G. CIOBANU

Department of Economic Cybernetics
Academy of Economic Studies

A Branch-and-Bound Algorithm to Solve an Assembly Line Balancing Problem

This paper presents a branch-and-bound algorithm to solve the problem of an assembly line balancing.

Based on the notion of feasible set introduced by Gutjahr and Nemhauser, the field of the feasible partitions of the set of operations is explored by lexicographical enumeration. A scheduling program is associated to a partition. To eliminate the partitions which are not potentially optimal, a lower bound of the duration of assembly corresponding to a scheduling program is defined.

A production line balancing problem appears when a set of operations is grouped in stations along a continuous production line and the following conditions must be fulfilled [3]:

a) the allocation of operations in the work stations should not violate the precedence relations between operations;

b) any operation is entirely carried out in a station;

c) the time allocated to each work station of the technological line should not surpass a cyclic time pre-established by the production rates.

1. THE PROBLEM OF AN ASSEMBLY LINE BALANCING

Consider $G = (A, \Gamma)$ an oriented graph, without circuits attached to a project, where $A = \{a_1, a_2 \dots a_s\}$ is the set of operations, s is the number of operations and $\Gamma: A \to P(A)$. The precedence relation between operations in A is denoted by "$\ll$" ($a_i \ll a_j$ if $a_j \in \Gamma a_i$). The duration (process time) of an operation $a_k \in A$ is denoted by d_k and it is a positive number. A *station* is a subset B of the set of operations A.

Definition 1. A production line with N stations is a partitioning of set A, denoted $\mu(N) = [B_1, B_2 \dots B_N]$, where

$\bigcup_{i=1}^{N} B_i = A$, $B_i \cap B_j = \varnothing$, $i \neq j$, $i, j = 1, 2, \dots, N$ and with the property that if $a_k, a_l \in A$, $a_k \ll a_l$ and $a_k \in B_i$, $a_l \in B_j$, in this case $i \leqslant j$. The length of a production line is the number of stations $L(\mu(N)) = N$.

Definition 2. We call a *cyclic time* of a production line $\mu(N)$ with N stations, the size

$$\tau(\mu(N)) = max_i \left\{ \sum_{a_k \in B_i} d_k \right\}. \tag{1}$$

As to the balancing of a production line (the definition of partitioning μ) we shall define the following two problems [10]:

Problem 1.

Let C be a given cyclic time, $C \geqslant \max\limits_{a_k \in A} d_k$. Find a production line

$$\mu(n) = [B_1, B_2, \dots, B_n]$$

with n stations and $\tau(\mu(n)) \leqslant C$ for which the number of stations "n" is minimum.

Problem 2.

Let $N > 0$ be a given number of stations. Determine a production line $\mu(n) = [B_1, B_2, \dots, B_n]$ with $n \leqslant N$ for which the cyclic time $\tau(\mu(n))$ is minimum.

To solve problem 1 the dynamic programming algorithms [5], [7] and [8] have been proposed, algorithms based on the enumeration of the feasible partitioning combinations [6]. Gutjahr and Nemhauser [4] reduce problem 1 to the problem of the minimum path in a network by using the notion of feasible set. A branch-and-bound algorithm has been defined by Nevins [11] by using as a lower bound for a partial partition $B_1, B_2, \dots, B_n$ at n stations $n \leqslant N$, an average time for the remaining stations $n + 1, n + 2, \dots N$ defined by the size:

$$C_n = \frac{\sum\limits_{a_k \in A} d_k - \sum\limits_{i=1}^{n} \sum\limits_{a_k \in B_i} d_k}{N - n}.$$

If $C_n > C$, we give up the extension of partition $\mu(n) = [B_1, B_2, \dots, B_n]$.

Definition 3. [4] Set $Q \subset A$ is called feasible if for each operation $a_j \in Q$ we have $\Gamma^{-1} a_j \in Q$.

In this paper, we shall present a branch-and-bound algorithm for solving problem 1, in which the partitioning of the set of operations in stations is based on the lexicographical enumeration of the feasible sets. We shall attach to the partition a scheduling program and correspondingly a lower bound of the total scheduling length.

Problem 2 is solved by the iterative approach of problem 1 by increasing the cyclic time at each iteration until a balancing of the line is obtained. The lower bound of the cyclic time for a fixed length of the line is

$$C = \frac{\sum_{a_k \in A} d_k}{N}$$

where d_k is duration of $a_k \in A$ and N is the number of stations.

Definition 4. We call the *idle time* of a line with N stations and cyclic time C the quantity of time during which on the production line resources are not used for each product unit and it is given by expression

$$I = NC - \sum_{a_k \in A} d_k. \tag{2}$$

It is noticed from this relation that any improvement of the length of line N for a specified time cycle C, the idle time I decreases, also for any reduction of the time cycle C for a length of specified line N, the idle time decreases, thus, an optimum solution either for problem 1, or for problem 2 minimizes the idle time.

We denote by:

$$I_n = \sum_{j=1}^{n} [C - D(B_j)] = nC - D(B_1, B_2, ..., B_n)$$

the idle time in stations 1, 2 ..., n, where

$$D(B_j) = \sum_{a_k \in B_j} d_k$$

and

$$D(B_1, B_2, ..., B_n) = \sum_{j=1}^{n} D(B_j).$$

Definition 5. We call *free time* in station j

$$F_j = C - D(B_j).$$

It is noticed that size I_n fluctuates during the extension process which decreases during the allocation of operations at a station and it increases when a new station is created. Finally, I_N attains value I given by relation (2).

In any stage, the difference

$$R_n = I_n - F_n$$

is positive and it increases during the process. It is constant during the allocation of operations at a station and increases by F_n when a new station is added. This difference R_n which characterizes the allocation of operations up to station n and it is considered as a lower bound of the idle time, in the algorithms given by Mansoor and Yadin [9].

Definition 6. [9] An allocation of a set of operations $Q \subset A$ in n stations is a partition $\mu(n) = [B_1, B_2, ..., B_n]$ of Q in n subsets $B_1, B_2, ..., B_n$ with the property that for any $k = 1, 2, ..., n$

$$Q_k = \bigcup_{j=1}^{k} B_j$$

is a feasible set.

Thus, the allocation of operations in stations must not violate the precedence relations.

Definition 7. [9], A partition $\mu(n) = [B_1, B_2, ..., B_n]$ of the set of operations A in n stations with the cyclic time C is feasible if

$$\tau(\mu(n)) \leqslant C.$$

Definition 8. [9] A feasible partition of the set of operations Q in n stations with the cyclic time C is said to be *extensible*, if it can be continued by allocating the operations in $Q^c = A - Q$ at stations $n + 1, n + 2, ..., N$.

By allocating an operation at a work station, also attached to it is a starting term, thus, a scheduling programme U of operations $A = \{a_1, a_2, ..., a_s\}$ is attached to a partition μ.

Definition 9. We call a scheduling program attached to a feasible partition μ a vector $U = (u_1, u_2, ..., u_s)$ with the property that for each pair of operations $a_k a_l \in A$ with $a_k \ll a_l$ (that is $a_l \in \Gamma a_k$) on verify the relation $u_k + d_k \leqslant u_l$ where u_k is the starting term of operation a_k, u_l the starting term of operation a_l, and d_k the duration of operation a_k.

An operation a_k, $k = 1, 2, ..., s$ to which a starting term u_k has been attached is called a programmed operation.

Definition 10. We call the duration of a scheduling program U attached to partition μ a size $T(\mu)$ equal, to

$$T(\mu) = \max_{a_k \in A} (u_k + d_k)$$

$T(\mu)$ is also called the duration of assembly corresponding to partition μ.

This duration is considered in this paper as the measuring criterion of the quality of the feasible partitions.

Theorem 1. Any program $U = (u_1, u_2, ..., u_s)$ of the scheduling of operations $A = \{a_1, a_2, ..., a_s\}$ determines a sequence of feasible sets $Q_1, Q_2, ..., Q_p$ which verify the following properties:

a) $Q_i \subset Q_{i+1}$

b) for any $a_k \in Q_{i+1}$, $a_k \notin Q_i \Rightarrow \Gamma^{-1} a_k \subset Q_i$.

Proof. We increasingly order the elements of program U and by renumbering them we suppose that

$$u_1 \leqslant u_2 \leqslant ... \leqslant u_s.$$

Let $A = X_1 \cup X_2 \cup ... \cup X_p$, p be a natural number, a partition of set A, formed as follows: if $a_k \in X_\alpha$, $a_l \in X_\alpha$ in this case $u_k = u_l$ (if $a_k \in X_\alpha$ and $a_l \in X_\beta$, in this case $u_k \neq u_l$) and if $\alpha < \beta$ and $a_k \in X_\alpha$, $a_l \in X_\beta$ then $u_k < u_j$.

Let $Q_1 = X_1$, $Q_2 = X_1 \cup X_2$, ..., $Q_i = \bigcup_{\alpha=1}^{i} X_\alpha, ..., Q_p = \bigcup_{\alpha=1}^{p} X_\alpha$.

We shall point out that $Q_1, Q_2, ..., Q_p$ form a sequence of feasible sets verifying properties a) and b).

It is noted first that $Q_p \equiv A$, as being the sum of the p subsets X_α. Consider $a_k \in Q_i$ and $a_l \in A$ so that $a_l \ll a_k$. It results from definition 9 that $u_k \geqslant u_l + d_l$, where d_l represents the length of operation a_l. If $d_l > 0$ then $u_k > u_l$ thus $a_k \in X_\alpha$, $a_l \in X_\beta$ and $\beta < \alpha$. Because $a_k \in Q_i$ it results $X_\alpha \subset Q_i$. From the construction of sequence it results that $X_\beta \subset Q_i$, consequently $a_l \in Q_i$ and Q_i is a feasible set according to definition 3.

Property a) is obvious from the construction.

It is noted that $Q_{i+1} = Q_i \cup X_{i+1}$. Consider $a_k \in X_{i+1}$ and $a_k \notin \Gamma Q_i$, where $\Gamma Q_i = \bigcup_{a_h \in Q_i} \Gamma a_h$. There is $a_j \in Q_{i+1}$ so that $a_j \ll a_l \ll ... \ll a_n \ll a_k$ with $a_l \in \Gamma Q_i$. According to program U we have $u_j + d_j \leqslant u_l, ..., u_m + d_m \leqslant u_k$, thus between u_j and u_k there is at least another number u_q so that $u_i < u_q < u_k$ which is in contradiction with the procedure of constructing set X_{i+1}, thus $a_k \in \Gamma Q_i$. Consequently, line $Q_1, Q_2, ..., Q_p$ also verifies property b).

In this way, the problem of a production line balancing can be analysed and solved by taking into account theorem 1 and definitions 7 and 8 for generating feasible partitions. As a matter of fact, both the problem of a project scheduling (or scheduling on machines) and the problem of a production line balancing consider the set of operations represented by a network type graph.

The problem on a production line balancing finds a solution if graph $G = (A, \Gamma)$ contains no circuits.

The following algorithm called the ALB algorithm (the Assembly Line Balancing) generates feasible partitions having in view special rules for the distribution of operations in stations. The operations are increasingly ordered according to the minimum starting terms (as a first sorting key) and if they are equal according to a second criterion (as a second sorting key) which may be, for example: the duration of operations, the maximum starting term, the maximum finishing term, the time reserve (the difference between the maximum finishing term and the maximum starting term).

The ALB algorithm is thus generating feasible partitions and maintains only that with the smallest idle time.

The *ALB algorithm* (an Assembly Line Balancing)

Let $G = (A, \Gamma)$ be the graph of operations, $A = \{a_1, a_2, ..., a_s\}$ and C a given cyclic time.

Initializing. We calculate the program $U^\circ = (u_1^0, u_2^0, ..., u_s^0)$ where $u_j^0 = \max_{a_i \in \Gamma^{-1} a_j} (u_i^0 + d_i)$. Consider $i = 0$, $n = 1$, $t_0 = \min_{a_j \in A} u_j^0 \cdot Q_0 = \varnothing$, $R_0 = 0$, $S_1 = \varnothing F_1 = CR^* = \infty$ and δ a sorting criterion.

We suppose in the general stage that there is a feasible set Q_i, a scheduling program $U^{(i)} = (u_1^{(i)}, u_2^{(i)}, ..., u_s^{(i)})$ and a time t_i when an operation can be distributed in station n.

Step 1. Determine

$$\Pi_i = \{a_j \in A \mid a_j \notin Q_i,\ u_j^{(i)} = t_i \cdot u_j^{(i)} \in U^{(i)}\} \tag{3}$$

the set of operations candidates at moment t_i.

If $\Pi_i = \varnothing$ the algorithm is finished: the number of stations is $N = n$. (Π_i is obtained by sorting the operations according to the starting terms $u_j^{(i)}$, $j = 1, 2, ..., k$).

Step 2. Π_i is ordered according to criterion δ.

Let a_l be the first operation of Π_i (charted). $a_l \in \Pi_i$ with duration d_l. If $d_l > F_k$ go to step 4.

Step 3. Consider $Q_{i+1} = Q_i \cup \{a_l\}$. Calculate $t_{i+1} = t_i + d_l$, $F_n \leftarrow F_n - d_l$, $B_n \leftarrow B_n \cup \{a_l\}$ where d_l is the duration of operation a_l and go to step 5.

Step 4. Consider $t_i \leftarrow t_i + F_n$, $R_n = R_{n-1} + F_n$ also $n \leftarrow n + 1$, $F_n = C - d_l$, $B_n = \{a_l\}$, $t_{i+1} = t_i + d_l$, d_l duration of operation a_l.

$Q_{i+1} = Q_i \cup \{a_l\}$ and go to step 5.

Step 5. Calculate partly program $U^{(i+1)} = (u_1^{(i+1)}, u_2^{(i+1)}, ..., u_s^{(i+1)})$ thus

$$u_j^{(i+1)} = \begin{cases} u_j^{(i)} & a_j \in Q_i \\ t_i & a_j \equiv a_l \\ t_{i+1} & a_j \in \Pi_i - \{a_l\} \\ max & (u_r^{(i+1)} - d_r) \\ a_j \ll a_r & \end{cases} \tag{4}$$

Consider $i \leftarrow i + 1$ thus $t_i \leftarrow t_{i+1}$, $U^{(i)} \leftarrow U^{(i+1)}$, $Q_i \leftarrow Q_{i+1}$ and go to step 1.

By changing the sorting criterion δ several feasible partitions can be obtained. The partition with the smallest idle time is kept.

The feasible partition $\mu(N) = [B_1, B_2, ... B_N]$ is formed in steps 3 and 4 and that was the idle time R_N. The number of stations is $N = n$.

2. A BRANCH-AND-BOUND ALGORITHM FOR AN ASSEMBLY LINE BALANCING

Let $\mu(n) = [B_1, B_2, ..., B_n]$ be a partly feasible partition and $Q = \bigcup_{i=1}^{n} B_i$, $Q^c = A - Q$. According to relation (4) we determine the scheduling program $U = (u_1, u_2, ..., u_s)$ corresponding to the partial partition $\mu(n)$.

$$T(\mu(n)) = \max_{a_j \in Q} (u_j + d_j) + \sum_{a_j \in Q^c} d_j \tag{5}$$

is the estimation of the total duration for effectuating the operations in the conditions of a production line.

If $\mu(N) = [B_1, B_2, ..., B_N]$ is an entirely feasible partition then

$$T(\mu(N)) = \max_{a_j \in A} (u_j = d_j)$$

the idle time R_N is determined by the ALB algorithm.

It is noticed that

$$T(\mu(N)) = R_N + D(A).$$

As $D(A) = \sum_{a_i \in A} \alpha_i$ (the sum of durations) is constant it results that the minimizing of the idle time R_N is equivalent to the minimizing of the total duration $T(\mu(N))$.

Theorem 2. Let $\mu^* = [B_1^*, B_2^*, ..., B_N^*]$ be a known feasible partition with the total duration $T(\mu^*)$ and $\mu(n) = [B_1, B_2, ... B_n]$, $n \leqslant N$ a partly feasible partition with $T(\mu(n)) > T(\mu^*)$ then partition $\mu(n)$ is not extensible in stations $n+1, n+2, ..., N$.

Proof. It is noticed from (5) that the sum of durations of operations in Q^c (thus those which were not scheduled) exceeds the time distributed in stations $n+1, n+2, ... N$.

Size $T(\mu(n))$ calculated with relation (5) is used in the exact algorithm called the BB-ALB algorithm, as a lower bound satisfying the axioms of a lower bound in a branch-and-bound algorithm.

In the BB-ALB algorithm, the partitions are enumerated lexicographically. For this, we should define the lexicographical precedence between the partitions.

We denote by $\mu = [B_1, B_2, ..., B_n, Q_{n+1}, ..., Q_N]$ a partial partition where $B_1, B_2, ..., B_n$ are sets of operations distributed respectively in stations $1, 2, ..., n$ and $Q_{n+1}, ..., Q_N$ sets unspecified in stations $n+1, ..., N$. From (5) let

$$t_n = \max_{a_j \in Q_n} (u_j + d_j) \quad Q_n = \bigcup_{k=1}^{n} B_k$$

be the term at which the last allocated operation ends.

It is determined similarly as in relation (3).

$$\Pi = \{a_j \in A \mid a_j \notin Q_n, \ u_j = t_n\}$$

which is ordered in an increasing order according to the duration of operations. Consider

$$\Pi = \{a_{j_1}, a_{j_2}, ..., a_{j_h}\} \tag{6}$$

with $u_{j_1} = u_{j_2} = ... = u_{j_h}$ and $d_{j_1} \leqslant d_{j_2} \leqslant ... \leqslant d_{j_h}$.

Operation a_{j_α} is distributed. In case that $F_n \geqslant d_{j_\alpha}$, where $F_n = C - \sum_{a_i \in Bn} d_i$, operation a_{j_α} is distributed in station n and $B_n' = B_n \cup \{a_{j_\alpha}\}$. If $F_n < d_{j_\alpha}$ consider $B_{n+1} = \{a_j\}$. Consequently one of the following two partitions is formed

$$\mu' = [B_1, B_2, ..., B_n', Q_{n+1}, ..., Q_N] \quad \text{or}$$

$$\mu'' = [B_1, B_2, ..., B_n, B_{n+1}, Q_{n+2}, ..., Q_N].$$

Definition 11. Partition μ precedes lexicographically partitions μ' or μ'', relatively at operation a_{j_α}.

The following algorithm, called the BB-ALB algorithm (Branch-and-Bound-an Assembly Line Balancing) is an implicite enumeration of the partial partitions in which size $T(\mu)$ given by (5) is a lower bound of the total

duration associated with the complete partitions obtained from the partial partition μ. According to theorem 2, duration $T(\mu)$ is used as a criterion of eliminating the partitions which are not potentially optimal.

The BB-ALB algorithm

Initializing. Determine a complete partition μ^* with the ELA algorithm corresponding to a scheduling program U^* and consider $f^* = T(\mu^*)$ the duration of scheduling. Consider $n = 1$, $\varepsilon = 1$, $t = 0$.

Step 1. Let $\mu = [B_1, B_2, ..., B_N, Q_{n+1}, ..., Q_N]$ be a partial partition formed on a certain step at moment t and according to (6) consider $\Pi = \{a_{j_1}, a_{j_2}, ..., a_{j_h}\}$. Consider $\alpha = 1$.

Step 2. The following prospective partition is formed

$\mu' = [B_1, B_2, ..., B'_n, Q_{n+1}, ..., Q_N]$ or

$\mu'' = [B_1, B_2, ..., B_n, B_{n+1}, Q_{n+2}, ..., Q_N]$ regarding operation a_{j_α} (μ lexicographically precedes μ' or μ'').

Calculate $f' = T(\mu)$ (or $f' = T(\mu'')$).

If $f' > f^*$ go to step 4.

Step 3. Specify operation a_{j_α} (thus B'_n or B_{n+1}) and consider the following time moment $t \leftarrow t + d_{j_\alpha}$ in case μ' or $t \leftarrow t + F_n + d_{j_\alpha}$ and $n \leftarrow n + 1$ in case of μ''. Consider $\mu \leftarrow \mu'$ (or $\mu \leftarrow \mu''$).

If μ is a complete partition go to step 6, otherwise go to step 1.

Step 4. $\alpha = h$ (h = the number of elements in Π) and go to step 5, otherwise consider $\alpha \leftarrow \alpha + 1$ and go to step 2.

Step 5. If $B_n = \varnothing$ consider $n \leftarrow n - 1$. If $n = 0$, STOP the last complete partition is the optimal solution. If a_{j_α} is the last operation distributed in station n, specify this operation, namely $B_n \leftarrow B_n - \{a_{j_\alpha}\}$. If $B_n \neq \varnothing$ consider $t \leftarrow t - d_{j_\alpha}$ and otherwise, if $B_n = \varnothing$ consider $t \leftarrow t - d_{j_\alpha} - F_{n-1}$ (by taking the free time $F_0 = 0$).

Determine $\Pi = \{a_{j_1}, a_{j_2}, ..., a_{j_h}\}$ consider $\alpha \leftarrow \alpha + 1$ and go to step 2.

Step 6. Consider $f^* = T(\mu')$, $\mu^* \leftarrow \mu'$ (or $f^* = T(\mu'')$, $\mu^* \leftarrow \mu''$). Consider $f^* \leftarrow f^* - \varepsilon$. Specify the last operation distributed by $B_n \leftarrow B_n - \{a_{j_\alpha}\}$ and go to step 1.

Numerical example.

Consider the set of operations $A = \{A, B, C, D, E, F, G, H, I, J\}$ described by graph in figure 1. The duration of the operations is written in the graph and the cyclic time of the line is $C = 15$.

By using the four rules for the distribution of the operations in stations at the ALB algorithm, the following partitions result.

1. Ordering the operations according to pair (t_i, d_i) where t_i is the minimum starting term and d_i the duration of operation a_i, $a_i \in A$. The production line:

$$\mu_1 = [\{A, B, E\}, \{C, D\}, \{F, G\}, \{H\}, \{I, J\}]$$

has 5 stations and the idle time is $R_5 = 3 + 4 + 1 + 7 + 0 = 15$.

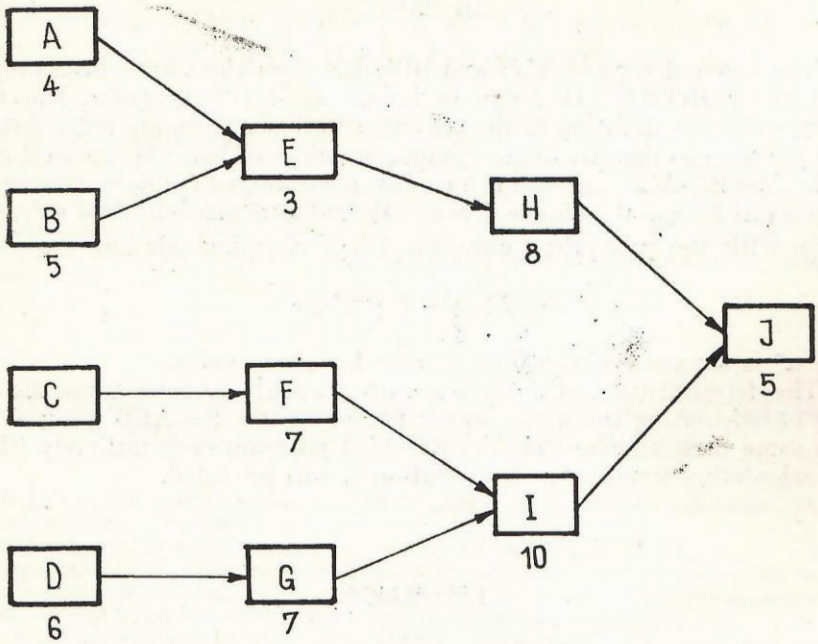


Fig. 1.

2. Ordering operations according to pair $(t_i \cdot t_i^1)$ where t_i' is the maximum starting term of operation $a_i \in A$. The production line:

$$\mu_2 = [\{D, C, A\}, \quad \{F, G\}, \quad \{B, E\}, \quad \{I\}, \quad \{H, J\}]$$

has 5 stations and the idle time $R_5 = 0 + 1 + 7 + 5 + 2 = 16$.

3. Ordering operations according to pair $(t_i \cdot t_i^*)$ where t_i^* is the maximum finishing term of operation $a_i \in A$. The production line:

$$\mu_3 = [\{C, D, A\}, \quad \{B, F, E\}, \quad \{G, H\}, \quad \{I, J\}]$$

has 4 stations and the idle time $R_4 = 0$.

4. Ordering operations according to pair (t_i, ν_i) where $\nu_i = t_i^* - t_i - d_i$ is the time reserve for operation $a_i \in A$. Production line:

$$\mu_4 = [\{D, C, A\}, \quad \{F, G\}, \quad \{B, I\}, \quad \{E, H\}, \quad \{J\}]$$

has 5 stations and the idle time $R_5 = 15$.

It is noticed that partition μ_3 is just the optimal solution.

Another optimal solution which is determined with the BB-ALB, algorithm is

$$\mu_5 = [\{A, B, D\}, \quad \{E, C, G\}, \quad \{F, H\}, \{I, J\}]$$

with 4 stations and the idle time $R_4 = 0$.

For partition μ_5 the scheduling programme is the following:

$$U_5 = (u_A = 0, \; u_B = 4, \; u_D = 9, \; u_E = 15, \; u_C = 18,$$
$$u_G = 23, \; u_F = 30, \; u_H = 37, \; u_I = 45, \; u_J = 55).$$

CONCLUSIONS

The above-described ALB and BB-ALB algorithms have been programmed in the FORTRAN IV language for the IBM 370 computer. Due to the lexicographical exploration of the feasible solutions, there are not constraints in the program as regards the maximum number of feasible sets that can be stored. The BB-ALB program has the possibility to obtain the optimal solutions (all for $\varepsilon = 0$ and one for $\varepsilon = 1$) and can also obtain a suboptimal solution with the scheduling duration $T(\mu)$ verifying relation

$$| T(\mu) - T(\mu^*) | \leqslant \varepsilon$$

where μ^* is the optimal solution, and $\varepsilon > 1$ a given value.

The determination of a solution with the ALB program gives the possibilitiy of calculating the initial upper bound in the BB-ALB program and at the same time, in case that the BB-ALB program ends untimely (the expired calculating time) a feasible solution is still provided.

REFERENCES

1. BOWMAN, E.H.: *Assembly Line Balancing by Linear Programming*, Operations Research, 8(3), 385—389, (1960).
2. BRENNECKE, D.: *Two Parameter Assmebly Line Balancing Model*, 215—235 M.P. Hottenstein: Models and Analysis for Production Management, Pennsylvania, 1968.
3. GEHRLEIN, W. V. PATTERSON, J.H.: *Sequencing for Assembly Lines with Integer Task Times*, Management Science 21(9), 1064—1070, (1975).
4. GUTJAHR, A.L. NEMHAUSER, G.L.: *An Algorithm for the Line Balancing Problem*. Management Science 11.2, 308—315 (1964).
5. HELD, M., KARP, R.M., SHARESHIAN, R.: Assembly Line Balancing-Dynamic Programming with Precedence Constraints. Operations Research 11(3), 442—459, (1963).
6. HOFMAN, T.R.: *Assembly Line Balancing with a Precedence Matrix*. Management Science 9(4), 551—562, (1963).
7. JACKSON, J.R.: *A Computing Procedure for a Line Balancing Problem*. Management Science 2(3) 261—271, (1956).
8. KILBRIDGE, M.D., WESTER, L.: *A Review of Analytical Systems of Line Balancing*, Operations Research 10(5), 626-638 (1962).
9. MANSOOR, E.M., YADIN M.: *On the Problems of Assembly Line Balancing*. Developments in Operations Research ed. Benjamin Avi-Itzhak, vol. II, 361—375, (1971).
10. MASTOR, A.A.: *An Experimental Investigation and Comparative Evaluation of Production Line Balancing Techniques*, Management Science 16(11), 728—746 (1970).
11. NEVINS, A.J.: *Assembly Line Balaning using best bud Search*. Management Science 18(9), 529—539, (1972).
12. PATTERSON, J.H., ALBRACHT, J.J.: *Assembly Line Balancing Zero-one Programming with Fibonnci Search*, Operations Research 23(1), 166—172, (1975)
13. REITER, R.: *On Assembly Line Balancing Problems*. Operations Research 17(4), 685—700, (1969).
14. SHKURBA V.V., BELETSKY, S.A.: *Numerical Methods to Solve the Problem on the Assembly Line Balancing*, Kibernetika 1, 96—108, 1977.

UN ALGORITHME BRANCH-AND-BOUND POUR RESOUDRE L'EQUILIBRE D'UNE LIGNE D'ASSEMBLAGE

L'auteur présente un algorithme branch-and-bound pour résoudre le problème de l'équilibre d'une ligne d'assemblage.

Prenant comme point de départ la notion d'ensemble admissible, introduite par Gutjahr et Nemhauser, on explore, à l'aide de l'énumération lexicographique, le domaine des partitions réalisables de l'ensemble des opérations dans les stations. On associe à une partition un programme d'ordonnancement. Pour éliminer les partitions qui, de façon potentielle ne sont pas optimales, on définit une limite inférieure de la durée de l'assemblage correspondant à un programme d'ordonnancement.

АЛГОРИТМ ВЕТВЕЙ И ГРАНИЦ ДЛЯ РЕШЕНИЯ ПРОБЛЕМЫ БАЛАНСИРОВАНИЯ СБОРОЧНОЙ ЛИНИИ

В работе показан алгоритм „ветвей и границ" для решения проблемы сбалансированности сборной линии.

Основываясь на понятии допустимого множества, внедрённом Гутьяром и Немхаусером, при лексикографическом перечислении, исследуется область реализуемых партиций множества операций в станциях. Одной партиции ассоциируется одна программа эшелонирования операций. Для исключения партиций, потенциально не являющихся оптимальными, определяется нижний предел продолжительности сборки, соответствующей программе эшелонирования операций.

27. IX. 1977