$$

We note that Theorem (2) and its proof are valid for the formulation given in (18), which has an optimal solution for $0 \leq p_{ij} \leq 1$. For the optimization problem given in (11), the matrix P is assumed to be a permutation matrix, and therefore, the elements P_{ij} are 0 or 1. It follows that Theorem (2) is also valid for (5) since $P_{ij} = 0$ or 1 is included in the interval (0, 1).

V. Comparison with Other Methods

The linear programming approach does not require that the eigenvalues of the adjacency matrix of a graph be distinct, which is a requirement in the eigendecomposition approach [3]. We believe that this is a limitation for the approach in [3]. However, to evaluate and compare the performance on graph matching and computation time, the following computer experiments were conducted. The following algorithms were implemented on an AMDHAL 5850 mainframe: a) linear programming approach, b) eigendecomposition method [3], c) the symmetric polynomial transform [7].

Graphs of different sizes with random weights at each arc were generated. Weights ranging from 0–1.0 were assigned to each arc in a graph G. A matching graph H was generated from graph G by adding uniformly distributed noise in the range of $-e$ to $+e$ to each weight in G and then shuffling the order of nodes to produce the matching graph H.

Graphs of sizes ranging from 5 to 10 were produced as input to the algorithm. Noise levels ranging from 0 to 0.20 were generated for each graph H. Fifty pairs of weighted directed and undirected graphs were generated and matched with each other by the above three algorithms.

The criterion value for the correct match for method a) is

$$J(P) = \|A_G - PA_HP^T\|_1 \tag{19}$$

and the average of the expected value $E_a(n, e)$ of the criterion $J(P)$ of method a) is given by

$$E_a(n, e) = en(n-1)/2$$

since the variance of the noise uniformly distributed in the range of $-e$ to $+e$ for the L_1 norm is equal to $e/2$. The criterion value for the correct match for methods b) and c) is

$$J(P) = \|A_G - PA_HP^T\|_2 \tag{20}$$

and the average of the expected value $E_{b,c}(n, e)$ of the criterion $J(P)$ of methods b) and c) is given by

$$E_{b,c}(n, e) = e^2 n(n-1)/3$$

since the variance of the noise uniformly distributed in the range of $-e$ to $+e$ for the L_2 norm is equal to $e^2/3$.

The number of real optimum matchings obtained by each of the above methods was recorded. Means and standard deviations of the criterion value and the number of optimum matching (g) by each method for graphs of size $n = 5$ to 10 and noise levels $e = 0.0$, 0.05, 0.10, 0.15, and 0.20 were recorded. Results for size $n = 10$ are the only ones shown.

Table I shows the results obtained from the above three algorithms for matching weighted directed and undirected graphs. Table II provides CPU computation time for the case of undirected and directed graphs, respectively.

Table I shows the number of the correct matching for graphs of

TABLE I
PERFORMANCE EVALUATION OF THE LP-WEIGHTED GRAPH MATCHING ALGORITHM (Comparison of the performance of the present algorithm a) with an eigendecomposition method b) and the symmetric polynomial transform algorithm c). (Fifty trials per datum were generated.)

Case of Undirected Graphs for n = 10

e	Method (a) Mean (S.D.)	g	Method (b) Mean (S.D.)	g	Method (c) Mean (S.D.)	g	$E_a(n,e)$	$E_{b,c}(n,e)$
0.00	0.00 (0.00)	50	0.00 (0.00)	50	0.00 (0.00)	50	0.000	0.000
0.05	2.27 (0.04)	50	1.51 (6.87)	36	3.59 (12.8)	21	2.250	0.075
0.10	4.42 (0.10)	50	5.78 (17.4)	21	5.35 (20.5)	13	4.500	0.300
0.15	6.63 (0.87)	46	7.13 (26.9)	14	9.22 (25.6)	4	6.750	0.675
0.20	10.6 (14.9)	38	10.7 (28.7)	2	9.30 (27.3)	1	9.000	1.200

Case of directed Graphs for n = 10

e	Method (a) Mean (S.D.)	g	Method (b) Mean (S.D.)	g	Method (c) Mean (S.D.)	g	$E_a(n,e)$	$E_{b,c}(n,e)$
0.00	0.00 (0.00)	50	0.00 (0.00)	50	0.00 (0.00)	50	0.000	0.000
0.05	2.24 (0.01)	50	1.29 (9.01)	42	0.21 (0.48)	48	2.250	0.075
0.10	4.26 (0.08)	50	3.15 (19.8)	33	1.89 (5.44)	34	4.500	0.300
0.15	6.30 (0.16)	50	7.81 (28.7)	13	4.69 (7.59)	19	6,750	0.675
0.20	8.50 (0.24)	50	9.95 (29.8)	10	6.34 (10.3)	10	9.000	1.200

[a] n and e are the size of graphs and noise level, respectively. Mean and (S.D.) give the means and standard deviations of the criterion value J(P), and g is the number of optimum matching. The expected value of the criterion is given by $E_a(n,e)$ for method (a), and by $E_{b,c}(n,e)$ for methods (b) and (c).

TABLE II
CPU COMPUTATION TIME (SECONDS) FOR WEIGHTED UNDIRECTED GRAPH MATCHING

	e = 0.10			e = 0.20		
n	Method (a) Mean (S.D.)	Method (b) Mean (S.D.)	Method (c) Mean (S.D.)	Method (a) Mean (S.D.)	Method (b) Mean (S.D.)	Method (c) Mean (S.D.)
5	0.78 (0.02)	0.05 (0.01)	0.04 (0.08)	0.75 (0.01)	0.05 (0.01)	0.04 (0.06)
6	1.97 (0.12)	0.08 (0.03)	0.07 (0.09)	1.83 (0.10)	0.08 (0.03)	0.07 (0.07)
7	4.45 (0.18)	0.11 (0.08)	0.13 (0.12)	4.17 (0.13)	0.12 (0.10)	0.13 (0.10)
8	9.06 (2.34)	0.16 (0.13)	0.25 (0.14)	8.76 (2.01)	0.16 (0.12)	0.24 (0.14)
9	18.8 (9.40)	0.24 (0.19)	0.49 (0.23)	17.6 (8.23)	0.23 (0.20)	0.50 (0.27)
10	34.1 (16.7)	0.32 (0.25)	1.07 (0.52)	32.1 (14.3)	0.32 (0.31)	1.07 (0.55)

CPU COMPUTATION TIME (MILLISECONDS) FOR WEIGHTED DIRECTED GRAPH MATCHING

	e = 0.10			e = 0.20		
n	Method (a) Mean (S.D.)	Method (b) Mean (S.D.)	Method (c) Mean (S.D.)	Method (a) Mean (S.D.)	Method (b) Mean (S.D.)	Method (c) Mean (S.D.)
5	0.71 (0.02)	0.06 (0.05)	0.05 (0.08)	0.68 (0.02)	0.06 (0.05)	0.05 (0.08)
6	1.73 (0.09)	0.10 (0.07)	0.09 (0.09)	1.67 (0.13)	0.09 (0.07)	0.09 (0.10)
7	3.91 (0.45)	0.14 (0.10)	0.17 (0.28)	3.76 (0.37)	0.15 (0.09)	0.18 (0.24)
8	8.30 (1.91)	0.20 (0.13)	0.37 (0.39)	7.96 (1.78)	0.21 (0.15)	0.37 (0.42)
9	16.7 (7.05)	0.29 (0.19)	0.82 (0.53)	15.5 (6.95)	0.30 (0.20)	0.81 (0.54)
10	30.4 (45.9)	0.38 (0.39)	1.91 (0.62)	28.9 (38.6)	0.40 (0.42)	1.92 (0.73)

[a] n and e are the size of graphs and noise level, respectively. Mean and (S.D.) give the means and standard deviations of the total CPU time.

size $n = 10$ with noise levels ranging from 0 to 0.20. In all cases, the proposed algorithm a) gave better results than the other two methods. Especially when the noise $\epsilon \leq 0.10$, algorithm a) found all correct matchings, whereas other methods start to deteriorate from $\epsilon = 0.05$. In the general method, a) is more robust to changes in size and noise level. For the case of directed graphs, the proposed method identifies all the correct matchings, irrespective of size and error level. The other methods b) and c) improve in identifying the correct matchings, but they are still inferior to the proposed method.

Table II shows the computation times for both undirected and directed graphs of the three methods with respect to size and noise level changes. The computation time of the proposed method is higher than methods b) and c). For a graph of size $n = 10$ with $\epsilon = 0.20$, the CPU time is about 30 s, where the computation times for methods b) and c) are about 0.5 and 2.0 s, respectively.

The complexity of the present algorithm is polynomial, i.e., $O(n^6 L)$, since LP can be solved in $O(m^3 L)$, where m is the number of variables, and L is the size of the LP problem. It is to be noted that the proposed method was implemented using a simplex-based code. This may result in higher computation time, which is within reach, since the Simplex method is an exponential algorithm but has shown acceptable performance in most practical applications, unlike branch-and-bound methods, which provide 100% correct matching with a computation time out of reach.

VI. Summary and Conclusion

A method for matching pairs of weighted directed or undirected graphs has been proposed. The problem of matching two graphs is formulated as a linear programming problem. The resulting linear program is solved using a simplex-based code of IMSL. Results obtained from the proposed method have shown that the obtained solutions are clearly better than the ones obtained by the eigendecomposition method [3] and the symmetric polynomial transform [8]. Due to their poor results, the latter two methods cannot be used for optimum graph matchings when the graph size and error magnitudes increase. The proposed method has almost always given the correct results even when the graph size increases ($n = 10$) and when the error injected is fairly high ($\epsilon = 0.20$). However, computation time of the proposed method is relatively high but within reach. This computation time can be improved by using an interior point method-based algorithm for linear programming, such as the Karamaker algorithm [10]. Interior point methods are polynomial time algorithms for linear programming, unlike the simplex method, which is exponential time. Finally, the advantage of the proposed method is that it is robust to changes in graph sizes and noise levels.

References

[1] W. H. Tsai, and K. S. Fu, "Error-correcting isomorphisms of attributed relation graphs for pattern recognition," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-9, pp. 757–768, Dec. 1979.

[2] L. Kitchen, "Relaxation applied to matching quantitative relational structures," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-10, pp. 96–101, Feb. 1980.

[3] S. Umeyama, "An eigendecomposition approach to weighted graph matching problems," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 10, no. 5, pp. 695–703, Sept. 1988.

[4] M. Garey and D. S. Johnson, *Computer and Interactability: A guide to the Theory of NP-Completeness*. San Francisco, CA: W. H. Freeman, 1984.

[5] M. You and A. K. C. Wong, "An algorithm for graph optimal isomorphism," in *Proc. ICPR*, 1984, pp. 316–319.

[6] L. Kitchen and A. Rosenfeld, "Discrete relaxation for matching relational structures," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-9, pp. 869–864, Dec. 1979.

[7] H. A. Almohamad, "A polynomial transform for matching pairs of weighted graphs," *J. Applied Math. Modeling*, vol. 15, no. 4, Apr. 1991.

[8] M. Bazaraa, J. Jarvis, and H. Sherali, *Linear Programming and Network Flows*. New York: Wiley, 1990.

[9] A. Charnes and W. W. Cooper, *Management models and industrial applications of linear programming, Vol. 1*. New York: Wiley, 1963.

[10] C. C. Gonzaga, "An algorithm for solving linear programming programs in $O(n^3 L)$ operation," in *Progress in Mathematical Programming, Interior Point and Related Methods* (N. Megiddo, Ed.). New York: Springer-Verlag, 1988.

[11] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithm and Complexity*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[12] D. Dubois and H. Prade, *Fuzzy Sets and Systems: Theory and Applications*. New York: Academic, 1980.