A FRAMEWORK FOR CAPACITY PLANNING AND MACHINE CONFIGURATION IN FLEXIBLE ASSEMBLY SYSTEMS

Heungsoon Felix Lee
Department of Industrial Engineering
Southern Illinois University
Edwardsville, IL 62026-1802

Mandyam M. Srinivasan
and
Candace Arai Yano
Department of Industrial & Operations Engineering
University of Michigan

Technical Report 92-38

September 1991
Revised May 1992

A Framework for Capacity Planning and Machine Configuration in Flexible Assembly Systems

Abstract

We consider the problem of simultaneously determining the number of machines (and/or workers), the assignment of tasks (and related tools and components) to these machines, and the number of jobs circulating in a flexible assembly system, to satisfy steady-state throughput requirements for a family of similar products at minimum cost. We focus on situations where there are precedence relations among the various tasks, as is common in assembly systems. We present a framework for solving this problem based on a heuristic decomposition approach which involves the solution of only a few types of subproblems. We demonstrate the efficiency and effectiveness of the overall procedure using a number of example problems.

A Framework for Capacity Planning and Machine Configuration in Flexible Assembly Systems

1. Introduction

Many companies producing automobiles, electronic components, computers, and electric consumer products are in the process of designing or implementing flexible assembly systems (FASs) because an increasing number of product variations, shorter product life cycles, the need to react rapidly to short–term variations of demand, and a highly competitive market, all necessitate a more flexible means of production (Owen 1984 and Spur et al. 1987).

Making decisions regarding equipment acquisitions in these contexts is not easy. Unlike more traditional 'inflexible' systems, many different products may be assembled concurrently, and the production rate of the system as a whole may be very sensitive to exactly which machines are performing which tasks and the resultant product routings. Consequently, standard capacity planning models that simply constrain the total processing time assigned to a machine to be less than the total number of machine hours available are unable to account for the impact of these factors.

Clearly, important equipment acquisition decisions should not be made without a very careful and detailed analysis. Yet, oftentimes, the most difficult part of this process is "getting in the right ballpark," and understanding the interactive effects of various decisions. Our goal in this paper is to present a framework that permits one to identify "ballpark" solutions quickly. In the process, we also provide tools that allow a better understanding of the interactive effects of various operational decisions. We consider systems with a particular structure in order to illustrate our framework. Systems with significantly different features may need to be modeled and analyzed using different methods. However, our framework provides a pragmatic approach to decomposing the problem into manageable pieces, and we view this decomposition as the primary contribution of this paper. We now turn to a description of the structure of the system that we use to illustrate our framework.

We consider flexible assembly *flow* systems, which consist of multiple workstations in series, where each workstation contains one or more machines (and/or workers, as appropriate) with identical capabilities. The system is flexible in the sense that each machine is capable of performing a variety of different tasks. In the case of automated systems, this flexibility is achieved as a consequence of loading multiple tools, and/or a collection of numerical-control (NC) programs. With less automated systems, flexibility is achieved through training (of workers), in addition to loading of multiple tools and/or NC programs. A part proceeds from one workstation to the next, in the specified sequence, bypassing those where no work needs to be performed.

In most assembly systems, processing times are short relative to the time required to move a job from machine to machine. Consequently, performing multiple tasks on the same machine may considerably reduce job flow time and material handling effort. More than one machine may be assigned to a workstation to ensure that there is sufficient capacity to satisfy production targets. When a part needs to be processed by a workstation, it is processed by only one machine at that station, since all machines in the workstation are capable of performing the same set of tasks.

Our reason for considering multiple workstations is that it is often impossible or impractical for a single machine or worker to perform all of the tasks required to complete a finished product. In assembly systems, one primary reason for this is that all of the assembly components cannot be conveniently stored at or near the workstation. We associate a *staging space* with each task: typically, the staging space would be based on the size of the various assembly components and the associated feeders needed for the task (see Figure 1). Although a machine may be technically capable of performing many tasks, due to either space restrictions or other types of restrictions it can perform only a subset of the tasks at any one time. We reflect this in a *staging capacity* constraint for each machine. Such circumstances arise in automobile assembly because of the size of the various parts; they also occur in the electronics industry as a consequence of the design of assembly equipment.

When the need arises for multiple workstations, the machines are often configured as a flow system. There are a variety of reasons for this, which we do not discuss in detail here. It is evident, however, that a flow layout simplifies material handling and production control. Note that the highly-touted U-shaped layout (Harmon and Peterson 1990) is an example of a flow system.

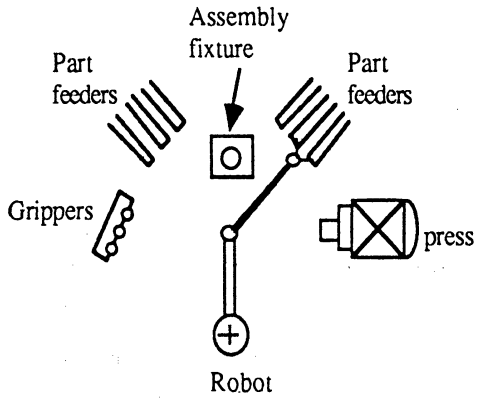

Figure 1. A robot assembly cell

In this paper we consider the problem of simultaneously determining the number of machines (and/or workers), the assignment of tasks (and related tools and components) to these machines, and the number of jobs circulating in a flexible assembly *flow* system, to satisfy steady-state throughput requirements for a family of similar products at minimum cost. We present a

framework for solving the problem of finding the minimum-cost design of such a system. We focus on situations with a single machine type (e.g., human workers, general purpose pick-and-place robots), which, in turn, allows us to consider precedence relations among the tasks. To our knowledge, most of the existing work on related problems has not considered precedence relations, assuming instead, that parts were allowed multiple visits to each station if necessary. Since we are considering flow systems, precedence relations must be accounted for explicitly in the assignment of tasks to the stations.

We assume that the steady–state mix of products has already been determined. In the concluding section of the paper we discuss how the procedure might be adapted and extended to consider situations with unstable product mixes. If there are equipment alternatives, our procedure can be applied to each alternative.

The primary decision variables are the number of machines and their allocation among the stations, the assignment of tasks (or operations) to the machines, and the number of pallets, fixtures, and/or material handling equipment to support jobs circulating in the system. The goal is to find a design that satisfies a given throughput for a specified product mix at minimum cost. The objective function may include both amortized capital costs and annual operating costs.

Comprehensive design and planning aids for FASs have yet to be developed. There is a need for design methods that can incorporate complex interdependencies among the various decisions and provide a small number of good design alternatives quickly, to which more detailed analysis such as simulation can be applied. The main contribution of this paper is the development of a framework for solving this problem based on a heuristic decomposition approach. The decomposition breaks the large problem into a small number of subproblem types, each of which can be solved efficiently by procedures reported in the the literature. Consequently, for variations of the problem, only portions of the solution procedure need to be changed within our framework. Our approach also provides *ex post* error bounds for the problem. We should emphasize that our goal is not one of demonstrating the efficiency of the procedure, since little effort has been devoted to making the computer programs efficient. Rather, our intent is to describe the framework and to demonstrate its effectiveness in bringing a very complex problem within the realm of CPU times that could be expended for such large capital acquisition decisions.

2. Literature Review

Flexible assembly systems are a subset of flexible manufacturing systems (FMSs). These systems have received little attention in the research literature, since much of the work has focused on Flexible Machining Systems (FMaSs), which typically perform metal cutting and/or metal forming operations.

Flexible assembly systems differ from FMaSs in several important respects. First, FASs often have many machines of the same type, whereas a typical FMaS has only one or a few machines of each type. Further, the different machines in an FAS are somewhat more substitutable than those in an FMaS. This is a consequence of the differences between assembly operations and machining operations. Second, the number of different tasks actually performed by an FAS is usually much larger than the number of different tasks performed by a typical FMaS, since in many instances an FAS will assemble hundreds or thousands of different components (e.g., printed circuit board assembly) into one product. These factors usually make it more difficult to design FASs because of the much larger number of decisions that must be made.

There are relatively few papers on FAS design problems in the literature, and only a few papers on FMS design that are applicable in an FAS context. The related literature can be divided into three groups on the basis of the modeling techniques employed: simulation, queueing networks, and integer programming.

Simulation can be used to represent an FAS at any level of detail. However, in the early stages of design of these complex systems, 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, and to validate and run simulation software for many design alternatives before one good alternative is chosen. Further, we cannot assess the quality of the selected alternative because simulation does not usually provide an optimal solution or benchmark with which the chosen alternative can be compared.

Queueing network models have been used in the solution of flexible manufacturing (FMS) design problems. A product form (PF) single–class closed queueing network (CQN) model (Jackson 1963, Gordon and Newell 1967) is commonly used. Vinod and Solberg (1985) and Dallery and Frein (1986) use PF CQN models in determining the number of flexible machines at each station, and the number of pallets assuming the number of stations and their workloads are known. The objective is to minimize the capital and operating costs while meeting production requirements. Shanthikumar and Yao (1987, 1988a) study the problem of allocating a given number of flexible machines among stations to maximize the throughput rate. They assume that the number of stations and pallets, and the workloads at the stations are known.

The papers based on queueing network models deal with specific decisions (such as the allocation of flexible machines to stations), *assuming that many other design decisions (such as the number of stations and the workloads) are already known in advance.* Also, they do not consider task assignments. Generally, queueing network models provide good estimates of performance measures such as throughput and station utilizations for a given physical configuration and task

assignment. However, in general these measures cannot be expressed in closed form. As a consequence, it is very difficult to use these models directly to prescribe the optimal configuration and the task assignments since there are several types of discrete decisions involving many variables.

Hybrid methodologies involving queueing network models and simulation have been proposed. Seliger et al. (1984, 1987a,b) have developed the MOSYS interactive support system for planning FASs which incorporates queueing networks and simulation. The approach involves trial-and-error by the designer who works interactively with the support system. Liu and Sanders (1988) present a hybrid algorithm which first uses a queueing network model to set the number of pallets in the system and then uses a simulation and a gradient-based search technique to set the number of buffers for optimal system throughput. Bulgak and Sanders (1989) extend the work of Liu and Sanders to consider FASs with repair loops.

Researchers have also used integer programming. Whitney and Suri (1985) provide computer–aided decision tools which select machine types and part types from a large number of candidates. 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, machine capacity, and tool magazine capacity. Because of the size of the problem and some nonlinear interactions, they develop a solution procedure which uses two heuristic algorithms sequentially.

Graves and Redfield (1988) present an optimization procedure to assign tasks to workstations and select a single assembly machine for each workstation in a multiproduct assembly system. They assume that only one product is being assembled at any given time. Their objective is to minimize the fixed capital costs for the assembly equipment and tools, and the variable operating costs. There is a cycle time constraint which limits the number of tasks that can be assigned to a workstation. The solution is obtained through implicit enumeration.

These integer programming models do not take into account the fact that contention for resources causes queues. With a finite number of jobs in the system, queue buildups at a subset of stations lead to starvation at other stations. This, in turn, makes it impossible to utilize 100% of the stated capacity of the equipment. On the other hand, it is difficult to determine the achievable capacity utilization without knowing the physical configuration of the system and the assignment of tasks to machines. We resolve this difficulty by combining integer programming–based approaches with non-linear programming and queueing network models.

A paper by Lee and Johnson (1991) uses queueing network and integer programming models to address a problem similar to the one under consideration here. However, their procedure is limited in that it (a) seeks only balanced designs (the same number of parallel machines at all

stations), (b) does not *simultaneously* determine the number of machines and pallets/material handling resources, (c) assumes that each task requires the same staging space to store the feeder or components, and (d) does not provide a basis to evaluate the resulting design. This paper overcomes these limitations.

3. Problem Statement and Formulation

As mentioned earlier, we assume that the FAS is a flow system consisting of a series of assembly stations, each of which may have multiple identical machines in parallel. In the context of an automated system, there may also be a load/unload (L/UL) station, and the machines may be connected by conveyors or automated guided vehicle (AGV) paths. A part (or batch of parts) is loaded into the system either at the L/UL station, or the first workstation, as applicable. It may be carried on a pallet or mounted onto a fixture.

We make a few assumptions to facilitate the modeling process. We chose to model the system as a PF CQN. (Other models could have been used, but the PF CQN is well-studied, and related results proved to be useful in our analysis.) The PF CQN has several assumptions in our context: (1) Processing times are exponentially distributed if the first-come-first-served (FCFS) discipline is used. (2) The parts can be aggregated into a single, representative, "weighted average" part. (3) A constant number of parts circulate in the system. (4) There is (effectively) unlimited buffer capacity at each station. (5) Material handling is not the bottleneck, or it can be represented as other "machine centers" in our system. Assumptions (3)-(5) are not unrealistic for well-designed systems. We assume a FCFS service discipline, which is common in practice, and thus must assume exponential processing times. Many studies (see, for example, Suri 1983) indicate that estimates of performance measures from PF CQN models are reasonably accurate even when the exponential assumption is violated. Assumption (2) is discussed in more detail below.

We assume that the collection of products to be produced on the system can be aggregated into a single "representative" aggregate product type. Precedence diagrams of all the product types are merged and represented by a super-precedence diagram for the aggregate product type. We assume that this super precedence diagram is *acyclic*. This implies that if two or more product types require the same tasks, say tasks 1 and 2, then task 1 precedes task 2 in all these product types. This assumption makes sense where the product types have similar assembly patterns, such as in automobile assembly lines and in the manufacture of printed circuit boards. Since the task processing times may differ among the product types, the processing times for the aggregate product type are specified as weighted average processing times of the individual product types.

On the other hand if, say, task 1 precedes task 2 in one product type while task 2 precedes task 1 in the other product type, we can still ensure that the super-precedence diagram is acyclic if we

simply reassign a unique task number to one of the tasks for either product type. As a consequence of this renumbering, however, a common task may now be performed at one station for one product and a different station for the other. Figure 2 presents a simple example of how the aggregate product is constructed, along with its associated graph.

Example. Suppose an FAS produces two product types simultaneously, each of which has the same throughput requirement per period, say, 100. Then, the throughput requirement for the aggregate product type is 200 per period. Product type 1 requires five tasks (1,2,3,4,6) to be performed in the listed order, and product type 2 requires tasks (1,2,5,6) in the listed order. Let the processing time for task j be j time units for j=1 to 6. The aggregate graph and weighted average processing times are shown in Figure 2.

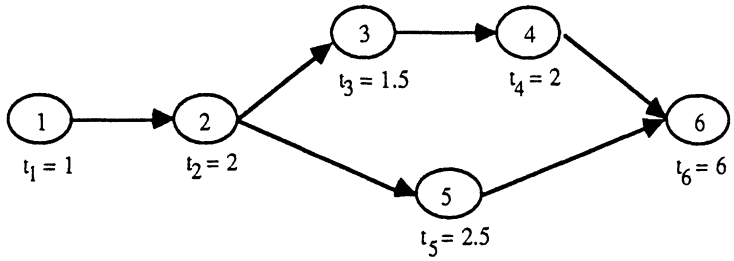

Figure 2. The Aggregate Product

We assume that the staging space needed for each component or set of components is a known integer multiple of a basic unit of space. It is always possible to define a sufficiently small unit of space so that this representation is exact, although in practice, such accuracy may not be important.

Our procedure addresses the following design decisions: (i) the assignment of tasks to stations; (ii) the number of assembly machines per station; and (iii) the number of jobs circulating in the system. In some systems, such as stop–and–go AGV systems, there may be one material handling vehicle supporting each pallet of jobs. We assume that the cost of any such material handling equipment is combined with the cost of a pallet. The objective is to minimize the sum of WIP inventory cost, and operating, maintenance and amortized purchase costs for material handling equipment and assembly machines. The solution must satisfy the following constraints.

(a) Steady-state throughput requirements for each product type must be satisfied.
(b) The staging capacity must be satisfied.
(c) A part may visit each station at most once, and in the specified sequence.
(d) Task assignments must satisfy precedence constraints.

We now turn to a formulation of the problem. In the following formulation, we assume that the total number of stations, M, is given. In Section 4, we explain how to determine the minimum feasible value of M. Let:

t_j = mean processing time for (aggregate) task j,
TW = sum of the workloads at all processing stations = $\sum_j t_j$,

$\bar{W}$ = vector indicating the mean workload per part at each station,
R = staging capacity (multiple of the basic space increment),
N = number of pallets circulating in the system,

$\bar{S}$ = vector indicating the number of machines at each station,
K = number of machines = $\sum_i S_i$

$TH(N,\bar{S},\bar{W})$ = throughput of the system, given N, $\bar{S}$ and $\bar{W}$,
d = aggregate throughput requirement (units per unit time),
z(N,K) = a cost function that increases with N and K,

$$X_{ij} = \begin{cases} 1, & \text{if task j is assigned to the } i^{th} \text{ station,} \\ 0, & \text{otherwise.} \end{cases}$$

The problem can be formulated as:

P: Minimize z(N,K)

subject to: $K = \sum_i S_i$,

$TH(N,\bar{S},\bar{W}) \geq d$,

$$\sum_{j=1}^{n} t_j X_{ij} = W_i , \qquad i=1,...,M, \qquad (1)$$

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

$$X_{ij} = 0 \text{ or } 1, \qquad i=1,...,M, \quad j=1,...,n, \qquad (3)$$

$$\sum_{i=1}^{M} X_{ij} = 1, \qquad j=1,...,n, \qquad (4)$$

$$\sum_{i=1}^{M} i X_{ij} \leq \sum_{i=1}^{M} i X_{ik}, \quad \text{when task j is an immediate predecessor of task k.} \qquad (5)$$

Equation (1) defines W_i, the workload at station i, as the sum of the processing times of the tasks assigned to station i. Constraints (2) ensure that staging capacity constraints are satisfied. Constraints (3) and (4) ensure that each task to be assigned to exactly one station. Constraints (5) model the precedence relations among tasks and ensures that a part does not revisit any station.

4. Approach to the Problem

This problem is very difficult to solve optimally for several reasons: the throughput of the CQN model is very complex and nonlinear, the total cost function may be nonlinear, and all decisions are integer valued. To overcome these difficulties, the proposed framework decomposes the decisions in two sets: (1) the number of machines at each station, and the number of pallets, and (2) the assignment of tasks to stations. Note that if the optimal workload allocation were known, then the problem could be optimally decomposed in this manner.

The basic strategy underlying our procedure is to identify a set of "good" target workloads. For each, we first assign tasks to stations so as to match the targets as closely as possible. For this assignment, we then find the configuration that provides the desired throughput at minimum cost. Since the number of machines and pallets must be integers, a range of workload allocations lead to the same optimal configuration. Moreover, for a given number of machines, the machines may be partitioned into stations in various ways, and each partition will be optimal for a range of workload allocations. Consequently, many different workload allocations (and thus also, many different task assignments) will lead to (one of) the least cost configuration(s). This is our reason for attempting to identify a set of "good" target workloads. If we do not decompose the problem via the target workloads, to find the optimal solution, it would be necessary to consider each feasible assignment of tasks to stations, determine the minimum cost configuration satisfying the throughput for each assignment, and then choose a configuration with the lowest cost. This is clearly impractical for all but very small problems.

We also consider it important to be able to evaluate solutions by an absolute standard. To do so, as part of our procedure, we find a lower bound on the cost of the optimal solution by solving a relaxed problem where the workloads are allowed to be continuous, and determine the minimum cost configuration which meets the throughput requirement.

The procedure uses five modules plus an initialization module. We briefly describe each module below. We should note that the decomposition approach does not rely upon our specific solution procedure for each module, although we are not aware of other solution procedures for the *specific* subproblems in Modules 1, 2, and 4. At the end of this section, we explain the overall procedure and the role of each module within it. The procedure solves the problem for a given

number of stations, and can be repeated to consider different values. However, it is necessary to determine the minimum feasible value, M_0, for the number of stations. The value of M_0 is obtained from Module 0. Note that we try to obtain a solution with the smallest number of stations for at least two reasons: it reduces material handling, and results in more parallel machines assigned to each station, thereby enhancing the "pooling effect" (see, for example, Stecke and Solberg 1985).

Module 0: Finding the Minimum Feasible Number of Stations

This problem is equivalent to the well-known assembly line balancing problem with the objective of minimizing the number of stations for a fixed cycle time. For our problem, the staging capacity corresponds to the cycle time, and the staging space requirements correspond to the processing times. We use Johnson's algorithm (Johnson 1988) to solve the line balancing problem.

One by-product of solving this problem is the identification of the subset of stations to which a task can be assigned while satisfying both precedence and staging capacity constraints. The direct results pertain to situations with the minimum feasible number of stations, but it is straightforward to obtain similar results for any specified number of stations. Essentially, the earliest station to which a task can be assigned is found by loading all of its predecessors as compactly as possible into the early stations without violating precedence and staging capacity constraints. Similarly, the latest station is found by loading all of the task's successors as compactly as possible into the final stations. Form this, it is straightforward to determine the subset of tasks that are candidates for assignment to any station. This information is used in Module 1.

Module 1: The Workload Bound Problem

This problem involves finding (tight) upper and lower bounds on the workload at each station subject to the staging capacity and precedence constraints. These upper and lower bounds serve as surrogates for the staging capacity and precedence constraints when we solve a continuous workload allocation problem (Module 2), where these constraints cannot be considered explicitly. In preliminary investigations, we found that if these bounds were not included, the resulting target workloads might be far from achievable, which in turn would lead to long computation times and very poor solutions.

It is important to point out that this problem is solved without specifying the number of machines at each station. Thus, the resulting workload bounds are not a function of the physical configuration of the system (except the number of stations). Rather, they are induced solely because of staging capacity and task precedence considerations. Let

U_i = upper bound on the workload at station i, and

L_i = lower bound on the workload at station i.

Then the problem of finding the upper bound at station i can be formulated as:

P1: Maximize $U_i = \sum_{j=1}^{n} t_j X_{ij}$ (6)

subject to $\sum_{m=1}^{M} X_{mj} = 1, \quad j = 1,...,n,$ (7)

$\sum_{m=1}^{M} m X_{mj} \leq \sum_{m=1}^{M} m X_{mk},$ if task j is an immediate predecessor of task k; $j = 1,...,n; \quad k = 1,...n,$ (8)

$\sum_{j=1}^{n} a_j X_{mj} \leq R, \quad m = 1,...,M,$ (9)

$X_{mj} = 0 \text{ or } 1, \quad m=1,...,M; \quad j=1,...,n.$ (10)

The constraints ensure that all tasks are assigned, precedence relations are satisfied, and that staging capacity constraints are satisfied. To determine lower bounds, the objective becomes one of minimization rather than maximization. These problems are difficult to solve optimally, but it is possible to solve relaxations of these problems to obtain looser bounds. For the problem of determining upper bounds, one simple relaxation involves ignoring the assignment constraints (7) and precedence constraints (8), and considering only m = i in (9). Instead of (7) and (8) we consider only those tasks that are candidates for assignment to station i, which can be obtained from Module 0. This yields small knapsack problems that can be solved quickly. To obtain lower bounds, equation (9) is replaced by $\sum_{j=1}^{n} a_j X_{ij} \geq \sum_{j=1}^{n} a_j - (M-1)R$, when $M = M_0$. The right hand side of this equation represents the staging space that would need to be filled at station i if the other M-1 stations had their capacities filled. Consequently, this represents the minimum staging space that must be filled at station i. When $M > M_0$, all tasks will fit into fewer than M stations, so we need to ensure that each station is assigned at least one task. To do so, equation (9) is replaced by $\sum_{j=1}^{n} a_j X_{ij} \geq A_i$, where A_i is the smallest a_j among the tasks assignable to station i. Preliminary studies indicated that the upper bounds are more important than the lower bounds, so we can solve the minimization problem using a linear programming relaxation of the problem, which, in turn, can be solved by inspection (greedy is optimal).

Module 2: The Optimal Configuration and Workload Allocation Problem

The problem here is to determine the number of pallets, the number of machines at each station, and the (continuous) allocation of total workload among the stations. The objective is to minimize total cost subject to two constraints: (1) the allocated workloads must lie between the lower and upper bounds obtained from Module 1, and (2) the aggregate throughput requirement must be satisfied. The FAS is modeled as a PF CQN. The number of servers at station i, S_i, i = 1,...,M, and the number of pallets, N, define a configuration. Since a continuous allocation of workloads is permitted, the resulting cost provides a lower bound on the total cost of problem **P**. A mathematical formulation of the problem is

P2: Minimize $z(N,K)$

subject to:

$$K = \sum_{i=1}^{M} S_i , \tag{11}$$

$$TH(N,\bar{S},\bar{W}) \geq d, \tag{12}$$

$$\sum_{i=1}^{M} W_i = TW, \tag{13}$$

$$L_i \leq W_i \leq U_i, \quad i=1,...,M, \tag{14}$$

where our decision variables are N, $\bar{S}$, and $\bar{W}$.

To solve **P2**, we obtain lower bounds, N^{LB} and K^{LB}, on N and K from the asymptotic bound analysis of Muntz and Wong (1974). From this analysis, N^{LB} is $\lceil d\cdot(TW+W_0) \rceil$ where $\lceil x \rceil$ is the smallest integer greater than or equal to x. This simply says that the number of pallets in the system should be at least as large as the demand (arrival) rate multiplied by the minimum sojourn time in the system. K^{LB} is $\max (\lceil d\cdot TW \rceil, \sum_{i=1}^{M} S_i^{LB})$ where S_i^{LB} is the lower bound on the number of servers at station i and is given as $S_i^{LB} = [d\cdot L_i]$ with [y] denoting the smallest integer strictly greater than y. This essentially says that the total number of servers must be large enough so that the total system utilization and the utilization levels of the individual stations are less than one.

We developed an implicit enumeration procedure which allows us to efficiently consider all combinations of $N \geq N^{LB}$ and $K \geq K^{LB}$. During the implicit enumeration procedure, many alternatives are eliminated by considering fundamental properties of the throughput function. For example, we use the fact that throughput is monotonically non–decreasing in N and S_i. Other results, such as the concavity and other second-order properties of the throughput function with respect to N, and bounds on throughput for a given N and $\bar{S}$ (Shanthikumar and Yao 1988b,c) also

may help to eliminate a few alternatives, but the search for the best value of N for a given $\bar{S}$ turns out to be fast and easy because of the form of the throughput function for PF networks. Thus, we found that the monotonicity property was sufficient to obtain solutions (for N) quickly. For further details, see Lee et al. (1991a) where several other fathoming rules are presented.

For each candidate (N,K) pair, we must determine whether there is a feasible $\bar{S}$ and $\bar{W}$ for **P2**, i.e., one satisfying the throughput constraint. We generate all feasible partitions of the K machines into M stations, and implicitly enumerate the resulting server vectors, $\bar{S}$, solving the following problem for a given ($\bar{S}$,N):

Module 3: The Workload Allocation Problem

P2a: Maximize $TH(\bar{W})$

$$\text{subject to: } \sum_{i=1}^{M} W_i = TW, \tag{15}$$

$$0 \le L_i \le W_i \le U_i, \qquad i=1,..,M. \tag{16}$$

We derived some properties of the throughput function for a product-form CQN and used these properties to characterize the optimal workload allocation. With this characterization, we were able to develop very efficient nonlinear programming procedures to solve this problem (see Lee et al. 1991b).

Thus, the overall procedure for Module 2 involves implicit enumeration of the (N,K) pairs at the higher level, and implicit enumeration and evaluation of the server vectors (using Module 3) at the lower level. Details appear in Lee et al. (1991a). We note that there may be more than one solution to this problem satisfying the throughput requirement at the minimum cost.

Module 4: Assembly Loading Problem

The assembly loading problem involves assigning tasks to stations such that the actual workloads are as close as possible to given target workloads, subject to the staging capacity constraint and precedence requirements. A formulation of the problem follows.

P3: Minimize $D(\bar{W},\bar{W}^*)$

subject to: (1), (2), (3),(4), and (5),

where $D(\bar{W},\bar{W}^*)$ is a function which measures the closeness of the actual workload vector $\bar{W}$ to the target workload vector $\bar{W}^*$.

Kim and Yano (1989) test several functions for $D(\bar{W},\bar{W}^*)$ to determine which measure gives the best throughput for a PF CQN model. They suggest $D(\bar{W},\bar{W}^*) = \max_i (W_i - W_i^*) / W_i^*$ when the S_is are not equal. When the S_is are equal, they suggest $D(\bar{W},\bar{W}^*) = \max_i (W_i - TW/M)$, where TW/M represents an equal workload allocation to each station. This balanced workload is shown to maximize the throughput function for a CQN where all S_is are the same (e.g. Shanthikumar and Stecke 1986, Yao and Kim 1987). We use these results to form our objective function. Thus, the problem becomes:

P3a: Minimize δ

subject to: constraints (2) through (5) and

$$\sum_{j=1}^{n} q_{ij} X_{ij} \leq \delta \quad i = 1,\cdots,M, \tag{12}$$

where δ is the maximum among the scaled workloads (the scaled workload at station i is defined as the actual workload at station i divided by the corresponding target workload); q_{ij} is equal to t_j / (TW/M) if all S_is are the same, and is equal to t_j / W_i^* otherwise.

Problem **P3** without constraints (5) is a special case of the problem for which branch–and–bound algorithms have been developed by Berrada and Stecke (1986) and Kim and Yano (1989). However, they do not consider precedence constraints, and their branching rules make it difficult to check for prospective precedence violations resulting from a partial assignment. Instead, we developed a heuristic branch-and-bound method by generalizing the assembly line balancing algorithm of Johnson (1988). If the target workloads are the same for all workstations, we can use the algorithm presented in Lee and Johnson (1991) with minor modifications to account for unequal staging space requirements. For the case of unequal target workloads, we further modify this procedure to accommodate processing time differences across stations, with associated changes in the fathoming rules. For further details see Lee (1989).

Module 4 results in a discrete allocation of tasks to the stations. Note, however, that the throughput resulting from this allocation may not satisfy the throughput requirement. Module 5 finds the optimal configuration (N and K) for a given task allocation (namely, the task allocation obtained from Module 4).

Module 5: Optimal Configuration Problem

This problem is to find the number of pallets and the number of machines at each station given a set of task assignments, so as to minimize total cost subject to meeting the throughput

requirement. Here, the system is modeled using a CQN representation. This problem has been studied by Vinod and Solberg (1985) and Dallery and Frein (1986) and their algorithms can be used. We used a modification of the Dallery and Frein procedure in which all (N,K) pairs with a cost less than the theoretical minimum cost (from Module 3) are eliminated, thereby speeding up the search process.

On completion of Module 5, we have a feasible solution to problem **P**, and an upper bound on the total cost. The procedure now begins a search on the total cost. Figure 3 presents a flowchart of the procedure. In this flowchart, a superscript "c" denotes the solution for the current set of parameters.

Overall Procedure

The overall procedure proceeds as follows. Module 0 (minimum number of stations), Module 1 (workload bound problem) and Module 2 (optimal configuration and workload allocation problem) are solved in sequence. At the end of these steps, we have one or more (minimum cost) configurations that satisfy the throughput requirement with the total workload allocated in a continuous fashion. The solution to Module 2 provides a lower bound on the cost of the optimal solution. The minimum cost configurations are sorted in decreasing order of throughput. The rationale for sorting the configurations is that the workloads resulting from the assignment of (discrete) tasks will deviate from the optimal continuous workloads. Consequently, the throughput with the discrete task assignments will be lower than the throughput with continuous allocations. Thus, configurations giving a higher throughput with the optimal continuous workloads are more likely to be feasible after the assignment of (discrete) tasks.

The procedure now begins a search on the total cost. The lower bound is obtained as described above, and this serves as the initial trial cost. Initially, the upper bound is set to an arbitrarily large value, but an improved upper bound is generated at the completion of module 5. If the difference between the upper and lower bounds exceeds a prespecified tolerance, then we generate all possible (N,K) pairs with $N \geq N^{LB}$ and $K \geq K^{LB}$ such that z(N,K) lies between these bounds, and sort them in non-decreasing order of cost. This provides the set of trial costs to be considered. We successively consider the trial costs in increasing order of cost. For the current trial cost, the following steps are performed:

(i) Generate all configurations with the given trial cost.

(ii) Solve the continuous workload allocation problem for each configuration, and rank configurations meeting the throughput requirement in decreasing order of throughput.

(iii) For each configuration on the list generated in Step (ii), lexicographically order the server vector such that $S_1 \leq \ldots \leq S_M$. Eliminate any duplicates.

(iv) For each candidate configuration in the resulting list, generate two target workload vector permutations and construct a feasible solution for each permutation.

Step (i) is self-explanatory. Since we are considering only machines and pallets, it is easy to generate all such configurations. Step (ii) executes Module 3 for each configuration to determine whether the maximum throughput achievable with a continuous workload allocation satisfies the throughput requirement. Such configurations are ranked in decreasing order of throughput.

There could be possibly many configurations on the list produced in Step (ii). Note that the throughput function of a PF CQN is permutation-invariant. For example, if $\bar{S}_1 = (1,2,3)$, $\bar{W}_1 = (10,30,50)$, and $\bar{S}_2 = (1,3,2)$, $\bar{W}_2 = (10,50,30)$, then $TH(\bar{S}_1,\bar{W}_1) = TH(\bar{S}_2,\bar{W}_2)$. Since Step (iv) solves the unconstrained workload allocation problem, this allows us to consider only one lexicographic ordering of each partition of N servers. Consequently, we significantly reduce the number of unconstrained workload allocation problems and associated assembly loading problems (which are CPU intensive) that need to be solved in Step (iv).

Step (iv) is now described in more detail. In the first part of Step (iv) we solve the unconstrained workload allocation problem, namely, **P2a** without constraints (16), for each configuration in the list obtained from Step (iii). The resulting workload vector, $\bar{W}^*$, is the target workload vector for that configuration. Our reason for using the unconstrained solution here is that Module 4, which uses these workloads as targets, explicitly accounts for staging capacity and precedence constraints. Since the throughput function of a PF CQN is unimodal and well-behaved with respect to the workload vector (Stecke 1986, Lee et al. 1991a), the closer the workload vector obtained from Module 4 is to $\bar{W}^*$, the higher is the resulting throughput.

Although the throughput function of a PF CQN is permutation-invariant we found, however, that differently permuted target workloads can have a considerable impact on the performance of Module 4. We investigated various sequencing rules and found that alternating high and low target workloads worked well. One reason for this appears to be the opportunity to assign tasks with larger processing times, which tend to be the most difficult to "fit" in the context of the loading problem, to various stations spread throughout the system. We found this to be preferable to having stations with high workload targets concentrated in one portion of the flow system, which makes it difficult to achieve actual workloads close to the targets while simultaneously satisfying precedence constraints. We developed a procedure to find two sequences of alternating target workloads (which we call target workload vector permutations) that satisfy the bounds from

Module 1 (see Lee 1989 for details). In practice, it would not be difficult to perform this step by inspection. We have simply automated the process. Of course, if it is known that the high workload tasks are concentrated in a particular portion of the assembly sequence, target workload vectors can be chosen accordingly.

The second part of Step (iv) considers each target workload vector permutation in turn and solves an assembly loading problem (Module 4). Since the resulting workloads will deviate from their respective targets, we solve an optimal configuration problem (Module 5) to determine the best feasible configuration for the resulting workloads. (The configuration that gave rise to the target workload vector may now be infeasible if the resulting workloads do not match the targets well. Also, the best configuration for the resulting workloads may differ from the original configuration.)

Every time Module 5 is solved, a feasible solution is generated. If its cost is equal to the lower bound, the solution is optimal and the procedure terminates. The procedure also terminates if the cost is less than or equal to the current trial cost, or if the cost is within the prespecified tolerance from the current trial cost. If the cost exceeds the trial cost but is less than the current upper bound, the upper bound is updated, and we continue with Step (iii).

If all configurations at a given trial cost have been evaluated, then the next larger trial cost is chosen if there is one which is less than the current upper bound. Otherwise the procedure terminates.

This procedure is not optimal since we do not consider every possible target workload vector permutation for a given $\bar{S}$. Moreover, even if we could consider all target workload vector permutations, we still cannot guarantee optimal task assignments since we are using a surrogate objective for the assembly loading problem, namely, minimizing $D(\bar{W},\bar{W}^*)$. In preliminary tests, we found that little was gained by considering all target workload vector permutations. We therefore decided to use only two candidate permutations for each $\bar{S}$.

Example:

We present a small example to illustrate the procedure. The problem has three stations and 14 tasks, with processing times and precedence constraints as shown in Figure 4. For simplicity, we assume in this example that $a_j = 1$ for all j. The staging capacity is set at 5. The throughput requirement is assumed to be 650 units per period. The processing capacity is assumed to be 10,000 time units per machine per period. The total transfer time for one part is 20 time units. The cost function $z(N,K) = 12{,}000\ N + 20{,}000\ K$. We used a tolerance value of 4,000 which is the greatest common divisor of 12,000 and 20,000.

The upper and lower bounds on the workloads were obtained from Module 1 as

$$\bar{L} = (18, 10, 10), \quad \bar{U} = (31, 34, 31).$$

The optimal configuration and workload allocation problem (Module 2) gave an initial lower bound on the total cost of 236,000. The corresponding solution was:

$$N^* = 8, \quad \bar{S}^* = (2, 3, 2), \quad \bar{W} = (20.5, 34, 20.5).$$

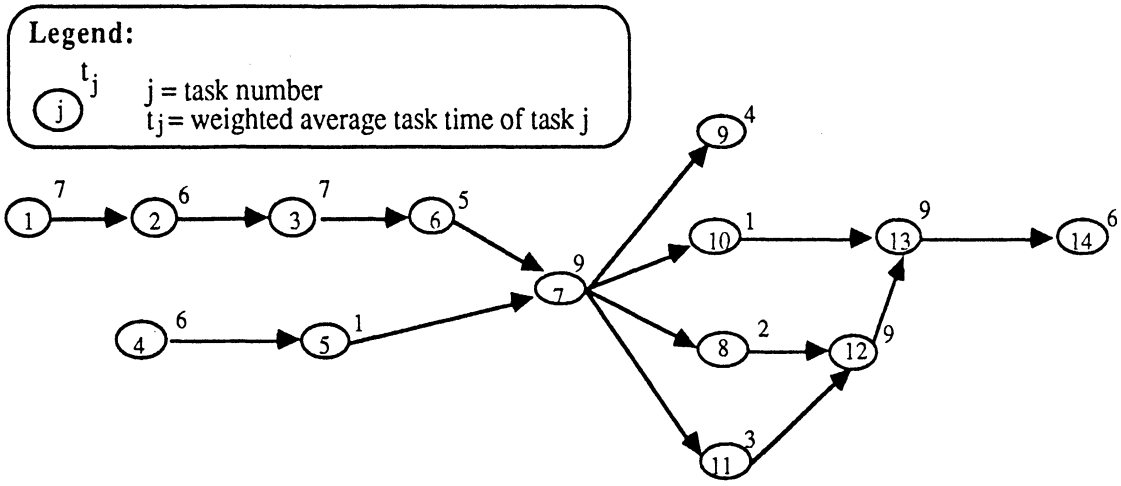

Figure 4: Precedence diagram for example problem

Note that the only (N,K) pair with a total cost of 236,000 is N=8, and K=7. We generated all possible $\bar{S}$s for K=7, and solved the workload allocation problem with the workload bounds (Module 3) for each ($\bar{S}$,N), with N = 8. The only $\bar{S}$ which satisfied the throughput requirement was $\bar{S}$ = (2,3,2). For this ($\bar{S}$,N), we solved the workload allocation problem without workload bounds, to get a target workload vector $\bar{W}^*$ = (19.7, 35.6, 19.7). The resulting throughput was 655.1.

To solve the assembly loading problem (Module 4), we generated two target workload vector permutations, namely $\bar{W}_1^*$ = (35.6, 19.7, 19.7), and $\bar{W}_2^*$ = (19.7, 35.6, 19.7). The solution from the corresponding assembly loading problem gave

i) $\bar{W}_1$ = (31, 24, 20), with the following task assignments: tasks 1, 2, 3, 4 and 6 to station 1; tasks 5, 7, 9, 10 and 11 to station 2; tasks 8, 12, 13 and 14 to station 3.

ii) $\bar{W}_2$ = (20, 28, 27), with the following task assignments: tasks 1, 2, 4 and 5 to station 1; tasks 3, 6, 7, 9 and 11 to station 2; tasks 8, 10, 12, 13 and 14 to station 3.

Using $\bar{W}_1$ we solved the optimal configuration problem (Module 5) to obtain $N(\bar{W})^*$ = 9, and $\bar{S}(\bar{W})^*$ = (3, 2, 2). This provides a new upper bound on total cost of 248,000. The throughput

for this configuration was 676.2. The other workload vector, $\bar{W}_2$, gave a solution with the same cost, but its throughput was only 651.4.

We now have a lower bound of 236,000 and an upper bound of 248,000. There are only two potential trial costs between these bounds, namely 240,000 and 244,000 because the greatest common divisor is 4,000. There was no combination of N and K with a cost of 240,000 satisfying $N \geq N^{LB}$ and $K \geq K^{LB}$.

At a trial cost of 244,000, there was only one configuration which met the throughput requirement based on the continuous workload allocation problem (with workload bounds): N = 7, $\bar{S}$ = (3, 3, 2), $\bar{W}^*$ = (29.9, 29.9, 15.2). We found this workload vector was also optimal for the unconstrained workload allocation problem. The solution of the assembly loading problem with a target workload vector of (29.9, 15.2, 29.9) gave $\bar{W}$ = (31, 18, 26) and a throughput of 653.1. The corresponding task assignment were: tasks 1, 2, 3, 4, and 6 to station 1; tasks 5, 7, 9, 10, and 11 to station 2; and tasks 8, 12, 13, and 14 to station 3. The procedure terminated since a feasible solution was found at the current trial cost.

Since this was a small problem, we were able to generate all feasible task assignments and solve the optimum configuration problem for each resulting workload vector. The optimal configuration also had a cost of 244,000.

5. Experimental Results

We ran the procedure outlined above on many test problems. In this section, we report on some of these experiments. We first consider the case where the staging space requirement is the same for all tasks.

5.1 Equal Staging Space for All Tasks

We studied the effects of three factors on the performance of the procedure. These factors are lumpiness of task times (i.e., variation among task times), density of the precedence diagram, and the staging capacity. We specified two levels of lumpiness by controlling the range of task times. These task times were randomly generated from a discrete uniform distribution using a range of 1 to 9 for the lumpy task times and a value of 5 for the non–lumpy task times. For the former case, three task time scenarios were randomly generated for each of the combinations of other factors.

We used three levels of density: δ = (0.05, 0.25, 0.50) where density δ is the ratio of the number of precedence arcs present in a graph to the maximum number of arcs that the graph can

have for a given number of tasks (n), i.e., $\binom{n}{2}$. We randomly generated arcs such that each arc is equally likely. The graphs were generated in such a way that the $d = 0.25$ graph includes all the arcs (and their task times) in the $d = 0.05$ graph. A similar relationship holds for the $d = 0.5$ and $d = 0.25$ graphs. The intent here was to isolate the effect of added precedence arcs. We used two levels of staging capacity: R=15 or 30.

We fixed the number of tasks at 100 and the throughput requirement per period (for the aggregate product) at 200. The processing capacity per period and average transfer time between two stations were set to 10,000 and 5, respectively. For the case of R = 15, the number of stations was set at 7, and for the case of R = 30, the number of stations was set at 4. In both cases, the minimum number of stations was used. The cost function we used was z(N,K) = 12,000 N + 20,000 K. This might be reflective of the annual operating and amortized purchase costs of, say, N stop-and-go AGVs with pallets, and K assembly robots. The tolerance value was 12,000 (cost of one stop-and-go AGV with pallet). Statistics for each problem are summarized in Tables 1 through 4. (The assignment of tasks was also determined, but not reported here for the sake of brevity.) Note that for cases with identical task times, the results are the same for all three replications. We can show that whenever t_j is the same for all j, the actual workload vectors obtained from the assembly loading problem are identical, regardless of the precedence diagram.

The procedure worked well for both levels of lumpiness of task times. It found optimum solutions to 11 of the 18 problems under lumpy task times, as shown in Tables 1 through 3 and for 3 of the 6 problems under identical task times, as shown in Table 4. Even for problems for which we could not verify that the solution is optimum, the maximum deviation of the cost of the solution from the corresponding lower bound on cost did not exceed 2.4% and 2.3% under the lumpy and identical task times, respectively.

In these examples, the solutions were insensitive to the density of the precedence diagram. Within each processing time scenario, the solutions were identical for all densities at a given R with only one exception. This exception occurred for the case of $d = 0.50$ and R=15 in Table 1. For this instance, the feasible solution costs only \$4,000 more than the solutions for R=15 with $d = 0.05$ and 0.25 in Table 1. One possible reason for this insensitivity is that even with R=15, the staging capacity is large enough, i.e., up to 15 tasks can be assigned to a station. Hence additional precedence constraints do not have an adverse effect on the assignment of tasks to machines. The density of the precedence diagram may have effect on the solution when the staging capacity constraints are tighter.

The staging capacity had a stronger effect on the solutions. The lower and upper bounds on cost under R=30 are always smaller than their counterparts under R=15. This can be attributed to

the efficiency of pooled resources for the smaller number of stations (M=4 when R=30 vs. M=7 when R=15).

Overall, the procedure performed well. It found the optimum solutions to 14 of the 24 problems listed in the tables. Even for problems in which the solution could not be verified as optimal, the maximum deviation of the costs from the respective lower bound on cost did not exceed 2.4%. In addition, for a majority of the problems, no more than 20 seconds of CPU time was required to find the solutions on an IBM 3090–600. These results suggest that the procedure can be applied in problems of realistic size with very good results.

5.2 Experiments with Unequal Staging Space Requirements

We tested eight sample problems by varying three factors; the values used for these factors are shown in parentheses: throughput requirement (150 and 300), density (.1 and .5), and staging capacity (15 and 30). The number of tasks was set to 50. Task times ranging from 1 to 9, and staging space requirements ranging from 1 to 3 were randomly generated from discrete uniform distributions. We assume that conveyors are used to move pallets between stations and the transfer times between stations were set to 10 time units. The cost function used was z(N,K) = 1,000 N + 20,000 K and the tolerance value was 3,000 (cost of three pallets/fixtures). The other parameter values remained the same as before.

The procedure also performed well for experiments with unequal staging space requirements. Experimental results for the eight problems are summarized in Tables 5 and 6. The minimum number of stations, M_0, was found to be 4 and 8 for R=15 and R=30, respectively. When the throughput requirement increased from 150 to 300, more machines and pallets were required, which resulted in higher cost and required more CPU time to evaluate a larger number of configurations. With regard to the effects of the other two factors, we observed a similar pattern as in the experiments we conducted with equal staging space requirements in section 5.1. The procedure found optimum solutions for 3 of the 8 problems. Even for those solutions not verified as optimal, the maximum deviation of the costs from the lower bound did not exceed 4.4%.

6. Extensions

In this section, we briefly discuss how the procedure can be used in the design of cellular manufacturing systems, and also indicate how the procedure can be extended to handle systems where a part may visit a station more than once.

6.1 Cell Design

The proposed methodology can be used together with group technology (GT) to help design a more general class of manufacturing systems. GT groups a large number of products into a small

number of product families using similarities of products and processes. The proposed methodology can be applied to each product family to design an efficient flow system (cell). This is a topic for future research.

6.2 Multiple Visits

When material handling is not an important issue, the procedure can be applied to FASs where a job may visit a station more than once. In fact, it is easier to solve the workload bound and assembly loading problems since precedence constraints are not needed. The solution method for the relaxed problem in Module 2 remains unchanged except that we now have to treat the material handling system as another constrained resource (i.e., another station in the system) and estimate the material handling workload. This is because the actual material handling workload depends on the number of visits which, in turn, is determined by the task assignments. When the relaxed problem is solved with the minimum material handling workload (i.e., as in a flow system without revisits), the lower bound on total cost may be loose, depending on the magnitude of transfer times and the number of revisits.

7. Summary and Conclusions

In this paper, we presented a procedure for solving the capacity planning and machine configuration problem for Flexible Assembly Systems with a flow structure and precedence constraints among operations. Such systems are common, but most related work ignores both of these complicating features. In addition, this procedure differs from earlier work in that it simultaneously determines: (i) the assignment of tasks to stations; (ii) the number of (parallel) assembly machines per station; and (iii) the number of pallets and/or fixtures, whereas most other procedures only consider a subset of these decisions. To our knowledge, this is the first time that such a comprehensive approach has been developed for the capacity planning and machine configuration problem. Since the procedure also produces a lower bound on cost for our version of the problem, it is possible to judge the absolute quality of the solution. The procedure was shown to perform very well in a number of problems of realistic size.

The problem is made tractable partially by certain assumptions about precedence constraints and queueing phenomena, but principally by our heuristic decomposition approach. To relax some of these assumptions, it is only necessary to modify certain subproblems, while the remainder of the framework remains intact. Further research is needed to relax these assumptions.

In many FASs, the machines themselves are quite flexible and the staging capacity is more constraining. In such systems, the staging capacity might be viewed as a key measure of the

flexibility of the system. Our procedure can be used to obtain a quantitative assessment of the value of this flexibility.

In this paper, we address a steady–state version of the problem where the product mix is given. In reality, the product mix may change over time and new products may be introduced. Since our procedure is relatively fast, it would be possible to perform sensitivity analysis with respect to the product mix. Several different viable configurations could be generated, and each could be evaluated over a longer horizon. Generating good configurations for a non–stationary environment is not easy to do. However, since the machines are flexible, the critical decisions relate to the number of machines of each type; the station configuration may be less important. For example, it may be possible to modify the partitioning of machines into stations to accommodate product mix changes if the material handling system is set up to accommodate a variety of routings.

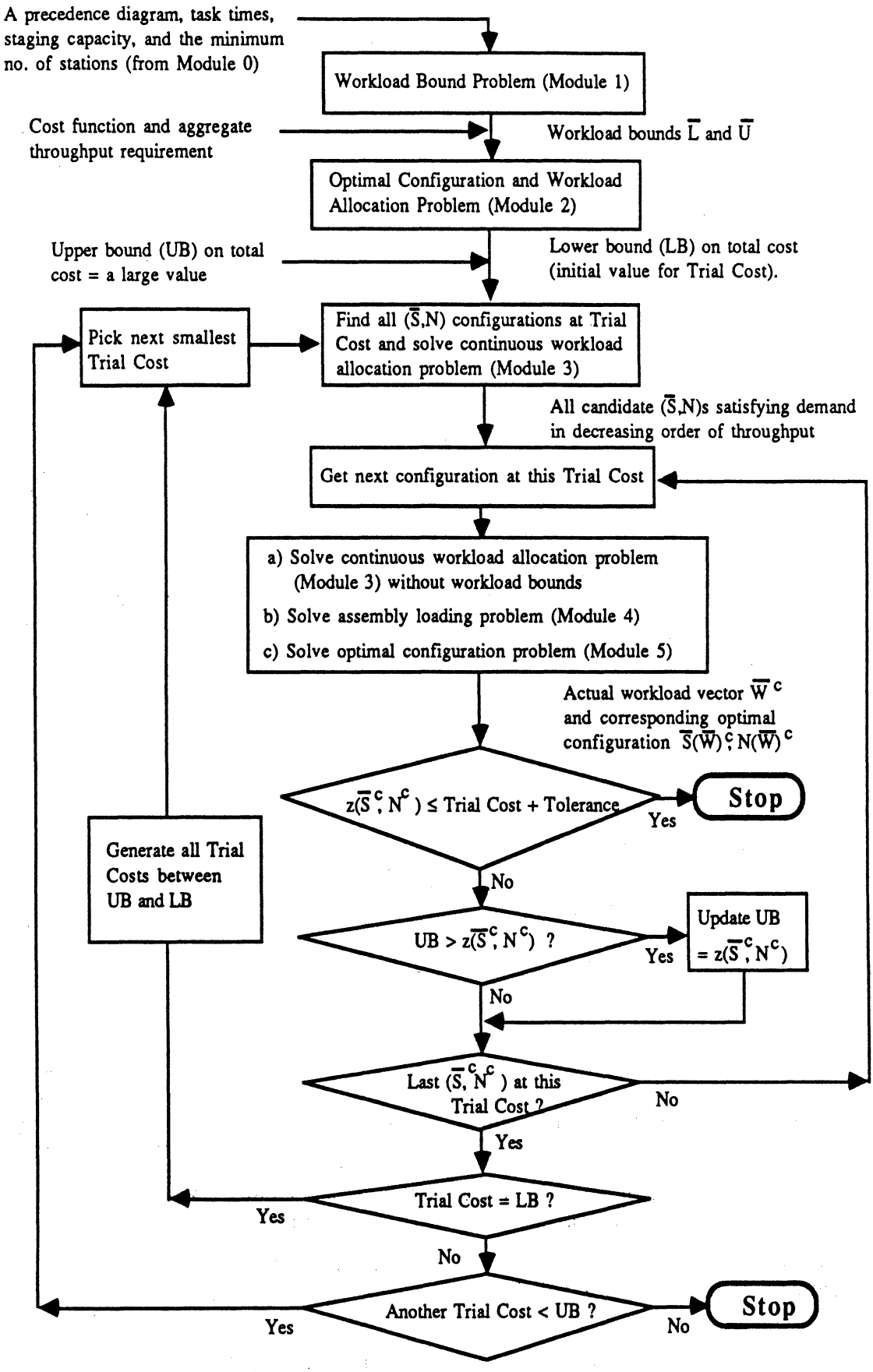

Figure 3. Flowchart of the proposed methodology

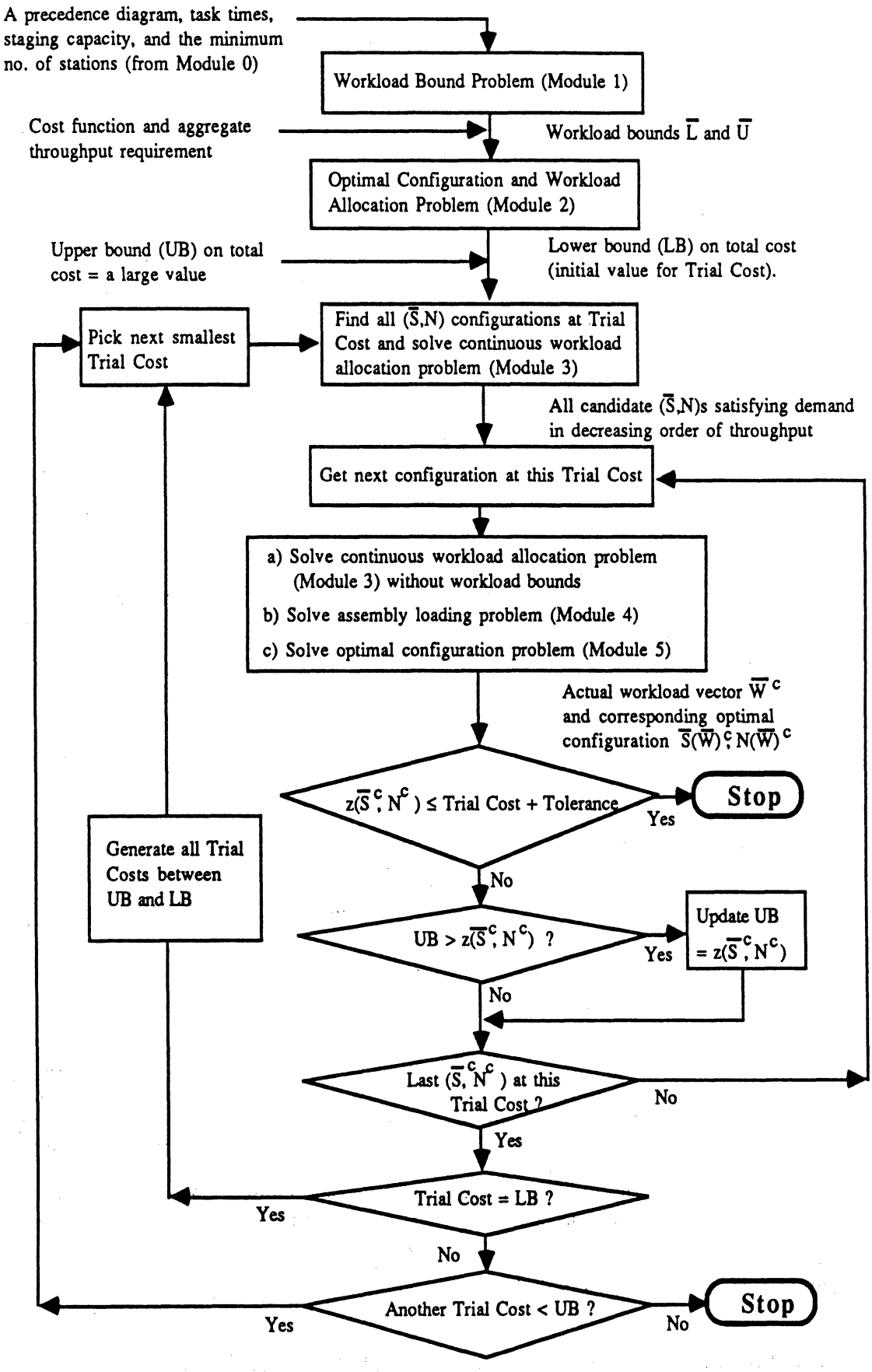

Figure 3. Flowchart of the proposed methodology

task time t_i	1 to 9					
density d	.05		.25		.50	
staging capacity R	15	30	15	30	15	30
cost lower bound	$468,000	$416,000	$468,000	$416,000	$468,000	$416,000
M_0, N	7, 14	4, 13	7, 14	4, 13	7, 16	4, 13
server vector $\bar{S}$	(2,2,2,3,2,2,2)	(3,3,3,4)	(2,2,3,2,2,2,2)	(3,3,4,3)	(2,2,2,2,2,2,2)	(3,3,4,3)
cost upper bound	$468,000	$416,000	$468,000	$416,000	$472,000	$416,000
verified optimum	yes	yes	yes	yes	no (0.8%)	yes
no. of iterations	1	1	1	1	2	1
CPU time (sec.)	3.5	1.2	2.6	1.0	3.3	.8

Table 1. Experiments with a general graph & lumpy task times: Replication 1

task time t_i	1 to 9					
density d	.05		.25		.50	
staging capacity R	15	30	15	30	15	30
cost lower bound	$512,000	$456,000	$512,000	$456,000	$512,000	$456,000
M_0, N	7, 15	4, 13	7, 15	4, 13	7, 15	4, 13
server vector $\bar{S}$	(2,2,3,2,3,2,3)	(3,4,4,4)	(2,2,3,2,3,2,3)	(4,3,4,4)	(2,2,3,2,3,2,3)	(4,3,4,4)
cost upper bound	$520,000	$456,000	$520,000	$456,000	$520,000	$456,000
verified optimum	no (1.6%)	yes	no (1.6%)	yes	no (1.6%)	yes
no. of iterations	3	1	3	1	3	1
CPU time (sec.)	80.8	.9	46.4	.8	7.6	2.3

Table 2. Experiments with a general graph & lumpy task times: Replication 2

task time t_i	1 to 9					
density d	.05		.25		.50	
staging capacity R	15	30	15	30	15	30
cost lower bound	$492,000	$436,000	$492,000	$436,000	$492,000	$436,000
M_0, N	7, 17	4, 13	7, 17	4, 13	7, 17	4, 13
server vector $\bar{S}$	(2,2,2,3,2,2,2)	(4,3,4,3)	(2,2,2,3,2,2,2)	(4,3,4,3)	(2,2,3,2,2,2,2)	(4,3,4,3)
cost upper bound	$504,000	$436,000	$504,000	$436,000	$504,000	$436,000
verified optimum	no (2.4%)	yes	no (2.4%)	yes	no (2.4%)	yes
no. of iterations	1	1	1	1	1	1
CPU time (sec.)	14.7	0.7	21.8	.8	11.5	1.0

Table 3. Experiments with a general graph & lumpy task times: Replication 3

task time t_i	5					
density d	.05		.25		.50	
staging capacity R	15	30	15	30	15	30
cost lower bound	$508,000	$448,000	$508,000	$448,000	$508,000	$448,000
M_0, N	7, 20	4, 14	7, 20	4, 14	7, 20	4, 14
server vector $\bar{S}$	(2,2,2,2,2,2,2)	(4,3,4,3)	(2,2,2,2,2,2,2)	(4,3,4,3)	(2,2,2,2,2,2,2)	(4,3,4,3)
cost upper bound	$520,000	$448,000	$520,000	$448,000	$520,000	$448,000
verified optimum	no (2.3%)	yes	no (2.3%)	yes	no (2.3%)	yes
no. of iterations	1	1	1	1	1	1
CPU time (sec.)	3.5	.6	3.5	.6	3.8	.7

Table 4. Experiments with a general graph & identical task times

throughput reqmt.	150			
density d	.1		.5	
staging capacity R	15	30	15	30
cost lower bound	$169,000	$112,000	$169,000	$113,000
M_o, N	8, 9	4, 14	8, 9	4, 18
server vector $\bar{S}$	(1,1,1,1,1,1,1,1)	(1,1,2,1)	(1,1,1,1,1,1,1,1)	(2,1,1,1)
cost upper bound	$169,000	$114,000	$169,000	$118,000
verified optimum	yes	no (1.8%)	yes	no (4.4%)
no. of iterations	1	2	1	3
CPU time (sec.)	0.9	8.6	0.3	1.5

Table 5. Experiments with unequal staging space requirements and throughput requirement 150

throughput reqmt.	300			
density *d*	.1		.5	
staging capacity R	15	30	15	30
cost lower bound	$228,000	$201,000	$228,000	$204,000
M_o, N	8, 48	4, 24	8, 35	4, 25
server vector $\bar{S}$	(1,1,2,1,1,1,1,1)	(2,2,3,2)	(1,2,1,1,1,1,1,2)	(3,2,2,2)
cost upper bound	$228,000	$204,000	$235,000	$205,000
verified optimum	yes	no (1.5%)	no (3.1%)	no (0.5%)
no. of iterations	1	2	5	2
CPU time (sec.)	5.5	14.2	38.5	1.1

Table 6. Experiments with unequal staging space requirements and throughput requirement 300

References

Ammons, J.C., C.B. Lofgren and L.F. McGinnis. 1985. A Large Scale Machine Loading Problem in Flexible Assembly. *Annals of Operations Research*, Vol. 3, pp. 319-322.

Berrada, M. and K. E. Stecke. 1986. A Branch and Bound Approach for Machine Load Balancing in Flexible Manufacturing Systems. *Management Science*, Vol. 32, pp. 1316-1335.

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 Conf. on Flexible Manufacturing Systems*, K. Stecke and R. Suri, Editors, M.I.T., Cambridge, MA, Elsevier Science Publishers B. V., Amsterdam, pp. 275-280 (August 1989).

Dallery, Y. and Y. Frein. 1986. An Efficient Method to Determine the Optimal Configuration of a Flexible Manufacturing System. *Proc. of the 2nd ORSA/TIMS Conf. on Flexible Manufacturing Systems,* Ann Arbor, MI, pp. 269-282.

Gordon, W. J. and G. F. Newell. 1967. Closed Queueing Networks with Exponential Servers. *Operations Research*, Vol. 15, pp. 252-267.

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

Graves, S. C. and C. H. Redfield. 1988. Equipment Selection and Task Assignment for Multiproduct Assembly System Design. *International Journal of Flexible Manufacturing Systems*, Vol. 1, pp. 31-50.

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

Harmon, R. L. and L. D. Peterson. 1990. *Reinventing the Factory*, The Free Press, New York, NY.

Jackson, J. R. 1963. Jobshop-like Queueing Systems. *Management Science*, Vol. 10, No. 1, pp. 131-142.

Johnson, R.V. 1988. Optimally Balancing Large Assembly Lines with FABLE. *Management Science*, Vol. 34, No. 2, pp. 240-253.

Kim, Y. D. and C. A. Yano. 1989. A New Branch and Bound Approach for Loading Problems in Flexible Manufacturing Systems. Working Paper No. 89-5, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, MI.

Lee, H. F. and R. V. Johnson. 1991. A Balancing Strategy for Designing Flexible Assembly Systems. *International Journal of Flexible Manufacturing Systems*, Vol. 3, pp. 91-120.

Lee, H. F., M. M. Srinivasan and C. A. Yano. 1991a. The Optimal Configuration and Workload Allocation Problem in Flexible Manufacturing Systems. *International Journal of Flexible Manufacturing Systems* Vol. 3, 213-230.

Lee, H. F., M. M. Srinivasan and C. A. Yano. 1991b. Characteristics of Optimal Workload Allocation in Closed Queueing Networks. *Performance Evaluation*, Vol. 12, pp. 255-268.

Lee, H. F. 1989. A Methodology for Capacity Planning in Flexible Assembly Systems. Ph.D. dissertation, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, MI.

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

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

Owen, T. 1984. *Flexible Assembly Systems*. Prentice Hall, New Jersey.

Seliger, G. and B. Wieneke. 1984. Analytical Approach for Function Oriented Production System Design. *Robotics and Computer-Integrated Manufacturing*, Vol. 1, pp. 307-313.

Seliger, G., B. Wiehweger and B. Wieneke. 1987a. Descriptive Methods for Computer-Integrated Manufacturing and Assembly. *Robotics and Computer-Integrated Manufacturing*, Vol. 3, pp. 15-21.

Seliger, G., B. Wiehweger and B. Wieneke. 1987b. Decision Support in Design and Optimization of Flexible Automated Manufacturing and Assembly. *Robotics and Computer-Integrated Manufacturing*, Vol. 3, pp. 221-227.

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

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

Shanthikumar, J. G. and D. D. Yao. 1988a. On Server Allocation in Multiple Center Manufacturing Systems. *Operations Research*, Vol. 36, pp. 333-342.

Shanthikumar, J. G. and D. D. Yao. 1988b. Second-Order Properties of the Throughput of a Closed Queueing Network. *Mathematics of Operations Research*, Vol. 13, pp. 524-534.

Shanthikumar, J.G. and D. D. Yao. 1988c. Throughput Bounds for Closed Queueing Networks with Queue-Dependent Service Rates. *Performance Evaluation*, Vol. 9, pp. 69-78.

Spur, G., I. Furgac and U. Kirchhoff. 1987. Robot System Integration into Computer-Integrated Manufacturing. *Robotics and Computer-Integrated Manufacturing*, Vol. 3, pp. 1-10.

Stecke, K.E., and J.J. Solberg. 1985. 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.

Suri, R. 1983. Robustness of Queueing Network Formulae. *Journal of the ACM*, Vol. 30, No. 3, pp. 564-594.

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

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

Yano, C. A., H. F. Lee and M. M. Srinivasan. 1991. Issues in the Design and Operation of Flexible Assembly Systems for Large Products: a Simulation Study. *Journal of Manufacturing Systems* 10(1), 54-66.

Yao, D. D.and S. C. Kim. 1987. Some Order Relations in Closed Networks of Queues with Multiserver Stations. *Naval Research Logistics*, Vol. 34, pp. 53-66.