J. Operations Research Soc. of Japan
Vol. 13, No. 2, October 1970

COMPUTATIONAL EXPERIENCE ON ZERO-ONE PROGRAMMING APPROACH TO VARIOUS COMBINATORIAL PROBLEMS

SAID ASHOUR AND A. R. CHAR

Kansas State University
Manhattan, Kansas

(Received October 26, 1970)

Abstract

The combinatorial problems deal with the study of the arrangement of elements into sets. Various combinatorial problems such as shop scheduling, assembly-line balancing, delivery, traveling salesman, capital allocation, and fixed-charge problems are formulated as zero-one programming models. Sixty-five problems of various types, ranging from simple to rather difficult, are tried by two different codes, namely, pseudo-boolean and adaptive binary codes. A brief description of these codes is included. The computational results of running these problems are reported. The computational difficulties in using these codes are also included.

Introduction

The combinatorial problem is concerned with the study of the arrangement of elements into sets. The elements are usually finite in number, and the arrangement is restricted by certain boundary conditions

imposed by the particular problem under investigation. In various combinatorial problems such as shop scheduling, assembly-line balancing, delivery, traveling salesman, capital allocation, and fixed-charge problems, a given objective is to be optimized subject to a set of constraints arising due to the characteristics of the problem. Because the number of combinations increases non-linearly, direct search is not practically feasible except for very small problems. Hence method have to be devised to limit the search to a smaller subset of all solutions. In real situations, all the elements are integers and therefore the solution obtained must be integer-valued. Thus these problems can be formulated as integer programming models so that the results are integers. On the other hand, some of these variables are limited to either zero or one, and thus the problem can be formulated as a zero-one programming model. In fact, any integer linear programming problem can usually be converted into zero-one programming problem by using either simple expansion technique [12] or Balas binary device [19].

If the solution of a linear programming problem does not have the required integer property then integer constraints have to be incorporated. Numerous algorithms have been proposed for the solution of general integer linear programming models. These algorithms can be broadly divided into four classes according to the method employed: (1) algebraic approach, (2) combinatorial approach, (3) enumerative approach, and (4) heuristic approach.

First, the algebraic approach is based on methods which generate new constraints, called cuts or cutting planes so as to restrict the solution space without eliminating any feasible integer points. Second, the combinatorial approach is the method which is combinatorial in nature for which algebraic rather than exponential bounds are available for the number of steps required to solve a problem. Third, the enumerative approach is the method of search over all possible solutions which limit the extent of search. Finally, the heuristic approach refers to collection of heuristic rules for obtaining local optimal solutions utilizing computers.

Considerable research has been done on developing efficient integer and zero-one programming algorithms. Reviews by Beale [7] and Balinsky [4, 5] provide an excellent coverage of the available literature. These reviews cover various important algorithms classified according to the above outlined scheme.

Theoretically, each zero-one programming problem can be solved. In practice, however, several difficulties arise which make attempts to obtain solutions to zero-one programming models very unpredictable. One difficulty is the control of computer round-off error. The magnitude of the round-off error can sometimes be cotrolled, but it is very time-consuming. Other difficulty is the size of the computer storage. This would limit the size of problems (in terms of the number of variables and constraints) to be solved. A third difficulty is in the amount of time required to actually solve a given zero-one programming problem. Although the computational time is known to be finite, it may still be impractically long even when fastest available computers are used. Because of these difficulties, some practical problems are considered unsolvable. Furthermore, there exist problems which is difficult to solve using a certain zero-one programming code, whereas they may be easier to solve using another code. Generally, no one can predict such situations. It is obvious that the computional time required to solve a problem depends upon the characteristics of both the computer and code being used.

Because of the complexities involved in such situations, this paper has two-fold. First, a number of combinatorial problems are formulated as zero-one programming models. These are shop scheduling, assembly-line balancing, delivery, traveling salesman, capital allocation, and fixed-charge problems. These problems have been selected to show different facets of the generalized zero-one programming problem. Second, two zero-one programming codes, namely pseudo-boolean code and adaptive binary code are used in solving sixty-five problems of various types. Each of these two codes contains some features which it was hoped

would help alleviate the computational time problem. Some empirical information about the behavior of these two codes are obtained through the computational experiments conducted. The properties of the codes being investigated will be described briefly.

A pseudo-boolean algorithm proposed by Hammer and Rudeanu [15] is used to solve the various zero-one programming problems. The algorithm manes use of the properties of pseudo-boolean functions. A pseudo-boolean function may be defined as a real-valued function with zero-one variables. A pseudo-boolean program is a procedure to optimize a psedo-boolean function. The code uses a set of rules dependent on the properties of pseudo-boolean functions. These properties represent three cases: (1) when some of the variables are fixed; (2) when there is no solution; and (3) when equation on inequality is redundant. Using a branching and bounding procedure the search of all the branches is avoided. Improved results at each successive trial are utilized to improve the convergence to the optimum value. This algorithm was coded by Char [11] for the IBM 360/50 computer. The maximum number of variables and constraints are 60 and 25, respectively. However, the capacity of the code can be increased by making changes in the dimension statements.

The adaptive binary algorithm has been proposed and coded by Salkin and Spielberg [20]. The basic approach of this algorithm is a zero-one search with a dynamic origin technique, augumented by a set of auxiliary techniques invoked at each node of the each search tree. This code considers an adaptive device which allows the search to commence, and restart at a general search origin. The code is set up to treat problems with a maximum of 150 variables and 50 constraints. However, it also can be easily modified to accomodate larger problems by simple changes in dimension statements.

Combinatorial Problems as Zero-one Programming Models

As mentioned earlier, shop scheduling, assembly-line balancing,

delivery, traveling salesman, capital allocation and fixed-charge problems are different types of combinatorial problems. Since the solutions obtained must be integer-valued, these problems can be formulated as integer programming models. By the proper utilization of zero-one variables, these problems can be solved by a zero-one computer code. This section describes the formulations of shop scheduling, assembly-line balancing, delivery, traveling salesman, capital allocation and fixed-charge problems as zero-one programming models. A sample problem of each type is presented and the associated solution discussed.

Shop scheduling problem

The shop schedling problem in its simplest form consists of J jobs to be performed on M machines. Each job has a number of operations to be performed on the various machines in a prespecified machine ordering. It is required to determine a feasible sequence which results in the minimum completion time. A complete set of assumptions is provided by Ashour [2, 3].

The objective function and constraints are linear and therefore linear programming formulation provides a suitable approach. Since the results must be integers, integer linear programming is necessary. At present there exist three such formulations [9, 17, 18]. The following notation is used in the formulation.

J total number of jobs

j job designation, $j=1, 2, \cdots, J$

j_k job j in sequence position $k, k=1, 2\cdots, J$

M total number of machines ($M=3$ in this case)

m machine designation, $m=1, 2, 3$

$t_{j_k m}$ processing time of job j_k on machine m

$u_{j_k m}$ waiting time of job in sequence position k between machines m and $m+1$

$v_{j_k m}$ idle time on machine m between jobs in sequence position k and $k+1$

X_{jk} zero-one variable having a value one if job j is scheduled in sequence position k, zero otherwise

$X_{.k}$ a column vector $[X_{1k}, X_{2k}, \cdots, X_{Jk}]$

P_m row vector of integer processing times for jobs $1, 2, \cdots, J$ on machine m

The three-machine flow-shop problem is distinguished by the fact that, without loss of optimality, the search may be confined to schedules which sequence the J jobs in the same order on all three machines.

The constraints are given such that

1. A job j is assigned to the sequence position k.

$$\sum_{j=1}^{J} X_{jk}=1, \qquad k=1, 2, \cdots, J$$

2. One of the sequence positions is assigned to job j.

$$\sum_{k=1}^{J} X_{jk}=1, \qquad j=1, 2, \cdots, J$$

3. A job is not processed on two machines simultaneously and a machine does not process two jobs at once.

$$v_{jk2}+P_2X_{.k+1}+u_{jk+12}-u_{jk2}-P_3X_{.k}-v_{jk3}=0, k=1, 2, \cdots, J-1,$$

and

$$P_1X_{.k+1}+u_{jk+11}-u_{jk1}-P_2X_{.k}-v_{jk2}=0, k=1, 2, \cdots, J-1.$$

It has been shown by Johnson [16] and Bellman [8] that minimizing the total time span to complete all items is equivalent to minimizing the idle time on machine 3. Hence this formulation suggests the minimization of the following objective function:

$$f=[P_2+P_3]X_{.1}+\sum_{k=1}^{J-1} v_{jk3}.$$

In such a formation, the total number of variables is $\{J^2+4(j-1)\}$ and the number of constraints becomes $(4J-3)$. The integer valued

variables u_{j_k} and v_{j_k} are converted into zero-one variables using Balas binary technique [19].

The following sample problem will illustrate the above formulation. Consider a (2×3) flow shop problem having the following machine ordering and processing time matrix. It is required to minimize the total processing time.

$$M=\begin{pmatrix}11 & 12 & 13\\ 21 & 22 & 23\end{pmatrix} \qquad T=\begin{pmatrix}2 & 1 & 4\\ 1 & 5 & 3\end{pmatrix}$$

The solution is as follows:
The objective function is to minimize

$$f=3X_{1_1}+6X_{2_1}+v_{j_13}.$$

Subject to the following constraints:

1. One of the job j is assigned to the sequence position k.

$$X_{1_1}+X_{2_1}=1$$

$$X_{1_2}+X_{2_2}=1$$

2. One of the sequence position is assigned to a job j.

$$X_{1_1}+X_{1_2}=1$$

$$X_{2_1}+X_{2_2}=1$$

In the above four equations, one equation is redundant and therefore can be dropped.

3. A job j is not processed on two machines simultaneously and a machine m does not process two jobs at once.

$$4X_{1_1}+3X_{2_1}-X_{1_2}-5X_{2_2}-v_{j_12}+v_{j_13}-u_{j_22}=0$$

$$X_{1_1}+5X_{2_1}-2X_{1_2}-X_{2_2}+v_{j_12}-u_{j_21}=0$$

Substituting the original variables with zero-one variables $x_j, j=1, 2, \cdots, 8$, we get

$$x_1=X_{1_1},\ x_2=X_{2_1},\ x_3=X_{1_2},\ x_4=X_{2_2}$$

$$x_5=v_{j_12},\ x_6=v_{j_13},\ x_7=u_{j_21},\ x_8=u_{j_22}$$

Thus, the problem reduces to the following:
minimize

$$f=3x_1+6x_2+x_6$$

subject to

$$4x_1+3x_2-x_3-5x_4-x_5+x_6-x_8=0$$

$$x_1+5x_2-2x_3-x_4-x_5-x_7=0$$

$$x_1+x_3=1$$

$$x_2+x_4=1$$

$$x_1+x_2=1$$

and

$$x_j=0 \text{ or } 1, \qquad j=1, 2, \cdots, 8.$$

The total number of zero-one variables is 8 and the number of constraints is 5. The solution is given by

$$x_1=1,\ x_2=0,\ x_3=0,\ x_4=1,$$

$$x_5=0,\ x_6=1,\ x_7=0,\ x_8=0$$

and the minimum $f=f^*=4$. Thus the completion time is the processing time of 2 jobs on machine 3 plus the value of f^*, or $4+3+4=11$. The variables $x_1=x_{1_1}=1$ and $x_4=x_{2_2}=1$ indicate that the optimal sequence, $S^*=\{1, 2\}$.

Assembly-line balancing problem

An assembly-line consists of a number of work stations. To assemble a product, a number of tasks must be performed subject to certain sequencing requirements concerning the order in which they are performed. Given a cycle time, the assembly-line balancing problem

consists of minimizing the number of work stations. Following Bowman [9], following notation is used:

K total number of work stations

J total number of tasks

X_j the initial time when task j is started
$j=1, 2, \cdots, J$

I_{cd} $\begin{cases} 1, & \text{if task } c \text{ precedes task } d \\ 0, & \text{otherwise} \end{cases}$

T maximum clock time a product takes to come out of the assembly-line

c cycle time

t_j processing time for task j, $j=1, 2, \cdots, J$

τ number of time units the product is on the assembly-line

u_j integer-valued variable which can take any value from 0 to X_J, $j=1, 2, \cdots, J$

The constraints are given such that

1. Each task is performed in accordance with the ordering requirements.

$$X_j+t_j \leq X_{j+1}, \qquad j=1, 2, \cdots, J-1.$$

2. Each work station can take up a task only after it leaves the previous station.

$$(T+t_{j+1})I_{j(j+1)}+(X_j-X_{j+1}) \geq t_{j+1}$$

and

$$(T+t_j)(1-I_{j(j+1)})+(X_{j+1}-X_j) \geq t_j, \qquad j=1, 2, \cdots, J-1.$$

3. Each work station should not be overloaded and the tasks must be completed before being passed on to the next station.

$$X_j+t_j \leq cu_j+c$$

and

$$X_j \geq cu_j, \qquad j=1,2,\cdots,J.$$

4. All operations are over within the total completion time with no followers in a specified ordering.

$$X_s+t_s \leq \tau \qquad \text{for each } s,$$

where s is a set of stations without any succeeding stations.

The objective of minimizing the number of work stations is to distribute the work load uniformly on all work stations. This will reduce the number of time units the product is on the assembly-line. Hence the objective function becomes the minimization of

$$z=\tau.$$

The formulation utilizes $2J+1$ intege-valued variables and about J zero-one variables (the exact number depends on the ordering requirements). The total number of constraints is about $5J$ (again the exact number depends on the ordering requirements).

The following sample problem illustrates the formulation of the assembly-line balancing problem as an integer programming problem. Consider an assembly-line as shown in Figure 1. It is required to reduce the number of time units the product is on the assembly-line. The objective is to minimize the total number of time units the product is on the assembly line. Hence the objective function is given such that minimize

$$z=\tau.$$

The constraints are such that

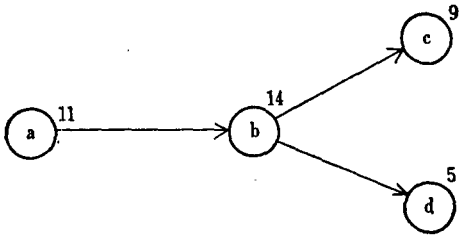
1. Each task is done in accordance with the ordering requirements.

$$X_a+11 \leq X_b$$

$$X_b+14 \leq X_c$$

$$X_b \quad 14 \leq X_d$$

2. Each work station can take up a task only after it leaves the



Station	a	b	c	d
Initial time	1−20	21−40	41−60	61−80

Fig. 1. Order Position for Sample Problem

previous station.

$$(80+14)I_{ab}+(X_a-X_b)\geq 14$$

$$(80+11)(1-I_{ab})+(X_b-X_a)\geq 11$$

$$(80+9)I_{bc}+(X_b-X_c)\geq 9$$

$$(80+14)(1-I_{bc})+(X_c-X_b)\geq 14$$

$$(80+5)I_{bd}+(X_b-X_d)\geq 5$$

$$(80+14)(1-I_{bd})+(X_d-X_b)\geq 14$$

3. Each work station is not overloaded and the tasks must be completed before being passed on to the next station.

$$X_a+11\leq 20u_a+20$$

$$X_a \geq 20u_a$$

$$X_b+14\leq 20u_b+20$$

$$X_b \geq 20u_b$$

$$X_c+9\leq 20u_c+20$$

$$X_c \geq 20u_c$$

$$X_d+5\leq 20u_d+20$$

$$X_d \geq 20u_d$$

4. All operations are completed within the total completion time with no followers in a required ordering.

$$X_c+9\leq\tau$$

$$X_d+5\leq\tau$$

This problem utilizes 9 integer-valued variables and 3 zero-one variables. The total number of constraints is 19. The integer-valued variables are converted to zero-one variables using Balas binary techniques [19] in which 7 zero-one variables are used for each of the integer-valued variables X_a to X_d, 3 zero-one variables are used for each of the integer-valued variables u_a to u_d and 7 zero-one variables are used for τ. This substitution results in the problem size of 50 variables and 19 constraints.

Solving this problem by zero-one programming, we get

$$X_a=0,\ X_b=23,\ X_c=40,\ X_d=43,$$

$$u_a=0,\ u_b=1,\ u_c=1,\ u_d=1$$

$$I_{ab}=I_{bc}=I_{bd}=1$$

and

$$\text{minimum } \tau=49$$

This is the minimum time that a job takes to come out of the assembly-line. Stations *b* and *d* are grouped together. The job takes 20 units of cycle time in Stations *a* and *b*. After completing 9 time units in Station *c* the job emerges from the assembly-line, thus requiring a total of 49 time units.

Delivery problem

The delivery problem arises whenever commodities are to be transported from a central warehouse to a number of customers at different destinations within a specified region. The orders received at the warehouse are grouped and delivered in batches. The deliveries are arranged so that each customer receives his entire order in one delivery but the delivery schedules are set by the shipper on the basis of the availability of carriers. The objective of the shipper is to minimize the total cost of transportation in fulfilling customer orders. Due to Balinskiand and Quandt [6], the following notation is used in the formulation:

m number of destinations

n number of feasible combination of orders—number of activities

A_j activities column vector each having m entities such that the ith entry of A_j is 1, if activity j delivers order i and $A_j=0$, otherwise where $j=1, 2, \cdots, n$

c_j cost of the activity A_j

r number of possible geographical routes

E unit vector of size m

X_j zero-one variable having a value 1 if the activity A_j is used, zero otherwise.

The constraints are given such that

1. A given carrier can combine a number of orders to be delivered together, provided their destinations lie along one of a number of permissible geographical routes and a given destination can receive delivery via a number of different routes.

$$\sum_{j=1}^{n} A_j x_j = E$$

The objective is to minimize total shipping cost:

$$\sum_{j=1}^{n} c_j x_j$$

The total number of zero-one variables used in this formulation in n and the total number of constraints becomes m.

The following sample problem will illustrate the formulation of the delivery problem as a zero-one integer programming problem. Consider a warehouse shipping orders to 4 destinations. The total number of permissible geographical routes, m is 4. These routes are represented by the following vectors:

$$A_1=\begin{pmatrix}1\\0\\1\\0\end{pmatrix};\quad A_2=\begin{pmatrix}1\\0\\0\\1\end{pmatrix};\quad A_3=\begin{pmatrix}0\\1\\1\\0\end{pmatrix};\quad A_4=\begin{pmatrix}0\\1\\0\\1\end{pmatrix}$$

The costs associated with the four routes are shown below;

$$c_1=6;\quad c_2=8;\quad c_3=9;\quad c_4=6$$

The objective is to minimize the total cost of transportation.

The delivery problem can now be formulated as
minimize

$$z=6x_1+8x_2+9x_3+6x_4$$

subject to

$$\begin{aligned}x_1+x_2 &=1\\ x_3+x_4&=1\\ x_1+x_3&=1\\ x_2+x_4&=1\end{aligned}$$

and

$$x_j=0 \text{ or } 1,\qquad j=1,2,3,4.$$

The total number of zero-one variables is 4 and the number of constraints is 4. Solving this problem by zero-one programming, we get

$$x_1=1, x_2=0, x_3=0, x_4=1,$$

and $z=12$. This indicates that shipping in the routes 1 and 4 will minimize the delivery cost.

Traveling salesman problem

The traveling salesman problem may be stated as follows. A salesman, starting from one city, visits each of the other n cities once and only once and returns to the starting city. The problem is to find the order in which he should visit the cities to minimize the total distance traveled (or cost).

The distances between the city pairs can be arranged in a matrix form. Since it is not possible to travel from one city to the same city in one step, the corresponding element in the matrix is a very large value. Thus an infinitely large number is placed in each element on the diagonal of such a matrix. Due to Miller *et al.* [18], the following notation is used in the formulation:

n number of cities to be visited
d_{ij} distance from city i to city j, where $i, j=0, 1, 2, \cdots, n$
u_i arbitrary real-valued variables used to eliminate subtours
X_{ij} zero-one variable having a value of one if the salesman proceeds from city i to city j, and zero otherwise

The constraints are given such that

1. Arrival at each city from any other city is only once excluding the starting city which can be visited any number of times.

$$\sum_{\substack{i=0 \\ i\neq j}}^{n} X_{ij}=1, \qquad j=1, 2, \cdots, n.$$

2. Departure from each city to any other city is once only excluding the starting city which can be visited any number of times.

$$\sum_{\substack{j=0 \\ j\neq i}}^{n} X_{ij}=1, \qquad i=1, 2, \cdots, n.$$

3. Tour should commence and end at the starting city and no tour

should visit more than n cities.

$$u_i-u_j+nX_{ij}\leq n-1, \qquad 1\leq i\neq j\leq n.$$

The objective is to minimize the total distance covered and hence the objective is given by
minimize

$$z=\sum_{0<i\neq j<n}\sum d_{ij}X_{ij}.$$

The total number of variables is n^2+2n and the number of constraints becomes n^2+n. The integral-valued variables $u_i, i=1, 2, \cdots, j$, are converted into zero-one variables using Balas binary technique.

The following sample problem will illustrate the integer linear programming formulation of the traveling salesman problem. Consider a problem in which there are 3 cities to be visited starting from city 0. The distance matrix is as shown below and it is required to find the route which minimizes the total distance traveled.

$$D=\begin{pmatrix} 0 & 1 & 2 & 3 \\ \infty & 4 & 2 & 3 \\ 5 & \infty & 2 & 6 \\ 3 & 5 & \infty & 4 \\ 4 & 3 & 5 & \infty \end{pmatrix}$$

The objective function is to minimize the total distance such that

$$z=4X_{01}+2X_{02}+3X_{03}+5X_{10}+2X_{12}+6X_{13}+3X_{20}+5X_{21}+4X_{23}+4X_{30}+3X_{31}+5X_{32}.$$

The constraints are given such that

1. Arrival to each city from any other city is only once.

$$X_{01}+X_{21}+X_{31}=1$$

$$X_{02}+X_{12}+X_{32}=1$$

$$X_{03}+X_{13}+X_{23}=1$$

2. Departure from each city to any other city is only once.

$$X_{10}+X_{12}+X_{13}=1$$

$$X_{20}+X_{21}+X_{23}=1$$

$$X_{30}+X_{31}+X_{32}=1$$

3. Tour should commence and end at the starting city, and no tour should cover more than n cities.

$$u_1-u_2-3X_{12}\leq 2$$

$$u_1-u_3-3X_{13}\leq 2$$

$$u_2-u_1-3X_{21}\leq 2$$

$$u_2-u_3-3X_{23}\leq 2$$

$$u_3-u_1-3X_{31}\leq 2$$

$$u_3-u_2-3X_{32}\leq 2$$

The three-city problem results in a 15 variables, 9 constraints integer linear programming problem. The following substitution is made to convert the problem into zero-one integer programming problem.

$$x_1=X_{01}, \qquad x_7=X_{20}$$

$$x_2=X_{02}, \qquad x_8=X_{21}$$

$$x_3=X_{03}, \qquad x_9=X_{23}$$

$$x_4=X_{10}, \qquad x_{10}=X_{30}$$

$$x_5=X_{12}, \qquad x_{11}=X_{31}$$

$$x_6=X_{13}, \qquad x_{12}=X_{32}$$

and

$$8x_{13}+4x_{14}+2x_{15}+x_{16}=u_1$$

$$8x_{17}+4x_{18}+2x_{19}+x_{20}=u_2$$

$$8x_{21}+4x_{22}+2x_{23}+x_{24}=u_3$$

The problem now reduces to
minimize

$$z=4x_1+2x_2+3x_3+5x_4+2x_5+6x_6 \\ +3x_7+5x_8+4x_9+4x_{10}+3x_{11}+5x_{12}$$

subject to

$$x_1+x_8+x_{11}=1$$

$$x_2+x_5+x_{12}=1$$

$$x_3+x_6+x_9=1$$

$$x_4+x_5+x_6=1$$

$$x_7+x_8+x_9=1$$

$$x_{10}+x_{11}+x_{12}=1$$

$$8x_{13}+4x_{14}+2x_{15}+x_{16}-8x_{17}-4x_{18}-2x_{19}-x_{20}-3x_5\leq 2$$

$$8x_{13}+4x_{14}+2x_{15}+x_{16}-8x_{21}-4x_{22}-2x_{23}-x_{24}-3x_6\leq 2$$

$$8x_{17}+4x_{18}+2x_{19}-x_{20}-8x_{13}-4x_{14}-2x_{15}-x_{16}-3x_8\leq 2$$

$$8x_{17}+4x_{18}+2x_{19}+x_{20}-8x_{21}-4x_{22}-2x_{23}-x_{24}-3x_9\leq 2$$

$$8x_{21}+4x_{22}+2x_{23}+x_{24}-8x_{13}-4x_{14}-2x_{15}-x_{16}-3x_{11}\leq 2$$

$$8x_{21}+4x_{22}+2x_{23}+x_{24}-8x_{17}-4x_{18}-2x_{19}-x_{20}-3x_{12}\leq 2$$

and

$$x_j=0 \text{ or } 1, \qquad j=1,2,\cdots,24.$$

The total number of zero-one variables is 24 and the number of constraints is 5. The solution of the problem is given by

$$x_3=1, \quad x_5=1, \quad x_7=1, \quad x_{11}=1$$

$$x_j=0, \text{ otherwise}$$

and $z=11$. This indicates that the salesman travels from city 0 to 3, 3 to 1, 1 to 2 and 2 to 0 resulting in a minimum distance of 11 units.

Capital allocation problem

The allocation problem arises in the capital budgeting of a firm. It consists of finding an optimal way in which a firm should allocate the available capital to various projects. This problem can be formulated as an integer programming problem [13]. The following notation is used in the formulation.

- n total number of projects under consideration
- b total amount of investment available
- c_j present worth of all future profits from project j, $j=1, 2, \cdots, n$
- d_j amount of capital required for project j, $j=1, 2, \cdots, n$
- x_j zero-one variable having a value one if project j is taken, zero otherwise

The constraint is that the total capital invested on all the projects undertaken is less than or equal to the capital available.

$$\sum_{j=1}^{n} d_j x_j \leq b$$

The objective is to maximize the present worth of all the future profits from the projects undertaken and is given by

$$z=\sum_{n=1}^{n} c_j x_j.$$

The total number of zero-one variables is n and the constraint is one only.

The following sample problem will illustrate the above formulation. Consider a case where there are 10 projects under consideration. The total available capital is 55. The amount of capital required for the projects and the present worth of all future profits from the projects is as shown below.

$$d_1=30,\quad d_2=25,\quad d_3=20,\quad d_4=18,\quad d_5=17,$$
$$d_6=11,\quad d_7=5,\quad d_8=2,\quad d_9=1,\quad d_{10}=1;$$

$$c_1=20,\quad c_2=18,\quad c_3=17,\quad c_4=15,\quad c_5=15,$$
$$c_6=10,\quad c_7=5,\quad c_8=3,\quad c_9=1,\quad c_{10}=1$$

The objective is to select the projects such that the present worth of all future profits is minimized.

The problem can now be formulated as
maximize

$$z=20x_1+18x_2+17x_3+15x_4+15x_5+10x_6+5x_7+3x_8+x_9+x_{10}$$

subject to

$$30x_1+25x_2+20x_3+18x_4+17x_5+11x_6+5x_7+2x_8+x_9+x_{10}\leq 55$$

and

$$x_j=0 \text{ or } 1, \qquad j=1,2,\cdots,10.$$

Solving this problem of 10 variables and 1 constraint the solution yields

$$x_1=x_2=x_3=0,$$
$$x_4=x_5=x_6=x_7=x_8=x_9=x_{10}=1$$

and a maximum profit of 50. This indicates that the available of 55 units is distributed to projects 4, 5, 6, 7, 8, 9 and 10. The projects 1, 2 and 3 are dropped. This decision results in a maximum profit of 50 units.

Fixed-charge problem

The fixed-chrage problem arises in situations where a certain fixed amount of cost is incurred whenever an activity takes place. The corresponding costs are known as fixed-charges. For example, in transportation, a fixed-charge is incurred regardless of the quantity shipped, or in the building of production facilities where a plant under construction must have a certain minimum size. Because of these fixed-charges, such problems attain special characteristics. If there is a fixed-charge associated with each variable, then every extreme point of the

convex set of feasible solutions yields a local optimum and this complicates the task of solving fixed-charge problems. The following notation is used in the formulation due to Hadley [13].

n number of activities
f_j fixed-charge for activity X_j, $j=1, 2, \cdots, n$
c_j variable cost of activity j, $j=1, 2, \cdots, n$
A coefficient matrix
P column vector of right hand side
d_j zero-one variable having a value 1 if the activity X_j is used, zero otherwise, $j=1, 2, \cdots, n$
u_j upper bound on the variable X_j, $j=1, 2, \cdots, n$
X column vector of x_j, $j=1, 2, \cdots, n$

The constraints are given such that

1. The sum of the resources needed for all activities is equal to the available resources.

$$AX=P$$

2. A fixed-charge is incurred when an activity x_j is used

$$X_j-u_jd_j\leq 0, \quad j=1, 2, \cdots, n$$

The objective is to minimize the total cost incurred and is given such that
minimize

$$z=\sum_{j=1}^{n}(f_jx_j+c_jx_j).$$

The total number of integral-valued variables x_j is n. This is converted to zero-one variable using Balas binary technique.

The following sample problem will illustrate the above formulation. Consider a case where there are 3 activities each with a fixed-charge of 1 and variable cost of 1. The upper bounds on x_1, x_2 and x_3 are given by 5, 4 and 3 respectively. The problem is to

minimize

$$z=2x_1+2x_2+2x_3$$

subject to

$$\begin{aligned}
x_1+x_2+x_3 &= 6\\
2x_1+x_2+3x_3 &= 10\\
x_1-5d_1 &\leq 0\\
x_2-4d_2 &\leq 0\\
x_3-3d_3 &\leq 0
\end{aligned}$$

and

$$x_j\geq 0 \qquad j=1, 2, \cdots, n.$$

The following substitution is made to convert the problem into zero-one integer programming problem:

$$\begin{aligned}
4w_1+2w_2+w_3 &= x_1\\
4w_4+2w_5+w_6 &= x_2\\
2w_7+w_8 &= x_3\\
w_9 &= d_1\\
w_{10} &= d_2\\
w_{11} &= d_3
\end{aligned}$$

The problem now reduces to the followidg:
minimize

$$z=8w_1+4w_2+2w_3+8w_4+4w_5+2w_6+4w_7+2w_8$$

subject to

$$4w_1+2w_2+\ w_3+4w_4+2w_5+w_6+2w_7+w_8\ =\ 6$$

$$8w_1+4w_2+2w_3+4w_4+2w_5+w_6+6w_7+3w_8=10$$

$$4w_1+2w_2+\ w_3-5w_9 \leq 0$$

$$4w_4+2w_5+w_6+4w_{10} \leq 0$$

$$2w_7+w_8-3w_{11}\leq 0$$

and

$$w_j=0 \text{ or } 1, \quad j=1,2,\cdots,11.$$

The total number of zero-one variables is 11 and the number of constraints is 5. Solving this problem by zero-one programming, we get after substitution

$$x_1=4, \quad x_2=2, \quad x_3=0$$

and the value of the objective function is $z=12$. This indicates that activity 1 and 2 are used and activity 3 is dropped with the resultant minimum cost of 12.

Computational Experience

Results using the pseudo-boolean and adaptive binary algorithm codes are detailed in this section and summarized in Tables 1—6. Capital allocation [21] and fixed-charge problems [14] were taken from the literature and all other problems were randomly generated. The problems were converted to zero-one programming models. Each equality constraint had to be broken into two inequality constraints when the adaptive binary was used. Since a pseudo-boolean code can handle equality constraints, they were retained.

The flow shop problems have all equality constraints. The constraints arise mainly due to sequencing and non interference restrictions. The (3×3) problem requires about 33 zero-one variables and 9 constraints whereas a problem of size (4×3) utilizes 52 zero-one variables and 13

Table 1. Scheduling Problems

Problem No.	Problem Size (J×M)	Pseudo-Boolean Code			Adaptive Binary Code
		No. of Variables	No. of Constraints*	Computer Time (Sec.)	Computer Time (Sec.)
1	3×3	33	9	20.73	365.19
2				35.26	157.64
3				28.83	228.02
4				26.63	113.75
5				32.60	212.09
6				44.03	234.11
7				62.57	93.27
8				100.13	115.29
9				11.88	153.74
10				68.55	94.94
11	4×3	52	13	2098.67	2756.46
12				257.66	899.29
13				—	143.06
14				56.16	162.16
15				—	802.52

* Number of constraints are doubled when adaptive binary code is used.

constraints. The variables increase quadratically with the increase in the number of jobs but the increase in the number of constraints is only linear. Since the total number of branches to be investigated is 2^n for n variables, the computation time increases nonlinearly with the increase in the number of jobs. The constraints include an "assignment constraint matrix" and this favors the computational aspect of the pseudo-boolean code by fixing the values of many variables in one branch. This process reduces the number of branches to be investigated to a great extent. Two (4×3) flow shop problems did not converge within 15 minutes and the problem was terminated while using the pseudo-boolean code. This was due to the large number of branches generated in these problems and the fixation of values of the variables in the

Table 2. Line Balancing Problems

Problem No.	Problem Size (No. of Tasks)	Adaptive Binary Code		
		No. of Variables	No. of Constraints*	Computation Time (Sec.)
1	4	50	19	487.98
2				427.60
3				1049.31
4				426.42
5				496.39
6				118.72
7				497.83
8				121.53
9				65.26
10				51.84

* Numbers of variables and constraints are the same for both codes. No solutions are obtained when pseudo-boolean code was used.

branches was poor.

The assembly-line balancing problems have all inequality constraints. A large number of matrix and cost coefficients are zero. The constraints arise mainly due to ordering and noninterference restrictions. A 4-task problem requires about 50 zero-one variables and 19 constraints; whereas an 8-task problem requires about 96 zero-one variables and 42 constraints. The increase in the number of variables and constraints is linear. Because of the absence of "assignment matrix constraints", the fixation of values to the variables in various branches was very poor. This increases the number of branches and the amount of search to a great extent. The convergence was very slow while using the pseudo-boolean code and the program had to be terminated after 15 minutes without reaching the optimal value.

Moreover, an 8-task line balancing problem taken from Bowman [10] was tried using the adaptive binary code. It resulted in a problem size of 96 variables and 42 constraints. The program failed to attain

Table 3. Delivery Problems

Problem No.	Problem Size (n×m)	Pseudo-Boolean Code		
		No. of Variables	No. of Constraints*	Computation Time (Sec.)
1	(7×3)	7	3	0.70
2				1.30
3				0.68
4				1.35
5				0.71
6	(14×5)	14	5	4.53
7				4.54
8				2.01
9				2.37
10				2.77
11	(32×6)	32	6	29.34
12				53.60

* Number of constraints is doubled when adaptive binary code is used. No solutions are obtained when adaptive binary code was used.

the optimal value within one hour and has to be terminated.

The delivery problems have all equality constraints with the constraint matrix coefficients being either zero or one. The constraints arise mainly to satisfy the requirement that the demand is equal to the supply and the commodity has to be shipped in one of the permissible geographical routes. The total number of zero-one variables is equal to the number of feasible combinations of orders and the number of constraints is equal to the number of destinations. The increase in the number of variables and the number of constraints is linear. Due to the zero-one coefficients and unity in the right hand side in the constraint matrix, the number of branches is reduced to a great extent and the search converges very rapidly. In the case where the adaptive binary code, the equality constraints were split into two inequality constraints. The greater than or equal to constraints were converted to

Table 4. Traveling Salesman Problems

Problem No.	Problem Size (Cities)	Pseudo-Boolean Code			Adaptive Binary Code
		No. of Variables	No. of Constraints*	Computation Time (Sec.)	Computation Time (Sec.)
1	4	24	12	4.00	63.30
2				5.16	64.46
3				4.17	67.99
4				3.89	106.54
5				8.75	136.28
6				5.40	148.67
7				3.91	235.15
8				3.93	72.47
9				3.84	165.77
10				4.02	197.82

* Number of constraints is 18 when adaptive binary code is used.

Table 5. Capital Allocation Problems

Problem No.	No. of Variables	No. of Constraints	Pseudo-Boolean Code	Adaptive Binary Code
			Computation Time (Sec.)	Computation Time (Sec.)
1	10	1	1.28	5.05
2			1.64	4.71
3			1.43	4.80
4			2.11	4.85
5			2.08	1.45
6			1.93	4.69
7			1.45	5.89
8			1.36	5.07
9			1.40	1.40

* Numbers of variables and constraints are the same for both codes.

Table 6. Fixed-Charge Problems

Problem No.	No. of Variables	No. of Constraints*	Pseudo-Boolean Code	Adaptive Binary Code
			Computation Time (Sec.)	Computation Time (Sec.)
1	11	4	4.10	14.33
2			5.75	19.88
3			6.24	16.93
4			4.37	10.92
5	17	6	7.18	37.58
6			8.37	34.02
7			7.17	69.36
8			8.62	54.29
9	9	6	1.72	4.30

* Numbers of variables and constraints are the same for both codes.

less than or equal to constraints. The adaptive binary program fixed all the variables at zero value thereby violating the constraints. It failed to reach the optimal value. Several parameter modifications were tried without any success. The reason for this failure could not be determined.

The traveling salesman problems have half equality and half inequality constraints. The constraints arise due to the fact that each city should be visited only once without any overlapping of the tour. The increase in the number of variables and constraints is quadratic with the increase in the number of cities to be visited. This fact imposes a severe restriction of the size of the problem that can be solved by utilizing this formulation. The constraints include an "assignment constraint matrix" and this favors the use of pseudo-boolean code, since the convergence was very good.

All the nine capital allocation problems are the same except for the right hand side of the constraint matrix. These problems differ from others by having dense coefficient matrix. That is, all the coefficients are greater than zero. Pseudo-boolean code shows better results in

solving the capital allocation problems than the adaptives binary code did.

The solution of the fixed-charge problems is made difficult by a number of local optimal solutions which obscure the global optimum. Results on nine test problems from Haldi [14] indicate that the convergence of pseudo-boolean code is faster than that of adaptive binary code in solving fixed-charge problems.

Conclusions

The size of the problem which can be solved by using the pseudo-boolean code has to be restricted because of the large storage requirements. It was observed that a considerable amount of time is spent in substituting the value of the variable obtained in one equation or inequality, in all the remaining equations and/or inequalities and simplifying the system. Further, if one variable is fixed in a simplification, again the substitution and simplification are to be made which consume a lot of computer time. Several different methods were tried to reduce this time. It was found that starting the constraints with equations, if any, produced better results.

Due to the zero-one coefficients and unity in the right hand side in the constraint matrix, delivery problems converge to the optimal value rapidly while using the pseudo-boolean code. The failure of the adaptive binary code in obtaining the solution of simple delivery problems came as a surprise. The reason for this failure could not be found out. Because of the assignment matrix constraints pseudo-boolean code converge better than the adaptive binary code. Test problems in capital allocation and fixed-charge indicates the superiority of the pseudo-boolean code over the adaptive binary code in solving those problems.

The main drawback of the pseudo-boolean code is the large amount of core locations it requires to store the node values of the branching tree. Hence, pseudo-boolean code is a very efficient technique in solving small and medium sized problems.

REFERENCES

[1] Ashour, S., "A Statistical Analysis of Production Scheduling Systems," *Journal of Operations Research Society of Japan*, **12**, 2, 1970.

[2] Ashour, S., "An Experimental Investigation and Comparative Evaluation of Flow-Shop Scheduling Techniques," *Operations Research*, **18**, 3, 1970.

[3] Ashour, S., ***Introduction to Scheduling: Concepts, Analyses and Performances***, John Wiley and Sons, in press.

[4] Balinski, M.L., "Integer Programming: Methods, Uses, Computation," *Management Science*, **12**, 3, 1965, pp. 253–313.

[5] Balinski, M.L., "On Recent Developments in Integer Programming," *Proceedings of the International Symposium on Mathematical Programming*, Princeton University Press, Princeton, N.J., 1967.

[6] Balinski, M.L. and Quandt, R.E., "On an Integer Program for a Delivery Problem," *Operations Research*, **12**, 2, 1964, pp. 300–304.

[7] Beale, E.M.L., "Survey of Integer Programming," *Operational Research Quarterly*, **16**, 2, 1965, pp. 219–228.

[8] Bellman, R., *Dynamic Programming*, Princeton University Press, Princeton N.J., 1957.

[9] Bowman, E.H., "The Schedule Sequencing Problem," *Operations Research*, **7**, 5, 1959, pp. 621–624.

[10] Bowman, E.H., "Assembly-Line Balancing by Linear Programming," *Operations Research*, **8**, 3, 1960, pp. 385–389.

[11] Char, A.R., "Application of Linear Pseudo-Boolean Programming to Combinatorial Problems," M.S. Thesis, Kansas State University, Manhattan, Kansas, 1970.

[12] Dantzig, G.B., *Linear Programming and Extensions*, Princeton University Press, Princeton, N.J. 1963.

[13] Hadley, G., *Nonlinear and Dynamic Programming*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1964.

[14] Haldi, J., "25 Integer Programming Test Problems," Working Paper No. 43, Graduate School of Business, Stanford University, Stanford, California, 1964.

[15] Hammer, P.L. and Rudeanu, S., "Pseudo-Boolean Programming," *Operations Research*, **17**, 2, 1969, pp. 233–261.

[16] Johnson, E.L., Edmonds, J. and Lockhart, S., "The Degree Constrained Subgraph Problem," Lecture, Math. Programming Symposium, Princeton, N.J., 1967.

[17] Manne, A.S., "On the Job-Shop Scheduling Problem," *Operations Research*, **8**, 2, 1960, pp. 219–223.

[18] Miller, C.E., Tucker, A.W. and Zemlin, R.A., "Integer Programming Formulation of Traveling Salesman Problems," *Journal of the Association for Computing Machinery*, **7**, 4, 1960, pp. 326–329.

[19] Rao, A., "Zero-One Integer Linear Programming," Ph.D. Thesis, University

of Iowa, 1968.

[20] Salkin, H. and Spielberg, K., "Adaptive Binary Programming," IBM New York Scientific Center Report No. 320-2951, 1968.

[21] Trauth, C.A. and Woolsey, R.E., "Integer Linear Programming: A Study in Computational Efficiency," *Management Science*, 15, 9, 1969, pp. 481-493.

[22] Wagner, H.M., "An Integer Linear Programming Model for Machine Shop Scheduling, *Naval Research Logistics Quarterly*, 6, 2, 1959, pp. 131-140.