Marsten, Roy E.

Working Paper

Approximate and Exact Solutions for a Class of Fixed Charge Problems

Discussion Paper, No. 10

Provided in Cooperation with:
Kellogg School of Management - Center for Mathematical Studies in Economics and Management Science, Northwestern University

Suggested Citation: Marsten, Roy E. (1972) : Approximate and Exact Solutions for a Class of Fixed Charge Problems, Discussion Paper, No. 10, Northwestern University, Kellogg School of Management, Center for Mathematical Studies in Economics and Management Science, Evanston, IL

This Version is available at:
https://hdl.handle.net/10419/220370

Discussion Paper No. 10

APPROXIMATE AND EXACT SOLUTIONS FOR A CLASS OF FIXED CHARGE PROBLEMS

by

Roy E. Marsten

August 21, 1972

This research was supported by National Science Foundation Institution Grant GU 3339.

APPROXIMATE AND EXACT SOLUTIONS FOR A CLASS OF FIXED CHARGE PROBLEMS

by

Roy E. Marsten

Northwestern University

Graves and Whinston [1968, 1970] have pioneered the application of probability theory to problems of combinatorial optimization. Their basic idea is to regard the set of feasible solutions as a discrete sample space and the objective function as a random variable defined over that sample space. The moments of the objective function, when they can be derived, are used to guide a search of the sample space for near-optimal solutions. The present paper will use this approach to construct an algorithm for a class of fixed charge problems. A near optimal solution will be found by means of a mean value search, and then the optimal solution determined by a branch and bound search.

1. The Fixed Charge Problem

The fixed charge problem that will be addressed here is a slight generalization of the form

$$(1.1)\quad \text{minimize} \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{j=1}^{n} f_j y_j$$

$$\text{subject to} \sum_{j=1}^{n} x_{ij} = 1 \qquad \text{for } i=1,\ldots,m$$

$$\sum_{i=1}^{m} x_{ij} \leq my_j \qquad \text{for } j=1,\ldots,n$$

$$x_{ij} = 0 \text{ or } 1 \qquad \text{for all } i,j$$

$$y_j = 0 \text{ or } 1 \qquad \text{for all } j$$

This problem has several practical interpretations, some of which are discussed by Spielberg [1968]. One of the most interesting is the "lockbox problem" of Ciochetto, Swanson, Lee, and Woolsey [1972]. They introduce the problem as follows:

> Companies which operate over wide geographic areas are often faced with the problem of "accounts receivable float", that is, inaccessible revenue for which investment opportunity is being lost. "Float" is a product of customer remittances which are either within the postal system or in the process of clearing the banks on which they were drawn. An effort to reduce float cost involves the location of "lock boxes" within a firm's distribution area. A "lock box" is a post office box to which remittances are sent (and a local bank which processes the checks), selected so that the time necessary to have available the funds remitted to a firm by its customers is a minimum. Maintaining a lock box is itself a somewhat costly operation. The usual banking procedure is to charge the firm a fixed fee for acting as a lock box bank, plus a variable charge determined by the number of checks processed by the bank. These expenses need not be actual "out of pocket" payments, but may be met by maintaining an appropriate "compensating balance" at the bank. These costs vary from bank to bank. Thus determining the "best" number and location of lock boxes requires balancing decreasing float costs against increasing banking costs.

To express this in the form (1.1), let A be the set of customers or customer groups and let B be the set of potential lock box sites. Then

c_{ij} = cost of assigning customer group $i \in A$ to a lock box at site $j \in B$.

f_j = fixed charge for a lock box at site $j \in B$.

$$x_{ij} = \begin{cases} 1 \text{ if customer group } i \text{ is assigned to site } j \\ 0 \text{ otherwise} \end{cases}$$

$$y_j = \begin{cases} 1 \text{ if a lock box is opened at site } j \\ 0 \text{ otherwise} \end{cases}$$

Form (1.1) also expresses the simple (uncapacitated) plant location problem if we let

x_{ij} = the fraction of the demand at city i met by a plant at site j,

c_{ij} = the cost of meeting all of the demand at city i from a plant at site j.

Since there are no capacity constraints on the plants every x_{ij} will be zero or one at the optimum. See [Efroymson and Ray,1966] or [Spielberg, 1968].

A generalization of (1.1) is obtained by supposing that for each $i \in A$ there is a non-empty set $B_i \subseteq B$ such that i must be assigned to one of the elements of B_i. Equivalently, for each $j \in B$ there is a set

$$(1.2) \quad A_j = \{i \in A \mid j \in B_i\}$$

such that only members of A_j can be assigned to j. Such a situation would arise if, for example, each customer must be assigned to a site no more than 1000 miles away. This generalization is made primarily to simplify data collection. Taking $B_i = B$ for all i will produce (1.1). The configuration for a typical problem is displayed in Figure 1. The circles are the elements of A, while the squares represent the elements of B. The domains A_j are also shown. The costs for this problem will be given later.

The problem to be addressed is therefore

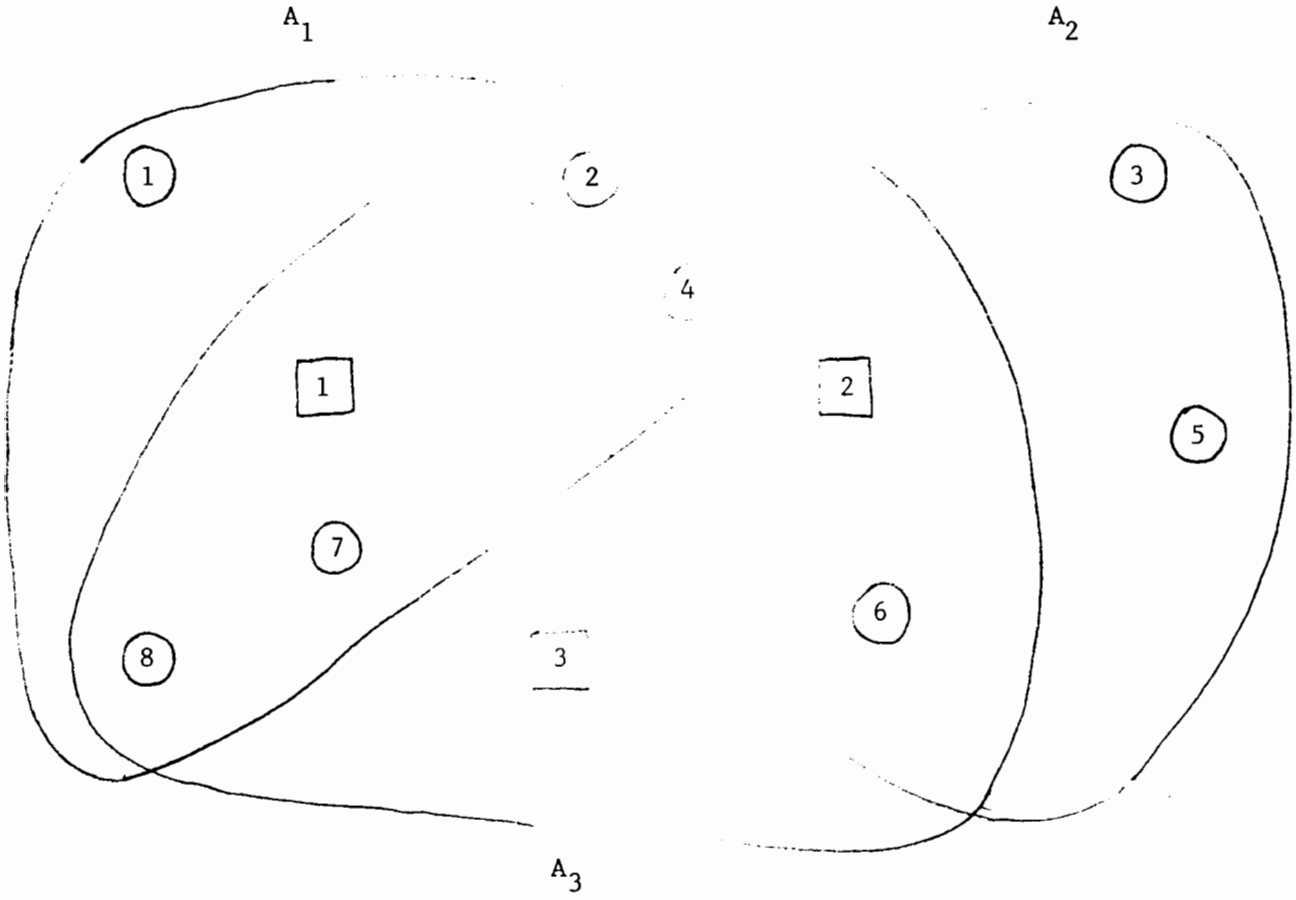

Figure 1. Typical Problem Configuration.

$$\text{(FC)} \quad \text{minimize} \sum_{i=1}^{m} \sum_{j \in B_i} c_{ij} x_{ij}$$

$$\text{subject to} \quad \sum_{j \in B_i} x_{ij} = 1 \qquad \text{for } i=1,\ldots,m$$

$$\sum_{i \in A_j} x_{ij} \leq a_j y_j \qquad \text{for } j=1,\ldots,n$$

$$x_{ij} = 0 \text{ or } 1 \qquad \text{for } i=1,\ldots,m;\ j \in B_i$$

$$y_j = 0 \text{ or } 1 \qquad \text{for } j=1,\ldots,n$$

where a_j is the number of elements in A_j. Notice that if y is fixed, the resulting subproblem is trivial. Let Y be the set of binary n-vectors and for any $y \in Y$ let us make the following definitions.

$$\text{(1.3)} \quad J(y) = \{j \in B \mid y_j = 1\}$$

$$\text{(1.4)} \quad s_i(y) = \begin{cases} \min \{c_{ij} \mid j \in B_i \cap J(y)\} \\ \qquad \text{if } B_i \cap J(y) \neq \emptyset \\ +\infty \text{ otherwise} \end{cases}$$

$$\text{(1.5)} \quad v(y) = \sum_{i=1}^{m} s_i(y)$$

Then problem (FC) is equivalent to (FC')

$$\text{(FC')} \quad \min_{y \in Y} fy + v(y)$$

in the sense that y^* is an optimal solution of (FC) if and only if it is also optimal for (FC'). Thus the x variables drop out of the problem entirely.

Problem (FC') can be solved by an implicit enumeration of the set Y. The amount of searching required to find the optimal solution can be greatly reduced if a near optimal starting solution is known. The next section is therefore devoted to finding such a starting solution.

2. Approximate Solution

The set Y contains 2^n points. Let us make it a discrete sample space by assigning probabilities according to the rule

$$(2.1) \quad P[y_j = 1] = p \qquad \text{for } j=1,\ldots,n$$

where $0 < p < 1$. The objective function of (FC') is then a random variable defined on the sample space Y. Its expected value can be derived by the following argument. Let

$$(2.2) \quad w(y) = fy + v(y) \qquad \text{for all } y \in Y$$

then

$$(2.3) \quad Ew(y) = Efy + Ev(y).$$

Clearly

$$(2.4) \quad Efy = \sum_{j=1}^{n} f_j Ey_j$$

$$= \sum_{j=1}^{n} f_j P[y_j = 1]$$

$$= p \sum_{j=1}^{n} f_j$$

and

$$(2.5) \quad Ev(y) = \sum_{i=1}^{m} Es_i(y).$$

The expected value of $s_i(y)$ can be made finite by using a sufficiently large number, PEN, in place of the $+\infty$ in (1.4). The equivalence between (FC) and

(FC') holds as long as PEN is sufficiently large. (If PEN is not large enough, then the optimal solution of (FC') will not be feasible in (FC).) PEN is a penalty incurred for each $i \in A$ that cannot be assigned to any of the opened elements of B, $B_i \cap J(y) = \emptyset$.

To compute the expected value of $s_i(y)$ let

$$(2.6) \quad Bi = \{j_1, j_2, \ldots, j_{b_i}\}$$

be arranged so that

$$(2.7) \quad c_{ij_1} \leq c_{ij_2} \leq - - - \leq c_{ij_{b_i}} < PEN$$

where b_i is the number of elements in B_i. The random variable $s_i(y)$ must assume one of the $(b_i + 1)$ values ordered in (2.7). It assumes the value c_{ij_1} if and only if $y_{j_1} = 1$. In general

$$(2.8) \quad s_i(y) = c_{ij_t} \text{ if and only if}$$

$$y_{j_1} = \ldots = y_{j_{t-1}} = 0 \text{ and } y_{j_t} = 1$$

for $t = 1, \ldots, b_i$ and

$$(2.9) \quad s_i(y) = PEN \text{ if and only if}$$

$$y_{j_1} = \ldots = y_{j_{b_i}} = 0.$$

It follows that if q = (1-p), then

$$(2.10)\quad P[s_i(y) = c_{ij_t}] = q^{t-1}p$$

for $t = 1,\ldots,b_i$ and

$$(2.11)\quad P[s_i(y) = PEN] = q^{b_i}.$$

Consequently

$$(2.12)\quad Es_i(y) = p \sum_{t=1}^{b_i} q^{t-1} c_{ij_t} + q^{b_i} \cdot PEN.$$

Combining (2.4), (2.5) and (2.12) we arrive at an expression for the mean value of w(y) over the whole sample space Y.

$$(2.13)\quad Ew(y) = p \sum_{j=1}^{n} f_j + p \sum_{i=1}^{m} \sum_{t=1}^{b_i} q^{t-1} c_{ij_t}$$

$$+ \quad PEN \sum_{i=1}^{m} q^{b_i}.$$

Notice that each set B_i must be sorted so that its costs are in increasing order.

Consider the sample problem of Figure 1. Let the costs be as in Table 1. With PEN = 1000 the simple average of w(y) over the 8 elements of Y can be computed directly as 2845. To apply (2.13) we construct two tables as in Figure 2. For each $i \in A$, row i of the C TABLE gives the cost coefficients in increasing order, while row i of the J TABLE gives the corresponding values of j. Using p = 1/2 in formula (2.13) corresponds to making the elements of Y equally likely. Hence with p = 1/2, (2.13) yields Ew(y) = 2845. Setting $p < 1/2$ ($> 1/2$) makes $Ew(y) > 2845$ (< 2845).

		B		
		1	2	3
A	1	10		
	2	20	10	30
	3		10	
	4	10	20	15
	5		20	
	6		15	10
	7	5		15
	8	10		20

$f_1 = 100,\ f_2 = 150,\ f_3 = 300$

Table 1. Costs for the Sample Problem.

C TABLE			
1.	10		
2.	10	20	30
3.	10		
4.	10	15	20
5.	20		
6.	10	15	
7.	5	15	
8.	10	20	

J TABLE			
1.	1		
2.	2	1	3
3.	2		
4.	1	3	2
5.	2		
6.	3	2	
7.	1	3	
8.	1	3	

Figure 2. Sorted Tables for the Sample Problem.

If some of the components of y are fixed, then formula (2.13) is easily modified to give the conditional expected value of w(y). Thus we can construct a solution $\bar{y}$ in n steps, at each step opening or closing that element of B that results in the minimum conditional expected value. This is the idea of the mean value search.

MEAN VALUE SEARCH

1. Set $R = B$, $k = 1$, $V = Ew(y)$
2. Compute $M(j,h) = E[w(y) \mid y_j = h]$

   for $h = 0,1$ and for all $j \in R$.
3. Choose j_k and h_k so that

$$M(j_k,h_k) = \min_{j \in R} \min_{h=0,1} M(j,h)$$

4. Set $\bar{y}_{j_k} = h_k$, $R = R - \{j_k\}$, $V = M(j_k,h_k)$.
5. Set $k = k + 1$. If $k > n$, stop.
6. Compute

$$M(j,h) = E[w(y) \mid y_{j_1} = h_1, \ldots, y_{j_{k-1}} = h_{k-1}, y_j = h]$$

   for $h = 0,1$ and all $j \in R$.

   Then go to 3.

At each step in the construction of $\bar{y}$ we have

$$M(j,0)\, q + M(j,1)p = V \tag{2.14}$$

for all $j \in R$ so that

$$M(j,0) = [V - M(j,1)p]/q. \tag{2.15}$$

Therefore all that is needed is a formula for M(j,1). Consider the first step. If k = 1 and r ∈ R we have

$$(2.16)\quad M(r,1) = E[w(y) \mid y_r = 1].$$

If $i \notin A_r$, then

$$(2.17)\quad E[s_i(y) \mid y_r = 1] = Es_i(y).$$

If $i \in A_r$, then $y_r = 1$ means that $B_i \cap J(y)$ is non-empty and that i does not have to be assigned to any j with $c_{ij} > c_{ir}$. If r is the b^{th} entry in row i of the J TABLE, then

$$(2.18)\quad E[s_i(y) \mid y_r = 1] = p \sum_{t=1}^{b-1} q^{t-1} c_{ij_t} + q^{b-1} c_{ir}.$$

Finally

$$(2.19)\quad E[fy \mid y_r = 1] = f_r + p \sum_{j \neq r} f_j$$

So M(r,1) can be expressed as

$$(2.20)\quad M(r,1) = f_r + p \sum_{j \neq r} f_j + \sum_{i \notin A_r} Es_i(y) + \sum_{i \in A_r} E[s_i(y) \mid y_r = 1]$$

where the appropriate substitutions are made from (2.12) and (2.18). The extension to the later steps k= 2,...,n is straightforward.

The mean value search for the sample problem is shown in Figure 3. The tabulated values of M(j,h) are shown for k = 1,2,3. The calculations were made with p = 1/2 and PEN = 1000. The solution $\bar{y}$ constructed is in fact optimal.

First step, k = 1

		h = 0	h = 1	
	1	4041 1/4	1648 3/4	
R	2	4252 1/2	1437 1/2*	$\bar{y}_2 = 1$
	3	3680	2010	

Second step, k = 2

		h = 0	h = 1	
	1	2387 1/2	487 1/2*	$\bar{y}_1 = 1$
R				
	3	1782 1/2	1092 1/2	

Third step, k = 3

		h = 0	h = 1	
R	3	340 *	635	$\bar{y}_3 = 0$

Figure 3. Mean Value Search for the Sample Problem.

3. Exact Solution

In this section a branch and bound algorithm for finding the exact optimal solution of problem (FC') will be presented. To begin, let the set B be sorted according to the order in which its elements were selected by the mean value search, and then renumbered from 1 to n. In the example, elements 2, 1, and 3 of B would be renumbered as 1, 2, and 3 respectively. This will force the branch and bound search to use the same branching order as the mean value search. This is not essential but appears to give consistently good results.

For $1 \leq k \leq m$, let Y_k denote the set of partial solutions that have $y_1,\ldots,y_k$ fixed and $y_{k+1},\ldots,y_n$ still free. Let Y_o contain the null solution with no components fixed. A partial solution will also be called a candidate. If $z \in Y_k$, the set of possible completions of z is denoted C(z) and defined as

$$(3.1) \quad C(z) = \{y \in Y \mid y_1 = z_1,\ldots,y_k = z_k\}.$$

The lower bound functional must be defined for every partial solution z and must satisfy

$$(3.2) \quad \text{BOUND}(z) \leq w(y) \quad \text{for all } y \in C(z)$$

and

$$(3.3) \quad \text{BOUND}(z) = w(z) \quad \text{if } z \in Y_n = Y.$$

Any such bounding functional can be used. The one employed in this study will be derived below. All of the pending candidates are kept in the candidate list, each with its associated bound. We shall always select the candidate with

the minimum bound as the next one for consideration.

BRANCH AND BOUND SEARCH

1. Set $V = w(\bar{y})$ where $\bar{y}$ is the solution found by the mean value search. Set $y^* = \bar{y}$.
2. Place the null solution $\emptyset$ in the candidate list, with BOUND $(\emptyset) = 0$.
3. If the candidate list is empty, stop.
4. Select from the candidate list the candidate with the minimum associated bound. Call it z and suppose that it belongs to Y_k. If BOUND$(z) \geq V$, stop.
5. Set $z_j^o = z_j$ for $j = 1,\ldots,k$ and $z_{k+1}^o = 0$.
6. If BOUND $(z^o) \geq V$, go to 9.
7. If $k + 1 = n$, set $V = $ BOUND(z^o) and $y^* = z^o$, then go to 9.
8. Place z^o and BOUND(z^o) in the candidate list.
9. Set $z_j^1 = z_j$ for $j = 1,\ldots,k$ and $z_{k+1}^1 = 1$.
10. If BOUND $(z^1) \geq V$, go to 3.
11. If $k + 1 = n$, set $V = $ BOUND(Z^1) and $y^* = z^1$, then go to 3.
12. Place z^1 and BOUND(z^1) in the candidate list, then go to 3.

Upon termination, y^* is the optimal solution of problem (FC') and therefore of (FC) as well.

The approximate solution $\bar{y}$ is used in two ways. It is used at Step 1 to provide an upper bound on the optimal value $w(y^*)$. The amount of searching required to find y^* is very directly related to the gap between $w(\bar{y})$ and $w(y^*)$, as can be seen in Steps 6 and 10. When a partial solution z is extended, as in Steps 5 and 9, there is a choice as to which component of z to fix next.

These choices are resolved by always fixing the components in the same order as in the construction of $\bar{y}$.

The lower bound functional used to obtain the computational results of the next section is constructed as follows. For any partial solution $z \in Y_k$, let

$$(3.4) \quad J(z) = \{1 \leq j \leq k \mid y_j = 1\}$$

$$(3.5) \quad O(z) = \{1 \leq j \leq k \mid y_j = 0\}$$

and

$$(3.6) \quad s_i(z) = \begin{cases} \min \{c_{ij} \mid j \in B_i - O(z)\} \\ \qquad \text{if } B_i - O(z) \neq \emptyset \\ \text{PEN otherwise} \end{cases}$$

Thus $s_i(z)$ is a lower bound on the cost of assigning element $i \in A$. There may be some $i \in A$ for which no member of B_i has yet been opened. Let i^* be any such i, say the first one,

$$(3.7) \quad i^* = \min \{i \mid B_i \subseteq B - J(z)\}.$$

Any completion of z that does not incur the penalty for i^* must open at least one element of B_{i^*}. Therefore let

$$(3.8) \quad t(z) = \min \{f_j \mid j \in B_{i^*}\}$$

and combine (3.6) with (3.8) to give

$$(3.9) \quad \text{BOUND}(z) = \sum_{j \in J(z)} f_j + \sum_{i=1}^{m} s_i(z) + t(z).$$

It is easy to show that this BOUND satisfies (3.2) and (3.3). A somewhat stronger bound could be obtained by defining A^* as the set of all i^* satisfying (3.7) and replacing (3.8) by

$$(3.10) \quad t(z) = \max_{i \in A^*} \min_{j \in B_i} f_j .$$

4. Computational Results

Some test problems of the simple plant location type have been constructed and solved. The set A consists of 100 major U.S. cities. These are listed in Table 2. In each problem the set B is some subset of these 100 cities. A fixed cost f_j is assigned to each element of B and C_{ij}'s are computed as

$$(4.1) \quad c_{ij} = Kw_i d_{ij}$$

where

w_i = the population of city i (1960 census) divided by 1000,

d_{ij} = the approximate air distance in miles between city i and city j,

K = a constant (.0134)

The problems were constructed in this way so as to resemble, very crudely, real data. The cost of serving city i from a plant in city j is assumed to be proportional to the demand in city i and the inter-city distance. The demand in each city is further assumed proportional to its population.

The results are presented in Table 3. The column headings used are

ID - problem number.

m - number of points in A, i.e. demand points.

n - number of points in B, i.e. potential plant sites.

α - the solution found by the mean value search was within α% of being optimal.

T_{mv} - the time, in seconds, required for the mean value search.

T_{bb} - the time, in seconds, required for the branch and bound search.

CAND - the number of partial solutions placed in the candidate list during the branch and bound search.

All times reported are for a CDC6400 computer. T_{mv} includes the initial sorting to set up the C TABLE and J TABLE. All of the problems were run with p = 1/4 and PEN = 10 million.

In every case the branch and bound search was started from the near optimal solution found by the mean value search. The number α is defined by

$$(4.2) \quad \alpha = \frac{w(\bar{y}) - w(y^*)}{w(y^*)} \cdot 100$$

In problems 4 and 7 the mean value search actually found the optimal solution (α = 0%). The poorest starting solution (α = 9%) was for problem 3. In this case $\bar{y}$ had 3 plants open while y^* had only 2.

In problems 1 and 2 the overlapping domains A_j were drawn in by hand. In the remainder of the problems each A_j contains every city within 1000 miles of the site j.

To illustrate the solutions found by the mean value search, consider problem 2. The potential sites and their fixed costs are given in Table 4. The solution $\bar{y}$ found by the mean value search is shown in Figure 4. The optimal solution, y^*, appears in Figure 5. Solution $\bar{y}$ has cost \$5,345,087 while the cost for y^* is \$5,240,727.

In spite of the near optimal starting solution, the time required for the branch-and-bound search is increasing exponentially as the number of potential sites increases. Some of the test problems were repeated with a

stronger lower bound functional, that of Efroymson and Ray [1966]. While stronger, their bound takes longer to compute and the net result was an increase in running time. Problem 8, for example, generated only 286 candidates but took 13.3 seconds. Another way to attempt to improve performance would be with a strategy for choosing the branching order. The essential difficulty, however, is that the branch and bound search must account for every path, whereas the mean value search need only choose one path.

1. Seattle, Washington
2. Spokane, Washington
3. Portland, Oregon
4. Sacramento, California
5. San Francisco, California
6. Los Angeles, California
7. San Diego, California
8. Reno, Nevada
9. Las Vegas, Nevada
10. Boise, Idaho
11. Eugene, Oregon
12. Helena, Montana
13. Butte, Montana
14. Salt Lake City, Utah
15. Flagstaff, Arizona
16. Phoenix, Arizona
17. Cheyenne, Wyoming
18. Denver, Colorado
19. Pueblo, Colorado
20. Albuquerque, New Mexico
21. El Paso, Texas
22. Amarillo, Texas
23. Ft. Worth, Texas
24. Dallas, Texas
25. San Antonio, Texas
26. Houston, Texas
27. Bismark, North Dakota
28. Fargo, North Dakota
29. Pierre, South Dakota
30. Sioux Falls, South Dakota
31. Grand Island, Nebraska
32. Lincoln, Nebraska
33. Omaha, Nebraska
34. Topeka, Kansas
35. Wichita, Kansas
36. Tulsa, Oklahoma
37. Oklahoma City, Oklahoma
38. Duluth, Minnesota
39. Minneapolis, Minnesota
40. Des Moines, Iowa
41. Cedar Rapids, Iowa
42. Kansas City, Missouri
43. St. Louis, Missouri
44. Little Rock, Arkansas
45. Shreveport, Louisiana
46. New Orleans, Louisiana
47. Green Bay, Wisconsin
48. Madison, Wisconsin
49. Milwaukee, Wisconsin
50. Chicago, Illinois
51. Peoria, Illinoia
52. Memphis, Tennessee
53. Jackson, Mississippi
54. Gary, Indiana
55. Indianapolis, Indiana
56. Louisville, Kentucky
57. Nashville, Tennessee
58. Knoxville, Tennessee
59. Chattanooga, Tennessee
60. Birmingham, Alabama
61. Montgomery, Alabama
62. Mobile, Alabama
63. Atlanta, Georgia
64. Columbus, Georgia
65. Tallahassee, Florida
66. Jacksonville, Florida
67. Tamps, Florida
68. Ft. Myers, Florida
69. Miami, Florida
70. Cincinnati, Ohio
71. Columbus, Ohio
72. Akron, Ohio
73. Cleveland, Ohio
74. Pittsburgh, Pennsylvania
75. Charleston, West Virginia
76. Roanoke, Virginia
77. Winston-Salem, North Carolina
78. Columbia, South Carolina
79. Charleston, South Carolina
80. Charlotte, North Carolina
81. Richmond, Virginia
82. Baltimore, Maryland
83. Washington, D. C.
84. Wilmington, Delaware
85. Philadelphia, Pennsylvania
86. Newark, New Jersey
87. Buffalo, New York
88. Hartford, Connecticut
89. Syracuse, New York
90. New York, New York
91. Providence, Rhode Island
92. Boston, Massachusetts
93. Montpelier, Vermont
94. Concord, New Hampshire
95. Bangor, Maine
96. Detroit, Michigan
97. Lansing, Michigan
98. Long Beach, California
99. Santa Barbara, California
100. San Jose, California

Table 2. Cities Used in the Test Problems.

ID	m	n	α	Tmv	Tbb	CAND
1	100	8	2%	0.9	0.6	37
2	100	9	2%	1.0	0.8	50
3	100	9	9%	1.5	2.3	123
4	100	10	0%	1.9	3.4	122
5	100	11	1%	2.3	15.2	448
6	100	12	1%	2.7	29.3	701
7	100	10	0%	1.7	3.3	122
8	100	11	2%	2.1	12.4	415
9	100	12	2%	2.8	30.6	739
10	100	16	-	4.9	-	-

Table 3. Computational Results

Site	Fixed Cost (in millions)
Los Angeles	1.00
San Francisco	1.50
Chicago	1.00
Atlanta	.60
Denver	.60
New York	2.00
Dallas	.75
Washington	1.20
Boise	.60

Table 4. Fixed Costs for Problem 2

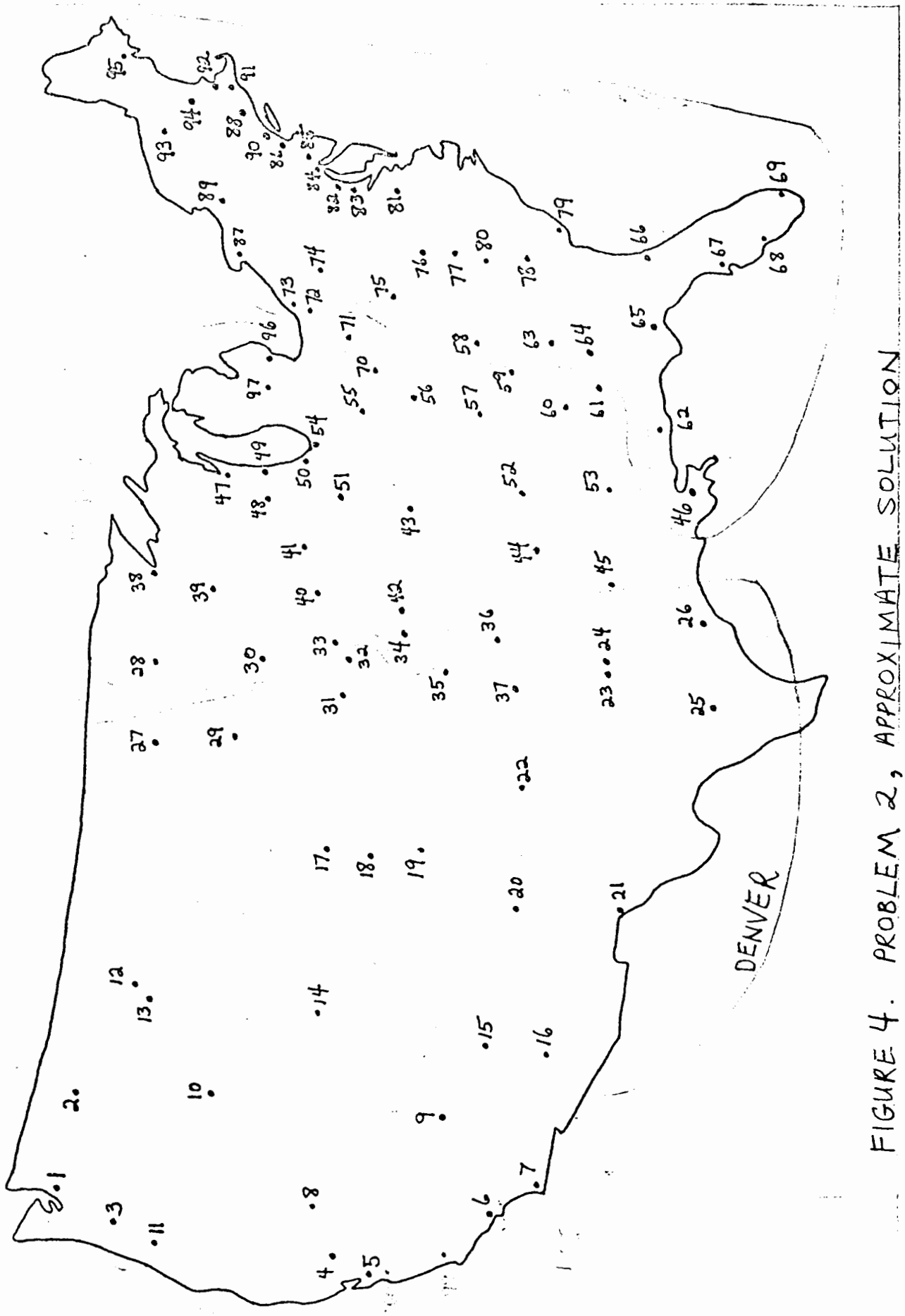

FIGURE 4. PROBLEM 2, APPROXIMATE SOLUTION

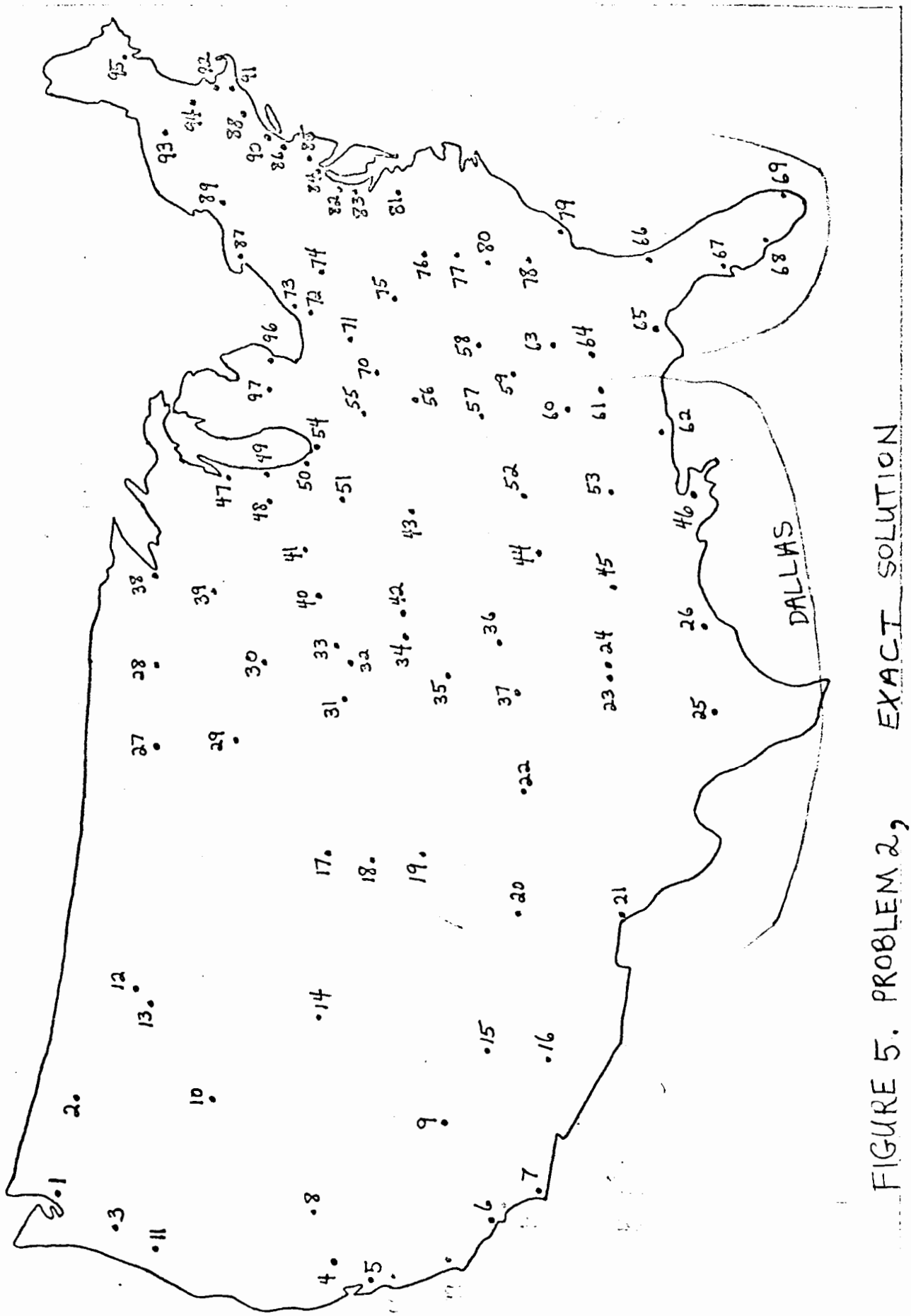

FIGURE 5. PROBLEM 2, EXACT SOLUTION

REFERENCES

Ciochetto, F.F., H.S. Swanson, J.R. Lee, and R.E.D. Woolsey, (1972), "The Lockbox Problem and Some Startling But True Computational Results for Large Scale Systems," Institute for Operations Research, Colorado School of Mines, Golden, Colorado. (Presented at the 41st National ORSA Meeting, New Orleans, April 26-28.)

Efroymson, M.A. and T.L. Ray, (1966), "A Branch and Bound Algorithm for Plant Location," Operations Research, 14, 3, 361-368.

Graves, G.W. and A.B. Whinston, (1968), "A New Approach to Discrete Mathematical Programming," Management Science, 15, 3, 177-190.

Graves, G.W. and A.B. Whinston, (1970), "An Algorithm for the Quadratic Assignment Problem," Management Science, 16, 7, 453-471.

Spielberg, K.,(1963), "Enumerative Methods for Integer and Mixed Integer Programming," Report No. 320-2938, IBM New York Scientific Center, 590 Madison Ave., New York, New York, March.