Design and operation of dynamic assembly lines using work-sharing

ROUIE ANUAR and YOSSI BUKCHIN*

Department of Industrial Engineering, Faculty of Engineering, Tel-Aviv University, Tel-Aviv 69978, Israel

Dynamic line balancing (DLB) deals with dynamic shifting of assembly tasks (shared tasks) between consecutive stations during the operation of the assembly line. In traditional assembly lines, work-sharing is not allowed. Consequently, the cycle time is determined by the amount of work performed in the most loaded station. The DLB approach enables achieving a shorter cycle time and a higher throughput rate, as compared to the traditional lines. In the implementation of the DLB approach the following has to be determined: (1) the identity of the shared tasks, and (2) the operational task allocation rule. In some cases, the data regarding the percentage of cycles each shared task will be performed in a station is also required. We first analyse a case in which an initial task assignment to stations (line balance solution) and the identity of the shared tasks is known and given. Conditions for the balanceability (the ability of the DLB to achieve an equivalent performance to a perfect balanced line in the steady state) are developed, as well as the formulation for minimizing the cycle time. Next, the sharing costs are considered and approaches for minimizing the shared time and sharing cost are presented, based on the new concept of bottleneck segment. The operational aspect is also addressed, while examining the effect of several state-dependent and state-independent operating rules for task allocation on the cycle time and the system work in process.

Keywords: Work-sharing; Assembly lines; Dynamic line balancing; Cross training

1. Introduction and literature review

The traditional approach for assembly line balancing consists of allocating tasks to workstations, while trying to balance the load among the different stations as much as possible. This combinatorial problem has been widely addressed in the literature, and many optimal and heuristic algorithms have been suggested (see literature reviews in Baybars 1986, Ghosh and Gagnon 1989, Scholl and Becker 2006). However, even an optimal solution, which is hard to achieve due to the nature of the problem, cannot guarantee a perfect balance in which an equal time is assigned to all stations. Consequently, the throughput rate is determined by the most loaded station, while other stations suffer from idle times.

*Corresponding author. Email: bukchin@eng.tau.ac.il
The DLB approach, presented in Ostolaza et al. (1990), suggests shifting of tasks between consecutive stations during the operation of the line and thus enabling a dynamic change of the stations’ load. The concept of work-sharing, which refers to the ability to perform a certain task (a shared task) in more than one station/worker, is the basis of DLB operation, and cross-trained workers are required to this end. Consequently, a shared task is defined as a task that can be performed last in some station, \( i \), or first in the downstream station, \( i + 1 \).

The dynamic line-balancing problem consists of the design and operation stages. The design stage involves the decision regarding the identity of the shared tasks in each station. The operational problem deals with the dynamic decision regarding the station to perform the shared task in each cycle. In some cases, the \textit{a priori} data regarding the percentage of cycles each shared task will be performed in a station is also required.

The design issue is scarcely addressed in the literature. Ostolaza et al. (1990) suggests using 30–40% of the total line operation time as shared time for exponentially distributed operation times. McClain et al. (1992) suggested using 20–30% of the total line operation time as shared time when the coefficient of variation of the operation times ranges from 0.3 to 0.6. Neither of these approaches suggests the exact amount of the shared time and the identity of the tasks to be shared.

Concerning the operation of the line, there exist various approaches for dividing the load efficiently among stations using work-sharing. Hopp and Van Oyen (2004) divide the different operational approaches into the following five categories. In the \textit{Scheduled Rotation}, the workers assignment to tasks is changed, either by a predetermined pattern of rotation or by local managerial decisions. The former may take into consideration attributes such as the worker speed and buffer capacities. The \textit{Worker Prioritized Worksharing} approach allocates workers to tasks based on a prioritization of workers, for example, worker performance speed. The bucket brigade policy of Bartholdi and Eisenstein (1996) belongs to this category. With bucket brigade, when the design is preemptive (item can be passed between workers during an operation), operation times are deterministic and workers are allocated to stations from slowest to fastest, a spontaneous balance emerges with no buffers needed. Allowing preemption will increase the system’s efficiency and reduce idle time via work-sharing; however, due to the nature of the tasks and the assembly environment, preemption is not always possible. Moreover, when worker speeds are nearly equal (McClain et al. 2000), or differ dramatically (Villalobos et al. 2003), the performance of bucket brigades worsens. In \textit{Floating Workers} policy some workers in the line are defined as specialists/experts, and may roam freely along the line and provide assistance where needed. The D-skill chaining suggested in Hopp et al. (2004), may belong to this category. In \textit{Zoned Worksharing}, some machines/tasks on the line are shared between workers. A distinction between the non-overlapping and overlapping zones is mentioned as the latter is much more addressed in the literature (see for example, Ostolaza et al. 1990, McClain et al. 1992, Bischak 1996, Zavadlav et al. 1996). The last approach mentioned in Hopp and Van Oyen (2004) is the \textit{Craft} in which workers are fully trained to do all tasks. This approach can be implemented by individual workers or in teams (Van Oyen et al. 2001).

The main contribution of this paper is suggesting a comprehensive approach to apply DLB using work-sharing on a \textit{given} traditional balancing solution to improve
its throughput rate by reducing the cycle time. The balancing solution, which may be associated with an operational working line, includes the task assignment to stations along with the given sequence of tasks within each station. The proposed approach consists of an analytical analysis of work-sharing followed by implementation algorithms which take into account the tradeoff between the improved systems performance due to work-sharing and the additional costs due to employee training, equipment duplication and additional complexity of line operation. Several operating rules are compared via simulation, and recommendations for operational implementation of the proposed approach are given.

The rest of the paper is organized as follows. We first analyse the given balancing solution, in section 2, using the balanceability concept, and draw some properties of balanceable and non-balanceable lines. The balancing proportion, which is defined as the percentage of cycles the shared task has to be done in a certain station, is then discussed, assuming that the shared tasks are already given. The balancing proportions are obtained using a closed-form solution and via an LP solution when the former is not possible. The obtained balancing proportions are those, which result in leveling or balancing the load among the stations in the steady state as much as possible. The assumption regarding the given shared tasks is relaxed in section 3, and the selection process of tasks to be shared is then considered. Given the extra cost associated with sharing tasks, we suggest algorithms for selecting the minimal set of tasks to share for reaching the optimal or a desired cycle time. Using these algorithms, we also address aspects of cost and revenue when using DLB. The operational issue associated with the implementation of work-sharing is addressed in section 4, and some concluding remarks are given in section 5.

2. Load balancing using work-sharing

In this section, the load balancing is discussed, assuming that the division of work in each station into fixed and shared component is already given. We distinguish between a case where a perfect balance of equal load in all stations can be obtained in the steady state and cases where such a solution is not attainable.

The proposed model deals with a single-model assembly line in a tandem queuing network environment. The line is assumed to be un-paced with buffers between stations, the first station never starves and the last is never blocked. One can expect that the advantages of work-sharing will be better utilized in a buffered line, due to the additional variability in the stations’ which may cause blockage and starvation. Hence, in sections 2 and 3 an unlimited buffer size is assumed; in section 4, the affect of the buffer size on the cycle time is examined, along with a recommendation regarding its size in practice. We also assume that the allocation of tasks to stations (a balancing solution) is given, where each Station $i$ contains an assembly time, $TST_i$, $i=1, \ldots, N$. The work-sharing concept is materialized by dividing the station time, $TST_i$, into two elements: the fixed time, $F_i$, and the shared time, $S_i$, where, $F_i + S_i = TST_i$. The fixed time is performed in Station $i$ for all the products/cycles, however, in each cycle, the shared time can be performed either in Station $i$ or in Station $i+1$. In the steady state of operating the line with work-sharing, we use the term of balancing proportion of station $i$, denoted by $p_i$, to express the percentage of
cycles in which the shared time, $S_i$, is performed in station $i$. The percentage of cycles the shared time is performed in Station $i+1$ is clearly $(1 - p_i)$.

When applying work-sharing, one should determine the amount of shared work in each station, as well as the value of the balancing proportion parameter, $p_i$. In this section we assume that the identity of the shared tasks is known and given and focus on finding the balancing proportions, $p_i$, which yields the minimum cycle time in the steady state. This assumption is relaxed later on.

### 2.1 Balanceable lines

The minimum theoretical cycle time that can be pursued in the line, $T$, is equal to the total assembly time, $\sum_{i=1}^{N} TST_i$, divided by the number of stations. Nevertheless, in most cases this cycle time cannot be obtained via traditional line balancing techniques. The work-sharing concept suggests shifting tasks between consecutive stations in order to obtain, on the average, equal load, $T$, in all stations. This result is not always feasible, as can be seen below.

For a given balancing solution, with predetermined fixed and shared tasks, a balanceable line is defined as a line for which the load can be equally divided among the station using work-sharing, resulting in a minimal average cycle time value, $T$. Without loss of generality, as shown later on, we assume that the shared time in each station constitutes a single shared task which can be done either at the current station or at the downstream neighbouring station.

Next, we have to find the value of $p_i$, which obtain a perfect balance of the load in the line. Consequently, $p_i$ should satisfy

$$F_i + (1 - p_{i-1})S_{i-1} + p_i S_i = T, \quad i = 1, \ldots, N$$

where, $p_0$, $p_N$, $S_0$ and $S_N$ are equal to zero by definition.

The balancing equation (1), expresses the total work time assigned to Station $i$, where, $F_i$ represents the time of the fixed task always done by Station $i$, $(1 - p_{i-1})S_{i-1}$ denotes the amount of the shared task of Station $i-1$, done in Station $i$, and $p_i S_i$ is the part of the shared time of Station $i$, which is done in Station $i$. In order to obtain a perfect balance, the summation of these components should be equal to the minimum theoretical cycle time, $T$, where

$$T = \frac{\sum_{i=1}^{N} (F_i + S_i)}{N}. \quad (2)$$

The solution of the above linear system is

$$p_i = \sum_{j=i}^{N-1} \frac{(F_{j+1} + S_j - T)}{S_i}, \quad 1 \leq i \leq N - 1. \quad (3)$$

By examining the solution of the balancing equation, we can determine whether the line is balanceable. In particular, the line is balanceable if there exists a feasible solution $p_i$ where

$$0 \leq p_i \leq 1, \quad 1 \leq i \leq N - 1 \quad (4)$$
or,
\[
0 \leq \sum_{j=i}^{N-1} \frac{(F_{j+1} + S_j - T)}{S_i} \leq 1, \quad 1 \leq i \leq N - 1. 
\]  
(5)

From the right hand side of the inequality (5), we get
\[
\frac{\sum_{j=i+1}^{N} (F_j + S_j)}{(N - i)} \leq T, \quad 1 \leq i \leq N - 1.
\]  
(6)

Defining \( T_i \) as the average station time over stations \( i \) to \( N \), namely,
\[
T_i = \frac{\sum_{j=i}^{N} (F_j + S_j)}{(N - i + 1)}, \quad 1 \leq i \leq N, 
\]  
(7)

we get
\[
T_{i+1} \leq T, \quad 1 \leq i \leq N - 1, \text{ or } T_i \leq T, \quad 2 \leq i \leq N. 
\]  
(8)

From the left hand side of inequality (5), using (7) we get
\[
T_i \geq T - \frac{S_{i-1}}{(N - i + 1)}, \quad 2 \leq i \leq N.
\]  
(9)

Combining (8) and (9), we can conclude that a line is balanceable if
\[
T - \frac{S_{i-1}}{(N - i + 1)} \leq T_i \leq T, \quad 2 \leq i \leq N.
\]  
(10)

or,
\[
0 \leq T - T_i \leq \frac{S_{i-1}}{(N - i + 1)}, \quad 2 \leq i \leq N. 
\]  
(11)

Note that inequality of (9) indicates the minimal amount of shared needed to assure balanceability, as \( S_{i-1} \geq (N - i + 1)(T - T_i) \) for \( i = 2, \ldots, N \), or \( S_i \geq (N - i)(T - T_{i+1}) \) for \( i = 1, \ldots, N - 1 \). This property is used later on in section 3.

2.2 Non-balanceable lines

When the line is not balanceable the closed form solution of the balanceable case does not hold. Nevertheless, using work-sharing, one can improve the cycle time significantly over the traditional solution, even for non-balanceable lines. In such
cases, the following linear formulation, P1, should be solved, where $Z$ is the desired cycle time.

\[
\begin{align*}
\text{[P1]} \\
\text{Min}(Z) \\
\text{Subject-to:} \\
Z & \geq F_i + (1 - p_{i-1})S_{i-1} + p_i \cdot S_i, \quad 1 \leq i \leq N \\
p_i & \leq 1, \quad 1 \leq i \leq N - 1 \\
p_i & \geq 0, \quad 1 \leq i \leq N - 1 \\
p_0 = 0; \quad p_N = 0.
\end{align*}
\]

Note that when the $S_i$ values are set to the total station time (and $F_i$ values are set to zero), the solution of P1 yields the minimal possible cycle time using work-sharing. Nevertheless, this cycle time may be obtained for fewer shared tasks. This issue is discussed in the next section.

3. A cost model for work-sharing

When implementing work-sharing in an assembly line, one should take into consideration that shared tasks may involve additional costs associated with employee training, equipment duplication and additional complexity of operating the line. Consequently, the tradeoff between the amount of shared time and the line performance should be addressed. In this section we discuss the following cost oriented problem types:

1. DLB-Type I problem—minimizing the amount of shared task time/cost for a given cycle time.
2. DLB-Type II problem—minimizing the cycle time as a primary objective and the amount of shared time/cost, as a secondary objective, subject to a limited time/cost of shared tasks.
3. DLB-Type III problem—maximizing the total profit associated with the throughput rate and the sharing cost.

Recall that in all the above problems, the throughput rate of a given balancing solution, determined by the most loaded station, is improved by using work-sharing. For simplicity, we assume that a strict sequence of the tasks within each station is given. Consequently, sharing tasks within a station should follow this sequence, while starting with the last task and ending with the first.

These three problem types are discussed next.

3.1 Minimizing the shared time/cost (DLB-Type I problem)

The solution of P1 for $S_i = TST_i$ and $F_i = 0$ for all $i$ results in the minimum theoretical cycle time. Nevertheless, this cycle time may be attained for a lesser amount of shared time. Moreover, in some cases, due to marketing or other considerations, the desired cycle time is larger than the minimum theoretical
cycle time, and a question arises regarding the minimal shared time required to obtain the above cycle time.

When the minimal amount of shared time is required to achieve a cycle time of $T$ for a balanceable line, the solution can be obtained simply by using equation (9) and allocating the minimal amount of shared time required to satisfy (9), namely,

$$ S_i \geq (T - T_i)(N - i) \quad \text{for all } i. $$

Since a task cannot be split, for each Station $i$ we will share tasks starting from the last task of the station, until inequality (9) is satisfied.

If the value of $S_i$ is larger than the possible station time, $TST_i$, the line is not balanceable. In this case, P1 can be modified to address the minimal shared time problem resulting in P2.

$$ \begin{align*}
[\text{P2}] \\
& \text{Min} \left( \sum_{i=1}^{N-1} S_i \right) \\
\text{Subject-to:} \\
& Z^* \geq F_i + (1 - p_{i-1})S_{i-1} + p_i \cdot S_i, \quad 1 \leq i \leq N \\
& F_i + S_i = TST_i, \quad 1 \leq i \leq N \\
& F_i \geq 0, \quad 1 \leq i \leq N \\
& S_i \geq 0, \quad 1 \leq i \leq N - 1 \\
& p_i \leq 1, \quad 1 \leq i \leq N - 1 \\
& p_i \geq 0, \quad 1 \leq i \leq N - 1 \\
& S_0 = 0; S_N = 0; p_0 = 0; p_N = 0.
\end{align*} $$

The desired cycle time is denoted by $Z^*$. Note that in case the minimum theoretical cycle time is required, its value can be obtained from P1 as mentioned above. Since $S_i$ and $F_i$ are now decision variables, due to the non-linear constraints the formulation is no longer linear and non-convex, and thus hard to solve. Fortunately, the problem can be solved to optimal via a simple sequential procedure based on allocating shared time station by station starting at Station 1 and ending at Station $N - 1$ (no shared time is allocated to Station $N$).

Given the cycle time $Z^*$, the assembly time of station $i$ will not exceed the cycle time if $F_i + (1 - p_{i-1})S_{i-1} + p_i \cdot S_i \leq Z^*$ for $i = 1, \ldots, N$. Let $LTP_i$ be the amount of shared time shifted from the preceding upstream station, $i - 1$, namely, $(1 - p_{i-1})S_{i-1}$. Then, $LTP_i + F_i + p_i \cdot S_i \leq Z^*$. The algorithm for minimizing the shared time is described as follows.

**Algorithm T-I:**

1. **Step 0:** Start with a given balancing solution and a desired cycle time $Z^*$. Set all tasks as shared and solve P1.
   - If $Z^*$ is lower than the solution of P1, no feasible solution exists, exit the algorithm.
   - For each station—set all tasks as fixed, and set $LTP_1 = 0$.
   - If the cycle time, given that all tasks are fixed, is lower than or equal to $Z^*$, exit the algorithm.
   - Set the station counter, $i = 1$. Set $N =$ number of stations.
Step 1: If $LTP_i + TST_i > Z^*$, proceed to Step 2 (the station is over-loaded). Otherwise, Set $LTP_{i+1} = 0$ and go to Step 3 (the station is under-loaded; no shared time should be transferred to the next station).

Step 2: Share tasks of Station $i$ (last task first) until $LTP_i + F_i / C_20 \leq Z^*$. Calculate $p_i = (Z^* - (LTP_i + F_i))/S_i$. Set $LTP_{i+1} = (1 - p_i)S_i$. Proceed to Step 3.

Step 3: If the cycle time is lower than or equal to $Z^*$, exit the algorithm. Otherwise, set $i = i + 1$. If $i = N$, end the algorithm. Otherwise go to Step 1.

Theorem 1: Algorithm T-I provides an optimal solution to model formulation P2.

Proof: Since $Z^*$ is larger than or equal to the optimal solution value of P1, there is a feasible solution to P2 with a cycle time of $Z^*$. The rest of the proof shows that Algorithm T-I provides such a solution with minimum total shared time ($\sum_{i=1}^{N-1} S_i$). Since the load of every station $i$ is equal to $LTP_i + F_i + p_iS_i$ ($LTP_i = 0$) one can see that the only other station affecting the load of station $i$ is station $i - 1$. Clearly, the station load increases with $LTP_i$, and if $LTP_i$ increases the minimum required shared time $S_i$ to achieve feasibility of station $i$ cannot decrease. Hence, we show next that the minimum amount of shared time, $LTP_i$, is transferred from each station $i - 1 (i = 2, \ldots, N)$ to station $i$, followed by a minimum amount of shared time. Starting with station 1, if $TST_1 \leq Z^*$, no sharing is required ($LTP_2 = 0$). Otherwise, according to Algorithm T-I, the minimal amount of time $S_1$ is shared to satisfy $F_1 + p_1S_1 = Z^*$, where $0 \leq p_1 \leq 1$ (note that since tasks cannot be split, $p_1$ is not necessarily equal to 1). Consequently, a minimum amount of load, $LTP_2 = F_1 + S_1 - Z^* = (1 - p_1)S_1$, is transferred to Station 2. Similarly, in general, for every $i > 1$, the minimum amount of shared time (minimum $S_i$ which satisfies $LTP_i + F_i + p_iS_i = Z^*$) is assigned to station $i$ as minimum load, $LTP_{i+1} = LTP_i + F_i + S_i - Z^* = (1 - p_i)S_i$, is transferred to station $i + 1$. As a result, the minimum amount of shared time is assigned at station $i$ resulting in a minimum total shared time, ($\sum_{i=1}^{N-1} S_i$). If there is no $S_i$ that satisfies $LTP_i + F_i + p_iS_i = Z^*$, $i = 1, \ldots, N - 1$, or $TST_N > Z^*$, no feasible solution exists, contradicting the condition at the beginning of the proof.

3.2 Tradeoff analysis between the cycle time and the sharing cost

In this section, the tradeoff between the sharing cost and the throughput rate (or the cycle time) is discussed. When the sharing cost is limited or when looking for the high profit solution, associated with the throughput rate and the sharing cost, the sequential algorithm described in section 3.1 is no longer valid and a new solution approach has to be developed.

3.2.1 Preliminaries.

Definition 1: An optimal DLB-Type II solution is a solution, which minimizes the cycle time as a primary objective and the amount of shared time as a secondary objective subject to a given maximal amount of shared time.

Definition 2: Station $i^*$ is defined as a bottleneck (BN) station in a line with shared tasks if: $i^* = \arg \max_i ((1 - p_i)S_i + F_i + p_iS_i)$, using $p_i$ values from the
solution of P1. If more than one station satisfies this definition, each of these stations is a bottleneck.

**Definition 3:** Station $i$ is a single bottleneck if Stations $i-1$ and $i+1$ are not bottlenecks.

**Definition 4:** A Bottleneck Segment (BNS) is a set of $L$ consecutive bottleneck stations. A single bottleneck is a special case of a bottleneck segment with $L=1$.

Note that a DLB-Type I solution of a balanceable line yields a single BNS including the entire line. This occurs since, with balanceable lines, all stations have the same operation time, as seen in equation (1).

**Lemma 1:** In a DLB-Type II solution one can find at most one shared task on each BN station. If more than one task is shared in a station, all but one of the tasks should be fully performed either at the current station or at the downstream station (and hence should not be considered as shared).

**Proof:** Assume there exists some BNS consisting of Station $i$ and Station $i+1$. Let tasks $J_1$ and $J_2$ be the shared tasks of Station $I$ ($J_2$ is the last task in Station $i$) with time durations $t_1$ and $t_2$, respectively, so that $S_i = t_1 + t_2$. Since Stations $i$ and $i+1$ belong to the BNS: $F_i + (1 - p_{i-1})S_{i-1} + p_iS_i = F_{i+1} + (1 - p_i)S_i + p_{i+1}S_{i+1}$. The following three cases are possible:

**Case 1:** If $p_iS_i > t_1$ then $J_1$ can be considered as a fixed task in Station $i$ and $J_2$ remains the only shared task in that station. In order to maintain the balancing equation, the total station time should remain the same as $t_1 + p'_1t_2 = p_iS_i$ (the remaining load on Station $i$) or $p'_1 = (p_iS_i - t_1)/t_2$, where $p'_1$ is the new balancing proportion of the shared time in Station $i$. The new balancing equation is then:

$$F_i + t_1 + (1 - p_{i-1})S_{i-1} + p'_1t_2 = F_i + (1 - p_{i-1})S_{i-1} + p_iS_i$$

$$= F_{i+1} + (1 - p_i)S_i + p_{i+1}S_{i+1}.$$ 

Since $p_i < 1$ and $p_iS_i > t_1$, then, $t_1 < p_iS_i < t_1 + t_2$ and $0 < p'_1 < 1$; hence the balancing solution is feasible.

**Case 2:** If $p_iS_i < t_1$ then $J_2$ can be considered as a fixed task on Station $i+1$ and $J_1$ remains the only shared task in Station $i$. In order to maintain the balancing equation, the total station time should remain the same as $t_2 + (1 - p'_2)t_1 = (1 - p_i)S_i$ (the shared time transferred to Station $i+1$) or $p'_2 = (p_iS_i)/t_1$. The new balancing equation is then:

$$F_i + (1 - p_{i-1})S_{i-1} + p_iS_i = F_i + (1 - p_{i-1})S_{i-1} + p'_2t_1$$

$$= F_{i+1} + t_2 + (1 - p'_2)t_1 + p_{i+1}S_{i+1}$$

$$= F_{i+1} + t_2 + \left(1 - \frac{p_iS_i}{t_1}\right)t_1 + p_{i+1}S_{i+1}$$

$$= F_{i+1} + t_2 + t_1 - p_iS_i + p_{i+1}S_{i+1}$$

$$= F_{i+1} + (1 - p_i)S_i + p_{i+1}S_{i+1}.$$ 

Since $p_i > 0$ and $p_iS_i < t_1$, then, $0 < p_iS_i < t_1$ and $0 < p'_2 < 1$, hence the balancing solution is feasible.
Case 3: If $p_i S_i < t_1$, based on the above analysis one can see that $J_1$ can be considered as a fixed task on Station $i$, $J_2$ can be considered as a fixed task on Station $i + 1$, and no shared time remains in Station $i$.

If $S_i$ is composed of more than two tasks, we could batch them anyway into two tasks, perform the above procedure and split them again. We do this until one shared task remains on the station.

Denote the first station of the BNS as $BNSF$ and the last station, $L$, as $BNSL$. Since a single BN is a BNS of length $L = 1$, a single BN is a BNSF as well as a BNSL.

Lemma 2: There exists an optimal DLB-Type II solution, in which in all BNSs (1) the $BNSF$ station does not share tasks with its preceding station and (2) the $BNSL$ station does not contain shared tasks.

Proof: We show that every set of stations constituting a BNS which does not satisfy either of the Lemma’s conditions can be modified, such that (1) each of the above stations is no longer a bottleneck, or, (2) a subset of the above stations constitutes a new BNS which satisfies the Lemma’s conditions. Let us refer to such a BNS, say, $BNS_i$, in which its first station, $BNSFi$, shares a task with its preceding station, and/or its last station, $BNSLi$, contains a shared task. We refer separately to the $BNSF$ and the $BNSL$.

**BNSF**: Say, $BNSFi$ shares a task with its preceding upstream station. Clearly, the upstream station is a non-bottleneck station (otherwise, $BNSFi$ is not the first station in the BNS). $BNSFi$ correspond to either of the following two cases:

Case 1: Every station in $BNSi$ shares time with its preceding station. For every Station $k$ of $BNSi$, a small amount of load, $\varepsilon_k$, can be transferred to station $k - 1$, where $\varepsilon_{k1} > \varepsilon_{k2}$ for $k_1 < k_2$. As a result, the load of all stations in $BNSi$ is reduced, and each of these stations is no longer a bottleneck.

Case 2: There is at least one station in $BNSi$ that does not share time with its preceding station. Say, Station $j$ is the first such station in $BNSi$ and $j$ is not the first station in $BNSi$ (otherwise, contradicting the condition at the beginning of this part). A similar load transfer procedure as performed in Case 1 can be applied here starting from station $j - 1$ backward down to the first station in $BNSi$. As a result, station $j$ constitutes the new $BNFSi$ of $BNSi$ which does not share time with its preceding station; namely, $BNSi$ satisfies condition (1) of the Lemma.

**BNSL**: Say, $BNSLi$ contains a shared task. Clearly $BNSLi$ cannot be the last station in line, and its downstream station in not a bottleneck. $BNSi$ corresponds to either of the following two cases:

Case 1: Every station in $BNSi$ contains a shared task. For every Station $k$ of $BNSi$, a small amount of load, $\varepsilon_k$, can be transferred to station $k + 1$, where $\varepsilon_{k1} < \varepsilon_{k2}$ for $k_1 < k_2$. As a result, the load of all stations in $BNSi$ is reduced, and each of these stations is no longer a bottleneck.

Case 2: There is at least one station in $BNSi$ with no shared task. Say, Station $j$ is the last such station in $BNSi$ having no shared tasks and $j$ is not the last station in $BNSi$ (otherwise, contradicting the condition at the beginning of this part). A similar load transfer procedure as performed in Case 1 can be applied here starting
from station \(j + 1\) up to the last station in \(BNS_i\). As a result, station \(j\) constitutes the new \(BNS_{Li}\) of \(BNS_i\) which does not contain a shared task; namely, \(BNS_i\) satisfies condition (2) of the Lemma.

When both the BNSF shares time with its upstream station and the BNSL contains a shared task, the above can be applied on both the BNSF and the BNSL until the new form of \(BNS_i\) satisfies the Lemma’s condition, or it is no longer a BNS.

Let Solution \(S^*\) be an optimal DLB-Type II solution in which each BNSF does not share time with its preceding station and each BNSL has no shared tasks. This solution can be reached from any other optimal DLB-Type II solution, as was shown in the proof of Lemma 2.

**Theorem 2:** For any given optimal solution of \(S^*\) type, setting the last task of every BNSL station as shared is a necessary condition for reducing the cycle time of the line.

**Proof:** The cycle time of the line is determined by a single BNS or more. In order to decrease the cycle time of the line, one should reduce the performance time of each of the stations of each BNS, and in particular, the performance time of each BNSL. Say, we would like to reduce the time of the last station of \(BNS_i\), \(BNS_{Li}\). The time reduction can be done only by either of the following two ways: (1) Shifting load backward to the upstream station, \(L - 1\) of \(BNS_i\), or (2) Passing load forward from \(BNS_{Li}\) to the downstream station. Shifting load backward is possible only when the length of \(BNS_i\) is greater than or equal to 2 (otherwise, contradicting the conditions on \(S^*\)). However, since \(BNSF_i\) does not share time with its preceding station and since all stations of \(BNS_i\) are fully loaded, shifting load from \(BNS_{Li}\) backward to any station within \(BNS_i\) will result in increasing the cycle time of \(BNS_i\). Hence, the only way for decreasing the cycle time resulting from \(BNS_i\) is by shifting load from \(BNS_{Li}\) to the downstream station. Since \(BNS_{Li}\) has no shared tasks, setting the last task of each \(BNS_{Li}\) as shared is the only way for improving the cycle time of each \(BNS_i\), thus a necessary condition to improve the cycle time of the line.

### 3.2.2 Algorithms DLB-Type II and DLB-Type III.

Let \(C_j\) be the sharing cost of each task \(j\) and \(TAR\) be the total cost available for sharing (note that since \(C_j\) consists of design and operational components, all cost values are expressed from now on per unit of time). The following algorithm of DLB-Type II minimizes the cycle time as a primary objective and the shared time/cost as the secondary objective, given the limited sharing cost, \(TAR\).

**Algorithm T-II:**

1. **Step 0:** Start with a given balancing solution. Let \(CAR\) be the remaining available resources for sharing. Set \(CAR = TAR\). Set all tasks as shared. Find \(Z^*\) using P1. Set all tasks as fixed and \(k = 0\). Set \(Z_k = \max_i(TST_i)\), the current cycle time, and \(ST_k = \phi\), the initial empty set of shared tasks. If \(Z_k = Z^*\), exit the algorithm—the cycle time cannot be further improved. Set \(k = 1\).

2. **Step 1:** Identify every BNSL of the line. If the last station of the line is a BNSL, go to Step 3—the cycle time cannot be further improved. Otherwise, let \(B\) be the set containing every last task in any existing BNSL and compute \(C = \sum_{j \in B} C_j\). If \(C \leq CAR\), set the last task of each BNSL as shared,
update $ST_k = ST_{k-1} + B$ and $CAR = CAR - C$. Otherwise go to Step 3—the available resources are not sufficient for further improvement.

Step 2: Solve $P_1$ to find $Z_k$, the current cycle time of the line. If $Z_k = Z^*$, go to Step 3—the cycle time cannot be further improved. Otherwise, set $k = k + 1$, adjust the load to reach a type $S^*$ solution based on Lemma 2 and go to Step 1.

Step 3: Let $i^*$ be the first iteration in which $Z_{i^*} = Z_k$ (the first iteration where the best cycle time was obtained). Set $ST_{i^*}$ contains the minimal set of tasks which have to be shared to obtain the minimal possible cycle time, $Z_{i^*}$, subject to the given amount of resources for sharing.

This algorithm follows Theorem 2 and shares only necessary tasks in order to improve the cycle time. The algorithm terminates either when the desired cycle time has been achieved or when the available sharing cost has been exceeded. Note that the case where the total sharing time, rather than cost, is limited is a special case of the above, where $C_j = t_j$, $\forall j$.

Assuming that all assembled products are sold, the operational profit is a linear function of the throughput rate of the line. Taking this fact into consideration, a clear tradeoff between the operational profit and the sharing cost is identified. Accordingly, as more tasks are shared, the sharing cost increases as well as the throughput rate of the line, and consequently, the operational profit. While summing up the two, an economical solution should be found. This problem is defined as DLB-Type III.

Let $P$ denote the total profit, $R$ is the marginal profit from selling one unit of product, $CT$ is the cycle time of the line and, $SC$ is the sharing cost per unit of time. Then, the above tradeoff is expressed by the following equation. Note that the buffer size required to achieve a cycle time which is equal to the average load should be considered in the cost analysis. The effect of the buffer size on the cycle time as well as the recommended buffer size is addressed in section 4.

$$P = \frac{R}{CT} - SC. \quad (12)$$

The following algorithm for DLB-Type III is an extension of Algorithm T-II as we gradually share tasks until reaching the minimum theoretical cycle time, $Z^*$. Then, we look backward and identify the stage with the highest profit; this stage yields the global optimum of the problem.

**Algorithm T-III:**

Step 0: Start with a given balancing solution. Let $SC_k$ be the cumulative sharing cost of Stage $k$. Set $SC_0 = 0$. Set all tasks as shared. Find $Z^*$ using $P_1$. Set all tasks as fixed. Set $Z_0 = \max_i(TST_i)$—the current cycle time, and $ST_0 = \emptyset$—the initial empty set of shared tasks. Let $P_k$ be the profit of Stage $k$ calculated using equation (12). Calculate $P_0$; if $Z_0 = Z^*$, go to Step 3—the cycle time cannot be further improved. Set $k = 1$.

Step 1: Identify every BNSL of the line. If the last station of the line is a BNSL, go to Step 3—the cycle time cannot be further improved. Otherwise, let $B$ be the set containing every last task in any existing BNSL. Set the last
task of each BNSL as shared, update \( ST_k = ST_{k-1} + B \) and \( SC_k = SC_{k-1} + \sum_{j \in B} C_j \).

**Step 2:** Solve \( P_k \) to find \( Z_k \), the current cycle time of the line. Calculate \( P_k \) using equation (12). If \( Z_k = Z^* \), go to Step 3—the cycle time cannot be further improved. Otherwise, set \( k = k + 1 \), adjust the load to reach type \( S^* \) solution based on Lemma 2 and go to Step 1.

**Step 3:** Find \( k^* = \arg \max_k (P_k) \)—the stage in which the maximum profit obtained. The optimal profit solution is \( p_{k^*} \) with a cycle time of \( Z_{k^*} \) and a set of shared tasks \( ST_{k^*} \).

Note that while maximizing the profit, one should execute Algorithm T-II until reaching the minimum theoretical cycle time and then look for the highest profit. Still, since the number of stages is bounded by the number of tasks, \( K \), the complexity of the algorithm is \( O(N \cdot \log_2 K) \) times the complexity of solving LP problem. Consequently, the algorithm can solve any real-size problem in only a few seconds.

### 3.2.3 Illustrative example.

The following example demonstrates the principles of Algorithm T-II and Algorithm T-III on a six-station, 18-task assembly line. Table 1 contains the time duration and the sharing cost of each task. Consider first a situation where the profit is not taken into account, and the purpose is to reduce the cycle time as much as possible given that the hourly sharing cost is limited to $50.

In this case, Algorithm T-II runs until the sharing costs exceed the predefined limit or the minimum theoretical cycle time, \( Z^* = 9.5 \), has been reached. One can see that at the beginning, when no sharing is applied, the cycle time is 12 min and \( CAR = 50 \) (see figure 1(a)). This is the cycle time of a traditional assembly line. We see that the BNS is Station 2 and the last task in the sequence, Task 7, then has to be shared in order to reduce the cycle time. Sharing Task 7 via the solution of P1 results in a decreased cycle time value of 10.5 and \( CAR = 27 \) (see figure 1(b)), with \( p_2 = 0.625 \). Then, Stations 2 and 3 constitute a two-station BNS. The last task of the BNS’s last station, Task 10, has to be shared. We can see the result in figure 1(c), where the cycle time reduces to 10 min, with \( CAR = 11 \), \( p_2 = 0.5 \) and \( p_3 = 0.8 \) (note that the value of \( p_2 \) has changed in the new solution of P1).

Next, a modification has to be done in order to obtain an \( S^* \) type solution. One can see that shifting a small amount of load forward from Task 7 and Task 10 will result in two single BNSs, Station 1 and Station 5 (Stations 2 and 3 are no longer bottleneck). Consequently, in order to achieve further improvement, Tasks 3 and 15 have to be shared. However, their sharing costs are equal to $30 which is larger than the value of \( CAR \). Hence, the algorithm terminates with a cycle time of 10 min and hourly sharing costs of $30 resulting from sharing tasks 7 and 10.

#### Table 1. Data of the example problem.

<table>
<thead>
<tr>
<th>Task ( j )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_j ) (min)</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>( C_j ) ($/h)</td>
<td>6</td>
<td>21</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>11</td>
<td>23</td>
<td>13</td>
<td>5</td>
<td>16</td>
<td>10</td>
<td>18</td>
<td>7</td>
<td>22</td>
<td>17</td>
<td>24</td>
<td>18</td>
<td>11</td>
</tr>
</tbody>
</table>
When the total profit is concerned, Algorithm T-III is applied, while the current procedure continues until reaching the minimum theoretical cycle time, \( Z^* \).

Consequently, tasks 3 and 15 become shared resulting in the optimal cycle time of 9.5 (figure 1(d)) with 
\[
p_1 = 0.875, \quad p_2 = 0.25, \quad p_3 = 0.5, \quad p_5 = 0.875.
\]

To demonstrate the profit issue, let us assume that we gain a marginal profit of $50 per unit of product. A simple calculation shows that the economical solution is sharing task 7 alone. In this case, the hourly profit is 
\[
(50/10.5) \cdot 60 - 23 = \$262.7.
\]

### 4. Line operation

During the line operation under a DLB policy one should decide in each cycle whether the current station or the downstream station performs the shared task. To this end, several operating rules were developed and are compared below.

The operating rules are divided into two types: state-independent rules and state-dependent rules. State-independent rules rely on the balancing proportions, \( P_\iota \), as a target value and aims in keeping the sharing proportions as close as possible to the values of \( P_\iota \). The state-dependent rules, on the other hand, rely on dynamic characteristics of the line during its operation (e.g. the buffer state). Note that state-independent rules are much easier to implement since no collection of on-line data is needed. Moreover, since state-independent rules do not depend on the dynamic behaviour of the line, an exact production plan can be done in advance, when needed.
4.1 State-independent rules

Three state-independent rules were developed and are described as follows.

4.1.1 Coin-Toss. According to the Coin-Toss rule, a random number, \( x \sim \text{U}(0, 1) \), is generated each time a product arrives at station \( i \). If \( x < p_i \), then station \( i \) performs the shared task, otherwise, station \( i + 1 \) performs this task. This way, the proportion of times station \( i \) will perform the shared task will converge to the value of \( p_i \) as the number of cycles increases.

4.1.2 Goal-Chasing. A rule based on the Toyota Goal-Chasing Method (Monden 1983) is used at each station. Each time a product arrives at the station, a decision which produces a cumulative actual balancing proportion as close as possible to \( p_i \) is taken.

Let \( j_i \) denote the number of products that have entered Station \( i \) since the beginning of the line operation, and, \( s_i \), the number of products which performed their shared task in Station \( i \) since the beginning of the line operation.

The following procedure is then applied:

Step 0: Set \( j_i = 0 \), \( s_i = 0 \) for each Station \( i \).

Step 1: Each time Station \( i \) has completed performing the fixed task of a product, calculate:

\[
p'_i = s_i + 1/j_i + 1 - \text{the predicted balancing proportion if Station } i \text{ performs the shared task, and}
\]

\[
p''_i = s_i/j_i + 1 - \text{the predicted balancing proportion if Station } i \text{ does not perform the shared task.}
\]

Step 2: If \(|p'_i - p_i| < |p''_i - p_i|\), perform the shared task at Station \( i \), set \( s_i = s_i + 1 \). Otherwise, perform the shared task at Station \( i + 1 \).

Step 3: Set \( j_i = j_i + 1 \). Return to 1.

4.1.3 Over/Under. This rule is similar to Goal-Chasing, and the decision whether to perform the shared task of a product at station \( i \) or at station \( i + 1 \) depends on a simple comparison between the cumulative balancing proportion, \( \hat{p}_i = s_i/j_i \), and \( p_i \), as described in the following procedure.

Step 0: Set \( j_i = 0 \), \( s_i = 0 \) for each Station \( i \).

Step 1: Each time Station \( i \) has completed performing the fixed task of a product, calculate:

\[
\hat{p}_i = \frac{s_i}{j_i} \quad \text{(for the first product, set } \hat{p}_i = 0).\]

Step 2: If \( \hat{p}_i \leq p_i \), perform the shared task at Station \( i \), set \( s_i + s_i + 1 \). Otherwise perform the shared task at Station \( i + 1 \).

Step 3: Set \( j_i + j_i + 1 \). Return to Step 1.

4.2 State-dependent rules

Using the state-dependent rules, each time a decision regarding the station to perform the shared task has to be taken, the state of attributes of the line, such as the
cumulative station load or the buffer state are detected, and influence the decision. The main objective is to maintain the best balance of the load among the stations during the operation. However, while state-independent rules aim at keeping the cumulative balancing proportions as close as possible to the values of $p_i$ in each station $i$, this factor is not considered by the state-dependent rules.

The implementation of state-dependent rules requires a control system, which keeps on-line data of the line's attributes along the operation period. If such a system is not available, state-independent rules should be applied.

Three state-dependent rules were developed and are described as follows.

4.2.1 Equal-load. In a perfect work allocation in an assembly line, no idle time occurs and the load of all stations is expected to be the same, and equals 100%. Using the proposed rule, the station load is addressed directly instead of using the balancing proportions as in the state-independent rules.

The load of the station is equal to its cumulative operation time excluding the idle time related to blockage of the station, divided by the total operation time of the line. The decision regarding the performance location of the shared task is done based on a comparison between the load of each Station $i$ and the downstream station, $i + 1$. If the latter is larger, the shared task is performed at Station $i$, and vice versa.

4.2.2 Buffer-usage. This rule aims in balancing the load based on the buffer state. According to this rule, if the number of items in the output buffer of station $i$ exceeds a predetermined value (e.g. the half of the buffer size), the shared task is performed at Station $i$, otherwise, the shared task is performed at station $i + 1$. This rule corresponds to the RSPT rule presented in Ostolaza et al. (1990). Still, since we do not allow changing the sequence of the products, the SPT rule is not applied here. Note that this rule cannot be used in a zero sized buffer.

4.2.3 Time-per-product. The time-per-product (TPP) method is based on the average time a station invests in each product. For each product, this time is computed as the elapsed time from which the product entered into operation until it leaves the station (including blockage time). If the TPP of the current station is smaller than the TPP of the next station, the shared task is performed on the current station. Otherwise, the shared task will be performed on the following station.

4.3 Rule comparison

4.3.1 Experimental design. In order to evaluate the performance of the proposed operating rules, a wide experimentation has been conducted, and is presented below. A simulation model has been developed to compare the operating rules under a wide range of line configurations.

The factors considered in the experimentation are described as follows:

1. Number of stations. This factor expresses the length of the line. Three levels were selected: 4, 8, and 12 stations.
2. The quality of the given balancing solution. This factor was expressed by the variability of the stations' assembly times. A large variability indicates a poor balancing solution, and a small variability is associated with a good balancing solution. Two levels were defined for this factor, where a high and a low quality balancing solution was identified by a station time generated from U(90, 110) and U(75, 125), respectively. Note that a relatively large potential for improvement via using work-sharing is associated with the latter, while just a little improvement potential exists for the former.

3. The maximal allowed shared time (in percentage of the total station time). Two levels were selected for this factor: 10 and 30% of the station time.

4. The buffer size. The buffer size was set to three levels: 0 (no buffer between consecutive stations), 2 and 10 units.

Note that while the load balancing approach mentioned above assumes unlimited buffers, we test it in a realistic limited buffer environment. Later on we discuss the effect of buffer size on the performance of DLB and suggest the recommended buffer size.

For each level of the above factors a different problem was generated. The problem tested is the same for all of the operating rules. Thirty-six experiments were conducted for each of the six operating rules, resulting in 216 experiments in total.

To assess which of the rules operates better in each scenario, the following performance measures (responses) were considered:

1. Cycle Time Improvement (CTI)—is the amount of improvement in percentage of the potential improvement achieved via using work-sharing. 
   \[
   CTI = \frac{NS - S}{NS - LP},
   \]
   where NS is the cycle time value when sharing is not allowed, S is the simulated cycle time, and LP is the minimum theoretical cycle time obtained by the solution of P1.

2. Convergence Time—the rate of approaching the optimal cycle time (measured by the time elapsed until reaching a 1% distance from the minimum theoretical cycle time, LP.

3. Average WIP—the average number of units in buffer.

Each simulation run lasts 3,000,000 time units (assembling about 30,000 units).

4.3.2 Results. The main purpose of DLB using work-sharing is improving the cycle time of a traditional assembly line. Hence, we consider the CTI performance measure as the most important measure. Consequently, the performance evaluation of the operating rules with regard to the three performance measures was made as follows. First, the CTI values of the six operating rules were compared. Rules that were found comparable in the first examination (and superior to the other rules) were further compared with regard to their convergence time, resulting in the best operating rules with respect to the line cycle time performance. Another independent comparison was conducted with regard to the WIP level, and the existence of tradeoff between the throughput rate and the WIP level has been examined. Analysis of variance has been conducted to analyse the results.

Cycle time improvement. In general, results show that DLB using work-sharing provides a significant improvement in the cycle time for most rules. However, one
can suspect that the amount of improvement is dependent on the buffer size. We can see, in figure 2, that there is a significant effect of the buffer size on the CTI. In particular, DLB performs very well for most rules under the existence of buffers. However, when a zero buffer is concerned, it is recommended, in most cases, to use a no-sharing policy rather than DLB regardless of the operating rule. Nevertheless, in some cases as shown in the following example, work-sharing in a zero buffered line may yield a significant improvement over the no-sharing situation. Consider the following two-station line, where $F_1 = 30$, $S_1 = 20$, $F_2 = 35$ ($S_2 = 0$ by definition).

When no sharing is allowed, a cycle time of 50 time units is determined by Station 1, as can be seen in figure 3(a). Figure 3(b) presents the behaviour of the system using work sharing. Following the warm-up period, the time between departures is 35 and
55 alternatively. Thus the cycle time with work sharing is 45 time units—a 10% improvement. Despite the above, we do not recommend using work-sharing in zero-
buffer lines, and in the next experiments we consider the two cases of the non-zero buffers.

A Duncan multiple range test (DMRT) was conducted to compare the six rules with regard to the CTI, when buffer size of two and 10 units is considered. This test is applied to compare multiple samples while controlling the overall Type I error at a desired level. The results are shown in figure 4, which contains the average CTI values of each of the rules with a confidence interval of 95% attached to each data point. The test identified two groups of rules; the first contains Over/Under, Goal-Chasing, Buffer-Usage and TPP, which perform significantly better than the others, Coin-Toss and Equal-Load, which form the second group. The above comparison indicates that using the above four rules, DLB provides us with close to 100% of the CTI potential improvement. Another interesting observation indicates that in all cases of balanceable lines, no buffer larger than one unit was needed when Goal-Chasing and Over/Under methods were applied. This result is important when considering using zero buffered traditional assembly lines versus the currently suggested lines.

Convergence time analysis. The convergence time was measured as the time elapsed until reaching a cycle time within 1% of the minimum theoretical value. As noted above, only the four methods which performed best for the CTI measure were considered here. The DMRT was applied again and identified two groups of rules; the first includes the Goal-Chasing and Over/Under, which performed better than TPP and Buffer-Usage, included in the second group (see figure 5).
Through this analysis it appears that Over/Under and Goal-Chasing may converge in less than 100 products, while it took around twice the time for the other two rules to reach the 1% distance from optimality. Consequently, one can conclude that these rules may be used even for relatively small lots of production volumes.

Other significant effects indicate that the convergence time increases with the number of stations for all rules as well as with the allowed shared time. The latter result may be explained by the fact that the higher the allowed shared time, the lower the minimum theoretical cycle time resulted from P1.

Average WIP analysis. The average WIP is clearly bounded by the buffer size. Hence, this measure was tested separately for a buffer size of 2 and 10 units. In figure 6(a), the average WIP is presented for each rule and each buffer size. One can see that when a 2-unit buffer size is concerned, relatively small differences among the WIP levels of the different operating rules are identified. The analysis of the DMRT, which was conducted here for the 2-unit buffer size, supports this finding, as no multiple groups were identified. Nevertheless, one can see that the Goal-Chasing and Over/Under rules outperform the other rules, however, no statistical significance has been identified for this regard.

When a 10-unit buffer is concerned, a different picture arises. In this case, one can see that the average WIP of the Coin-Toss and Equal-Load is much higher than of the other rules. The DMRT, which was conducted here for the 10-unit buffer size, supports this result as it finds a significant different between the two groups mentioned above (see figure 6(b)).
The results of the above experimentation can be concluded as follows. In the first stage of the experiments, when the cycle time and the convergence time measures were considered, Goal-Chasing and Over/Under operating rules have performed significantly better than all other rules. In the second stage, when the WIP level was considered, Goal-Chasing, Over/Under, Buffer-Usage and TPP outperformed the other two rules. Hence, no tradeoff has been identified as the Goal-Chasing and Over/Under have performed significantly better than some other rules, and at least as good as all other rules for all performance measures. Due to the lack of tradeoff it is recommended using Goal-Chasing and Over/Under in all cases.

Figure 6. Average WIP level versus method. (a) Average WIP (2 and 10 unit buffer). (b) One factor plot (10 unit buffer).
5. Summary and conclusions

In this paper, the design and operation of dynamic assembly lines using work-sharing was addressed. The proposed approach suggests applying work-sharing on an already existing traditional assembly line balancing solution to reduce its cycle time. In such a line, the task assignment to stations along with the sequence of performing the tasks within each station is given. The situation, in which the identity of the shared tasks in each station is known, was addressed first. The balancing proportions (the proportions of cycles each shared task is performed in a certain station) were obtained under this assumption; in some cases, via a closed-form solution, and in others, using an LP solution. Keeping the obtained balancing proportions guarantees balancing the load among stations, and consequently, approaching the optimal cycle time in the steady state of the operation of the line. Next, some fast optimal algorithms have been suggested for minimizing shared time/cost for a given desired cycle time and maximizing the profit associated with the operation of the line. To this end, the concept of the Bottleneck Segment (BNS) has been developed.

The cycle time improvement resulting from work-sharing was examined via simulation, where various state-dependent and state-independent operating rules were compared. Three performance measures were considered: the cycle time improvement (CTI), the convergence time and the average WIP level. We have shown that given small sized buffers between consecutive stations, some operating rules yield close to 100% of the potential improvement of the cycle time. In particular, two operating rules, the Goal-Chasing and the Over/Under, have been found superior to some other rules and at least good as all other rules for all the considered performance measures.

References

