1995

Linearly-Constrained Entropy Maximization Problem with Quadratic Costs and Its Applications to Transportation Planning Problems

Shu-Cherng Fang, North Carolina State University at Raleigh
Jacob Tsao, University of California - Berkeley

Available at: https://works.bepress.com/jacob_tsao/36/
Many transportation problems can be formulated as a linearly-constrained convex programming problem whose objective function consists of entropy functions and other cost-related terms. In this paper, we propose an unconstrained convex programming dual approach to solving these problems. In particular, we focus on a class of linearly-constrained entropy maximization problem with quadratic cost, study its Lagrangian dual, and provide a globally convergent algorithm with a quadratic rate of convergence. The theory and algorithm can be readily applied to the trip distribution problem with quadratic cost and many other entropy-based formulations, including the conventional trip distribution problem with linear cost, the entropy-based modal split model, and the decomposed problems of the combined problem of trip distribution and assignment. The efficiency and the robustness of this approach are confirmed by our computational experience.

Many transportation problems can be formulated as a linearly-constrained convex programming problem whose objective function consists of entropy functions and other cost-related terms. To study these transportation planning problems, we first consider the following mathematical optimization model:

Program $P_{\mu}$:

\[ \text{Minimize} \quad f_\mu(x) = \mu \sum_{j=1}^{n} x_j \ln x_j + c^T x + \frac{1}{2} x^T D x \]

subject to:
\[ Ax = b, \quad x \geq 0, \quad (1.a) \]

where $\mu > 0$, $x \in \mathbb{R}^n$, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $D$ is an $n \times n$ diagonal matrix with diagonal elements $d_j \geq 0$, for $j = 1, \ldots, n$, $A$ is an $m \times n$ ($m \leq n$) matrix, and $0$ is the $n$-dimensional zero vector.

Note that when $\mu = 1$, $c = 0$, $D = 0$, and $A$ denotes a trip matrix, Program $P_{\mu}$ represents the basic entropy-based trip distribution model; when only $D = 0$, it becomes the trip distribution model with linear cost. Furthermore, when $c = 0$, $D = 0$, and $A$ represents the modal trip matrix and associated cost-weighing constraints, Program $P_{\mu}$ reduces to an entropy-based modal split model. Therefore Program $P_{\mu}$ not only encompasses many existing transportation planning models but also extends them.

Tomlin\cite{Tomlin22} pointed out that the linear cost (e.g., travel time) per unit flow from an origin to a destination in a trip distribution problem is an oversimplification. Assuming a convex quadratic cost that
reflects the cost increase due to congestion in Program $P_{\mu}$ should be a better approximation of the actual cost. The diagonal matrix $D$ in the quadratic term of the objective function accommodates the possibility of difference in quadratic approximations for the $n$ origin/destination pairs. Strongly motivated by the trip distribution problem with quadratic cost, we shall focus on studying the more general setting described by Program $P_{\mu}$.

Although the optimization model $P_{\mu}$ is more realistic and desirable, the task of designing an efficient algorithm is challenging. While relatively little theory has been known about the general model, various algorithms have been proposed for solving an important special case of this model, namely entropy maximization with linear cost subject to linear constraints. Better known algorithms include MART (multiplicative algebraic reconstruction technique), Bregman's method, Jacobi method, Gauss-Seidel method, and Newton's method. None of them possesses the property of being globally convergent with a quadratic rate of convergence. In fact, for some of these methods, even the number of iterations required to reach the optimum could be very sensitive to the individuality and size of the problem. This presents a potential problem for solving real-life large-scale applications.

In this paper, we take a dual approach and design an efficient and robust computational algorithm for solving not only the entropy maximization problem with linear constraints but also the more general model of Program $P_{\mu}$. With this approach, the solution methods for several important classes of models can be treated in a unified manner.

The rest of the paper is organized as follows. In Section 1, we discuss some basic ideas which motivated model $P_{\mu}$, via a specific example of trip distribution problem with quadratic cost. In Section 2, we derive the Lagrangian dual $D_{\mu}$ for Program $P_{\mu}$, which turns out to be a convex optimization problem without any explicit constraints. Moreover, under some typical and mild conditions, it is shown that no duality gap exists between $P_{\mu}$ and $D_{\mu}$. We also provide a simple conversion formula for obtaining a primal optimal solution from a corresponding dual optimal solution. Section 3 briefly introduces the curved search method and customizes it to obtain a quadratically convergent global algorithm for solving Program $D_{\mu}$ and hence Program $P_{\mu}$. In Section 4, the dual approach developed in Section 2 is specialized for the specific problem of trip distribution with quadratic cost. Some computational results on the trip distribution problem are reported in Section 5. Section 6 concludes this paper.

1. A LINEARLY-CONSTRAINED ENTROPY MAXIMIZATION MODEL WITH QUADRATIC COST

For simplicity, we assume that all mathematical programs stated in this section have a unique optimal solution that satisfies the Kuhn-Tucker condition. To motivate the study of model $P_{\mu}$, we consider a trip distribution problem with a quadratic cost structure. Suppose that there are $K$ origins and $L$ destinations. Let $x_{ij}$ denote the number of trips from origin $i$ to destination $j$. The problem can be represented by:

Program Q:

\[
\begin{align*}
\text{Minimize} & \quad \sum_{i=1}^{K} \sum_{j=1}^{L} x_{ij} \ln x_{ij} + \alpha \sum_{i=1}^{K} \sum_{j=1}^{L} c_{ij} x_{ij} \\
& \quad + \frac{\beta}{2} \sum_{i=1}^{K} \sum_{j=1}^{L} d_{ij} x_{ij}^2 \\
\text{subject to:} & \\
\sum_{j=1}^{L} x_{ij} &= O_i, \quad i = 1, 2, \ldots, K, \\
\sum_{i=1}^{K} x_{ij} &= D_j, \quad j = 1, 2, \ldots, L, \\
x_{ij} &\geq 0, \quad i = 1, 2, \ldots, K, \\
\text{and} & \quad j = 1, 2, \ldots, L.
\end{align*}
\]

Note that, without the quadratic cost terms, Program Q is equivalent to the well-known gravity model. In this special case, the optimal solution has the following form:

\[
x_{ij} = cr_i e^{-\alpha c_{ij} s_j},
\]

where $r_i = e^{\lambda_i}$, $s_j = e^{\mu_j}$, $c = e^{-1}$, and $\lambda_i$ and $\mu_j$ are the Lagrange multipliers associated with the constraints. The exponential term is a decreasing function of $c_{ij}$, which may represent distance, cost or some notion of generalized cost. This exponential term is referred to as the "exponential deterrence" in the literature, which is different from the "quadratic deterrence" in the Newtonian gravity law.

With the quadratic terms, the model is no longer a classic gravity model. Instead, its optimal solution, as can be seen later, satisfies the following equation:

\[
x_{ij} = cr_i e^{-\alpha c_{ij} - \beta d_{ij} x_{ij}^2 s_j}.
\]
This differs from Equation (5) of the gravity model in that \( x_{ij} \) appears in the exponent in the right-hand-side expression and the optimal solution is no longer explicitly solvable as a function of the program parameters. When all \( \beta d_{ij} \)’s are very small, Program Q produces approximately the gravity model. However, the quadratic terms have the following “self-deterrence” or “self-regulation” effect on the optimal solution. By increasing the value of one particular \( \beta d_{ij} \) but keeping all other parameters the same, the corresponding optimal trip number \( x_{ij} \) decreases. Moreover, for any particular positive value of \( \beta d_{ij} \), the larger the optimal \( x_{ij} \) of the gravity model is, the larger the decrease to the optimal \( x_{ij} \) of the new model is. This characteristic can be used to model deterrence due to congestion in the stage of trip distribution. Moreover, solving the combined trip distribution and assignment problem often involves a problem decomposition where the trip distribution is a component which is required to be solved repeatedly. This self-regulation feature of Program Q could benefit the convergence rate due to its ability to reflect, at the stage of distribution, the congestion cost obtained in the assignment stage. This possibility is being studied by the authors. For ease of discussion, we will refer to this new model as the “self-deterrent gravity model” in the rest of this paper. We keep the notion of gravity in the name because the optimal solution \( x_{ij} \) is still a decreasing function of \( c_{ij} \).

We now offer an alternative motivation for studying Program Q. It is well-known that the gravity model can be derived via several other approaches. In particular, the optimal solution of the following mathematical program is of the form:

Program S: \[
\begin{align*}
\text{Minimize} & \quad \sum_{i=1}^{K} \sum_{j=1}^{L} x_{ij} \ln x_{ij} \\
\text{subject to:} & \quad \sum_{j=1}^{L} x_{ij} = O_i, \quad i = 1, 2, \ldots, K, \\
& \quad \sum_{i=1}^{K} x_{ij} = D_j, \quad j = 1, 2, \ldots, L, \\
& \quad \sum_{i=1}^{K} \sum_{j=1}^{L} c_{ij} x_{ij} = C, \\
& \quad \sum_{i=1}^{K} \sum_{j=1}^{L} d_{ij} x_{ij}^2 = D, \\
& \quad x_{ij} \geq 0, \quad i = 1, 2, \ldots, K, \\
& \quad j = 1, 2, \ldots, L.
\end{align*}
\]

It is easy to verify that when the quadratic constraint in Program S is replaced by a slightly generalized constraint

\[
\frac{1}{2} \sum_{i=1}^{K} \sum_{j=1}^{L} d_{ij} x_{ij}^2 + \sum_{i=1}^{K} \sum_{j=1}^{L} f_{ij} x_{ij} = D,
\]

the self-deterrent gravity model has the following form:

\[
x_{ij} = cr_i e^{-(\alpha x_{ij} + \beta f_{ij}) - \beta d_{ij} x_{ij}}.
\]
from the target values \( p_{ij} \), i.e.,

\[
\sum_{i=1}^{K} \sum_{j=1}^{L} w_{ij}(x_{ij} - p_{ij})^2 = T,
\]

(9a)

or, equivalently,

\[
\frac{1}{2} \sum_{i=1}^{K} \sum_{j=1}^{L} d_{ij}x_{ij}^2 + \sum_{i=1}^{K} \sum_{j=1}^{L} f_{ij}x_{ij} = D,
\]

(9b)

where \( d_{ij} = 2w_{ij}, f_{ij} = -2w_{ij}p_{ij} \) and \( D = T - \sum_{i=1}^{K} \sum_{j=1}^{L} w_{ij}p_{ij}, \sum_{i=1}^{K} \sum_{j=1}^{L} w_{ij}(x_{ij} - p_{ij})^2 \) has been used as a measure for the "distance" between \( x_{ij} \) and \( p_{ij} \) [e.g., 4].

Gravity models with additional constraints other than the production, attraction, and cost have been proposed and studied [10]. So far all those additional constraints have been considered in linear form. The proposed quadratic cost term (or the quadratic constraint) provides a new dimension of possible models for transportation planning. The additional parameter \( \beta \) of the self-determining gravity model provides one more degree of freedom in model identification and calibration. In addition, as will be shown later in this paper, the new model can be solved efficiently, in both theory and practice.

### 2. AN UNCONSTRAINED DUAL APPROACH

We first provide the condition under which Program \( P_\mu \) achieves a finite minimum at a unique point \( x^*(\mu) \in \mathbb{R}^n \) for each \( \mu > 0 \).

**Lemma 1.** If Program \( P_\mu \) has a feasible solution, then it achieves a finite minimum at a unique point \( x^*(\mu) \in \mathbb{R}^n \) for each \( \mu > 0 \).

**Proof.** With the convention of \( 0ln0 = 0 \), the objective function \( f_\mu(x) \) is continuous on its entire domain \( \{x| x \geq 0\} \). For each \( j = 1, \ldots, n \), if \( x_j = 0 \), \( f_{ij}(x_j) = \mu x_j + c_j x_j + \frac{1}{2}d_j x_j^2 = 0 \). It is obvious that \( f_{\mu_{ij}}(x_j) \rightarrow -\infty \) as \( x_j \rightarrow \infty \). Therefore, \( f_{\mu}(x) \rightarrow \infty \) as \( \|x\| \rightarrow \infty \) in the effective domain \( \{x| x \geq 0\} \) of \( f_{\mu} \) and hence all level sets \( \{x| f_{\mu}(x) \leq \alpha\}, \alpha \in \mathbb{R}, \) of \( f_{\mu} \) are bounded. The continuity of the objective function implies that these level sets are also closed. Therefore, all level sets are compact. Clearly, the intersection of any of these level sets and the feasible region \( \Omega = \{x| Ax = b, x \geq 0\} \) is also compact. By the assumed feasibility of \( P_\mu \), there exists one level set \( L_{\mu}^0 \) such that \( L_{\mu}^0 \cap \Omega \) is non-empty. Since this intersection is compact and \( f_{\mu}(x) \) is continuous on it, by the well-known fact that any continuous function on a compact set always attains its optimum (Theorem of Weierstrass), Program \( P_\mu \) attains its finite minimum. Finally, the fact that \( f_{\mu_{ij}}(x_j) \) is strictly convex over the domain \([0, \infty)\) for each \( j \), \( f_{\mu} \) is strictly convex. This and the convexity of \( \Omega \) imply that \( P_\mu \) achieves the finite minimum at a unique point \( x^*(\mu) \geq 0 \), for each \( \mu > 0 \). \( \Box \)

Like all interior-point methods, we assume throughout this paper that Program \( P_\mu \) has an interior feasible solution \( x > 0 \). We now derive the Lagrangian dual \( D_\mu \) of \( P_\mu \). By defining the Lagrangian

\[
L(x, w) = \mu \sum_{j=1}^{n} x_j lnx_j + \sum_{j=1}^{n} c_j x_j + \frac{1}{2} \sum_{j=1}^{n} d_j x_j^2 - \sum_{i=1}^{m} w_i \left( \sum_{j=1}^{n} a_{ij} x_j - b_i \right),
\]

where \( w_i \in \mathbb{R}, i = 1, 2, \ldots, m \), the Lagrangian dual program becomes

Program \( D_\mu^1: \quad \max \min L(x, w). \)

Setting the partial derivative of \( L(x, w) \) with respect to \( x_j \) to zero gives us

\[
\mu + \mu lnx_j + c_j + d_j x_j - \sum_{i=1}^{m} a_{ij} w_i = 0. \quad (10)
\]

**Lemma 2.** Equation (10) defines an implicit function \( x_j = h_j(w) > 0 \) for all \( w \in \mathbb{R}^m \). Moreover, \( h_j \) is continuously differentiable.

**Proof.** For \( x_j > 0 \) and \( w \in \mathbb{R}^m \), define

\[
F_j(x_j, w) = d_j x_j + \mu lnx_j - \sum_{i=1}^{m} a_{ij} w_i + c_j + \mu.
\]

(11)

Accordingly, \( F_j \) is continuously differentiable over its effective domain. Since, in addition, \( \partial F_j/\partial x_j \neq 0 \) throughout its domain, by the Implicit Function Theorem [1, p. 147], Equation (11) indeed defines an implicit function \( x_j = h_j(w) \) for all \( w \in \mathbb{R}^m \). Moreover, by the same theorem, the implicit function is continuously differentiable over its entire domain. \( \Box \)

The strict convexity of \( f_{\mu}(x) \) and the linearity of the equality constraints (1a) of Program \( P_\mu \) imply that, given any \( w \in \mathbb{R}^m \), \( L(x, w) \) is strictly convex in \( x \). Therefore, \( h_j(w) \geq 0 \) indeed defines the minimal solution of \( \min_{w \geq 0} L(x, w) \) for each \( w \). Consequently, the dual Program \( D_\mu^1 \) is equivalent to

Program \( D_\mu^2: \quad \max \min L(x, w), \quad \text{where} \ x_j = h_j(w). \)
Multiplying both sides of Equation (10) by \( x_j \) and then summing both sides over \( j \) give
\[
\mu \sum_{j=1}^{n} x_j \ln x_j + \sum_{j=1}^{n} c_j x_j + \frac{1}{2} \sum_{j=1}^{n} d_j x_j^2
- \sum_{i=1}^{m} \sum_{j=1}^{n} w_i a_{ij} x_j = \sum_{j=1}^{n} d_j x_j^2 - \mu \sum_{j=1}^{n} x_j.
\]

Therefore, Program \( D_n^2 \) is equivalent to

Program \( D_\mu \):
\[
\max_{w \in \mathbb{R}^m} d_\mu(w)
= -\frac{1}{2} \sum_{j=1}^{n} d_j h_j(w) - \mu \sum_{j=1}^{n} h_j(w) + \sum_{i=1}^{m} b_i w_i,
\]

where, for any \( w \in \mathbb{R}^m \), \( h_j(w) > 0 \) is defined as the unique solution of the following equation:

\[
d_j h_j(w) + \mu \ln h_j(w) = \sum_{i=1}^{m} a_{ij} w_i - c_j - \mu. \tag{12}
\]

Note that, treating \( h_j(w) \) as a variable, Equation (12) is a single-variable equation which could be solved by a straightforward one-dimensional search. Also note that, in the absence of the quadratic term in the primal objective function, \( h_j(w) \) can be solved explicitly and be expressed as a gravity model in the form of Equation (5).

We devote the rest of this section to establishing duality theorems.

**Theorem 1. (Weak Duality Theorem)** Suppose Program \( P_\mu \) has a feasible solution. Then, \( \min(P_\mu) \geq \sup(D_\mu) \).

**Proof.** This theorem follows immediately from the Lagrangian duality theory. \( \square \)

**Theorem 2.** Given that \( w^* \in \mathbb{R}^m \) and \( x^* \in \mathbb{R}^n \) such that \( Ax^* = b \) and \( x^* > 0 \), if
\[
d_j x_j^* + \mu \ln x_j^* = \sum_{i=1}^{m} a_{ij} w_i^* - c_j - \mu, \tag{12'}
\]

then \( x^* \) is an optimal solution to Program \( P_\mu \) and \( w^* \) is an optimal solution to Program \( D_\mu \). Moreover, \( \min(P_\mu) = \max(D_\mu) \).

**Proof.** Multiplying both sides of Equation (12') by \( x_j^* \) and then summing over \( j \) establish the equality of the two objective function values. Given the feasibility of \( w^* \) and \( x^* \), this theorem follows from Theorem 1. \( \square \)

**Lemma 3.** The dual objective function \( d_\mu(w) \) is continuously twice differentiable.

**Proof.** By Lemma 2, \( h_j(w) \) is continuously differentiable. Therefore, \( d_j(w) \) is also continuously differentiable and the \( k \)-th element of its gradient vector is given by

\[
\frac{\partial d_j(w)}{\partial w_k} = -\frac{1}{2} \sum_{j=1}^{n} 2d_j h_j(w) \frac{\partial h_j(w)}{\partial w_k} - \mu \sum_{j=1}^{n} \frac{\partial d_j(h_j(w))}{\partial w_k} + b_k,
\]

where \( \partial h_j(w)/\partial w_k \) can be obtained as follows. First denote \( h_j(w) \) by \( x_j \) and define
\[
F_j(x_j, w) = d_j x_j + \mu \ln x_j
- \sum_{i=1}^{m} a_{ij} w_i + c_j + \mu. \tag{14}
\]

Then, by Equation (12) and the chain rule, we have
\[
\frac{\partial F_j}{\partial x_j} + \frac{\partial F_j}{\partial w_k} = 0, \tag{15}
\]
or equivalently,
\[
\left( d_j + \frac{\mu}{x_j} \right) \frac{\partial h_j(w)}{\partial w_k} = a_{kj} = 0. \tag{16}
\]

In other words,
\[
\frac{\partial h_j(w)}{\partial w_k} = \frac{a_{kj}}{d_j + \frac{\mu}{x_j}} = \frac{a_{kj} h_j(w)}{d_j h_j(w) + \mu}. \tag{17}
\]

Plugging (17) into equation (13), we finally have

\[
\frac{\partial d_\mu(w)}{\partial w_k} = -\frac{1}{2} \sum_{j=1}^{n} 2d_j h_j(w) \frac{a_{kj} h_j(w)}{d_j h_j(w) + \mu}
- \mu \sum_{j=1}^{n} \frac{a_{kj} h_j(w)}{d_j h_j(w) + \mu} + b_k
= -\sum_{j=1}^{n} a_{kj} \left( \frac{1}{d_j h_j(w) + \mu} \right) \times [ h_j(w)(d_j h_j(w) + \mu) ] + b_k
= -\sum_{j=1}^{n} a_{kj} h_j(w) + b_k.
\]

Since \( h_j(w) \) is continuously differentiable, \( d_\mu(w) \) is continuously twice differentiable. \( \square \)

**Theorem 3.** Program \( D_\mu \) has a concave objective function \( d_\mu(w) \). If the constraint matrix \( A \) in Program \( P_\mu \) has full row-rank, then \( d_\mu(w) \) is strictly concave.
Proof. By Lemma 3, $d_\mu(w)$ is continuously twice differentiable. The $(k_1, k_2)$-th element of the Hessian Matrix is simply

$$\frac{\partial^2 d_\mu(w)}{\partial w_{k_1} \partial w_{k_2}} = \frac{\partial}{\partial w_{k_2}} \left[ -\sum_{j=1}^{n} a_{k_1 j} h_j(w) + b_{k_1} \right]$$

$$= \sum_{j=1}^{n} -a_{k_1 j} \left[ \frac{\partial}{\partial w_{k_2}} h_j(w) \right]$$

$$= \sum_{j=1}^{n} -a_{k_2 j} h_j(w) + a_{k_1 j} h_j(w) + \mu$$

$$= \sum_{j=1}^{n} \left( -\frac{h_j(w)}{d_j h_j(w) + \mu} \right) a_{k_1 j} a_{k_2 j}.$$  

Since $h_j(w) > 0$, $d_j > 0$, and $\mu > 0$, we have $-h_j(w)/(d_j h_j(w) + \mu) < 0$. Consequently, the Hessian matrix can be written as $AD_\mu wA^T$, where $D_\mu(w)$ is an $n \times n$ diagonal matrix with negative diagonal elements $r_j(w) = -h_j(w)/(d_j h_j(w) + \mu)$. (Since $\mu > 0$, $d_j > 0$ and $h_j(w) > 0$, $r_j(w)$ is well defined for all $w \in \mathbb{R}^m$.) Therefore, $d_\mu(w)$ is concave. (In fact, concavity follows immediately from the Lagrangian duality theory.) By matrix theory, the Hessian matrix must be nonsingular and negative definite as long as $A$ has full row-rank. Therefore, $d_\mu(w)$ is strictly concave if $A$ has full row-rank. $\square$

**Theorem 4.** Given $\mu > 0$, if Program $P_\mu$ has an interior feasible solution then Program $D_\mu$ attains a finite minimum and $\min(P_\mu) = \max(d_\mu)$. If, in addition, its constraint matrix $A$ has full row-rank, then Program $D_\mu$ has a unique optimal solution $w^*(\mu) \in \mathbb{R}^m$. In either case, formula (12') provides a dual-to-primal conversion that defines the optimal solution $x^*(\mu)$ of Program $P_\mu$.

**Proof.** By Lemma 1, Program $P_\mu$ attains its minimum at a unique optimal solution. By the Interior-Point Assumption and the Lagrangian duality theory [e.g., 20, p. 154], there is no duality gap between Programs $P_\mu$ and $D_\mu$ and there exists a $w$ at which Program $D_\mu$ attains its finite maximum. Since $d_\mu(w)$ is differentiable over $\mathbb{R}^m$, the first order optimality conditions hold at any optimal solution $w^*(\mu)$. By setting $Vd_\mu(w^*(\mu)) = 0$, the conditions are

$$\sum_{j=1}^{n} a_{k_1 j} h_j(w^*(\mu)) = b_k,$$

for $k = 1, 2, \ldots, m$.  

By definition $x^*(\mu) = h_1(w^*(\mu)) > 0$ according to (12'), Equation (20) becomes $Ax^*(\mu) = b$. The optimality of $x^*_j(\mu)$ follows from Theorem 2. The matrix $A$ having full row-rank implies the strict concavity of $d_\mu$ and hence the uniqueness of the optimal solution $w^*(\mu)$. $\square$

Based on this strong duality theorem, we can solve the constrained Program $P_\mu$ by first solving the unconstrained Program $D_\mu$ and then obtaining a primal optimal solution via the conversion equation. This unique setting allows us to exploit various unconstrained convex optimization methods for solving the general model. Different methods of course lead to different performance. Later in this paper, we will customize the “curved-search algorithm”,[21] a global algorithm with a quadratic rate of convergence, for solving the Lagrangian dual of the trip distribution problem with quadratic cost, which is a special case of Program $P_\mu$.

### 3. A QUADRATICALLY CONVERGENT GLOBAL ALGORITHM

In this section, we outline a computational procedure for solving $P_\mu$, briefly describe the curved-search method, and prove the convergence properties.

The computational procedure is as follows:

**Step 1:** Select an arbitrary initial dual solution $w_0$.

**Step 2:** Determine $h_j(w)$ according to equation (12) and use unconstrained convex optimization techniques to find an optimal solution $w^*(\mu)$ of Program $D_\mu$.

**Step 3:** Compute the optimal solution $x^*(\mu)$ of Program $P_\mu$ according to equation (12').

It is clear that Step 2 accounts for the major computational effort. In this paper, we report our computational experience with the “curved-search method.”[22] Instead of solving Program $D_\mu$ directly, we solve the following equivalent convex minimization problem:

**Program $D'_\mu$:** $\min_{w \in \mathbb{R}^m} -d_\mu(w)$.

Most classical iterative methods improve a current solution by moving along a straight line. The basic idea of the curved-search method is to improve a current solution by moving along a quadratic curve which is determined by minimizing a certain model of the objective function subject to suitable constraints. BEN-TAL ET AL.[23] showed that the quadratic curve used in their method turns out to be a nonlinear combination of the “signed” Newton direction and the steepest descent direction.

More precisely, for an unconstrained convex minimization problem with a twice continuously differentiable objective function $f$, their curved-search
method moves from one solution $w_k$ to the next solution $w_{k+1}$ along the quadratic curve (in variable $t$)

$$q_k(t) = w_k + t d_k + \frac{1}{2} t^2 z_k \quad (\text{for } t \geq 0),$$

where

$$d_k = -\beta_k \frac{|\nabla f(w_k)|^2}{\nabla f^2(w_k) \left[ \nabla^2 f(w_k) \right]^{-1} \nabla f(w_k)} \times \left[ \nabla^2 f(w_k) \right]^{-1} \nabla f(w_k) \quad \text{and} \quad z_k = -\alpha_k \nabla f(w_k) \nabla f(w_k).$$

By searching for an appropriate step-length $t_k$ such that

$$t_k \in \arg \min_{t > 0} f\left( w_k + t d_k + \frac{1}{2} t^2 z_k \right), \quad (21)$$

a new solution $w_{k+1}$ is defined by

$$w_{k+1} = w_k + t_k d_k + \frac{1}{2} t_k^2 z_k. \quad (22)$$

Given a small $\varepsilon > 0$, the optimality is considered reached if $|\nabla f(w_{k+1})| \leq \varepsilon$. Notice that $\alpha_k$ and $\beta_k$ are adjustable positive parameters associated with the constraints with which the above-mentioned model of the objective function is minimized. Therefore, Equations (21) and (22) actually define a family of curved-search algorithms. The performance of this family of algorithms can be fine-tuned by choosing appropriate values of $\alpha_k$ and $\beta_k$. Also notice that when $\alpha_k = 0$, the curved-search algorithm becomes the (signed) Newton method. When $\beta_k = 0$, it becomes the steepest descent algorithm.

Note that, in applying this curved search algorithm to solving Program $D_\mu$, the presence of the implicit functions $h_j$, $j = 1, 2, \ldots, n$, actually does not require any additional care except the need for a simple line search routine to solve for $x_j = h_j(w)$. Since the dual objective function, the gradient vector, and the Hessian matrix can all be expressed in terms of $h_j$’s, calling the simple line search routine prior to their calculation suffices.

We now establish the property of global convergence and the quadratic rate of local convergence.

**Theorem 5.** Suppose that the constraint matrix $A$ has full row-rank and Program $P_\mu$ is strictly feasible. Then, the curved-search algorithm either stops after finitely many steps or generates an infinite sequence $(w_k \in R^n \mid k \in N)$ such that

(i) $d_\mu(w_{k+1}) > d_\mu(w_k)$ for all $k$;

(ii) $(w_k \in R^n \mid k \in N)$ has at least a cluster point $w^c$ in the level set $L_0 = \{w \in R^n \mid d_\mu(w) \geq d_\mu(w_0)\}$;

(iii) for each such cluster point $w^c$, $|\nabla d_\mu(w^c)| \leq \varepsilon$.

**Proof.** Lemma 3 establishes that $d_\mu$ is twice continuously differentiable. If we can further show that the level set $L_0$ is compact, then Theorem 5 is a direct consequence of Theorem 3.1 of [2]. As shown in the proof for Theorem 4, the level set of the dual objective associated with the maximum contains only the optimal solution $w^*(\mu)$. Since $-d_\mu$ is a convex function, Corollary 8.7.1 of [19] directly implies that all level sets are bounded. The continuity of $d_\mu$ guarantees the closedness of the level sets. Therefore, all level sets, including $L_0$, are compact.

Note that, under the assumptions of Theorem 5, since $D_\mu$ has a unique optimal solution $w^*(\mu)$ satisfying the first order optimal condition, the sequence generated by the algorithm, with $\varepsilon = 0$, converges to the unique cluster point $w^c = w^*(\mu)$. Moreover, this infinite sequence has a quadratic rate of convergence.

**Theorem 6.** Under the assumptions of Theorem 5, suppose that the sequence $(w_k \in R^n \mid k \in N)$ generated by the curved-search algorithm converges to $x^*(\mu)$. The rate of convergence is quadratic.

**Proof.** We establish the quadratic rate of convergence by showing that the following four conditions of Theorem 4.4 of [2] are satisfied: (i) $(w_k \in R^n \mid k \in N)$ converges to $w^*(\mu)$, (ii) $\nabla d_\mu(w^*(\mu)) = 0$, (iii) $\nabla^2 d_\mu(w^*(\mu))$ is negative definite and (iv) $H(\cdot) = \nabla^2 d_\mu(\cdot)$ is locally Lipschitz-continuous in a neighborhood of $w^*$.

The first three conditions have already been established. To prove condition (iv), recall that $H = AD_\mu(w)A^T$, where $D_\mu(w)$ is a diagonal matrix with $r_j(w)$ as its $j$-th diagonal element. Since $d_h_j(w) + \mu > 0$ and $h_j(w)$ is differentiable, $r_j(w)$, $j = 1, 2, \ldots, n$, is also differentiable. Therefore, every element of $H$ is a finite combination of differentiable functions. Since differentiability implies Lipschitz continuity, condition (iv) is also satisfied and the proof is complete.

**4. THE TRIP DISTRIBUTION PROBLEM WITH QUADRATIC COST**

In this section, we apply the theory developed in Section 2 to the specific problem of trip distribution with quadratic cost defined in Section 1, i.e., Program Q. To put the problem in the form of Program
Pα defined in Introduction, rewrite Program Q as follows.

Program Pα:

Minimize \( f_\alpha(x) \)

subject to:

1. \( \sum_{i=1}^{K} \sum_{j=1}^{L} h_i \cdot x_{ij} + \sum_{i=1}^{K} \sum_{j=1}^{L} c_{ij} x_{ij} \)

2. \( x_{ij} \geq 0, \quad i = 1, 2, \ldots, K, \quad j = 1, 2, \ldots, L. \) (25)

Note that \( \sum_{i=1}^{K} O_i = \sum_{j=1}^{L} D_j \) and one of the \( K + L \) equality constraints is redundant. Therefore, for obtaining a full-rank constraint matrix, we delete the last equality constraint. To make the format of Program Pα consistent with that of Program Pα', we relabel the variables from 1 through \( K \times L \) row-wise, i.e.,

\[ x_k = x_{(k/L)+1,(k-1)\times L+1}, \quad k = 1, 2, \ldots, K \times L, \] (26)

where \( \lfloor \cdot \rfloor \) is the integer truncation symbol. Moreover, the constraint matrix \( A \) can be partitioned as:

\[ A = \begin{bmatrix} A_1^1 & A_2^1 \\ A_1^2 & A_2^2 \end{bmatrix}, \] (27)

where \( A_1^1 = [a_{ij}^1], A_2^1 = [a_{ij}^2], \) and

\[ a_{ij}^1 = 1, \quad \text{if} \ (i-1) \times L < j < i \times L, \] (28)

0 otherwise;

\[ a_{ij}^2 = 1, \quad \text{if} \ j = (k-1) \times L + i \]

for some \( k = 1, 2, \ldots, K, \) (29)

0 otherwise.

The \( k \)-th element of the gradient, after some algebraic manipulation, becomes:

\[ g_k = - \sum_{j=1}^{L} h_j(w) + b_k, \]

if \( k = 1, 2, \ldots, K; \) (30)

\[ - \sum_{j=1}^{L} h_{(k-K)+L \times (j-1)}(w) + b_k, \]

if \( k = K + 1, K + 2, \ldots, K + L - 1. \)

The Hessian matrix of the dual objective function can be partitioned into a \( 2 \times 2 \) block symmetric matrix, in which the upper-left \((1,1)\) block and the lower-right \((2,2)\) block are diagonal. The \( k_1 \)-th, \( k_1 = 1, 2, \ldots, K \), diagonal element of the Hessian matrix, i.e., the \( k_1 \)-th diagonal element of the \((1,1)\) diagonal block, is

\[ H_{k_1,k_1} = - \sum_{j=(k_1-1) \times L+1}^{k_1 \times L} \frac{h_j(w)}{d_j h_j(w) + \mu}. \] (31)

The \( k_2 \)-th, \( k = K + 1, K + 2, \ldots, K + L - 1 \), diagonal element of the Hessian matrix, i.e., the \((k_1 - K)\)-th diagonal element of the \((2,2)\) diagonal block, is

\[ H_{k_1,k_1} = - \sum_{k=1}^{K} \frac{h_{(k-1) \times L + k - K}(w)}{d_{(k-1) \times L + k_1 - K} h_{(k-1) \times L + k_1 - K}(w) + \mu}. \] (32)

The \((k_1, k_2)\), \( k_1 = 1, 2, \ldots, K \) and \( k_2 = K + 1, K + 2, \ldots, K + L - 1 \), element of the Hessian matrix, i.e., the \((k_1, k_2 - K)\), element of the upper-right \((1,2)\) block, is

\[ H_{k_1,k_2} = - \frac{h_{(k_1-1) \times L + (k_2-2)(w)}}{d_{(k_1-1) \times L + (k_2-2)} h_{(k_1-1) \times L + (k_2-2)}(w) + \mu}. \] (33)

Note that the special structure of the matrix \( A \) associated with the trip distribution problems simplifies the calculation of both the gradient and the Hessian matrix. In the case of gradient, each element of the gradient vector requires a summation of only either \( L \) or \( K \) terms, instead of \( K \times L \) terms required otherwise. As for the Hessian matrix, each of its diagonal element involves the summation of either \( K \) or \( L \) terms, as opposed to \( K \times L \) terms. Moreover, the calculation of any off-diagonal element involves no such summations at all. It turns out that the inversion of the Hessian matrix can also be facilitated by using the block matrix inversion formula.

Given any invertible matrix

\[ B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, \] (34)
its inverse $B^{-1}$ can be obtained as follows:

$$
B^{-1} = \begin{bmatrix}
[B_{11} - B_{12}B_{22}^{-1}B_{21}]^{-1} & -B_{11}^{-1}B_{12}[B_{22} - B_{21}B_{11}^{-1}B_{12}]^{-1} \\
-B_{22}^{-1}B_{21}[B_{11} - B_{12}B_{22}^{-1}B_{21}]^{-1} & [B_{22} - B_{21}B_{11}^{-1}B_{12}]^{-1}
\end{bmatrix},
$$

(35)

Since the (1,1) and (2,2) blocks of the Hessian matrix are diagonal, the inversion of the Hessian matrix can be reduced to the inversion of one $K \times K$ matrix and one $(L - 1) \times (L - 1)$ matrix plus some elementary matrix multiplications.

5. COMPUTATIONAL EXPERIENCE

We first briefly describe how the test problems are randomly generated. The quadratic coefficients are generated randomly by multiplying a random number uniformly distributed in $(0,1)$ by a scaling factor. The positivity guarantees the convexity of the primal objective function. We will report the sensitivity of the algorithm performance with respect to the quadratic scale ranging from 0.01 to 1.0. The linear coefficients are generated similarly. The scaling factor for the linear coefficients ranges from 1.0 to 100.0. The weight of the entropy term is set to three possible values, 5.0, 0.5, or 0.05. The right-hand-side of each of the primal constraint is generated by multiplying a random number uniformly distributed between $(0,1)$ by a demand scale, which ranges from 100.0 to 5000.0.

A major indicator for the robustness of a particular algorithm is its performance as a function of the size of the problem. In what follows, we will use the notation of $#O \times #D$ to denote the size of a problem with $#O$ origins and $#D$ destinations. To highlight the sensitivity of the algorithm to the size of the problem, we chose to (i) generate the right-hand-sides, for most of the test problems, using a common demand scale, 1000.0, (ii) use a common linear scale of 10.0, 0.5 and 1000.0 respectively, (iii) use the common linear scale of 10.0. With these parameters fixed and the quadratic scale set at a particular value, we compare the performance of the algorithm with respect to the problem size. We also vary the quadratic scale and compare its performance as a function of this scale. The reason for varying the quadratic scale is to gauge the sensitivity of the performance with respect to the coefficients of the highest-order polynomial terms. Due to the large number of possible combinations of parameter values, we fix the size of 100 origins and 100 destinations as the base size and vary some other parameters for gauging the performance sensitivity. Also, to make sure that this algorithm can consistently solve large-size problems, we ran four sets of $400O \times 400D$ problems, with only the quadratic scale varying and the linear scale, the weight of the entropy term and the demand scale fixed at 10.0, 0.5 and 1000.0 respectively.

The selected demand scale of 1000.0, in our opinion, should be large enough. For example, in a $400O \times 400D$ problem, this scale would amount to on average 80 million trips. We had selected this large number to demonstrate the power of this algorithm. In solving $P_\alpha$, only the relative magnitude, not their absolute magnitude, of the quadratic scale, linear scale and the weight of the entropy term matters. For example, the optimal solution of the problem defined by the three scales set respectively at 1.0, 100.0, and 5.0 is the same as that of the problem defined by 0.1, 10.0, and 0.5. We believe that the (relative) ranges we have chosen indeed cover most practical cases of interest. The relative magnitude selected for studying the sensitivity of the performance with respect to the problem size is 0.1, 10.0, and 0.5. At these scales, none of the three terms in the primal objective dominates any of the other terms.

Our goal here is to demonstrate the property of global convergence, the rate of convergence, and the robustness of this algorithm with respect to varying individuality of the test problems. It is not our intention to provide evidence on the fastest possible implementation of this algorithm. The matrix inversion used in our implementation is done by Gaussian elimination, not by Cholesky factorization or any other more efficient methods which could further improve the performance. Both $\alpha_k$ and $\beta_k$ are set to 1 in all test cases. The initial solution for the unconstrained dual problem is set at $(0,0,\ldots,0)$ for all test problems. We stop the algorithm only when the norm of gradient becomes zero or no better point can be found by the line search. The $\varepsilon$'s used are $10^{-5}$ and $10^{-10}$ respectively and, for easier comparison, they are set at these same values regardless of problem size. The progression of the objective values does not play any role in the stopping rule. (It is apparent in our test results that the already low number of iterations can be further cut even in half had we chosen to stop the algorithm when the progression does not improve the current objective by more than 0.1%). For use of this algorithm on real problems, the matrix inversion could be performed using more efficient methods, the progression of the objective values may also be used.
as a stopping criterion, and the $\alpha_k$, $\beta_k$, and $\varepsilon$'s and other parameters needed for numerical search should be “fine tuned” for the specific problem size. Also, the performance could benefit from a more intelligent initial solution.

All tests are done on a SUN SPARCstation 2 and the results are tabulated in Table I. Each horizontal block in the table corresponds to one common seed in random problem generation. This arrangement is to enable the study of the role played by the different parameters, e.g., the size of the quadratic scale. The first column contains the test case number. The next two columns, the number of origins and that of destinations, represent the size of the problem. The fourth and the fifth columns give the quadratic scale and the linear scale. The sixth column gives the demand scale. The seventh column contains the weight of the entropy term. The last three columns provide the performance measures: the number of iterations, and the gross and the per-iteration CPU time consumption. Note that the CPU seconds are provided only as a reference.

Here are some observations derived from the test results.

1. The number of iterations increases very slowly as the size of the problem grows. (See the blocks for $100 \times 10^2$ and $500 \times 50^2$, the first three test problems in each of the two $100 \times 100D$ blocks, the blocks for $200 \times 200D$ and $300 \times 300D$, and finally the four $400 \times 400D$ blocks.) Aside from the first block of $400 \times 400D$ problems and test case 35, the number of iterations hardly increases with the problem size at all and basically remain at around 10. Let us reiterate that had we chosen to stop the algorithm when the objective function does not improve by more than 0.1%, most of these numbers can even be further cut in half. This shows an amazingly low degree of sensitivity to the problem size.

2. For the smaller problems, the number of iterations tends to increase as the magnitude of the quadratic scale goes up. However, this is no longer valid for the larger problems. (Compare the three test problems within each of the blocks pointed out in (1).)

3. The number of iterations tends to increase as the magnitude of the linear scale goes up. (Compare, for example, the second and the sixth test problems in the second $1000 \times 100D$ block, i.e., cases 13 and 17.)

4. The number of iterations tends to increase, but only very slightly, as the weight of the entropy term goes up. (Compare, for example, the fourth and the fifth test problems of the second $1000 \times 100D$ block, i.e., cases 15 and 16.)

5. Also, the number of iterations tends to increase, but very slightly, as the magnitude of the demand scale goes up. (Compare, for example, the second, the eighth and the ninth test problems in the second $1000 \times 100D$ block, i.e., cases 19 and 20.)

6. When we increase the magnitude of more than one parameter, the effect on the number of iterations seems to compound. For example, increasing both the linear scale and the demand scale simultaneously would increase noticeably the number of iterations. (Compare the second and the fourth test problems in the first $1000 \times 100D$ block, i.e., cases 8 and 10.) Also, increasing the quadratic scale, the linear scale and the demand scale simultaneously would increase the number of iterations even more. (Compare the second and the fifth test problems in the first $1000 \times 100D$ block, i.e., cases 8 and 11.)

7. Note that some of these test problems may not be realistic and are created simply to test the limit of this algorithm. As mentioned earlier, the performance can be further improved in several ways, some of which depend on the specifics of the practical problems being solved.

For the pure entropy maximization problem with linear constraints, the average number of iterations grows very slowly from 5 for $10 \times 100$ problems to 14 for $500 \times 1000$ problems.\[11\] Our experience with two other classes of problems\[12,13\] is very similar. For larger problems, the major computational requirements at each iteration come from the inversion of the Hessian matrix. Since the calculation of the Hessian matrix has been significantly simplified, the proportion of computation effort spent on solving the one-dimensional implicit functions is significant but decreases as the problem size increases.

By setting the unit quadratic costs $d_{ij}$ in Program $P_\mu$ to zero, we can use the computational procedure and the curved search algorithm to solve the trip distribution problem with only linear cost. A well-known algorithm for solving such a problem is the Bregman’s balancing algorithm.\[17\] In the rest of this section, we compare and discuss the difference in performance between these two methods.

Bregman’s balancing method is applicable in a setting more general than entropy optimization with linear constraints. When applied to solving such entropy optimization problems, Bregman’s method can be viewed as one which minimizes the dual objective with respect to one variable at a
TIME MAXIMIZATION WITH QUADRATIC COST

TABLE I

<table>
<thead>
<tr>
<th>Case #</th>
<th>#O</th>
<th>#D</th>
<th>q_a</th>
<th>l_a</th>
<th>d_a</th>
<th>μ</th>
<th>#(iter)</th>
<th>cpu(s)</th>
<th>cpu/iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>10</td>
<td>0.01</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>10</td>
<td>3.83</td>
<td>0.38</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>10</td>
<td>0.1</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>10</td>
<td>4.00</td>
<td>0.40</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>10</td>
<td>1.0</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>20</td>
<td>10.13</td>
<td>0.51</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>50</td>
<td>0.01</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>8</td>
<td>70.28</td>
<td>8.78</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>50</td>
<td>0.1</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>8</td>
<td>74.94</td>
<td>9.37</td>
</tr>
<tr>
<td>6</td>
<td>50</td>
<td>50</td>
<td>1.0</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>17</td>
<td>192.27</td>
<td>11.31</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>100</td>
<td>0.01</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>7</td>
<td>329.76</td>
<td>47.10</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>100</td>
<td>0.1</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>8</td>
<td>376.02</td>
<td>47.00</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>100</td>
<td>1.0</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>10</td>
<td>503.54</td>
<td>50.35</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>100</td>
<td>0.1</td>
<td>10.0</td>
<td>3000.0</td>
<td>0.5</td>
<td>22</td>
<td>1981.59</td>
<td>90.07</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>100</td>
<td>1.0</td>
<td>10.0</td>
<td>3000.0</td>
<td>0.5</td>
<td>36</td>
<td>3346.44</td>
<td>92.96</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>100</td>
<td>0.01</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>9</td>
<td>432.81</td>
<td>48.09</td>
</tr>
<tr>
<td>13</td>
<td>100</td>
<td>100</td>
<td>0.1</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>9</td>
<td>415.67</td>
<td>46.18</td>
</tr>
<tr>
<td>14</td>
<td>100</td>
<td>100</td>
<td>1.0</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>16</td>
<td>770.62</td>
<td>48.16</td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>100</td>
<td>0.1</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>11</td>
<td>748.14</td>
<td>68.01</td>
</tr>
<tr>
<td>16</td>
<td>100</td>
<td>100</td>
<td>0.01</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>16</td>
<td>848.80</td>
<td>53.05</td>
</tr>
<tr>
<td>17</td>
<td>100</td>
<td>100</td>
<td>0.1</td>
<td>1.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>7</td>
<td>322.77</td>
<td>46.11</td>
</tr>
<tr>
<td>18</td>
<td>100</td>
<td>100</td>
<td>0.01</td>
<td>10.0</td>
<td>5000.0</td>
<td>0.5</td>
<td>10</td>
<td>591.92</td>
<td>59.19</td>
</tr>
<tr>
<td>19</td>
<td>100</td>
<td>100</td>
<td>0.1</td>
<td>1.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>5</td>
<td>378.48</td>
<td>47.31</td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>100</td>
<td>0.1</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>8</td>
<td>432.81</td>
<td>48.09</td>
</tr>
<tr>
<td>21</td>
<td>200</td>
<td>200</td>
<td>0.01</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>12</td>
<td>2391.57</td>
<td>199.29</td>
</tr>
<tr>
<td>22</td>
<td>200</td>
<td>200</td>
<td>0.1</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>10</td>
<td>2211.48</td>
<td>221.15</td>
</tr>
<tr>
<td>23</td>
<td>200</td>
<td>200</td>
<td>1.0</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>11</td>
<td>3033.25</td>
<td>274.84</td>
</tr>
<tr>
<td>24</td>
<td>300</td>
<td>300</td>
<td>0.01</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>12</td>
<td>8196.24</td>
<td>683.02</td>
</tr>
<tr>
<td>25</td>
<td>300</td>
<td>300</td>
<td>0.1</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>10</td>
<td>7242.34</td>
<td>724.23</td>
</tr>
<tr>
<td>26</td>
<td>300</td>
<td>300</td>
<td>1.0</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>13</td>
<td>9953.42</td>
<td>765.65</td>
</tr>
<tr>
<td>27</td>
<td>400</td>
<td>400</td>
<td>0.01</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>24</td>
<td>39390.30</td>
<td>1641.26</td>
</tr>
<tr>
<td>28</td>
<td>400</td>
<td>400</td>
<td>0.1</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>23</td>
<td>37666.60</td>
<td>1637.67</td>
</tr>
<tr>
<td>29</td>
<td>400</td>
<td>400</td>
<td>1.0</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>28</td>
<td>51081.20</td>
<td>1824.33</td>
</tr>
<tr>
<td>30</td>
<td>400</td>
<td>400</td>
<td>0.01</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>11</td>
<td>17745.00</td>
<td>1613.18</td>
</tr>
<tr>
<td>31</td>
<td>400</td>
<td>400</td>
<td>0.1</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>10</td>
<td>16861.70</td>
<td>1686.17</td>
</tr>
<tr>
<td>32</td>
<td>400</td>
<td>400</td>
<td>1.0</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>12</td>
<td>21771.70</td>
<td>1814.31</td>
</tr>
<tr>
<td>33</td>
<td>400</td>
<td>400</td>
<td>0.01</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>12</td>
<td>20850.90</td>
<td>1737.58</td>
</tr>
<tr>
<td>34</td>
<td>400</td>
<td>400</td>
<td>0.1</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>13</td>
<td>21614.30</td>
<td>1662.64</td>
</tr>
<tr>
<td>35</td>
<td>400</td>
<td>400</td>
<td>1.0</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>24</td>
<td>45410.50</td>
<td>1892.10</td>
</tr>
<tr>
<td>36</td>
<td>400</td>
<td>400</td>
<td>0.01</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>8</td>
<td>13613.00</td>
<td>1701.63</td>
</tr>
<tr>
<td>37</td>
<td>400</td>
<td>400</td>
<td>0.1</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>9</td>
<td>15279.60</td>
<td>1697.73</td>
</tr>
<tr>
<td>38</td>
<td>400</td>
<td>400</td>
<td>1.0</td>
<td>10.0</td>
<td>1000.0</td>
<td>0.5</td>
<td>10</td>
<td>18138.20</td>
<td>1813.82</td>
</tr>
</tbody>
</table>

Therefore, the number of iterations required grows very rapidly with respect to the size of the problem. As pointed out by Jornsten and Lundgren,[16] Bregman's method is computationally efficient when all constraint coefficients $a_{ij}$ have the value of zero or one. This is because, with this special structure, no line searches are required. However, with the presence of any non-zero-one coefficients $a_{ij}$ in a constraint, the computational effort required for the associated iteration becomes much larger. Compounded with the rapid increase of the number of iterations as the problem size grows, the computational effort required for general problems grows rapidly. Therefore, the number of general constraints must be limited.[18] Our experience shows that, for the trip distribution problems with only linear cost, in general, Bregman's method tends to be more efficient than our method. Although the number of Bregman iterations grows very fast with respect to the problem size while its counterpart in our approach barely grows, the small computational effort per iteration compensates for the huge number of iterations. However, one clear advantage of our approach is robustness. We demonstrate it by comparing the performance of the two algorithms against two
well-known “benchmark” entropy optimization problem with trip distribution (transportation) constraints. The problems are posed as matrix scaling problems.

The matrix scaling problem is to find two diagonal matrices, $D_1$ and $D_2$, for a given square matrix $M$ such that the scaled matrix $D_1MD_2$ has prescribed row and column sums. This problem is equivalent to solving for a gravity model and hence equivalent to a trip distribution problem with linear cost. Two matrix scaling problems, denoted by $P_1$ and $P_2$, respectively, with

$$
M_1 = \begin{bmatrix}
10^4 & 10^2 & 10^2 \\
10^2 & 1 & 1 \\
10^2 & 1 & 1
\end{bmatrix}
$$

and

$$
M_2 = \begin{bmatrix}
10^2 & 10^2 & 0 \\
10^2 & 10^4 & 1 \\
0 & 1 & 10^2
\end{bmatrix}
$$

were studied by Elfving and others. Note that scaling $M_1$ is equivalent to a trip distribution problem with $\mu = 1$,

$$
e_1 = \begin{bmatrix}
-ln(10^4) \\
-ln(10^2) \\
-ln(10^2) \\
-ln(10^2) \\
-ln(1) \\
-ln(10^2) \\
-ln(1) \\
-ln(1)
\end{bmatrix}, \quad A_1 = \begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}
$$

and

$$
b_1 = \begin{bmatrix}
1 \\
1 \\
1 \\
1 \\
1 \\
1
\end{bmatrix}.
$$

Since the rank of matrix $A_1$ is 5 (one of the scaling constraints is redundant), we can eliminate the last constraint to obtain a full-rank matrix. $P_2$ is defined similarly. Note that there are two zeros in $M_2$. We replace them by a small number $10^{-10}$ before applying the two algorithms.

Both algorithms stop when the maximum violation of the primal constraints is no larger than a preset threshold. (For the curved-search algorithm, we set the other parameters to the same values used for the earlier tests.) Two selected thresholds are 0.001 and 0.00001. The performance results for the two methods are shown in Table II and Table III, respectively. Note that the performance of the Bregman's method is very sensitive to the individuality of the problem and is not affected as drastically as the curved-search method when the constraint matrix contains non-zero-one entries. The number of iterations, as a function of problem size, grows very slowly, which makes it especially attractive for the case of large-size general constraint matrix $A$. As mentioned earlier, our algorithm can be fine tuned to improve the performance, e.g., using the factorization techniques to speed up matrix inversion. A comprehensive computational comparison among several major algorithms for the entropy optimization problem with linear constraints is a worthy subject for future study.

### 6. CONCLUSION

In this paper, we present a general entropy optimization model, together with an efficient dual-based algorithm, which not only encompasses many existing transportation planning models but also extends them. In addition to the capability of accommodating quadratic cost in Program $P_n$, our approach has several important advantages. First
of all, the strong duality theorem holds under very mild conditions and the dual program is an unconstrained convex program. A primal optimal solution can be easily obtained from the dual optimal solution through a simple conversion equation. Second, the post-optimality analysis, including parametric programming and sensitivity analysis, can benefit from the unconstrained nature of the dual. Third, the robustness and efficiency of the curved-search algorithm make this approach particularly suitable for large-size real-life problems. More sophisticated implementations of the algorithm can further improve the already impressive performance and the high degree of "parallelizability" of most of the computation can be further exploited for higher efficiency.

ACKNOWLEDGMENT

The authors would like to thank two anonymous referees and an Associate Editor of Transportation Science for their very helpful comments and suggestions. The work of the first author was partially supported by the North Carolina Supercomputing Center, the 1992 Cray Research Award, and the National Science Council Grant #NSC 81-0415-E-007-10 of the Republic of China.

REFERENCES


(Received, February 1993; revisions received: November 1993, September 1994; accepted October 1994)