Discrete Applied Mathematics 1 (1979) 253–259

AN ALGORITHM FOR ALGEBRAIC ASSIGNMENT PROBLEMS

A.M. FRIEZE

Queen Mary College, London University, London, England

Received 9 January 1979
Revised 3 July 1979

An $O(n^3)$ algorithm is described for solving algebraic assigment problems.

1. Introduction

In a recent series of papers [1]–[6]- Burkard and Zimmermann and others have introduced an algebraic approach for solving certain network flow problems. This provides a unifying framework within which otherwise distinct problems can be tackled by similar methods.

In particular the algebraic assignment problem was introduced in Burkard et al. [2]. In that paper an $O(n^4)$ algorithm was given for its solution. In Burkard and Zimmermann [4] an $O(n^3)$ algorithm was constructed which is a generalisation of an algorithm of Tomizawa [10] for the classic assignment problem.

This paper gives an $O(n^3)$ algorithm which is based on the algorithm of Dinic and Kronrod [7].

2. The problem

The algebraic structure described here was defined in Burkard and Zimmermann [4].

Let S be a non-empty set with a binary relation $+$ and an order relation $\leqslant$ satisfying

(1a) S is totally ordered by $\leqslant$;

(1b) $(S, +)$ is a commutative semi-group;

(1c) S contains an identity e;

(1d) $b \geqslant e$ implies $a \leqslant a + b$ for all a;

(1e) $a < b$ implies there exists $c \geqslant e$ such that $a + c = b$;

(1f) $a + c = b + c$ implies $a = b$ or $a + c = b + c = c$;

where $a, b, c \in S$ throughout (as usual $\geqslant$ denotes the inverse relation of $\leqslant$).

We shall denote the c in (1e) by $b - a$. It is unique because if $b = a + c = a + c'$, then $c = c'$ from (1f). If $b = a$, then we let $b - a = e$.

Definition (general linear assignment problem (GLAP)). Let $(S, +, \leqslant)$ satisfy (1) and let $c_{ij} \in S$ for $i, j \in N = \{1, 2, \ldots, n\}$. Find a permutation ψ of the set N which minimises

$$\sum_{i \in N} c_{i\varphi(i)} = c_{1\varphi(1)} + c_{2\varphi(2)} + \cdots + c_{n\varphi(n)}$$

over all permutations φ of N.

Let MIN denote the minimum of this "sum".

Several examples of GLAP are given in [2]. It suffices here to note that

(2a) if $S = R$ the set of reals and $+$ and $\leqslant$ have their normal interpretation, then GLAP is the classic assignment problem;

(2b) if $S = R \cup \{-\infty\}$, $a + b = \max(a, b)$ and $\leqslant$ is the usual ordering, then GLAP is the bottleneck assignment problem [8].

One can deduce from the axioms (1) that the following decomposition is possible: there exists a totally ordered index set I (whose order relation can be written $\leqslant$ without confusion) and a function $i : S \to I$ satisfying:

(3a) $a < b$ implies $i(a) \leqslant i(b)$;

(3b) $i(a+b) = \max(i(a), i(b))$;

(3c) $i(a) < i(b)$ implies $a + b = b$;

(3d) $a + c = b + c$ and $i(a) = i(b) = i(c)$ implies $a = b$.

This decomposition is described in [9]. For completeness we give a justification for (3) in an Appendix.

For example (2a) $I = \{0\}$ and $i(a) = 0$ for $a \in S$. For example (2b) $I = R \cup \{-\infty\}$ and $i(a) = a$.

For the next section we need the following simple lemmas.

Lemma 1. *Let $a, b, c \in S$ satisfy*

(4a) $a + c \leqslant b + c$;

(4b) $i(c) \leqslant \min(i(a), i(b))$;

then $a \leqslant b$.

Proof. If $i(c) < \min(i(a), i(b))$, then the result follows directly from (3c) and (4a). If $i(a) < i(b)$, the result follows from (3a). If $i(c) = i(a) = i(b)$, then $a > b$ would imply $a + c \geqslant b + c$ and hence $a + c = b + c$ and hence $a = b$. Finally if $i(a) > i(b) = i(c)$ we have $i(a + c) = i(a) > i(b) = i(b + c)$ which implies $a + c > b + c$.

Lemma 2. *$a \leqslant b$ and $c \leqslant d$ implies $a + c \leqslant b + d$.*

Proof. $a + c \leqslant a + (b - a) + c + (d - c) = b + d$.

3. The algorithm

The algorithm described is based on the following simple theorem:

Theorem 1. *Let $u_i, v_j, w_j \in S$ for $i, j \in N$ satisfy*

(5a) $u_i + v_j \leqslant c_{ij} + w_j$;

(5b) $i(w_j) \leqslant i(\min)$.

If permutation ψ satisfies

(6) $u_i + v_{\psi(i)} = c_{i\psi(i)} + w_{\psi(i)}$ *for* $i \in N$,

then ψ solves GLAP.

Proof. Let φ be any permutation of N, then

$$\sum_{i\in N} c_{i\varphi(i)} + \sum_{i\in N} w_{\varphi(i)} \geqslant \sum_{i\in N} u_i + \sum_{i\in N} v_{\varphi(i)} \quad \text{(using Lemma 2)}$$
$$= \sum_{i\in N} u_i + \sum_{i\in N} v_{\psi(i)}$$
$$= \sum_{i\in N} c_{i\psi(i)} + \sum_{i\in N} w_{\psi(i)} = \sum_{i\in N} c_{i\psi(i)} + \sum_{i\in N} w_{\varphi(i)}.$$

Now let $a = \sum_{i\in N} c_{i\psi(i)}$, $b = \sum_{i\in N} c_{i\varphi(i)}$ and $c = \sum_{i\in N} w_{\varphi(i)}$. (5b) implies $i(c) \leqslant \min(i(a), i(b))$. Now apply Lemma 1.

We now describe the algorithm which can be seen to be based on that of Dinic and Kronrod [7].

Step 1:

$$u_i := e, \quad i \in N; \qquad w_j := e, \quad j \in N;$$
$$v_j := c_{p(j)j} = \min(c_{ij} : i \in N), \quad j \in N.$$

Define any $\psi : N \to N \cup \{0\}$ which satisfies

(7a) $\psi(i) = \psi(i') = j \neq 0$ implies $i = i' = p(j)$, and

(7b) $i = p(j)$ implies $\psi(i) \neq 0$.

Note that given ψ satisfying (7a) e.g. $\psi = 0$ it is easy to satisfy (7b). For if $i = p(j)$ and $\psi(i) = 0$ one can put $\psi(i) = j$.

Step 2. Find i_0 such that $\psi(i_0) = 0$ – if no such i_0 exists output $\psi(i)$ for $i = 1, \ldots, n$ as an optimal permutation and terminate[1] –

$$m_j := c_{i_0 j} + w_j, \quad j \in N; \qquad q(j) := i_0, \quad j \in N;$$
$$I := \{i_0\} \quad \text{and} \quad J := \emptyset$$

Step 3: For each $j \notin J$ compute $d_j = m_j - v_j$ (see 10d) and let $d_k = \min(d_j : j \notin J)$.

[1] The optimal objective value can then be computed.

Step 4:

$$u_i := u_i + d_k, \quad i \in I; \qquad w_j := w_j + d_k, \quad j \in J.$$

If $k \notin \psi(N)$ go to Step 6, otherwise

Step 5: For $j \notin J \cup \{k\}$

(a) compute $e_j = d_j - d_k$ and then let $m_j' = v_j + e_j$;

(b) compute $f_j = c_{p(k)j} + w_j - (u_{p(k)} + v_j)$ and then let $m_j'' = v_j + f_j$;

then $m_j := \min(m_j', m_j'')$ and

$$q(j) := p(k) \quad \text{if } m_j' > m_j''.$$

$I := I \cup \{p(k)\}$ and $J := J \cup \{k\}$ go to Step 3

Step 6: Define the bi-partite digraph $G = (I, J \cup \{k\}, E\}$ where

$$E = \{(q(j), j) : j \in J \cup \{k\}\} \cup \{(j, p(j)) : j \in J\}.$$

Construct the unique path $P = (i_0, j_0, \ldots, i_s, j_s = k)$ from i_0 to k using any labelling method. Then $p(j_r) := i_r$ and $\psi(i_r) := j_r$ for $r = 0, 1, \ldots, s$ go to Step 2.

4. Validity of the algorithm

We observe first that (7) holds throughout. We observe also that $u_{i_0} = e$ in Step 2 and that $j \in J \leftrightarrow p(j) \in I$ throughout. This is a consequence of ensuring (7b) initially.

We next show that throughout the algorithm

(8a) $u_i + v_j \leq c_{ij} + w_j, \quad i, j \in N$;

(8b) $u_{p(j)} + v_j = c_{p(j)j} + w_j, \quad j \in N$

and that on each completion of Step 4

(9) $u_{q(k)} + v_k = c_{q(k)k} + w_k$

and that on each completion os Step 5

(10a) $u_i + m_j \leq c_{ij} + w_j, \quad i \in I, j \notin J$;

(10b) $u_{q(j)} + v_j = c_{q(j)j} + w_j, \quad j \in J$;

(10c) $u_{q(j)} + m_j = c_{q(j)j} + w_j, \quad j \notin J$;

(10d) $m_j \geq v_j, \quad j \notin J$.

It is trivially true that (8) holds on completion of Step 1. It is also trivial (given $u_{i_0} = e$) that (10) holds on completion of Step 2.

We now show that these relationships hold after the updates in Step 4, 5. We use to $\hat{}$ to indicate an updated value. (Note that the value of v_j is constant throughout.)

$$i \in I, j \in J, \quad \hat{u}_i + v_j = u_i + d_k + v_j \leq c_{ij} + w_j + d_k = c_{ij} + \hat{w}_j;$$

$$i \in I, j \notin J, \quad \hat{u}_i + v_j = u_i + d_k + v_j \leq u_i + d_j + v_j = u_i + m_j \leq c_{ij} + w_j = c_{ij} + \hat{w}_j;$$

$$i \notin I, j \in N, \quad \hat{u}_i + v_j = u_i + v_j \leq c_{ij} + w_j \leq c_{ij} + \hat{w}_j.$$

Thus (8a) remains true

$$j \in J, \quad \hat{u}_{p(j)} + v_j = u_{p(j)} + d_k + v_j = c_{p(j)j} + w_j + d_k = c_{p(j)j} + \hat{w}_j;$$

$$j \notin J, \quad \hat{u}_{p(j)} + v_j = u_{p(j)} + v_j = c_{p(j)j} + w_j = c_{p(j)j} + \hat{w}_j.$$

Thus (8b) remains true.

$$\hat{u}_{q(k)} + v_k = u_{q(k)} + d_k + v_k = u_{q(k)} + m_k = c_{q(k)k} + w_k = c_{q(k)k} + \hat{w}_k.$$

Thus (9) is true.

$$i \in I, j \notin J, \quad \hat{u}_i + \hat{m}_j \leq u_i + d_k + m'_j = u_i + m_j \leq c_{ij} + w_j = c_{ij} + \hat{w}_j;$$

$$j \notin J, \quad \hat{u}_{p(k)} + \hat{m}_j \leq u_{p(k)} + m''_j = c_{p(k)j} + w_j = c_{p(k)j} + \hat{w}_j.$$

Thus (10a) remains true.

$$j \in J, \quad \hat{u}_{q(j)} + v_j = u_{q(j)} + d_k + v_j = c_{q(j)j} + w_j + d_k = c_{q(j)j} + \hat{w}_j.$$

This together with (9) implies (10b) remains true.

$$j \notin J, \quad \hat{u}_{\hat{q}(j)} + \hat{m}_j = u_{q(j)} + d_k + m'_j = u_{q(j)} + m_j = c_{q(j)j} + w_j = c_{q(j)j} + \hat{w}_j$$

or

$$= u_{p(k)} + m''_j = c_{p(k)j} + w_j = c_{p(k)j} + \hat{w}_j.$$

Thus (10c) remains true. Inequality (10d) remains true because $\hat{m}_j = v_j + \min(e_j, f_j)$.

We next show that path P exists in Step 6. We can show that on completion of any Step 4 a path exists from i_0 to k if G is defined as in Step 6.

On the first execution of Step 4 after a Step 2 we have $P = (i_0, k)$. Assume inductively that paths exist up to a certain execution of Step 4. Now either $q(k) = i_0$ and $P = (i_0, k)$ or $q(k) = \hat{i} \in I$. It follows that there exists $\hat{k} \in J$ with $\hat{i} = p(\hat{k})$. By assumption there is a path $\hat{P}$ from i_0 to $\hat{k}$ and then $P = (\hat{P}, \hat{i}, k)$.

It follows from (9) and (10b) that (8b) continues to hold after ψ and p are changed in Step 6.

The algorithm must terminate as each execution of Step 6 increases the number of indices i such that $\psi(i) \neq 0$ by 1 and furthermore Steps 3–5 can be gone through at most n times before jumping to Step 6.

Step 1 can be completed in $O(n^2)$ time and each of Steps 2–6 can be completed in $O(n)$ time.

There can be no more than n executions of Step 6 and associated with each of them there is 1 execution of Step 2 and no more than n executions of Steps 3–5.

Thus the algorithm terminates in $O(n^3)$ time.

It follows from (7a) and (8) that on termination (5a) and (6) hold.

It remains only to verify (5b). It holds initially as $i(e) \leq i(a)$ for $a \in S$ (see Appendix). So assume it holds prior to execution of Step 4.

Now for $j \notin J$ $m_j = v_j = v_j + d_j$ implies $i(d_j) \leq i(m_j)$ and (10a) implies that $i(m_j) \leq \max(i(c_{ij}), i(w_j))$ for $i \in I$. The induction hypothesis implies $i(w_j) \leq i(\min)$.

Therefore $i(d_k) \leqslant i(\min)$ or $i(d_k) \leqslant \min(i(c_{ij}) : i \in I, j \notin J)$. Assume the latter inequality. Now $|I| = |J| + 1$. It follows that for any permutation ϕ there exists $t \in N$ such that $t \in I$, $\phi(t) \notin J$. Thus $i(d_k) \leqslant i(c_{t\phi(t)})$ and hence $i(d_k) \leqslant i(\sum_{j\in N} c_{jp(j)})$ and so again $i(d_k) \leqslant i(\min)$. Thus

Theorem 2. *The algorithm described above finds an optimal permutation in* $O(n^3)$ *time.*

Appendix

Let the relation ρ on S defined by

$$a\rho b \leftrightarrow a = b \quad \text{or} \quad a + b \notin \{a, b\}.$$

ρ is clearly reflexive and symmetric.

(A1) *ρ is transitive.* Suppose $a\rho b$ and $b\rho c$ and $b \neq a, c$.

$$a + c = a \to a + b + c = a + b \to b + c = b \quad \text{or} \quad a + b = a \quad \text{(contradiction)},$$

$$a + c = c \to c + a + b = c + b \to c + b = c \quad \text{or} \quad a + b = b \quad \text{(contradiction)}.$$

Thus ρ is an equivalence relation.

(A2) *$a < b < c$ and $b \neq e$ and $a\rho c \to a\rho b$.*

$a + b = a \to a < e$ (else $a + b \geqslant b > a$) $\to b = e$ (adding $e - a$ to both sides);

$a + b = b \to a + c = a + b + (c - b) = b + (c - b) = c$ (contradiction).

As usual let $[a]$ denote the equivalence class of a.

(A3) $a + a = a \to [a] = \{a\}$. If $a \neq b$, then $a + b = a + a + b \to a + b = b$ or $a + b = a$.

We note next that $b \leqslant e \to a + b \leqslant a$ as $a = a + b + (e - b) \geqslant a + b$. We note also that $a + b = e \to a \leqslant e$ or $b \leqslant e$ or $b \leqslant e$ as $a, b > e \to a + b \geqslant a > e$.

(A4) *$a \neq b$ and $a\rho b$ and $a + b \neq e$*

or $\qquad \to a\rho(a + b)$

$a = b$ and $a + a \neq a$.

Case 1: $a > e > b \to a > a + b > b \to a\rho(a + b)$ by (A2).

Case 2: $a \geqslant b > e \to a + b > a \to a + a + b > a$. But

$$a + a + b = a + b \to a + b = a \quad \text{or} \quad a + b = b \quad \text{(contradiction)}.$$

Case 3: $e > a \geqslant b$

$$a + a + b = a \to a + b = e;$$

$$a + a + b = a + b \to a = e \quad \text{(contradiction)}.$$

If a or $b = e$ there is nothing to prove.

(A5) $a, b < e \rightarrow a\rho b.$

$$a + b = a \rightarrow b = e.$$

Next let $S_0 = \{a \in S : a \leq e \text{ or } a\rho b \text{ for some } b < e\}$. Define $I = \{S_0\} \cup \{[a] : a \notin S_0\}$ and $i : S \rightarrow I$ by

$$i(a) = S_0 \quad a \in S_0$$
$$= [a] \quad a \notin S_0.$$

The ordering in I is defined by $i_1 < i_2$ if $a \in i_1, b \in i_2 \rightarrow a < b$. This is well-defined by (A2).

We next verify (3).

(3a) This is trivial.

(3b, 3c) Suppose first $i(a) = i(b)$, then $i(a+b) = i(a)$ from (A3) and (A4) and the remark preceding it. Suppose next $i(a) < i(b)$, then $b > e$ and $a + b = a$ or b. If $a + b = a$ and $a < e$, then $b = e$ (contradiction). If $a \geq e$, then $a + b \geq b > a$. Thus $a + b = b$ is the only possibility.

(3d) The possibilities for a, b, c are:

(i) $c = e$: $a = b$ trivially.

(ii) $c \neq e$ and $a \neq c$ and $a\rho c$: $a + c \neq c$ from the definition of ρ and so $a = b$ by (1f).

(iii) $c \neq e$ and $a = c$: $a + c = c$ implies $[a] = \{a\}$ and $a > e$ as $a < e$ implies $a\rho(e - a)$ and $a \neq (e - a)$. Thus $i(b) = i(a)$ implies $b = a$.

References

[1] R.E. Burkard, A general Hungarian method for the algebraic transportation problem, Discrete Math. 22 (1978) 219–232.

[2] R.E. Burkard, W. Hahn and U. Zimmermann, An algebraic approach to assignment problems, Math. Programming 12 (1977) 318–327.

[3] R.E. Burkard, H. Hamacher and U. Zimmermann, The algebraic network flow problem, Report 1976–7, Mathematische Institut der Universitat zu Koln (1976).

[4] R.E. Burkard and U. Zimmermann, Weakly admissible transformations for solving algebraic assignment and transportation problems, Report 1977–8, Mathematische Institut der Universitat zu Koln (1977).

[5] U. Derigs and U. Zimmermann, An augmenting path method for solving linear bottleneck assignment problems, computing 19 (1978) 285–295.

[6] U. Derigs and U. Zimmermann, An augmenting path method for solving time transportation problems, Report 1977–9, Mathematische institut der Universitat zu Koln (1977).

[7] E.A. Dinic and M.A. Kronrod, An algorithm for the solution of the assignment problem, Soviet Math. Dokl. 10 (1969) 1324–1326.

[8] O. Gross, The bottleneck assignment problem: an algorithm, in P. Wolfe, ed., Proceedings Rand Symposium on Mathematical Programmin, Rand Publication R-351 (1960) pp. 87–88.

[9] X. Lugowski, Uber gewisse geordnete Halbmoduln mit negativen Elementen, Publ. Math. Debrecen 11 (1964) 23–31.

[10] N. Tomizawa, On some techniques useful for solution of transportation network problems, networks 1 (1972) 179–194.