TECHNICAL REPORT NO. 40

ADJACENT EXTREME POINT ALGORITHMS FOR THE FIXED CHARGE PROBLEM
by
Warren Walker

January 30, 1968

This work was supported, in part, by a contract from the US Public Health Service (ES 00098-02)

CENTER FOR ENVIRONMENTAL QUALITY MANAGEMENT
and
DEPARTMENT OF OPERATIONS RESEARCH
CORNELL UNIVERSITY
ITHACA, NEW YORK
ABSTRACT

Three algorithms are presented for the approximate solution of fixed charge problems. Computational experience shows them to be extremely fast and to yield very good solutions.

The basic approach in all three is (1) to obtain a local optimum by using the simplex method with its "come-in" criterion changed, and (2) once at a local optimum to search for a better extreme point by jumping over adjacent extreme points to resume iterating two or three extreme points away.

Problems in which economies of scale give rise to separable piecewise-linear concave objective functions are shown to be easily formulated as fixed charge problems.
I Introduction

The fixed charge problem can be stated as

\[
\begin{align*}
\min & \quad z = \sum_{j=1}^{n} f_j(x_j) \\
\text{s.t.} & \quad Ax = b \\
& \quad x \geq 0
\end{align*}
\]

where \( f_j(x_j) = c_j x_j + k_j \delta_j \)

\[
\delta_j = \begin{cases} 
0 & \text{if } x_j = 0 \\
1 & \text{if } x_j > 0
\end{cases}
\]

This type of problem arises in many practical settings. Two of the most common of these are in warehouse or plant location, where there is a charge associated with opening the facility, and transportation problems, where there are fixed charges for transporting any goods between supply points and demand points.

If all the fixed charges \( k_j \) were zero, then problem (1) would be a linear programming problem. If some or all of the fixed charges are positive, the objective function \( z \) is concave [4, p.103]. Hirsch and Dantzig [8] and Charnes and Cooper [3, p.286] prove that the minimization of a concave functional, defined over a convex polyhedron, takes on its minimum at an extreme point. Thus, all of the methods developed so far for the solution of (1) are extreme point methods.

The fixed charge problem (1) can be written as a mixed-integer linear program [7, p.253]. Any mixed-integer linear
programming algorithm can then be used to solve it exactly. However, Gomory's algorithm [5, p.521-535] and Benders' partitioning algorithm [2] are computationally feasible only for small problems. Steinberg's branch and bound algorithm [11] requires as much as 47 minutes on an IBM 360/50 to solve a 15x30 problem. Gray's decomposition approach [6] requires an average of 16 minutes to solve a 5x7 fixed charge transportation problem and as much as 22 minutes to solve a 30-site warehouse location problem on the Burroughs B-5500. Murty [10] has developed an exact algorithm which solves (1) with all $k_{ij}=0$ to bound the total cost and then searches systematically among the extreme points adjacent to the LP optimum for the minimum total cost. As far as is known, this algorithm has yet to be tested.

Since the currently available exact algorithms generally require long computation times, a good deal of effort has been devoted to finding approximate solutions to fixed charge problems. The fixed charge transportation problem (where the A-matrix is in the form of a transportation matrix) has been investigated by Kuhn and Baumol [9] and Balinski [1]. Kuhn and Baumol suggest that an approximate solution to the problem may be obtained by forcing a highly degenerate solution. This is accomplished by making small adjustments to the right hand side (demands and supplies). The approximation is a rough one, but the computation is quite simple.

Balinski replaces the non-linear fixed charge objective function by an approximate linear objective function, and solves
the resulting problem using the standard transportation algorithm. He also finds bounds on the optimal exact solution. This approach yields a rather rough approximation, and does not work well in many cases.

Cooper and Drebes [4] and Steinberg [11] have developed approximate heuristic adjacent extreme point algorithms for the general fixed charge problem. Steinberg modifies the linear programming criterion for a vector to enter the basis, a technique also used in this paper. His algorithm will be discussed more fully below. Cooper and Drebes modify the objective function at certain stages in their algorithms, and also change the criteria for vectors to enter and leave the basis. At certain times in their calculations a vector is chosen to enter the basis with the least fixed charge of the non-basic valuables. At other times a vector is chosen to leave with the highest fixed charge in the basic set.

The computational experience reported for these methods [4,11] indicates that they will yield the optimal solution a high percentage of the time and, when not optimal, they provide a good approximation.

This paper will describe another set of adjacent extreme point algorithms which appear to be faster than those of [4] and [11], and yield the optimal solution a higher percentage of the time. The algorithms will be referred to by the name SWIFT, for Simplex WITh Forcing Trials. They are the result of continuing research into solution techniques for the fixed charge problem and, more generally, problems requiring optimization of
a concave objective function subject to linear constraints.

II Development of the SWIFT Algorithm

A. Phase 1

Let $x_B$ = vector of basis variables
$c_B$ = vector of prices for $x_B$
$k_B$ = vector of fixed charges for $x_B$
$a_j$ = $j^{th}$ column of $A$
$B$ = basis matrix
$y_j = B^{-1}a_j$ representation of $a_j$ in terms of $B$
$z_j = c_B y_j$

The algorithm consists of two distinct phases. The first phase is identical to the standard simplex procedure except that the "come-in" criterion is modified. In a linear program, the objective function will decrease if the entering vector, $x_j$, is one such that $z_j - c_j > 0$. For the fixed charge problem this criterion must be modified to account for the fixed charges. The new criterion requires non-negativity of another quantity involving the $z_j - c_j$ and the fixed charges.

Suppose that $x_j$ is to enter the basis on the next iteration. Then the leaving vector is $x_{Br}$, where $x_{Br}$ is determined by

$$
\theta_j = \min_k \left( \frac{x_{Br}^k}{y_{kj}} : y_{kj} > 0 \right) = \frac{x_{Br}^r}{y_{rj}}
$$

and $\theta_j$ is the value which $x_j$ assumes upon entering the basis.

If $\theta_j > 0$, as a result of such a basis change the objective
function is increased by \( k_j \), decreased by \( k_{B_r} \), and increased or decreased by \( \theta(z_j - c_j) \) depending upon the sign of \( z_j - c_j \). (If \( \theta_j = 0 \) the objective function remains the same.) In addition, if the choice of \( x_{B_r} \) was not unique, one or more of the basic variables which were positive will become zero. In this case, even though they remain in the basis, the objective function is reduced by their fixed costs. Conversely, it is possible that in the course of bringing \( x_j \) into the basis, some basic variables which were at a zero level will become positive. If this occurs their fixed costs must be added in to determine the new value of the objective function.

Let \[ S = \{ i \mid \frac{x_{B_i}}{y_{ij}} = \frac{x_{B_r}}{y_{rj}} \} \]

\[ T = \{ i \mid x_{B_i} = 0, y_{ij} < 0 \} \]

Then the entering vector, \( x_j \), should be chosen such that

\[ \Delta_j = k_j - k_{B_r} - \theta_j(z_j - c_j) - \sum_{i \in S} k_i + \sum_{i \in T} k_i < 0. \]

It is possible to continue iterating using criterion (3) to choose a vector to enter the basis until \( \Delta_j \geq 0 \) for all non-basic columns \( j \). However, because the objective function is concave, it is not true that when all \( \Delta_j \geq 0 \) a global minimum has been reached. Even though no adjacent extreme point will yield a smaller value of \( z \), it is still possible that some other extreme point of the convex set will be better (see [9], p.13).

This difficulty leads to phase 2 of the algorithm—a search
for a better extreme point non-adjacent to the current point. Three closely related methods have been developed for phase 2. Taken together with phase 1, which is the same for each of these methods, they constitute three heuristic algorithms for the fixed charge problem, which will be called SWIFT-1, SWIFT-2 and SWIFT-3.

B. Phase 2

At the end of phase 1, \( \Delta_j \geq 0 \) for all non-basic columns \( j \). The phase 1 solution may or may not be the optimum. In phase 2 one or more vectors will be forced into the basis, increasing the objective function, in the hope that, by continuing iterations from a new point, it will be possible to get away from the local optimum found at the end of phase 1. That is, from a local optimum we force an investigation of nearby extreme points with larger objective values which might be adjacent to points with smaller objective values.

The three phase 2 methods used in the algorithms described below differ (1) in the number of vectors forced into the basis at one time and (2) in what action is taken if a forcing attempt fails to produce a better solution.

C. The Three Algorithms

**SWIFT-1** (single forcing, non-return)

0. Find an initial feasible solution to (1).

1. Iterate with the simplex method, using criterion (3) to choose a vector to enter the basis, until \( \Delta_j \geq 0 \) for all non-basic columns \( j \).
a. Let \( x_0 \) be this phase 1 solution.

b. Let \( z_0 \) be the corresponding value of the objective function.

2. Force a currently non-basic variable, not yet tried, into the basis, yielding a new solution \( x_1 \) with objective value \( z_1 \geq z_0 \). If all non-basic variables in solution \( x_0 \) have been tried without an improvement, STOP and call \( x_0 \) the solution.

3. Iterate as in step 1 until \( \Delta_j \geq 0 \) for all non-basic columns \( j \) and a local optimum \( x_s \) is found.

   a. If \( x_s = x_1 \) (i.e. no iterating was possible), return to solution \( x_0 \). Go to step 2.

   b. If \( z_s < z_0 \) a better solution has been found. Rename this solution \( x_0 \). Go to step 2.

   c. If \( z_s \geq z_0 \), go to step 2.

A flow diagram for SWIFT-1 is shown in figure 1.

SWIFT-2 (single forcing, return)

Same as SWIFT-1 except change step 3(c) to read:

   c. If \( z_s \geq z_0 \) return to solution \( x_0 \). Go to step 2.

The flow chart for SWIFT-2 is the same as in figure 1 except at point \( B \) we have:

\[ \text{Return to basis } B_0 \]

\[ \text{Rename this solution } x_0 \text{'s objective value } z_0 \text{'s and its basis } B_0 \]
FIGURE 1: SWIFT-1

Find initial feasible solution to (1).

Use simplex method with criterion (3) until $A_{ij} > 0 \vee j \notin B$.

Call the present solution $x_0$, the objective value $z_0$, and the basis $B_0$.

(A)

Is there a column $j \notin B_0$ which has not been forced into $B_0$?

- NO

STOP.

$x_0$ is the best solution.

- YES

Force column $j$ into $B_0$. Call this solution $x_1$.

Use simplex method with criterion (3) until $A_{ij} > 0 \vee j \notin B$.

Call the present solution $x_s$, the objective value $z_s$, and the basis $B_s$.

(B)

Is $x_s = x_1$?

- NO

Return to solution $x_0$ with basis $B_0$.

- YES

Is $z_s < z_0$?

- NO

Rename this solution $x_0$, its objective value $z_0$, and its basis $B_0$.

(A)

- YES

Rename this basis $B_0$.  

(A)
SWIFT-3 (double forcing, return)

Same as SWIFT-2 except change step 2 to read:

2. Force an *untried pair* of non-basic variables from solution \( x_0 \) into the basis, yielding a new solution \( x_1 \) with objective value \( z_1 \geq z_0 \). If all pairs of non-basic variables in solution \( x_0 \) have been tried without an improvement, STOP and call \( x_0 \) the solution.

The flow chart for SWIFT-3 is the same as that for SWIFT-2 except that at point A we have:

![Flow Chart](image)

D. Steepest Descent

The criterion used to determine the vector to enter the basis in phase 1 could be changed to a steepest descent criterion. That is, choose vector \( x_j \) to enter if \( \Delta_j < 0 \) and

\[
\Delta_j = \min_{i | x_i \notin B_0} \Delta_i
\]

where \( \Delta_i \) is given by (3).

The steepest descent criterion is rarely used in solving a normal linear programming problem because it involves finding \( \theta_i \) for each non-basic column \( x_i \) which has \( z_i - c_i > 0 \). However,
the algorithm described above for the fixed charge problem requires the calculation of $\theta_i$ for at least a subset of the non-basic columns. As a result this criterion is easy to implement and does not add much time to the calculations per iteration.

III Test Results and Comparisons with Other Algorithms

In order to test their heuristic fixed charge algorithms Cooper and Drebes [4] randomly generated 290-$(5\times10)$ problems with the following properties:

$$|a_{ij}| \leq 20$$
$$1 \leq c_i \leq 20$$
$$1 \leq k_i \leq 999$$

The density of $A$ is 50%.

The optimal solutions to these problems were obtained by total enumeration. These problems and their solutions are included in Steinberg's thesis [11].

Cooper and Drebes applied their algorithms to a set of 253 of these problems. Of these, 240 were solved optimally by their algorithm MI, and 245 by their algorithm MII. Steinberg solved 235 out of 250 problems using his Heuristic One algorithm, and 255 out of 268 problems using his Heuristic Two. The SWIFT algorithms were tested on the 22 problems for which Steinberg got suboptimal solutions using his algorithms. All three of the algorithms obtained optimal solutions to the 22 problems. In 16 of the 22 optimality was attained by the end
of phase 1.

Subsequently the algorithms were tried on a random sample of 30 problems solved successfully by Steinberg. Again the optimal solution was obtained for all the problems. The optimum was reached by the end of phase 1 in 26 out of the 30 problems.

The problems were solved with and without steepest descent. As an example of the effect of steepest descent, method 2 averaged 18 iterations/problem without and 15 iterations/problem with steepest descent. Steepest descent also reduced the average number of iterations needed to reach the optimal solution by about 3 iterations (in method 2, from 10.5 iterations to 7 iterations).

Table 1 gives a comparison of the SWIFT algorithms with those of Steinberg for the 5x10 test problems.

Cooper and Drebes constructed 15x30 test problems by aggregating sets of three 5x10 problems. In these the A-matrix was formed as follows:

\[
A = \begin{bmatrix}
A_1 & 0 & 0 \\
0 & A_2 & 0 \\
0 & 0 & A_3
\end{bmatrix}
\]

where \( A_1, A_2, A_3 \) are 5x10 matrices. The optimal objective value for any of the 15x30 problems is the sum of the optimal objective values from the 5x10 problems associated with \( A_1, A_2 \) and \( A_3 \).

Six such 15x30 problems were constructed and optimal
solutions were obtained. Two of these problems were optimal by the end of phase 1. These two had A-matrices which were comprised of submatrices which had led to optimal phase 1 solutions to these 5x10 problems. The other four A-matrices contained at least one submatrix from a 5x10 problem which did not produce an optimal phase 1 solution.

The 15x30 problems were substantially harder to solve than the 5x10 problems. SWIFT-2 went from an average of 15 iterations/problem to an average of 87 iterations/problem. SWIFT-3 required as many as 392 iterations to solve one 15x30 problem, however, it had reached the optimal solution by iteration 35. The other iterations were spent searching for a better solution.

It is interesting to observe that Cooper and Drebes averaged 1200 iterations/problem (15 minutes) using their heuristics on 15x30 problems. Comparisons of algorithms for 15x30 problems are given in table 2.

Cooper and Drebes also applied their algorithm to a fixed charge problem given in [7], pages 111-113. This is a 7x13 problem. Hadley, using polygonal approximation, obtains solutions in error by 15-20 percent. Cooper and Drebes get an answer which is about 1 percent in error. All three SWIFT algorithms get the exact solution.

SWIFT-1 was tried on the fixed cost transportation problem used as an example in Balinski's paper [1]. This problem has eight warehouses supplying twelve locations and is, therefore, a 20x96 fixed charge problem. Balinski finds that the true least cost lies between $451.05 and $504.55. SWIFT-1 ended
### TABLE 1
COMPARISON OF ALGORITHMS - 5x10 PROBLEMS

<table>
<thead>
<tr>
<th>Source</th>
<th>Algorithm</th>
<th>No. Tried</th>
<th>No. Optimal</th>
<th>% Opt.</th>
<th>Avge No. Iterations per Problem</th>
<th>Avge Time per Problem/Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooper &amp; Drebes</td>
<td>MI</td>
<td>253</td>
<td>240</td>
<td>95</td>
<td>75</td>
<td>20 sec./IBM 7072</td>
</tr>
<tr>
<td></td>
<td>MII</td>
<td>253</td>
<td>245</td>
<td>97</td>
<td>100</td>
<td>&quot;</td>
</tr>
<tr>
<td></td>
<td>Both</td>
<td>253</td>
<td>250</td>
<td>99</td>
<td>175</td>
<td>&quot;</td>
</tr>
<tr>
<td>Steinberg</td>
<td>Heuristic 1</td>
<td>250</td>
<td>235</td>
<td>94</td>
<td>6.8</td>
<td>.7 sec./IBM 360/50</td>
</tr>
<tr>
<td></td>
<td>Heuristic 2</td>
<td>268</td>
<td>255</td>
<td>95</td>
<td>15.2</td>
<td>1.5 sec./IBM</td>
</tr>
<tr>
<td>Walker</td>
<td>SWIFT-1</td>
<td>52</td>
<td>52</td>
<td>100</td>
<td>23</td>
<td>5 sec./CDC 1604</td>
</tr>
<tr>
<td></td>
<td>SWIFT-1 - S.D.*</td>
<td>49</td>
<td>49</td>
<td>100</td>
<td>17</td>
<td>0.5 sec./IBM 360/65</td>
</tr>
<tr>
<td></td>
<td>SWIFT-2</td>
<td>52</td>
<td>52</td>
<td>100</td>
<td>18</td>
<td>4 sec./CDC 1604</td>
</tr>
<tr>
<td></td>
<td>SWIFT-2 - S.D.*</td>
<td>52</td>
<td>52</td>
<td>100</td>
<td>15</td>
<td>1.5 sec./CDC 1604</td>
</tr>
<tr>
<td></td>
<td>SWIFT-3</td>
<td>52</td>
<td>52</td>
<td>100</td>
<td>36</td>
<td>6 sec./CDC 1604</td>
</tr>
</tbody>
</table>

*S.D. = steepest descent

### TABLE 2
COMPARISON OF ALGORITHMS - 15x30 PROBLEMS

<table>
<thead>
<tr>
<th>Source</th>
<th>Algorithm</th>
<th>No. Tried</th>
<th>No. Optimal</th>
<th>% Opt.</th>
<th>Avge No. Iterations per Problem</th>
<th>Avge Time per Problem/Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooper &amp; Drebes</td>
<td>Both MI &amp; MII</td>
<td>70</td>
<td>63</td>
<td>90</td>
<td>1200</td>
<td>15 min./IBM 7072</td>
</tr>
<tr>
<td>Steinberg</td>
<td>Heuristic 1</td>
<td>90</td>
<td>75</td>
<td>83</td>
<td>12</td>
<td>2.16 sec./IBM 360/50</td>
</tr>
<tr>
<td></td>
<td>Heuristic 2</td>
<td>84</td>
<td>74</td>
<td>88</td>
<td>43</td>
<td>7.74 sec./IBM 360/50</td>
</tr>
<tr>
<td>Walker</td>
<td>SWIFT-2</td>
<td>5</td>
<td>5</td>
<td>100</td>
<td>86</td>
<td>18 sec./CDC 1604</td>
</tr>
<tr>
<td></td>
<td>SWIFT-3</td>
<td>2</td>
<td>2</td>
<td>100</td>
<td>370</td>
<td>1 min./CDC 1604</td>
</tr>
<tr>
<td></td>
<td>SWIFT-3 - S.D.*</td>
<td>1</td>
<td>1</td>
<td>100</td>
<td>376</td>
<td>.5 min./IBM 360/50</td>
</tr>
</tbody>
</table>
with a cost of $471.55 after 180 iterations and two minutes of CDC 1604 time. The best solution was reached after only 90 iterations.*

It is evident from these tests that computation time could be cut by as much as 50% for these algorithms if an optimality criterion could be found. This might be a very fruitful area for further research.

IV A Counter-Example

Since, when the SWIFT algorithms were used, none of the test problems produced suboptimal solutions, a simple problem was constructed for which at least one of the algorithms would produce a suboptimal solution. The following problem will not yield the optimum when solved by SWIFT-1 or SWIFT-2, but SWIFT-3 (double forcing) will give the optimal solution.

\[
\begin{align*}
\text{min} & \quad -x_1 + 3\delta_1 - x_2 + 3\delta_2 \\
\text{s.t.} & \quad -x_1 + x_2 \leq 1 \\
         & \quad x_1 - x_2 \leq 1 \\
         & \quad -3x_1 + 4x_2 \leq 5 \\
         & \quad 4x_1 - 3x_2 \leq 5 \\
\end{align*}
\]

\[
\delta_j = 0 \text{ if } x_j = 0 \\
\delta_j = 1 \text{ if } x_j > 0
\]

Although double forcing was all that was needed to reach optimality in the above problem, this will not always be the case. It appears that even in two dimensions, problems can be

*Gray [6] lists twelve fixed charge transportation problems which were solved by his algorithm. SWIFT-2 was applied to these problems. Table 3 contains a comparison of Gray's computation times with those achieved by SWIFT-2.
### TABLE 3
**FIXED-CHARGE TRANSPORTATION PROBLEM SOLUTIONS: A COMPARISON OF THE GRAY AND WALKER ALGORITHMS**

<table>
<thead>
<tr>
<th>Problem No.</th>
<th>Dimensions</th>
<th>Gray Solution</th>
<th>Time (sec) B-5500</th>
<th>Walker Solution (SWIFT-2)</th>
<th>Time (sec) IBM 360/65</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3x4</td>
<td>328.5</td>
<td>7.7</td>
<td>328.5</td>
<td>1.12</td>
</tr>
<tr>
<td>1a*</td>
<td>3x4</td>
<td>428.5</td>
<td>7.6</td>
<td>428.5</td>
<td>.42</td>
</tr>
<tr>
<td>1b†</td>
<td>3x4</td>
<td>580</td>
<td>7.8</td>
<td>578.5</td>
<td>.43</td>
</tr>
<tr>
<td>2</td>
<td>4x6</td>
<td>202</td>
<td>32.6</td>
<td>202.35</td>
<td>1.36</td>
</tr>
<tr>
<td>3</td>
<td>4x6</td>
<td>1999</td>
<td>26.3</td>
<td>1999</td>
<td>1.70</td>
</tr>
<tr>
<td>4</td>
<td>4x8</td>
<td>273</td>
<td>171.4</td>
<td>273.55</td>
<td>3.69</td>
</tr>
<tr>
<td>5</td>
<td>5x7</td>
<td>245</td>
<td>263.8</td>
<td>246.95</td>
<td>3.68</td>
</tr>
<tr>
<td>6</td>
<td>5x7</td>
<td>317</td>
<td>146.9</td>
<td>316.7</td>
<td>6.11</td>
</tr>
<tr>
<td>7</td>
<td>5x7</td>
<td>1638</td>
<td>97.0</td>
<td>1668.35</td>
<td>3.01</td>
</tr>
<tr>
<td>8</td>
<td>5x7</td>
<td>2289</td>
<td>3262.8</td>
<td>2289</td>
<td>3.89</td>
</tr>
<tr>
<td>9</td>
<td>6x8</td>
<td>314#</td>
<td>1510.1</td>
<td>283.9</td>
<td>5.94</td>
</tr>
<tr>
<td>9aΔ</td>
<td>6x8</td>
<td>2357#</td>
<td>71.4</td>
<td>2283.9</td>
<td>8.09</td>
</tr>
</tbody>
</table>

*Identical to problem 1 with 20 added to each fixed charge
†Identical to problem 1 with 50 added to each fixed charge
ΔIdentical to problem 9 with 250 added to each fixed charge
#It is assumed that this figure reflects some computational or typographical error
formulated which require any arbitrary number of forcings from some local optimum to get to an optimal solution. Obviously, the number of forcings is dependent upon the initial extreme point solution as well as the level of the fixed charges.

It should be noted that all three algorithms will yield the optimal solution to this problem if the initial basic feasible solution used at the start of the algorithm is (5,5), the optimal linear programming solution to the problem without fixed costs. It may be that such a modification of these algorithms will improve them.

V The Concave-Cost Linear Programming Problem

Economies of scale problems give rise to concave cost functions. The optimization of such functions subject to linear constraints is a branch of non-linear programming which has received considerable attention in the literature [5, p.543] [7, pp.111,123,202,305] [3, Chapter X]. The proposed algorithms for this class of problem tend to involve lengthy calculations.

However, it will be shown below that, if the functional is piecewise-linear and separable, or can be approximated by a piecewise-linear separable concave functional, the problem can be formulated as a fixed charge problem. Thus, if a computationally efficient algorithm can be found for the solution of the fixed charge problem, it can be used to solve efficiently a whole class of non-linear programming programs.

For this development we assume that any strictly concave
objective function has already been approximated by a piecewise-linear separable concave functional. Then the concave-cost linear programming problem can be stated as:

$$
\begin{array}{c}
\text{min } z = \sum_{j=1}^{n} \phi_j(x_j) \\
\text{s.t. } \sum_{j=1}^{n} a_{ij} x_j = b_i \quad i=1,2,\ldots,m \\
\quad x_j \geq 0 \quad j=1,2,\ldots,n
\end{array}
$$

where each of the $\phi_j(x_j)$ is piecewise linear and concave.

Dantzig [5, pp.482-486] treats the problem in the case where each of the $\phi_j(x_j)$ is piecewise linear and convex. He exhibits two methods of obtaining equivalent linear programs, which may then be solved by the standard simplex method.

These two methods can be applied, with difficulty, to the concave problem. They result in equivalent programs which include several 0-1 integer variables. The 0-1 variables are needed to keep segments with small slopes out of the solution until all segments with larger slopes are in the solution. Thus a 0-1 integer programming algorithm is required to use either of the two formulations.

However, a fixed charge equivalent formulation is possible. The resulting problem can then be solved by any fixed charge algorithm.

Consider $\phi_j(x_j)$, the part of the objective function corresponding to $x_j$. Suppose that $\phi_j(x_j)$ is composed of $r_j$ linear segments.
Let
\[ c_{ij} \equiv \text{slope of } i^{th} \text{ segment of } \phi_j(x_j): \]
\[ c_{1j} > c_{2j} > \ldots > c_{r_{ij}} \]
\[ f_{ij} \equiv \text{y-intercept of } i^{th} \text{ segment of } \phi_j(x_j) \]
when extended to the y-axis:
\[ 0 \leq f_{1j} < f_{2j} < \ldots < f_{r_{ij}} \]

\[ 0 \leq h_{0j}, \ldots, h_{r_{ij}} \equiv \text{break-points of } \phi_j(x_j) \text{ on the } x_j \text{ axis} \]

Using this notation \( \phi_j(x_j) \) can be represented graphically as:

Break \( x_j \) up into \( r_j \) new variables \( \Delta_{1j}, \Delta_{2j}, \ldots, \Delta_{r_{ij}} \). Associate with \( \Delta_{ij} \) the variable cost \( c_{ij} \) and the fixed cost \( f_{ij} \).
The fixed charge problem equivalent to (4) is:

\[
\begin{align*}
\min \quad & z' = \sum_{j=1}^{n} \sum_{i=1}^{r_j} \left[ c_{ij} \Delta_{ij} + f_{ij} \delta_{ij} \right] \\
\text{s.t.} \quad & \sum_{j=1}^{n} a_{kj} x_j = b_k \quad (k=1,2,...,m) \\
& x_j - \sum_{i=1}^{r_j} \Delta_{ij} = 0 \quad (j=1,2,...,n) \\
& x_j, \Delta_{ij} \geq 0 \quad \text{for all } i, j \\
& \delta_{ij} = \begin{cases} 
0 & \text{if } \Delta_{ij} = 0 \\
1 & \text{if } \Delta_{ij} > 0
\end{cases}
\end{align*}
\]

(5-1)

It should be noted that

(1) In an optimal solution, at most one \( \Delta_{ij} \) will be positive for each \( j \).

(2) No bounds need be placed on the \( \Delta_{ij} \). If a \( \Delta_{ij} \) is positive in an optimal solution its value will always fall between \( h_{i-1,j} \) and \( h_{ij} \).

The proof of observations (1) and (2) follows:

(1) Theorem: In an optimal solution, at most one \( \Delta_{ij} \) will be positive for each \( j \).

Proof: Suppose an optimal solution had more than one \( \Delta_{ij} \) positive for some \( j \). Consider any two of them, say \( \Delta_{a_j} > 0 \) and \( \Delta_{b_j} > 0 \). Without any loss of generality, let \( a < b \).

The cost of this solution is:

\[
(5-2) \quad z_1 = K + f_{a_j} + c_{a_j} \Delta_{a_j} + f_{b_j} + c_{b_j} \Delta_{b_j}
\]
Consider reducing \( \Delta_{aj} \) to 0 and increasing \( \Delta_{bj} \) by \( \Delta_{aj} \), while leaving all other variables unchanged. This is still a solution, and its cost is

\[
z_2 = K + f_{bj} + c_{bj} (\Delta_{aj} + \Delta_{bj}).
\]

Then

\[
z_2 - z_1 = c_{bj} \Delta_{aj} - f_{aj} - c_{aj} \Delta_{aj}
\]

\[= \Delta_{aj} (c_{bj} - c_{aj}) - f_{aj} \]

But, by concavity, \( c_{aj} > c_{bj} \), and, by assumption, \( f_{aj} > 0 \). Thus \( z_2 - z_1 < 0 \) or \( z_2 < z_1 \), contradicting the assumption that we had an optimal solution.

By repeated application of the above method, all but one variable for each \( j \) can be reduced to 0.

Q.E.D.

(2) Theorem: If a \( \Delta_{ij} \) is positive in an optimal solution, its value will always fall between \( h_{i-1,j} \) and \( h_{ij} \).

Proof: Consider any variable \( x_j > 0 \) in any solution to (5-1).

Suppose \( \Delta_{aj} = x_j \). If \( h_{a-1,j} \leq \Delta_{aj} \leq h_{a,j} \), the theorem is proved, so assume that (a) \( \Delta_{aj} > h_{a,j} \) or (b) \( \Delta_{aj} < h_{a-1,j} \).

Case (a): \( \Delta_{aj} > h_{aj} \): The objective function corresponding to this solution is

\[
z_a = \sum_{s \neq j} k_s + f_{aj} + c_{aj} x_j
\]

where

\[
k_s = \sum_{i=1}^{r_s} [c_{is} \Delta_{is} + f_{is} \delta_{is}]
\]
Suppose $\Delta_{a+1,j}$ were increased from 0 to $x_j$ and $\Delta_{a,j}$ were decreased from $x_j$ to 0.

The objective function for this new solution would be

\[
(5-4) \quad z_{a+1} = \sum_{s \neq j} k_s + f_{a+1,j} + c_{a+1,j} x_j
\]

Subtracting (5-3) from (5-4):

\[
(5-5) \quad z_{a+1} - z_a = (f_{a+1,j} - f_{a,j}) + (c_{a+1,j} - c_{a,j}) x_j
\]

But at breakpoint $h_{ij}$ the contributions to the objective value from $\Delta_i$ and $\Delta_{i+1}$ are the same; that is

\[
f_{ij} + c_{ij} h_{ij} = f_{i+1,j} + c_{i+1,j} h_{ij}
\]

or

\[
(5-6) \quad (f_{i+1,j} - f_{i,j}) = (c_{ij} - c_{i+1,j}) h_{ij}
\]

Substituting (5-6) into (5-5)

\[
(5-7) \quad z_{a+1} - z_a = (c_{aj} - c_{a+1,j}) h_{aj} + (c_{a+1,j} - c_{aj}) x_j
\]

\[
(5-8) \quad z_{a+1} - z_a = (c_{aj} - c_{a+1,j}) (h_{aj} - x_j)
\]

by concavity, $c_{aj} > c_{a+1,j} \Rightarrow c_{a, j} - c_{a+1,j} > 0$

by assumption, $h_{aj} < x_j \Rightarrow h_{a, j} - x_j < 0$

Thus $z_{a+1} - z_a < 0$ or $z_{a+1} < z_a$

and it would be better to have $\Delta_{a+1,j}$ at value $x_j$ than $\Delta_{a,j}$ at $x_j$. 

Case (b): $\Delta_{a,j} < h_{a-1,j}$: The objective function corresponding to this solution is (5-3).

Suppose $\Delta_{a-1,j}$ were now increased from 0 to $x_j$ and $\Delta_{a,j}$ were decreased from $x_j$ to 0.

The objective function for this new solution would be

$$z_{a-1} = \sum_{s \neq j} k_s + f_{a-1,j} + c_{a-1,j}x_j$$

Subtracting (5-3) from (5-7)

$$z_{a-1} - z_a = (f_{a-1,j} - f_{a,j}) + (c_{a-1,j} - c_{a,j})x_j$$

Using (5-6) in (5-8)

$$z_{a-1} - z_a = (c_{a,j} - c_{a-1,j})h_{a-1,j} - (c_{a,j} - c_{a-1,j})x_j$$

$$= (c_{a,j} - c_{a-1,j})(h_{a-1,j} - x_j)$$

By concavity, $c_{a,j} < c_{a-1,j} \Rightarrow c_{a,j} - c_{a-1,j} < 0$

By assumption, $h_{a-1,j} > x_j \Rightarrow h_{a-1,j} - x_j > 0$

Thus $z_{a-1} - z_a < 0$ or $z_{a-1} < z_a$

and it would be better to have $\Delta_{a-1,j}$ in the solution at value $x_j$ than $\Delta_{a,j}$ at $x_j$.

By repeated use of Cases (a) and (b) it follows that a variable $\Delta_{ij}$ will be positive in an optimal solution only if its value falls between $h_{i-1,j}$ and $h_{ij}$.

Q.E.D.
ACKNOWLEDGEMENTS

The author wishes to thank Professor Walter Lynn, Director of the Center for Environmental Quality Management, for suggesting the problem, supporting the research, and contributing many helpful suggestions during the development and testing of the algorithms. Professor Abraham Charnes of Northwestern University also contributed several suggestions for which the author is grateful. This work was supported, in part, by a grant from the U.S. Public Health Service (ES-00098).
BIBLIOGRAPHY


