BI – DIRECTIONAL WORK SHARING IN ASSEMBLY LINES
UNDER STRICT ORDER OF TASKS

Y. Bukchin and T. Cohen
Department of Industrial Engineering
Tel Aviv University, Tel Aviv 69978, Israel

ABSTRACT

Sharing of tasks between stations in assembly lines can reduce cycle time and improve throughput rate. This research deals with an assembly line in which tasks may be shared with their immediate upstream or downstream stations.

A perfect balance of an assembly line is obtained if the same workload is assigned to all stations. If the perfect balance of a line can be obtained via task sharing, the line is defined as balanceable. First, the conditions for balanceability of lines are investigated, assuming that all tasks can be shared. The identity of the tasks to be shared to this end is then given. Next, we address a situation where only some of the tasks can be shared, either downstream or upstream and find the optimal balance with respect to the cycle time objective. Finally a model formulation along with an optimal algorithm to minimize the cycle time under a given limited budget, are developed for the cases of strict precedence relationships among tasks. Experimentation to evaluate the algorithm performance is conducted.

1. INTRODUCTION

In this paper an approach for improving throughput rate of a non-balanced working assembly lines, called task sharing, is discussed. When applying task sharing, tasks are performed in more than one station in different cycles, resulting in a lower cycle time on the average. Next we analyze the design phase of work sharing when determining some properties of the problem and solution approaches for determining the identity of the shared task.

1.1 Research Objective

The main objective of this research is to analyze work-sharing when tasks can be performed by the adjacent upstream or downstream station. We first intend to characterize the conditions for balanceability of lines, namely, its ability to be perfectly balanced via sharing. These conditions should help to identify the shared tasks and the time proportion they should
be performed in each station, to minimize cycle time. The second aim is to incorporate sharing costs in the model and develop a design algorithm for such an environment. We assume that sharing tasks involves a monetary aspect, such as workers training or tools duplication.

1.2 Literature Review

Ostolaza et al. [1] were the first to present the term of dynamic line balancing (DLB). This term refers to the operational side of work sharing as "allowing tasks to be assigned 'on the fly' based on the current state of system". The basic idea is shifting additional capacity from low utilization stations to high ones by allowing different workers to perform the same task. This approach can be applied in existing lines to reduce idle time, increase flexibility and reduce risk of injury caused by repetitive work. Ostolaza et al. [1] used a dynamic control on the buffer inventory quantities between every two stations to decide in which station the shared task will be done. Another approach of DLB named Bucket Brigade (BB) was presented by Bartholdi and Eisenstein [2]. They suggested that the concept of "line that balances itself" by proposing a mechanism that assigns the work dynamically to the workers. Anuar and Bukchin [3] suggested analytical conditions for line balanceability in lines where forward sharing is allowed (a task can be done by its current or the adjacent downstream station). They proposed several tools for the design and operation of such assembly line.

2. LINE WITH FULL SHARING

2.1 Problem description

Let \( N \) denotes the number of stations and \( F_i \) is the fixed time of the station \( i \), namely time which cannot be shared. \( S_i^d (S_i^u) \) is the time which can be shared with the downstream (upstream) adjacent station. Clearly, \( S_1^u \equiv 0 \) and \( S_N^d \equiv 0 \) since station 1 (\( N \)) cannot shared time with its upstream (downstream) station. The cycle time of the line which is determined by the most loaded station, or the bottleneck station is denoted by \( c \). \( p_i (q_i) \) is the proportion of cycles that station \( i \) \((i = 1, \ldots, N)\) performs the shared time task \( S_i^d (S_i^u) \), where \( 0 \leq p_i \leq 1 \) \((0 \leq q_i \leq 1)\). Therefore, the proportion that station \( i+1 \) performs task \( S_i^d \) is \( 1 - p_i \), and the proportion that station \( i - 1 \) performs task \( S_i^u \) is \( 1 - q_i \). \( T_i \) is the total station time of station \( i \).
(i = 1, ..., N) and $T$ is the ideal average time in each station:

$$T = \frac{\sum_{i=1}^{N} T_i}{N} = \frac{\sum_{i=1}^{N} (S_i^u + F_j + S_i^d)}{N}.$$

**Definition 1**: A line is balanceable if each station can have an average workload of $T$ units of time using work sharing tasks between consecutive stations.

### 2.2 Condition for balanceability

**Theorem 1**: If all the assembly time can be shared in consecutive station (full sharing), the conditions for balanceability are:

$$T \leq T_1 + T_2$$

$$\sum_{j=1}^{i-1} T_j \leq T \leq \sum_{j=1}^{i+1} T_j. \quad 2 \leq i \leq N - 1 \quad (1)$$

**Proof**: See [4].

A sequential algorithm, based on the above condition, is proposed below for setting the sharing proportions to achieve a perfectly balanced line.

**Algorithm A1**

**Step 1**: Set $i = 1$ and $T_i' = T_i$

**Step 2**: If $i = N$ - the line is balanceable – exit.

If $T_i' > T$, transfer $(T_i' - T)$ from station $i$ to station $i + 1$.

If $T_i' - T > T$, the line is non-balanceable – exit.

Update the time left in station $i + 1$: $T_{i+1}' = \sum_{j=i}^{i+1} T_i' - iT$.

Otherwise, if $T_i' \leq T$, transfer $(T - T_i')$ from station $i + 1$ to station $i$.

If $T_{i+1}' < T - T_i'$ - the line is non-balanceable – exit.

Update the time left in station $i + 1$: $T_{i+1}' = \sum_{j=i}^{i+1} T_i - iT$.

**Step 3**: Set $i = i + 1$, Go to step 2.

### 2.2 Solution for non-balanceable lines

When the line is non-balanceable the cycle time and proportion of work-sharing can be found by the following Linear Programming (LP) formulation [P1].

**[P1]**

\[
\text{Min}(c) \quad \text{Subject – to} \\
c \geq p_i T_i + (1 - q_2)T_2 \\
c \geq (p_i + q_i - 1) T_i + (1 - q_{i+1}) T_{i+1} + (1 - p_{i+1}) T_{i+1} \\
c \geq (1 - p_{N-1}) T_{N-1} + q_N T_N
\]
\[
0 \leq p_i \leq 1 \quad \text{for } i = 1, \ldots, N-1
\]
\[
0 \leq q_i \leq 1 \quad \text{for } i = 2, \ldots, N
\]
\[
(1 - p_i) + (1 - q_i) \leq 1 \quad \text{for } i = 2, \ldots, N-1
\]

Unfortunately, LP solution may yield \( p_i < 1 \) and \( q_{i+1} < 1 \) simultaneously, thus, two tasks are shared while sharing only one task could reach the same cycle time. Although reducing the number of shared tasks may be formulated as a Mixed Integer Linear Programming (MILP), the solution with minimal shared tasks can be obtained by the following algorithm, which modifies the solution of P1. The new sharing proportions change from \( p_i \) and \( q_i \) to \( \overline{p}_i \) and \( \overline{q}_i \) respectively.

**Algorithm A2:**

**Step 1:** Solve [P1]. Set \( i = 1 \)

**Step 2:**
- If \( i = N \) - exit. If \( p_i = 1 \) or \( q_{i+1} = 1 \), go to step 3
- Else, if \( T_i (1 - p_i) = T_{i+1} (1 - q_{i+1}) \), then \( \overline{p}_i = 1 \) and \( \overline{q}_{i+1} = 1 \), go to 3
- Else, if \( T_i (1 - p_i) > T_{i+1} (1 - q_{i+1}) \), then \( \overline{q}_{i+1} = 1 \) and \( \overline{p}_i = \frac{T_i p_i + T_{i+1} (1 - q_{i+1})}{T_i} \),
  go to step 3
- Else, if \( T_i (1 - p_i) < T_{i+1} (1 - q_{i+1}) \), then \( \overline{p}_i = 1 \) and \( \overline{q}_{i+1} = \frac{T_{i+1} q_{i+1} + T_i (1 - p_i)}{T_{i+1}} \),
  go to step 3

**Step 3:** \( i = i + 1 \) and start step 2 again

3. **LINE WITH FIXED AND SHARED TIME**

In this section a more realistic situation is addressed, where only part of the time in each station can be shared. Consequently, the fixed time is defined in each station along with the time that can be shared either forward or backward. Due to the strict order of the tasks, in each station \( j \) the time which can be shared backward, \( S_j^d \), is followed by the fixed time, \( F_j \), which is followed by the time which can be shared forward, \( S_j^u \).

3.1 **Conditions for balanceable lines**

**Theorem 2:** A line is balanceable if the following conditions hold:

\[
\frac{F_i + S_i^d + \sum_{j=i+1}^{N} (F_j + S_j^d + S_j^u)}{N - i + 1} \leq T \quad \text{for } i = 1, \ldots, N \tag{2}
\]
\[
T \leq \frac{S_{i+1}^d + \sum_{j=1}^{N} (F_j + S_j^u + S_j^d)}{N - i + 1} \quad \text{for } i = 1, \ldots, N \tag{3}
\]

**Proof:** See [4].
3.2 Solution for non-balanceable lines

When the line is non-balanceable the cycle time and proportion of work-sharing can be found by the following formulation, P2.

\[
\text{Min } c \\
\text{Subject to} \\
c \geq F_i + p_i S_i^d + (1 - q_i) S_i^u \\
c \geq F_i + (1 - p_{i-1}) S_{i-1}^d + p_i S_i^d + q_i S_i^u + (1 - q_{i+1}) S_{i+1}^u \\
c \geq F_N + (1 - p_{N-1}) S_{N-1}^d + q_N S_N^u \\
0 \leq p_i \leq 1 \\
0 \leq q_i \leq 1
\]

Based on the solution of P2 an Algorithm similar to A2 (see [4]) can be applied here to minimize the number of shared tasks.

4. CYCLE TIME MINIMIZATION SUBJECT TO LIMITED SHARING COST

4.1 Model Formulation

When sharing costs are associated with the tasks, we aim in finding an optimal solution that minimizes the cycle time as a primary objective and the actual amount of sharing costs as a secondary objective. Let \( S_i \) denote the station to which task \( i \) is initially assigned, \( i \in 1, ..., m \), \( \sigma_j \) is the set of tasks assigned initially to station \( j \), \( j = 1, ..., N \), and \( L_j \ (F_j) \) is the last (first) task assigned in station \( j \). Let \( T_i \) and \( C_i \) denote the processing time and sharing time of task \( i \), respectively. The total budget available for sharing is denoted by \( TAR \). The following variables are needed for the mixed-integer linear model, presented next. \( ac_i \) is a decision variable which equals \( C_i \) if task \( i \) is shared and 0 otherwise. Note that if the task is transferred entirely to the adjacent station, no sharing cost occurs. \( x_i \) is equal to 1 if task \( i \) is shared partially or completely with the downstream station, and 0 otherwise. \( y_i \) is equal to 1 if task \( i \) is shared partially or completely with the upstream station, and 0 otherwise. \( z_i \) is equal to 1 if task \( i \) is transferred completely to the downstream station, and 0 otherwise. \( w_i \) is equal to 1 if task \( i \) is transferred completely to the upstream station, and 0 otherwise.
The MILP formulation is shown in [P3].

\[\begin{align*}
\text{Min} & \quad M \cdot c + \sum_i ac_i \\
\text{Subject to} & \quad c \geq \sum_{i \in \sigma, j} (1 - p_j) T_i + \sum_{i \in \sigma} (p_i + q_i - 1) T_i + \sum_{i \in \sigma, j, i} (1 - q_i) T_i \quad j = 2..N - 1 \\
& \quad c \geq \sum_{i \in \sigma} p_i T_i + \sum_{i \in \sigma, x} (1 - q_i) T_i \\
& \quad c \geq \sum_{i \in \sigma, x, i} (1 - p_i) T_i + \sum_{i \in \sigma} q_i T_i \\
& \quad 1 - p_i \leq x_i \quad i = 1..m \\
& \quad M(1 - p_i) \geq x_i \quad i = 1..m \\
& \quad p_i + 1 \leq x_i \quad \forall i, i + 1 \in \sigma, \forall j \in J \\
& \quad 1 - M p_i \leq z_i \quad i = 1..m \\
& \quad 1 - p_i \geq z_i \quad i = 1..m \\
& \quad 1 - q_i \leq y_i \quad i = 1..m \\
& \quad M(1 - q_i) \geq y_i \quad i = 1..m \\
& \quad q_i + 1 \leq y_i + 1 \quad \forall i, i + 1 \in \sigma, \forall j \in J \\
& \quad 1 - M q_i \leq w_i \quad i = 1..m \\
& \quad 1 - q_i \geq w_i \quad i = 1..m \\
& \quad x_j + y_{j+1} \leq 1 \quad \forall j = 1,...,N - 1 \\
& \quad x_i + y_j \leq 1 \quad i = 1..m \\
& \quad ac_i \geq C_i - (1 - x_i) M - z_i M \quad i = 1..m \\
& \quad ac_i \geq C_i - (1 - y_i) M - w_i M \quad i = 1..m \\
& \quad ac_i \geq 0 \quad i = 1..m \\
& \quad TAR \geq \sum ac_i \\
& \quad 0 \leq p_i \leq 1 \quad \forall i = 1,...,N - 1 \\
& \quad 0 \leq q_i \leq 1 \quad \forall i = 1,...,N \\
& \quad x_i \in \{0,1\}, \quad y_i \in \{0,1\}, \quad z_i \in \{0,1\}, \quad w_i \in \{0,1\} \quad i = 1..m \\
\end{align*}\]

Definition 2: A Bottleneck Segment (BNS) is a set of consecutive stations that satisfy the following conditions: (1) each of the stations is the most loaded station in the line; (2) the first (last) station in the set does not share tasks with the upstream (downstream) station.

4.2 Bottleneck Search Algorithm (BNSA)

An optimal solution for P[3] can be obtained by using the following algorithm.

Algorithm BNSA

Step 1. Initialization. Calculate the cycle time when all tasks are fixed. Let \(TAR\) be the total budget for sharing. Define the above problem as the new parent node.
Step 2. Creation of descendant. Identify the first BNS in the line of the parent node.
   a. If the first task of the BNS is not the first task in the line – define the task as shared while constructing a new node.
   b. If the last task of the BNS is not the last task in the line – define the task as shared while constructing a new node.
   c. If no descendants were created, exit: the line is balanceable within TAR.

Step 3. Duplication elimination. If there is a created node in the same level of the tree with the same marked tasks as the nodes created – combine the nodes.

Step 4. Cycle time calculation. For each newly created node, calculate the cycle time using formulation P2 and algorithm similar to A2 (see [4]). Calculate the sharing cost of the cycle time found. If the sharing cost is larger than TAR set cycle time to infinite.

Step 5. Parent selection. Define the node with the smallest cycle time as the parent node and go to step 2. If all nodes have infinite cycle time – no more improvement in cycle time can be done within the given budget – exit.

We have compared the performance of the MILP (using OPL-Studio© package) and the BNSA algorithm (coded in Visual Basic©) via the number of nodes created over 192 experiments. The following factors were considered: A - Length of line, B - average number of tasks per station in initial assignment, C – variance of the task time, D - variance of task sharing cost and E - level of budget available (TAR) for sharing. In short lines and relatively low budget the BNSA generated always fewer nodes than the MILP. In long lines, on the other hand, the MILP has an advantage over the algorithm except for cases where the budget was relatively low. The statistics are given in Table 1.

<table>
<thead>
<tr>
<th></th>
<th>BNSA</th>
<th>MILP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>7,980.08</td>
<td>62,813.84</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>17,909.61</td>
<td>263,391.01</td>
</tr>
<tr>
<td>Maximum</td>
<td>88,336</td>
<td>2,345,800</td>
</tr>
</tbody>
</table>

The probability that BNSA is better than MILP as a function of the available budget for work-sharing, based on the empirical distribution is presented in the Figure 1. This graph demonstrates the results indicated above, as the algorithm outperforms the MILP for relatively low budget available. The budget available is calculated as a percentage of the sum.
of all cost sharing. Note that the difference is less significant for very low budget; the reason is that in such cases the problem is easy to solve in both solution approaches.

![Figure 1](image_url)

**REFERENCES**


