Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Assembly-Line Balancing by Linear Programming

E. H. Bowman,

To cite this article:
E. H. Bowman, (1960) Assembly-Line Balancing by Linear Programming. Operations Research 8(3):385-389. http://dx.doi.org/10.1287/opre.8.3.385

ASSEMBLY-LINE BALANCING BY LINEAR PROGRAMMING

E. H. Bowman

Massachusetts Institute of Technology, Cambridge, Massachusetts

(Received November 2, 1959)

Linear-programming solutions to the assembly-line balancing problem are offered in two forms Feasible solutions depend on work recently presented on integer solutions to linear-programming problems As yet, the computation involved for a practical problem would be quite large

THIS PAPER develops two different linear-programming approaches to the assembly-line balancing problem They are solutions in the sense that the problem is stated as sets of linear constraints in variables that can be evaluated in linear objective functions Algorithms have been developed to produce solutions to these problems,[1] as well as the integer requirements for the variables in the linear program [2]

The 'assembly-line balancing problem'[3] exists where a number of operations must be performed sequentially with partial ordering constraints Given an output rate (e g , three assemblies per hour) to flow from the line, how can the operations be grouped and wholly assigned to stations so that a minimum number of stations is required

A simple example of the problem is shown in Fig 1 The amount of computation required by the linear programs presented here may be unrealistically extensive, i e , they may not be practical Complete enumeration of the possibilities of these problems may be set forth as an alternative approach, and for a problem as simple as that shown in Fig 1, such an approach would be more sensible

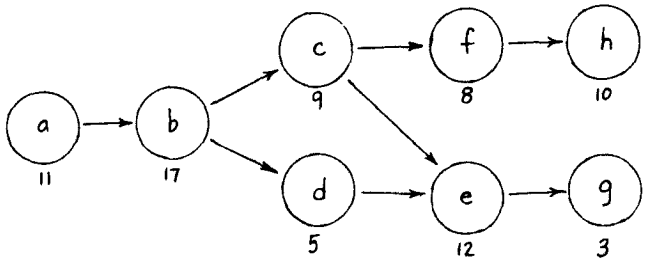

Fig 1 Partial ordering constraint schematic

Operation *a* must precede operation *b*, which must precede both operations *c* and *d* That is, the arrows indicate the order which is required The numbers represent the time (e g , minutes) required to complete each

operation The sum of the times required to do each operation equals 75 If these represent minutes, and three assemblies are required per hour, then one assembly must leave the line every 20 minutes Each station on the line then has up to 20 minutes to complete its set of assigned operations before it must start on the next assembly A minimum of four stations is required $(75/20=4^-)$ (By inspection of Fig 1, it can be seen that four stations will not be enough and five will be enough, so if the answer were of importance here rather than the method, the problem would be finished) Consider that seven stations is an upper limit of the stations that might be required The seven possible stations, in order on the line, will then be identified as A, B, C, D, E, F, G The objective will be to pack the operations into the earlier stations on the line, leaving the latter stations free—i e , not requiring them

FIRST LINEAR PROGRAM*

THE NATURE of the variables is typified by A_a meaning the number of time units (minutes) devoted to operation a at station A

$$\begin{aligned} &A_a+A_b+A_c+A_d+ \quad +A_h\leqq 20,\\ &B_a+B_b+B_c+ \quad +B_h\leqq 20,\\ &\qquad\qquad\qquad\qquad\\ &G_a+G_b+ \quad +G_h\leqq 20 \end{aligned} \tag{1}$$

This set of constraints (1) assures that none of the stations is overloaded

$$\begin{aligned} &A_a+B_a+C_a+D_a+ \quad +G_a=11,\\ &A_b+B_b+C_b+ \quad +G_b=17,\\ &\qquad\qquad\qquad\qquad\\ &A_h+B_h+ \quad +G_h=10 \end{aligned} \tag{2}$$

This set of constraints (2) assures that each operation is performed

The following set of constraints includes integer variables of the form $A_a I$ which must take the values zero or one

$$\begin{aligned} &\tfrac{1}{11}A_a+A_a I=1, \quad \tfrac{1}{11}B_a+B_a I=1, \quad \tfrac{1}{11}G_a+G_a I=1,\\ &\tfrac{1}{17}A_b+A_b I=1, \quad \tfrac{1}{17}B_b+B_b I=1, \quad \tfrac{1}{17}G_b+G_b I=1,\\ &\qquad\qquad\qquad\qquad\qquad\qquad\\ &\tfrac{1}{10}A_h+A_h I=1, \quad \tfrac{1}{10}B_h+B_h I=1, \quad \tfrac{1}{10}G_h+G_h I=1 \end{aligned} \tag{3}$$

The simplex algorithm[1] assures naturally that the integer variables will

* This formulation partially parallels an earlier paper,[4] which develops a linear programming approach to the job-shop scheduling problem

be nonnegative Constraint set (3) assures that the integer variables are not greater than one The integer algorithm[2] then assures that the value taken will be either zero or one That is, separate constraints need not initially be written into the statement of the problem to assure this The basic purpose of constraint set (3) is to assure that the operations are not split between stations, that is, that they are assigned to only one station For instance, the constraint on A_a insists that it take the value of either eleven or zero, considering that $A_a I$ must take the value of either zero or one

$$\begin{aligned} &\tfrac{1}{17} A_b \leqq \tfrac{1}{11} A_a, \quad \tfrac{1}{17} B_b \leqq \tfrac{1}{11} A_a + \tfrac{1}{11} B_a, \\ &\qquad\qquad \tfrac{1}{17} C_b \leqq \tfrac{1}{11} A_a + \tfrac{1}{11} B_a + \tfrac{1}{11} C_a, \quad , \\ &\tfrac{1}{10} G_h \leqq \tfrac{1}{8} A_f + \tfrac{1}{8} B_f + \tfrac{1}{8} C_f + \tfrac{1}{8} D_f + \tfrac{1}{8} E_f + \tfrac{1}{8} F_f + \tfrac{1}{8} G_f, \quad , \\ &\tfrac{1}{3} G_g \leqq \tfrac{1}{12} A_e + \tfrac{1}{12} B_e + \tfrac{1}{12} C_e + \tfrac{1}{12} D_e + \tfrac{1}{12} E_e + \tfrac{1}{12} F_e + \tfrac{1}{12} G_e \end{aligned} \tag{4}$$

Constraint set (4) assures proper ordering The first constraints for instance assure that operation a precedes operation b on the line Only the immediately preceding operation in an ordering need be considered, as the chain of constraints built up will assure that earlier operations will be preceded by still earlier operations in the ordering

The objective function (to be minimized) of this linear program takes the form $z = 1\,(E_g + E_h) + 14\,(F_g + F_h) + 196\,(G_g + G_h)$

The purpose of this objective function is to make later stations exceedingly costly, pushing the operations as far forward as is physically possible Stations A through D must certainly be used and need assume no cost Only operations with no succeeding operations in an ordering need positive costs in the objective function, i e , they may be last on the line The nature of the cost explosion, 1, 14, 196 is to make one unit of a later assignment more costly than the sum of all preceding station assignments, $[14 > 10+3,\ 196 > 10(14)+3(14)+10+3]$

SECOND LINEAR PROGRAM*

THE SAME problem will now be placed in an entirely different linear program The nature of the variable is typified by X_a meaning the 'clock time' when operation a is started The stations are as before, and as shown in Fig 2, but need not be included explicitly in the formulation

$$\begin{aligned} &X_a + 11 \leqq X_b, \quad X_b + 17 \leqq X_c, \quad X_b + 17 \leqq X_d, \\ &\qquad , \quad X_f + 8 \leqq X_h, \quad X_e + 12 \leqq X_g \end{aligned} \tag{5}$$

* This formulation partially parallels a paper by ALAN S MANNE [5] Constraint sets (6), (7), and (8) follow Manne's paper directly

Station	A	B	C	D	E	F	G
Clock time	1–20	21–40	41–60	61–80	81–100	101–120	121–140

Fig 2 Assembly-line stations

Constraint set (5) assures proper ordering That is, operation a will precede operation b, as a's starting clock time is at least 11 minutes processing time before the starting time of operation b

In order to assure that operations do not use the same clock time (noninterference) a new integer valued variable, of the type I_{cd}, is introduced Operations c and d are used as potential interfering examples

$$I_{cd} \leqq 1, \tag{6}$$

$$(140+5)I_{cd}+(X_c-X_d) \geqq 5, \tag{7}$$

$$(140+9)(1-I_{cd})+(X_d-X_c) \geqq 9 \tag{8}$$

The simplex algorithm assures that no variables are negative, the integer algorithm that desired variables are integer, so constraint set (6) assures values of either zero or one

As $|X_c-X_d| \leqq 140$, the effect of constraint sets (7) and (8) may be summarized as follows If $(X_c-X_d)\{>0, =0, <0\}$, constraint set (7) implies that $I_{cd}=\{0,1, 1, 1\}$, while the constraint set (8) implies that $I_{cd}=\{0, 0, 0,1\}$

Neither the value of zero or one for I_{cd} permits $(X_c-X_d)=0$ If I_{cd} equals zero, then operation d precedes operation c, i e , $X_d<X_c$ If I_{cd} equals one, then operation c precedes operation d, i e , $X_c<X_d$ In both cases the difference will at least be as large as the processing time on the earlier operation Operation pairs need only be included in the constraint sets (7) and (8) where they are not constrained in an ordering requirement, where otherwise the constraint set (5) would exclude interference

As yet the main requirements for the assembly line balancing problem have not been met—that the station not be overloaded, and that the operations be wholly assigned within station

$$\begin{aligned} X_a+11 &\leqq 20\, I_a+20, \quad X_a \geqq 20\, I_a, \\ X_b+17 &\leqq 20\, I_b+20, \quad X_b \geqq 20\, I_b, \\ X_c+9 &\leqq 20\, I_c+20, \quad X_c \geqq 20\, I_c, \\ X_h+10 &\leqq 20\, I_h+20, \quad X_h \geqq 20\, I_h \end{aligned} \tag{9}$$

Constraint set (9) simultaneously satisfies both of these requirements Station A only handles from 1 to 20 on the clock, station B only from 21 to 40, continuing to station G, which only handles from 121 to 140 (*see* Fig 2) That is, each station covers only a 20-minute assignment New integer valued variables are introduced in constraint set (9) of the form I_a They may take any positive integer value starting with zero If I_a equals zero, then the right-hand constraint indicates that operation a may start in station A, and the left-hand constraint requires that operation a finish within station A If for instance I_c equaled three, then operation c is fully contained within station D (61–80)

$$X_g+3\leqq\tau, \qquad X_h+10\leqq\tau \tag{10}$$

Constraint set (10), made up of all the operations with no followers in a required ordering, permits the objective function (to be minimized) to be simply $z=\tau$

COMPARISON

THE FIRST linear program for the simple problem used here as an example uses 135 constraint equations (or inequalities) with 56 variables plus 56 special integer variables that equals 112 variables, not counting slacks The second linear program for the same problem uses 33 constraint equations (or inequalities) with 8 variables plus 15 special integer variables, plus the variable τ which equals 24 variables not counting slacks The second formulation would seem far superior to the first The algorithm required for integer solution may make the computations required for an assembly-line balancing problem of even modest size quite considerable Sample problems and their computation may provide experience from which some generalizations can be made

REFERENCES

1 G B DANTZIG, "Maximization of a Linear Function of Variables Subject to Linear Inequalities," T C Koopmans (ed), *Activity Analysis of Production and Allocation*, Cowles Commission 13, Wiley, New York, 1951
2 R GOMORY, "Outline of an Algorithm for Integer Solutions to Linear Programs," *Bull Am Math Soc* (September 1958)
3 M E SALVESON, "The Assembly Line Balancing Problem," *Trans ASME* **77**, 939–948 (1955)
4 E H BOWMAN, "The Schedule Sequencing Problem," *Opns Res* **7**, 621–624 (1959)
5 ALAN S MANNE, "On the Job-Shop Scheduling Problem," to be published