Journal of the Operations Research Society of Japan
Vol. 24, No. 2, June 1981

AN ALGORITHM FOR THE HITCHCOCK TRANSPORTATION PROBLEMS WITH QUADRATIC COST FUNCTIONS

Azuma Ohuchi
Hokkaido University

Ikuo Kaji
Hokkaido University

(Received August 21, 1980; Final February 19, 1981)

Abstract In this paper we give an algorithm for the Hitchcock transportation problems (HTPQ). The algorithm described below has been studied in connection with "Network transportation problems with quadratic costs functions (NTQ)" by I. Takahashi and "Algorithms for optimal allocation problems having quadratic objective function (APQ)" by authors. Takahashi proposed a procedure which involves successively minimizing the Lagrangian with respect to each of its dual coordinates. We investigated the fact that the algorithms for APQ can be used for maximizing procedure in NTQ and studied APQ in detail. We consider the application of algorithms proposed by authors to solving HTPQ. In particular we give several examples of relatively large-scale (3000-20000 variables) problems with computational times (1-10 sec. on a HITAC M-180 computer for these large-scale problems).

1. Introduction

Hitchcock transportation problems with quadratic costs (HTPQ) are frequently used to model transportation systems [7], [9] and energy systems [2], [4]. The use of quadratic cost functions has been especially important in these models.

In this paper we give an algorithm for HTPQ. The algorithm described below has been studied in connection with "Network transportation problems with quadratic cost functions (NTQ)" by Takahashi [11] and "Algorithms for optimal allocation problems having quadratic objective function (APQ)" by authors [10]. Takahashi proposed a procedure which involves successively maximizing the Lagrangian with respect to each of its dual coordinates. We investigated the fact that the algorithm for APQ can be used for maximizing procedure in NTQ and studied APQ in detail.

We shall consider here the application of algorithms proposed in [10] to solve HTPQ. Comparing with [11], our treatment is a restriction in the sense that the HTPQ is a special problem of NTQ. However, it is an extention upon which the upper bound constraits on the variables is imposed (this constraint is essential for practical applications) and the efficient systematic algorithm for maximizing the Lagrangian is given (only a graphical algorithm is given in [11]).

Such problems can of course be solved by the general quadratic programming algorithms [6]. But the algorithm proposed here takes advantage of the specially simple nature of the constraints. It is also iterative in nature but whose computational requirements are significantly less than the general quadratic programming algorithms. m-sources and n-destinations problems require repeated solution of $(m+n)$ linear search problems with respect to one variable. This is especially important for large-scale problems where the number of arcs $(m\times n)$ is usually much greater than the number of nodes $(m+n)$.

2. Problem Formulation

Consider the HTPQ as follows.

HTPQ:

(2.1) minimize $f(x)=\sum_{i\in M}\sum_{j\in N} f_{ij}(x_{ij})=\sum_{i\in M}\sum_{j\in N}(a_{ij}+b_{ij}x_{ij})x_{ij}$

(2.2) subject to $\sum_{j\in N} x_{ij}=A_i,\ i\in M,$

(2.3) $\sum_{i\in M} x_{ij}=B_j,\ j\in N,$

(2.4) $x_{ij}\in C_{ij}=[l_{ij},\ d_{ij}]$, all $i\in M,\ j\in N,$

where $M=\{1,2,\ldots,m\}$, $N=\{1,2,\ldots,n\}$ and $0<l_{ij}<d_{ij}$, $b_{ij}>0$ for all $i\in M$, $j\in N$.

Put $C=\prod_{i,j} C_{ij}$.

In the terminology of the electric power dispatch problem [2], [4], the i represent power generating units, the j represent periods, and x_{ij} represents active power generations generated by the i-th unit in the j-th period. The constraint (2.3) expresses the active power balance equation neglecting losses in the j-th period. The constraints (2.2) and (2.4) express the fact that the power generated by the i-th unit over the planning periods must be limited to A_i and each unit output remained within permissible limits, respectively.

We assume that $\sum_{i\in M} A_i = \sum_{j\in N} B_j$ and $m,n \geqq 2$. Assuming $m,n \geqq 2$, we have $mn \geqq m+n$.

3. Method of Solution

Define the Lagrange function and its dual function associated with HTPQ.

$$(3.1)\qquad L(x,\lambda,\mu)=\sum_{i\in M}\sum_{j\in N}\{(a_{ij}+\lambda_i-\mu_j)x_{ij}+b_{ij}x_{ij}^2\}+\sum_{j\in N}B_j\mu_j-\sum_{i\in M}A_i\lambda_i,$$

$$(3.2)\qquad D(\lambda,\mu)=\min_{x\in C} L(x,\lambda,\mu),$$

where λ and μ are the Lagrange multipliers for the constraints (2.2) and (2.3), respectively. Since the Lagrange function $L(x,\lambda,\mu)$ is separable, the $x_{ij}\in C_{ij}$ minimizing the $L(x,\lambda,\mu)$ is given by

$$(3.3)\qquad \langle x_{ij}(\lambda_i,\mu_j)\rangle=\langle(\mu_j-\lambda_i-a_{ij})/2b_{ij}\rangle$$

and the dual function $D(\lambda,\mu)$ is written as

$$(3.4)\qquad D(\lambda,\mu)=\sum_{i\in M}\sum_{j\in N}\{(a_{ij}+\lambda_i-\mu_j)\langle x_{ij}(\lambda_i,\mu_j)\rangle+b_{ij}\langle x_{ij}(\lambda_i,\mu_j)\rangle^2\}$$
$$+\sum_{j\in N}B_j\mu_j-\sum_{i\in M}A_i\lambda_i,$$

where brackets symbol $\langle x_{ij}(\lambda_i,\mu_j)\rangle$ means

$$(3.5)\qquad \langle x_{ij}(\lambda_i,\mu_j)\rangle=\begin{cases} l_{ij}, & \text{if } x_{ij}(\lambda_i,\mu_j)\leqq l_{ij} \\ x_{ij}(\lambda_i,\mu_j), & \text{if } l_{ij}\leqq x_{ij}(\lambda_i,\mu_j)\leqq d_{ij} \\ d_{ij}, & \text{if } d_{ij}\leqq x_{ij}(\lambda_i,\mu_j)\end{cases}$$

Now we can summarize the algorithm as shown in Fig. 1. The proof that this general algorithm converges, i.e., that the sequence (λ^*, μ^*), the optimal dual solution, can be found in [3]. The main part of the algorithm is the coordinatewise maximization process which finds the maximal point of $D(\lambda,\mu)$. In the next section, we will discuss this procedure in detail.

4. Maximization Procedure of $D(\lambda,\mu)$

From the definition of bracket symbol and (3.3), existence of the partial derivatives of $D(\lambda,\mu)$ with respect to λ_i and μ_j are guaranteed. ([5], [11], [12]) Then coordinatewise maximization of $D(\lambda,\mu)$ can be performed by solving the equation $\partial D(\lambda,\mu)/\partial\lambda_i=0$ or $\partial D(\lambda,\mu)/\partial\mu_j=0$, i.e.,

(4.1) $$\partial D(\lambda,\mu)/\partial\lambda_i=\sum_{j\in N}<(\mu_j-\lambda_i-a_{ij})/2b_{ij}>-A_i=0,$$

or

(4.2) $$\partial D(\lambda,\mu)/\partial\mu_j=-\sum_{i\in M}<(\mu_j-\lambda_i-a_{ij})/2b_{ij}>+B_j=0$$

Thus the following equation are obtained on refering to (3.3).

(4.3) $$\sum_{j\in N}<x_{ij}(\lambda_i,\mu_j)>=A_i,$$

or

(4.4) $$\sum_{i\in M}<x_{ij}(\lambda_i,\mu_j)>=B_j.$$

Let $\delta(\lambda_i)$ be the left hand side of (4.3) and $\delta(\mu_j)$ be the left hand side of (4.4). Define the linear function $\delta^O(\lambda_i)$ and $\delta^O(\mu_j)$ by deleting the brackets in the left hand side of (4.3) and (4.4), respectively.

(4.5) $$\delta^O(\lambda_i)=\sum_{j\in N}x_{ij}(\lambda_i,\mu_j),$$

(4.6) $$\delta^O(\mu_j)=\sum_{j\in M}x_{ij}(\lambda_i,\mu_j).$$

Put $\phi(\lambda_i)=\delta^O(\lambda_i)-\delta(\lambda_i)$ and $\phi(\mu_j)=\delta^O(\mu_j)-\delta(\mu_j)$.

We shall consider various properties of $\delta(\lambda_i)$ and $\delta(\mu_j)$ in the next section, i.e., we shall give 4 Lemmas which can be used for constructing efficient algorithms. We shall only show this for the case of $\delta(\mu_j)$, as 4 Lemmas are also hold for $\delta(\lambda_i)$ by only performing variable transformation $\lambda_i=-\mu_j$.

4.1 Properties of $\delta(\mu_j)$

Let $\alpha_i^j=\lambda_i+a_{ij}+2b_{ij}l_{ij}$, $\beta_i^j=\lambda_i+a_{ij}+2b_{ij}d_{ij}$ for all $i\in M$, $\alpha^j=\max_i \alpha_i^j$, $\beta^j=\min_i \beta_i^j$. Define a new sequence $\{\gamma_k^j|\gamma_k^j=\alpha_i^j$ or $\beta_i^j\}$ out of α's and β's such that $\gamma_1^j<\gamma_2^j<\ \dots\ <\gamma_p^j$ $(p\leqq 2m)$. If one or more α_i^j or β_i^j are same value, they are considered as identical. Note that $\alpha_i^j<\beta_i^j$ holds for all $i\in M$ since $0\leqq l_{ij}<d_{ij}$, and $p=2m$ if all α's and β's are different.

Put $\Gamma_k^j=[\gamma_k^j,\ \gamma_{k+1}^j]$, then there are three possible cases in relation with

$[\alpha_i^j, \beta_i^j]$.

$$(4.7)\qquad [\alpha_i^j, \beta_i^j] \cap \Gamma_k^j = \begin{cases} \Gamma_k^j \\ \{\gamma_k^j\} \quad \text{or} \quad \{\gamma_k^{j+1}\} \\ \phi \end{cases}$$

We define the index set Λ_k^j associated with Γ_k^j,

$$(4.8)\qquad \Lambda_k^j=\{i\,|\,[\alpha_i^j, \beta_i^j] \supset \Gamma_k^j\}.$$

For simplicity, we shall drop the superscript j in the following discussion, i.e., $\alpha_i=\alpha_i^j$, $\beta_i=\beta_i^j$, $\alpha=\alpha^j$, $\beta=\beta^j$, $\gamma_k=\gamma_k^j$, $\Gamma_k=\Gamma_k^j$ and $\Lambda_k=\Lambda_k^j$

As it is easily seen that the function $\delta(\mu_j)$ is a non-decreasing piece-wise linear function with the vertices $\{\gamma_k\}$, we write the equation of the linear function on Γ_k as $y_k=h_k\mu_j+c_k$. h_k and c_k are determined by the index sets Λ_k, Λ_{k_α} and Λ_{k_β},

$$(4.9)\qquad h_k=\sum_{i\in\Lambda_k} 1/2b_{ij},\ c_k=-\sum_{i\in\Lambda_k}(\lambda_i+a_{ij})/2b_{ij} + \sum_{i\in\Lambda_{k_\alpha}} l_{ij} + \sum_{i\in\Lambda_{k_\beta}} d_{ij},$$

where $\Lambda_{k_\alpha}=\{j\,|\,\alpha_j\geqq\gamma_{k+1}\}$ and $\Lambda_{k_\beta}=\{j\,|\,\beta_j\leqq\gamma_k\}$. Note that $\Lambda_k\cap\Lambda_{k_\alpha}\cap\Lambda_{k_\beta}=\phi$ and $\Lambda_k\cup\Lambda_{k_\alpha}\cup\Lambda_{k_\beta}=M$.

We can apply the Lemmas in [10] for $\delta(\mu_j)$. The reader can refer to [10] in detail proof.

Lemma 1.

$$\Lambda_{k+1}=\{\Lambda_k\cup I_\alpha\}-I_\beta,\quad k=1,2,\dots,p-1.$$

(4.10) or

$$\Lambda_k=\{\Lambda_{k+1}-I_\alpha\}\cup I_\beta,\quad k=p-1,\dots,2,1,$$

where $I_\alpha=\{i\,|\,\alpha_i=\gamma_{k+1}\}=\Lambda_k^c\cap\Lambda_{k+1}$ and $I_\beta=\{i\,|\,\beta_i=\gamma_{k+1}\}=\Lambda_k\cap\Lambda_{k+1}^c$

Lemma 2.

$$h_0=\max_k h_k.$$

Lemma 3. If $\alpha<\beta$, then there exists a single subinterval $\Gamma_k=[\alpha,\beta]$ such that $\phi(\mu_j)=0$, for all $\mu_j\in\Gamma_k$. Otherwise if $\alpha\geqq\beta$, then there exists a single point $\hat{\mu}_j$ such that $\phi(\hat{\mu}_j)=0$ and $\beta \leq \hat{\mu}_j \leq \alpha$.

Lemma 4. $\delta(\mu_j)$ is a monotone increasing piecewise-linear convex function for $\gamma_1 \leq \mu_j \leq \beta$, and is a monotone increasing piecewise-linear concave function for $\alpha \leq \mu_j \leq \gamma_p$.

4.2 HB algorithm for coordinatewise maximization

Although several algorithms proposed in [10] can be applied to the coordinatewise maximization of $D(\lambda,\mu)$ with respect to λ_i or μ_j, we apply the HB algorithm. Because the computational results in [10] show that the HB algorithm is efficient. The algorithm HB is summarized as follows.

```
begin {algorithm for the equation $\delta(\mu_j)=B_j$}
    obtain a current solution $\mu_j^0$ by using appropriate approximating function
    of $\delta(\mu_j)$. (properties discussed in section 4.1 can be used for this
    purpose);
    determine the subinterval $\Gamma_k$ and corresponding index set $\Lambda_k$ such that
    $\Gamma_k \ni \mu_j^0$;
    solve the equation $h_k \mu_j + c_k = B_j$;
end
```

When the binary search algorithm [1] is used for searching Γ_k which includes μ_j^0, the number of searched subintervals is at most $O(\mathrm{Log}_2\ 2m/2)$ $=O(\mathrm{Log}_2\ m)$, so $O(n\mathrm{Log}_2\ m + m\mathrm{Log}_2\ n)$ for all λ's and μ's in the worst case.

5. Remarks for Implementing the Algorithm

The algorithm in Fig. 1 includes two kinds of iteration processes. One of them (inner loop) is to obtain solution for coordinatewise maximization, i.e., to solve the equation $\delta(\lambda_i)=A_i$ or $\delta(\mu_j)=B_j$.

The other (outer loop) is to iterate the inner loop until satisfying the convergence condition. The number of iterations of the outer loop might strongly depend upon initial solution and convergence condition. We shall here consider about initial solution and convergence condition.

```
begin {algorithm for HTPQ}
  begin {maximizing the $D(z)$ by coordinatewise maximization}
    choose initial $z^0=(z_1^0,\ldots,z_{m+n}^0)$;
    k:=0;
    repeat $k$:=$k$+1
      for $j$:=1 to $m+n$ do
        maximize$_{z_j}$ the $D(z_1^k,\ldots,z_{j-1}^k,z_j,z_{j+1}^{k-1},\ldots,z_{m+n}^{k-1})$. let $z_j^k$ be the solution;
    until satisfy the convergence condition
  end
  substitute the optimal $z^*$ into (3.3) to obtain the optimal x:
end
```

General description of the algorithm for HTPQ, where the notation $z=(z_1,\ldots,z_{m+n})=(\lambda,\mu)=(\lambda_1,\ldots,\lambda_m,\mu_1,\ldots,\mu_n)$ is used.

Fig. 1

5.1 Initial solution

Three alternatives can be chosen for obtaining initial solution. First, arbitrary values can be used for initial solution z^0, for instance, $z^0=(0,\ldots,0)$. Secondly, as seen in Section 4.1, the solution to the following simultaneous linear equations of $m+n$ unknowns will give better solution than arbitrary values.

(5.1) $\delta^0(\lambda_i)=\sum_{j\in N}(\mu_j-\lambda_i-a_{ij})/2b_{ij}=A_i,\ i\in M,$

(5.2) $\delta^0(\mu_j)=\sum_{i\in M}(\mu_j-\lambda_i-a_{ij})/2b_{ij}=B_j,\ j\in N,$

If we write (2.1), (2.2) and (2.3) as matrix form

(5.3) $F(x)=x'Bx + a'x,$

(5.4) $Gx=h$.

x is an $mn \times 1$ (column) vector, B is an $mn \times mn$ positive difinite and diagonal matrix whose non-zero elements are b_{ij}'s, a' is a $1 \times mn$, h is an $(m+n) \times 1$. G is an $(m+n) \times mn$ matrix as follows:

$$(5.5) \quad G=\begin{bmatrix} I_1 & & & & \\ & I_2 & & & \\ & & \cdot & & \\ & & & \cdot & \\ & & & \cdot & \\ & & & & I_m \\ E_1 & E_2 & \cdot & \cdot\ \cdot & E_m \end{bmatrix}$$

where, I_i is an $n \times 1$ row vector having unity as a value for each component, E_i is an $m \times n$ identity matrix.
h is an $m+n$ column vector such that $h'=(A_1, \dots , A_m, B_1, \dots ,B_n)$.

Then (5.1) and (5.2) can be rewrite as follows:

(5.6) $Qz=c$,

where,

(5.7) $Q=GB^{-1}G'$, $c'=a'B^{-1}G' + 2h'$, and $z=(\lambda,\mu)$.

We can easily show that the rank of Q, r(Q), is $m+n-1$. In fact, r($GB^{-1}G'$)$\leqq m+n-1$, since r(G)=r(G')$=m+n-1$ and B is positive definite. Arbitrary minor in Q of order $m+n-1$, $|Q|_{m+n-1}$, is diagonally dominant, i.e., $q_{ii} \geqq \sum_{\substack{j=1 \\ j\neq i}}^{m+n} q_{ij}$ and strict inequality is hold for $1 \leqq i \leqq m$. Applying the Gershgorin's theorem, we obtain $|Q|_{m+n-1}>0$. Hence, r(Q)$=m+n-1$.[8] Furthermore, as Q is a real symmetric matrix, we can apply several efficient algorithms for simultaneous linear equations with positive definite Hermitian coefficient matrices to (5.6), such as Choleski decomposition technique.[8]

Thirdly, a hybrid of the two is considered. That is, set all $\lambda_i=0$ (or $\mu_j=0$), then solve eq. (5.6). The solution can be used for initial solution.

Computational results by these three methods will be shown in the next section.

5.2 Convergence condition

The convergence condition for the algorithm termination is $|D(z^k)-D(z^{k+1})|<\varepsilon$, for small $\varepsilon>0$. From the stand point of numerical analysis, the following condition is recommended.

(5.8) $$|\{D(z^k)-D(z^{k+1})\}/D(z^k)|<\varepsilon.$$

As shown, above, the average number of outer loop iteration is $O(m\text{Log } n + n\text{Log } m)$. If the algorithm need I times outer loop iterations, the total amount of computation is $O\{I(m\text{Log } n + n\log m)\}$. I, however, depends on ε.

6. Computational Results

The algorithm developed above was coded in FORTRAM IV on a HITAC M-180 computer. The algorithm was tested with 144 randomly generated problems. These types of the problems were solved. The parameters l_{ij} and d_{ij} were generated as uniform random numbers distributed over the interval [0,100], and $l_{ij}<d_{ij}$ for all i, j. The parameters a_{ij} and b_{ij} were randomly generated from four uniform distributions: (1) $0\leq a_{ij}\leq 10$, $0<b_{ij}\leq 1$, (2) $0\leq a_{ij}\leq 5$, $0<b_{ij}\leq 2$, (3) $0\leq a_{ij}\leq 2$, $0<b_{ij}\leq 5$, (4) $0\leq a_{ij}\leq 1$, $0<b_{ij}\leq 10$. In case (1) or (2), the objective functions were chosen as to be nearly linear, in case (3) or (4), the non-linear terms tend to dominate.

The demands and supplies were generated as follows:

$$A_i=\sum_j l_{ij} + \sum_j (d_{ij}-l_{ij}) \times h,$$

$$B_j=\sum_i l_{ij} + \sum_i (d_{ij}-l_{ij}) \times h,$$

where h=0.1, 0.2,...,0.9.

Three algorithms, A, B and C, were compared. The A started from the initial solution obtained by setting $\lambda=\mu=0$, the C started from the initial solution obtained by solving the simultaneous linear equations considered in previous section and the B employed a hybrid method of the two.

One hundred and fourty four problems were solved using the random data mentioned previously. The results are shown in Table 1. The following observations were made from Table 1. For the simplicity of explanation, letters A, B and C are used to distinguish the used algorithms. For example, T(A) and I(B) stand for total cpu times by algorithm A and number of outer loop iterations by algorithm B, respectively.

(1) In every case, the solution time was within 12 sec. (T(A)<11 sec., T(B)<7 sec. and T(C)<12 sec.).

Table 1. Computational results

Problem			Algorithm A			Algorithm B**				Algorithm C					
	m x n	case	I*	T*	ST*	I	T	ST	T2*	I	T	ST	T1*	T2	T1/T,(%)
(I)	40 x 80	1	22	2.87	0.29	17	2.28	0.37	0.01	12	1.94	0.18	0.29	0.01	15
		2	22	3.19	0.47	15	2.11	0.32	0.01	13	2.03	0.20	0.29	0.02	14
		3	21	2.83	0.41	15	2.01	0.36	0.01	13	2.01	0.18	0.30	0.02	15
		4	23	3.24	0.37	16	2.14	0.30	0.01	13	1.97	0.19	0.28	0.02	14
(II)	60 x 120	1	14	4.28	0.54	10	2.82	0.45	0.02	8	3.32	0.30	0.97	0.04	29
		2	15	4.45	0.45	10	2.83	0.44	0.02	8	3.37	0.32	0.97	0.05	29
		3	13	4.29	0.46	10	2.72	0.45	0.02	8	3.39	0.32	0.98	0.05	28
		4	15	4.54	0.47	10	2.89	0.50	0.02	8	3.26	0.33	0.96	0.05	29
(III)	80 x 160	1	12	6.62	0.69	9	4.94	0.82	0.03	8	6.75	0.44	2.28	0.10	33
		2	13	7.27	0.86	9	4.90	0.76	0.03	9	6.85	0.42	2.31	0.11	33
		3	15	7.77	0.41	12	6.31	1.00	0.03	8	6.80	0.42	2.32	0.10	34
		4	15	8.14	1.31	10	5.32	0.71	0.03	8	6.46	0.42	2.42	0.16	37
(IV)	100 x 200	1	9	8.47	0.66	7	6.38	0.76	0.05	7	10.36	0.51	4.50	0.20	43
		2	11	9.80	0.99	7	6.23	0.60	0.05	7	11.15	0.63	4.55	0.20	41
		3	12	10.58	1.80	8	6.69	0.96	0.05	7	10.75	0.56	4.52	0.19	42
		4	11	9.61	1.18	7	6.10	0.74	0.05	7	10.59	0.64	4.44	0.19	42

* I = average number of outer loop iterations.

T = total cpu times (sec.).

ST = standard deviation of T.

T1 = computing times for the coefficient matrix Q in eq. (5.6).

T2 = computing times for solving eq. (5.6).

** T1 of the algorithm B was negligeble.

The convergence condition, $|\{D(z^k)-D(z^{k+1})\}/D(z^k)|<\varepsilon$, $\varepsilon=10^{-6}$, was used.

(2) In every problem, there was no much difference of the solution times among cases 1-4. This means that the algorithm did not depend upon a degree of nonlinearity.

(3) The average number of outer loop iterations decreased in the order of A, B, C (i.e., I(A)>I(B)>I(C)).

(4) T(A)>T(C) held for problems (I)-(III), but T(A)<T(C) in problem (IV).

It was also seen that the percentage of computing the coefficient matrix Q in eq. (5.6) (i.e., T1(C)) rapidly increased as (m,n) increased. For example, a large percentage of the solution times (42 %) was consumed for computing the Q in case 4 of problem (IV). This tendency will become more pronounced as (m,n) increases.†

(5) In every case except the case that T(B)>T(C) for problem (I), T(B) was less than T(A) and T(C). The differences, however, were negligeble even when T(B)>T(C). It is to be expected that this result would be general tendency, since (1) solution times of the algorithm A depends strongly on the initial solution which is arbitrarily given, (2) the algorithm C consumes its large percentage of the solution times for computing matrix Q, and (3) the algorithm B improves the inadequate parts of the two.

Considering the above observations, it would appear that the algorithms proposed are efficient and stable. Particularly, the algorithm B would be efficient for large scale problems.

7. Conclusion

An application of Lagrange relaxation method and coordinatewise maximization technique for the HTPQ has been presented. The efficiencies and stabilities in the algorithms described in this paper are derived from the structure of the APQ studied in [10].

The results indicate great promise for the algorithm and commend attempts to apply to other problems (the network transportation problems having quadratic shipping costs, the transportation-production problems having quadratic shipping costs) using this structure.

8. Acknowledgements

The authors wish to thank Mr. Hideaki Arisaka, Mr. Masayuki Fukuhara and Mr. Hiroaki Sawato for their help in executing the numerical computations, and to the referees for their helpful comments and correcting mistakes.

† The amount of computation of Q is $O((m+n)(mn)^2)$ by usual method. [1]

References

[1] Aho, A. V., Hopcroft, J. E. and Ullman, J. D.: *The design and analysis of computer algorithms*. Johnson-Wesley, (1974).

[2] Elgard, O. I.: *Electric energy systems theory*. McGraw-Hill, (1971).

[3] D'Esopo, D. A.: A convex programming procedure. *Naval Research Logistics Quarterly*, Vol.1, No.1 (1959), 33-42.

[4] Fukao, T. and Toyoda, J.: *Application of computer to the electric network system*. Sangyotosho, (1972).

[5] Geoffrion, A. M.: Elements of large-scale mathematical programming. *Management Science*, Vol.16, No.11 (1970), 652-675.

[6] Hadley, G.: *Nonlinear and dynamic programming*. Addison-Wesley, (1964).

[7] Inose, H. and Hamada, T.: *Traffic flow theory and control*. Sangyotosho, (1972).

[8] Hitotsumatsu, S.: *Linear mathematics*. Chikumashobo, (1976).

[9] LeBlanc, L. J.: The conjugate gradient technique for certain Quadratic network problems. *Naval Research Logistics Quarterly*, Vol.6, No.1 (1959), 327-337.

[10] Ohuchi, A. and Kaji, I.: Algorithms for optimal allocation problems having quadratic objective function. *Journal of the Operations Research Society of Japan*, Vol.23, No.1 (1980), 64-80.

[11] Takahashi, I.: A method for solving network transportation problems with quadratic cost functions. *Bulletin of the Institute for Research in Productivity, Waseda University*, Vol.1, No.1 (1970), 25-31.

[12] Takahashi, I.: Variable separation principle for mathematical programming. *Journal of the Operations Research Society of Japan*, Vol.6, No.1 (1964), 82-105.

The referees suggested that the following paper has been published recently;

[13] Helgason, R. et et al.: A polynomially bounded algorithm for a single constrained quadratic program. *Mathematical Programming*, Vol.18, No.3 (1980), 338-343.

Azuma OHUCHI: Department of Electrical Engineering, Faculty of Engineering, Hokkaido University, Kita 13 Jyo, Nishi 8 Chome, Kita-ku, Sapporo, Japan