AN EXACT ALGORITHM FOR THE CONCAVE TRANSPORTATION PROBLEM

LEON COOPER and MARY W. COOPER
Department of Computer Science and Operations Research, Institute of Technology, Southern Methodist University, Dallas, Texas 75275, U.S.A.

(Received June 1975)

Abstract—An exact method for solving a class of concave transportation problems which reflect economies of scale is presented. By exploiting concepts of dynamic programming and an analysis of the nature of the recursion, an analytic representation of the optimal allocation at each stage has been developed. This completely avoids the impossible storage requirements of higher dimensional dynamic programming.

1. INTRODUCTION
The ordinary Hitchcock transportation is concerned with finding the combination of amounts to be shipped from a set of sources to a set of destinations which minimizes a linear cost function subject to constraints relating to supply and demand. However, departures from linearity are common in real world applications. Indeed, even departures from convex cost functions are the more applicable and interesting cases. Linear cost functions assume that costs are independent of the amount shipped, a circumstance that rarely applies. If costs increased with the amount shipped, then we would have a convex cost function. While such cases are known, they are not common. A case of great interest, but unfortunately the most intractable to deal with mathematically, is that of the case where there are economies of scale, i.e., the costs of shipping tend to decrease as the amount shipped increases. For this case, we have a concave cost function. Such a concave cost function, when restricted to a convex set, may possess many local optima. This is what makes the problem difficult to solve.

The problem to be considered in this paper is a version of the concave transportation problem. Specifically, we address the following problem:

\[
\min \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}(x_{ij})x_{ij}
\]

\[
\sum_{j=1}^{n} x_{ij} = s_i, \quad i = 1, 2, \ldots, m
\]

\[
\sum_{i=1}^{m} x_{ij} = r_j, \quad j = 1, 2, \ldots, n
\]

\[
x_{ij} \geq 0, \quad \text{all } i, j
\]

In (1)–(4), \(x_{ij}\) represents the amount to be shipped from source \(i\) to destination \(j\). \(s_i\) is the amount on hand at source \(i\) and \(r_j\) is the required amount at destination \(j\). The functions \(f_{ij}(\cdot)\) are assumed to be non-decreasing piecewise linear concave functions such that, if \(I_\rho\) denotes the set of non-negative integers and \(R_\rho\) denotes the non-negative real line, then \(f_{ij}(\cdot)\) are defined as:

\[
f_{ij} : R_\rho \cap I_\rho \rightarrow I_\rho, \quad i = 1, 2, \ldots, m
\]

\[
j = 1, 2, \ldots, n
\]

A typical \(f_{ij}(\cdot)\) is shown in Fig. 1. The non-decreasing piecewise linear concave functions will be defined only for integral values of \(x_{ij}\) since if \(s_i, i = 1, 2, \ldots, m\) and \(r_j, j = 1, 2, \ldots, n\) are integers, the \(x_{ij}\) will take on only integral values. Any continuous concave function of the type under discussion can be suitably approximated by a piecewise linear approximation. Multiplication by an appropriate scalar will yield the property set forth in (5).

We shall assume that \(z\) is bounded and that the constraints (2)–(4) define a convex set with at least one feasible integer point.

Previous approaches to non-convex programming problems have typically involved the use of
branch-and-bound methodology (see e.g. [2], [3]). The approach taken in this paper is quite different as will become apparent in the description in the next section. It is related to some previous work on integer programming[1].

2. GENERAL DESCRIPTION OF THE ALGORITHM

The general idea of the proposed algorithm is to search candidate hypersurfaces for lattice points. This proceeds roughly as follows. For each of the \( f_i(\cdot) \), a linear underestimator is easily determined so that the problem given by (1)-(4) can be solved as a linear transportation problem. Suppose the optimal value of the objective function for this linear approximation problem is \( z^0 \).

In addition, let the optimal value of the objective function for (1) (4) be designated \( z^* \). It is clear that \( z^* \geq z^0 \). The basic idea behind the hypersurface search algorithm we propose, is to start at the transportation problem solution and search the hypersurface \( \sum_{i=1}^{m} \sum_{j=1}^{n} f_i(x_{ij})x_{ij} = \lfloor z^0 \rfloor \) (where \( \lfloor \alpha \rfloor \) indicates the least integer greater than or equal to \( \alpha \)) to see whether or not it contains any feasible lattice points. If it does, we are done. If it does not, we move the hypersurface in a direction parallel to itself and then search the hypersurface \( \sum_{i=1}^{m} \sum_{j=1}^{n} f_i(x_{ij})x_{ij} = \lfloor z^0 \rfloor + 1 \). Since \( f_i(\cdot) \) was defined as in (5) and if all \( s_i \) and \( r_j \) are integers, then \( \sum_{i=1}^{m} \sum_{j=1}^{n} f_i(x_{ij})x_{ij} \) will be integral as will the \( x_{ij} \).

If the hypersurface \( \sum_{i=1}^{m} \sum_{j=1}^{n} f_i(x_{ij})x_{ij} = \lfloor z^0 \rfloor + 1 \) contains at least one feasible lattice point, we are done. If it does not, we continue the process. This procedure is clearly finite. Since the convex set defined by (2)-(4) was assumed to contain at least one integer point, we must eventually find it.

We summarize the notation we will use:

- \( z^* = \) optimal value of the objective function in (1)
- \( z^* = \) optimal value of the objective function for transportation problem using linear costs (see Fig. 1) and constraints (2)-(4)
- \( z_k = \lfloor z^0 \rfloor + k, \ k = 0, 1, 2, \ldots \)
- \( S = \{ x_{ij} \mid \sum_{j=1}^{n} x_{ij} = s_i, v_i \wedge \sum_{i=1}^{m} x_{ij} = r_i, v_i \wedge x_{ij} \geq 0, v_i \} \)

In addition, it is clear that each of the linear segments that make up the piecewise linear function (see Fig. 1) \( f_i(\cdot) \) has the form

\[
f_i(x_{ij}) = a_i x_{ij} + b_i.
\]

To distinguish which linear segment we refer to, we shall use a third subscript \( v \). Therefore

\[
f_{iv} = a_{iv} x_{ij} + b_{iv} \quad v = 1, 2, \ldots, V \tag{6}
\]

Correspondingly, for each line segment it will be the case that:

\[
0 = l_{iv} \leq x_{ij} \leq u_{iv}
\]

\[
l_{iv} \leq x_{ij} \leq u_{iv}
\]

\[
l_{iv} < x_{ij} < u_{iv} \tag{7}
\]
Hypersurface search algorithm

1. Solve the transportation problem

\[
\min z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}
\]

\[
\sum_{j=1}^{n} x_{ij} = s_i \quad i = 1, 2, \ldots, m
\]

\[
\sum_{i=1}^{m} x_{ij} = r_j \quad j = 1, 2, \ldots, n
\]

\[x_{ij} \geq 0, \quad \text{all } i, j\]

2. Lower bounds for each \(x_{ij}\) are zero, i.e., \(u_{ij} = 0\). Upper bounds for each \(x_{ij}\) are \(\min(s_i, r_j)\), i.e., \(u_{ij} = \min(s_i, r_j)\).

3. Find all combinations of \(x_{ij}\), \(i = 1, 2, \ldots, m; j = 1, 2, \ldots, n\) which satisfy:

\[
\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}(x_{ij}) x_{ij} = z_k
\]

\[0 \leq x_{ij} \leq u_{ij}, \quad \text{all } i, j\]

\[x_{ij} \text{ integer, } \quad \text{all } i, j\]

4. If no integer valued vector \(\{x_{ij}\}_k\) can be found, increase \(k\) by 1 and return to Step 3. If at least one \(\{x_{ij}\}_k\) is all integer, go to Step 5.

5. If at least one of the \(\{x_{ij}\}_k \in S\) then we are done. If for all \(\{x_{ij}\}_k, \{x_{ij}\}_k \in S\), increase \(k\) by 1 and return to Step 3.

We may note that since the Set \(S\) is non-empty, bounded and contains at least one integer point, the finiteness of the algorithm is guaranteed. How efficient such an algorithm can be depends very strongly on how Step 3 is carried out. Steps 1, 2, 4 and 5 are self-evident. It should be noted that the constraints of the problem, (2)-(4) do not explicitly enter into Step 3. They are used only in Step 5 to check feasibility, and also to determine upper bounds. In the next section we consider an approach to solving the problem of Step 3.

3. Hypersurface search by dynamic programming

We shall now consider how we may employ a problem formulation and solution method that utilizes dynamic programming methodology. We wish to deal with the following problem.

Find all combinations of \(x_{ij}, \quad i = 1, 2, \ldots, m; \quad j = 1, 2, \ldots, n\) which satisfy:

\[
\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}(x_{ij}) x_{ij} = z_k
\]

\[0 \leq x_{ij} \leq u_{ij} \quad i = 1, 2, \ldots, m
\]

\[x_{ij} \text{ integer } \quad j = 1, 2, \ldots, n\]

where

\[f_{ij}(x_{ij}) = a_{ij} x_{ij} + h_{ij}, \quad v = 1, 2, \ldots, v
\]

\[l_{ij} \leq x_{ij} \leq u_{ij}, \quad v = 1, 2, \ldots, V
\]

We shall assume that \(l_{ij} = 0\) in all cases. We shall further assume, without loss of generality, that \(a_{11v}\) is such that for some \(v = p\),

\[x_{11} = \lambda - \frac{b_{11p}}{a_{11p}} = \text{integer. } \lambda \text{ integer}\]

This assumption results in a great simplification in the computational equations to be derived. If it
is not true in the problem as stated, we can append to the problem a variable \( x_{i0} \) such that \( a_{i0} = 1 \), \( b_{i0} = 0 \) and force this variable to be zero in the final solution by adding a constraint of the form \( x_{i0} = 0 \) to the set of constraints (2)-(4).

We may formulate the problem stated in (10) in the following manner.

\[
\begin{align*}
\min z &= \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}(x_{ij}) x_{ij} \\
\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}(x_{ij}) x_{ij} &= z_k \\
0 &\leq x_{ij} \leq u_{ij} \quad i = 1, 2, \ldots, m \\
x_{ij} \text{ integer} &\quad j = 1, 2, \ldots, n
\end{align*}
\]

(14)

The fact that we already know the minimum value of \( z \) for the problem given in (14), viz., \( z_k \), does not render (14) a trivial problem, since what we seek is to determine whether or not there exists a set of integer \( x_{ij} \) satisfying the constraints of (14).

The problem stated in (14) can be solved by means of dynamic programming. Since the objective function and the single structural constraint of (14) are separable and non-decreasing in the variables \( x_{ij} \), the sufficient conditions for invoking the principle of optimality and deriving a dynamic programming solution are satisfied (see [4]). Before applying the principle of optimality to derive the recursion formulae, we define some notation we require.

\[
S_{ijv} = \text{set of indices } v = 1, 2, \ldots, V \text{ for } f_{ij}(\cdot)
\]

\[
\Lambda_{st} = \sum_{j=1}^{n} f_{ij}(u_{ijv}), \quad t = 1, 2, \ldots, n
\]

\[
\Lambda_{st} = \sum_{j=1}^{n} f_{ij}(u_{ijv}) + \sum_{j=1}^{i-1} f_{ij}(u_{ijv}), \quad s > 1, \quad t = 1, 2, \ldots, n
\]

Applying the principle of optimality to the problem given by (14) yields the following recursion relations for the optimal return functions.

\[
g_{st}(\lambda) = \min_{x_{i1}, x_{i1} + b_{11v} \in S_{i1v}} \{a_{i1} x_{11} + b_{11v} | v \in S_{i1v} \} = \lambda, \quad 0 = \lambda \leq u_{i1v}
\]

(15)

where \( p \in S_{i1v} \)

\[
g_{st}(\lambda) = \min_{x_{st} = x_{s1} \in \delta_{s1t}} \{a_{st} x_{st} + b_{ste} + g_{s-1}(\lambda - a_{ste} x_{st} - b_{ste})\},
\]

\[
s = 1, 2, \ldots, m; \quad st \neq 11
\]

\[
t = 1, 2, \ldots, n
\]

\[
\lambda = 0, 1, 2, \ldots, \Lambda_{st}
\]

(16)

where \( \delta_{s1t}(\lambda) = \min \left[ \left\lceil \frac{\lambda}{a_{ste}} \right\rceil, u_{ste} \right] \).

The subscript \( st - 1 \) on \( g_{st-1} \) in (16) is meant to be read as "one less than \( st \"", i.e., if \( m = 3 \), \( n = 4 \), \( g_{22} \) is \( g_{22}, g_{31} \) is \( g_{31} \), etc. i.e. the \( g_{st} \) functions are numbered sequentially across the rows beginning with \( g_{11} \). We use \( g_{st} \) rather than \( g_{st} \) in (16) since \( g_{st}(\lambda) = \min_{1 \leq v < V} g_{st}(\lambda) \).

The usual dynamic programming approach would be to calculate

\[
g_{st}(\lambda), x^{*}_{st}(\lambda) \quad \lambda = 0, 1, 2, \ldots, \Lambda_{st}
\]

for \( s = 1, 2, \ldots, m; t = 1, 2, \ldots, n - 1 \), where \( x^{*}_{st}(\lambda) \) is the value of \( x_{st} \) which produced \( g_{st}(\lambda) \) for each value of \( \lambda \). Finally, we would calculate \( g_{m0}(z_k) \) and \( x^{*}_{m0}(z_k) \), assuming a solution exists. We
would then subtract \( f_{mn}(x^*_{m'n})x^*_{m'n} \) from \( z_k \) and find, corresponding to \( \lambda = z_k - f_{mn}(x^*_{m'n})x^*_{m'n} \) in the tabulation of \( x^*_{m'n-1}(\lambda) \), the value of \( x^*_{m'n-1}(z_k - f_{mn}(x^*_{m'n})x^*_{m'n}) \). This backwards process would yield successively, \( x^*_{m'n}, x^*_{m'n-1}, \ldots, x^*_{11} \).

The principal objection to the approach delineated in the foregoing is the amount of storage required. While it is orders of magnitude less than what would be required by the simple-minded approach of using a state variable for each constraint of the original problem (1)-(4), the amount of storage required is still quite considerable. For each variable a vector \( x^*_{s}(\lambda) \) must be stored. Furthermore, there are often many alternate optimal values of \( x^*_{s}(\lambda) \) which minimize

\[
a_{st}x_{st} + b_{st} + g_{st} - (\lambda - a_{st}x_{st} - b_{st})
\]

and they must all be stored. Hence \( x^*_{s}(\lambda) \) is actually a matrix, say of average dimension \( = (\lfloor u_{st}/2 \rfloor \times \lambda_{st}) \). Hence the total amount of storage required is approximately \( \sum_{s=1}^{m} \sum_{t=1}^{n} (u_{st} \lambda_{st}/2) \). Even for relatively small problems, hundreds of millions of storage words might be required.

In the following section, a set of equations will be derived which will give explicit formulae for \( x^*_{s}(\lambda) \) for any \( \lambda \). Hence the need for a complete tabulation of \( x^*_{s}(\lambda) \) will be eliminated entirely. Indeed, as will be seen, \( g_{st}(\lambda) \) need never be explicitly calculated. The reduction in storage is drastic and renders the hypersurface algorithm just described of practical use. The storage requirements, as will be seen, are minimal. The entire calculation process will be reduced to calculating \( x^*_{m'n}(z_k), x^*_{m'n-1}(z_k - f_{mn}(x^*_{m'n})x^*_{m'n}), \ldots, x^*_{11}\left(z_k - \sum_{s=1}^{m} \sum_{t=1}^{n} f_{st}(x^*_{s})x^*_{st}\right) \) directly.

4. DERIVATION OF EQUATIONS FOR OPTIMAL SOLUTION

The derivations that follow are concerned with obtaining a set of expressions that yield \( x^*_{s}(\lambda) \) by applying the recursion relations (15) and (16) to the following problem:

\[
\begin{align*}
\min z &= \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}(x_{ij})x_{ij} \\
\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}(x_{ij})x_{ij} &= z_k \\
0 &\leq x_{ij} \leq u_{ij} \quad i = 1, 2, \ldots, m \\
x_{ij} &\text{ integer } \quad j = 1, 2, \ldots, n
\end{align*}
\]

(17)

It will be recalled that we assumed earlier that for some \( v = p \),

\[
x_{11} - \frac{\lambda - b_{11p}}{a_{11p}} \quad \text{integer}
\]

In what follows equations are derived for the optimal allocations \( x^*_{s}(\lambda), s = 1, 2, \ldots, m; t = 1, 2, \ldots, n \). The validity of these results is established in the lemmas which follow.

**Lemma 1**

\[
x_{11}(\lambda) = \frac{\lambda - b_{11p}}{a_{11p}} \quad \text{integer}, \quad 0 \leq \lambda \leq u_{11v}
\]

for some \( p \in S_{11} \).

**Proof.** The dynamic programming solution which results from the application of the principle of optimality yields for a single stage (variable) process the following optimal return function:

\[
g_{s}(\lambda) = \min_{x_{11}, x_{11} + b_{11v} \in S_{11}} \{a_{11}, x_{11} + b_{11v} | v \in S_{11}\} = \lambda.
\]

This must be the case since, in the backwards recursion, the first stage is reached last. If there is
an amount \( \lambda \) left to allocate, then \( a_{11}x_{11} + b_{11} = \lambda \). Since we assumed that it will be the case initially, or can be made to be the case, that for some \( v = p \),

\[
x_{11} = \frac{\lambda - b_{11}}{a_{11}} = \text{integer}
\]

and \( l_{11} \leq x_{11} \leq u_{11} \), then the calculation of \( x_1^*(\lambda) \) is a simple search over the possible values of \( x_{11} \) to find the integer value. This concludes the proof.

We emphasize again that in the Lemmas which follow, the subscript \( st-1 \) on \( g_{st-1} \) and \( \Lambda_{st-1} \) refers to the function \( g_{s-1, t-1}(\cdot) \) and \( \Lambda_{s-1, t-1}(\cdot) \) or \( g_{s-1, m}(\cdot), \Lambda_{s-1, m}(\cdot) \) since the rows are numbered consecutively in this recursion.

**Lemma 2**

\[
x_1^*(\lambda) = 0, \quad \lambda < a_{st} + b_{st} - 1 < \Lambda_{st-1}, \qquad v \in S_{st},
\]

\[
s = 1, 2, \ldots, m; \quad t = 1, 2, \ldots, n; \quad st \neq 11
\]

**Proof.** Since the \( f_0 \) consist of piecewise linear segments, it then follows that:

\[
g_{st}(\lambda) = \min_{t, \lambda - \lambda} g_{stu}(\lambda).
\]

Therefore, from the application of the principle of optimality of dynamic programming, we obtain the following recursion relations:

\[
g_{st}(\lambda) = \min_{t, \lambda - \lambda} \left\{ a_{stu}x_{st} + b_{stu} + g_{stu}(\lambda - a_{stu}x_{st} - b_{stu}) \right\}.
\]

\[
s = 1, 2, \ldots, m
\]

\[
t = 1, 2, \ldots, n; \quad st \neq 11
\]

\[
\lambda = 0, 1, \ldots, \Lambda_{st}
\]

where

\[
\delta_{stu}(\lambda) = \min \left( \left[ \frac{\lambda - b_{stu}}{a_{stu}}, u_{stu} \right] \right)
\]

If we can show that \( x_{st}^* \neq 0 \) for all \( v \in S_{stu} \), we are done. Consider \( \delta_{stu}(\lambda) \). By definition, it is

\[
\delta_{stu}(\lambda) = \min \left( \left[ \frac{\lambda - b_{stu}}{a_{stu}}, u_{stu} \right] \right)
\]

However, by hypothesis, we have that:

\[
\lambda \leq a_{stu} + b_{stu} - 1
\]

Therefore, the maximum value of \( \lambda \) is \( a_{stu} + b_{stu} - 1 \). Substituting this value of \( \lambda \) into (19) yields:

\[
\delta_{stu}(\lambda) = \min \left( \left[ \frac{a_{stu} + b_{stu} - 1 - b_{stu}}{a_{stu}}, u_{stu} \right] \right) = 0.
\]

Since \( x_{st} \leq \delta_{stu}(\lambda) = 0 \), we have established the result.

**Lemma 3.** We are given any pair of straight lines, \( y = m_1x_1 + b_1 \) and \( y = m_2x_2 + b_2 \) with \( m_1, m_2, b_1, b_2 > 0 \) and such that the two lines intersect at a point \( (x_0, y_0) \) with \( x_0 \geq 1 \). If \( m_1 > m_2 \) and \( b_1 < b_2 \), then \( m_2 + b_2 > m_1 + b_1 \).
Proof. Since $x_0 \geq 1$ and $b_1, b_2 \geq 0$, then it follows that:

$$x_0(b_2 - b_1) \geq b_2 - b_1$$

(21)

Since $m_1, m_2, b_1, b_2 \geq 0$ then $y_0 \geq 0$. Therefore, we can add $y_0$ to both sides of (21) to yield:

$$y_0 + b_2x_0 - b_2x_0 \geq y_0 + b_2 - b_1$$

which upon rearrangement becomes:

$$y_0 - b_2 + b_2x_0 \geq y_0 - b_1 + b_1x_0.$$  
(22)

Dividing (22) by $x_0 \geq 1$, we have:

$$\frac{y_0 - b_1}{x_0} + \frac{b_2}{x_0} \geq \frac{y_0 - b_1}{x_0} + b_1$$

or

$$m_2 + b_2 \geq m_1 + b_1$$

which was to be proven.

**Lemma 4**

$$x^*_t(\lambda) = \hat{\delta}_{at}(\lambda) = h_{at}(\lambda), k_{at}(\lambda), ..., k_{at}(\lambda), \delta_{at}(\lambda), ..., \delta_{at}(\lambda),$$

where

$$\hat{\delta}_{at}(\lambda) = \begin{cases} \delta_{at}(\lambda) & \text{if } \delta_{at}(\lambda) \text{ exists} \\ \infty & \text{if } \delta_{at}(\lambda) \text{ does not exist} \end{cases}$$

and

$$\delta_{at}(\lambda) = \min \left( \frac{\lambda - b_{at}}{a_{at}} > I_{at}, u_{at} \right)$$

Proof. By Lemma 3, we see that if $\lambda \geq a_{at} + b_{at}$, then $\lambda \geq a_{at} + b_{at}$ for $n = p - 1$, $p - 2, ..., 1$.

However, this fact alone does not guarantee that $[(\lambda - b_{at})/a_{at}]$ for $n = p, p - 1, ..., 1$ will necessarily be in the range $l_{at} \leq \lambda <= u_{at}$. Bearing this in mind, we invoke the principle of optimality to derive the following recursion relations:

$$g_{at}(\lambda) = \min_{l_{at}, u_{at} = \delta_{at}(\lambda)} [a_{at}x_{at} + b_{at} + g_{at-1}(\lambda - a_{at}x_{at} - b_{at})], \quad v = 1, 2, ..., V$$

(23)

where

$$\delta_{at}(\lambda) = \min \left( \frac{\lambda - b_{at}}{a_{at}} > I_{at}, u_{at} \right)$$

If $[(\lambda - b_{at})/a_{at}] < l_{at}$, then $\delta_{at}(\lambda)$ will not exist. Hence for these cases no $x^*_t(\lambda)$ will exist.
However, if $\delta_{stv}(\lambda)$ does have a minimum, then since $\lambda \leq \Lambda_{st-1}$ and $\lambda - a_{stv}x_{stv} - b_{stv} \geq 0$, $v = 1, 2, \ldots, p$, we see that $g_{stv}(\cdot)$ will have values and be defined for all $\xi$ in the range: $l_{st1} \leq \xi \leq \delta_{stv}$ except where $\delta_{stv}(\lambda)$ does not exist.

In other words, if we define:

$$
\begin{align*}
I_{stv}(\lambda) &= \begin{cases} 
\delta_{stv}(\lambda) & \text{if } \delta_{stv}(\lambda) \text{ exists} \\
- & \text{if } \delta_{stv}(\lambda) \text{ does not exist}
\end{cases} \\
\delta_{stv}(\lambda) &= \begin{cases} 
\delta_{stv}(\lambda) & \text{if } \delta_{stv}(\lambda) \text{ exists} \\
- & \text{if } \delta_{stv}(\lambda) \text{ does not exist}
\end{cases}
\end{align*}
$$

Then we have established that under the conditions of the hypothesis, that

$$
\mathbf{x}_{stv}^*(\lambda) = I_{stv}(\lambda), \hat{I}_{stv}(\lambda) + 1, \ldots, \delta_{stv}(\lambda), \hat{I}_{stv}(\lambda), \ldots, \delta_{stv}(\lambda), \ldots, \delta_{stv}(\lambda)
$$

**Lemma 5.** Let

$$
x_{stv}^*(\lambda) = \max \left( \frac{\lambda - (\Lambda_{stv} + 1) - b_{stv}}{a_{stv}}, \ I_{stv} \right)
$$

Then

$$
x_{stv}^*(\lambda) = x_{\text{min}}(\lambda), x_{\text{min}}(\lambda) + 1, \ldots, w_{stv} - 1, w_{stv}, \Lambda_{stv} \leq \lambda \leq \Lambda_{stv}
$$

where

$$
q = \{ \min \nu \in S_{stv} | \lambda - a_{stv} - b_{stv} \leq \Lambda_{stv} + 1 \}
$$

and

$$
w_{stv} = u_{stv} - \theta^*, \lambda \geq a_{stv}(u_{stv} - \theta^*) + b_{stv}
$$

and

$$
\theta^* = \max \{ a_{stv}(u_{stv} - \theta) + b_{stv} - \lambda \mid 0, \theta = 0, 1, 2, \ldots, t = 1, 2, \ldots, n, st \neq 11/n
$$

**Proof.** The application of the principle of optimality of dynamic programming results in the following recursion relations.

$$
g_{stv}(\lambda) = \min_{u_{stv} = x_{stv}, \delta_{stv}} \left[ a_{stv}x_{stv} + b_{stv} + g_{stv-1}(\lambda - a_{stv}x_{stv} - b_{stv}) \right],
$$

$v = 1, 2, \ldots, V$

$$
\Lambda_{stv} \leq \lambda \leq \Lambda_{stv}
$$

(24)

where

$$
\delta_{stv}(\lambda) = \min \left( \left[ \frac{\lambda - b_{stv}}{a_{stv}}, u_{stv} \right] \right)
$$

If we expand (24) we have:

$$
g_{stv}(\lambda) = \min \left[ 0 + b_{stv} + g_{stv-1}(\lambda - b_{stv}), a_{stv} + b_{stv} + g_{stv-1}(\lambda - a_{stv} - b_{stv}), \ldots, a_{stv}u_{stv} + b_{stv} - g_{stv-1}(\lambda - a_{stv}u_{stv} - b_{stv}) \right].
$$

(25)
Since \( \lambda > \Lambda_{st-1} \), by hypothesis and by definition, \( g_{st-1}(\cdot) \) is undefined for \( \lambda > \Lambda_{st-1} \), only those terms for which \( g_{st-1}(\cdot) \) has an argument less than or equal to \( \Lambda_{st-1} \) are defined. Without loss of generality, we can take:

\[
\Lambda_{st-1} = \sum_{i=1}^{n} \sum_{j=1}^{n} f_i(u_{ij}) + \sum_{j=1}^{n} f_i(u_{ij}).
\]  

(26)

In addition, all of the following relations are a consequence of the characteristics of the \( f_i \):

\[
f_i(u_{ij}) > f_i(b_{st1}) \quad \text{since} \quad u_{ij} > b_{st1}
\]

\[
f_i(b_{st1}) \geq b_{st1}
\]

\[
f_i(u_{ij}) \geq u_{ij}.
\]

(27)

Equations (26), (27) and \( \lambda > \Lambda_{st-1} \), taken together imply that \( x^*_{st}(\lambda) > 0 \), when they are substituted into (25).

To determine the minimum value of \( x^*_{st}(\lambda) \), we note that since \( \lambda > \Lambda_{st-1} \) and \( x^*_{st}(\lambda) > 0 \), then

\[
\lambda - a_{st} - b_{st} \leq \Lambda_{st-1} + 1 \quad \text{for some} \quad v = q \in S_{stq}
\]

therefore

\[
x_{st} \geq \frac{\lambda - (\Lambda_{st-1} + 1) - b_{stq}}{a_{stq}}
\]

(29)

However, \( x_{st} \) must be an integer. Hence we can rewrite (29) as:

\[
x_{st} = \left\lfloor \frac{\lambda - (\Lambda_{st-1} + 1) - b_{stq}}{a_{stq}} \right\rfloor.
\]

(30)

In addition we know that \( x_{st} \geq l_{stq} \). Therefore, we have that:

\[
x_{st} \geq \max \left( \left\lfloor \frac{\lambda - (\Lambda_{st-1} + 1) - b_{stq}}{a_{stq}} \right\rfloor, l_{stq} \right).
\]

(31)

If the max is taken on by \( l_{stq} \), then the minimum value of \( x_{st} \) is \( l_{stq} \). However, if the max is taken on by

\[
\left\lfloor \frac{\lambda - (\Lambda_{st-1} + 1) - b_{stq}}{a_{stq}} \right\rfloor,
\]

then we know that since \( x^*_{st}(\lambda) > 0 \) that

\[
\lambda \geq \Lambda_{st-1} + 1 + a_{stq} + b_{stq}
\]

(32)

If we substitute \( \lambda - \text{right-hand side of (32)} \) into (30), we then have

\[
x_{st} \geq \left\lfloor \frac{\Lambda_{st-1} + 1 + a_{stq} + b_{stq} - \Lambda_{st-1} - 1 - b_{stq}}{a_{stq}} \right\rfloor = 1.
\]

However, \( x_{st} \) cannot equal 1 when \( \lambda = \Lambda_{st-1} + 1 + a_{stq} + b_{stq} \) for then we would have:

\[
g_{stq}(\lambda) = \min \{ a_{stq}l_{stq} + b_{stq} + g_{stq}(\Lambda_{st-1} + d - a_{stq}l_{stq} - b_{stq}), \]

\[
a_{stq}(l_{stq} + 1) + b_{stq} + g_{stq}(\Lambda_{st-1} + d - a_{stq}(l_{stq} + 1) - b_{stq}), \ldots ,
\]

\[
a_{stq}l_{stq}(\lambda) + b_{stq} + g_{stq}(\Lambda_{st-1} + d - a_{stq}l_{stq}(\lambda) - b_{stq})\},
\]

(33)
The expression (33) for \( g_{st}(\lambda) \) will contain terms for which \( g_{st-1}(\cdot) \) is not defined. This will occur precisely for those terms for which
\[
d > a_{st}l_{st} + b_{st}.
\]
This leads then to the minimum value for \( x_{st}^{*} \) as:
\[
\left\lfloor \frac{\lambda - (\Lambda_{st-1} + 1) - b_{st}}{a_{st}} \right\rfloor + 1.
\]
This can be seen as follows. If \( d = a_{st}l_{st} + b_{st} + 1 \) then
\[
\left\lfloor \frac{\lambda - (\Lambda_{st-1} + 1) - b_{st}}{a_{st}} \right\rfloor = \left\lfloor \frac{\Lambda_{st-1} + a_{st}l_{st} + b_{st} + 1 - \Lambda_{st-1} - 1 - b_{st}}{a_{st}} \right\rfloor = l_{st}.
\]
However,
\[
\left\lfloor \frac{\lambda - (\Lambda_{st-1} + 1) - b_{st}}{a_{st}} \right\rfloor > l_{st}.
\]
Therefore, the minimum value is:
\[
\left\lfloor \frac{\lambda - (\Lambda_{st-1} + 1) - b_{st}}{a_{st}} \right\rfloor + 1.
\]
In summary then, the minimum value of \( x_{st}^{*}(\lambda) \) is given by:
\[
x_{st}^{*}(\lambda) = \max \left( \left\lfloor \frac{\lambda - (\Lambda_{st-1} + 1) - b_{st}}{a_{st}} \right\rfloor + 1, l_{st} \right).
\] (34)
Larger values of \( x_{st}^{*} \), will be permitted, i.e. for \( v > q \), since \( \lambda - a_{st}x_{st} - b_{st} \) will decrease as \( x_{st} \) increases and hence values of \( g_{st-1}(\cdot) \) will exist for these arguments. However, there is an upper bound on \( x_{st}^{*}(\lambda) \) for \( \Lambda_{st-1} < \lambda \leq \Lambda_{st} \). This will be called \( w_{s,t} \) and is derived as follows. If \( \lambda \geq a_{stv}u_{stv} + b_{stv} \), then clearly, the largest value of \( x_{st} \) is \( u_{stv} \). However, if \( \lambda < a_{stv}u_{stv} + b_{stv} \), we wish to find the largest value of \( x_{st} \) compatible with that value of \( \lambda \). We recall that when \( \lambda \geq a_{stv}u_{stv} + b_{stv} \), the largest value of \( x_{st} \) is \( u_{stv} \). Let us now suppose that:
\[
\lambda \geq a_{st}w_{st} + b_{st} = a_{stv}(u_{stv} - \theta^{*}) + b_{stv}. \] (35)
In order to make \( w_{st} \) as large as possible in (35), it is clear that:
\[
\theta^{*} = \max_{\theta, v} \left[ a_{stv}(u_{stv} - \theta) + b_{stv} - \lambda \right] \leq 0, \ \theta = 0, 1, 2, \ldots,
\]
\[
v = V, V - 1, \ldots.
\]
In other words, \( \theta^{*} \) is the minimum value of \( \theta \) such that \( \lambda = a_{stv}w_{st} + b_{stv} - a_{stv}(u_{stv} - \theta^{*}) \) \& \( b_{stv} \). Then \( w_{st} = u_{stv} - \theta^{*} \). It is seen that when \( \lambda \geq a_{stv}u_{stv} + b_{stv}, \theta^{*} = 0 \).
We have not considered the possibility that \( w_{st} \) may be less than
\[
\max \left( \left\lfloor \frac{\lambda - (\Lambda_{st-1} + 1) - b_{st}}{a_{st}} \right\rfloor + 1, l_{st} \right).
\]
The significance of this will be treated in two subsequent lemmas.

**Theorem.** The optimal returns \( x_{st}^{*}(\lambda) \) for any \( \lambda \) and \( s = 1, 2, \ldots, m, t = 1, 2, \ldots, n \), which constitute a solution to (17) are given by:
\[
x_{st}^{*}(\lambda) = \frac{\lambda - b_{stv}}{a_{stv}} = \text{integer, } 0 \leq \lambda \leq u_{stv} \text{ for some } p \in S_{stv}.
\] (36)
An exact algorithm for the concave transportation problem

\[ x^*_v(\lambda) = \begin{cases} 
0, & \lambda \leq a_{stv} + b_{stv} - 1 = \Lambda_{st-1}, \quad v \in S_{stv}; \quad s = 1, 2, \ldots, m; \\
\frac{\lambda - \Lambda_{st-1} + 1}{a_{stq}} - 1, & \Lambda_{st-1} < \lambda < \Lambda_v \\
\text{undefined}, & \lambda > \Lambda_v, \quad s = 1, 2, \ldots, m; \quad t = 1, 2, \ldots, n; \quad st \neq 11 \\
\text{for some } p \in S_{stv}, \quad s = 1, 2, \ldots, m; \quad t = 1, 2, \ldots, n; \quad st \neq 11 \\
\end{cases} \]

where \( \delta_{stv}(\lambda) = \begin{cases} 
\delta_{stv}(\lambda) \quad & \text{if } \delta_{stv}(\lambda) \text{ exists} \\
- & \text{if } \delta_{stv}(\lambda) \text{ does not exist} \\
\end{cases} \)

and \( \delta_{stv}(\lambda) = \min \left( \frac{\lambda - b_{stv}}{a_{stv}} \right) \quad v \leq p \)

Proof. Lemmas 1, 2, 4, 5.

In Lemma 5, we did not consider the possibility that \( w_{st} < x_{\min}(\lambda) \). We develop the consequences of this in the following lemmas.

**Lemma 6.** If \( w_{st} < l_{stq} \), \( \Lambda_{st-1} < \lambda < \Lambda_v \), then \( x^*_v(\lambda) \) is undefined and no solution exists.

**Proof.** From the definition of the function \( f_{st}(\cdot) \), we know that \( l_{stq} \leq x_{st} \leq u_{stv} \) and from Lemma 5, we have that \( w_{st} \leq u_{stv} \). Hence, if \( w_{st} < l_{stq} \), there can be no value of \( x_{st} \) satisfying these conditions and \( x^*_v(\lambda) \) is undefined.

**Lemma 7.** If \( x_{\min}(\lambda) = \left[ \frac{\lambda - (\Lambda_{st-1} + 1) - b_{stq}}{a_{stq}} \right] + 1 \) and if \( w_{st} < x_{\min}(\lambda) \), then \( a_{stq} > \Lambda_{st-1} + 1 \).

**Proof.** By hypothesis

\[ w_{st} < \left[ \frac{\lambda - (\Lambda_{st-1} + 1) - b_{stq}}{a_{stq}} \right] + 1 \]  \hspace{1cm} (38)

Since \( \lfloor \alpha \rfloor \leq \alpha \), we can remove the integer requirement of

\[ \left[ \frac{\lambda - (\Lambda_{st-1} + 1) - b_{stq}}{a_{stq}} \right] \]

in (38) and strengthen the inequality to obtain:

\[ w_{st} < \frac{\lambda - (\Lambda_{st-1} + 1) - b_{stq}}{a_{stq}} + 1 \]  \hspace{1cm} (39)
Since $a_{stq} > 0$, we can rewrite (39) as:

$$\lambda + a_{stq} - b_{stq} - (\Lambda_{st-1} + 1) > a_{stq}w_{st}. \tag{40}$$

By definition, when $\Lambda_{st-1} < \lambda \leq \Lambda_{st}$ we know that

$$\lambda \geq a_{st}w_{st} + b_{stq}, \quad v \geq q. \tag{41}$$

Since by Lemma 3:

$$a_{st}w_{st} + b_{stq} \geq a_{stq}w_{st} + b_{stq}, \quad v \geq q. \tag{42}$$

From (40)-(42) we see that:

$$a_{stq} - (\Lambda_{st-1} + 1) > 0. \tag{43}$$

Rewriting (43) we have:

$$a_{stq} > \Lambda_{st-1} + 1$$

which was to be proven.

**LEMMA 8.** If $a_{stq} > \Lambda_{st-1} + 1$, then $g_{stq}(\lambda)$ is not defined for $\lambda = \Lambda_{st-1} + 1 + pa_{stq} + b_{stq}, p = 0, 1, \ldots$ and $x^*_{st}(\lambda)$ is undefined.

**Proof.** The dynamic programming recursion relations are, for $v \geq q$:

$$g_{stq}(\lambda) = \min_{\lambda = \Lambda_{st-1} + 1 + pa_{stq}, p = 0, 1, \ldots} \{a_{stq}x_{st} + b_{stq} + g_{st-1}(\lambda - a_{stq}x_{st} - b_{stq})\},$$

$$s = 1, 2, \ldots, m$$

$$t = 1, 2, \ldots, n$$

$$st \neq 11$$

Since $\lambda > \Lambda_{st-1}$, we let $\lambda = \Lambda_{st-1} + 1 + pa_{stq}, p = 0, 1, \ldots$. We then have:

$$g_{stq}(\Lambda_{st-1} + 1 + pa_{stq} + b_{stq}) = \min \{a_{stq}l_{stq} + b_{stq} + g_{st-1}(\Lambda_{st-1} + 1 + pa_{stq} + b_{stq} - a_{stq}l_{stq} - b_{stq})\},$$

$$a_{stq}(l_{stq} + 1) + b_{stq} + g_{st-1}(\Lambda_{st-1} + 1 + pa_{stq} + b_{stq} - a_{stq}(l_{stq} + 1) - b_{stq}) \cdots$$

$$a_{stq}b_{stq}(\lambda) + b_{stq} + g_{st-1}(\Lambda_{st-1} + 1 + pa_{stq} + b_{stq} - a_{stq}d_{stq}(\lambda) - b_{stq})\} \tag{44}$$

where

$$\delta_{stq}(\lambda) = \min \left( \frac{[\Lambda_{st-1} + 1 + pa_{stq} + b_{stq} - b_{stq}]}{a_{stq}} \right). \tag{44}$$

Consider the terms in brackets in (44). Since $a_{stq} > \Lambda_{st-1} + 1$, all the arguments of $g_{st-1}(\cdot)$ will either exceed the limit $\Lambda_{st-1}$, and hence be undefined, or be undefined because the argument is negative. This will depend upon the relative magnitudes of $p$ and $x_{st}$. However, we will show that in no case can the value of the argument of $g_{st-1}(\cdot)$ be both less than or equal to $\Lambda_{st-1}$ and also non-negative. This follows because the general argument of $g_{st-1}(\cdot)$ is

$$\Lambda_{st-1} + 1 + a_{stq}(p - x_{st}), \quad l_{stq} \leq x_{st} \leq \delta_{stq}(\lambda)$$

**Case 1.** $p < x_{st}$

Then $p - x_{st} < 0$. Since $a_{stq} > \Lambda_{st-1}$, $\Lambda_{st-1} + 1 + a_{stq}(p - x_{st}) < 0$ and negative arguments of $g_{st-1}(\cdot)$ are not defined.

**Case 2.** $p = x_{st}$

Then $\lambda_{st-1} + 1 + a_{stq}(p - x_{st}) = \lambda_{st-1} + 1$ and $g_{st-1}(\lambda_{st-1} + 1)$ is not defined.

**Case 3.** $p > x_{st}$

Then $a_{stq}(p - x_{st}) > 0$. Therefore, $\Lambda_{st-1} + 1 + a_{stq}(p - x_{st}) > \Lambda_{st-1}$ and $g_{st-1}(\cdot)$ is not defined for this argument.

This completes the proof.
Lemmas 6–8 settle the question of the meaning of $w_{st} < x_{min}(\lambda)$ for $\lambda_{st} - 1 < \lambda \leq \lambda_{st}$. We see that this indicates that $x^*_t(x) = 1$ is undefined. Hence, the backwards recursion may be discontinued for the current value of $z_k$ and the next trial for $z_k + 1 = z_k + 1$ is begun.

The significance of the Theorem and Lemmas 6–8 is that the entire backwards recursion for the optimal values of $x^*_m(z_k), x^*_m(z_k - f_{mn}(x^*_m) x^*_m), \ldots, x^*_t(z_k - \sum_{s=1}^{n} t_s f_t(x^*_t) x^*_t)$ may be calculated, given any value of $z_k$, from equations (36) and (37) without the necessity of ever explicitly carrying out the forward calculations. The need for the storage of lengthy tables has been eliminated.

5. COMPUTATIONAL CONSIDERATIONS

The details of how the above theory should be worked into a computer code are beyond the scope of this paper. Some considerations are as follows. These are based on our examination of some hand calculations.

First, we may note that if the linear underestimator $z = \sum_{i,j} c_{ij} x_{ij}$ is “reasonably close,” i.e., the departures from linearity by the piecewise concave function are not great, it is likely that the optimal basic feasible solution obtained to (2)-(4), will also be the optimal solution to the problem using the objective function given by (1). It will be recalled that any solution we seek to (1)-(4) will be a basic feasible solution to (2)-(4). Hence, a good solution strategy is to test the optimal solution to the linear problem. If it is one of the alternate optima generated by the optimal solutions given by equations (36) and (37), it is automatically the optimal solution we seek.

A second point to be noted is that even if the solution to the linear problem is not optimal, it is very probable that the values of $x_{ij}$ obtained may be close to the optimal values in another basic feasible solution. Hence search strategies through combinations of alternate optima should embody this consideration.

A further point to be noted is that since only basic feasible solutions need be considered as candidates for optimal solutions and since there are $mn$ variables, of which at most only $m + n - 1$ will be positive in a basic feasible solution, only a small fraction of the total number of combinations of the values for each $x_{ij}$ that make up an alternate optimal solution need be tested for feasibility. Any potential solution which has more than $m + n - 1$ positive $x_{ij}$ need not be considered. This should be an important feature of the search strategy.

Lastly, many combinations of values of variables making up alternate optimal solutions can be seen to violate one or more of the constraints before the entire solution is constructed in the backwards calculation using (36) and (37).

All of the above considerations will be important elements in the construction of a large-scale computer code for the solution of concave transportation problems.

6. SUMMARY AND CONCLUSIONS

An exact method for solving a class of concave transportation problems has been presented. It is assumed that the concave cost functions can be represented as a piecewise linear concave function and that the objective function is separable. Dynamic programming methodology has been used to search candidate hypersurfaces for the optimal feasible integer solution. The explosively great storage requirements for high dimensional dynamic programming has been avoided by the development of an analytic representation of the optimal allocation at each stage as a function of the amount to be allocated.

REFERENCES