The International Journal of Flexible Manufacturing Systems, 3 (1991): 91-120

A Line-Balancing Strategy for Designing Flexible Assembly Systems

HEUNGSOON FELIX LEE
Department of Industrial Engineering, Southern Illinois University, Edwardsville, IL 62026

ROGER VIVIAN JOHNSON
School of Business Administration, The University of Michigan, Ann Arbor, MI 48109

Abstract. We present a rough-cut analysis tool that quickly determines a few potential cost-effective designs at the initial design stage of flexible assembly systems (FASs) prior to a detailed analysis such as simulation. It uses quantitative methods for selecting and configuring the components of an FAS suitable for medium to high volumes of several similar products. The system is organized as a series of assembly stations linked with an automated material-handling system moving parts in a unidirectional flow. Each station consists of a single machine or of identical parallel machines. The methods exploit the ability of flexible hardware to switch almost instantaneously from product to product. Our approach is particularly suitable where the product mix is expected to be stable, since we combine the hardware-configuration phase with the task-allocation phase.

For the required volume of products, we use integer programming to select the number of stations and the number of machines at each station and to allocate tasks to stations. We use queueing network analysis, which takes into account the mean and variance of processing times among different products to determine the necessary capacity of the material-handling system. We iterate between the two analyses to find the combined solution with the lowest costs. Work-in-process costs are also included in the analysis. Computational results are presented.

Key Words: assembly-line balancing, closed queueing networks, flexible assembly systems, integer programming, minimum cost design.

1. Introduction

In this article, we present a methodology for approximate (rough-cut) but fast analysis of cost-effective flexible assembly system (FAS) design alternatives over a broad range of parameter values. Rough-cut analysis, quickly determining a few potential FAS designs, can be very useful to the designer, especially at the preliminary design stage. After the determination of a small number of alternative designs, the designer can proceed to a more detailed analysis such as simulation (Suri and Diehl 1987). The proposed methodology consists of quantitative methods for selecting and configuring the components of an FAS suitable for medium to high volumes of several similar products.

The FAS is organized as a series of assembly stations linked with an automated material-handling system. Each station consists of a single machine or of identical parallel machines, usually assembly robots. To accommodate high volume, we avoid circuitous product routing to ensure more-or-less straight-line product flow. However, a part bypasses an assembly station if no task is performed there. The resulting system allows a clean flow of mixed items through the system with little unnecessary capacity. The methods exploit the ability

of flexible hardware to switch almost instantaneously from product to product. One example for such an FAS is an automated printed circuit board (PCB) assembly system (Gershwin et al. 1985). Our approach is particularly suitable where the product mix is expected to be stable, since we combine the hardware-configuration phase with the task-allocation phase, thereby building in buffer capacity to account for idle time caused by task indivisibility only where it is needed.

We use two analyses in an integrated manner. For the required volume of products, we use enumeration and integer programming to select the number of stations and the number of machines at each station and to allocate tasks to stations. We use queueing network analysis, in which we take into account the fact that producing multiple products simultaneously causes usage fluctuations on the various machines by imposing a suitable variance on task processing times. The objective of the queueing analysis is to determine the necessary number of automated guided vehicles (AGVs) if conveyors are not used, as well as the number of pallets and fixtures. We iterate between the two analyses to find the combined solution with the lowest costs. Work-in-process costs are included in the analysis if they are significant.

This is an opportune time to study FASs. Manufacturers of automobiles, electronic components, computers, and electric consumer products are in the process of designing or implementing FASs because an increasing number of product variations, shorter product life cycles, the need to react flexibly to short-term variations of demand, and a highly competitive market necessitate a more flexible means of production (Owen 1984; Spur et al. 1987). More FASs will be installed because they are becoming technically more feasible, they are much cheaper than flexible machining systems, and the cost of manual assembly is high (Boothroyd 1982; Riley and Yarrow 1983; Owen 1985; Hitz 1987).

Flexible and powerful design methods using analytical models are therefore needed to retain an overview of the complex interdependencies between the various elements of FASs and to provide a small number of good design alternatives quickly, to which more detailed models such as simulation can be applied. This is because the variety of possible solutions increases with the number of functions integrated in FASs, which leads to a high work load for the designers (Spur et al. 1985; Suri and Diehl 1985, 1987). Yet there are few procedures in the literature that can be used for FAS design. These are reviewed in section 2. Section 3 describes FASs as considered in this article. Section 4 states the FAS design problems under investigation. Section 5 incorporates the two analyses into a systematic solution framework. In section 6, a deterministic mathematical formulation and an optimal algorithm are presented, followed by computational results. Section 7 describes the stochastic analysis (SA) that employs a queueing network method. Section 8 describes a termination rule that exploits a monotonic property of the deterministic analysis (DA) and a bounding method for the SA. Section 9 reports our computational experience, while our conclusions are in section 10.

2. Literature review

Browne et al. (1985) present a three-stage design procedure: design planning, detailed design, and implementation. Our methodology contributes to their detailed design phase, which specifies the type of flexible manufacturing system (FMS) (flexibility, amount of automation,

type and capability of material-handling systems), the number of machines, the number of pallets, location of machines, type and size of buffers, tooling strategy, control hierarchy, and maintenance strategy. Spur et al. (1985, 1987) give a more complete design procedure for an FMS using robots. Their procedure is divided into 1) manufacturing system analysis, 2) determination of basic data, 3) documentation of information, 4) layout planning and assessment, 5) decision of economic feasibility, 6) final detailing of the planned system, and 7) planning of the system installation. Our methodology contributes to the fourth and fifth phases of Spur's scheme.

The FAS design literature can be divided into three categories, based upon the modeling techniques employed. These are simulation, queueing networks, and integer programming.

Simulation can represent an FAS at any level of detail. However, in the early stages of design of complex systems such as FASs, we may understand very little; hence, we may not know which aspects of the system to represent in the model and at what level of detail, or which to ignore (Graham 1978). It is also very costly and time-consuming to develop, validate, and run simulation software for many design alternatives before one good alternative is chosen. Further, we cannot tell how good the chosen alternative is because simulation does not usually provide an optimal solution or benchmark with which the chosen alternative can be compared. Several researchers have used simulation to address FMS design-related problems. Liu and Sanders (1986) attempt to find optimal buffer sizes and the optimal number of pallets for an FAS, maximizing the throughput rate. They use a discrete simulation and a gradient-based search technique. Yano et al. (1988) perform a simulation study to evaluate design and operating policies for an FAS producing a large product with many options, such as automobiles. Given the assignment of assembly tasks to stations, their decisions include the number of parallel machines, buffer sizes for base parts and subparts at each station, types of buffers (random or sequential, depending on whether resequencing of parts is allowed), and policies for dispatching mating subparts to the stations.

Many researchers have used queueing network models to solve FMS design-related problems. A Markovian single-class closed queueing network model is commonly used. Vinod and Solberg (1985) and Dallery and Frein (1986) study the optimal configuration problem in FMSs, in which the decisions are the number of pallets, AGVs, and flexible machines at each station, assuming the number of stations and their workloads are known. The objective is to minimize the capital and operating costs while meeting product requirements. A more general problem is studied by Lee et al. (1989) and Dallery and Stecke (1990), where allocation of a total workload among stations is treated as a decision variable as well. Shanthikumar and Yao (1987, 1988) study the optimal server allocation problem in FMSs, which involves allocation of a given number of flexible machines to stations in order to maximize the throughput rate. They assume that the number of stations and pallets as well as the workloads of the stations are known. Shanthikumar and Yao (1989) also study the allocation of buffer space maximizing the net profit function (total production profits minus total buffer allocation costs). On the other hand, Pourbabai (1987) models an FAS as a Markovian open queueing network instead of a closed one. His objective is to maximize the throughput rate subject to an upper bound on the probability that the sojourn time for a part at a station exceeds a given value. Non-Markovian queueing models also appear in the literature. Yao and Buzacott (1985), Whitt (1985), and Kamath et al. (1988)

develop approximations to queueing networks with general arrival and service time distributions, especially to the GI/G/1 queue, and apply them in the analysis of FMSs. All the researchers in this category deal with specific decisions, assuming that many other design decisions are already known. Also, they neither consider task assignment nor relate machine flexibility to their decisions explicitly.

Researchers have also used integer programming. Whitney and Suri (1985) provide computer-aided decision tools that select machine types and parttypes from a large number of candidates after a decision is made that FMS technology is viable for a given application. Their objective is to maximize cost savings, relative to a conventional system, subject to the following constraints: limit on the total number of machines to be purchased, available machine times, and available tool slots. Because of the large size of the problem and nonlinear interaction among parts such as tool sharing, they develop a solution procedure that uses two heuristic algorithms sequentially. Graves and Redfield (1988) present and illustrate an optimization procedure that assigns tasks to workstations and selects assembly equipment for each workstation in a multiple-product assembly system. Their objective is to find a system that is capable of producing all the products in the desired volumes at minimum cost. The system cost includes the fixed capital costs for the assembly equipment and tools as well as the variable operating costs for the various workstations. These integer programming models do not take into account the aspects of material-handling systems and product flow, of resource contention and machine idle time, and of random events occurring on the assembly floor.

In addition to these studies, several researchers used hybrid methodologies involving the iterative or complementary use of at least two of these modeling techniques. Seliger et al. (1984, 1987a, b) developed the MOSYS interactive support system for planning FASs, which incorporates queueing networks and simulation. FASs with a wide range of parameters can be evaluated by either of these two techniques at different stages of design. Therefore, the approach involves trial and error by the designer who interactively works with the support system. Liu and Sanders (1988) present a hybrid algorithm that uses a queueing network model to set the number of pallets in the system; then a discrete simulation and a gradient-based search technique is used to set the buffer spacings to obtain the optimal system throughput. Bulgak and Sanders (1989) extend their work to more complex FASs where the flow of assemblies splits and merges due to repair loops. The methodology proposed in this article also falls in this category in that it combines queueing networks and integer programming.

3. Flexible assembly systems under study

An FAS consists of a set of assembly stations and a loading/unloading (L/UL) station connected by conveyors or AGV paths. A base part of an assembly—for example, a PCB—is loaded on a pallet and enters the FAS at the L/UL station. As the pallet is carried by conveyors or AGVs through assembly stations, components are assembled with the base part. When all the required components are assembled with the base part, it is carried back to the L/UL station and leaves the FAS.

The FAS under study has three characteristics. The first characteristic is that the FAS is a flow system where a base part enters the system and is processed by a series of stations containing flexible machines of type 1, followed by a series of stations containing flexible machines of type 2, continuing in this manner to completion. A part may bypass one or more stations but does not revisit any station. A flow system is common for FASs, since a large volume and short task times necessitate the efficiency of a flow system. The flexible assembly machines (FAMs) of each type are typically of a particular technology with different capabilities from the other machine types in the system. For example, in the assembly of PCBs, the common ones are single in-line package inserters (SIPs), dual in-line package inserters (DIPs), multiform modular inserters (MODIs), and variable center distance inserters (VCDs), and each type inserts one mechanically distinct type of component (Gershwin et al. 1985). Often, there is just one type of FAM.

The second characteristic is that FAMs, usually robots, have finite work space due to their physical configurations. Because a component-feeding mechanism associated with each assembly task uses some of the finite work space, we can assign only a finite number of tasks to a robot (see figure 1, similar to Groover et al. 1986). We assume that each task uses the same amount of the work space. This assumption is realistic when components are all of relatively similar sizes (Ammons et al. 1985) or when components are carried on a part magazine by an AGV to standardized docking stations installed around robots (Ranky 1986). Under this assumption, the finite work space of a robot can be redefined as its staging capacity, R, where R specifies the maximum number of tasks that can be assigned to the robot. The staging capacity can be used as a measure for machine flexibility, since among tasks assigned to a robot, there are negligible setup times between task changes. We also assume that components and assembly tools are always available when a base part is ready to be assembled at each station. This assumption is realistic, since assembly tools

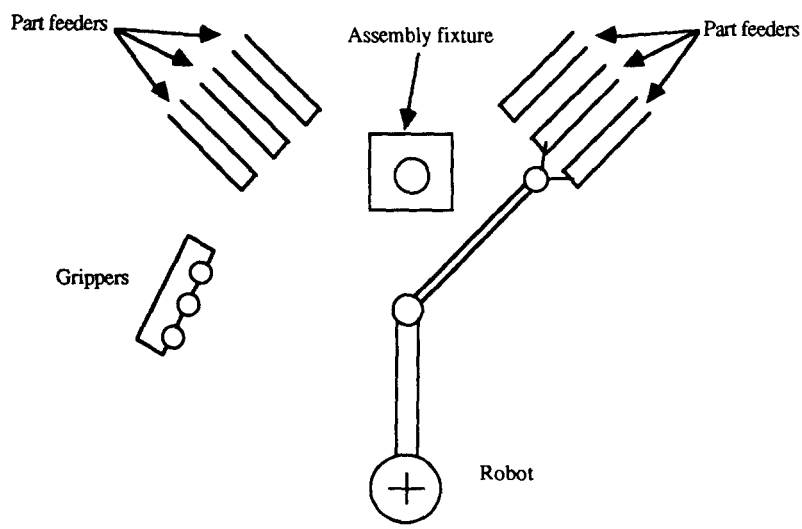

Figure 1. A robot assembly cell.

are less perishable than the cutting tools of FMSs (Hall and Stecke 1986) and computer controllers at each station keep track of the inventory levels of components.

The third characteristic is that the FAS operates in mixed-model lines on which different product types are assembled simultaneously with a known mix ratio. Processing a mix of parts makes it possible to utilize the machines more fully than otherwise. This is because different parts spend different amounts of time at the machines (Akella et al. 1988). These production lines can also achieve lower inventory of final products than multiproduct model lines, where one product type is produced at a time in a cyclic fashion. In the latter, while one product type is being produced, demand for other product types is satisfied from their inventories. These advantages are possible because of the flexibility of FAMs: negligible setup times between task or product changes.

We use one aggregate product type to collectively represent the individual product types. Precedence diagrams of all the product types are merged and represented as one superprecedence diagram for the aggregate product type. For example, in automobile assembly lines, which assemble products with a wide range of customer options, each workstation is responsible for a subset of the entire population of assembly tasks, and each vehicle requires only a subset of the available tasks at each workstation. We further assume that this superprecedence diagram is *acyclic*. For example, we do not allow task 1 to precede task 2 in one product type and task 2 to precede task 1 in another product type. This assumption makes sense for assemblies such as automobiles or PCBs, where the product types have similar assembly patterns. This precedence representation is used by Thomopolous (1970), Macaskill (1972), Graves and Redfield (1988), and Liu and Sanders (1988) for mixed-model production. Task times may differ among the product types. Thus, demand and task times for the aggregate product type are specified as the sum of demands and the weighted average task times among the product types, respectively. Since different product types may require different sets of tasks to be performed at each station, the total time required to process a part at the station may differ among product types. The variance of processing times between product types is an important input to the design of the material handling and buffer capacities, and is considered in stochastic analysis (SA), described in section 6. We give the following example for clarification.

Example. Suppose an FAS produces two product types simultaneously, each of which has the same demand per period, say, 100. Product type 1 requires five tasks (1, 2, 3, 4, 6) to be assembled in the listed order, and product type 2 requires tasks 1, 2, 5, 6 in the listed order. Let the task time for task j be j time units for $j = 1$ to 6. Then, demand of the aggregate product type is 200 per period and the superprecedence diagram and weighted-average task time t_j are shown in figure 2. Suppose the FAS consists of one type of assembly machine, which is flexible enough to be capable of performing all six tasks. Let its staging capacity $R = 4$. Suppose we have the following task assignments: tasks 1, 2, 3 and 4 are assigned to station 1 and tasks 5 and 6 to station 2. Then, average processing time (i.e., total time required to process one unit of the aggregate product) at station 1 is 6.5 time units by summing t_1 to t_4. In reality, however, it is either 10 or 3 time units, depending on which product type is processed at station 1. Variance of processing times at

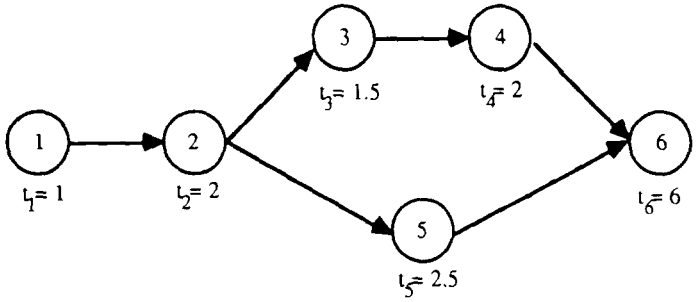

Figure 2. An aggregate product type.

station 1 while producing 100 units of each product type is computed as $\{(10 - 6.5)^2 + (3 - 6.5)^2\} \times 100/200 = 12.25$. Note that the average processing time and variance depend on product types to be produced simultaneously, their demand levels, and task assignment.

Notion of the aggregate product is introduced in order to make the proposed methodology more tractable. The methodology will provide design alternatives (capacities of machine and material-handling resources) that meet the demand of the aggregate product at minimal cost with other constraints. Thus, it does not guarantee whether the design alternatives can meet demands of individual product types. However, the above aggregation is achieved such that when demand of the aggregate product is met, the FAS will have sufficient resource capacities required to produce individual product types by their demands. Such aggregation can be justified in a rough-cut analysis at the initial design stage, where so many decisions and parameters are unknown and there is a need to determine a few potential FAS designs quickly.

An FAS having the above three characteristics is depicted in figure 3.

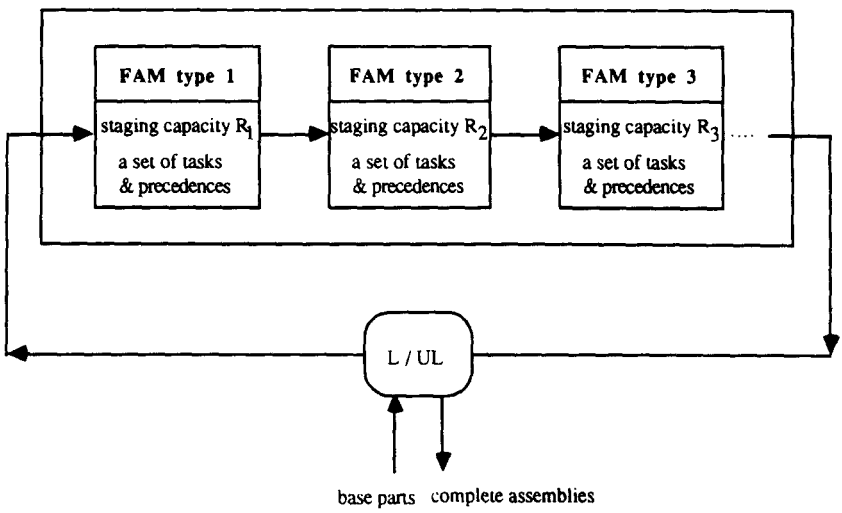

Figure 3. Flexible assembly system under study.

4. Problem statement

Given the description of the FAS in the previous section, we now state the problem that will be investigated in this article in the following order: basic data available, decisions of interest, objective, and constraints.

Basic Data. We assume that the products to be produced and the types of assembly machines have already been selected. The basic data are classified into four groups.

1. Aggregate Product
    - A (super) precedence diagram
    - Weighted-average task times
    - Demand per period
2. Assembly Machines
    - A sequence of machine types that a part visits
    - For each machine type:
        - Staging capacity
        - Total available processing time (processing capacity) per machine per period
        - A set of assembly tasks that it can perform
3. Material-Handling Systems (MHSs)
    - AGVs or conveyors
    - Types of pallets and fixtures
    - Average transfer time between each pair of stations
4. Cost Data
    - Work-in-process (WIP) inventory cost
    - Purchase and maintenance cost for each resource (machines and MHS equipment)

Design Decisions. Taking the above basic data as input, the problem is to determine the following five critical design decisions for the FAS:

1. The number of stations
2. Assignment of tasks to stations
3. The number of parallel assembly machines per station
4. The number of pallets and fixtures
5. The number of AGVs (not necessary if conveyors are used instead)

Objective. The objective of the problem is to minimize the total cost. The total cost is the sum of WIP inventory cost, and maintenance and amortized purchased costs for assembly machines and material-handling equipment.

Constraints. The solution must satisfy the following three types of constraints.

1. The demand for the aggregate product type must be satisfied
2. The number of tasks assigned to each station must not exceed the staging capacity
3. Tasks must be assigned to stations such that precedence relations among the tasks ensure that a part does not revisit any station in a flow system

5. Line-balancing approach to FAS design

The solution approach we propose to solve the problem is based upon a strategy of balancing the average workload per machine. This strategy has been commonly used when designing traditional manufacturing systems as well as FMSs (Berrada and Stecke 1986). The solution approach decomposes the entire decision process into two analyses: deterministic analysis (DA) and stochastic analysis (SA).

DA treats the weighted-average task times as constant to determine the processing capacities. It assigns tasks to machines such that the number of assembly machines is minimized, the aggregate demand is met, and the average workload per machine is balanced. Given decisions from DA, SA takes into account the variance of task times over different products in order to determine the material-handling capacities. The objective of SA is to minimize the associated resource costs plus the WIP inventory cost, subject to a constraint of meeting the demand.

The solution approach iteratively performs DA and SA as cycle time decreases. Cycle time is the average interdeparture time by a completed part from the system. Thus, in order to meet the demand d, cycle time must be less than or equal to $1/d$. The reason that cycle time below $1/d$ may be needed is as follows. Variable processing times at stations, a situation that results from simultaneous production of different products, can force robots to be idle and causes loss of some processing capacities. As the variance increases, this effect becomes more significant. Consequently, the FAS may not satisfy demand if it sticks with the decisions obtained from DA with cycle time $1/d$. To resolve this problem, we need to add some slack processing capacity to the FAS. One way to do this in the solution approach we propose is to repeat all the analyses with a smaller cycle time. These concepts lead to the following methodology (see figure 4):

A line-balancing methodology for the FAS design

Step 1. Apply DA with an initial cycle time $1/d$.
Step 2. Apply SA, given the decisions from DA.
Step 3. One design alternative is obtained at the current cycle time. Check a termination condition. If it is not met, reapply DA with a smaller cycle time and go to step 2.

This methodology generates a set of FAS design alternatives as the cycle time reduces from $1/d$ by trading off machine cost versus cost for other resources and WIP inventories.

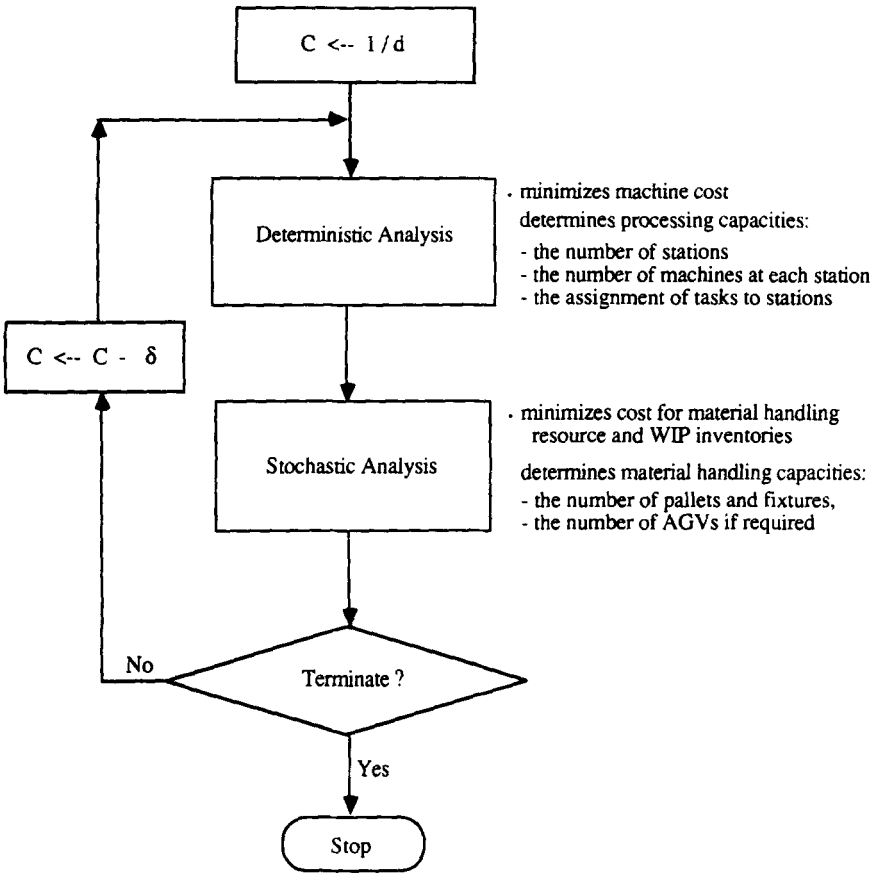

Figure 4. A line-balancing methodology for the FAS design.

One termination condition is introduced in section 8, where experimental results for the methodology are presented. The following two sections elaborate DA and SA in order.

6. Deterministic analysis (DA)

6.1. Mathematical formulation

We first focus on a special case of DA by making two assumptions. The first assumption is that all machines in the FAS are of the same type, which is flexible enough to perform all tasks. The second assumption is that an FAS consists of a single line (stations having a single machine), but not of parallel-machine stations. Later, in section 6.3, we will relax

Table 1. Notation.

C	Cycle time of the FAS
R	Staging capacity of a flexible assembly machine
d	Demand of the aggregate product
i	Index for stations
j	Index for tasks
n	The number of tasks in the aggregate product type
a_j	Staging space of task j, set to 1 for $j = 1$ to n from the assumption of equal staging space
t_j	Weighted average task time of task j for $j = 1$ to n
M	The number of stations in the FAS
N	The number of pallets circulating in the FAS
S_o	The number of AGVs if required
S_i	The number of parallel machines at station i for $i = 1$ to M
W_i	The sum of average task times assigned to station i, i.e., average service time at station i
C_N	Amortized cost for a pallet and fixture, and WIP
C_S	Amortized cost per machine for purchase and maintenance
TC	Total system cost
TH	Throughput of the FAS that is estimated from the closed queueing network model

the assumptions to study more general cases. In that section, we will also address the issue of balancing the average workload per machine. First, in table 1, we summarize notation that is frequently used.

DA for FASs assigns tasks to stations, subject to the constraints of cycle time and staging capacity. The cycle-time constraint recognizes that the average processing time (the sum of average task times) at each station should not exceed a given cycle time. This assures that the average interdeparture time by a completed part from the system does not exceed the cycle time. The staging-capacity constraint limits the number of tasks that may be assigned to a particular station to that of its staging capacity. Task assignments are also constrained by precedent requirements among tasks, since a part does not revisit any station in a flow system. The objective is to minimize the number of single-machine stations, i.e., the number of the flexible machines.

Thus, the special case of DA can be mathematically stated as

Problem 1.

Min M

subject to

$$\sum_{j=1}^{n} t_j X_{ij} \leq C, \qquad i = 1, \ldots, M, \tag{1}$$

$$\sum_{j=1}^{n} a_j X_{ij} \leq R, \qquad i = 1, \ldots, M, \tag{2}$$

$$\sum_{i=1}^{M} X_{ij} = 1, \qquad j = 1, \ldots, n, \tag{3}$$

$$X_{ij} = 0 \text{ or } 1, \qquad \text{for all } i \text{ and } j, \tag{4}$$

$$\sum_{i=1}^{M} i\, X_{ij} \leq \sum_{i=1}^{M} i\, X_{ik} \qquad \text{when task } j \text{ must precede task } k, \tag{5}$$

where X_{ij} is an assignment decision variable that is set to 1 when task j is assigned to the ith station; otherwise, it is set to 0. Constraints (1) and (2) are the cycle-time and staging-capacity constraints, respectively. Constraints (3) and (4) are assignment constraints that force each task to be assigned to only one station. Constraint (5) models the precedence relations among tasks and ensures that a part does not revisit any station in a flow system.

6.2. *Solution procedure for problem 1*

We develop an optimal algorithm for problem 1 by generalizing an optimal algorithm for the traditional assembly-line balancing problem (ALBP) for two reasons. Problem 1 without constraint (2), that is, problem 1 with $R = \infty$, becomes exactly a traditional ALBP. Constraint (2) itself is well structured due to $a_j = 1$, which makes the generalization easy. We will name the optimal algorithm for problem 1 as the ALBP for FASs, to contrast with the traditional ALBP throughout this article.

We generalize Johnson's (1988) algorithm for the traditional ALBP for the following three reasons. First, experiments with 64 ALBPs used in the literature (Talbot et al. 1986) showed that it is the fastest optimal algorithm among those known. Second, it is designed to find a good feasible solution very quickly; hence, it can be easily modified to form a good heuristic algorithm. Third, it has low memory requirements.

Johnson's algorithm proceeds by assigning tasks to stations, such that no task is assigned to station i before station $i - 1$ is completely assigned. His algorithm enumerates, either explicitly or implicitly, a tree of feasible solutions using a depth-first branch-and-bound algorithm. In the tree, every node at level j corresponds to an assignment of j tasks among the first i stations for some i. Station i may still have capacity for further task assignments, in which case we say that there is a fractional number of stations corresponding to this node. More precisely, if a fraction f_C of the capacity of station i is used, then the fractional number of stations, M_C, for this node is $i - 1 + f_C$. His algorithm forms the tree one branch at a time and reaches feasibility quickly in a single path. In order to speed up the enumeration process, his algorithm exploits the precedence structure and uses eight fathoming methods, each of which identifies situations where a node cannot contain a solution that is superior to a solution already found. Thus, in executing the tree-generation procedure of Johnson's algorithm, a node is fathomed either by explicitly constructing the subtree that emanates from the node or by successful application of one of the eight fathoming methods. When the latter case occurs, the tree-generation procedure stops constructing the subtree

of the node and backtracks to reach another node. This process continues until an entire tree of feasible solutions are enumerated. We document below the necessary modifications to the algorithm, starting with the tree-generation procedure.

6.2.1. Modified tree-generating procedure

Step 1. Obtain problem data: number of tasks, n; task times, t_j; cycle time, C; staging space, $a_j = 1$; staging capacity, R; and precedence relations among tasks

Set the current station number, $k = 1$.
Set unused_Station_k_time to C.
Set unused_Station_k_space to R.
Set the newly_selected_task = 1.
Set the number of tasks currently assigned, $p = 0$.

Step 2. Add a task to the current station.

Assign the newly_selected_task to station k.
Subtract $t_{(\text{newly\_selected\_task})}$ from unused_Station_k_time.
Subtract $a_{(\text{newly\_selected\_task})}$ from unused_Station_k_space.
Add 1 to p.
Record newly_selected_task as Task_assignment_p.
Set backtrack_task = 0.

If p equals n, go to step 6.

Step 3. Add another task at station k.

Task j is selected as the newly_selected_task, where

1. Task j is not already assigned;
2. $j >$ previous newly_selected_task;
3. $j >$ backtrack_task;
4. $t_j \leq$ unused_Station_k_time;
5. $a_j \leq$ unused_Station_k_space; and
6. All required preceding tasks of task j are already assigned to a station.

If no such task exists and newly_selected_task > 0, then apply all eight fathoming methods; if the current node is fathomed, go to step 7, otherwise go to step 4.
If no such task exists and backtrack_task > 0, then go to step 7.
Apply fathoming methods 4 to 8; if fathomed, repeat step 3 to look for another task.
Otherwise, task j becomes the newly_selected_task. Go to step 2.

Step 4. Start a new station.

Add 1 to k.
Set unused_Station_k_time to C.
Set unused_Station_k_space to R.

Step 5. Task j is the newly_selected_task, where task j is the lowest-numbered task j that satisfies

1. Task j is not already assigned;
2. All required preceding tasks of task j are already assigned to a station; and
3. $j >$ backtrack_task.

If no task exists and $p = 0$, enumeration is complete. Stop.
If no task exists and $p > 0$, go to step 7.
Apply fathoming methods 5 to 8; if fathomed, repeat step 5 to look for another task.
Otherwise, go to step 2.

Step 6. A complete solution has been found.

Save it as the Incumbent_Solution if it is the first solution, or if it is better than the previous Incumbent_Solution.
If the number of stations of this Incumbent_Solution equals the lower bound of the number of stations, stop with the optimum solution.

Step 7. Backtrack.

If unused_Station_k_time $= C$, reduce k by 1.
Set backtrack_task = Task_assignment_p.
Remove backtrack_task from station k.
Increase unused_Station_k_time by $t_{(\text{backtrack\_task})}$.
Increase unused_Station_k_space by $a_{(\text{backtrack\_task})}$.
Decrease p by 1.
Set newly_selected_task = 0.
If unused_Station_k_time $< C$ and unused_Station_k_space $< R$, then go to step 3.
Otherwise, go to step 5.

The task renumbering procedure and task-duration incrementing rule of Johnson's algorithm are not affected by the addition of the staging-capacity constraint (2). This is because they exploit only precedence relations and the differences of average task times.

6.2.2. Modified node-fathoming methods. The eight fathoming methods are grouped into two parts. Four of the methods employ dominance rules and the other four employ bound arguments. We present the modifications of the eight methods that are necessary due to the additional constraint (2).

No modifications are necessary for the first three dominance rules. They are Jackson Rule-1 dominance, Jackson Rule-2 dominance, and the first-station dominance. This is true since swapping two tasks that are assigned to different stations does not violate the staging-capacity constraint due to $a_j = 1$ for all j.

The fourth dominance rule, which is the labeling-dominance rule, is affected by the generalization. This rule employs Schrage and Baker's (1978) labeling scheme, which assigns

a numerical label to each task in such a way that the sum of labels of any feasible set of tasks is unique. In Johnson's algorithm, a one-dimensional array with an array size of 32,600, addressed by the sum of labels, is maintained to store the fractional number of stations for each node. For our algorithm, we modify the concept of the fractional number of stations. Consider a node in the tree with i stations and let f_C and M_C have the same meaning as before. Let f_R denote the fractional (staging) capacity of station i that is occupied, and let $M_R = i - 1 + f_R$ denote the fractional number of stations occupied in terms of staging space requirements for tasks in the node. Our algorithm uses a two-dimensional array with an array size of 2 by 32,600. In this array, which is addressed by the sum of labels of tasks in the node, two values (M_C, M_R) are stored instead of one value M_C only. If each of the two numbers, M_C and M_R, associated with a newly formed set is greater than or equal to the corresponding number obtained from a previous grouping of the same set of tasks, then the node is fathomed.

In order to make the modified labeling dominance rule more effective, we form pairs of elements from this two-dimensional array. Let P_1, P_2, P_3, and P_4 be the different nodes (i.e., the different groupings) for the same set of tasks such that P_k is generated by the tree-generation procedure before P_i, if $k < i$. Clearly, the nodes have the same label. Denote as $M_C(P_k)$ and $M_R(P_k)$ the fractional number of stations occupied by node P_k for $k = 1$ to 4 in terms of the task times and staging space, respectively. Suppose that P_1 does not dominate P_2, nor vice versa by this dominance rule; hence, either $M_C(P_1) < M_C(P_2)$ and $M_R(P_1) > M_R(P_2)$ or $M_C(P_1) > M_C(P_2)$ and $M_R(P_1) < M_R(P_2)$. Then, these values are stored in one pair of the two-dimensional array addressed by this label. When the tree-generation procedure generates node P_3, the modified labeling-dominance rule executes the following logic. If $M_C(P_k) \leq M_C(P_3)$ and $M_R(P_k) \leq M_R(P_3)$ for $k = 1$ or 2, then node P_3 is fathomed and the tree-generation procedure backtracks. If $M_C(P_k) \geq M_C(P_3)$ and $M_R(P_k) \geq M_R(P_3)$ for *either* $k = 1$ or 2, then $M_C(P_3)$ and $M_R(P_3)$ replace $M_C(P_k)$ and $M_R(P_k)$, respectively. If $M_C(P_k) \geq M_C(P_3)$ and $M_R(P_k) \geq M_R(P_3)$ for *both* $k = 1$ and 2, then $M_C(P_3)$ and $M_R(P_3)$ replace $M_C(P_1)$ and $M_R(P_1)$, respectively, and memory spaces for $M_C(P_2)$ and $M_R(P_2)$ are left available for P_4 if P_4 is not fathomed. In any other case, no change takes place in the array. When node P_3 is not fathomed, the tree-generation procedure constructs a subtree of this node by adding the next task to the node.

A limitation of the labeling dominance rule is that the sum of labels of feasible sets of tasks frequently exceeds the available array space. Therefore, testing is performed only for sets of tasks that have a sum of labels less than the selected array size. Throughout the experimentation, an array size of 32,600 is used as in Johnson's algorithm. Limiting the array size in this way is not perceived to be a major drawback, as Johnson (1988) points out, since the greatest power of dynamic programming probably is elimination of nodes at the early portion of the tree, which is exactly where the sum of labels is sufficiently small.

All four bound arguments are modified slightly. Given node P with partial assignments, let $B_C(j, P)$ and $B_R(j, P)$ for $j = 1$ to 4 be the jth bound argument of Johnson's algorithm, specifying lower bounds on the number of stations needed to accommodate the task times and staging space, respectively, of the remaining tasks. Max $j\{\max\{M_C(P) + B_C(j, P), M_R(P) + B_R(j, P)\}\}$ is computed at each node and, if it is greater than or equal to the number of stations of the incumbent solution, the node is fathomed.

6.3. *Extensions*

We studied in the previous section ALBPs for FASs with a single line and one machine type. In this section, we consider more general ALBPs for FASs.

6.3.1. Parallel lines. We allow stations to have more than one machine. However, search in DA will be confined to parallel lines where the number of parallel machines is identical for all stations using the same machine type; search for general configurations is prohibited due to its search size unless an FAS uses a very small number of machines. Search for parallel lines may lead to a smaller number of flexible machines by finding a better workload balance. This may happen for an FAS with a single line in which staging capacity R is sometimes large compared to cycle time C, so that a large portion of R is not used up before a new station starts to be filled. Suppose an FAS has n_p parallel lines. Since each of the parallel lines is identical, each line is supposed to produce d/n_p in order to meet demand. This means that the cycle time for each line becomes at most n_p/d, which is n_p times that of a single line that produces d. Thus, when we let M_p^* be the minimum number of the machines obtained by solving the ALBP with a single line and cycle time n_p/d, the minimum number of the machines for FASs with n_p parallel lines is M_p^* times n_p.

Algorithm 1 below finds the minimum number of the machines when parallel lines are permitted for an FAS. It searches for the optimum number of parallel lines by sequentially increasing the number of parallel lines and applying each time the optimal ALBP algorithm for FASs with a single line. The algorithm terminates when increasing the number of parallel lines cannot lead to a better solution, that is, when the minimum number of machines found up to n_p parallel lines $\leq (n_p + 1) \times T_M$ where T_M is the theoretical minimum number of machines for each line by staging space, which is the rounded-up integer of the sum of staging space of tasks divided by staging capacity R.

Algorithm 1.

```
begin
minimum_no ← a large positive number;
n_p ← 1;
terminate ← false;
while not (terminate) do
  begin
  solve problem 1 with C specified as n_p/d
  if M_p × n_p < minimum_no then
    begin
    minimum_no ← M_p × n_p;
    save task assignments;
    end
  if minimum_no ≤ (n_p + 1) × T_M then
    terminate ← true
  else
    n_p ← n_p + 1;
  end {while}
end.
```

Since there are usually multiple optimal task assignments for problem 1, we introduce a balancing workload procedure that finds among them one that balances average workload per machine. (This is akin to the type II assembly line balancing problem as defined by Baybars (1986).) The procedure finds the smallest cycle time for which the optimal solution for problem 1 gives the same minimum number of machines. Given the optimal number of parallel lines, n_p^*, from algorithm 1, this procedure sequentially decreases the cycle time from $C = n_p^*/d$. The procedure checks if there exists a feasible solution for problem 1 for a given cycle time and M_p^*. This can be easily done by a variant of the optimal algorithm for problem 1. The procedure finds the smallest cycle time when it finds no feasible solution for the first time.

6.3.2. Multiple machine types. Since basic data available (see section 3) include a sequence of machine types visited by a part and a set of tasks performed by each machine type, we simply apply algorithm 1 followed by the balancing workload procedure to each machine type separately.

6.4. Experimental results

Experiments were performed on 64 ALBPs assembled by Talbot et al. (1986), which have been used in the literature of ALB as a benchmark of comparing different algorithms for ALBP. The number of tasks of these problems ranges from 7 to 111. The algorithm was coded in FORTRAN and run on an IBM 3090-600, using the VS-opt3 compiler. Each problem was run with a three-second time trap. When the run time exceeded three seconds, the program stopped executing and printed out the incumbent solution. Twelve different staging capacities were used for each of the 64 problems. They were $R = 2$ to 10, 15, 20, and 30. For each R, we collected the following statistics: total CPU time taken to solve the 64 problems, the number of the verified optimal solutions found among the 64, and the number of problems among the 64 for which the last new incumbent solution was found after .1 and .5 seconds of CPU time, respectively. The last two statistics were collected to see how quickly the algorithm found a (near) optimal solution. These statistics are summarized in table 2. Table 2 also includes the statistics for Johnson's algorithm, which was shown to be successful for the traditional ALBP (Johnson 1988).

The total CPU time monotonically increased and then decreased with respect to staging capacity, R. The number of the verified optimal solutions monotonically decreased and then increased wth respect to R. The interpretation of this monotonic behavior is as follows. For small R such as $R = 2$ and 3, the staging-capacity constraint was more constrained than the cycle-time constraint, which made task assignments easy. For $4 \leq R \leq 10$, it appeared that one constraint did not dominate the other and both constraints played an active role in assigning tasks to stations, whch resulted in more computation time. For large R such as $R > 10$, the cycle-time constraint was more constrained than the staging-capacity constraint. At $R = 7$, the algorithm had the longest CPU time (32.607 seconds) and the smallest number of the verified optimal solutions (55 out of 64). This means that for nine problems, the tree-generation procedure did not enumerate the entire tree of feasible task sequences with the three-second time trap.

Table 2. Experiment with the optimal ALB algorithm for FASs.

Problem Type	R	Total CPU Time[a]	No. of V. Opt.[b]	LNS > .1[c]	LNS > .5[d]
Traditional ALB	∞	6.309	64	3	1
ALB for FASs	2	1.811	64	2	0
	3	5.342	63	1	0
	4	11.931	61	0	0
	5	20.212	59	2	1
	6	20.625	58	3	0
	7	32.607	55	4	2
	8	28.694	56	5	2
	9	25.537	57	5	2
	10	15.820	60	3	1
	15	7.077	63	4	1
	20	6.907	63	3	1
	30	6.902	63	3	1

[a]Total CPU time: CPU time in seconds to solve the 64 ALB problems on IBM 3600-600, with a three-second time trap for each problem, using the FORTRAN VS-opt3 compiler.
[b]No. of V. Opt.: the number of problems among 64 for which the verified optimal solutions were found by the optimal algorithm with a three-second time trap.
[c]LNS > .1: the number of problems among 64 for which the last new incumbent solution was found after .1 CPU seconds.
[d]LNS > .5: the same as "LNS > .1" except .5 CPU seconds instead of .1.

The last two columns of table 2 show that there were at most five (two) problems for which the last new incumbent solution was found after .1 (.5) second. This means that the algorithm found an optimal solution very quickly and most computation time was spent to verify its optimality by enumerating the tree and showing that there was no better solution. Thus, it is possible that the optimal algorithm with a small time trap can be used as a good heuristic.

We also solved one example to demonstrate the ALBP algorithm for FASs with parallel lines and multiple machine types. In the example, a part is processed first by machine type 1 and then by machine type 2. We used Sawyer's 30-task ALBP (1970) and Kilbridge and Wester's 45-task ALBP (1961) to characterize the group of tasks processed by machine type 1 and 2, respectively.[1] The staging capacities of the two machine types were specified as $R_1 = 20$ and $R_2 = 15$, and cycle time C as 54 seconds. The theoretical minimum number of machines per line was found to be two for machine type 1 and three for machine type 2. Algorithm 1 was applied to each group of tasks. The results are summarized in table 3.

Table 3 shows that the smallest number of machines required for the FAS is six type-1 machines that are configured as two or three parallel lines, and ten type-2 machines that are configured as one or two parallel lines. When there are multiple optimal solutions, we choose one with a larger number of parallel lines, since an FAS with more parallel lines is more reliable and has the smaller number of stations visited by a part, which subsequently requires a smaller number of material-handling operations. Hence, in this example, the FAS is configured as two stations, each with three parallel machines of type 1, which are followed by five stations, each with two parallel machines of type 2. This FAS is depicted in figure 5. In order to find a solution that balances average workload among solutions

Table 3. Deterministic analysis for the FAS with parallel lines and two machine types.

Machine Type	Number of Parallel Lines n_p	Minimum Number of Machines per Line M_p^*	Total Number of Machines $n_p \times M_p^*$	The Smallest Cycle Time at M_p^* C_s[a]
1	1	7	7	[b]
	2	3	6	[b]
	3	2	6	162
2	1	10	10	[b]
	2	5	10	106
	3	4	12	[b]

[a]The required cycle time is 54 for a single line.
[b]Not computed, since the balancing workload procedure was applied to only the optimum solution with more parallel lines.

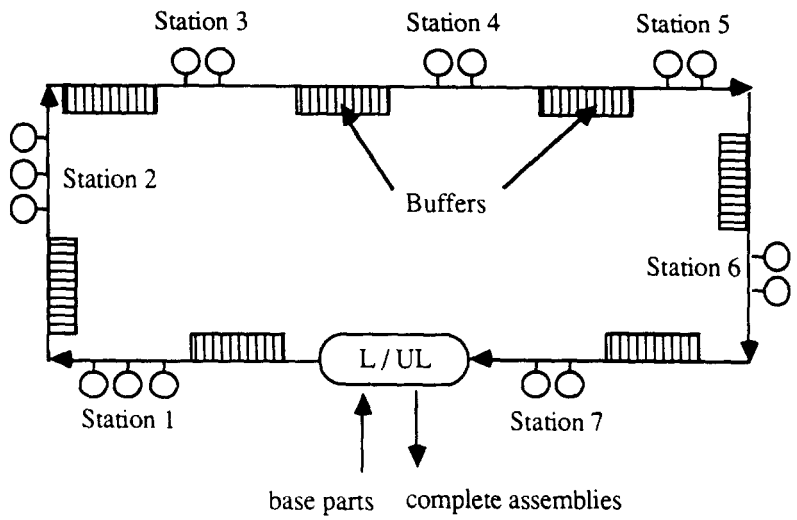

Figure 5. An example for deterministic analysis: an FAS with two machine types.

having this configuration, the balancing workload procedure was applied to $n_p = 3$ of machine type 1 and $n_p = 2$ of machine type 2, and the smallest cycle time was recorded in the last column of table 3. Note that two seconds were reduced for $n_p = 2$ of machine type 2.

7. Stochastic analysis (SA)

DA balances average workload per machine for a given cycle time C by determining task assignment to stations and the number of parallel machines at each station, with the objective of minimizing the machine cost only. SA complements DA by taking into account the variance of task times over different products. Taking the decisions from DA as input,

SA determines material-handling capacities. The objective is to minimize the WIP inventory cost plus the associated resource costs subject to a constraint of meeting the aggregate demand.

In order to build a model representing the FAS, we need to describe the dispatch policy that controls the release of base parts into the FAS. The dispatch policy under consideration releases a pallet containing a base part into the system when a pallet containing a complete assembly arrives at the L/UL station and leaves the system. This is the case when a base part is fixtured on a pallet outside the system and waits for a pallet containing a completed assembly to arrive at the L/UL station. At this point, the two pallets are exchanged by a human worker or automated pallet changer. Thus, under this policy a constant number of pallets and fixtures (and a constant WIP of base parts) perpetually circulate in the system. This policy has been widely used in the literature on FMSs (Solberg 1977; Stecke and Solberg 1985; Kamath et al. 1988). One reason for this is that it seems natural to regard the number of jobs in the system as the independent variable and the production rate of the system as the dependent variable (Whitt 1984). The number of jobs in the system is often subject to control. For example, in production systems, new jobs usually do not arrive at random; they are scheduled.

The FAS using this dispatch policy can be modeled as a single-class closed queueing network (CQN). We assume that service time at each station is exponentially distributed and that each station has sufficient buffer spaces such that the effect of buffer blocking on the production rate of FASs is negligible. No buffer blocking occurs when each station has N buffer spaces or more. With these two assumptions, we have a product-form CQN model for the FAS from which several performance measures can be computed exactly. A performance measure of particular interest to us is the throughput, which is defined as the number of pallets containing complete assemblies that leave the system per period. When the throughput is greater than or equal to the demand per period, we consider the FAS to be capable of meeting demand.

This product-form CQN model has been successful and widely used for effective representation of several practical systems, such as computer and communication systems and FMSs, despite the fact that the underlying assumptions are often seriously violated by real systems (Kleinrock 1976; Solberg 1977; Spragins 1980; Hildebrant 1981). Denning and Buzen (1978) explain this phenomenon using the concept of operational analysis. Suri (1983) provides a theoretical explanation that the main performance measures of the model, particularly throughput and utilization, are very robust to violations in the Markovian assumptions. In addition, readers can refer to Stecke and Solberg (1985) for more references for justifications and validations of the model.

The large variance of service time imposed by the exponential service time assumption can be justified in two ways for the FAS. First, the FAS assembles multiple products simultaneously, each of which may require a different set of tasks to be performed at an assembly station (see the example in section 3). Second, there are random perturbations affecting the system. These include machine tool jams, which occur when a machine jams while trying to insert a component. This small but regular disturbance (approximately once every 100 insertions in PCB assembly) can be modeled as part of the task time (Akella et al. 1988). The assumption of sufficient buffer spaces can be also justified to the PCB assembly system considered by Akella et al. (1988) where about 30 buffer spaces are available at each station.

The throughput per period, denoted as TH, is computed as a function of the following eight parameters: 1) the number of stations M, 2) the number of pallets N, 3) average transfer time between two stations T, 4) the number of AGVs S_o if required, 5) server vector $\bar{\mathbf{S}} = (S_1, S_2, \ldots, S_M)$ where S_i is the number of parallel machines at station i, 6) service time vector $\bar{\mathbf{W}} = (W_1, W_2, \ldots, W_M)$ where W_i is the sum of average task times assigned to station i, 7) processing capacity vector $\bar{\mathbf{P}} = (P_1, P_2, \ldots, P_M)$ where P_i is the total available processing time of a machine at station i per period, and 8) the total available handling time of an MHS (AGV or conveyor) per period P_o. Note that T, $\bar{\mathbf{P}}$, $\bar{\mathbf{W}}$, and P_o have a common time unit. After solving DA for a cycle time, we have M, $\bar{\mathbf{S}}$, and $\bar{\mathbf{W}}$ given. For example, for the FAS with two machine types in figure 5, we have $M = 7$ and $S_i = 3$ for $i = 1$ to 2 for the first two stations of machine type 1, and $S_i = 2$ for $i = 3$ to 7 for the following five stations of machine type 2.

Therefore, SA can be mathematically stated as follows.

Problem 2

$$\text{Min } z(N, S_o)$$

subject to

$$\text{TH}(M, \bar{\mathbf{S}}, \bar{\mathbf{W}}, \bar{\mathbf{P}}, P_o, T; N, S_o) \geq d \tag{6}$$

where z is any increasing cost function with respect to N and S_o. Letting W_o be the workload of MHSs required to produce one unit, we have $W_o = (M + 1)T$, since there are $M + 1$ stations including the L/UL station. Then, TH of equation (6) can be written below, using the central-server CQN model (Stecke and Solberg 1985) where the central server represents MHSs and is referred to as station 0. W denote $\mu_i = P_i/W_i$ where μ_i is a service rate per period by a server at station i for $i = 0$ to M. For convenience, we omit some obvious notation.

$$\text{TH}(N) = \frac{G(N-1)}{G(N)} \tag{7}$$

where $G(N)$ denotes the normalizing constant and is defined as

$$G(N) = \sum_{n_0+\ldots+n_M=N} \prod_{i=0}^{M} \prod_{j=0}^{n_i} f_i(j), \tag{8}$$

when n_i is a nonnegative integer for $i = 0$ to M and $f_i(j)$ is given as

$$\begin{aligned} f_i(j) &= 1 && \text{for } j = 0, \\ &= [\mu_i \cdot \min(j, S_i)]^{-1} && \text{for } j > 0. \end{aligned} \tag{9}$$

The throughput TH(N) can be computed using efficient algorithms such as the convolution algorithm (Buzen 1973) or the mean value analysis (MVA) algorithm (Reiser and Lavenberg 1980). TH(N) is approximate when MHSs are pick-and-drop AGVs[2] (i.e., $S_0 < N$), since the central-server CQN does not explicitly model contention of finished parts at stations over idle pick-and-drop AGVs to be delivered to the next stations. On the other hand, when MHSs are conveyors or stop-and-go AGVs[3] (i.e., $S_0 = N$), TH(N) is exact, since transfer time between two adjacent stations can be represented as a delay node and all the delay nodes ($M + 1$ nodes) can be exactly consolidated into one delay node (Posner and Bernholtz 1968). This delay node corresponds to station 0 of the central-server CQN model.

An algorithm to solve problem 2 can be easily developed using the property of the throughput by Shanthikumar and Yao (1987, 1988): TH is increasing and concave in N and S_0, respectively. Problem 2 is a special case of the problem studied by Vinod and Solberg (1985) and Dallery and Frein (1986), where decision variables are not only N and S_0 but also $\bar{\mathbf{S}}$. Their algorithms can be directly applied to solve problem 2.

We now briefly discuss how the proposed methodology can be extended to handle FASs with more complex topologies that allow splitting and merging of the flow of the assemblies due to repair loops for defective parts. For example, after a base part is completely processed at an assembly station, it is inspected and a defective part is sent to a repair station before it is sent to the next assembly station. Suppose that we are given defective ratio p and exponential repair time with average repair time τ for the repair station. We want to determine the number of servers at the repair station as well. Using station index $M + 1$ for the repair station, we have average workload $W_{M+1} = p\tau$ at the repair station. The product-form CQN model is still a valid representation for this FAS. Then SA is solved with the additional decision variable S_{M+1} using algorithms by Vinod and Solberg (1985) or Dallery and Frein (1986). As the number of repair stations increase, it takes more time to solve SA due to the increasing number of decision variables. One difficulty for this approach is that the defective ratio and repair time depend on a set of tasks performed at the assembly station and have to be estimated each time after DA is solved. FASs with feedback loops (or feedforward loops) are more challenging. For such FASs, the balanced workloads obtained from DA may not be achievable, since a part can be processed more than once in a station (or can skip a station). These issues are left for future research.

8. Monotonic behavior of the methodology

The line-balancing methodology generates a set of FAS design alternatives as the cycle time reduces from $1/d$ by trading off machine cost versus cost for other resources and WIP inventories. There is some evidence that this tradeoff behaves nicely. First, the following lemma states the monotonic behavior of the number of machines with respect to the cycle time.

Lemma 1. The number of machines obtained by DA is nondecreasing for each machine type as the cycle time C decreases from $1/d$.

Proof. Without any loss of generality, we fix a machine type arbitrarily and show that the lemma holds true. Let C_1 and C_2 be two cycle times such that $C_2 > C_1$. Suppose DA is

applied with each cycle time. Let $M_p^*(C_k)$ and $n_p^*(C_k)$ for $k = 1$ and 2 be the optimal number of stations and the optimal number of parallel lines obtained from DA with C_k, respectively. Clearly, the optimal task assignment from DA with C_1 is feasible to DA with C_2, that is, feasible to problem 1 with $M_p^*(C_1)$ and $C = n_p^*(C_1) \times C_2$. This is because $n_p^*(C_1) \times C_2 > n_p^*(C_1) \times C_1$ and other constraints of problem 1 are still met by the task assignment. Thus, the lemma follows directly.

Second, it appears that cost for material handling and WIP inventories also has a monotonic behavior with respect to the cycle time. When the cycle time is very small, the FAS is provided ample processing capacities and DA assigns many machines to the FAS. As a result, an arriving part seldom waits to be served, and the cost for material handling or WIP inventories can be minimal. In the other extreme case, when the cycle time is set to the initial value $1/d$, utilization must be near 1 to meet demand for some stations whose average service times are close to $1/d$. This will cause a long queue of parts in front of the stations, and the associated cost for material handling and WIP inventory will be maximal.

We now give one termination condition for the methodology. From lemma 1, we know that the resource cost for machines from DA is nondecreasing as the cycle time decreases. We also obtain lower bounds of the number of pallets and AGVs for SA using the asymptotic bound analysis (Muntz and Wong 1974). Subsequently, these bounds provide the lower bound of cost for those resources and WIP inventories. At a particular cycle time, if the cost of the incumbent solution is less than or equal to the sum of machine cost from DA at the cycle time and the lower bound of the cost for SA, then the methodology terminates, since it cannot find a better solution for any smaller cycle time.

9. Experiments with the line-balancing methodology

We conducted a number of experiments for the line-balancing methodology. The methodology was coded in FORTRAN and run on IBM 3090-600. The two-second time trap was used each time problem 1 was solved in DA. We set the size of cycle-time reduction, δ, to one time unit, since we used integer-valued task times.

We used the following parameter values for the aggregate product. We experimented with two demand levels for a period: $d = 100$ and 300. The number of tasks was set to $n = 100$. Task times were randomly generated from a discrete uniform distribution ranging from 1 to 10. The precedences between tasks was also randomly generated using a density parameter that was defined as the ratio of a number of present precedent arcs to the total number of possible precedent arcs, i.e., $\binom{n}{2}$. Each arc was equally likely. We experimented with two density values, 0.1 and 0.5. A graph with a higher density was generated by adding arcs to a graph with a lower density until the desired density was reached. Redundant precedent arcs are counted in the density computation.

We used the following parameter values for the resources. We considered an FAS with one machine type. Its staging capacity was specified as $R = 30$, which was used by Ammons et al. (1985) for a PCB assembly system manufacturing computers. The available processing (handling) time of each machine (MHS) was given as 10,000/period and the average transfer time between two stations was given as 20, about average processing time for four tasks.

We assumed that conveyors or stop-and-go AGVs were in use to move pallets between stations. This is realistic because in an FAS with high demand and short task times, an AGV is unlikely to drop the pallet off and then go and service other material-handling requirements (Hall and Stecke 1986). We had the total system cost specified as the following linear function:

$$\text{TC} = C_S \times \sum_{i=1}^{M} S_i + C_N \times N$$

where C_S is resource cost per machine and C_N is the sum of cost for one WIP inventory and resource cost for one AGV (if required), one pallet, and one fixture. We experimented with three sets of (C_S, C_N): (2000, 50), (2000, 400), and (2000, 1200). Note that since the total cost function is linear, the ratio of C_S to C_N only counts.

Experimental results for two replications are summarized in tables 4 and 5. For each problem, the following statistics are recorded: the best solution (M, N, (S_i)) found, its total cost (TC) and throughput TH, the number of times that the two analyses (DA and SA) were solved, and total CPU time taken. The assignment of tasks to stations were not shown for simplicity. All the solutions listed in both tables were verified to be the optimum, since problem 1 was successfully solved each time before the two-second time trap expired.

When demand d increased from 100 to 300, more machines and material-handling equipment were needed, as expected. The solutions obtained were insensitive to the density of the graph except that CPU time slightly increased for the higher density. Within each replication, the solutions were identical for the two densities. One possible interpretation for this insensitivity is that the staging capacity ($R = 30$) is large enough so that additional precedent arcs do not have an adverse effect on the assignment of tasks to stations. As C_N increased from 50, the number of pallets N decreased and the total number of machines, M times S_i, increased in order to look for a better tradeoff between the two cost terms in the linear function.

Table 4. Experiment with the line-balancing methodology: replication 1 with $R = 30$.

Problem (d, density, C_S, C_N)	Solution M, N, (S_i)	Total Cost TC	Throughput TH	Number of Analyses	CPU Time (sec.)
(100, 0.1, 2000, 50)	7, 32, (1)	15,600	100.2	3	1.4
(100, 0.1, 2000, 400)	4, 11, (2)	20,400	100.4	4	1.6
(100, 0.1, 2000, 1200)	4, 11, (2)	29,200	100.4	5	2.0
(100, 0.5, 2000, 50)	7, 32, (1)	15,600	100.2	3	3.1
(100, 0.5, 2000, 400)	4, 11, (2)	20,400	100.4	4	3.8
(100, 0.5, 2000, 1200)	4, 11, (2)	29,200	100.4	5	5.0
(300, 0.1, 2000, 50)	4, 34, (5)	41,700	301.2	4	2.5
(300, 0.1, 2000, 400)	4, 34, (5)	53,600	301.2	5	2.7
(300, 0.1, 2000, 1200)	4, 34, (5)	80,800	301.2	5	2.8
(300, 0.5, 2000, 50)	4, 34, (5)	41,700	301.2	4	4.3
(300, 0.5, 2000, 400)	4, 34, (5)	53,600	301.2	5	4.9
(300, 0.5, 2000, 1200)	4, 34, (5)	80,800	301.2	5	4.9

Table 5. Experiment with the line-balancing methodology: replication 2 with $R = 30$.

Problem (d, density, C_S, C_N)	Solution M, N, (S_i)	Total Cost TC	Throughput TH	Number of Analyses	CPU Time (sec.)
(100, 0.1, 2000, 50)	6, 63, (1)	15,150	100.1	3	1.2
(100, 0.1, 2000, 400)	4, 10, (2)	20,000	101.6	4	1.5
(100, 0.1, 2000, 1200)	4, 10, (2)	28,000	101.6	5	1.8
(100, 0.5, 2000, 50)	6, 63, (1)	15,150	100.1	3	2.6
(100, 0.5, 2000, 400)	4, 10, (2)	20,000	101.6	4	3.4
(100, 0.5, 2000, 1200)	4, 10, (2)	28,000	101.6	5	4.2
(300, 0.1, 2000, 50)	6, 72, (3)	39,600	300.1	4	2.1
(300, 0.1, 2000, 400)	4, 28, (5)	51,200	302.2	6	2.5
(300, 0.1, 2000, 1200)	4, 28, (5)	73,600	302.2	7	2.7
(300, 0.5, 2000, 50)	6, 72, (3)	39,600	300.1	4	3.6
(300, 0.5, 2000, 400)	4, 28, (5)	51,200	302.2	6	4.9
(300, 0.5, 2000, 1200)	4, 28, (5)	73,600	302.2	7	5.4

We also conducted experiments to see the effect of machine flexibility on the solution obtained by the methodology. We considered another type of FAM that can perform all the tasks with the same speed but is less flexible and less expensive. This machine type is capable of processing only up to ten tasks simultaneously, i.e., its staging capacity is 10. We used $C_S = 1500$. The rest of the problem parameters remain unchanged. The methodology was reapplied, and experimental results are summarized in tables 6 and 7.

For $R = 10$, the FAS needs at least ten stations to accommodate 100 tasks, compared to four stations for $R = 30$. Machines spread over the larger number of stations, which leads to the smaller number of parallel machines. A part needs to visit more stations, which leads to more material-handling operations and subsequently larger cost for the MHS. This becomes evident for larger C_N. When $C_N = 1200$, the solutions with $R = 10$ cost more than their counterparts with $R = 30$, except for one case (see table 6). However, this disadvantage is somewhat offset at larger demand, since larger demand requires more machines

Table 6. Experiment with the line-balancing methodology: replication 1 with $R = 10$.

Problem (d, density, C_S, C_N)	Solution M, N, (S_i)	Total Cost TC	Throughput TH	Number of Analyses	CPU Time (sec.)
(100, 0.1, 1500, 50)	10, 15, (1)	15,750	100.5	2	1.3
(100, 0.1, 1500, 400)	10, 15, (1)	21,000	100.5	3	1.4
(100, 0.1, 1500, 1200)	10, 15, (1)	33,000	100.5	6	1.7
(100, 0.5, 1500, 50)	10, 16, (1)	15,800	100.1	2	1.5
(100, 0.5, 1500, 400)	10, 16, (1)	21,400	100.1	3	2.1
(100, 0.5, 1500, 1200)	11, 14, (1)	33,300	100.5	6	3.3
(300, 0.1, 1500, 50)	10, 75, (2)	33,750[a]	300.4	5	3.7
(300, 0.1, 1500, 400)	12, 43, (2)	53,200[a]	301.1	10	4.6
(300, 0.1, 1500, 1200)	10, 28, (3)	78,600[a]	301.7	13	5.5
(300, 0.5, 1500, 50)	11, 54, (2)	35,700[a]	300.4	6	8.4
(300, 0.5, 1500, 400)	12, 43, (2)	53,200[a]	300.5	11	11.1
(300, 0.5, 1500, 1200)	13, 39, (2)	85,800	301.7	15	13.0

[a]This solution costs less than its counterpart with $R = 30$ in table 4.

Table 7. Experiment with the line-balancing methodology: replication 2 with $R = 10$.

Problem (d, density, C_S, C_N)	Solution M, N, (S_i)	Total Cost TC	Throughput TH	Number of Analyses	CPU Time (sec.)
(100, 0.1, 1500, 50)	10, 14, (1)	15,700	101.8	2	0.6
(100, 0.1, 1500, 400)	10, 14, (1)	20,600	101.8	3	0.7
(100, 0.1, 1500, 1200)	10, 14, (1)	31,800	101.8	6	1.0
(100, 0.5, 1500, 50)	10, 14, (1)	15,700	101.5	2	1.4
(100, 0.5, 1500, 400)	10, 14, (1)	20,600	101.5	3	1.8
(100, 0.5, 1500, 1200)	10, 14, (1)	31,800	101.5	6	2.8
(300, 0.1, 1500, 50)	10, 56, (2)	32,800[a]	300.0	5	4.1
(300, 0.1, 1500, 400)	11, 44, (2)	50,600[a]	301.5	9	4.6
(300, 0.1, 1500, 1200)	10, 27, (3)	77,400	306.8	14	6.2
(300, 0.5, 1500, 50)	11, 44, (2)	35,200[a]	301.1	7	8.6
(300, 0.5, 1500, 400)	11, 44, (2)	50,600[a]	301.1	10	11.4
(300, 0.5, 1500, 1200)	12, 28, (2)	81,600	300.4	13	12.7

[a]This solution costs less than its counterpart with $R = 30$ in table 5.

and since a machine with $R = 10$ costs 500 less than the other. When demand $d = 300$, more than half of the solutions with $R = 10$ cost less than their counterparts (see the footnote in tables 6 and 7). On the other hand, when $d = 100$, all the solutions with $R = 10$ cost more than their counterparts. It is important to mention that in this comparative study between two machine types, we did not take into account other design factors such as reliability or routing flexibility for which the solutions with $R = 30$ are much favored over their counterparts.

These experimental results show that the proposed methodology provides a small number of cost-effective design alternatives quickly over a broad range of parameter values. For all the problems tested, no more than 15 seconds of CPU time was required to find the verified optimal solutions on an IBM 3090-600.

10. Conclusions

In this article, we presented a methodology to design an FAS, organized as a flow system, in which different products are assembled simultaneously. This methodology uses two analytical models (integer programming and queueing network) and addresses five critical design decisions in an integrated manner. These decisions are the number of stations, the number of parallel machines at each station, task assignment to stations, the number of AGVs (if required), and the number of pallets and fixtures. Common approaches addressing these design issues rely on simulation, which is very time-consuming and costly when many alternatives must be evaluated at the early stages of design. Other approaches using analytical methods often ignore material-handling issues, or address only isolated issues, assuming that other decisions are already given. Through a large number of experiments, we demonstrated that the methodology indeed meets our research objective of providing a small number of cost-effective designs quickly under a broad range of parameter values. The methodology

enables us to quantitatively evaluate the effect of machine flexibility on the design alternative where machine flexibility is measured as the number of tasks that can be processed with negligible setup times between task changes.

The methodology is based on a strategy of balancing average workload per machine that is commonly used when designing traditional manufacturing systems as well as FMSs. The methodology takes the form of an iterative procedure that repeatedly solves deterministic and stochastic analyses as the cycle time decreases. DA balances the average workload among machines while SA takes into account the variance of processing times. The methodology generates a set of FAS design alternatives with respect to the cycle time by trading off machine cost versus cost for other resources and WIP inventories. We also showed some evidence that this tradeoff behaves nicely.

For DA, we presented an optimal algorithm for solving the ALBP for FASs. The optimal algorithm generalized Johnson's algorithm for the traditional ALBP, based on the justifications given earlier. The computational results showed that the algorithm with a three-second time trap found from 55 to 64 verified optimal solutions among 64 ALBPs, depending on the staging capacity, and that computation time is sensitive to the value of the staging capacity. The results also showed that the optimal algorithm found a (near) optimal solution very quickly; hence, the optimal algorithm with a small time trap such as .1 CPU seconds for about 100 tasks can be used as a good heuristic. Search in DA was confined to only balanced configurations in which the number of machines is identical among stations using the same machine type. This search may be rigid and may need to include unbalanced configurations in future research.

For SA, we used a single-class product-form CQN model to represent the FAS from which the throughput of the FAS was estimated. With some of CQN parameters specified from DA, SA determines design decisions for material-handling resources. In future research, we can use a more realistic queueing model. For example, in FASs assembling PCBs, the mean time between failures for insertion robots is of the order of ten hours, while the mean time to repair the machines is approximately an hour (Akella et al. 1988). Our CQN does not explicitly model this phenomenon. For this purpose, SA can use Vinod and Sabbagh's (1986) approximation method to compute the throughput of an FAS. SA can also incorporate other job-dispatch policies to the FAS. This necessitates other means of estimating the throughput of the FAS (Shanthikumar and Stecke 1986), but the rest of the methodology remains unaffected. One step further, SA can be interfaced to simulation for detailed analysis or to address other design decisions such as buffer size selection and subpart delivery policies (Ho et al. 1984; Liu and Sanders 1988; Yano et al. 1988). In addition, we briefly discussed, in the SA section, extensions and difficulties of the methodology to design FASs with more complex topologies such as repair loops. These issues are also left for future research.

Acknowledgments

We thank M.M. Srinivasan and C.A. Yano for their helpful comments on this article. Comments and suggestions of the editor, the associate editor, and the referees improved the quality and presentation of this article and are greatly appreciated.

Notes

1. We changed one task duration among Kilbridge and Wester's 45 tasks from 55 seconds to 30, since no task duration can be longer than the cycle time, 54 seconds.

2. A pick-and-drop AGV delivers a pallet to a station, and then leaves for another station to serve other material-handling requests before it has to return to the same station to pick up the pallet.

3. A stop-and-go AGV is dedicated to each pallet from start of the assembly to the finish.

References

Akella, R., Singh, M., and Bassok, Y., "Real Time Part Dispatch in Flexible Assembly Test and Manufacturing Systems," Technical Report, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA (1988).

Ammons, J.C., Lofgren, C.B., and McGinnis, L.F., "A Large Scale Machine Loading Problem in Flexible Assembly," *Annals of Operations Research*, Vol. 3, pp. 319–332 (1985).

Baybars, I., "A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem," *Management Science*, Vol. 32, No. 8, pp. 909–932 (August 1986).

Berrada, M. and Stecke, K.E., "A Branch and Bound Approach for Machine Load Balancing in Flexible Manufacturing Systems," *Management Science*, Vol. 32, No. 10, pp. 1316–1335 (October 1986).

Boothroyd, G., Poli, C., and Murch, L., *Automatic Assembly*, Marcel Dekker, New York, NY (1982).

Browne, J., Chan, W., and Rathmill, K., "An Integrated FMS Design Procedure," *Annals of Operations Research*, Vol. 3, pp. 207–237 (1985).

Bulgak, A. and Sanders, J., "Hybrid Algorithms for Design Optimization of Asynchronous Flexible Assembly Systems with Statistical Process Control and Repair," *Proceedings of the 3rd ORSA/TIMS Conference on Flexible Manufacturing Systems*, M.I.T., Cambridge, MA, K. Stecke and R. Suri (Eds.), Elsevier Science Publishers B.V., Amsterdam, pp. 275–280 (August 1989).

Buzen, J.P., "Computational Algorithms for Closed Queueing Networks with Exponential Servers," *Communications of the Association of Computing Machinery*, Vol. 16, No. 9, pp. 527–531 (1973).

Dallery, Y. and Frein, Y., "An Efficient Method to Determine the Optimal Configuration of a Flexible Manufacturing System," *Proceedings of the 2nd ORSA/TIMS Conference on Flexible Manufacturing Systems*, Ann Arbor, MI, K. Stecke and R. Suri (Eds.), Elsevier Science Publishers B.V., Amsterdam, pp. 269–282 (August 1986).

Dallery, Y. and Stecke, K., "On the Optimal Allocation of Servers and Workloads in Closed Queueing Networks," *Operations Research*, Vol. 38, No. 4, pp. 694–703 (July–August 1990).

Denning, P. and Buzen, J., "The Operational Analysis of Queueing Network Models," *Association of Computing Machinery Computing Surveys*, Vol. 10, No. 3, pp. 225–261 (1978).

Gershwin, S., Akella, R., and Choong, Y., "Short-term Production Scheduling of an Automated Manufacturing Facility," *Annals of Operations Research*, Vol. 3, pp. 392–400 (1985).

Graham, G.S., "Queueing Network Models of Computer System Performances," *Computing Surveys*, Vol. 10, No. 3, pp. 219–224 (1978).

Graves, S.C. and Redfield, C.H., "Equipment Selection and Task Assignment for Multiproduct Assembly System Design," *International Journal of Flexible Manufacturing Systems*, Vol. 1, No. 1, pp. 31–50 (September 1988).

Groover, M., Weiss, M., Nagel, R., and Odrey, N. *Industrial Robotics*, McGraw-Hill, New York, NY (1986).

Hall, D.N. and Stecke, K.E., "Design Problems of Flexible Assembly Systems," *Proceedings of the 2nd ORSA/TIMS Conference on Flexible Manufacturing Systems*, Ann Arbor, MI, K. Stecke and R. Suri (Eds.), Elsevier Science Publishers B.V., Amsterdam, pp. 145–156 (August 1986).

Hildebrant, R., "Scheduling Flexible Machining Systems Using Mean Value Analysis," *Proceedings of the IEEE Conference on Decision and Control*, IEEE, New York, pp. 701–706 (December 1981).

Hitz, K., "Flexible Integrated Computer-aided Manufacturing Systems Increase Productivity," *Robotics and Computer-Integrated Manufacturing*, Vol. 3, No. 1, pp. 123–128 (1987).

Ho, Y., Suri, R., Cao, X., Diehl, G., Dille, J., and Zazanis, M., "Optimization of Large Multiclass (Non-product-form) Queueing Networks Using Perturbation Analysis," *Large Scale Systems*, Vol. 7, pp. 1–16 (1984).

Johnson, R.V., "Optimally Balancing Large Assembly Lines with 'FABLE,' " *Management Science*, Vol. 34, No. 2, pp. 240–253 (February 1988).

Kamath, M., Suri, R., and Sanders, J., "Analytical Performance Models for Closed-Loop Flexible Assembly Systems," *International Journal of Flexible Manufacturing Systems*, Vol. 1, No. 1, pp. 51–84 (September 1988).

Kilbridge, M. and Wester, L., "A Heuristic Method of Assembly Line Balancing," *Journal of Industrial Engineering*, Vol. 12, pp. 292–298 (1961).

Kleinrock, L., *Queueing Systems II*, John Wiley and Sons, New York, New York (1976).

Lee, H.F., Srinivasan, M.M., and Yano, C.A., "Algorithms for the Minimum Cost Configuration Problem in Flexible Manufacturing Systems," *Proceedings of the 3rd ORSA/TIMS Conference on Flexible Manufacturing Systems*, M.I.T., Cambridge, MA, K. Stecke and R. Suri (Eds.), Elsevier Science Publishers B.V., Amsterdam, pp. 85–90 (August 1989).

Liu, C. and Sanders, J., "Stochastic Design Optimization of Asynchronous Flexible Assembly Systems," *Proceedings of the 2nd ORSA/TIMS Conference on Flexible Manufacturing Systems*, Ann Arbor, MI, K. Stecke and R. Suri (Eds.), Elsevier Science Publishers B.V., Amsterdam, pp. 191–202 (August 1986).

Liu, C. and Sanders, J., "Stochastic Design Optimization of Asynchronous Flexible Assembly Systems," *Annals of Operations Research*, Vol. 15, pp. 131–154 (1988).

Muntz, R.R. and Wong, J.W., "Asymptotic Properties of Closed Queueing Network Models," *Proceedings of the 8th Annual Princeton Conference on Information Sciences and Systems*, Princeton University, Princeton, New Jersey (1974).

Owen, T., *Flexible Assembly Systems*, Plenum Press, New York (1984).

Owen, T., *Assembly with Robots*, Prentice Hall, Englewood Cliffs, New Jersey (1985).

Posner, M. and Bernholtz, B., "Closed Finite Queueing Networks with Time Lags," *Operations Research*, Vol. 16, No. 5, pp. 962–976 (September–October 1968).

Pourbabai, B., "Optimal Control of a Flexible Assembly System," *Proceedings of the 1987 ASME Design Technology Conference on Advances in Design Automation*, Boston, MA, Vol. 1, pp. 335–337 (1987).

Ranky, P.G., "The Design of an End of Arm Tool Management System for Flexible Assembly Systems Utilizing Industrial Robots," Report No. RSD-TR-21-86, Center for Research on Integrated Manufacturing, College of Engineering, The University of Michigan, Ann Arbor, Michigan (1986).

Reiser, M. and Lavenberg, S., "Mean-Value Analysis of Closed Multichain Queueing Networks," *Journal of the Association for Computing Machinery*, Vol. 27, No. 2, pp. 313–322 (1980).

Riley, F. and Yarrow, E., "A New Approach to Assembly Machine Justification," *Proceedings of the 2nd European Conference on Automated Manufacturing*, Birmingham, U.K. (1983).

Sawyer, J.F., *Line Balancing*, Machinery and Allied Products Institute, Washington, D.C. (1970).

Schrage, L. and Baker, K., "Dynamic Programming Solution of Sequencing Problems with Precedence Constraints," *Operations Research*, Vol. 26, No. 3, pp. 444–459 (May–June 1978).

Seliger, G. and Wieneke, B., "Analytical Approach for Function Oriented Production System Design," *Robotics and Computer-Integrated Manufacturing*, Vol. 1, No. 3, pp. 307–313 (1984).

Seliger, G., Wiehweger, B., and Wieneke, B., "Descriptive Methods for Computer-Integrated Manufacturing and Assembly," *Robotics and Computer-Integrated Manufacturing*, Vol. 3, No. 1, pp. 15–21 (1987a).

Seliger, G., Wiehweger, B., and Wieneke, B., "Decision Support in Design and Optimization of Flexible Automated Manufacturing and Assembly," *Robotics and Computer-Integrated Manufacturing*, Vol. 3, No. 2, pp. 221–227 (1987b).

Shanthikumar, J.G. and Stecke, K.E., "Reducing Work-in-Process Inventory in Certain Classes of Flexible Manufacturing Systems," *European Journal of Operational Research*, Vol. 26, pp. 266–271 (1986).

Shanthikumar, J.G. and Yao, D.D., "Optimal Server Allocation in a System of Multi-Server Stations," *Management Science*, Vol. 33, No. 9, pp. 1173–1180 (September 1987).

Shanthikumar, J.G. and Yao, D.D., "On Server Allocation in Multiple Center Manufacturing Systems," *Operations Research*, Vol. 36, No. 2, pp. 333–342 (March–April 1988).

Shanthikumar, J.G. and Yao, D.D., "Second-order Properties of the Throughput of a Closed Queueing Network," *Mathematics of Operations Research*, Vol. 13, No. 3, pp. 524–534 (August 1988).

Shanthikumar, J.G. and Yao, D.D., "Optimal Buffer Allocation in a Multicell System," *International Journal of Flexible Manufacturing Systems*, Vol. 1, No. 4, pp. 347–356 (September 1989).

Solberg, J., "A Mathematical Model of Computerized Manufacturing Systems," *Proceedings of the 4th International Conference of Production Research*, Tokyo, Japan (August 1977).

Spragins, J., "Analytical Queueing Models: Guest Editor's Introduction," *IEEE Computer*, Vol. 13, No. 4, pp. 9–11 (1980).

Spur, G., Furgac, I., Deutschlander, A., Browne, J., and O'Gorman, P., "Robot Planning System," *Robotics and Computer-Integrated Manufacturing*, Vol. 2, No. 2, pp. 115–123 (1985).

Spur, G., Furgac, I., and Kirchhoff, U., "Robot System Integration into Computer-Integrated Manufacturing," *Robotics and Computer-Integrated Manufacturing*, Vol. 3, No. 1, pp. 1–10 (1987).

Stecke, K.E., "Formulation and Solution of Nonlinear Integer Production Problems for Flexible Manufacturing Systems," *Management Science*, Vol. 29, No. 3, pp. 273–288 (March 1983).

Stecke, K.E. and Solberg, J.J., "The Optimality of Unbalancing Both Workloads and Machine Group Sizes in Closed Queueing Networks of Multi-Server Queues," *Operations Research*, Vol. 33, No. 4, pp. 882–910 (July–August 1985).

Suri, R., "Robustness of Queueing Network Formulae," *Journal of the Association of Computing Machinery*, Vol. 30, No. 3, pp. 564–594 (1983).

Suri, R. and Diehl, G., "MANUPLAN—a Precursor to Simulation for Complex Manufacturing Systems," *Proceedings of the Winter Simulation Conference* (1985).

Suri, R. and Diehl, G., "Rough-Cut Modeling: An Alternative to Simulation," *CIM Review*, Vol. 3, pp. 25–32 (1987).

Talbot, F.B. and Patterson, J.H., "An Integer Programming Algorithm with Network Cuts Solving the Assembly Line Balancing Problem," *Management Science*, Vol. 30, No. 1, pp. 85–99 (January 1984).

Talbot, F.B., Patterson, J.H., and Gehrlein, W.V., "A Comparative Evaluation of Heuristic Line Balancing Techniques." *Management Science*, Vol. 32, No. 4, pp. 430–454 (April 1986).

Thomopolous, N.T., "Mixed-Model Line Balancing with Smoothed Station Assignments," *Management Science*, Vol. 16, No. 9, pp. 593–603 (May 1970).

Vinod, B. and Sabbagh, M., "Optimal Performance Analysis of Manufacturing Systems Subject to Tool Availability," *European Journal of Operational Research*, Vol. 24, pp. 398–409 (1986).

Vinod, B. and Solberg, J., "The Optimal Design of Flexible Manufacturing Systems," *International Journal of Production Research*, Vol. 23, No. 6, pp. 1141–1151 (1985).

Whitney, C.K. and Suri, R., "Algorithms for Part and Machine Selection in Flexible Manufacturing Systems," *Annals of Operations Research*, Vol. 3, pp. 239–261 (1985).

Whitt, W., "Open and Closed Models for Networks of Queues," *AT&T Bell Laboratories Technical Journal*, Vol. 63, pp. 1911–1979 (1984).

Whitt, W., "The Best Order of Queues in Series," *Management Science*, Vol. 31, No. 4, pp. 475–487 (April 1985).

Yano, C.A., Lee, H.F., and Srinivasan, M.M., "Issues in the Design and Operation of Flexible Assembly Systems for Large Products: a Simulation Study," Technical Report 88-10, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, Michigan, to appear in *Journal of Manufacturing Systems* (1988).

Yao, D.D. and Buzacott, J.A., "Queueing Models for a Flexible Machining Station. Part I: the Diffusion Approximation," *European Journal of Operational Research*, Vol. 19, pp. 233–240 (1985).