METHODS FOR CONFIGURING MANUFACTURING SYSTEMS

by

Moshin Lee

Submitted to the Department of
Mechanical Engineering in Partial Fulfillment of
the Requirements for the
Degree of

DOCTOR OF PHILOSOPHY
in Mechanical Engineering

at the

Massachusetts Institute of Technology

September 1993

Signature of Author _____
Department of Mechanical Engineering
August 6, 1993

Certified by _____
Professor George Chryssolouris
Thesis Supervisor

Accepted by _____
Professor Ain A. Sonin
Graduate Committee

To my parents

METHODS FOR CONFIGURING MANUFACTURING SYSTEMS

by

Moshin Lee

Submitted to the Department of Mechanical Engineering
on August 6, 1993 in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy in
Mechanical Engineering

Abstract

By describing a manufacturing system with the help of a vector of numerical decision variables $\mathbf{x}$, and the system's performance by a scalar performance index y, the manufacturing system configuration problem can be defined as the problem of finding $\mathbf{x}$ such that $y(\mathbf{x})$ is maximized. A general performance index y and decision variables $\mathbf{x}$ suitable for the configuration of an actual high-volume machining system from industry were defined. These decision variables specify the machines types, the assignment of machining operations to machines, and the buffer capacities and locations of a manufacturing system configuration. Although the industrial problem studied involved only two machine types, the decision variables were generalized to cover problems with an arbitrary number of machine types. The performance index accounts for the flexibility that is required of the system to be configured.

Two broad solution approaches to this problem were investigated. In the *forward approach*, a forward model $f(\mathbf{x}) \approx y(\mathbf{x})$ is constructed. Then, an enumeration procedure is used to iterate through the space of solutions $\mathbf{x}$, and the solution with the best value of $f(\mathbf{x})$ is prescribed. In the *inverse approach*, an inverse model $f^{-1}(y) \approx \mathbf{x}$ is given a goal performance index value y_{goal}, and the configuration $f^{-1}(y_{goal})$ is prescribed. Regression and neural network forward models and neural network inverse models were investigated. These computationally efficient empirical models were fit to $(\mathbf{x},y)$ data generated via accurate but computationally expensive discrete-event simulation. The topography of the $\mathbf{x} \rightarrow y$ mapping was evaluated via roughness measures based on finite-difference approximations of integrated first and second derivatives and compared to that of known analytical functions.

The performance of manufacturing system configurations prescribed via the empirical models were compared to that of configurations prescribed via analytical and simulation forward models. Given sufficient simulation data, forward and inverse network-prescribed configurations are superior to those prescribed via regression, simulation plus hill-climbing and analytical models. Beyond a further number of simulation data, simulation plus hill-climbing-prescribed configurations attain performance comparable to that of network-prescribed configurations.

Thesis Supervisor: Professor George Chryssolouris

Title: Associate Professor of Mechanical Engineering

Acknowledgments

I have been privileged to work with generous, dedicated people throughout the creation of this thesis. First and foremost, I would like to thank my advisor and friend, Professor George Chryssolouris, for giving me the opportunity to do all of this in the first place. Over these years, I have tried to absorb something of his gift for dreaming systematically; it has certainly been a fruitful journey.

The committee members, Professor Stephen Graves, Professor Michael Jordan and Professor Patrick Winston, commented incisively but always kindly on my work and made it a much better product. I would like to thank them for their efforts.

The lab (or, affectionately speaking, The Dog Pound) has been a deep well of friendship and energy over the past six years. I would like to thank numerous Buddies for guidance, help and good times: Paul Sheng, Jim Pierce, Stan Kyi, Mike Domroese, Nick Anastasia, Kristian Dicke, Subramaniam, Nick Nassuphis, Don Lee, Alvin Ramsey, Andrew Yablon and Andrew Heitner. Other Buddies have been particularly helpful as well: Eric Hopkins, Sean Arnold, Scott Tang, Robert Kallenberg, Hui Nam, Reinoud Hermans and Milan Nedeljkovic.

Finally, I thank my family – my parents, my sister and Christina – my source of strength.

TABLE OF CONTENTS

1. Introduction

A manufacturing system can be defined as a combination of humans with machinery and equipment that are bound by a common material and information flow. The configuration of a such a system can be viewed as the process of mapping the system's performance requirements (specified via numerical performance measures) onto a description of a physical system (specified via numerical decision variables) which will achieve the required performance (Figure 1-1). This thesis presents approaches for performing this process.

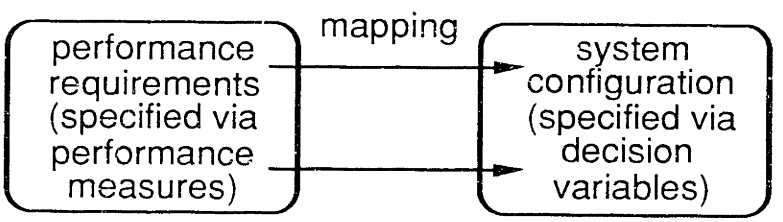


FIGURE 1-1. Configuration of a manufacturing system.

Typically, performance requirements are stated either in the form of an objective (i.e., maximize or minimize a *performance index* which is a function of one or more *performance measures*) or in the form of goals (i.e., restrictions on the allowable values of some performance measures). Each performance measure quantifies an individual aspect of manufacturing system performance. Performance measures are either benefit measures (the higher the better) or cost measures (the lower the better).

An example of a manufacturing system of the sort addressed in this thesis is shown below (Fig. 1-2). The configuration of such a system may specify the following attributes:

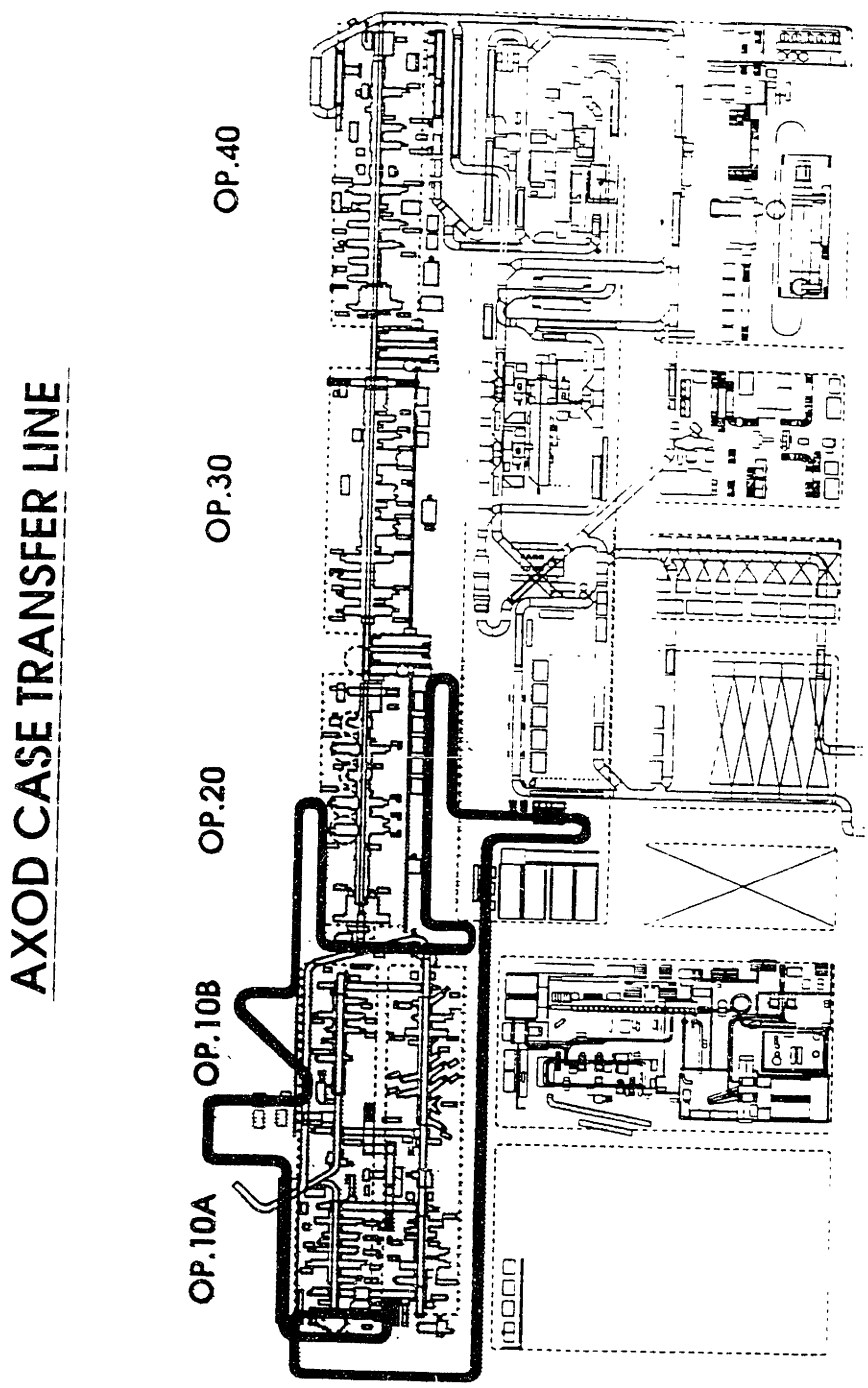


FIGURE 1-2. An example manufacturing system.

1. Which machine types comprise the system.
2. Which machine types perform which operations.
3. The capacities and locations of inventory storage buffers.

Examples of performance measures for such a manufacturing system include:

1. Production rate (a benefit measure).
2. Average system inventory level or "work-in-process" (a cost measure).
3. Maintenance cost (a cost measure).

1.1. Problem Formulation

The configuration of a manufacturing system may be characterized by a vector of decision variables $\mathbf{x} = \{x_1, \ldots, x_n\}$. Its performance may be characterized by one or more performance measures $\mathbf{z} = \{z_1, \ldots, z_m\}$. This performance is a function of the configuration (i.e., $\mathbf{z} = \mathbf{z}(\mathbf{x})$). In general, this function may be very difficult to evaluate analytically, because it depends on the dynamics of a complex manufacturing system.

In this thesis, the manufacturing system configuration problem is formulated as an optimization problem:

$$\text{Find } \mathbf{x} \text{ such that } y(\mathbf{z}(\mathbf{x})) \text{ is maximized,} \tag{1-1}$$

where $y(\mathbf{z})$ is a known algebraic objective function involving the performance measures $\{z_1, \ldots, z_m\}$, which are in turn generally unknown functions of the configuration $\mathbf{x}$. This

means that the performance index $y(\mathbf{x})$ is generally an unknown function of the configuration $\mathbf{x}$.

Two general categories of solution approaches may be used to solve the manufacturing system configuration problem. The *forward approach* requires a *forward model* f such that $f(\mathbf{x}) \approx y(\mathbf{x})$. The model $f(\mathbf{x})$ is called a forward model because it takes as input a configuration $\mathbf{x}$ and outputs an estimate of the configuration's performance $y(\mathbf{x})$, and this follows the direction of causality between configuration and performance. Such a model must be applied in conjunction with an enumeration procedure in order to generate a final solution. The basic solution approach is summarized in Figure 1-3:

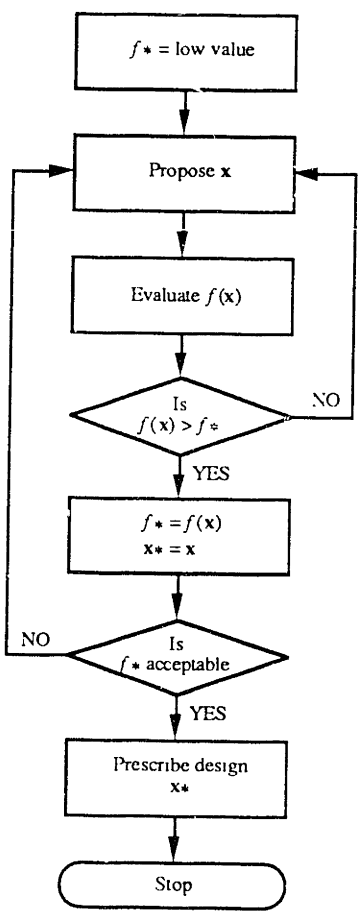


FIGURE 1-3. Forward approach to manufacturing system configuration.

Once a sufficient number of iterations have been completed, the prescribed solution is $\mathbf{x}*$.

Alternatively, the *inverse approach* may be used to solve the manufacturing system configuration problem. This approach requires an inverse model f^{-1} such that $f^{-1}(y(\mathbf{x})) \approx \mathbf{x}$. The model $f^{-1}(y)$ is called an inverse model because it takes as input performance and outputs a corresponding configuration, which is opposite the direction of the configuration-to-performance causality. The basic solution approach is:

1. Establish y_{goal}.
2. Evaluate $\mathbf{x}* = f^{-1}(y_{goal})$.

There are advantages and disadvantages to each solution approach. The advantages of the forward approach are:

- It is easy to tailor the mechanism for proposing trial configurations $\mathbf{x}$ so that illegal configurations (e.g., too large of a buffer size) are avoided.
- The mapping from decision variables $\mathbf{x}$ to the performance index $y(\mathbf{x})$ is one-to-one, which may make the model $f(\mathbf{x})$ relatively easier to construct than the model $f^{-1}(\mathbf{x})$.

The disadvantage of this approach is:

- It requires an iterative enumeration process. If the space of configurations $\mathbf{x}$ is large enough, or if the time taken for a single evaluation of $f(\mathbf{x})$ is long enough, then the solution procedure can be too time-consuming, even with respect to the

long time horizon that is normally allowed for a manufacturing system configuration decision.

The advantage of the inverse approach is:

- Since no iterations are required, it may be less expensive computationally than the forward approach. Once a goal performance value is given, then the solution procedure immediately yields the prescribed design.

The disadvantages of the inverse approach are:

- It may be difficult to establish the goal performance index value y_{goal}. Its ideal value is the maximum realizable value of the performance index $y(\mathbf{x})$, but this is generally not known *a priori*. In practice, a range of values for y_{goal} need to be submitted to the inverse model $f^{-1}(y)$, and the best of the resulting range of prescribed configurations selected.
- The mapping $y(\mathbf{x}) \rightarrow \mathbf{x}$ may be one-to-many, which may make the inverse model $f^{-1}(y)$ more difficult to construct than the forward model $f(\mathbf{x})$.

1.2. The Difficulty of Relating Configuration to Performance

In general the relationship between decision variables and performance measures is extremely complex, highly non-linear, and very difficult to establish analytically. As an example, we consider an analytical calculation of the work-in-process (WIP) performance measure for a transfer line (Gershwin and Schick 1979), a manufacturing system with serial part flow that is widely used for high-volume production (Figure 1-4). Parts enter the transfer line at the first machine. Each part is processed by Machine 1, after which it is

moved into Buffer 1. The part proceeds downstream, from Machine 1 to Buffer 1 to Machine 2 and so on. Finally, it is processed by Machine N, and then leaves the system. When Machine i breaks down, the upstream buffer, Buffer i-1, begins to fill up, and the downstream buffer, Buffer i, begins to empty. If Buffer i-1 fills up to capacity, then Machine i-1 becomes blocked; if Buffer i is emptied, then Machine i+1 becomes starved.

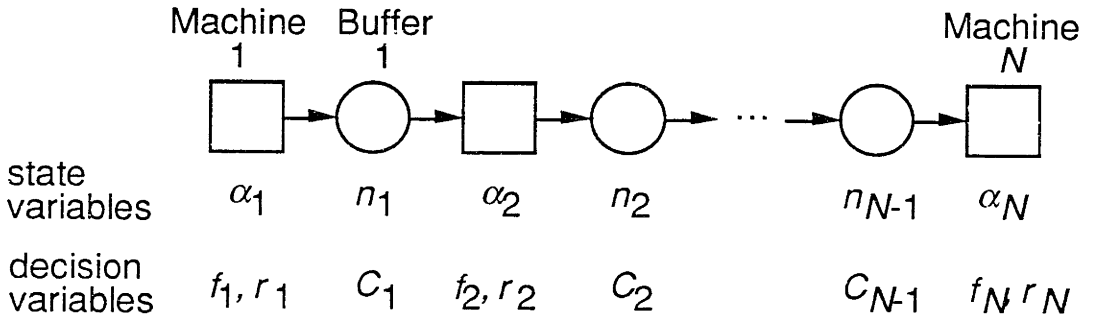


FIGURE 1-4. A schematic of a transfer line.

To calculate WIP analytically, we need to enumerate the possible system states and the probability of the system being in each state. The state of the transfer line is specified by the set of numbers:

$$S = \{n_1, n_2, \ldots, n_i, \ldots, n_N; \alpha_1, \alpha_2, \ldots, \alpha_i, \ldots, \alpha_N\} \tag{1-2}$$

where:

$n_i \equiv$ number of parts in Buffer i ($0 \le n_i \le C_i$, where C_i is the capacity of Buffer i)

$$\alpha_i = \begin{cases} 1, \text{ if Machine } i \text{ is operational} \\ 0, \text{ if Machine } i \text{ is under repair} \end{cases}$$

The state probabilities may be obtained by simultaneously solving the state transition equations, which may be written in the form (Gershwin and Schick 1979):

$$\begin{Bmatrix} Prob\{S_1(t+1)\} \\ Prob\{S_2(t+1)\} \\ \vdots \\ Prob\{S_M(t+1)\} \end{Bmatrix} = \begin{bmatrix} T_{11} & \cdots & \cdots & T_{1M} \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ T_{M1} & \cdots & \cdots & T_{MM} \end{bmatrix} \begin{Bmatrix} Prob\{S_1(t)\} \\ Prob\{S_2(t)\} \\ \vdots \\ Prob\{S_M(t)\} \end{Bmatrix}$$

$$\underline{p} \quad = \quad \underset{\sim}{T} \quad \underline{p} \qquad (1\text{-}3)$$

where:

$Prob\{S_i(t)\}$ ≡ probability of the system being in State i at time t

M ≡ the number of system states

T_{ij} ≡ probability of the system being in State i at time $t+1$, given that it is in State j at time t

The transition probabilities T_{ij} are a function of the system's decision variables, namely the f_i's (mean failure rate of Machine i), the r_i's (mean repair rate of Machine i), and the C_i's (storage capacity in number of parts of Buffer i). Equation 1-3 can be written as:

$$(\underset{\sim}{\mathbf{T}} - \underset{\sim}{\mathbf{I}})\,\underline{\mathbf{p}} = \underline{0} \qquad (1\text{-}4)$$

where:

$\underset{\sim}{\mathbf{I}}$ ≡ the identity matrix

Furthermore, the sum of the probabilities equals one:

$$\underline{\mathbf{v}}^T\,\underline{\mathbf{p}} = 1 \qquad (1\text{-}5)$$

where:

$\underline{v}$ ≡ a column vector of M ones

The unknown state probabilities $\underline{\mathbf{p}}$ are uniquely determined by Equations 1-4 and 1-5. Knowing these probabilities, the WIP of the system, assuming a unit cost per part, may be calculated as the expected value (in the probabilistic sense) of the number of parts in the system:

$$WIP = \sum_{i=1}^{N} Prob\{S_i\} \left[N + n_1(S_i) + n_2(S_i) + \ldots + n_{N-1}(S_i)\right] \tag{1-6}$$

where:

N ≡ the number of machines in the transfer line

$Prob\{S_i\}$ ≡ the probability of the system being in State i

$n_j(S_i)$ ≡ the number of parts in Buffer j in State i

Equations 1-4 and 1-5 are extremely difficult to solve for transfer lines of even moderate size because there are a large number of states, and there is one equation for each state. Equation 1-2 shows that the number of states is given by

$$M = 2^N (C_1 + 1)(C_2 + 1) \ldots (C_N + 1 \tag{1-7}$$

where:

C_i ≡ the capacity of Buffer i

For a four-machine line with three buffers each of capacity 10, $M = 21{,}296$. For a 20-machine line with the same buffer capacities, the number of equations to be solved is 6.4×10^{25}. Although the special structure of the transition matrix $\underline{\mathbf{T}}$ enables the equations to be solved more efficiently than in the general case, the resulting problem is still too large for most transfer lines of realistic size.

Computational effort aside, there are other factors which often make manufacturing systems configuration analytically intractable. For example, the above analysis applies only for transfer lines. Other systems with more complicated material flows are even more difficult to model analytically. The domain of manufacturing systems configuration is characterized by a lack of sound analytical models.

2. Literature Review

There are a number of approaches to the difficult endeavor of designing manufacturing systems. In the literature, the overall manufacturing system configuration problem has usually been decomposed into sub-problems of manageable complexity, which are then treated separately (Fig. 2-1).

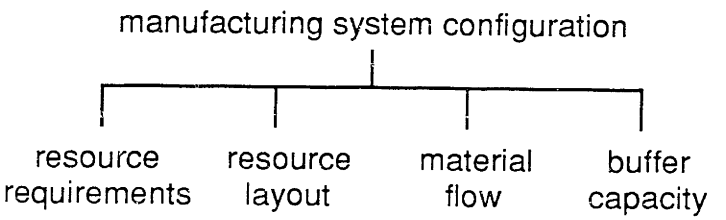


FIGURE 2-1. Sub-problems in the configuration of manufacturing systems.

One sub-problem is the resource requirements problem. For this problem, the task is to determine the appropriate quantity of each type of production resource (for example, machines or pallets) in a manufacturing system. The objective is usually cost-based, such as the maximization of investment efficiency, or time-based, such as the maximization of production rate.

The resource layout problem is the problem of locating a set of resources in a constrained floor space. The objective is typically to minimize some combination of material handling cost, travel time, and resource relocation cost.

In material flow problems, the objective is to determine the configuration of a material handling system such that some combination of flexibility, cost, production rate and reliability of the manufacturing system is maximized.

The buffer capacity problem is concerned with the allocation of work in process or storage capacity in a manufacturing system. Adequate levels of work in process maximize machine utilization and production rate, but add to floor space and inventory holding costs. The goal is to find an optimum trade-off between these conflicting benefits and costs.

Approaches that have been taken in the literature to solve these subproblems are described in the following sections.

2.1. Resource Requirements Problem

The resource requirements problem can be defined as the determination of the number of each type of production resource in a manufacturing facility during some planning horizon. A resource can be anything that is regarded as an individual production unit on the factory floor, such as a machine, an operator, a machining center with associated material handling equipment, or an automated guided vehicle (AGV).

The problem is difficult because its solution depends on a number of inter-related factors:

- The *facility layout* affects the number of resources needed because it determines which resources are accessible from each point in the manufacturing system. Lack of accessibility increases the number of resources required.
- The *process plans* and the *required volumes* of the parts to be manufactured in the system must be accounted for, since together they dictate the amount of demand that will occur for each type of resource.
- The *operational policy* affects the solution because it determines how efficiently each resource of a manufacturing system will be loaded over time.
- *Constraints* such as the equipment acquisition budget, floor space, and number of

workers force the decision of how many resources to have of one type to affect, and be affected by, the same decision for all the other types of resources.

2.1.1. Analytical Approach to the Resource Requirements Problem

Historically the first to be developed, prescriptive analytical models for the resource requirements problem are equations that express the required number of resources as a function of production factors such as the required production rate, the part scrap rate, and the breakdown frequency and duration of the resources. The resulting number is usually a fractional number which must be rounded to a neighboring integer value on the basis of intuitive considerations (Miller and Davis 1977). For example, the following equation has been proposed for determining the number of resources for a single work center. Here a work center is defined as a group of a particular type of resource, or of the same manual processing operations (Shubin and Madeheim 1951):

$$n_r = \frac{st \cdot n}{60 \cdot h \cdot sf} \tag{2-1}$$

n_r ≡ number of machines

n ≡ total required production in units per day

st ≡ standard time required to process one unit on a machine

sf ≡ scrap factor (the number of good pieces/number of scrapped pieces)

h ≡ standard numbers of hours available per day per machine

This is a single-period model which applies only to a single work center, single product, single operation facility. As with other analytical models, it is very easy to use, but solves a very limited problem. It neglects the dynamic nature of production requirements over a

planning horizon and the probabilistic nature of breakdowns and scrap parts. More importantly, it does not consider any of the interactions with the layout or the scheduling methods of the facility.

Extensions of the analytical approach have been made to accommodate the single product, multiple operation case (Apple 1950, Francis and White 1974). Other extensions have accounted for the uncertainty of the problem parameters in actual manufacturing facilities by modeling the parameters as random variables. Both single operation (Morris 1958) and multiple operation (Reed 1961) cases have been addressed in this context.

2.1.2. Mathematical Programming Approach to the Resource Requirements Problem

A mathematical programming formulation of the resource requirements problem has the advantage that constraints such as those on budget, floor space, and overtime hours are explicitly modeled. This allows interactions between the quantities of different resource types to be captured.

One of the more comprehensive mathematical programming formulations of the resource requirements problem (Miller and Davis 1978) treats the case in which resources are machines. The problem is to determine the number of machines to have in each of N work centers in each of T time periods. The manufacturing system is assumed to be a flow line which produces multiple products. Each of the N work centers in the flow line contains only one type of machine, but different work centers contain different machine types. The production characteristics of a work center (e.g., the production rate per machine and the scrap factor) vary from one period to the next, as does demand for the finished products. The objective is to find the least cost number of machines in each work center in each time

period. Machines in a work center can either be purchased at the beginning of a time period, or be left over from the previous time period. It is assumed that the relevant costs are machine investment cost, overtime operating expenses, undertime opportunity costs, and machine disposal cost. The present values of these costs are used for all calculations.

For a realistic problem, the formulation of a mathematical program may be very difficult: much effort is required to develop analytical expressions for the objective function and the constraints. Often, these expressions must then be linearized in order for the optimal solution to be found via standard techniques such as the Simplex method for linear programming. The *formulation* of a problem does not guarantee its *solution*. The formulation may be difficult to solve due to its size (i.e., great number of variables and constraints), or it might be virtually impossible to collect all of the required data (Kusiak 1987). On the other hand, simpler formulations may not capture the full difficulty of the problem.

2.1.3. Simulation Approach to the Resource Requirements Problem

In the configuration of relatively simple systems it is possible to employ simulation in a manner which is more efficient than blind trial and error. This is the case when determining the resource requirements of a flow line which must achieve a given production rate.

The production rate of a flow line (a manufacturing system with serial material flow) is limited by the slowest or *bottleneck* resource. Resources upstream of the bottleneck may experience blocking because they are constrained by the rate at which the bottleneck accepts input, while resources downstream of the bottleneck may experience starvation because they cannot be fed quickly enough. A bottleneck always exists, because in practice it is not

possible to perfectly match the output rates of all of the resources in the flow line. The bottleneck may be alleviated by:

- redistributing the work performed at the bottleneck resource,
- changing the design or the operating settings of the bottleneck resource, or
- adding additional resources at the bottleneck.

Once one of these remedial actions is taken, however, another bottleneck (albeit less severe than the first) develops elsewhere in the flow line. Note that addition of resources is only one of the options available for dealing with a bottleneck. In industry, it would be the least desirable option, because of the capital expense involved.

One procedure for designing a flow line, then, is to start with a minimal initial configuration (a given number of resources of each type). To be conservative, this configuration can be one with just one resource of each type. The next step is to determine the bottleneck via simulation. The production rate of the flow line can then be increased by taking one of the above remedial actions on the bottleneck. If the required production rate still has not been reached, then remedial action is taken on the new bottleneck, and so forth (Fig. 2-2, Ueno *et al.* 1991).

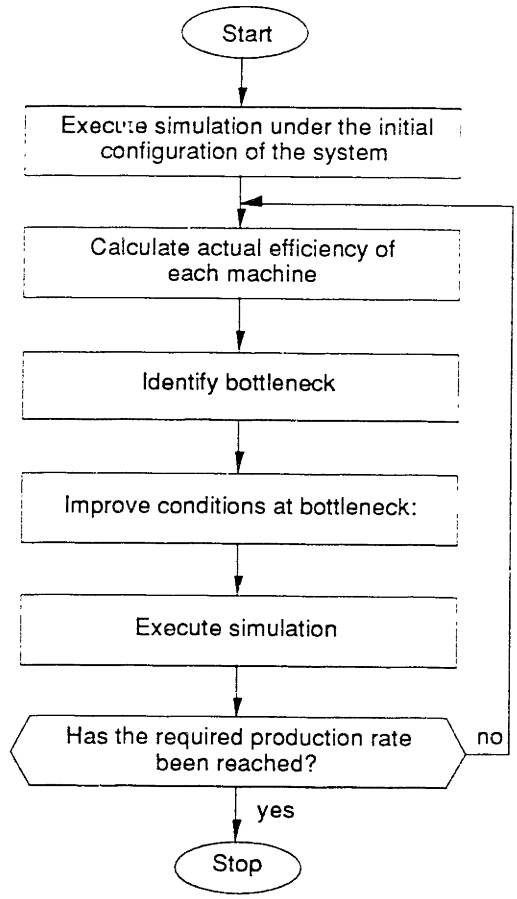


FIGURE 2-2. A procedure for designing a flow line via simulation.

The role of simulation in this procedure is to determine the bottleneck. This it can do by calculating the actual efficiency E_r of each resource, defined as (Ueno *et al.* 1991):

$$E_r = \frac{J_r}{T_r - S_r - B_r} \tag{2-2}$$

J_r ≡ Number of completed products at Resource r, except for those products which are reworked

T_r ≡ Total time at Resource r

S_r ≡ Total starving time at Resource r

$B_r \equiv$ Total blocking time at Resource r

The resource with the lowest actual efficiency E_r is deemed to be the bottleneck resource.

Statistics such as total starving time at Resource r and total blocking time at Resource r are difficult to track by any method other than simulation, especially when the system is large, the details of its operation are intricate, and the actual system does not yet exist.

This procedure has been applied in industry for the configuration of a flow line with 25 processes for the production of large diameter steel pipes. The desired production rate in this case was 7,000 pipes per month (Ueno *et al.* 1991).

2.1.4. Queueing Theory Approach to the Resource Requirements Problem

In the configuration of manufacturing systems, the role of queueing theory is similar to that of simulation. For simple manufacturing systems, the results normally provided by simulation can instead be provided by numerical solution of the algebraic equations which make up a queueing model.

As with all analytical models, queueing models have a limited range of applicability. Only certain types of manufacturing systems are easily modeled. In particular, systems in which the work in process (WIP) remains constant are particularly amenable to queueing analysis. The constant WIP condition may hold in many Flexible Manufacturing Systems (FMS), in which parts circulate about the system on a fixed number of pallets.

Queueing theory was applied to the case of an FMS which produces only a single part type, and which consists of M machine groups of one machine each. The analysis can be extended without difficulty cover the general case of multiple part types and multiple

machines at each machine group (Suri and Hildebrant 1984). Expressions were derived for the average time that a part spends in the FMS, the production rate, mean queue lengths and machine utilizations.

Queueing theory, like simulation, may be applied in a trial and error fashion for the design of manufacturing systems. The queueing model of a given system will only provide a subset of the performance measures provided by the simulation model. However, it will generally provide results within a shorter span of time (Suri and Hildebrant 1984).

2.2. Resource Layout Problem

The resource layout problem is concerned with the placement of resources on the factory floor so that some set of production requirements are met. The problem has been formulated in various ways:

1. The template shuffling formulation
2. The quadratic assignment problem (QAP) formulation
3. The relationship (REL) chart formulation

In the following sections, these formulations will be briefly discussed.

2.2.1. Template Shuffling Approach to the Resource Layout Problem

The template shuffling formulation is a manual method where a number of templates (geometric replicas of machines, material handling units, etc.) are arranged by trial and error on the prespecified floor area of the facility. It is the most widely used of all the resource layout formulations in industry (Filley 1983).

This formulation is not amenable to automatic solution because it is not well structured and layouts must be ranked by visual inspection. However, it is supported by practically all of the commercially available facilities planning software packages. These packages provide features that make the shuffling of templates more convenient than the actual manual manipulation of physical templates. Editing features enable templates to be copied, moved, deleted, and resized easily on screen (Fig. 2-3).

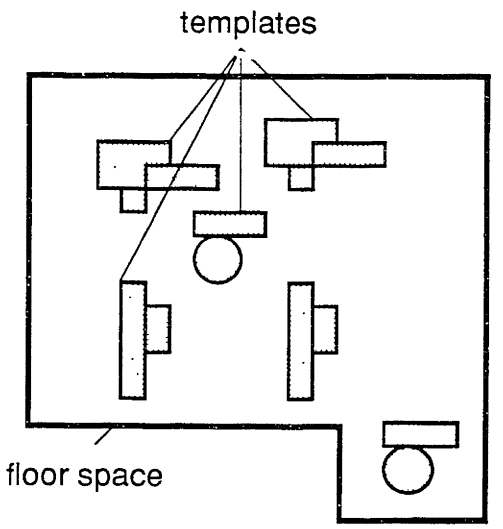


FIGURE 2-3. The template shuffling formulation of the resource layout problem.

Features such as layers are typically supported, and the designer can focus on a subset of templates by making the others invisible.

2.2.2. Quadratic Assignment Problem Approach to the Resource Layout Problem

The Quadratic Assignment Problem (QAP) formulation of the resource layout problem is to assign n resources to n spaces of equal area in order to optimize an objective. The most

common objectives include the total material handling distance, the total cost for material handling, and the cost for relocating existing resources. Material handling costs are usually assumed to be proportional to the product of the distance and the rate of material flow between each pair of resources.

The QAP formulation has several limitations:

- Realistically sized problems are difficult to solve optimally.
- Individual constraints, such as requiring that particular machines be located no further than a certain distance from each other, are not incorporated into the current problem formulation.
- It can lead to irregularly shaped resource areas in the final layout. This comes from having to break up resource areas into smaller areas so that all areas can be of equal size.
- It allows no explicit way to incorporate factors other than resource relocation cost and material movement cost into the model.
- It assumes a simple linear relationship between the cost of material movement between any pair of resources and the distance between those resources.

The mathematical programming formulation of the QAP is given by:

$$\text{Minimize} \quad \sum_i \sum_j \sum_k \sum_l c_{ijkl} x_{ik} x_{jl} \tag{2-3a}$$

Subject to:

$$\sum_j x_{ij} = 1, \quad \text{for } i = 1, \ldots, n \tag{2-3b}$$

$$\sum_i x_{ij} = 1, \quad \text{for } j = 1, \ldots, n \tag{2-3c}$$

$$x_{ij} \in \{0, 1\} \tag{2-3d}$$

The variable x_{ik} represents a (0–1) variable equal to 1 if and only if facility i is assigned to location k and c_{ijkl} is the cost of assigning facilities i and j to locations k and l respectively. The constraints (2-3b) express the fact that each resource i must be assigned to exactly one location and the constraints (2-3c) express that fact that each location j must have one facility assigned to it. This is an integer program.

Many solution approaches to this program and its variations have been proposed (Lawler 1963, Gavett and Plyter 1966, Graves and Whinston 1970, Kaufmann and Broeckx 1978, Bazaraa and Sherali 1980). However, the problem is non-polynomial-hard (NP-hard) (Sahni and Gonzalez 1976), meaning that the time taken by any algorithm to find the optimum solution must increase exponentially as the problem size (i.e., the number of resources) increases linearly. Consequently, optimal solutions for problems involving more than 15 or so resources (Liggett 1981) cannot be obtained within the realm of practical computational effort.

Because of this computational difficulty, many different heuristics have been devised for finding sub-optimal but "good" solutions to the QAP. These are either *construction procedures*, which place resources on the factory floor one after another until all resources have been placed, or *improvement procedures*, which start from an initial complete solution and then attempt to improve the solution by interchanging resources (Evans *et al.* 1987).

2.2.3. Relationship Chart Approach to the Resource Layout Problem

The relationship (REL) chart formulation of the resource layout problem is a more qualitative formulation which overcomes the stringent data requirements of the QAP formulation. A REL chart gives the desirability of having each pair of resources adjacent to

each other. This desirability is usually expressed in a letter code (Fig. 2-4):

A Absolutely essential that two departments be located adjacently

E Essential that two departments be located adjacently

I Important that two departments be located adjacently

O Marginally beneficial that two departments be located adjacently

U Unimportant that two departments be located adjacently

X The two departments should not be adjacent

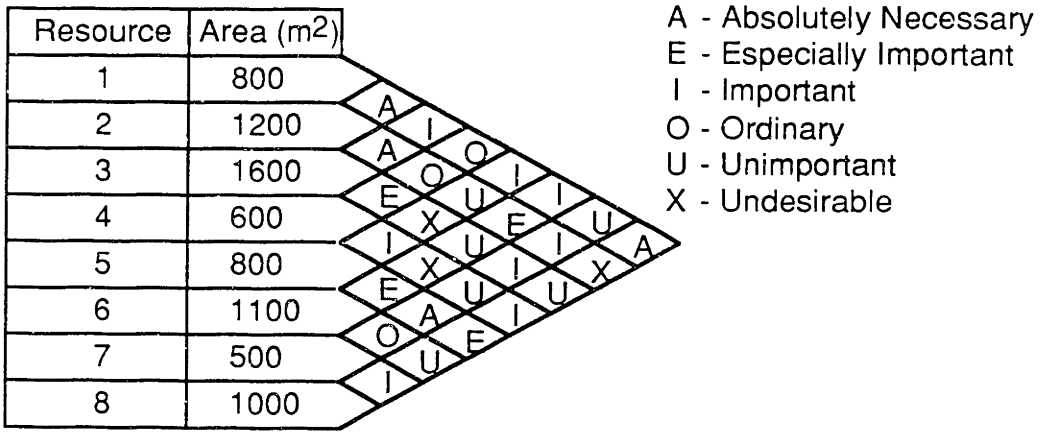


FIGURE 2-4. A REL chart for eight resources containing adjacency ratings.

These codes are transformed into numerical ratings when scoring a layout. For example, the ratings used by the Automated Layout Design Program (ALDEP) (Seehof and Evans 1967) are: $A = 4^3$, $E = 4^2$, $I = 4^1$, $O = 4^0$, $U = 0$, $X = -4^5$. A layout's score is simply the sum of the ratings for each pair of adjacent resources in the layout. The objective is to find the layout with the highest score.

Existing solution methods for the REL formulation are heuristics; they find only "good" solutions, not optimal solutions. The methods are mostly construction procedures which add one resource at a time to the facility until the layout is complete (Seehof and Evans

1967, Lee and Moore 1967). The quality of the layout is determined by the quality of the heuristic.

The REL chart formulation is based on the premise that maximizing adjacency ratings is good for the overall layout. However, the correlation between adjacency ratings and more global measures of performance, such as production rate or equipment acquisition cost, may be very weak.

2.2.4. Rule-Based Approach to the Resource Layout Problem

The application of rule-based systems to the resource layout problem has been limited. Because of the daunting combinatorial nature of the problem, it is impossible to establish generally applicable rules for its solution. Simplifications must be made.

One approach is to divide resource layouts into a few generic classes. Four classes of layouts that have been proposed are: linear single-row, circular single-row, linear double-row, and multi-row. In this approach, these layouts are to be mated with one of two classes of material handling systems: automated guided vehicles (AGVs) or robots (Fig. 2-5). Furthermore, only certain combinations of layout and material handling are considered: linear single-row/AGV, circular single-row/robot, linear double-row/AGV, and multi-row/AGV (Heragu and Kusiak 1988).

With the problem thus simplified, a rule-based system can be applied in two ways. First, based on floor space restrictions, the system can select one of the four layout/material handling combinations. The existing system always recommends the combinations in the order circular single-row/robot, linear single-row/AGV, linear double-row/AGV, and multi-row/AGV, passing to the next combination in the order only when the current

combination cannot fit within the specified factory floor space.

Once this step is complete, additional rules in the rule base are used to select one of a number of analytical models (similar to Equation 2-3, the Quadratic Assignment Problem model), plus a solution algorithm for the model. The selection of a model is based on the selected layout structure, and the numbers and sizes of the resources. The function of the selected analytical model is to locate the individual resources within the selected layout structure such that material handling cost is minimized.

It is important to note that the rule-based system itself does not generate any new design knowledge; it cannot prescribe designs that the author of its rules does not know how to design. Specifically, it can only "design" one of the four layouts in Figure 2-5. The role of the rule-based system is to apply the intuition of a human expert, as set forth in a set of rules, to prescribe the basic structure of the resource layout; then, again following the expert's judgement, prescribe an analytical model for optimizing the selected layout structure.

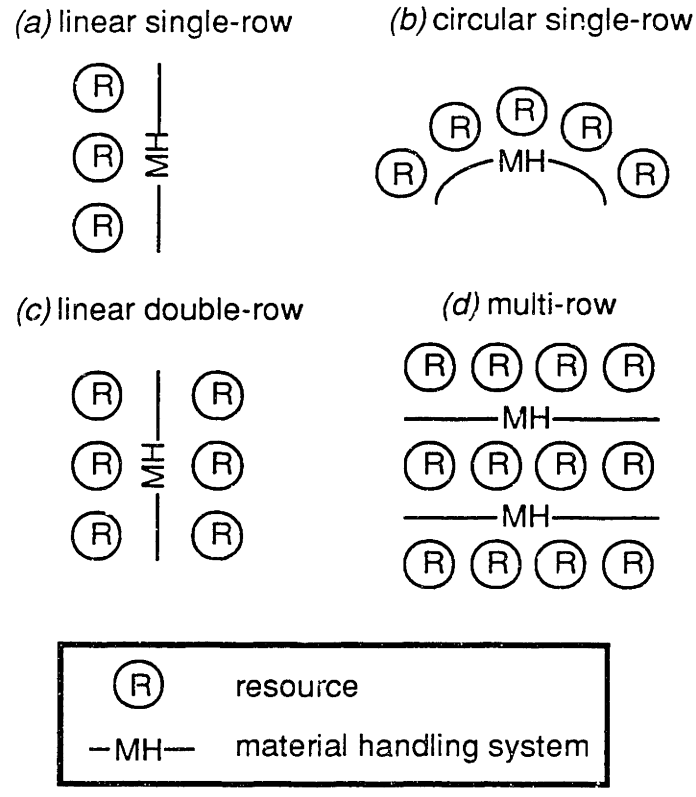


FIGURE 2-5. Classes of resource layouts.

2.3. Material Flow Problem

Material flow decision variables which must be specified in the configuration of a material handling system can be divided into two broad categories: those which specify the *type* of the material handling system, and those which specify the *configuration* of a given type of material handling system.

Configuration decision variables depend on the type of material handling system. For a material handling system based on conveyors, these variables may be the layout of the conveyors on the factory floor and their operating directions and speeds.

A different set of decision variables apply to material handling systems based on automated

guided vehicles (AGVs). AGVs are unmanned vehicles used to transport loads from one location in the factory to another. They are operated with or without wire guidance and are controlled by a computer. AGVs are often used in flexible manufacturing systems. The configuration decision variables that must be considered in the design and operation of an AGV-based material handling system are (Wilhelm and Evans 1987):

- The travel aisle layout
- The number and the locations of the pickup and delivery stations
- The pattern of material flow within the travel aisles (unidirectional, bidirectional or combinations)
- The number of vehicles required
- The routes used by vehicles during specific operations
- The dispatching logic used during operation
- The storage capacities of pickup and delivery stations

Mathematical programming has been applied to determining the pattern of material flow in an AGV-based material handling system (Gaskins and Tanchoco 1987). The objective of the proposed approach is to find the flow path which will minimize the total loaded travel of AGVs. The required inputs are: a layout of the departments of a manufacturing system, the aisles where AGV travel may occur, the locations of pickup (P) and delivery (D) stations for each department, and the material flow intensities between departments. The output of the program is a solution which indicates which aisles should be used for AGV travel, and what the direction of travel should be on each of these aisles. It is assumed that travel in a single direction only is permitted in each aisle.

The mathematical programming formulation is based on an abstraction of the input information in the form of a graph. The graph consists of nodes, which represent

pickup/delivery (P/D) stations and intersections and corners of aisles, and arcs connecting the nodes, which represent possible directions of travel along the aisles. Each aisle is therefore associated with two arcs, one for each possible direction of travel. Each node is identified by a number. An arc from Node i to Node j is identified by an integer variable x_{ij}. If x_{ij} equals 1, then the final material flow pattern will include AGV travel from the location represented by Node i to the location represented by Node j; if x_{ij} equals 0, then no material flow from location i to location j will be present in the final solution.

This abstraction can be demonstrated with the aid of a simple example (Gaskins and Tanchoco 1987) with two-departments (Fig. 2-6). The department and aisle layout is shown in Figure 2-6(a). The numbers represent distances between adjacent nodes (which are P/D stations, intersections, or corners). P is a pickup station, and D is a delivery station. The material flow intensity between these two stations (from Department 1 to Department 2) is 100. This layout is then converted into the graph of Figure 2-6(b). Deletion of arcs from this graph results in the final material flow pattern (Fig. 2-6(c)).

The objective is to minimize the loaded travel distance of the AGVs. In order to ensure a legal solution, a number of constraints must be observed. Travel must be unidirectional, which means that only one of the two arcs which connect each pair of nodes can be in the final solution. Furthermore, it must be possible to reach every node; there must be at least one incoming arc for each node. A node which violates this constraint is called a source node. It must also be possible to leave every node. A node which violates this constraint is called a sink node. Source and sink constraints apply at each node.

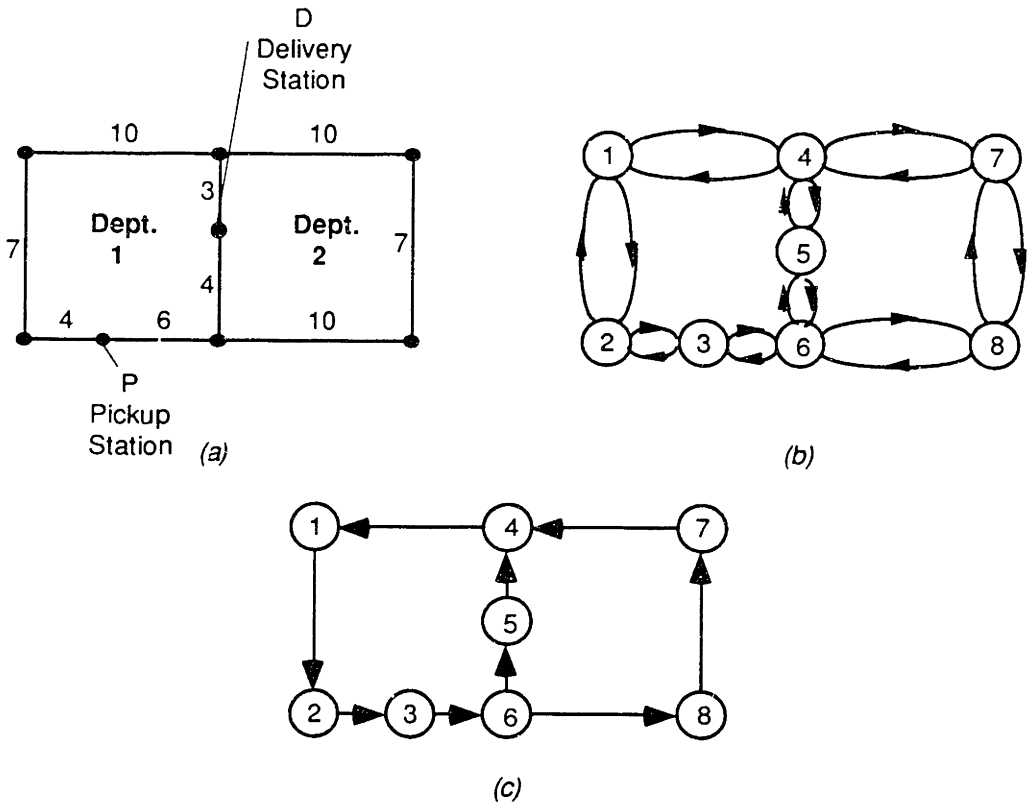


FIGURE 2-6. Steps in determining the material flow pattern in AGV-based systems.

The objective function in conjunction with the unidirectional constraints and the source and sink constraints together constitute the mathematical programming formulation of the example problem. The formulation is a nonlinear integer program.

Since its introduction, the above approach has been expanded to incorporate the function of optimally locating the P/D stations. The expanded approach (Usher *et al.* 1988) consists of two phases. Phase 1 is the original approach. In Phase 2, the locations of the P/D stations are altered by a heuristic to reduce the estimated total distance traveled by the AGVs. Since this relocation potentially spoils the optimality of the flow directions derived in Phase 1, both phases must be executed iteratively until Phase 2 no longer alters the optimality of Phase 1. In a further expansion of the approach, *unloaded* AGV travel has been

incorporated into the objective function as well (Rabeneck *et al.* 1989).

2.4. Buffer Capacity Problem

A buffer is a storage space in a manufacturing system for pieces between processing stages. Buffers serve to decouple the processing stages of a manufacturing system. By providing buffer space for inventory between machines, starvation and blockage are reduced, resulting in increased production rate. This comes at the expense of increased inventory, however.

Buffer allocation is a difficult problem because in general it is not possible to derive an analytical relationship between performance requirements and the proper buffer locations and sizes. Another aspect of the problem is that existing factory floor layouts often impose constraints upon the locations and capacities of buffers that can be implemented.

Dynamic programming has been used (Jafari and Shanthikumar 1989) to address the allocation of a fixed buffer capacity of Z pieces over the N-1 possible buffer locations in an N-machine automatic transfer line (Fig. 2-7).

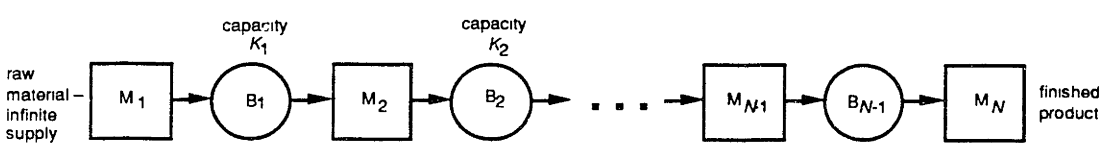


FIGURE 2-7. Transfer line system studied for the dynamic programming application.

Each individual buffer capacity K_n must not exceed a given local capacity constraint C_n. The objective is to maximize the production rate $f(\mathbf{K}_{N-1})$, where $\mathbf{K}_{N-1}$ is the vector of N-1 individual buffer capacities. If the total capacity, Z, exceeds the sum of the local capacities, $C_1 + C_2 + \ldots + C_{N-1}$, the trivial solution is to make each $K_n = C_n$. This case is excluded

from further consideration.

The above problem is cast in the form of the following dynamic programming problem:

$$\text{Maximize:} \quad f(\mathbf{K}_{N-1})$$

$$\text{Subject to:}$$

$$\sum_{n=1}^{N-1} K_N = Z \tag{2-4}$$

$$0 \le K_n \le C_n$$

In the dynamic programming approach, the multi-variable optimization problem involving the N-1 individual buffer capacities $K_1, \ldots, K_{N-1}$ is broken down into N-1 single-variable optimization problems involving the separate K's. These are solved in N-1 consecutive stages. The decisions at each stage n, $1 \le n \le N$-1, require the the calculation of the production rate of the portion of the transfer line consisting of M_1–B_1–…–B_n–M_{n+1}. Furthermore, the production rate calculated at the n^{th} stage must be expressible in terms of parameters calculated at the n-1^{th} stage. This recursive quality is a requirement of dynamic programming objective functions. Production rate can be recursively calculated via an approach in which the system is decomposed into N-1 two-stage transfer lines (M_1, B_1, M_2), (RM_1, B_2, M_3), …, (RM_{N-2}, B_{N-1}, M_N), where RM_i is an equivalent single machine replacing (RM_{i-1}, B_i, M_{i+1}). The production rate of each two-stage transfer line is analytically calculated, under the simplifying assumption of no second-stage blocking. The necessary recursion is achieved by defining the breakdown characteristics of RM_i in terms of the breakdown characteristics of RM_{i-1}, the capacity of the buffer B_i, and the breakdown characteristics of M_{i+1}.

The main requirement of this formulation is the recursive quality of the objective function. The production rate objective can be made to satisfy this condition, but generalization of the approach to other types of performance measures is difficult. For example, as inspection of Equation 2-4 will verify, the cost related to providing buffers is determined solely by the total storage space allocated and not by actual inventory levels in the storage space. In addition, the decomposition approach to production rate calculation can be implemented only for systems with serial material flow.

2.5. Multivariate Optimization Problem Formulations

A number of manufacturing system configuration approaches in the literature view the problem as a multivariate optimization problem in which a configuration is represented by a vector of decision variables $\mathbf{x} = \{x_1, \ldots, x_n\}$. This thesis investigates methods applicable to such a problem formulation.

One approach seeks to supplement the descriptive capabilities of simulation with prescriptive techniques capable of generating new manufacturing system configurations. The prescriptive technique is usually a type of search algorithm. The approach begins with an initial vector of decision variables, which represents the current configuration. The performance index of the current configuration is evaluated via simulation. Next, the search algorithm is used to modify the current configuration. If the performance of the modified configuration is better then the current configuration, then the modified configuration becomes the current configuration. In either case, the search algorithm is used to modify the current configuration again, and the process is repeated (Fig. 2-8).

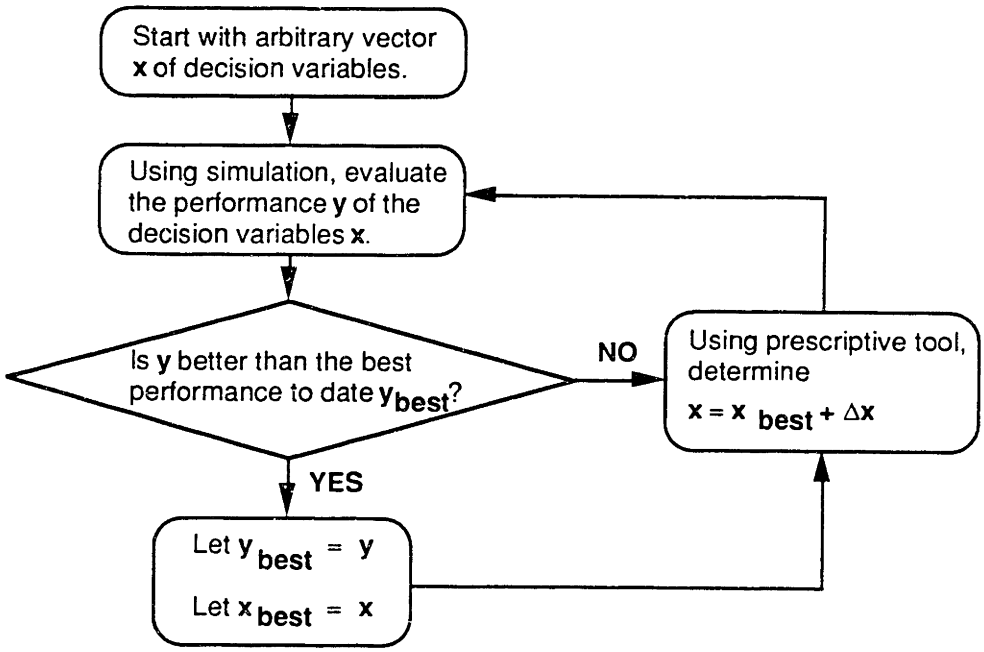


FIGURE 2-8. Use of (descriptive) simulation with (prescriptive) search for finding a configuration.

This approach has been applied to the problem of assigning buffer capacities for an automatic assembly system (AAS) consisting of N machines and N+1 buffers (Bulgak and Sanders 1988). Material flow in the category of systems considered consists of a main assembly loop and a repair loop for the repair of imperfectly-assembled parts (Fig. 2-9). A fixed number P of pallets carry parts about the system. The pallets ride on transfer chains or conveyors.

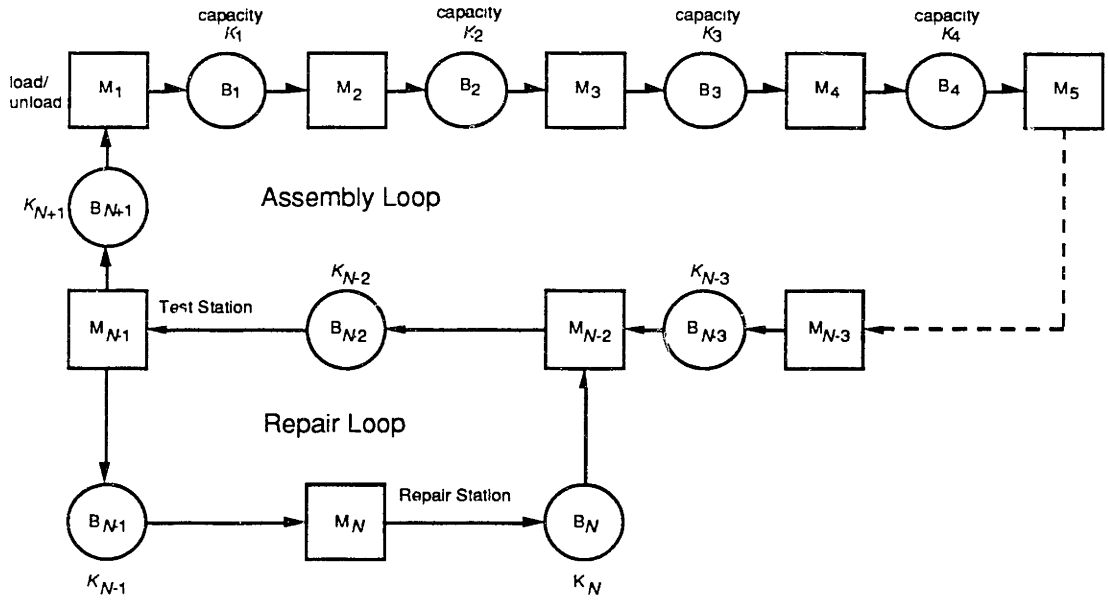


FIGURE 2-9. The automatic assembly system for the simulation with search application.

The major factors which affect the performance of an AAS are: the cycle times of the machines, the rate of occurence of parts jams for each machine, the distribution of jam clearing times, the pallet transfer times, the sampling policy for the inspection of the finished assemblies, and the buffer sizes between machines. In this application, only the buffer factor is addressed; the other factors are assumed to be given.

The objective is find buffer capacities $K_1, \ldots, K_{N+1}$ which maximize the production rate of the AAS. This problem is approached via simulation in conjunction with simulated annealing (Fig. 2-10). Simulated annealing is a search procedure (Kirkpatrick *et al.* 1983) based on an analogy between the process of annealing in solids, as described by the models of statistical mechanics, and the process of combinatorial optimization. In general terms, when a solid is annealed, (that is, heated and then gradually cooled), the positions of a vast number of molecules are gradually evolved into a configuration of very low internal strain energy in the solid. If we make the analogy between the molecular positions and internal strain energy of a solid and the state variables and objective function of a combinatorial

optimization problem, then the statistical mechanical models which describe solid annealing can provide a "recipe" for the minimization of an objective function of many state variables. In the combinatorial optimization problem, as in the solid, annealing performance is dictated by the time history of a temperature parameter T; this is the so-called "cooling schedule."

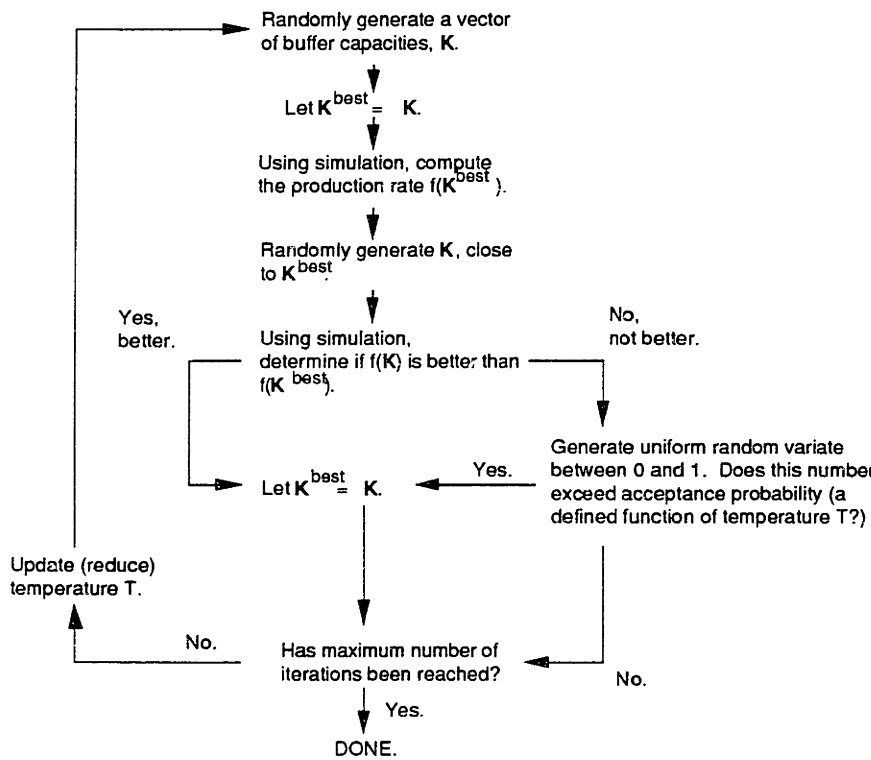


FIGURE 2-10. Use of simulated annealing for determining buffer capacities.

Since simulation is used to deal with the complexity of the relationship between the performance index (production rate) and the decision variables, this approach requires few restrictive assumptions about the nature of the performance measures and decision variables of the system to be configured. While only buffer capacities are optimized in this particular application, complicated logic such as inspection policy is incorporated into the simulation model and hence the solutions achieved are applicable for the assumed inspection policy. One possible limitation of the simulated annealing/simulation approach is that the required

number of numerical simulations may be high (in the thousands). The computational burden of the procedure may therefore be prohibitive. Other attempts to combine simulation with search (e.g., of the hill-climbing type, Caramanis 1985) suffer from the same limitation.

Other efforts have been made to systematically explore the space of decision variables via simulation. Perturbation analysis (Ho *et al.* 1984) provides, through simple numerical calculations performed during the running of a *single* simulation, measures of the sensitivity of the performance index to each decision variable. This allows attention to be focused on just the most important decision variables.

2.6. Evaluation of Existing Approaches

Most approaches to manufacturing system configuration are limited by the difficulty of modeling relationships between decision variables and relevant measures of performance. This is reflected by the following prevalent shortcomings:

- Only one type of decision variable (e.g., buffer sizes) may be prescribed. This neglects interactions between different configuration sub-problems (for example, the effect of buffer storages on the machine requirements of a system).
- Only manufacturing systems with simple structure (e.g., serial material flow systems) may be considered. Models for one system structure, say, serial material flow of a single part type, cannot be generalized to systems of different structures (for example, parallel material flow of multiple part types).
- In the construction of performance indices, relevance may be sacrificed for the sake of ease of evaluation. Examples include the sum of adjacency ratings objective used in the REL chart formulation and many of the math programming objective functions which

incorporate parameters which are impossible to collect in real manufacturing facilities (e.g., the profit of assigning a particular type of machine to a particular work center). Performance indices should be functions of meaningful performance measures such as production rate, amount of work-in-process, and tardiness.

Only the solution methods for the multivariate optimization problem formulation manage to avoid the above shortcomings by relying on numerical simulation to provide the decision variables to performance index mapping. This thesis presents methods which address the following shortcomings in the the reported applications of multivariate optimization solution methods to manufacturing system configuration problems:

- Existing applications have not explicitly addressed problems involving decision variables of multiple types (e.g., resource quantities and buffer capacities).
- A key shortcoming of existing multivariate optimization methods for manufacturing system configuration is that performance index evaluation, coming via numerical simulation, is computationally expensive. This is because computationally expensive Monte Carlo simulations are required to model stochastic elements of manufacturing system behavior such as breakdowns. As a result, relatively few configurations can be explored via simulation.

3. Proposed Methods

To motivate the proposed methods, we note that the relationships between performance measures and decision variables in manufacturing systems are complex and difficult to model by analytical means. Rather than pursuing this avenue, the proposed methods seek to establish the relationship between performance measures and decision variables empirically, by means of examples. Currently, the only general tool for creating such examples is discrete-event simulation, a computationally expensive process. The methods to be investigated in this thesis, although requiring examples generated via simulation, attempt to reduce the number of simulations required to arrive at good manufacturing system configurations by supplementing simulation models with other models such as regression and neural network models.

In all, five methods for solving the manufacturing system configuration problem (Eq. 1-1) are investigated in this thesis. Four are based on the forward approach and one is based on the inverse approach (an approach which has not been applied to manufacturing system configuration to date). These methods are (Fig. 3-1):

Forward Approach

1. Analytical forward model in conjunction with exhaustive enumeration.
2. Monte Carlo, discrete-event simulation forward model in conjunction with gradient ascent enumeration.
3. Empirical linear regression forward model fit to simulation data in conjunction with exhaustive enumeration.
4. Empirical neural network forward model fit to simulation data in conjunction with exhaustive enumeration.

Inverse Approach

5. Neural network inverse model.

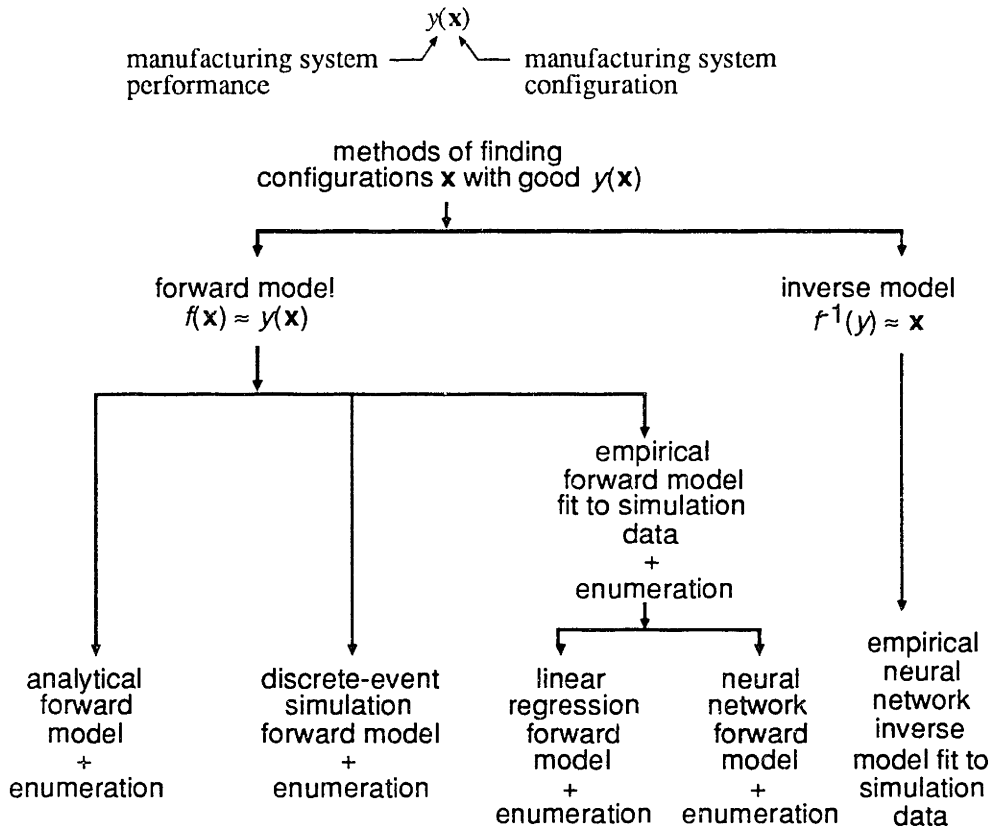


FIGURE 3-1. Manufacturing system configuration methods investigated in this thesis.

The first two methods are existing methods from the literature, and are included as references by which to assess the performance of the methods that employ empirical forward and inverse models fit to simulation data. In the next section, the key tools that underly some of these five methods will be discussed. Then, the five methods will be described.

3.1. Tools

The major tools used in the proposed methods are discrete-event simulation, regression and neural networks.

3.1.1. Discrete-Event Simulation

The discrete-event simulation software tool employed in this thesis for simulating the operation of a manufacturing system implements a Monte Carlo simulation. Conceptually, the inputs of such a computer simulator are decision variables which specify the configuration (e.g., machine processing and failure rates, machine layout), the workload (e.g., arrivals of raw materials over time, part routings), and the operational policy (e.g., "first come, first served") of a manufacturing system. The simulator assembles these data into a model of the manufacturing system which includes the rules on how the components of the system interact with each other. Given the initial state of the manufacturing system (e.g., the number and types of parts initially in inventory at various points in the system), the simulator follows the operation of the model over time, tracking events such as parts movement, machine breakdowns and machine setups. At the conclusion of the simulation, the output provided by the simulator is a set of statistical performance measures (e.g., the average number of parts in the system over time) by which the manufacturing system may be evaluated (Fig. 3-2).

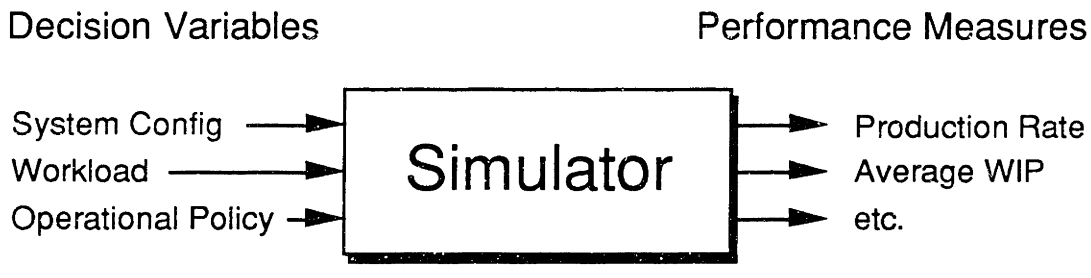


FIGURE 3-2. Function of a discrete-event simulator.

Simulation is an analysis tool because it determines only the performance of a given manufacturing system configuration. When used for the configuration of manufacturing systems, simulation must be combined with an external element which is capable of prescribing or synthesizing new configurations. Often, the external element is a person who creates a number of intuitively feasible alternative configurations, and then evaluates them with a simulator. On the basis of the simulation outputs, either the best alternative configuration is chosen, or new and possibly improved alternative configurations are proposed.

Mechanics of Simulation

Most simulation software programs model a manufacturing system as it evolves over time by a representation in which the variables that track the system's state (the state variables) change instantaneously at separate points in time (Law and Kelton 1991). These points in time are the ones at which an event occurs, where an event is defined as an instantaneous occurrence that may change the state of the system. Thus a model of this type is called a discrete-event simulation model.

Because of the dynamic nature of discrete-event simulation models, the current value of simulated time must be tracked as the simulation proceeds, and a mechanism to advance simulated time from one value to another is needed. The variable in a simulation model that gives the current value of simulated time is called the simulation clock. As for the simulation time advance mechanism, the most widely used approach is called the event-driven approach.

With the event-driven approach, the simulation clock is initialized to zero and the times of occurrence of future events are determined. The simulation clock is then advanced to the

time of occurrence of the most imminent (first) of these future events, at which point the state of the system is updated to account for the fact that an event has occurred, and knowledge of the times of occurrence of future events is also updated. Then the simulation clock is advanced to the time of the (new) most imminent event, the state of the system is updated, and future event times are determined, etc. This process of advancing the simulation clock from one event time to another is continued until eventually some prespecified stopping condition is satisfied. Since all state changes occur only at event times for a discrete event simulation model, periods of inactivity are skipped over by jumping the clock from event time to event time. Successive jumps of the simulation clock are generally unequal in size.

All discrete event-driven simulation models share the following components (Law and Kelton 1991):

- *System state:* The collection of state variables necessary to describe the system at a particular time.
- *System clock:* A variable giving the current value of simulated time.
- *Event list:* A list containing the next time when each type of event will occur.
- *Statistical counters:* Variables used for storing statistical information about system performance.
- *Initialization routine:* A subprogram to initialize the simulation model at time zero.
- *Timing routine:* A subprogram that determines the next event from the event list and then advances the simulation clock to the time when that event is to occur.
- *Event routine:* A subprogram that updates the system state when a particular type of event occurs (there is one event routine for each event type).
- *Library routines:* A set of subprograms used to generate samples from probability distributions that were determined as part of the simulation model.

- *Report generator:* A subprogram that computes estimates (from the statistical counters) of the desired measures of performance and produces a report when the simulation ends.
- *Main program:* A subprogram that invokes the timing routine to determine the next event and then transfers control to the corresponding event routine to update the system state appropriately. The main program may also check for termination and invoke the report generator when the simulation is over.

The logical relationships (flow of control) among these components is as follows. The simulation begins at time 0 with the main program invoking the initialization routine, where the simulation clock is set to zero, the system state and the statistical counters are initialized, and the event list is initialized. After control has been returned to the main program, it invokes the timing routine to determine which type of event is most imminent. If an event of type i is the next to occur, the simulation clock is advanced to the time that event type i will next occur and control is returned to the main program. Then the main program invokes event routine i, where typically three types of activities occur: (1) the system state is updated to account for the fact that an event of type i has occurred; (2) information about system performance is gathered by updating the statistical counters; and (3) the times of occurrence of future events are generated and this information is added to the event list. Often it is necessary to generate random observations from probability distributions in order to determine these future event times; such a generated observation is called a *random variate*. After all processing has been completed, either in event routine i or in the main program, a check is typically made to determine (relative to some stopping condition) if the simulation should now be terminated. If it is time to terminate the simulation, the report generator is invoked from the main program to compute estimates (from the statistical counters) of the desired mesures of performance and to produce a report. If it is not time for termination, control is passed back to the main program and the main program–timing

routine–main program–event routine–termination check cycle is repeated (Law and Kelton 1991) until the stopping condition is eventually satisfied (Fig. 3-3).

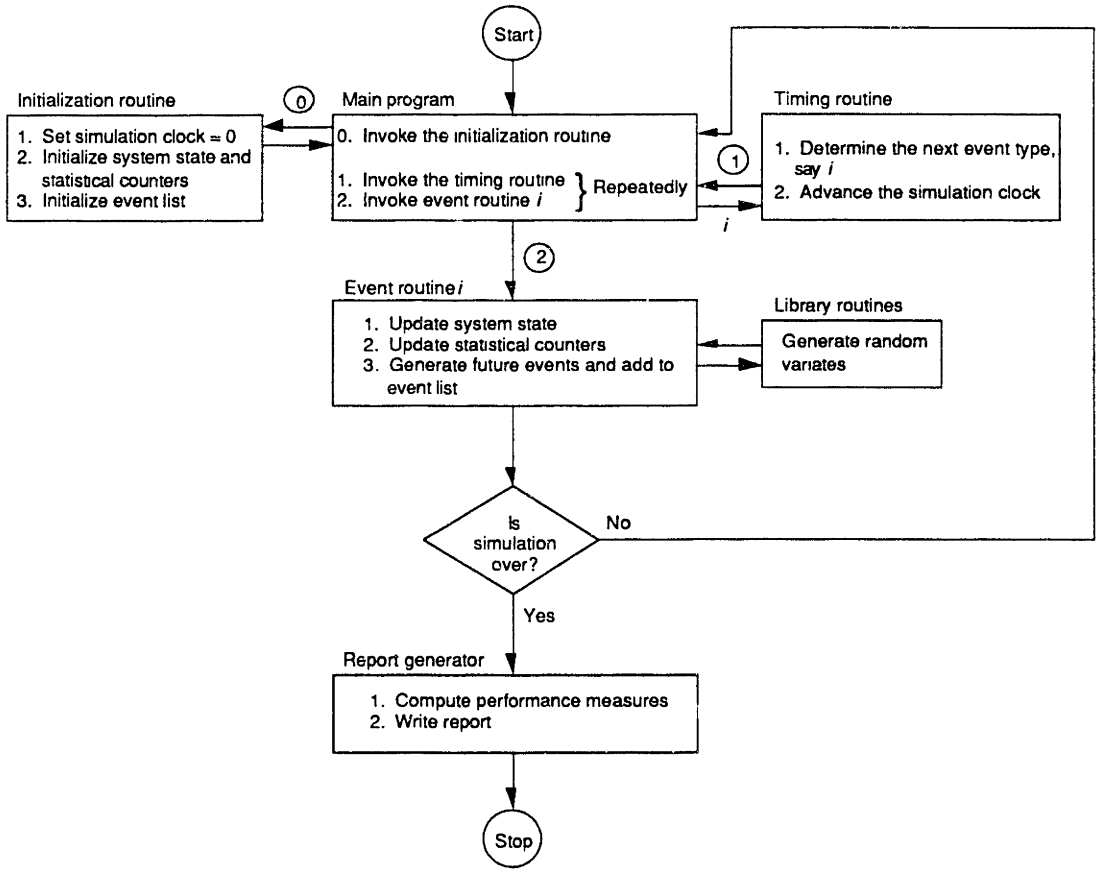


FIGURE 3-3. Flow of control in an event-driven, discrete-event simulation program.

3.1.2. Linear Regression

In a linear regression model, the output $f(\mathbf{x})$ is a linear function of empirically determined coefficients (Hogg and Ledolter 1987). The first linear regression model investigated in this thesis is a linear function of the decision variables $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$.

$$f(\mathbf{x}) = a_0 + \sum_{i=1}^{n} a_i x_i \tag{3-1}$$

The coefficients a_i are set to values which best fit the regression model to a set of input-output data $\{(\mathbf{x}^{[1]}, y^{[1]}), (\mathbf{x}^{[2]}, y^{[2]}), \ldots, (\mathbf{x}^{[s]}, y^{[s]}), \ldots, (\mathbf{x}^{[S]}, y^{[S]})\}$. Specifically, coefficients a_i are set to minimize the squared error (Draper and Smith 1981):

$$L_2 = \frac{1}{2} \sum_{s=1}^{S} \left(y^{[s]} - f\left(\mathbf{x}^{[s]}\right)\right)^2 \tag{3-2}$$

where:

S	≡	number of sample data points (in this thesis, generated via simulation)
$y^{[s]}$	≡	performance index value for the s^{th} configuration $\mathbf{x}^{[s]}$, as given by simulation
$f(\mathbf{x}^{[s]})$	≡	performance index value predicted by the regression model

The actual values a_i are obtained via least-squares solution of the equation

$$\begin{bmatrix} 1 & x_1^{[1]} & \cdots & x_n^{[1]} \\ 1 & x_1^{[2]} & \cdots & x_n^{[2]} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{[S]} & \cdots & x_n^{[S]} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y^{[1]} \\ y^{[2]} \\ \vdots \\ y^{[S]} \end{bmatrix} \tag{3-3}$$

$$\mathbf{X} \qquad \mathbf{a} \quad = \quad \mathbf{y}$$

which yields the result

$$\mathbf{a} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \tag{3-4}$$

A second linear regression model investigated in this thesis is

$$f(\mathbf{x}) = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=2}^{n} \sum_{j=1}^{i} a_{ij} x_i x_j \tag{3-5}$$

which differs from the model in Equation 3-1 via the addition of terms involving the product of pairs of different decision variables. However, the model is still linear in the coefficients a_i and a_{ij}. Again, the coefficients are obtained via Equation 3-4, with the appropriate redefinition of the matrix **X**.

3.1.3. Neural Networks

A neural network is a collection of densely interconnected, simple computational elements called nodes. Each node takes one or more numerical inputs and produces a single numerical output. This output may be propagated via network connections to other nodes, each connection multiplying it by some weight value, and the results become input values to those nodes. Certain nodes, the input nodes, receive external numerical inputs. The outputs of certain nodes, the output nodes, are externally monitored and form the numerical outputs of the neural network. The input-output transformation effected at each node can be adjusted by means of a node-specific threshold parameter. Adjusting the weights and thresholds of a neural network affects the numbers output by the neural network for a given set of input numbers. A neural network is a type of non-linear regression function.

Neural networks can be used to generalize the mapping from an input space U and to an output space V on the basis of examples. The procedure for doing so consists of two phases: a *training* phase and a *use* phase. In the training phase, the mappings from S different points in U to the corresponding points in V $\{(\mathbf{u}^{[1]},\mathbf{v}^{[1]}), (\mathbf{u}^{[2]},\mathbf{v}^{[2]}), \ldots, (\mathbf{u}^{[s]},\mathbf{v}^{[s]}), \ldots, (\mathbf{u}^{[S]},\mathbf{v}^{[S]})\}$ are provided to the neural network. (Here $\mathbf{u}^{[s]}$ and $\mathbf{v}^{[s]}$ are vectors $\{u_1{}^{[s]}, u_2{}^{[s]}, \ldots, u_n{}^{[s]}\}$ and $\{v_1{}^{[s]}, v_2{}^{[s]}, \ldots, v_m{}^{[s]}\}$ respectively). A training algorithm is then used to adapt the parameters (weights and thresholds) of the neural network so that the neural network will output something close to $\mathbf{v}^{[s]}$ when given an input

of $\mathbf{u}^{[s]}$ for all $1 \leq s \leq S$. At this point, the network has formed estimate of the mapping from U to V. In the use phase, the trained network is given an entirely new input $\mathbf{u}^{[S+1]}$, and is then expected to *generalize* the appropriate output $\mathbf{v}^{[S+1]}$ (Fig. 3-4).

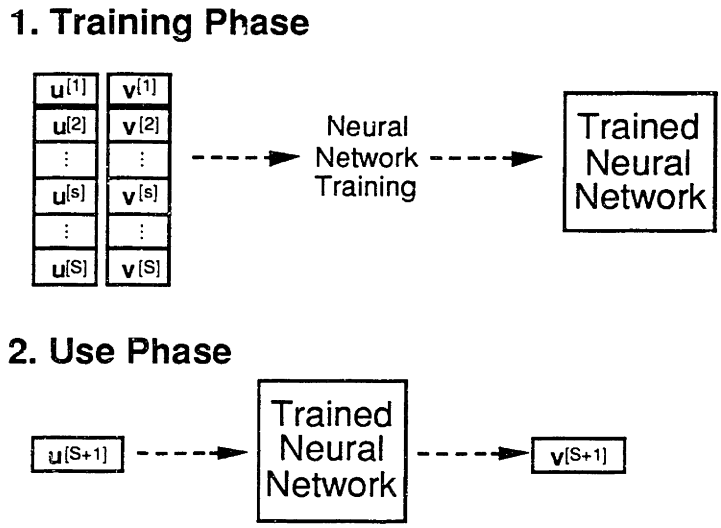


FIGURE 3-4. The use of a neural network for generalization.

Two of the methods presented in this thesis employ neural networks to generalize the relationship between performance measures and decision variables based on simulation examples. Neural networks have been used for generalization in previous research (Rumelhart *et al.* 1986, Dutta and Shekhar 1988), with the type of network called a multi-layer perceptron being the most popular. This is the type of neural network used in this thesis.

Such a network consist of individual processing elements or nodes that are arranged in layers: an input layer (layer 0), an output layer (layer L), and perhaps a number of itermediate or hidden layers (Fig. 3-5). A network maps inputs $\mathbf{u}$ to outputs $\mathbf{v}$ in the following way. The output of the i^{th} node in the input layer (layer 0) is clamped to the value of the i^{th} component of the network input:

$$out_i^{[0]} = u_i \tag{3-6}$$

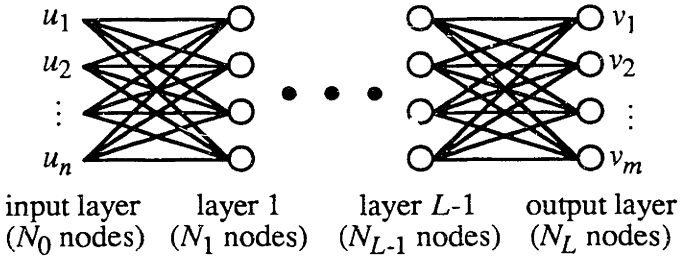


FIGURE 3-5. A multilayer perceptron neural network.

Beyond the input layer, a node j in layer m ($m > 0$) receives inputs $in_i^{[m]}$ from all nodes i in the previous layer, which it then aggregates into the net input

$$net_j^{[m]} = \sum_i w_{ji} in_i^{[m]} - w_{j0} \tag{3-7}$$

where the w_{ji}'s are the parameters or weights of the network model. If a node is in a hidden layer, this net input is then passed through a non-linear sigmoidal function to yield the node's output, $out_j^{[m]}$:

$$out_j^{[m]} = \sigma\left(net_j^{[m]}\right) = \frac{1}{1 + \exp\left(-net_j^{[m]}\right)} \tag{3-8}$$

This output is then propagated forward to the next layer of the network. That is,

$$out_j^{[m]} = in_j^{[m+1]} \tag{3-9}$$

If the node j is in the output layer (layer L), then the node's output is taken to be the j^{th} component of the network output $\mathbf{v}$, and is simply a copy of its net input:

$$v_j = out_j^{[L]} = net_j^{[L]} \tag{3-10}$$

A network's weights w_{ji} are adjusted on the basis of a number of training pairs $\{(\mathbf{u}^{[1]},\mathbf{v}_*^{[1]}), (\mathbf{u}^{[2]},\mathbf{v}_*^{[2]}), \ldots, (\mathbf{u}^{[s]},\mathbf{v}_*^{[s]}), \ldots, (\mathbf{u}^{[S]},\mathbf{v}_*^{[S]})\}$, in a process called supervised training. The most widely used training algorithm is the backprogagation algorithm (Rumelhart *et al.* 1986), which is based on gradient descent. We use a version of backpropagation in which the training pairs are presented one at a time, in a cyclical fashion. At each presentation of a training pair $(\mathbf{u}^{[s]},\mathbf{v}_*^{[s]})$, the vector $\mathbf{u}^{[s]}$ is input to the network, yielding the output vector $\mathbf{v}^{[s]}$. Each weight w_{ji} in the network is then adjusted according to the formula

$$\Delta w_{ji}[t+1] = -\eta \frac{\partial E^{[s]}}{\partial w_{ji}} + \alpha \Delta w_{ji}[t] \tag{3-11}$$

where $\Delta w_{ji}[t+1]$ is the prescribed weight adjustment, $\Delta w_{ji}[t]$ is the previous weight adjustment, η and α are constants, and $E^{[s]}$ is the squared error

$$E^{[s]} = \frac{1}{2}\left(\mathbf{v}_*^{[s]} - \mathbf{v}^{[s]}\right)^{\mathrm{T}}\left(\mathbf{v}_*^{[s]} - \mathbf{v}^{[s]}\right) \tag{3-12}$$

In Equation 3-11, the first term changes the weight in the direction of steepest descent of the error surface. The second term is a momentum term that discourages large fluctuations in the weight values. The intent is to minimize the same total squared error (Eq. 3-2) minimized by the regression models:

$$L_2 = \sum_{s=1}^{S} E^{[s]} \tag{3-13}$$

Once trained, the network may be used as a model of the mapping $\mathbf{v} = \mathbf{g}(\mathbf{u})$.

3.2. Analytical Forward Model Plus Enumeration Method

The construction of accurate analytical forward models is difficult for most manufacturing system configuration problems. The analytical model used in this thesis is a relatively crude model intended only as a benchmark against which the performance of other methods may be evaluated.

The specific outputs of the analytical model are estimates of the production rate and average work-in-process (WIP) of a manufacturing system configuration. These values are then substituted into the defining equations for the performance index (to be described in Chapter 4) in order to obtain a performance index value.

At this point, it is necessary to note that the configuration problems addressed in this thesis involve manufacturing systems that process parts in multiple stages. A part to be manufactured travels sequentially through a system in stages, each stage $i = 1, 2, \ldots, N$ being characterized by the following attributes:

$t_p(i)$ ≡ processing time of stage i

$n_m(i)$ ≡ the number of identical machines dedicated to stage i. This is set to the minimum number that will enable achievement of the demanded production rate.

$MTBF(i)$ ≡ mean time between failure of the machine type at stage i

$MTTR(i)$ ≡ mean time to repair of the machine type at stage i

The analytical model's estimate of the production rate of the system is:

$$PA = \min_{i \in [1,N]} \left\{ \frac{n_m(i)}{t_p(i)} \frac{MTBF(i)}{MTBF(i) + MTTR(i)} \right\} \tag{3-14}$$

Average WIP is estimated by assuming that at all times, all buffers are filled to capacity and each machine is processing a part.

$$\overline{WIP} = n_b b + \sum_{i=1}^{N} n_m(i) \tag{3-15}$$

$\overline{\text{WIP}}$	≡	average work-in-process
n_b	≡	number of buffers in the system
b	≡	buffer capacity (in number of parts) of each buffer in the system
$n_m(i)$	≡	number of machines in processing stage i
N	≡	number of processing stages

The evaluation time of the analytical model (the time taken to produce its output, given an input configuration) is very quick. Therefore, for the problem addressed in this thesis, the enumeration method used with the analytical model is exhaustive enumeration of all feasible manufacturing system configurations.

3.3. Discrete-Event Simulation Forward Model Plus Enumeration Method

A second solution method used as a benchmark in this thesis is the use of discrete-event simulation, as described in Section 3.1.1, coupled with the gradient ascent or hill-climbing enumeration method. The exhaustive enumeration method used in conjunction with the analytical model is not an option in this case because of the computational expense of

evaluation via simulation. The more directed and efficient hill-climbing enumeration method is therefore used.

Starting from a randomly generated initial configuration $\mathbf{x}^{[0]} = (x_1^{[0]}, x_2^{[0]}, \ldots, x_n^{[0]})$, subsequent configurations to be evaluated via simulation are generated via the rule

$$\mathbf{x}^{[k+1]} = \mathbf{x}^{[k]} + \rho \nabla y\left(\mathbf{x}^{[k]}\right) \tag{3-16}$$

ρ $\equiv$ step size

$\nabla y(\mathbf{x}^{[k]})$ $\equiv$ gradient of the performance index, evaluated at $\mathbf{x}^{[k]}$

$= (\partial y/\partial x_1|_{\mathbf{x}[k]} \ldots \partial y/\partial x_n|_{\mathbf{x}[k]})^{\mathrm{T}}$

The components of the gradient are estimated as:

$$\left.\frac{\partial y}{\partial x_i}\right|_{\mathbf{x}^{[k]}} \approx \frac{y\left(x_1^{[k]}, \ldots, x_i^{[k]}+\Delta x_i, \ldots, x_n^{[k]}\right) - y\left(x_1^{[k]}, \ldots, x_i^{[k]}, \ldots, x_n^{[k]}\right)}{\Delta x_i} \tag{3-17}$$

where the performance index values are evaluated via simulation.

If a local maximum is reached before an allowed maximum of $k = K$ iterations are completed, then a new initial configuration is randomly chosen and the hill-climbing process is started anew from there.

3.4. Linear Regression Forward Model Plus Enumeration Method

The first of the empirical modeling methods investigated in this thesis uses a linear regression model (of the form in Equation 3-1 or Equation 3-5) as the forward model. Data

for fitting the regression model, in the form of configuration-performance index pairs $\{(\mathbf{x}^{[1]}, y^{[1]}), (\mathbf{x}^{[2]}, y^{[2]}), \ldots, (\mathbf{x}^{[s]}, y^{[s]}), \ldots, (\mathbf{x}^{[S]}, y^{[S]})\}$, are generated via discrete-event simulation. The computational efficiency of the linear regression forward model enables the exhaustive enumeration method to be used.

3.5. Neural Network Forward Model Plus Enumeration Method

The second of the empirical modeling methods investigated in this thesis uses a neural network model (Section 3.1.3) as the forward model. In every other respect, it is identical to the linear regression forward model plus enumeration method (Section 3.4).

3.6. Neural Network Inverse Model Method

The inverse approach configuration method investigated in this thesis uses a neural network (Section 3.1.3) as an inverse model f^{-1} such that $\mathbf{x} \approx f^{-1}(y)$. Given a desired performance index value y_{goal}, the configuration that will achieve that performance is estimated by $\mathbf{x}* = f^{-1}(y_{goal})$.

In general, there are a number of approaches for constructing the required inverse model. The first is to use a forward model $f(\mathbf{x}) \approx y(\mathbf{x})$ to construct a lookup table containing an entry for each feasible design $\mathbf{x}_i$ and its predicted performance $f(\mathbf{x}_i)$. This table could then be searched for an entry in which $f(\mathbf{x}_i) \approx y_{goal}$, with the resulting prescribed configuration being $\mathbf{x}_i$. This approach has one principal difficulty. Certain performance measures comprising $y(\mathbf{x})$, such as the production rate and the average work-in-process, are difficult to express analytically as a function of the decision variables $\mathbf{x}$. This is due principally to the uncertain breakdown behavior of the machines in a system and the difficulty of predicting the effect of finite buffer capacities (Gershwin *et al.* 1986). Discrete-event

simulation offers an alternative way of evaluating the performance of a manufacturing system. Simulation models, being discrete-event models, can incorporate logical statements which govern a manufacturing system's operation (e.g., the actions that take place when a buffer becomes full, scheduling rules) Via random sampling, they also account for stochastic events. However, their construction is a lengthy process and their execution is computationally expensive, particularly if they are complex enough to reflect reality to a high degree of accuracy. Therefore, the use of simulation as a forward model $f(\mathbf{x})$ for this purpose is often infeasible because of excessive computational burden.

Another approach for constructing an inverse model $f^{-1}(y)$ is to generate via simulation a number of input-output training pairs of the form $\{(y(\mathbf{x}^{[1]}), \mathbf{x}^{[1]}), (y(\mathbf{x}^{[2]}), \mathbf{x}^{[2]}), \ldots, (y(\mathbf{x}^{[s]}), \mathbf{x}^{[s]}), \ldots, (y(\mathbf{x}^{[S]}), \mathbf{x}^{[S]})\}$. Each simulation run generates one training pair. A neural network model f^{-1} is fit to to the training data, with the performance index values $y(\mathbf{x})$ as the network inputs $\mathbf{u}$ and the configurations $\mathbf{x}$ as the network output targets $\mathbf{v}$. The model f^{-1} is used by supplying it with a desired performance goal y_{goal}, and recording the prescribed configuration $\mathbf{x}* = f^{-1}(y_{goal})$. This approach, which we shall call the *direct inverse approach*, may break down if the inverse function to be modeled is one-to-many, that is, if multiple configurations $\mathbf{x}$ have the same or very similar performance index values $y(\mathbf{x})$. Consider such a situation, as exemplified by the two training pairs $\{(y*, \mathbf{x}^{[1]}), (y*, \mathbf{x}^{[2]})\}$. Since the neural network cannot produce different outputs for the same input, it must "choose" to output either $\mathbf{x}^{[1]}$ or $\mathbf{x}^{[2]}$, given $y*$. However, direct inverse supervised training would, in this case, seek to minimize the total squared error

$$L_2 = \frac{1}{2}\left[\left(\mathbf{x}^{[1]} - \mathbf{x}\right)^{\mathrm{T}}\left(\mathbf{x}^{[1]} - \mathbf{x}\right) + \left(\mathbf{x}^{[2]} - \mathbf{x}\right)^{\mathrm{T}}\left(\mathbf{x}^{[2]} - \mathbf{x}\right)\right] \tag{3-18}$$

where $\mathbf{x}$ is the network output given y as input. This is minimized by the average configuration $\mathbf{x} = (\mathbf{x}^{[1]} + \mathbf{x}^{[2]})/2$. Unfortunately, the performance index value $y((\mathbf{x}^{[1]} + \mathbf{x}^{[2]})/2)$ will not in general be equal to the desired value $y*$.

In order to overcome the difficulty posed to direct inverse supervised training by one-to-many inverse mappings, an approach called *distal supervised learning* may be applied (Jordan 1992). In this approach, a neural network is first trained as a *forward* model $f(\mathbf{x})$ of the mapping $y(\mathbf{x})$ via a number of training pairs $\{(\mathbf{x}^{[1]}, y^{[1]}), (\mathbf{x}^{[2]}, y^{[2]}), \ldots, (\mathbf{x}^{[s]}, y^{[s]}), \ldots, (\mathbf{x}^{[S]}, y^{[S]})\}$ generated via simulation. This network is then appended to a second network whose function it is to perform the inverse mapping $y(\mathbf{x}) \rightarrow \mathbf{x}$ (Fig. 3-6).

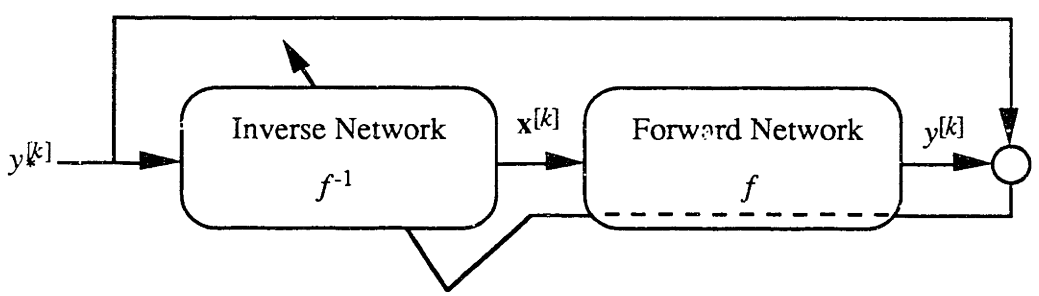


FIGURE 3-6. Distal supervised learning (Jordan 1992).

The weights of this second network are adjusted by minimizing the squared error

$$E^{[k]} = \frac{1}{2}\left(y_*^{[k]} - y^{[k]}\right)^T \left(y_*^{[k]} - y^{[k]}\right) \tag{3-19}$$

where $y*^{[k]}$ is a particular desired manufacturing system performance index value. This value is passed through the inverse network, resulting in a decision variable vector $\mathbf{x}^{[k]}$. This vector is then input to the forward network, which outputs a performance index value $y^{[k]}$. The weights of the inverse network are then adjusted according to Equation 3-11,

after replacing the subscript s with the subscript k. In this case, the derivative $\partial E^{[k]}/\partial w_{ji}$ can be evaluated using the chain rule as:

$$\frac{\partial E^{[k]}}{\partial w_{ji}} = -\frac{\partial \left(\mathbf{x}^{[k]}\right)^T}{\partial w_{ji}} \frac{\partial y^{[k]}}{\partial \mathbf{x}^{[k]}} \left(y_*^{[k]} - y^{[k]}\right) \tag{3-20}$$

The two derivatives can be calculated from the known algebraic structures of the inverse and forward networks, respectively. The important characteristic of this approach is that the weight adjustments to the inverse network are not based on explicitly specified target output vectors $\mathbf{x}*^{[k]}$. Rather, they are based on derivatives $\partial y^{[k]}/\partial \mathbf{x}^{[k]}$ which are functions of the "distal" output of the forward network (as opposed to the more "proximal" output of the inverse network itself). This has the effect of selecting a particular one-to-one mapping out of the one-to-many inverse mappings at any point $y*^{[k]}$.

One problem which may remain with the distal supervised learning is that the inverse network, once trained, may output in response to a desired performance index value $y*^{[k]}$ a configuration $\mathbf{x}^{[k]} = [x_1{}^{[k]}\ x_2{}^{[k]}\ \ldots\ x_n{}^{[k]}]^{\mathrm{T}}$ in which one or more of the decision variables are "out of bounds." For example, the allowable range for a buffer capacity decision variable may be [0, 100]; any value outside this range is out of bounds. Thus, even if the "out of bounds" configuration $\mathbf{x}^{[k]}$, when input to the forward neural network, results in an output $y^{[k]}$ which is close in value to the desired value $y*^{[k]}$, the configuration cannot be meaningfully interpreted. In this scenario the distal supervised learning method will have found a valid inverse mapping of the forward *model*, but not of the physical system represented by the forward model. What is needed is a way of constraining the inverse network outputs to acceptable ranges. This can be accomplished by modifying the squared error to be minimized to include terms that are non-zero when the components of the inverse network output $\mathbf{x}$ are out of bounds (Jordan 1992):

$$E = \frac{1}{2}(y_* - y)^T (y_* - y) + \frac{1}{2}(\mathbf{x}^+ - \mathbf{x})^T \mathbf{H}^+ (\mathbf{x}^+ - \mathbf{x}) + \frac{1}{2}(\mathbf{x}^- - \mathbf{x})^T \mathbf{H}^- (\mathbf{x}^- - \mathbf{x}) \qquad (3\text{-}21)$$

y_* ≡ desired performance index value

$\mathbf{x}$ ≡ configuration $[x_1\ x_2\ \ldots\ x_n]^T$ output by the inverse network given y_* as input

y ≡ output of the forward network given $\mathbf{x}$ as input

$\mathbf{x}^+$ ≡ vector $[x_1^+\ x_2^+\ \ldots\ x_n^+]^T$ of upper bounds on the decision variables $[x_1\ x_2\ \ldots\ x_n]^T$

$\mathbf{x}^-$ ≡ vector $[x_1^-\ x_2^-\ \ldots\ x_n^-]^T$ of lower bounds on the decision variables $[x_1\ x_2\ \ldots\ x_n]^T$

$\mathbf{H}^+$ ≡ diagonal matrix given by

$$h_{ii}^+ = \begin{cases} 1, \text{ if } x_i \geq x_i^+ \\ 0, \text{ otherwise} \end{cases}$$

$\mathbf{H}^-$ ≡ diagonal matrix given by

$$h_{ii}^- = \begin{cases} 1, \text{ if } x_i \leq x_i^- \\ 0, \text{ otherwise} \end{cases}$$

In this equation, the superscript $[k]$ from previous equations has been omitted for clarity. We will refer to the distal supervised learning method with the modified square error (Eq. 3-21) as *distal supervised learning with constraints*. This is the inverse modeling method used in this thesis.

4. An Evaluation Framework for Manufacturing Systems

In this thesis, an evaluation framework encompasses the definition of manufacturing system decision variables **x** and the definition of a performance index $y(\mathbf{x})$ (Chryssolouris *et al.* 1990). The evaluation framework proposed in this paper differs from existing frameworks (Kaplan 1983, Suresh and Meredith 1985, Wabalickis 1988, Swamidass and Waller 1990, Son 1991) primarily in the combination of specifically defined decision variables **x** and the consideration of costs that occur over the anticipated life of the system, particularly those due to part design changes. The latter are considered by quantifying the flexibility requirements typically imposed upon modern manufacturing systems and the ability of a system to accommodate these requirements.

In order to make the discussion less abstract, we will use the example of configuring a manufacturing system for multi-stage, high-volume machining. The manufacturing system to be configured must take as input raw castings and output machined parts (Fig. 4-1) at a given production rate. There is a single part type. Geometric features such as holes and slots are produced in the parts via machining operations such as milling, drilling, tapping, and boring. If a given feature requires more than one type of operation, then precedence constraints may apply (e.g., milling first, drilling second, tapping or boring third). The operations may be divided into *operation groups* such that the operations in each operation group can be performed simultaneously on the same part.

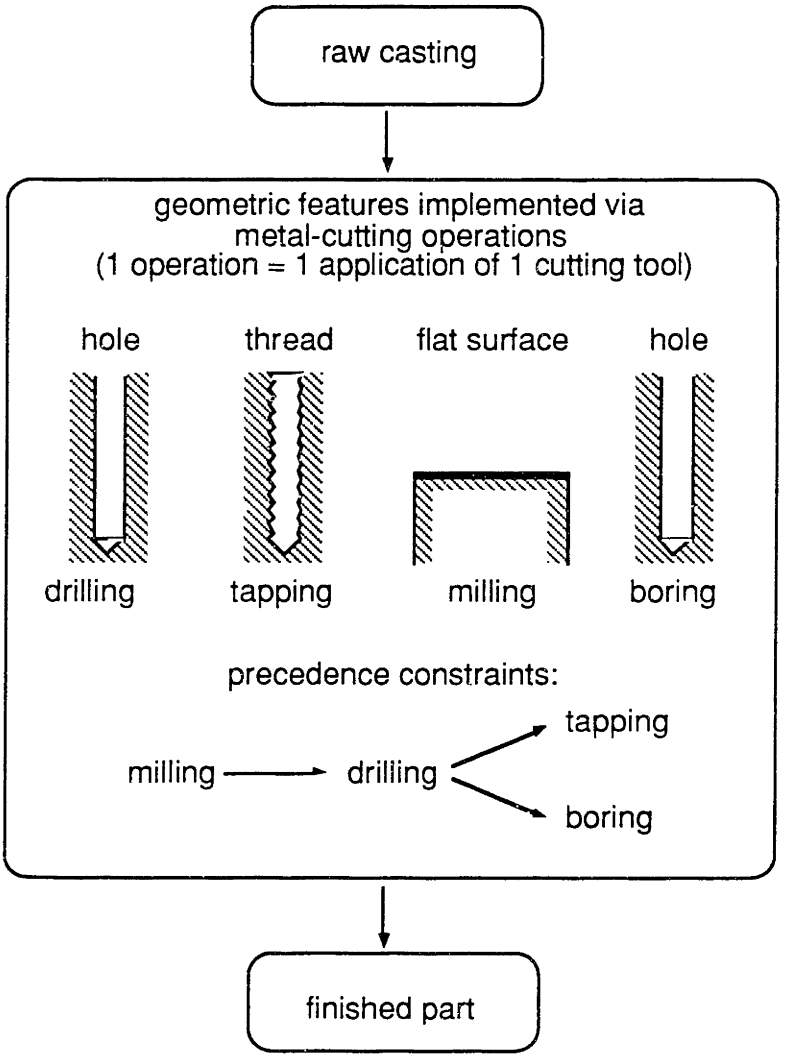


FIGURE 4-1. Required processing.

There are two generic machine classes for such a manufacturing system. A batch operation machine (BOM, Fig. 4-2a) has a tool head with multiple, simultaneously operating tools. These tools perform all the operations within an operation group simultaneously. Such a machine is dedicated to a specific operation group of a specific part design because the position of the tools relative to the tool head (and hence to the part) is fixed. If the design of the part changes (e.g., if a hole is relocated), then the machine must be replaced or substantially modified (e.g., via replacement of the tool head). A sequential operation

machine (SOM, Fig. 4-2b) has a single spindle which drives a single cutting tool. It performs individual operations sequentially. The tool is changed automatically between operations, if required, with unused tools being stored in a tool magazine. The movement of the tool spindle is programmable. This class of machine is more flexible because it does not have to be replaced when the part design changes; it only needs to be reprogrammed and possibly stocked with new cutting tools. Each *class* of machine in a manufacturing system (e.g., SOM) may be represented by multiple *types* (e.g., SOM_1, SOM_2), which are distinguished on the basis of their acquisition costs, operation costs, cost of accommodating part design changes, processing times, and so forth.

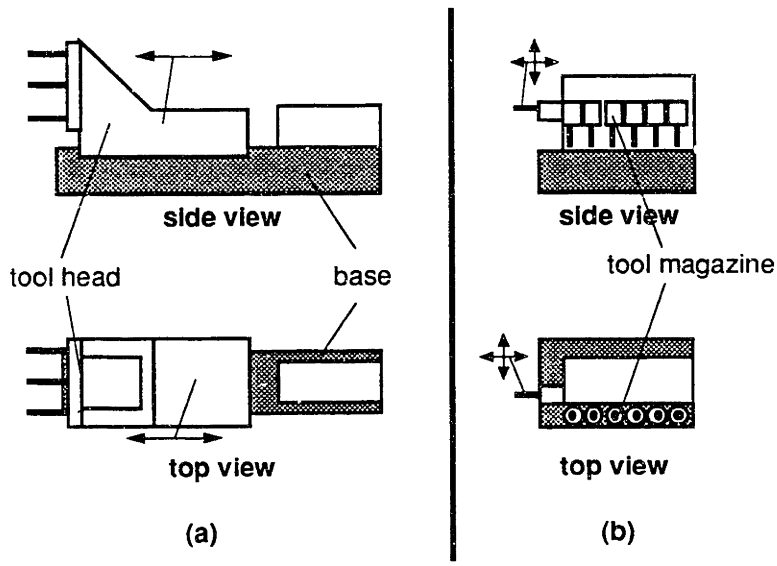


FIGURE 4-2. Schematics of the two generic machine classes: (a) batch operation machine, (b) sequential operation machine.

Batch operation machines (BOMs) are arranged serially in a transfer line (Fig. 4-3a). Parts pass from one end of the line to the other in a synchronous fashion, stopping at each position along the line for machining. The length of the line is determined by the part complexity, which determines the number of operation groups. There is one BOM per operation group. The production rate of BOM systems is limited by the processing time of

the slowest machine in the line. This proessing time cannot exceed $1/PR$, where PR is the required production rate. Sequential operation machines (SOMs) are arranged in parallel (Fig. 4-3b). Within a system, each SOM type processes the same operation groups. The number of each SOM type is dictated by the required production rate PR, and by the sequence in which the operations assigned to each machine type is performed. The optimum operation sequence will minimize the processing time of a SOM (and hence the number of them required) by minimizing the sum of spindle repositioning and tool change times. Hybrid systems are also possible (Fig. 4-3c).

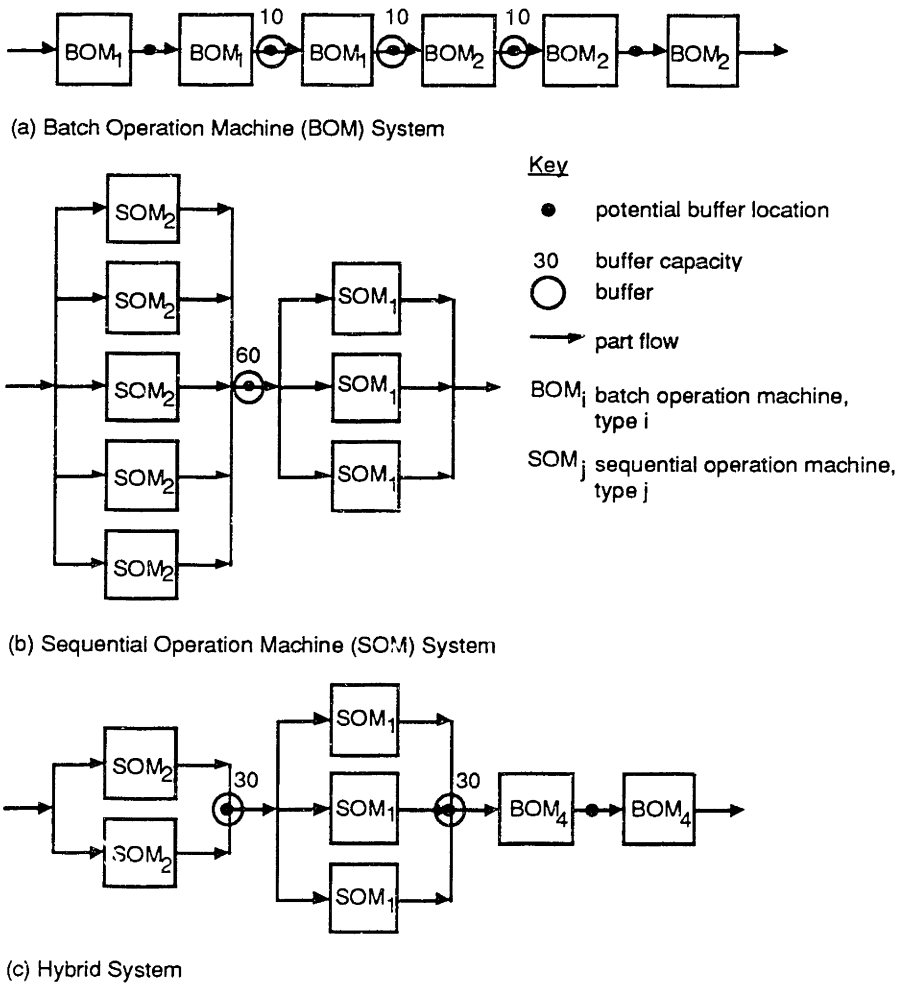


FIGURE 4-3. Generic manufacturing system types.

4.1. Decision Variables

Given performance requirements, the task of manufacturing system configuration requires the description of a suitable physical system by specifying a body of *information* (Suh 1990). This body of information can be represented either symbolically, in the form of a drawing, or numerically, in the form of values of a collection of decision variables. Symbolic representations are advantageous because each symbol can represent a large collection of information and because they are easily interpreted by humans. Numeric (decision variable-based) representations are advantageous because they can be easily stored and manipulated by computers. In this paper we will restrict our attention to numeric decision variables because they are easier to manipulate automatically and hence easier to incorporate into a procedure which automates the configuration process.

Consider the general case in which there are M machine types, where each machine type belongs to either the BOM class or the SOM class. Say that the part requires N operations, which are divided into G groups. We can define numeric manufacturing system decision variables as follows.

x_i The number of operation groups processed by machine type i ($i \in [1, M-1]$, $x_i \in [0, G]$). Only $M-1$ variables are necessary because the number of operation groups processed by machine type M is determined by default from the expression $G-(x_1+x_2+\ldots+x_{M-1})$.

s_j The j^{th} machine type that is encountered by a part as it travels through the manufacturing system ($j \in [1, M-1]$, $s_j \in [0, M]$). As a part travels through the system, the first machine type it encounters is s_1, the next type it encounters is s_2, and so on. If there are only k different machine types in the system, then s_{k+1}, s_{k+2}, ...

s_{M-1} are all set to 0. Within a system, machine type s_1 processes the first x_{s1} operation groups. Machine type s_2 then processes the next x_{s2} operation groups, and so on (Fig. 4-4). Only M-1 variables are necessary because only one block or section of each machine type is permitted in each manufacturing system configuration. For example, in Figure 4-4. once machine type 3 is assigned to the block of operation groups 1-4, it can no longer be assigned to any of the operation groups 5-8. Thus, knowing the machine types that appear in the first M-1 sections of the system (via s_1, ..., s_{M-1}), plus whether these sections account for all G operation groups (via x_1, ..., x_{M-1}), the presence or absence of an M^{th} machine type and the operation groups that it processes can be deduced. Later on, in Chapter 9, we will examine a way of generalizing the decision variables to overcome this restriction and thereby permit a greater richness of manufacturing system configuration solutions.

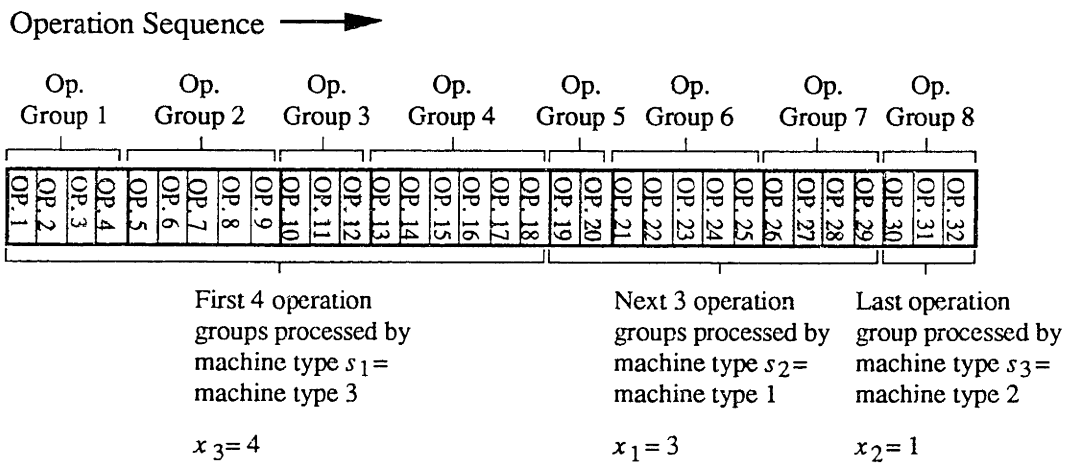


FIGURE 4 4. An illustration of the x_i and s_j decision variables for a hypothetical manufacturing system for a given operation sequence.

b The capacity of the buffers within the system ($b \in [0, b_{max}]$). It is assumed that buffers within a system have equal capacities. Since the function of a buffer is to

decouple a manufacturing system, the maximum buffer capacity b_{max} may be reasonably set to be the number of parts that would be accumulated in the buffer in the event of a breakdown immediately downstream, or the number of parts that would be drained from the buffer in the event of a breakdown immediately upstream, multiplied by some safety factor.

f The frequency with which buffers occur within the system. This is defined to be the number of buffers within the system divided by the number of potential buffer locations within the system. Within a system, buffers may be located between any two adjacent machines in the part flow.

For this general case, there are $2(M-1)+2 = 2M$ decision variables. This set of decision variables has the advantage of being relatively compact. The number of variables grows only linearly with the number of machine types M, one of the smaller parameters of the manufacturing system configuration problem; it does not, for instance, depend on the number of operations N, which is substantially larger than M.

A manufacturing system is specified by the vector of decision variables $\mathbf{x} = [x_1, x_2, \ldots, x_{M-1}; s_1, s_2, \ldots, s_{M-1}; b; f]$, and by the operation sequence of the SOMs. The latter determines the processing time and hence the number of each SOM type in the system. The space of decision variables can be represented as a tree in which a path from the start node to an end node represents a particular manufacturing system configuration (Fig. 4-5).

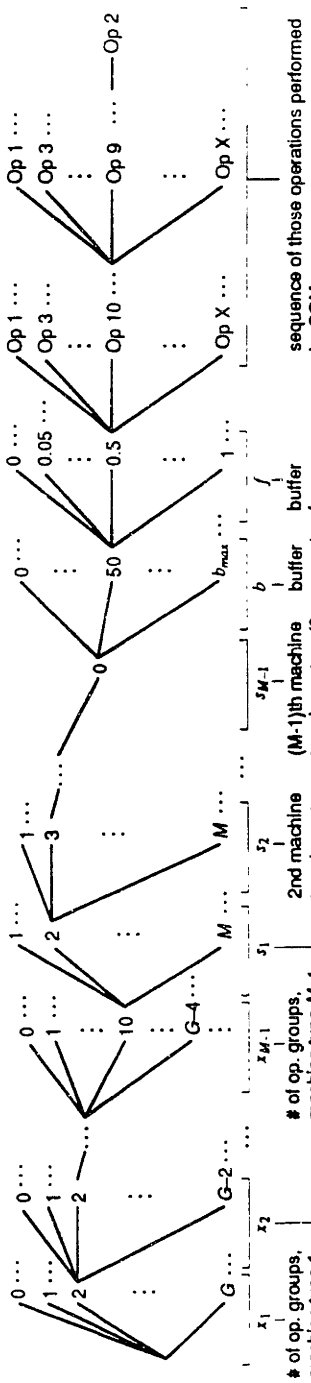


FIGURE 4-5. The space of manufacturing system configurations **x**.

From this tree (Fig. 4-5) it is apparent that the sequence of the operations performed by sequential operation machines is the most significant factor affecting the size of the solution space. In this thesis, the operation sequence is assumed to be given, an output of the process planning function. This considerably reduces the size of the solution space and thereby motivates the application of empirical forward and inverse models for solving the manufacturing system configuration problem.

As an example of the application of these decision variables, consider a specific case, the hybrid system (Fig. 4-3c), in which there are $M = 4$ available machine types: 1) SOM_1, 2) SOM_2, 3) BOM_3 and 4) BOM_4. Let there also be $G = 50$ operation groups. Machine type SOM_2 processes the first 10 operation groups, machine type SOM_1 processes the next 25, and machine type BOM_4 processes the remaining 15. Machine type BOM_3 does not appear in the system, and therefore does not process any operation groups. There are $2M = 2 \cdot 4 = 8$ decision variables. The first three are $x_1 = 25$, $x_2 = 10$, $x_3 = 0$, which are are the number of operation groups processed by machine types 1, 2 and 3 respectively. The next three decision variables are $s_1 = 2$, $s_2 = 1$, $s_3 = 4$, which indicates the sequence of the machine types encountered by a part as it travels through the system. The seventh decision variable is the capacity of the buffers, which is $b = 30$. Finally, there are 2 buffers and 3 buffer locations, so the eighth decision variable, buffer frequency f, is 2/3. To recap, the decision variable vector representing the system in Figure 4-3c under the stated conditions is ($x_1 = 25$, $x_2 = 10$, $x_3 = 0$; $s_1 = 2$, $s_2 = 1$, $s_3 = 4$; $b = 30$; $f = 0.667$).

The decision variable f establishes the number of buffers n_b in a system to be the integer closest to the product fn_{bl}, where n_{bl} is the number of buffer locations. However, there are no decision variables which explicitly specify buffer locations, and therefore these must be set according to a consistent convention. The following convention is used. Buffers are

placed every $\delta_b = \langle n_{bl}/n_b \rangle^-$ spaces, where $\langle \bullet \rangle^-$ denotes the greatest integer less than or equal to $\bullet$. In the hybrid system (Fig. 3c), $\delta_b = \langle 3/2 \rangle^- = 1$. The buffers are placed so that the number of buffer locations n_{bl-} before the first buffer is equal to (or one less than) the number of buffer spaces n_{bl+} after the last buffer space.

4.2. Performance Index

Once the decision variables **x** are defined, an evaluation framework requires the definition of a performance index $y(\mathbf{x})$. The performance index used in this thesis accounts for the inputs to a manufacturing system in terms of the costs incurred over the life of the system – particularly those related to part design change. Thus the evaluation framework considers the flexibility of a manufacturing system. The performance index also accounts for the output that is achieved by a manufacturing system, in terms of the number of parts that it produces. This index $y(\mathbf{x})$ is called efficiency, and has the general form *output/input*.

$$\text{efficiency} = y = \frac{\text{number of good parts produced during system life cycle}}{\text{total life cycle cost}} \qquad (4\text{-}1)$$

The units of this index are [parts/$]. If a figure for the revenue per part is available, then e can be converted to a unitless efficiency. The denominator, the total life cycle cost, consists of acquisition costs, operation costs and system modification costs due to part design change. These costs are incurred over the life T of the system, which consists of n_t periods of duration t ($T = n_t t$). The costs, described in detail below, are broken down by period and are incurred at the beginning of each period.

System Acquisition Cost

This cost is incurred not only when the system is first implemented, but also when

increased demand requires an expansion of system capacity.

$$C_a(p) = \sum_{i=1}^{M} c_m(i)\, n_{ma}(i, p) \tag{4-2}$$

$C_a(p)$ ≡ acquisition cost for period p

M ≡ number of machine types

$c_m(i)$ ≡ acquisition cost of one unit of machine type i

$n_{ma}(i, p)$ ≡ number of machine type i acquired in period p

Operation costs are assumed to consist of labor, inventory and maintenance costs:

Labor Cost

$$C_l(p) = c_l\, t\, n_l(p) \frac{PR(p)}{PA(p)} \tag{4-3}$$

$C_l(p)$ ≡ labor cost for period p

c_l ≡ labor cost rate

t ≡ duration of each period

$n_l(p)$ ≡ average number of workers at any given time during period p

$PR(p)$ ≡ demanded production rate for period p

$PA(p)$ ≡ actual production rate for period p

The product $c_l t n_l(p)$ represents the nominal labor cost. The ratio $PR(p)/PA(p)$ is an adjustment factor which accounts for: 1) overtime costs if the demanded production rate exceeds the system's achieved production rate without overtime; 2) labor cost savings from having to operate the system less than full time if the system's achieved production rate is

greater than the demanded production rate.

Inventory Cost

$$C_i(p) = c_i \, \overline{WIP}(p) \tag{4-4}$$

$C_i(p)$ ≡ inventory cost for period p

c_i ≡ inventory carrying cost per part per period

$\overline{WIP}(p)$ ≡ average work in process for period p

Maintenance Cost

$$C_m(p) = \sum_{i=1}^{M} n_m(i, p) \left[c_p(i) + c_r(i) \, \bar{t}_d(i) \right] \tag{4-5}$$

$C_m(p)$ ≡ total maintenance cost in period p

M ≡ number of machine types

$n_m(i, t)$ ≡ number of machines of type i in period p

$c_p(i)$ ≡ preventive maintenance cost of machine type i per period

$c_r(i)$ ≡ repair cost rate for machine type i

$\bar{t}_d(i)$ ≡ mean down time of machine type i per period

System Modification Cost Due To Part Design Change

The final cost considered in the evaluation of efficiency is the system modification cost due to part design change. This cost is an important and new contribution of the efficiency definition because it accounts for the flexibility of the system.

$$C_C(p, n_\Delta) = \delta(p, n_\Delta) \sum_{j=1}^{n_m(p)} \left[1 - \prod_{k=1}^{n_f(j)} \left(1 - P_C(j, k)\right) \right] c_C(j) \qquad (4\text{-}6)$$

$C_C(p, n_\Delta)$ ≡ part design change cost for period p if the number of periods per part design change is n_Δ

$\delta(p, n_\Delta)$ ≡ 1 if a part design change occurs in period p, 0 otherwise

$n_m(p)$ ≡ total number of machines in the system in period p, irrespective of type

$n_f(j)$ ≡ number of features worked on by the j^{th} machine in the system

$P_C(j, k)$ ≡ probability that the k^{th} feature worked on by the j^{th} machine will be modified in a design change

$c_C(j)$ ≡ cost of modifying/replacing the j^{th} machine to accommodate a part design change

The term in the square brackets is an expression for the probability that at least one of the features worked on by the j^{th} machine in the system will be modified whenever the part design changes. Referring to this probability as $P_{mod}(j)$, we see that it is a function of the probabilities $P_C(j, k)$ of individual features k requiring change when the part design changes and also of the number of features $n_f(j)$ processed by the machine. The greater these quantities are, the greater the value of $P_{mod}(j)$. Multiplying $P_{mod}(j)$ by the cost of modifying/replacing the j^{th} machine, $c_C(j)$, yields an expected modification/replacement cost for the j^{th} machine. These expected modification/replacement costs are then summed over all machines j in the system. Finally, the binary variable $\delta(p, n_\Delta)$ ensures that part design change costs are incurred only in periods in which design changes actually occur. For example, if part design changes occur every $n_\Delta = 2$ periods, they are assumed to occur at the beginning of periods 3, 5, 7, ... (There is no part design change at the beginning of period 1, because that is when the initial design is first produced.) Therefore, $\delta(3, 2) =$

$\delta(5, 2) = \delta(7, 2) = \ldots = 1$, while all other $\delta(p, 2) = 0$. In general:

$$\delta(p, n_\Delta) = \begin{cases} 1, \text{ if } p = n_\Delta i + 1, \; i \text{ a positive integer} \\ 0, \text{ otherwise} \end{cases} \tag{4-7}$$

Recognizing that the above acquisition, operation and system modification costs are incurred over the life cycle T of the system in question and taking into account the time value of money, the above costs may be combined into a single total life cycle cost.

$$C_T(n_\Delta) = \sum_{p=1}^{n_t} \frac{C_a(p) + C_l(p) + C_i(p) + C_m(p) + C_c(p, n_\Delta)}{(1 + r)^{p-1}} \tag{4-8}$$

$C_T(n_\Delta)$ ≡ present value of total life cycle cost

n_Δ ≡ number of periods per part design change

n_t ≡ number of periods in the life cycle of the system

$C_a(p)$ ≡ total machine acquisition cost for period p

$C_l(p)$ ≡ total labor cost for period p

$C_i(p)$ ≡ total inventory cost per period p

$C_m(p)$ ≡ total maintenance cost in period p

$C_c(p, n_\Delta)$ ≡ part design change cost for period p

r ≡ interest rate per period

The total life cycle cost is written as $C_T(n_\Delta)$ because it depends on the interval $n_\Delta t$ between part design changes that is imposed upon the system by external market demands. (Specifically, the part design change cost component of $C_c(p, n_\Delta)$ is affected.) Since n_Δ cannot be predicted with certainty, the approach taken in calculating efficiency is to associate probabilities $P_\Delta(n_\Delta)$ with different values of n_Δ, and to evaluate efficiency as the expected value:

$$y = \sum_{n_\Delta = 1}^{n_t} \frac{PA \bullet T}{C_T(n_\Delta)} \cdot P_\Delta(n_\Delta) \tag{4-9}$$

y ≡ efficiency

n_Δ ≡ number of periods between part design changes

n_t ≡ number of periods in the system's life cycle

PA ≡ production rate

$PA \bullet T$ ≡ number of parts produced in life cycle of duration T

$C_T(n_\Delta)$ ≡ total life cycle cost, given that the number of periods between part design changes is n_Δ

$P_\Delta(n_\Delta)$ ≡ probability that the number of periods between part design changes will be n_Δ

The probabilities $P_\Delta(n_\Delta)$ are important because they define the flexibility requirements imposed upon the system by external market forces. Coupled with the system modification costs due to part design change (Eq. 4-6), which measure a system's ability to accommodate design changes, they ensure that the efficiency index (Eq. 4-9) accounts for both the system's ability to react to change *and* the extent to which this ability is required by the market in which the system operates. The intent is to give credit to flexible capabilities only if they are indeed required (Chryssolouris and Lee 1992). Equation 4-9 is the final definition of the performance index $y(\mathbf{x})$ for the multi-stage, high-volume machining problem.

As an example of the behavior of the evaluation framework, particularly with respect to externally imposed flexibility requirements, we evaluate the efficiency (Eq. 4-9) of three different manufacturing systems for three different flexibility requirement scenarios. The

manufacturing systems to be evaluated are as follows (Fig. 4-6):

System A. Totally Dedicated System.

- Each of 27 operation groups processed by batch operation machines (BOMs), each hardware-dedicated to the operations that it performs.

System B. Hybrid System.

- First 17 operation groups processed by dedicated BOMs, the next 10 by flexible (programmable) sequential operation machines (SOMs).

System C. Totally Flexible System.

- All 27 operation groups processed by flexible SOMs.

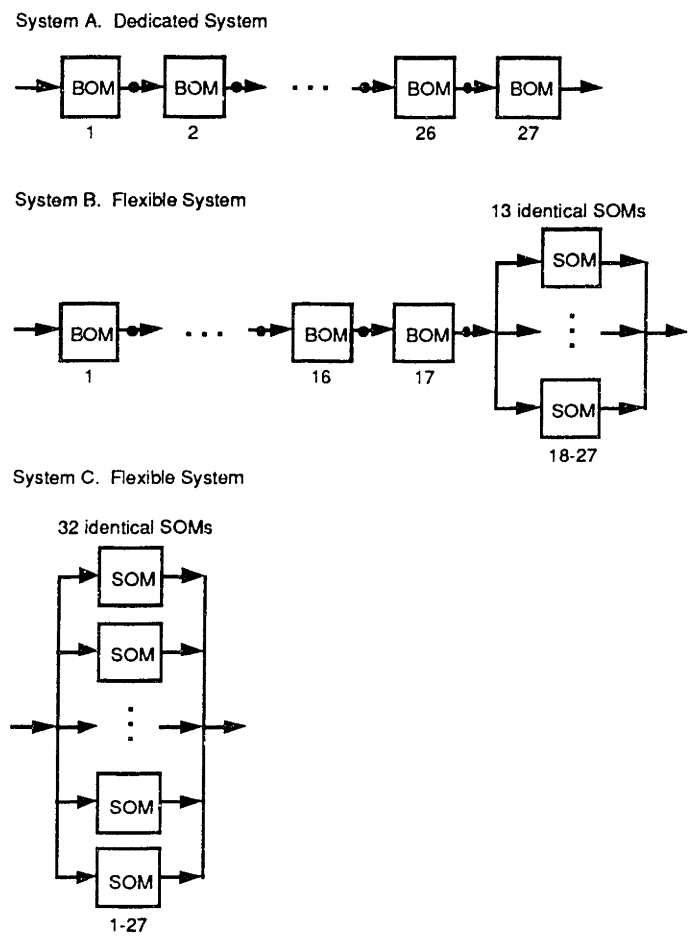


FIGURE 4-6. The manufacturing system configurations to be evaluated.

For the purpose of the example, we construct three scenarios, listed below in order of increasing flexibility requirement.

Scenario 1. Low Flexibility Requirement.

- Design change every 5 years.
- 55% of features are altered in each design change.
- Constant demand (120 parts/hour).

Scenario 2. Medium Flexibility Requirement.

- Design change every 3 years.
- 75% of features are altered in each design change.
- Constant demand (120 parts/hour).

Scenario 3. High Flexibility Requirement.

- Design change every year.
- 90% of features are altered in each design change.
- Constant demand (120 parts/hour).

The efficiencies of the three systems for the three scenarios are shown below (Fig. 4-7).

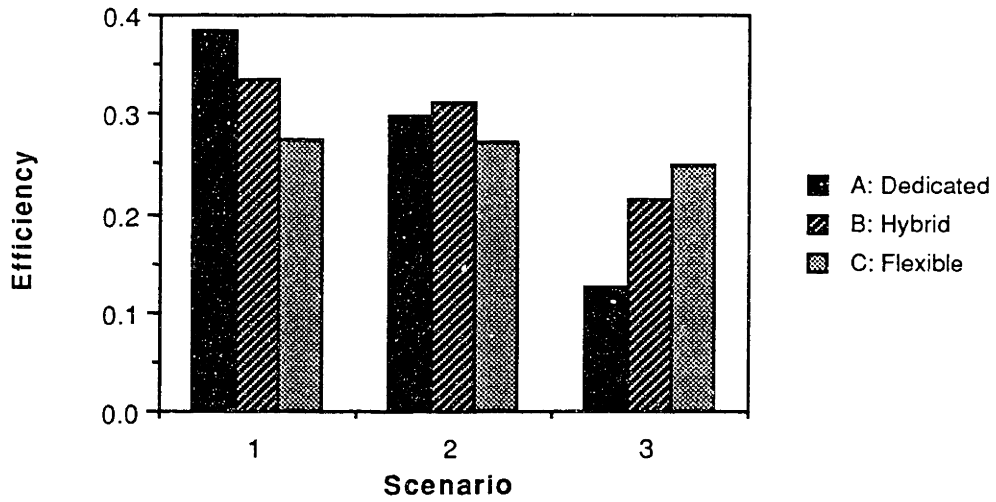


Figure 4-7. Efficiencies of dedicated, hybrid and flexible-hardware systems for increasing flexibility requirement.

When the flexibility requirement is low, the dedicated system is the best choice. When the flexibility requirement is high, the flexible system is the best choice. When the flexibility requirement is in between, the best choice may be a hybrid system. This example shows how the evaluation framework may be a useful tool for system selection given an assessment of the degree of flexibility required.

5. A Problem From Industry

The example problem addressed in this thesis contains $M = 2$ machine types. The first machine type is a dedicated transfer line station (TLS) of the batch operation (BOM) type. The second machine is a more flexible CNC machining center (CNC) of the sequential operation (SOM) type. The part to be processed is an automobile transmission case that contains 59 geometric features (e.g., holes, slots) requiring $N = 126$ operations for implementation (Fig. 5-1). The number of operations is approximately twice the number of geometric features because a feature may require more than one operation. For example, a feature such as a hole might require a drilling operation followed by a reaming operation. The operations have been grouped into $G = 27$ operation groups by process planners. The required production rate is 120 parts per hour (one part every 30 seconds), making the annual demand approximately 500,000 parts per year.

In accordance with the general evaluation framework defined earlier, the following $2M = 4$ decision variables may be used to define alternative manufacturing system designs.

x The number of operation groups implemented via CNC ($0 \le x \le 27$). The number of operation groups implemented via TLS is $27-x$.

Systems with many TLSs (low x) are likely to be preferred if the design of the part to be machined changes very infrequently, due to the lower acquisition cost of the TLS. On the other hand, if the part design changes frequently, then the lower change cost of the CNCs (consisting only of reprogramming and retooling costs) will tend to make systems with many CNCs (low x) preferable.

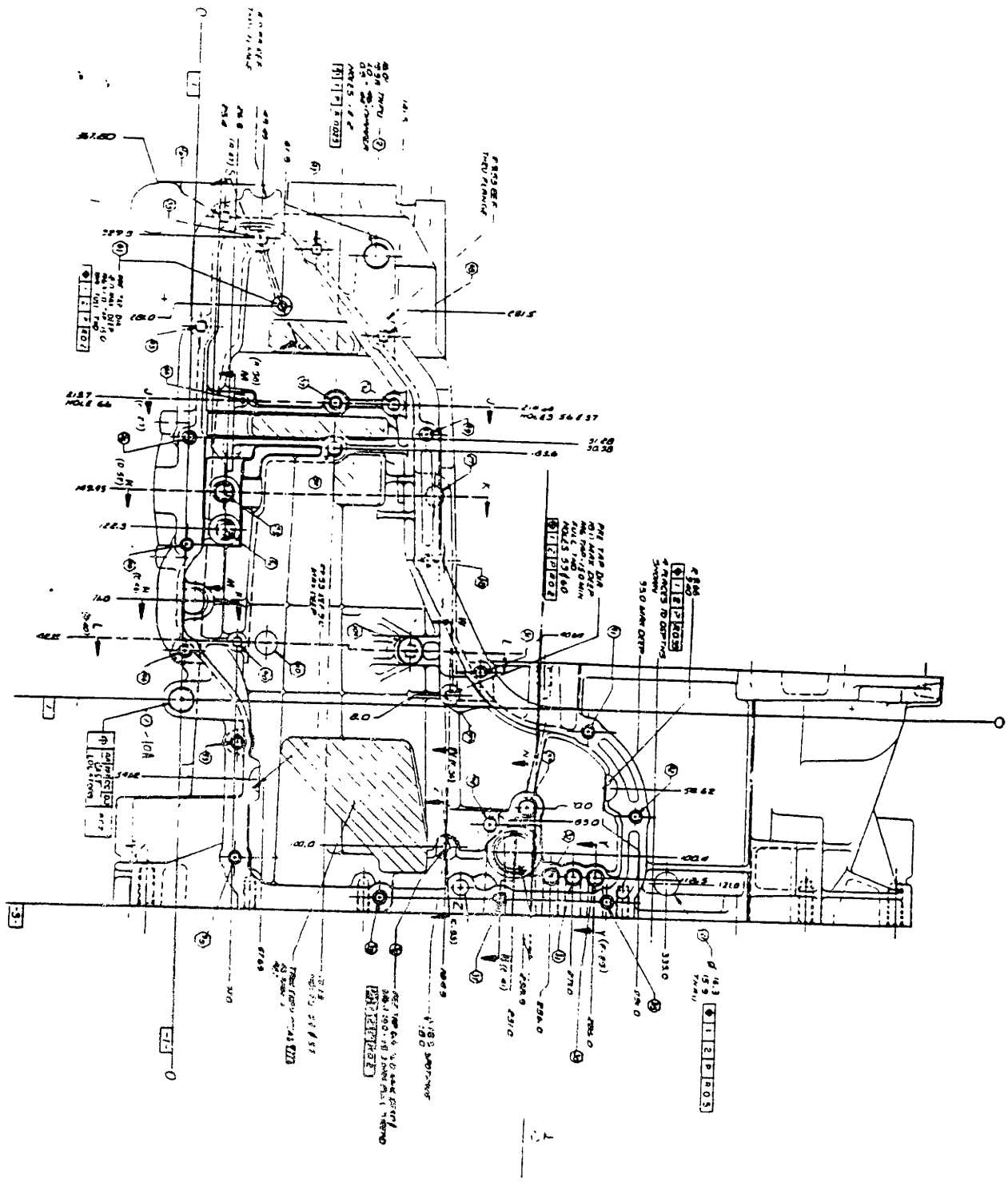


FIGURE 5-1. Part to be machined.

s Machine type sequence ($s \in \{0, 1\}$). $s = 0$ means that CNCs process the first x operation groups, and TLSs process the remaining $27-x$. $s = 1$ means that TLSs process the first $27-x$ operation groups, and CNCs process the remaining x.

b The capacity of inventory storage buffers in the system ($0 \leq b \leq 100$).

If b is too small, each machine breakdown will result in excessive blockage of upstream machines (since they will have no place to output their parts to) and excessive starvation of downstream machines (since they will have no place to receive their parts from). This reduces the system's production rate. If b is too large, then the inventory carrying costs of the system may to too high.

f Buffer frequency ($0 \leq f \leq 1$).

The processing sequence of the operations implemented via CNC is assumed to be given. This greatly reduces the size of the solution space for the example problem (Fig. 5-2). As shown, the solution space contains $28 \times 2 \times 101 \times 16 = 90{,}496$ solutions.

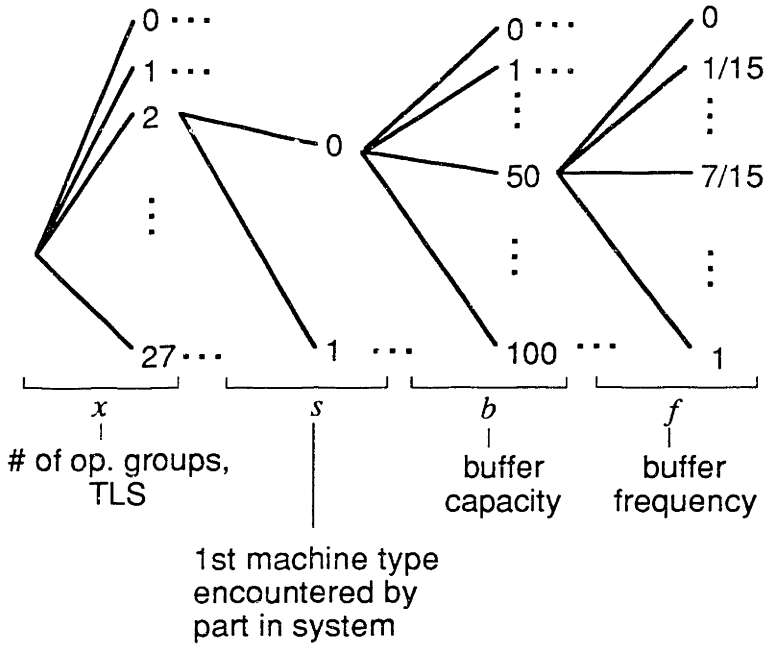


FIGURE 5-2. Solution space of the example problem.

In order to implement the evaluation framework values such as machine processing times, acquisition costs and labor rates are required. Table 5-1 shows the assumed processing times of the two machine types for each of the operation groups.

#	Operation Group Description	Proc Time TLS	CNC
1	Mill Diam F Face & ID Pad	25	7
2	Rough Mill Surface P	22	13
3	Finish Mill Surface P	25	0
4	Drill 2 Holes	24	33
5	Drill 7 Holes	25	61
6	Drill 10 Holes	25	68
7	Drill 4 Holes	23	32
8	Drill 6 Holes	24	42
9	Rough Bore 4 Holes	25	58
10	Finish Bore 7 Holes	25	82
11	Finish Bore 4 Holes	25	46
12	Mill Parking Pawl Slot	22	7
13	Drill 2 Holes	23	24
14	Drill 1 Hole	25	20
	Ream 1 Hole		
15	Drill 1 Hole	22	13
16	Spot Face 1 Hole	22	7
17	Burnish 1 Hole	24	13
18	Drill 1 Hole	24	9
19	Tap 1 Hole	25	7
20	Spot Face 4 Holes	23	27
	Drill 2 Holes		
21	Spot Face 4 Holes	25	63
	End Mill 5 Holes		
	Ream 3 Holes		
22	Drill 17 Holes	25	71
23	Bore 1 Hole	25	7
24	Bore 3 Holes	25	36
25	Tap 17 Holes	25	71
26	Tap 6 Holes	24	30
27	Tap 6 Holes	24	27

TABLE 5-1. Machine processing times for the example problem.

Other parameters and their assumed values are summarized in Table 5-2.

Name	Description	Value
M	Number of machine types	2
$MTBF_1$	Mean time between failures of TLS	10.0 [hours]
$MTBF_2$	Mean time between failures of CNC	6.0 [hours]
$MTTR_1$	Mean time to repair of TLS	0.167 [hours]
$MTTR_2$	Mean time to repair of CNC	0.500 [hours]
n_t	Number of periods in system life cycle	12
t	Duration of each period	1 [yr] = 4,136 [working hours]
$c_m(1)$	Acquisition cost, TLS	$220,000
$c_m(2)$	Acquisition cost, CNC	$400,000
c_l	Labor cost rate	$30/[hour]
$n_l(p)$	Average number of workers during period p	Number of buffers in the system
$PR(p)$	Demanded production rate for period p	120 [parts/hour]
c_i	Inventory carrying cost per part per period	$5,000
$c_p(1)$	Preventive maintenance cost of 1 TLS per period	$1,600
$c_p(2)$	Preventive maintenance cost of 1 CNC per period	$3,500
$c_r(i)$	Repair cost rate for machine type i	$50/[hour]
$P_c(j, k)$	Probability that the k^{th} feature worked on by the j^{th} machine will be modified in a design change	0.90
$c_c(1)$	Cost of modifying/replacing 1 TLS to accommodate a part design change	$220,000
$c_c(2)$	Cost of modifying/replacing 1 CNC to accommodate a part design change	$12,000
r	Interest rate per period	0.10
$P_\Delta(5)$	Probability that the number of periods between part design changes will be 5	1.0
$P_\Delta(t \neq 5)$	Probability that the number of periods between part design changes will be something other than 5	0.0

TABLE 5-2. Assumed parameter values for the example problem.

It is assumed that the extent of each design change is large: 90% of the geometric features in the part are altered with each design change. However, the design changes are relatively infrequent: once every 5 years.

The inventory cost per part per period, where a period equals one year, is set a high value of $5,000 in accordance with the following rationale. The reduction of inventory levels is

an industrial priority. High inventory level in a manufacturing system results in many costs beyond the opportunity cost of the material capital on the manufacturing floor (Hall *et al.* 1991):

- Inventory management costs include costs for the labor required to offload and reload a manufacturing system, as well as costs for tracking the processing state of floating inventory, so that it may be reloaded on to the manufacturing system at the appropriate point.
- Quality costs result from machining errors caused by parts being misaligned when reloaded on to a manufacturing system.
- Floor space costs.

Therefore, the inventory cost was set at a value which would make it significant in the evaluation of efficiency. At its chosen value, inventory cost represents up to about 10% of the total expected life cycle cost in the efficiency definition (Eq. 4-1).

6. The Nature of the Problem

In this section we perform a more in-depth analysis of the particular manufacturing system configuration problem that we seek to solve. We seek to determine the nature of the decision variables to performance index ($\mathbf{x} \rightarrow y$) mapping that both the forward and inverse formulations require.

6.1. Measures of Mapping Roughness

The "nature" of the decision variables to performance index mapping refers to the "bumpiness" of the efficiency function $y(\mathbf{x})$. One way of assessing this bumpiness is via a measure based on the values of the function's first derivative over the input space. A second way is via a measure based on the values of the function's second derivative over the input space. For the case of one decision variable, these roughness measures can be formulated as:

$$R_1 = \int \left[y'(x)\right]^2 dx \tag{6-1}$$

$$R_2 = \int \left[y''(x)\right]^2 dx \tag{6-2}$$

These roughness measures were suggested by Scott (1992) for use in the estimation of density functions of random variables.

The efficiency is a function of the four decision variables that represent each configuration $\mathbf{x}$, and is a hypersurface in 5-D space. Therefore multivariate versions of the roughness measures R_1 and R_2 are required. If we let the decision variables be $x_1, x_2, \ldots, x_n$, then the multivariate version of R_1 may be written as.

$$R_1 = \int \left(\nabla y(\mathbf{x})\right)^{\mathrm{T}} \left(\nabla y(\mathbf{x})\right) d\mathbf{x} \tag{6-3}$$

where:

$\mathbf{x} \equiv [x_1 \; x_2 \; \ldots \; x_n]^{\mathrm{T}}$

$d\mathbf{x} \equiv dx_1 dx_2 \ldots dx_n$

$\nabla y(\mathbf{x}) \equiv$ gradient of $y = [\partial y/\partial x_1 \;\; \partial y/\partial x_2 \;\; \ldots \;\; \partial y/\partial x_n]^{\mathrm{T}}$

For the problem at hand, $n = 4$, with $x_1 = x$, $x_2 = s$, $x_3 = b$ and $x_4 = f$.

The multivariate version of R_2 uses the Hessian matrix of second derivatives:

$$\mathbf{H} = \begin{bmatrix} \dfrac{\partial^2 y}{\partial x_1^2} & \dfrac{\partial^2 y}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 y}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 y}{\partial x_2 \partial x_1} & \dfrac{\partial^2 y}{\partial x_2^2} & \cdots & \dfrac{\partial^2 y}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \dfrac{\partial^2 y}{\partial x_n \partial x_1} & \dfrac{\partial^2 y}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 y}{\partial x^2} \end{bmatrix} \tag{6-4}$$

It is given by the expression

$$\begin{aligned} R_2 &= \int \left(\sum_{i=1}^{n} \sum_{j=1}^{n} h_{ij}^2 \right) d\mathbf{x} \\ &= \int trace\left(\mathbf{H}^T \mathbf{H}\right) d\mathbf{x} \end{aligned} \tag{6-5}$$

where:

h_{ij} ≡ element in the i^{th} row, j^{th} column of the Hessian matrix **H**

$d\mathbf{x}$ ≡ $dx_1 dx_2 \ldots dx_n$

trace(**A**) ≡ sum of the elements of matrix **A** on the main diagonal

6.2. Behavior of Mapping Roughness

The behavior of the above mapping roughness measures may be best illustrated by applying them to known functions which can be easily visualized. For this purpose, the function

$$y = a \sin(2\pi b x_1) \sin(2\pi b x_2), (0 \le x_1 \le 1, 0 \le x_2 \le 1) \qquad (6\text{-}6)$$

will be used because it can be easily visualized as a surface in 3-D space, and consists of a regular array of bumps, whose amplitudes are given by the parameter a and whose frequencies are given by the parameter b. Roughness measures for various values of a and b are shown below (Fig. 6-1). The domain of these functions is given by ($0 \le x_1 \le 1, 0 \le x_2 \le 1$).

FIGURE 6-1. Roughness measures for functions of varying bump amplitude a and bump frequency b.

Analytical evaluation of the roughness measure R_1 for the given function (Eq. 6-6), over the domain ($0 \le x_1 \le 1, 0 \le x_2 \le 1$), yields:

$$
\begin{aligned}
R_1 &= \int \left(\nabla y(\mathbf{x})\right)^T \left(\nabla y(\mathbf{x})\right) d\mathbf{x} \\
&= \int_0^1 \int_0^1 \left[\left(\frac{\partial y}{\partial x_1}\right)^2 + \left(\frac{\partial y}{\partial x_2}\right)^2 \right] dx_1 dx_2 \\
&= \int_0^1 \int_0^1 \left[\begin{array}{l} \left(2\pi ab\, cos\left(2\pi b x_1\right) sin\left(2\pi b x_2\right)\right)^2 \\ + \left(2\pi ab\, sin\left(2\pi b x_1\right) cos\left(2\pi b x_2\right)\right)^2 \end{array} \right] dx_1 dx_2 \\
&= 2\pi^2 a^2 b^2 \left(1 - \frac{sin^2\left(4\pi b\right)}{16\pi^2 b^2}\right)
\end{aligned}
\tag{6-7}
$$

Analytical evaluation of the roughness measure R_2 for the given function (Eq. 6-6), over the domain ($0 \le x_1 \le 1, 0 \le x_2 \le 1$), yields:

$$
\begin{aligned}
R_2 &= \int trace\left(\mathbf{H}^T\mathbf{H}\right) d\mathbf{x} \\
&= \int \left(\sum_{i=1}^{n} \sum_{j=1}^{n} h_{ij}^2 \right) d\mathbf{x} \\
&= \int_0^1 \int_0^1 \left[\left(\frac{\partial y}{\partial x_1^2} \right)^2 + \left(\frac{\partial y}{\partial x_1 \partial x_2} \right)^2 + \left(\frac{\partial y}{\partial x_2 \partial x_1} \right)^2 + \left(\frac{\partial y}{\partial x_2^2} \right)^2 \right] dx_1 dx_2 \\
&= \int_0^1 \int_0^1 \left[\begin{array}{l} \left(-4\pi^2 ab^2 \sin\left(2\pi b x_1\right) \sin\left(2\pi b x_2\right)\right)^2 \\ + \left(4\pi^2 ab^2 \cos\left(2\pi b x_1\right) \cos\left(2\pi b x_2\right)\right)^2 \\ + \left(4\pi^2 ab^2 \cos\left(2\pi b x_1\right) \cos\left(2\pi b x_2\right)\right)^2 \\ + \left(-4\pi^2 ab^2 \sin\left(2\pi b x_1\right) \sin\left(2\pi b x_2\right)\right)^2 \end{array} \right] dx_1 dx_2 \\
&= 16\pi^2 a^2 b^4 \left(1 + \frac{\sin^2\left(4\pi b\right)}{16\pi^2 b^2} \right)
\end{aligned}
\tag{6-8}
$$

Contour plots of R_1 and R_2 are shown below (Fig. 6-2). R_1 is an increasing function of the amplitude a and the frequency b. For $b^2 \gg 1/16\pi^2$, the sine term of R_1 becomes insignificant (Eq. 6-7), and the increase in R_1 per unit increase in the amplitude a is the same as the increase in R_1 per unit increase in the frequency b. R_2 is also an increasing function of a and b. For $b^2 \gg 1/16\pi^2$, the sine term of R_2 becomes insignificant (Eq. 6-8), and the increase in R_2 per unit increase in the amplitude a is much smaller than the increase in R_2 per unit increase in the amplitude b. Therefore b is the dominant factor in determining the value of R_2. The greater the height of the bumps and the closer together the bumps are, the greater the measures R_1 and R_2 become. R_2 is more sensitive to bump frequency than R_1.

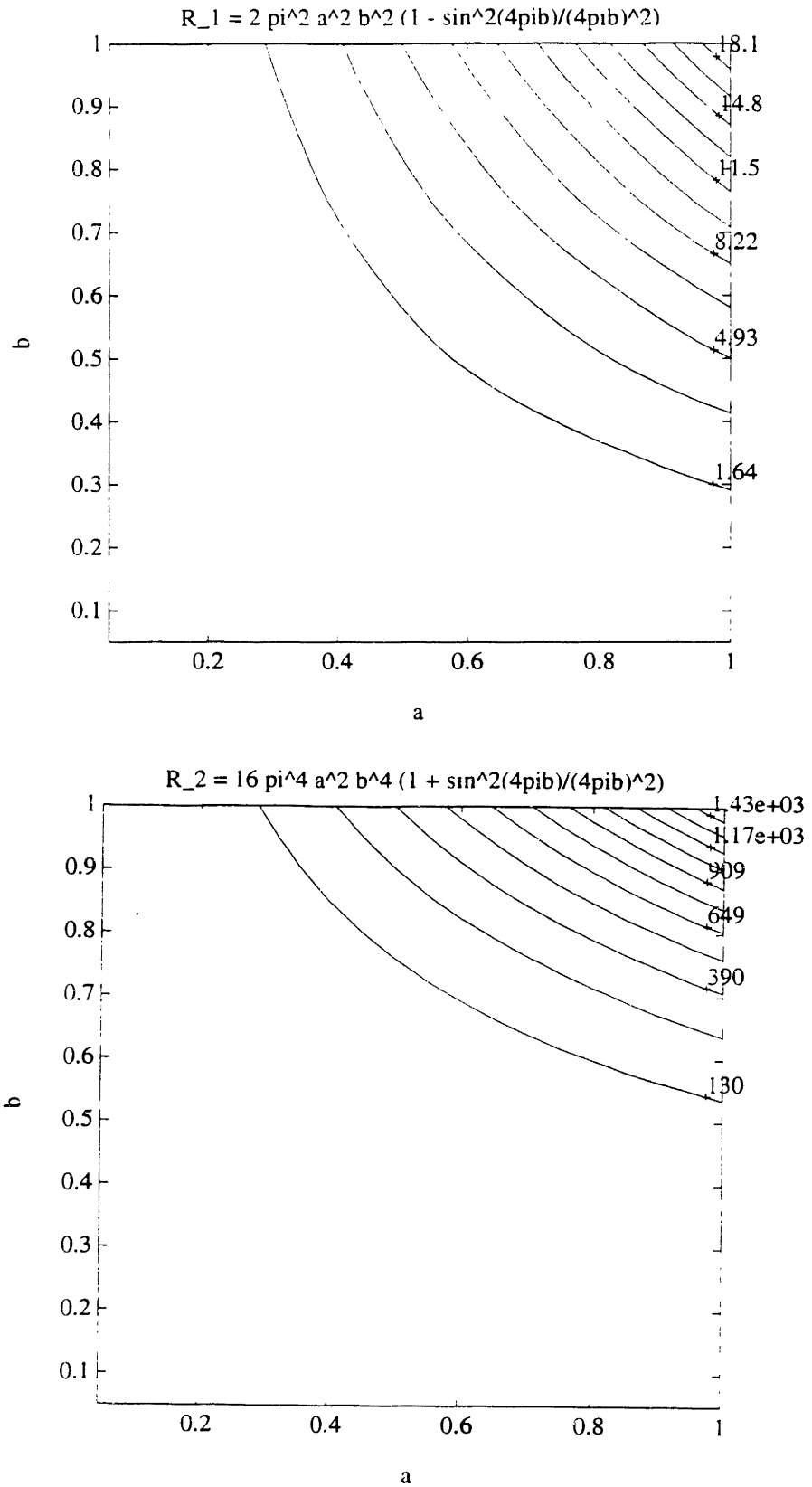


FIGURE 6-2. Contour plots of the roughness measures R_1 and R_2.

The above roughness measure calculations assume two decision variables. However, the manufacturing system design problem discussed in Chapter 5 has four decision variables. Therefore, the behavior of the roughness measures with respect to the number of decision variables n is relevant. If we generalize the reference function (Eq. 6-6) to n decision variables,

$$y_n = a\sum_{i=1}^{n} sin\left(2\pi b x_i\right),\ (0 \le x_i \le 1) \tag{6-9}$$

and apply the definitions of R_1 (Eq. 6-3) and R_2 (Eq. 6-5), the following roughness measure expressions may be derived:

$$R_1(y_1) = 2\pi^2 a^2 b^2\left(1 + \frac{sin\,(4\pi b)}{4\pi b}\right) \tag{6-10}$$

$$R_1(y_2) = 2\pi^2 a^2 b^2\left(1 - \frac{sin^2\,(4\pi b)}{16\pi^2 b^2}\right) \tag{6-11}$$

$$R_1(y_3) = \frac{3}{2}\pi^2 a^2 b^2\left(1 - \frac{sin\,(4\pi b)}{4\pi b} - \frac{sin^2\,(4\pi b)}{16\pi^2 b^2} + \frac{sin^3\,(4\pi b)}{64\pi^3 b^3}\right) \tag{6-12}$$

$$R_1(y_4) = \pi^2 a^2 b^2\left(1 - \frac{sin\,(4\pi b)}{2\pi b} + \frac{sin^3\,(4\pi b)}{32\pi^3 b^3} - \frac{sin^4\,(4\pi b)}{256\pi^4 b^4}\right) \tag{6-13}$$

$$R_2(y_1) = 8\pi^4 a^2 b^4\left(1 - \frac{sin\,(4\pi b)}{4\pi b}\right) \tag{6-14}$$

$$R_2(y_2) = 16\pi^4 a^2 b^4\left(1 + \frac{sin^2\,(4\pi b)}{16\pi^2 b^2}\right) \tag{6-15}$$

$$R_2(y_3) = 18\pi^4 a^2 b^4\left(1 - \frac{sin\,(4\pi b)}{12\pi b} + \frac{sin^2\,(4\pi b)}{48\pi^2 b^2} - \frac{sin^3\,(4\pi b)}{64\pi^3 b^3}\right) \tag{6-16}$$

$$R_2(y_4) = 16\pi^4 a^2 b^4\left(1 - \frac{sin\,(4\pi b)}{4\pi b} - \frac{sin^3\,(4\pi b)}{64\pi^3 b^3} + \frac{sin^4\,(4\pi b)}{256\pi^4 b^4}\right) \tag{6-17}$$

R_1 and R_2 are plotted versus n for various values of a and b in Figures 6-3 and 6-4 respectively.

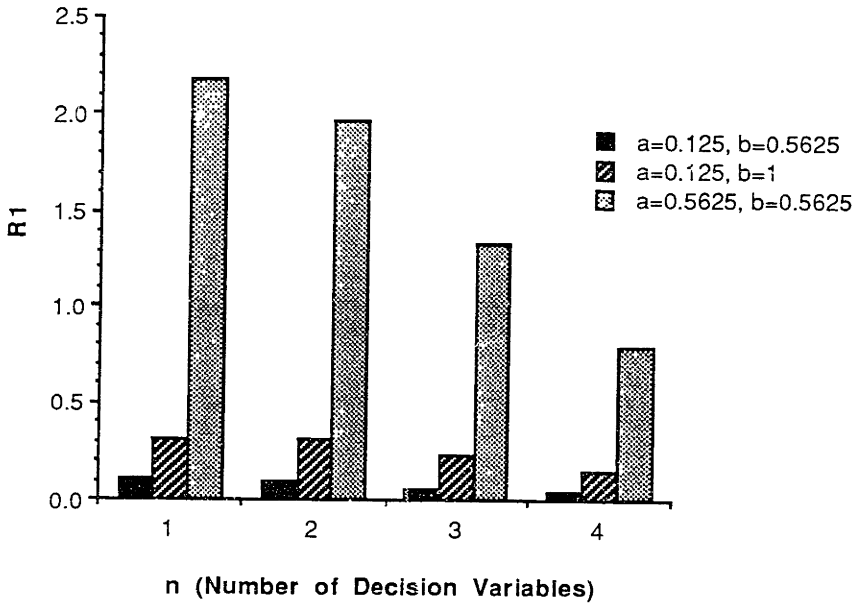


FIGURE 6-3. Roughness measure R_1 versus number of decision variables n for the reference function.

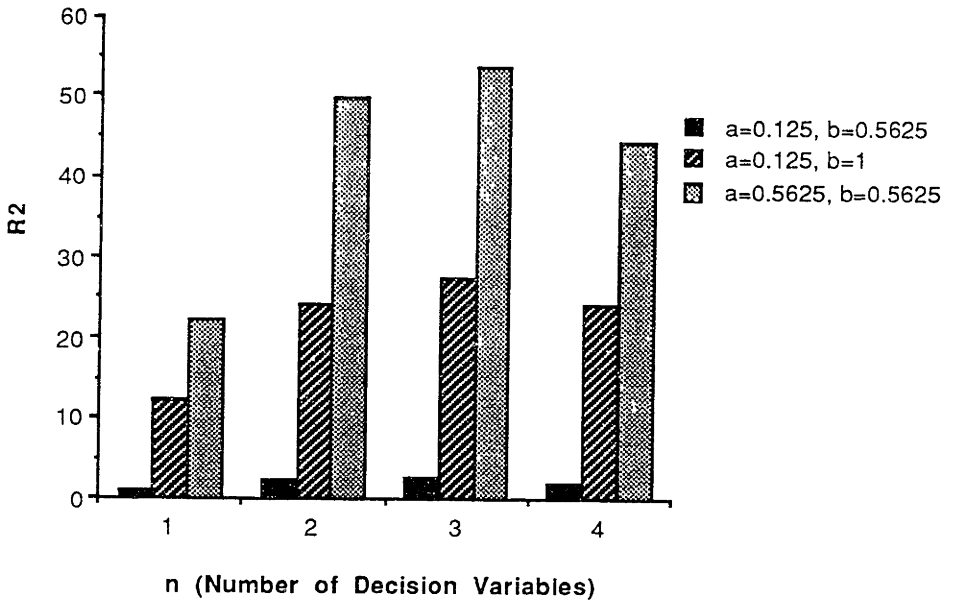


FIGURE 6-4. Roughness measure R_2 versus number of decision variables n for the reference function.

Roughness measure R_1 is relatively sensitive to n, the number of decision variables, and decreases with increasing n. In particular, $R_1(y_4)$ is about 1/2 the value of $R_1(y_2)$. Roughness measure R_2 is not very sensitive to the value of n, with $R_2(y_4)$ about equal to $R_2(y_2)$.

This means that although we will use 2-D surfaces in 3-D space (Fig. 6-1) to visualize the roughness of the decision variables to performance index mapping (a 4-D hypersurface in 5-D space), we should keep in mind that the 4-D hypersurface with the same R_1 and R_2 values as a given 2-D surface will in fact have more local extrema (greater frequency b) but less extreme extrema (smaller amplitude a) than the 2-D surface.

6.3. Evaluation of Mapping Roughness for the Problem from Industry

In order to assess the nature of the decision variables to efficiency mapping required to solve the manufacturing system design problem, roughness measures R_1 and R_2 can be assessed for this mapping. Since the functional form of this mapping is not known, the required derivatives will have to be estimated based on finite difference methods.

Two hundred and fifty manufacturing system configurations were simulated, and their corresponding efficiencies evaluated. The decision variable values of the systems simulated are shown below (Tbl. 6-1). These were normalized to fit into the range [0, 1] by dividing the x values by 27, the s values by 1, the b values by 100 and f values by 1 (Tbl. 6-2).

# Cfgs.	INPUT				OUTPUT
	x	s	b	f	y
250	$x_1 = 3$ $x_2 = 8$ $x_3 = 13$ $x_4 = 18$ $x_5 = 24$	$s_1 = 0$ $s_2 = 1$	$b_1 = 0$ $b_2 = 25$ $b_3 = 50$ $b_4 = 75$ $b_5 = 100$	$f_1 = 0$ $f_2 = 0.25$ $f_3 = 0.5$ $f_4 = 0.75$ $f_5 = 1$	

TABLE 6-1. Decision variable values of the simulated systems.

# Cfgs.	INPUT				OUTPUT
	$\hat{x} = \hat{x}_1$	$\hat{s} = \hat{x}_2$	$\hat{b} = \hat{x}_3$	$\hat{f} = \hat{x}_4$	y
250	$\hat{x}_1 = 0.11$ $\hat{x}_2 = 0.30$ $\hat{x}_3 = 0.48$ $\hat{x}_4 = 0.67$ $\hat{x}_5 = 0.89$	$\hat{s}_1 = 0.00$ $\hat{s}_2 = 1.00$	$\hat{b}_1 = 0.00$ $\hat{b}_2 = 0.25,$ $\hat{b}_3 = 0.50$ $\hat{b}_4 = 0.75$ $\hat{b}_5 = 1.00$	$\hat{f}_1 = 0.00$ $\hat{f}_2 = 0.25$ $\hat{f}_3 = 0.50$ $\hat{f}_4 = 0.75$ $\hat{f}_5 = 1.00$	

TABLE 6-2. Normalized decision variable values of the simulated systems.

Finite difference approximations such as the following were used in order to evaluate the derivatives in the roughness measure R_1:

$$\left.\frac{\partial y}{\partial x_1}\right|_{ijkl} = \frac{y|_{i+1,jkl} - y|_{ijkl}}{\hat{x}_1|_{i+1} - \hat{x}_1|_i} = y_{x_1}|_{ijkl} \tag{6-18}$$

where the subscripts i, j, k, l denote evaluation for $x = x_1$ at level i, $s = x_2$ at level j, $b = x_3$ at level k and $f = x_4$ at level l (Tbl. 6-2). Derivatives with respect to s, b and f were similarly estimated. R_1 was therefore evaluated as:

$$R_1 = \sum_{i=1}^{5}\sum_{j=1}^{2}\sum_{k=1}^{5}\sum_{l=1}^{5}\left\{\left[(y_{x_1}|_{ijkl})^2 + (y_{x_2}|_{ijkl})^2 + (y_{x_3}|_{ijkl})^2 + (y_{x_4}|_{ijkl})^2\right] \times (d\hat{x}_1|_i)(d\hat{x}_2|_j)(d\hat{x}_3|_k)(d\hat{x}_4|_l)\right\} \tag{6-19}$$

where:

$$d\hat{x}_1|_i \equiv (\hat{x}_1|_{i+1} - \hat{x}_1|_i)$$

Finite difference approximations such as the following were used in order to evaluate the second derivatives in the roughness measure R_2:

$$\left.\frac{\partial y}{\partial x_2 x_1}\right|_{ijkl} = \frac{y_{x_1}|_{i,j+1,kl} - y_{x_1}|_{ijkl}}{\hat{x}_2|_{j+1} - \hat{x}_2|_j} = y_{x_1 x_2}|_{ijkl} \tag{6-20}$$

Other second derivatives were similarly estimated. R_2 was therefore evaluated as:

$$R_2 = \sum_{i=1}^{5}\sum_{j=1}^{2}\sum_{k=1}^{5}\sum_{l=1}^{5}\left\{\left[\sum_{a=1}^{4}\sum_{b=1}^{4}(y_{x_a x_b}|_{ijkl})^2\right](d\hat{x}_1|_i)(d\hat{x}_2|_j)(d\hat{x}_3|_k)(d\hat{x}_4|_l)\right\} \tag{6-21}$$

Application of Equations 6-19 and 6-21 to the decision variables to efficiency mapping data (Tbl. 6-2) yielded the following results:

$$R_1 = 0.14 \tag{6-22}$$

$$R_2 = 4.17 \tag{6-23}$$

These values are similar to those for the reference function (Eq. 6-6) with an amplitude a of 0.2 and a frequency b of 0.6. This is one way of quantifying the nature of the decision variables to efficiency mapping. This mapping is about as rough as the function in the top row, second column from the left of Figure 6-1. It is fairly smooth. However, we must keep in mind, in accordance with the preceding discussion about the effect of the number of dimensions on the values of the roughness measures, that the mapping has more local extrema than would be implied by the picture in Figure 6-1. This will be significant later when results of the simulation plus hill-climbing method are discussed.

6.4. Relationship of Mapping Roughness to Forward Model Selection

The motivation for using neural networks to approximate the decision variables to efficiency mapping, as opposed to more standard empirical models such as linear regression models, is that they may better approximate mappings with roughness levels of the values seen for the decision variables to performance index mapping (Eqs. 6-22, 6-23) of the given industrial configuration problem. This can again be verified using the reference function (Eq. 6-6).

Three models types were compared: two linear regression models and a 2-25-15-1 neural network. The two regression models were:

$$f(\mathbf{x}) = a_0 + a_1 x_1 + a_2 x_2 \tag{6-24}$$

$$f(\mathbf{x}) = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2 \tag{6-25}$$

The three models were fit to the reference function (Eq. 6-6) sampled at the 121 points $\{x_1 = 0.0, 0.1, \ldots, 1.0; x_2 = 0.0, 0.1, \ldots, 1.0\}$. The models were then given 100 test inputs $\{x_1 = 0.05, 0.15, \ldots, 0.95; x_2 = 0.05, 0.15, \ldots, 0.95\}$ and compared on the basis of the absolute error

$$AE[t] = | y(\mathbf{x}^{[t]}) - f(\mathbf{x}^{[t]}) | \tag{6-26}$$

where:

$y(\mathbf{x}^{[t]})$ ≡ true value of the reference function for test input $\mathbf{x}^{[t]}$

$f(\mathbf{x}^{[t]})$ ≡ output of model for test input $\mathbf{x}^{[t]}$

The mean and standard deviation of this absolute error, over the 100 test inputs, are shown below for a reference function of amplitude $a = 0.25$ and frequency b ranging from 0.25 to 1 (Fig. 6-5).

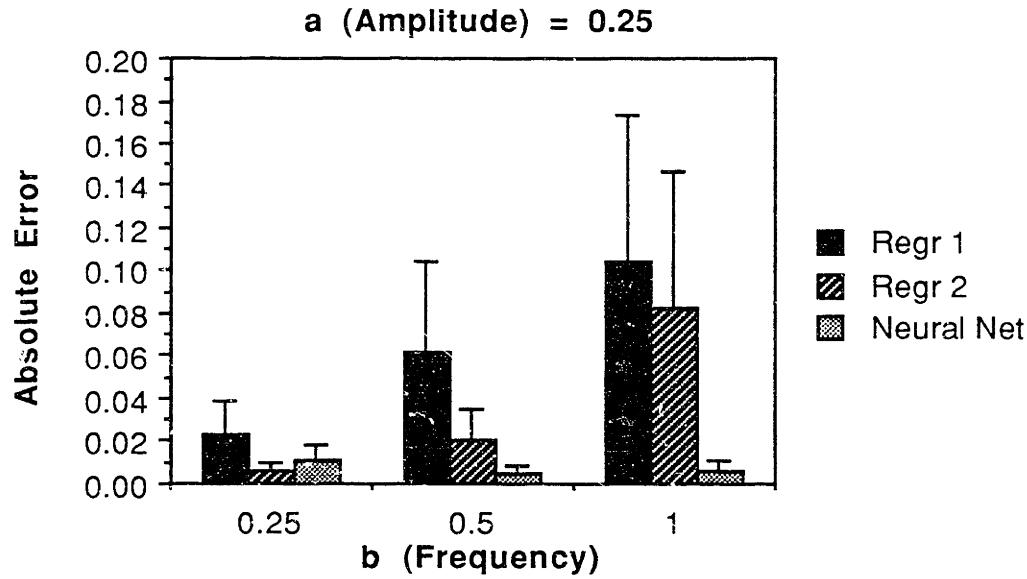


FIGURE 6-5. Mean absolute error for three empirical models fit to versions of the reference function with amplitude $a = 0.25$ and different frequencies b.

As the frequency (and hence roughness) of the reference function increases, the ratio of the neural network absolute error to that of the regression models shrinks. Furthermore, we have previously determined that the "roughness" of the given manufacturing system configuration problem is approximately equivalent to that of a reference function with $a = 0.2$, $b = 0.6$, a situation which is reflected in the second group of columns in Figure 6-5. This portion of the graph shows that the error of the neural network is much smaller than that of the regression models for a mapping of the same roughness as that of the given manufacturing system configuration problem from industry.

The results of this chapter quantify the nature of the mapping for which we seek to construct forward models. These results motivate the consideration of neural networks by showing that they perform well versus other empirical models in generalizing mappings of the nature of the given industrial problem.

7. Application of the Forward Approach

This chapter presents results of the application of an analytical model, simulation plus hill-climbing, linear regression models and neural network models in the forward approach to the manufacturing system configuration problem. The alternative model methods are summarized below.

7.1. Alternative Forward Model Methods

Model ANALYTICAL. Analytical Approximation

As described in Section 3.2, model ANALYTICAL provides crude estimates of a manufacturing system configuration's production rate and average WIP. The production rate is assumed to be the production rate of the slowest stage in the system. The average WIP is calculated under the assumption that the system is always filled to capacity with parts: all buffers are full all the time, and there is a part at each machine all the time. The above analytical approximations are input into the evaluation framework defined in Chapter 4 to obtain the estimated efficiency value. This model is used in conjunction with exhaustive enumeration of the space of configurations.

SIM+HC. Simulation Plus Hill-Climbing

This is the method described in Section 3.3.

Model REGR1. Linear Regression

The first linear regression model investigated in this thesis is a linear function of the decision variables x, s, b and f.

$$f(\mathbf{x}) = f(x, s, b, f) = a_0 + a_1 x + a_2 s + a_3 b + a_4 f \tag{7-1}$$

Model REGR2. Linear Regression

Model REGR2 contains some second-order decision variable terms, but is still a linear function of the coefficients and hence is still a linear regression model.

$$f(\mathbf{x}) = a_0 + a_1 x + a_2 s + a_3 b + a_4 f + a_5 xs + a_6 xb + a_7 xf + a_8 sb + a_9 sf + a_{10} bf \tag{7-2}$$

Model NET4-10-1. Neural Network

Model NET4-10-1 is a 3-layer neural network with 4 nodes in the input layer, 10 nodes in the hidden layer, and 1 node in the output layer.

7.2. Relative Accuracy of the Forward Models

In this section, the ability of the forward models of the previous section to approximate the decision variables to performance index mapping of the given manufacturing system configuration problem is described. Approximation ability is quantified via an accuracy measure called *fraction error* (E_f).

$$E_f = \frac{1}{T}\sum_{t=1}^{T}\left|\frac{f\left(\mathbf{x}^{[t]}\right) - y^{[t]}}{y^{[t]}}\right| \tag{7-3}$$

$\mathbf{x}^{[t]}$ ≡ a testing configuration different than any of the configurations simulated to fit the forward model $f(\mathbf{x})$

$y^{[t]}$ ≡ "true" value of the performance index for configuration $\mathbf{x}^{[t]}$, as evaluated via simulation

$f(\mathbf{x}^{[t]})$ ≡ value of the performance index for configuration $\mathbf{x}^{[t]}$, as estimated via the forward model $f(\mathbf{x})$

T ≡ total number of testing configurations

The smaller E_f is, the better the accuracy performance of the forward model.

7.2.1. Effect of Number of Simulation-Generated Training Samples

The choice of an analytical or empirical forward model for manufacturing system configuration may well depend on the number of training samples that can be made available for fitting an empirical model. In order to investigate this hypothesis, discrete-event simulation software was used to generate 9 sets of training samples, each set with a different number of samples. First the efficiencies of 250 configurations, namely all possible combinations of the 5 x values, 2 s values, 5 b values and 5 f values shown in Line 9 of Table 7-1, were evaluated via discrete-event simulation. The resulting 250 sets of efficiency values were then input to the evaluation framework defined in Chapter 4, resulting in 250 efficiency (y) values. Parameters such as machine acquisition cost, inventory carrying cost, etc., were set in accordance with Table 5-2. The set of 250 y values, combined with the set of 250 configuration vectors $\mathbf{x}$, formed the set of 250

training samples. Training sets of 16, 36, 54, 72, 96, 128, 160 and 200 training samples were then formed by taking subsets of the 250 training samples (Tbl. 7-1).

Training Set #	# Samples	INPUT				OUTPUT
		x	s	b	f	y
1	16	3, 13, 24	0, 1	0, 100	0, 1	
2	36	3, 13, 24	0, 1	0, 100	0, 0.5, 1	
3	54	3, 13, 24	0, 1	0, 50, 100	0, 0.5, 1	
4	72	3, 13, 18, 24	0, 1	0, 50, 100	0, 0.5, 1	
5	96	3, 13, 18, 24	0, 1	0, 50, 100	0, 0.25, 0.5, 1	
6	128	3, 13, 18, 24	0, 1	0, 25, 50, 100	0, 0.25, 0.5, 1	
7	160	3, 8, 13, 18, 24	0, 1	0, 25, 50, 100	0, 0.25, 0.5, 1	
8	200	3, 8, 13, 18, 24	0, 1	0, 25, 50, 100	0, 0.25, 0.5, 0.75, 1	
9	250	3, 8, 13, 18, 24	0, 1	0, 25, 50, 75, 100	0, 0.25, 0.5, 0.75, 1	

TABLE 7-1. Training sample sets for training sample quantity experiments.

$T = 32$ testing samples were similarly generated (Tbl. 7-2).

Testing Set #	# Samples	INPUT				OUTPUT
		x	s	b	f	y
1	32	6, 11, 16, 21	0, 1	13, 88	0.13, 0.88	

TABLE 7-2. Testing sample sets for training sample quantity experiments.

For each of the 9 training sets (16, 36, 54, 72, 96, 128, 160, 200 and 250 samples respectively), empirical models REGR1, REGR2 and NET4-10-1 were fit to the data. The decision variable values were normalized into the range [0, 1]. The neural networks were trained for 7,000 iterations via backpropagation (Rumelhart *et al.* 1986, Section 3.1.3), with a gain α of 0.05 and a momentum η of 0.80. The fraction error (Eq. 7-3) of the

forward models for different numbers of training samples was calculated and compared to that of model ANALYTICAL (Fig. 7-1).

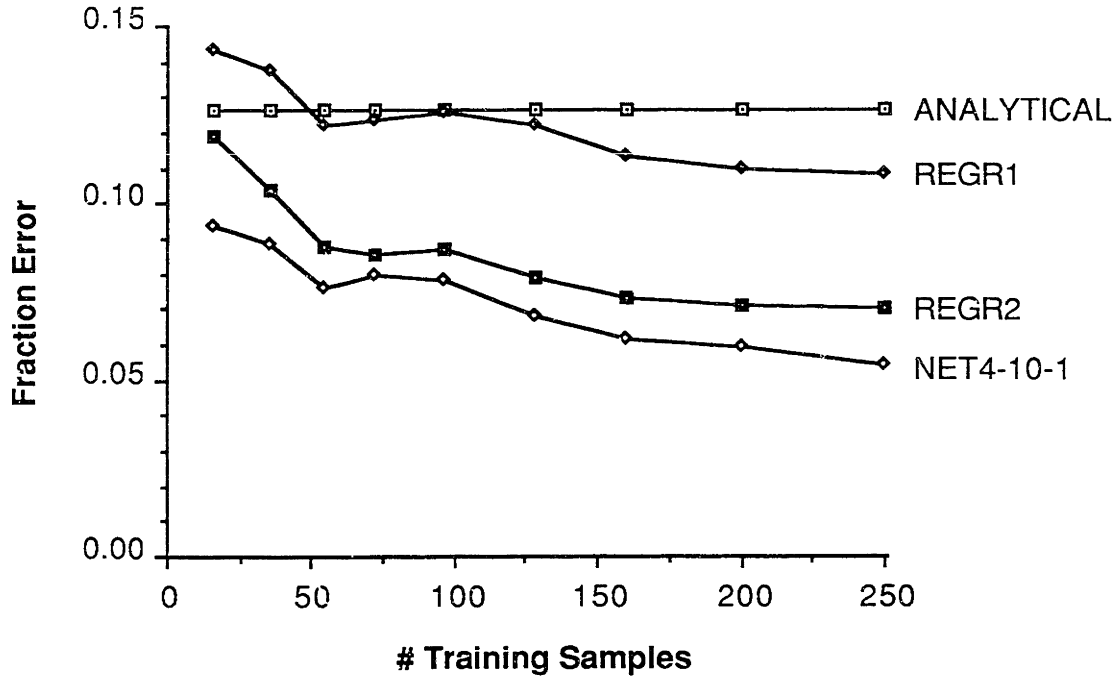


FIGURE 7-1. Fraction error of forward models for different numbers of training samples.

The empirical models outperform model ANALYT for all values of the number of training samples, with the exception of REGR1 at number of training samples ≤ 36. As expected, the accuracy of the empirical forward models, and hence their approximation capability with respect to the analytical model, improves with the number of training samples. Models that contain nonlinearities in the decision variables, REGR2 and NET4-10-1, are more accurate than the strictly linear REGR1 model. This clearly indicates the nonlinearity of the decision variables to efficiency mapping. Furthermore, NET4-10-1 is more accurate than REGR2, as would be expected from the mapping accuracy test performed on the sinusoidal surface with R_1 and R_2 roughness values comparable to that of the configuration to efficiency mapping (Section 6.4).

7.2.2. Effect of Mapping Roughness

One area in which an analytical forward model $f(\mathbf{x})$ such as model ANALYTICAL needs to make simplifying assumptions is in the calculation of the average WIP (e.g., Eq. 3-15). Such assumptions result in some model error $f(\mathbf{x}) - y(\mathbf{x})$, which increases as the inventory cost per part per year c_i increases. Therefore, as c_i increases, the assumptions in an analytical model become less viable, and its accuracy relative to empirical models fitted to simulation data should decrease.

In order to investigate this hypothesis, the 250 simulations of the previous section were used to generate 5 training sets, each containing 250 samples. The efficiency values of the 5 training sets were evaluated by setting the inventory carrying cost per part per year to \$1, \$10, \$100, \$1,000 and \$10,000 respectively. Otherwise, the efficiency parameters were set in accordance with Table 5-2. The training sets are summarized in Table 7-3.

Training Set #	c_i	# Samples	INPUT				OUTPUT
			x	s	b	f	y
1	\$1	250	3, 8, 13, 18, 24	0, 1	0, 25, 50, 75, 100	0.00, 0.25, 0.50, 0.75, 1.00	
2	\$10						
3	\$100						
4	\$1,000						
5	\$10,000						

TABLE 7-3. Training sample sets for WIP cost experiments.

Five sets of corresponding testing data (Tbl. 7-4) were similarly generated from the 32 simulations used to generate the testing data of Table 7-2.

Testing Set #	c_i	# Samples	INPUT				OUTPUT
			x	s	b	f	y
1	\$1	32	6, 11, 16, 21	0, 1	13, 88	0.13, 0.88	
2	\$10						
3	\$100						
4	\$1,000						
5	\$10,000						

TABLE 7-4. Testing sample sets for WIP cost experiments.

Empirical models REGR1, REGR2 and NET4-10-1 were fit to the data in each of the 5 training sets. The decision variable values were normalized into the range [0, 1]. The neural network model was trained for 7,000 iterations via backpropagation (Rumelhart *et al.* 1986, Section 3.1.3), with a gain α of 0.05 and a momentum η of 0.80. The fraction error (Eq. 7-3) of the empirical forward models for different values of the inventory carrying cost per part per year c_i, for the testing samples in Table 7-4, were calculated and compared to that of model ANALYTICAL (Fig. 7-2).

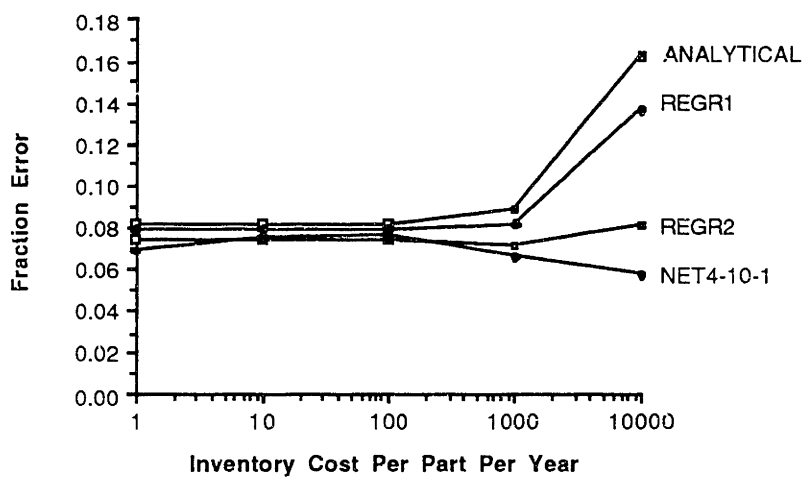


FIGURE 7-2. Fraction error of forward models for different values of inventory cost per part per year.

As expected, the accuracy of the analytical model relative to that of the empirical models degrades as c_i increases and makes the analytical model's assumptions less viable. In general, the most accurate model is NET4-10-1, followed by REGR2, REGR1, then ANALYTICAL. However, accuracy differences between the models are not significant until c_i increases beyond \$100. For solving the problem from industry (Chapter 5), c_i is set to \$5,000. At this value, model NET4-10-1 is a significantly more accurate approximation of the configuration **x** to efficiency y mapping than the other forward models investigated in this chapter.

We note that the roughness values R_1 and R_2 of the $\mathbf{x} \rightarrow y$ mapping, evaluated via the finite difference approximations of Section 6.3, increase sharply as c_i values increase above \$1,000 (Fig. 7-3). Thus the advantage in approximation accuracy held by the neural network over the other forward models increases as mapping roughness increases. This confirms a result that was observed with the product-of-sines reference function (Fig. 6-5).

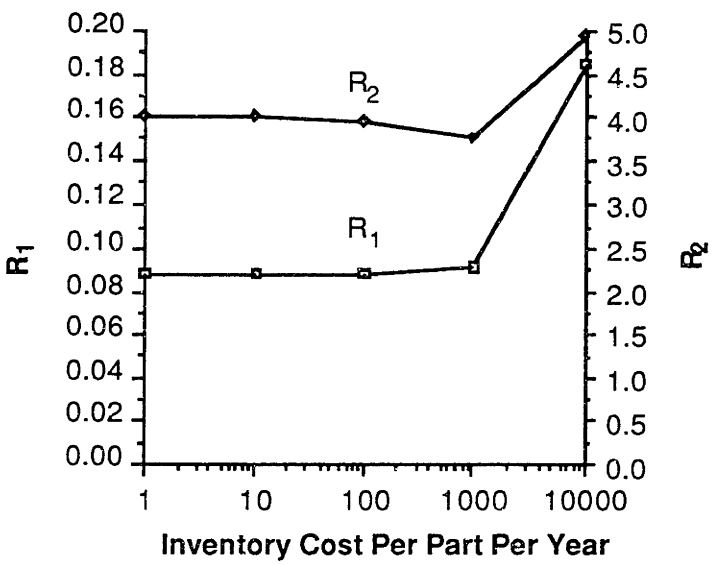


FIGURE 7-3. Roughness values of the configuration to efficiency mapping as a function of the inventory cost per part per year.

7.2.3. Effect of Simulation Accuracy

The sensitivity of the approximation capability of the forward models to simulation accuracy was also investigated. The efficiencies of 250 configurations (Line 9, Tbl. 7-1) were evaluated via simulation. The parameters used to calculate efficiency were those of Table 5-2. The calculated efficiencies were then corrupted by Gaussian noise with zero mean and standard deviation $\sigma = 0.05$, 0.10 and 0.15 respectively. Five corrupted training sets for each level of noise were generated. Models REGR1, REGR2, and NET4-10-1 were fit to each of the training sets, and then their accuracies in predicting the efficiencies of the testing configurations of Table 7-2 were evaluated via the fraction error (Eq. 7-3). The neural networks were trained for 7,000 iterations via backpropagation (Rumelhart *et al.* 1986, Section 3.1.3), with a gain α of 0.05 and a momentum η of 0.80. For each forward model, the fraction errors for each noise level were averaged (Fig. 7-4).

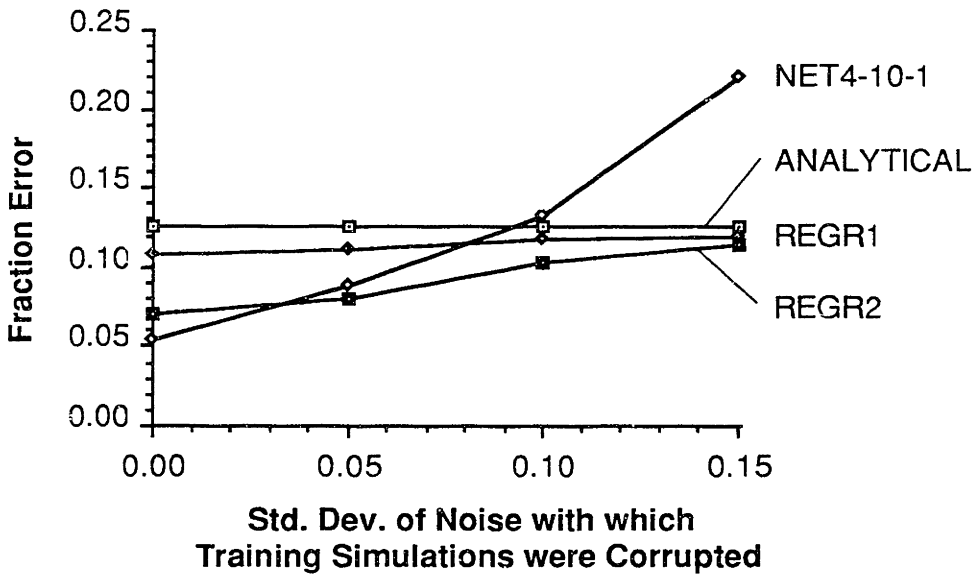


FIGURE 7-4. Fraction error of forward models for different simulation accuracies.

The rank of the models, in order of increasing sensitivity to simulation accuracy (as reflected by the slope of the lines in Figure 7-4), is: ANALYTICAL, REGR1, REGR2, then NET4-10-1. Model ANALYTICAL, of course, does not depend at all on simulation accuracy, while model NET4-10-1 is most sensitive to simulation accuracy. This behavior is predicted by the results of the statistical analysis presented of the next section, which describes the prediction confidence interval of an empirical model that is fit to simulation data with a given amount of inaccuracy or noise.

7.2.4. Confidence Intervals for Neural Network Predictions

The purpose of this section is to derive an estimate of a neural network's accuracy as a forward model based on the accuracy of the simulation models that it is trained with. Such an estimate may be used, in conjunction with similar estimates for alternative empirical models, to predict the range of training simulation accuracies over which neural network forward models are better approximations of the configuration to efficiency mapping than alternative empirical models.

7.2.4.1. Derivation of Confidence Intervals

The standard multilayer perceptron neural network is a nonlinear function $f(\mathbf{x}; \theta)$ of its inputs $\mathbf{x} = [x_1\ x_2\ \ldots\ x_n]^T$ and its parameters $\theta = [\theta_1\ \theta_2\ \ldots\ \theta_p]^T$.

We want to derive a confidence interval for y, the true value of the performance index for a given configuration $\mathbf{x}$. Assume that

$$y = f(\mathbf{x}; \theta^*) + \varepsilon_1 \tag{7-4}$$

$f(\mathbf{x}; \theta^*)$ ≡ neural network output for input $\mathbf{x}$ and "best" parameter values θ^*

ε_1 ≡ difference between true performance index value and the neural network output, arising from limitations in the functional form f. Assumed to be $N(0, \sigma_1^2)$.

For the manufacturing system configuration problem, the neural network must be trained with a set of samples generated via simulation:

Configurations	Observed Performance Index Values
$\mathbf{x}^{[1]}$	$y_{obs}^{[1]}$
$\mathbf{x}^{[2]}$	$y_{obs}^{[2]}$
⋮	⋮
$\mathbf{x}^{[S]}$	$y_{obs}^{[S]}$

Assume that

$$y_{obs}^{[i]} = y^{[i]} + \varepsilon_2 \quad (i = 1, 2, \ldots, S) \tag{7-5}$$

$y_{obs}^{[i]}$ ≡ simulation estimate of performance index for the i^{th} training sample

$y^{[i]}$ ≡ true value of performance index for the i^{th} training sample

ε_2 ≡ difference between simulation and true values arising from simulation inaccuracy. Assumed to be $N(0, \sigma_2^2)$.

Then, addition of Equations 7-4 and 7-5 yields

$$y_{obs}^{[i]} = f(\mathbf{x}^{[i]}; \theta^*) + \varepsilon_1 + \varepsilon_2 = f(\mathbf{x}^{[i]}; \theta^*) + \varepsilon \quad (i = 1, 2, \ldots, S) \tag{7-6}$$

ε $\equiv$ Difference between simulation and network output arising from both limitations in the functional form of f and simulation inaccuracy, $\sim N(0, \sigma_1^2 + \sigma_2^2) = N(0, \sigma^2)$.

Training the network via backpropagation yields a least squares estimate $\hat{\theta}$ of θ^*, for which the following linearization result is asymptotically true (Section 7.2.4.3):

$$\hat{\theta} \sim N_p(\theta^*, \sigma^2(\mathbf{F}.^{\mathrm{T}}\mathbf{F}.)^{-1}) \tag{7-7}$$

σ^2 $\equiv$ $\sigma_1^2 + \sigma_2^2$

$$\mathbf{F}. \equiv \begin{bmatrix} \frac{\partial f(\mathbf{x}^{[1]};\boldsymbol{\theta}^*)}{\partial \theta_1} & \frac{\partial f(\mathbf{x}^{[1]};\boldsymbol{\theta}^*)}{\partial \theta_2} & \cdots & \frac{\partial f(\mathbf{x}^{[1]};\boldsymbol{\theta}^*)}{\partial \theta_p} \\ \frac{\partial f(\mathbf{x}^{[2]};\boldsymbol{\theta}^*)}{\partial \theta_1} & \frac{\partial f(\mathbf{x}^{[2]};\boldsymbol{\theta}^*)}{\partial \theta_2} & \cdots & \frac{\partial f(\mathbf{x}^{[2]};\boldsymbol{\theta}^*)}{\partial \theta_p} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f(\mathbf{x}^{[S]};\boldsymbol{\theta}^*)}{\partial \theta} & \frac{\partial f(\mathbf{x}^{[S]};\boldsymbol{\theta}^*)}{\partial \theta} & \cdots & \frac{\partial f(\mathbf{x}^{[S]};\boldsymbol{\theta}^*)}{\partial \theta} \end{bmatrix}$$

We now proceed with the derivation of the confidence interval for y. Taylor expansion yields:

$$f(\mathbf{x}; \hat{\theta}) \approx f(\mathbf{x}; \theta^*) + \mathbf{f}.^{\mathrm{T}}(\hat{\theta} - \theta^*) \tag{7-8}$$

where

$$\mathbf{f}_\cdot^{\mathrm{T}} = \left[\frac{\partial f(\mathbf{x};\theta^*)}{\partial \theta_1} \; \frac{\partial f(\mathbf{x};\theta^*)}{\partial \theta_2} \; \cdots \; \frac{\partial f(\mathbf{x};\theta^*)}{\partial \theta_p} \right] \tag{7-9}$$

Hence

$$y - f(\mathbf{x};\, \hat{\theta}) \approx y - f(\mathbf{x};\, \theta^*) + \mathbf{f}_\cdot^{\mathrm{T}}(\hat{\theta} - \theta^*) = \varepsilon_1 - \mathbf{f}_\cdot^{\mathrm{T}}(\hat{\theta} - \theta^*) \tag{7-10}$$

From the statistical independence of ε_1 and $\hat{\theta}$,

$$E[y - f(\mathbf{x};\, \hat{\theta})] \approx E[\varepsilon_1] + \mathbf{f}_\cdot^{\mathrm{T}} E[\hat{\theta} - \theta^*] \approx 0 \tag{7-11}$$

$$\begin{aligned} \mathrm{var}[y - f(\mathbf{x};\, \hat{\theta})] &\approx \mathrm{var}[\varepsilon_1] + \mathrm{var}[\mathbf{f}_\cdot^{\mathrm{T}}(\hat{\theta} - \theta^*)] \\ &\approx \sigma_1^2 + \sigma^2 \mathbf{f}_\cdot^{\mathrm{T}}(\mathbf{F}_\cdot^{\mathrm{T}}\mathbf{F}_\cdot)^{-1}\mathbf{f} \\ &= \sigma_1^2 + (\sigma_1^2 + \sigma_2^2)\mathbf{f}_\cdot^{\mathrm{T}}(\mathbf{F}_\cdot^{\mathrm{T}}\mathbf{F}_\cdot)^{-1}\mathbf{f}_\cdot \end{aligned} \tag{7-12}$$

If detailed simulation models are used, simulation error and hence $\sigma_2^2 \approx 0$. In this case, an unbiased estimate of σ_1^2 is

$$s_1^2 = \frac{\left|\left| \mathbf{y}_{obs} - \mathbf{f}(\hat{\theta}) \right|\right|^2}{S - p} \tag{7-13}$$

S ≡ number of training samples

p ≡ number of parameters in the model f(x; θ)

$\mathbf{y}_{obs}$ ≡ $[y_{obs}^{[1]} \; y_{obs}^{[2]} \; \cdots \; y_{obs}^{[S]}]^T$

$\mathbf{f}(\hat{\theta})$ ≡ $[f(\mathbf{x}^{[1]};\, \hat{\theta}) \; f(\mathbf{x}^{[2]};\, \hat{\theta}) \; \cdots \; f(\mathbf{x}^{[S]};\, \hat{\theta})]^T$

Hence, asymptotically,

$$\frac{y - f(\mathbf{x};\hat{\boldsymbol{\theta}})}{\left(s_1^2 + \left(s_1^2 + \sigma_2^2\right)\mathbf{f}_\cdot^{\mathrm{T}}\left(\mathbf{F}_\cdot^{\mathrm{T}}\mathbf{F}_\cdot\right)^{-1}\mathbf{f}_\cdot\right)^{\frac{1}{2}}} \sim t_{S-p} \tag{7-14}$$

and an approximate $100(1 - \alpha)\%$ confidence interval for y is given by

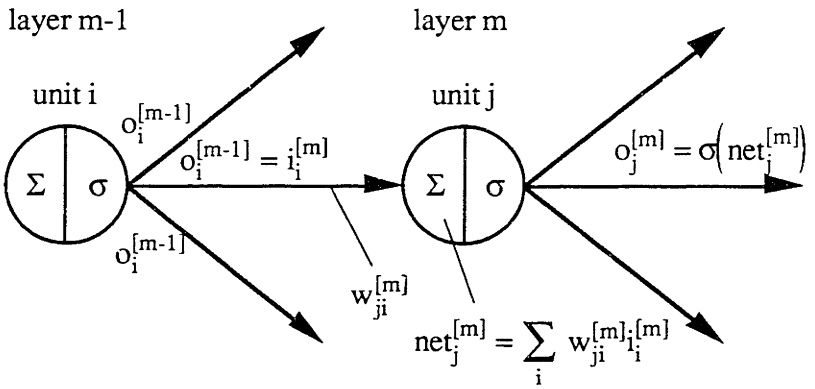
$$f(\mathbf{x};\hat{\boldsymbol{\theta}}) \pm t_{n-p}^{\alpha/2}\left(s_1^2 + \left(s_1^2 + \sigma_2^2\right)\mathbf{f}_\cdot^{\mathrm{T}}\left(\mathbf{F}_\cdot^{\mathrm{T}}\mathbf{F}_\cdot\right)^{-1}\mathbf{f}_\cdot\right)^{\frac{1}{2}} = f(\mathbf{x};\hat{\boldsymbol{\theta}}) \pm \Delta \tag{7-15}$$

We seek $\Delta/R << 1$, where R is the range of efficiency values encountered during network training.

7.2.4.2. Neural Network Implementation

In order to assess the above confidence interval for a neural network, we require evaluations of derivatives $\partial f/\partial\theta_j$, where f is the neural network output and θ_j is a network weight. Consider the network below (Fig. 7-5).

Nomenclature



Neural Network

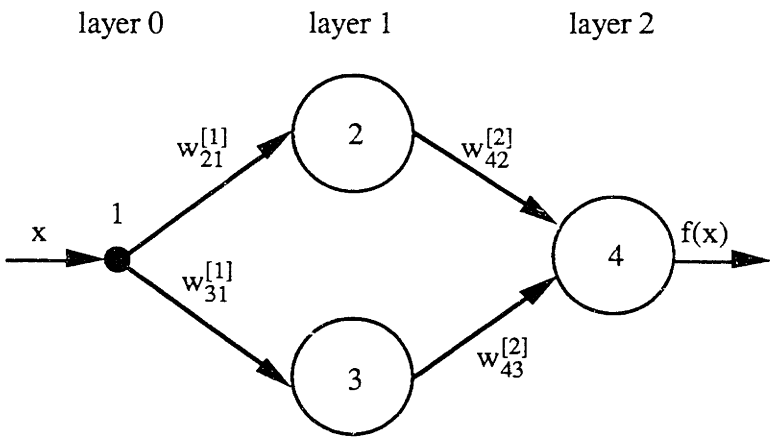


FIGURE 7-5. Sample neural network.

To calculate, for example, the derivative $\partial f/\partial w_{31}^{[1]}$, the steps are:

$$\begin{aligned}\frac{\partial f}{\partial w_{31}^{[1]}} &= \frac{\partial f}{\partial net_4^{[2]}}\left[\frac{\partial net_4^{[2]}}{\partial net_2^{[1]}}\frac{\partial net_2^{[1]}}{\partial w_{31}^{[1]}} + \frac{\partial net_4^{[2]}}{\partial net_3^{[1]}}\frac{\partial net_3^{[1]}}{\partial w_{31}^{[1]}}\right] \\ &= \frac{\partial f}{\partial net_4^{[2]}}\left[0 + \frac{\partial net_4^{[2]}}{\partial o_3^{[1]}}\frac{\partial o_3^{[1]}}{\partial net_3^{[1]}}\frac{\partial net_3^{[1]}}{\partial w_{31}^{[1]}}\right] \\ &= f[1-f]\left[w_{43}^{[2]}\, o_3^{[1]}\left(1-o_3^{[1]}\right)x\right]\end{aligned} \tag{7-16}$$

7.2.4.3. Distribution of the Weight Vector of a Trained Neural Network

In deriving the prediction confidence interval of a neural network (Section 7.2.4.1), the distribution of the weights $\hat{\theta}$ of a trained neural network was required (Eq. 7-7). This section derives this distribution, based on an analogous result for a linear model and a linearization of the nonlinear neural network model $f(\mathbf{x}; \hat{\theta})$.

Consider a linear function of a vector of decision variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. Let:

$$\mathbf{y} = \mathbf{X}\beta^* + \varepsilon, \varepsilon \sim N_S(\mathbf{0}, \sigma^2\mathbf{I}_S). \tag{7-17}$$

$\mathbf{y}$ $\equiv$ $S \times 1$ vector of S observed outputs

$\mathbf{X}$ $\equiv$ $S \times p$ matrix of S inputs (decision variable vectors); of rank p ($S > p$). In this instance, $p = n+1$.

β^* $\equiv$ $p \times 1$ vector of parameters

ε $\equiv$ $S \times 1$ vector of normal random variables with mean 0 and variance σ^2

The least squares estimate of β^* is $\hat{\beta}$, defined to be the vector β that minimizes

$$S(\beta) = \left|\left|\mathbf{y} - \mathbf{X}\beta\right|\right|^2, \tag{7-18}$$

has the property (Seber 1977)

$$\hat{\beta} \sim N_p\left(\beta^*, \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}\right). \tag{7-19}$$

The analogous nonlinear case begins with the following assumption:

$$\mathbf{y} = \mathbf{f}(\theta^*) + \varepsilon, \varepsilon \sim N_S(\mathbf{0}, \sigma^2 \mathbf{I}_S). \tag{7-20}$$

A least-squares estimate of θ^*, $\hat{\theta}$ (produced via backpropagation, for example), minimizes the squared error

$$S(\theta) = \left|\left|\mathbf{y} - \mathbf{f}(\theta)\right|\right|^2 \approx \left|\left|\mathbf{y} - \mathbf{f}(\theta^*) - \mathbf{F}.(\theta - \theta^*)\right|\right|^2, \tag{7-21}$$

or,

$$S(\beta) \approx \left|\left|\mathbf{z} - \mathbf{F}.\beta\right|\right|^2. \tag{7-22}$$

$$\beta \equiv \theta - \theta^*$$

$$\mathbf{z} \equiv \mathbf{y} - \mathbf{f}(\theta^*)$$

Since

$$\beta^* = \theta^* - \theta^* = \mathbf{0}, \tag{7-23}$$

we can restate Equation 7-20 as:

$$\mathbf{z} = \mathbf{F}.\beta^* + \varepsilon, \varepsilon \sim N_S(\mathbf{0}, \sigma^2 \mathbf{I}_S). \tag{7-24}$$

By analogy of Equations 7-24 and 7-22 with Equations 7-17 and 7-18 respectively, we see that

$$\hat{\beta} \sim N_p\left(\beta^*, \sigma^2 \left(\mathbf{F}.^{\mathrm{T}}\mathbf{F}.\right)^{-1}\right), \tag{7-25}$$

or,

$$\hat{\theta} \sim N_p\left(\theta^*, \sigma^2\left(\mathbf{F}.^{\mathrm{T}}\mathbf{F}.\right)^{-1}\right), \tag{7-26}$$

which is the desired expression for the distribution of the weights of a trained neural network.

7.2.4.4. Sensitivity of the Confidence Interval to Simulation Accuracy

We seek to assess the range of training simulation accuracies over which neural network models are preferable to alternative empirical models such as linear regression. The relative size of prediction confidence intervals over a range of simulation accuracies may be used as a gauge of when neural networks are preferable to alternative models: the smaller the confidence interval, the more preferable the model.

The efficiencies of 250 configurations (Line 9, Tbl. 7-1) were evaluated via simulation. The parameters used to calculate efficiency were those of Table 5-2. The calculated efficiencies were then corrupted by Gaussian noise with zero mean and standard deviation s = 0.05, 0.10 and 0.15 respectively. One corrupted training set was generated for each level of noise. Models REGR1, REGR2, and NET4-10-1 were fit to each of the training sets, and then their 90% prediction confidence intervals for the efficiencies of the 32 testing configurations of Table 7-2 were calculated via Equation 7-15. The neural networks were trained for 7,000 iterations via backpropagation (Rumelhart *et al.* 1986, Section 3.1.3), with a gain α of 0.05 and a momentum η of 0.80.

Figure 7-6 shows, for each level of simulation accuracy along the horizontal axis, the mean width of the 90% prediction confidence intervals over the 32 test configurations (Tbl. 7-2)

for each of the forward models. The error bars show one standard deviation of the confidence interval width, again evaluated over the 32 test configurations.

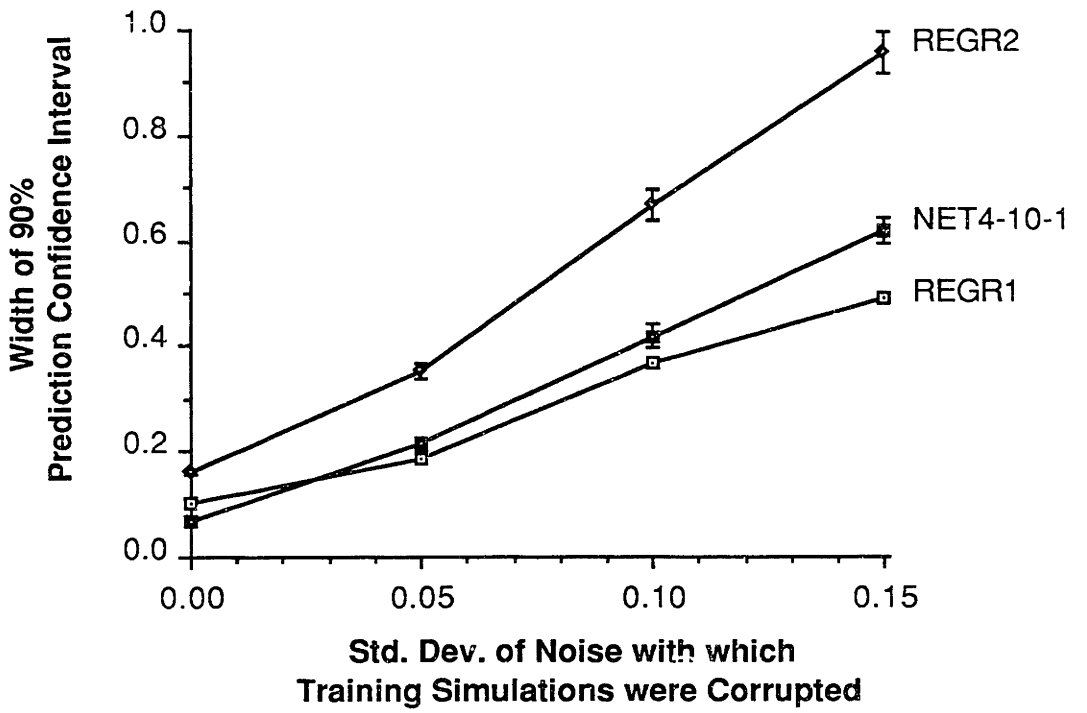


FIGURE 7-6. Width of 90% prediction confidence intervals of forward models for different simulation accuracies.

Consistent with the model approximation accuracy results of Section 7.2.3 (Fig. 7-4), model NET4-10-1 is preferable (has narrower confidence interval widths) than model REGR1 when the training simulations are accurate, but the preference becomes reversed as the simulation accuracy degrades. Although model REGR2 performs well relative to model REGR1 with respect to approximation accuracy (Fig. 7-4), its confidence interval widths are much worse (larger). There are two reasons for this. First, model REGR2 has more parameters than model REGR1, which increases the *t*-distribution value that the size of the confidence interval is proportional to. Second, the linearization assumptions of the confidence interval derivation are less accurate for model REGR2 (on account of its nonlinear decision variable terms, Eq. 3-2) than for model REGR1. Since the confidence

interval analysis predicts well the approximation accuracy performance of NET4-10-1 with respect to REGR1, the implication is that model NET4-10-1, for all its complex structure and large number of parameters, is more "linear" than model REGR2.

7.3. Achieved Efficiency of Configurations Prescribed by the Forward Models

Ultimately, the most relevant measure of the performance of a forward model $f(\mathbf{x})$ for manufacturing system configuration is the performance index (efficiency) value achieved by the configuration $\mathbf{x}*$ that is prescribed via its use. The higher this achieved efficiency, the better the model. In this section, the forward modeling methods described in Chapter 3 (Sections 3.2–3.5) are compared on this basis.

7.3.1. Effect of Number of Simulation-Generated Training Samples

One important questions is, for a given amount of input data in the form of samples of the configuration to efficiency mapping (generated via simulation), which forward model method will prescribe the configuration with the best efficiency? The forward model methods based on empirical models (REGR1 plus exhaustive enumeration, REGR2 plus exhaustive enumeration, NET4-10-1 plus exhaustive enumeration) were applied for different numbers of simulation training samples ranging from 16 to 250 (Tbl. 7-1). For comparison, simulation plus hill-climbing (SIM+HC) was applied over the same range of simulations, and model ANALYTICAL plus exhaustive enumeration was also used to prescribe a configuration (Fig. 7-7). Efficiency values were evaluated in accordance with the parameter values of Table 5-2.

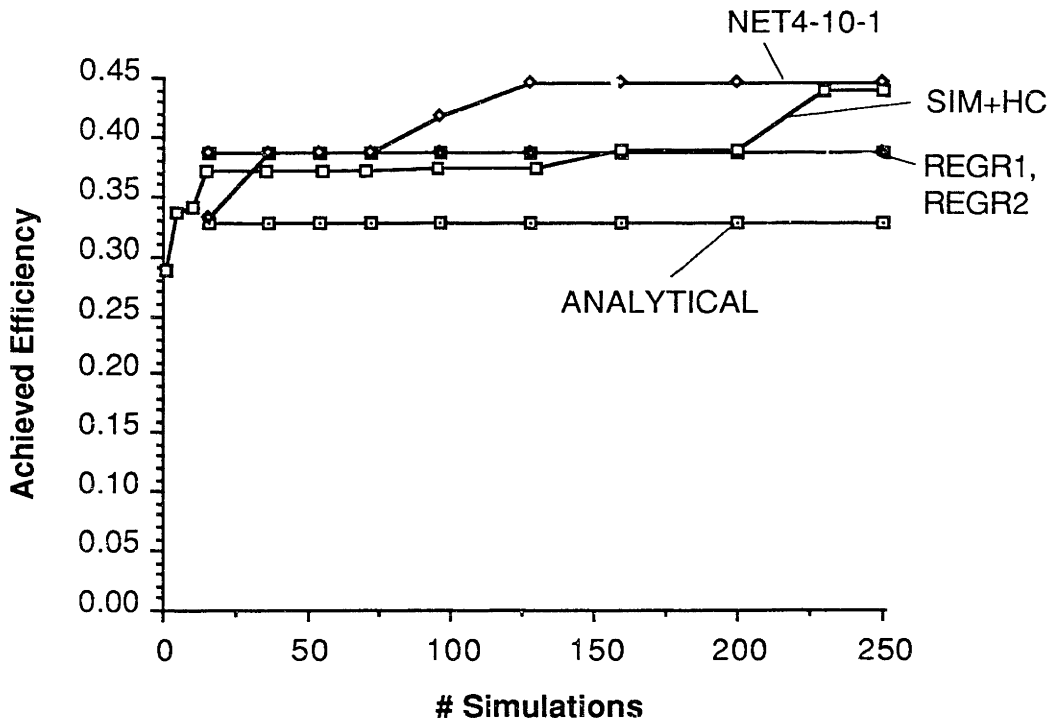


FIGURE 7-7. Achieved efficiency of different forward model methods versus the number of configuration-to-efficiency sample mappings provided via simulation.

Model ANALYTICAL prescribes the worst configurations. At lower amounts of training samples, the performance of configurations prescribed via NET4-10-1, SIM+HC, REGR1 and REGR2 are roughly the same. As the number of samples increases further, beyond about 100, the performance of configurations prescribed via NET4-10-1 is better than that achieved via other methods. Finally, as the number of samples increases even further, beyond about 225, the performance of SIM+HC catches up with that of NET4-10-1.

Even though the configuration to efficiency mapping was shown to be relatively smooth (Chapter 6), SIM+HC does not achieve the best efficiency with the least amount of computational effort (in the form of simulation modeling) because it often becomes stuck in local maxima. It was noted in Section 6.2 that the configuration to efficiency mapping, being a 4-D hypersurface in 5-D space, would have more local extrema than a 2-D surface

with the same roughness values (Fig. 6-1). This is borne out by the performance of the SIM+HC method.

An important observation is that there is a window of computational effort (in the form of simulation modeling) within which model NET4-10-1 prescribes better configurations than the other forward model methods.

7.3.2. Effect of Mapping Roughness

In Section 7.2.2, it was found that the efficiency prediction error of model NET4-10-1 becomes smaller relative to that of models ANALYTICAL, REGR1 and REGR2 as the inventory cost per part per year c_i increases, which corresponds to an increase in the roughness of the configuration to efficiency mapping. The same models (training samples summarized in Table 7-4) were used to prescribe configurations, and the resulting achieved efficiencies were recorded. Since efficiency values for different parameterizations (in this case, different values of c_i) are not directly comparable, the achieved efficiency values for each value of c_i were normalized by dividing by the achieved efficiency of model ANALYTICAL for that c_i value (Fig. 7-8).

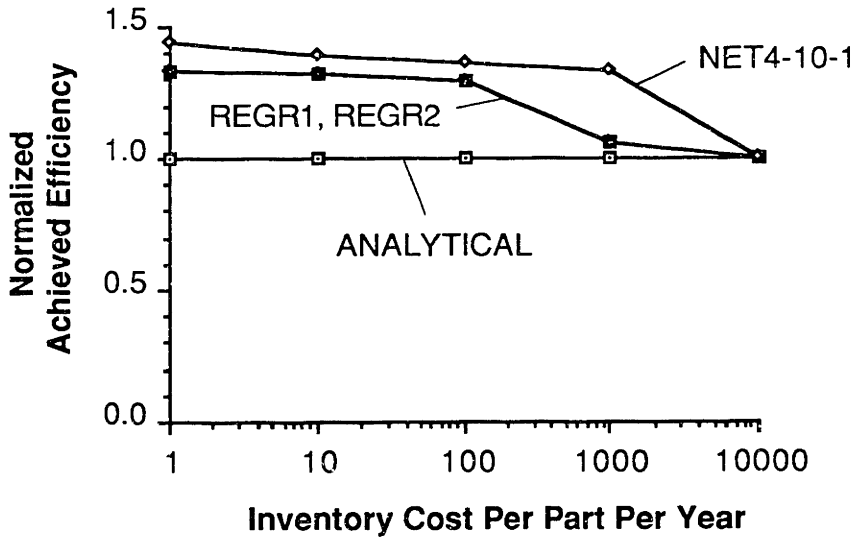


FIGURE 7-8. Achieved efficiency (normalized with respect to that of model ANALYTICAL) of different forward model methods versus the inventory cost.

The achieved efficiency of the empirical models relative to model ANALYTICAL decreases as c_i increases. Although the *accuracy* (Eq. 7-3) of model ANALYTICAL relative to that of the other forward models is worst at the highest c_i value of \$10,000 (Fig. 7-2), the *achieved efficiency* of the all the forward models is actually the same at this point. In fact, they all prescribe the same configuration: a totally flexible system consisting of one section of 32 identical CNC machines arranged in parallel, with no buffers (shown in the top portion of Figure 7-10). This configuration makes sense because it is the only one that combines the advantages of minimum WIP, which is very important at this high value of c_i, along with the ability to achieve the demanded production rate. The minimum WIP is achieved by the absence of buffers, while the required production rate is maintained because the starvation and blockage effects that reduce production rate cannot occur in such a parallel material flow configuration.

Model NET4-10-1 prescribes the best configurations across a wide range of inventory cost c_i, although its performance advantage is most pronounced in the region $\$100 \leq c_i \leq \$10{,}000$.

7.3.3. Effect of Simulation Accuracy

In Section 7.2.3, models REGR1, REGR2 and NET4-10-1 were fit to sets of 250 training samples (Line 9, Tbl. 7-1) corrupted by Gaussian, zero-mean noise. For each level of the noise standard deviation σ (0.05, 0.10, 0.15), five training sample sets were generated and used to fit each of the above model types. Five instances of each model type were thus created at each noise level. The mean achieved efficiency of the five model instances were then evaluated (Fig. 7-9).

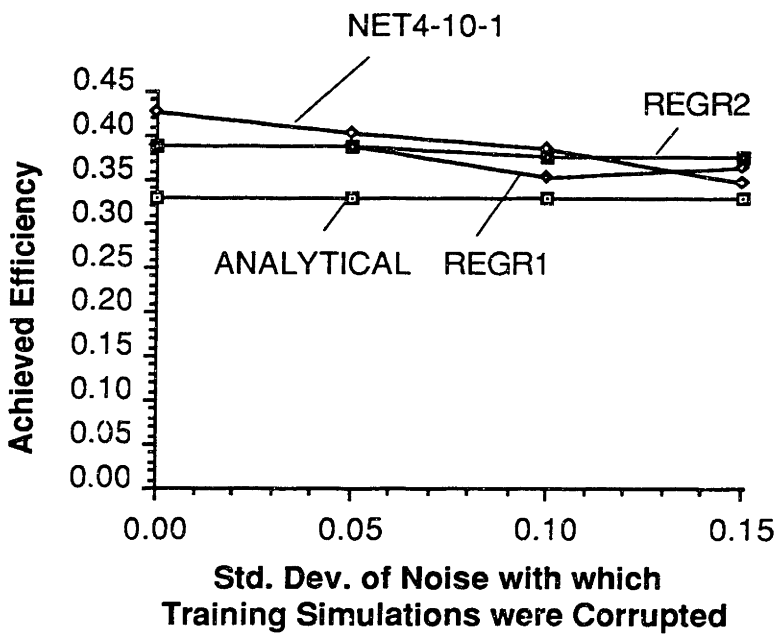


Figure 7-9. Achieved efficiency versus simulation accuracy.

As expected from the prediction accuracy experiments of Section 7.2.3 (Fig. 7-4), the achieved efficiency of model NET4-10-1 falls below that of the other empirical models as training simulation accuracy degrades.

7.4. How Prescribed Configurations Vary with the Required Flexibility

Model NET4-10-1 was trained with two sets of 250 training samples (Line 9, Tbl. 7-1). For the first training set, efficiency was evaluated assuming a large part design change frequency of once every two years. For the second training set, efficiency was evaluated assuming a low part design change frequency of once every five years. Other parameters values were set in accordance with Table 5-2. The trained models were then used in conjunction with exhaustive enumeration through the solution space to prescribe configurations (Fig. 7-10).

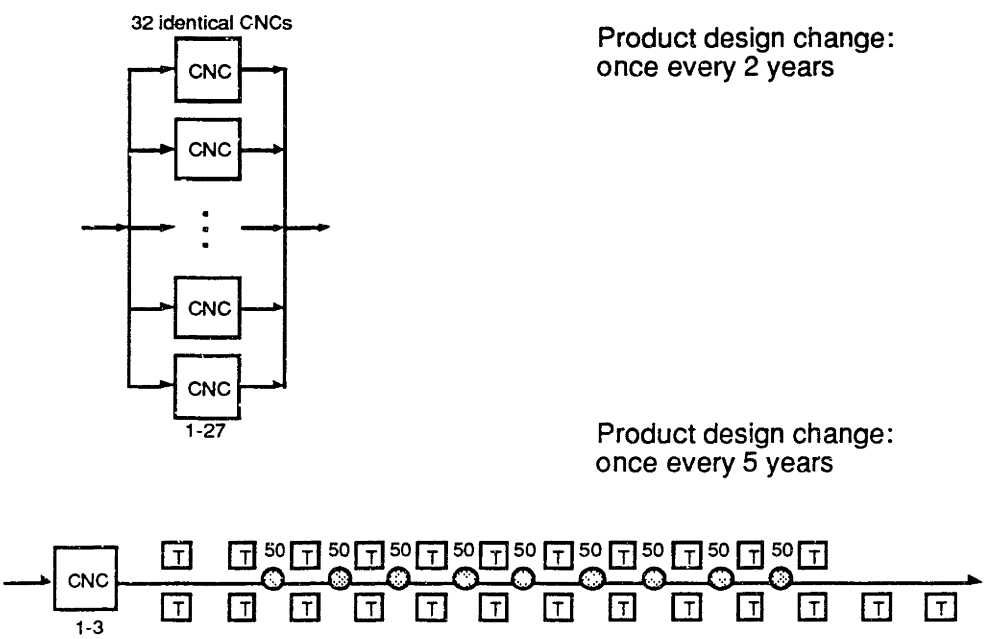


FIGURE 7-10. Configurations prescribed by a neural network for scenarios involving frequent and infrequent product design changes.

Given the likelihood of frequent product design changes, the neural network prescribes a configuration consisting solely of very flexible CNC machines. Given infrequent product design changes, the neural network prescribes a configuration dominated by dedicated transfer line stations that have lower acquisition cost. Thus the behavior of the neural network forward model is reasonable.

8. Application of the Inverse Approach

The direct inverse, distal supervised learning and distal supervised learning with constraints methods were applied to the high-volume machining system configuration problem defined in Chapter 5.

8.1. Direct Inverse Method

To evaluate the direct inverse method, 128 simulations were run to generate 128 training pairs in which efficiency values y were the inputs and decision variable values x, s, b, f were the outputs (Tbl. 8-1).

# Samples	INPUT y	OUTPUT			
		x	s	b	f
128		3, 10, 17, 24	0, 1	0, 50, 100, 150	0.25,0.50, 0.75, 1.00

TABLE 8-1. Training pairs for the direct inverse method.

Before being used for neural network training, all decision variable values were normalized to fall within the range [0, 1]. The efficiency values y fell in the range [0.0569, 0.2923]. A neural network with one node in the input layer, 4 nodes in the output layer and 2 hidden layers with 10 nodes each (a 1-10-10-4 neural network) was trained with these data. Training consisted of 50,000 iterations through the training data. The configurations $\mathbf{x}^{[k]}$ prescribed by the trained network for a range of desired performance index values $y_*^{[k]}$ are shown in Table 8-2.

k	INPUT $y_*[k]$	INVERSE NETWORK OUTPUT $\mathbf{x}_k$			
		x	s	b	f
1	0.20	0.910	0.860	0.248	0.417
2	0.22	0.909	0.803	0.187	0.400
3	0.24	0.894	0.732	0.132	0.392
4	0.26	0.836	0.622	0.101	0.395
5	0.28	0.593	0.374	0.218	0.421
6	0.30	-0.554	-0.467	1.191	0.520
7	0.32	-4.365	-2.988	4.791	0.805
8	0.34	-8.736	-5.967	8.786	1.165
9	0.36	-10.520	-7.544	9.892	1.420
10	0.38	-11.295	-8.632	9.785	1.655
11	0.40	-11.877	-9.626	9.446	1.886

TABLE 8-2. Configurations output by a neural network trained via the direct inverse method.

Since the process of generating training simulation data has already discovered a system with a performance index value of 0.2923, the range of $y_*[k]$ values of interest is $y_*[k] > 0.2923$. Looking at Table 8-2, most of the decision variable outputs produced for $y_*[k]$ inputs in this range (0.30, 0.32, etc.) are out of the interpretable range of [0, 1]. In addition, the root mean squared error across all n (= 4) network outputs and all S (= 128) training pairs at the end of training, as defined by

$$RMSE = \left[\frac{1}{nS} \sum_{k=1}^{S} \left(\mathbf{x}_*^{[k]} - \mathbf{x}^{[k]} \right)^T \left(\mathbf{x}_*^{[k]} - \mathbf{x}^{[k]} \right) \right]^{\frac{1}{2}} \tag{8-1}$$

where $\mathbf{x}_*[k]$ is a n-component vector specifying the desired network outputs for the k^{th} training pair and $\mathbf{x}[k]$ is the actual network output for the k^{th} training pair input, is a relatively large 0.3376 with respect to the desired output values, which are in the range [0, 1]. This is evidence of the inability of the direct inverse method to drive the squared error or equivalently the RMSE (Eq. 8-1) close to zero when one-to-many mappings are explicitly specified in the training data.

8.2. Distal Supervised Learning Method

In order to evaluate the distal supervised learning method, 72 simulations were run to generate 72 training pairs in which decision variable values x, s, b, f were the inputs and efficiency values y were the outputs (Tbl. 8-3).

# Samples	INPUT				OUTPUT
	x	s	b	f	y
72	3, 10, 17, 24	0, 1	0, 100, 150	0.25, 0.75, 1.00	

TABLE 8-3. Training pairs for the forward network in the distal supervised learning method.

Before being used for neural network training, all decision variable values were normalized to fall within the range [0, 1]. The efficiency values y fell in the range [0.0569, 0.2923]. A neural network with 4 nodes in the input layer, 1 node in the output layer and 2 hidden layers with 10 nodes each (a 4-10-10-1 neural network) was trained with these data. Training consisted of 1,021 cycles through the training data, resulting in a final RMSE (Eq. 8-1, $n = 1$, $S = 72$) of 0.02. This trained forward network was then used in the training of a 1-16-4 inverse network via the distal supervised learning method. Eleven performance index values $y_*^{[k]}$ were provided to the inverse network during the training process, ranging from 0.20 to 0.40 in increments of 0.02. The configurations $\mathbf{x}^{[k]}$ prescribed by the trained inverse network for a range of desired performance index values $y_*^{[k]}$ are shown in Table 8-4.

k	INPUT $y_*[k]$	INVERSE NETWORK OUTPUT $\mathbf{x}[k]$				FORWARD NET OUTPUT $y[k]$
		x	s	b	f	
1	0.20	4.783	-4.507	-1.191	1.709	0.20
2	0.22	4.748	-4.512	-1.075	1.714	0.22
3	0.24	4.712	-4.517	-0.959	1.719	0.24
4	0.26	4.677	-4.521	-0.844	1.724	0.26
5	0.28	4.642	-4.526	-0.729	1.729	0.29
6	0.30	4.606	-4.531	-0.615	1.733	0.30
7	0.32	4.572	-4.535	-0.502	1.738	0.32
8	0.34	4.537	-4.540	-0.389	1.743	0.34
9	0.36	4.502	-4.545	-0.277	1.748	0.36
10	0.38	4.468	-4.549	-0.166	1.752	0.38
11	0.40	4.434	-4.553	-0.056	1.757	0.39

TABLE 8-4. Configurations output by a neural network trained via the distal supervised learning method.

As with the direct inverse method, the decision variable values output by the inverse neural network fall outside the interpretable range of [0, 1]. The last column of Table 8-4 shows the outputs from the forward network, given the inverse network-prescribed designs as input. The forward network outputs almost exactly match the desired performance index values input to the inverse network. This shows that the distal supervised learning method is forming a true inverse of the forward neural network model, albeit an inverse whose outputs do not have a defined physical meaning.

8.3. Distal Supervised Learning with Constraints

In order to overcome this last difficulty, the distal supervised learning with constraints method was applied. The procedure was identical to that of distal supervised learning, with the exception of the squared error to be minimized during training of the inverse network. The squared error used was of the form given in Eq. 3-21, with the upper and lower bounds of the decision variables set to 0 and 1 respectively. The configurations $\mathbf{x}[k]$ prescribed by the trained inverse network for a range of desired performance index values $y_*[k]$ are shown in Table 8-5.

k	INPUT $y_*[k]$	INVERSE NETWORK OUTPUT $\mathbf{x}[k]$				FORWARD NET OUTPUT $y[k]$
		x	s	b	f	
1	0.20	0.435	1.000	0.444	0.453	0.20
2	0.22	0.384	1.000	0.393	0.402	0.22
3	0.24	0.333	1.000	0.343	0.353	0.23
4	0.26	0.284	1.000	0.294	0.304	0.25
5	0.28	0.235	1.000	0.246	0.257	0.26
6	0.30	0.188	1.000	0.199	0.210	0.27
7	0.32	0.142	1.000	0.153	0.165	0.29
8	0.34	0.097	1.001	0.109	0.120	0.30
9	0.36	0.053	1.001	0.065	0.076	0.31
10	0.38	0.010	1.001	0.022	0.034	0.32
11	0.40	-0.032	1.002	-0.020	-0.008	0.32

TABLE 8-5. Configurations output by a neural network trained via the distal supervised learning with constraints method.

In this case, the decision variable values prescribed by the inverse network do fall within the physically interpretable range of [0, 1]. The forward model does not produce output performance index values as high as the desired performance index values. If the forward model is a sufficiently accurate description of the actual design to performance index relationship, this indicates that the maximum physically achievable performance index value has been exceeded by the requested values.

8.4. Achieved Efficiency of Configurations Prescribed by an Inverse Neural Network

The NET4-10-1 models that were trained with various numbers of configuration to efficiency samples (16, 36, 54, 72, 96, 128, 160, 200 and 250) in Section 7.2.1 were used as forward models in nine applications of the distal supervised learning with constraints method (one application for each forward model trained with a given number of training samples). In each case, the inverse network used had a 1-5-5-4 structure. The desired efficiency values $y_*[k]$ that were used to train each inverse network were obtained by

incrementing the best efficiency value in the training data of the corresponding forward network by 5%, 10% and 15% respectively. For each inverse network, the three desired efficiency values resulted in the same prescribed configuration. The prescribed configuration varied across the different inverse networks, however. The achieved efficiency of the configurations prescribed by the trained inverse networks were then compared to that of some of the forward models applied in Section 7.3.1. The results are shown in Figure 8-1, which is identical to Figure 7-7 except for the addition of the inverse network results and the omission of the regression results (the latter simply to reduce clutter).

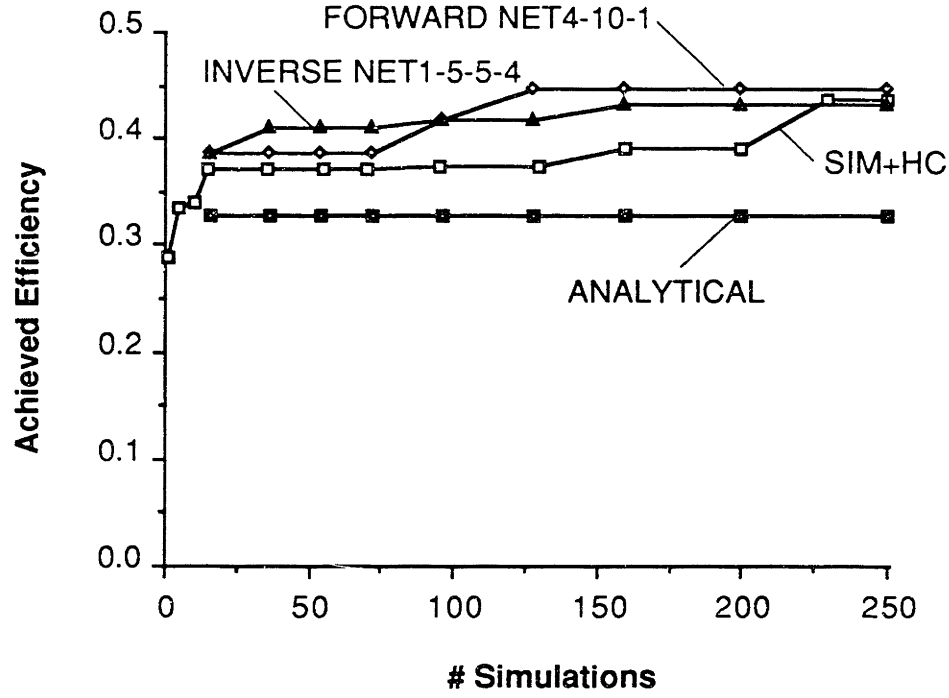


FIGURE 8-1. Achieved efficiency of the inverse modeling method compared to forward modeling methods.

The performance of the inverse network model closely matches that of the forward network model. The inverse network, like the forward network, has window of computational effort (incurred in the form of training simulations) within which its prescribed configurations outperform the more traditional forward model methods based on model

ANALYTICAL and simulation plus hill-climbing. As observed before, once the number of simulation data points reaches a certain point, about 225 for the given industrial problem, simulation plus hill-climbing is able to catch up with the network-based methods.

8.5. How Prescribed Configurations Vary with the Required Flexibility

Now that it has been established that the neural network trained via the distal supervised learning with constraints method is the best of the three inverse methods discussed for predicting a configuration **x** for a desired performance, an evaluation of how the network's predictions change with respect to the assumptions regarding the flexibility requirements imposed on the manufacturing system is possible. Because the efficiency index incorporates costs that occur over the life of the manufacturing system, including those due to part design changes, it is of particular importance that the effect of part design change frequency on the inverse network output be examined.

Model NET4-10-1 was trained with two sets of 250 training samples (Line 9, Tbl. 7-1). For the first training set, efficiency was evaluated assuming a large product design change frequency of once every two years. For the second training set, efficiency was evaluated assuming a low product design change frequency of once every five years. Other parameters values were set in accordance with Table 5-2. The trained models were then used to train 1-5-5-4 inverse networks via the distal supervised learning with constraints method. The trained inverse networks were then used to prescribe configurations (Fig. 8-2).

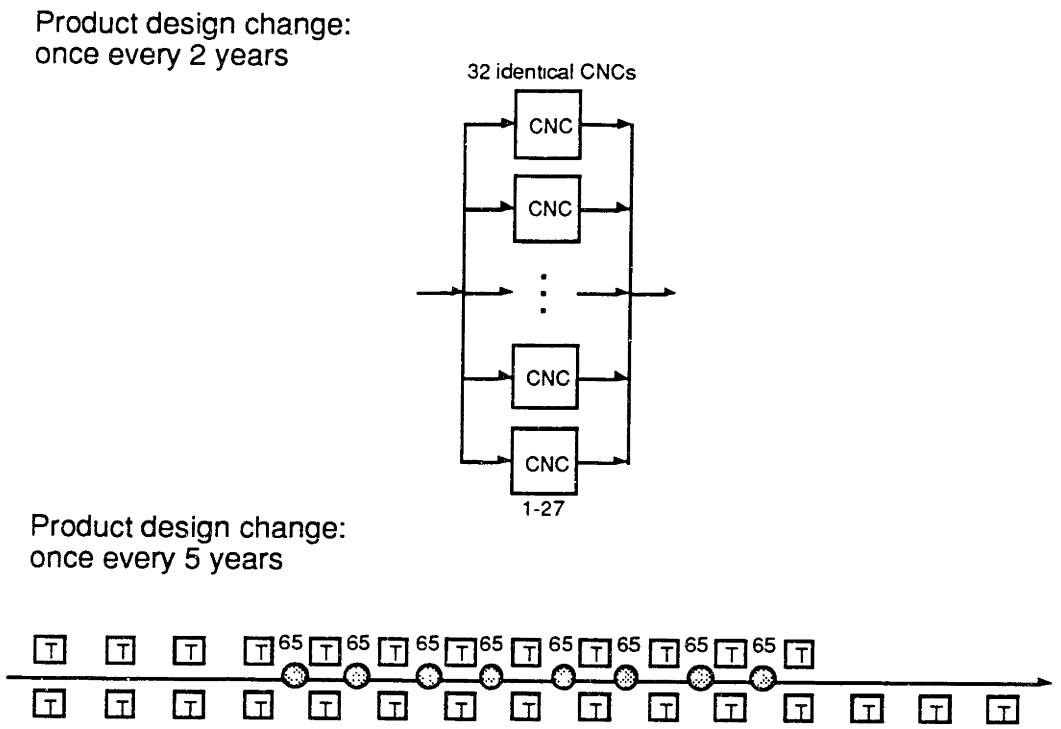


FIGURE 8-2. Configurations prescribed by an inverse neural network for scenarios involving frequent and infrequent product design changes.

Given the likelihood of frequent product design changes, the inverse network prescribes a configuration consisting solely of flexible CNC machines. Given infrequent product design changes, the neural network prescribes a configuration dominated by dedicated transfer line stations that have lower acquisition cost. This behavior is reasonable and is consistent with that of the forward neural network model-based method.

9. Larger Configuration Problems

For the given industrial problem, both forward and inverse neural network based methods can prescribe better configurations for a given amount of computational effort than forward methods based on a crude analytical model, on empirical linear regression models, and on simulation plus hill-climbing. However, the given problem is limited in size, with about 90,000 total solutions. In this chapter, the forward and inverse model methods discussed in this thesis will be applied to a larger problem to see if the relative performance of the methods remain the same.

9.1. Generalization of Decision Variables

The given configuration problem from industry can be made "larger" several ways. One way is to increase the number of machine types from the current two. A second way is to lift the restriction of a configuration to at most two sections of machines – one section of dedicated transfer line stations or batch operation machines arranged in series and one section of flexible CNC machining stations or sequential operation machines arranged in parallel. The second way requires a generalization of the existing decision variables. Given M machine types, the existing decision variables, of which $x_1, x_2, \ldots, x_{M-1}$ and $s_1, s_2, \ldots, s_{M-1}$ describe the assignment of operation groups to machine types (Section 4.1, Fig. 9-1), are capable only of indicating how many operation groups are assigned to each machine type, but not of indicating the distribution of these operation groups amongst multiple sections of a given machine type. For example, if a configuration consists of a section of CNCs, followed by a section of TLSs, followed by a section of CNCs, the existing decision variables would not be capable of describing how the operations groups allocated to CNCs are distributed between the two CNC sections.

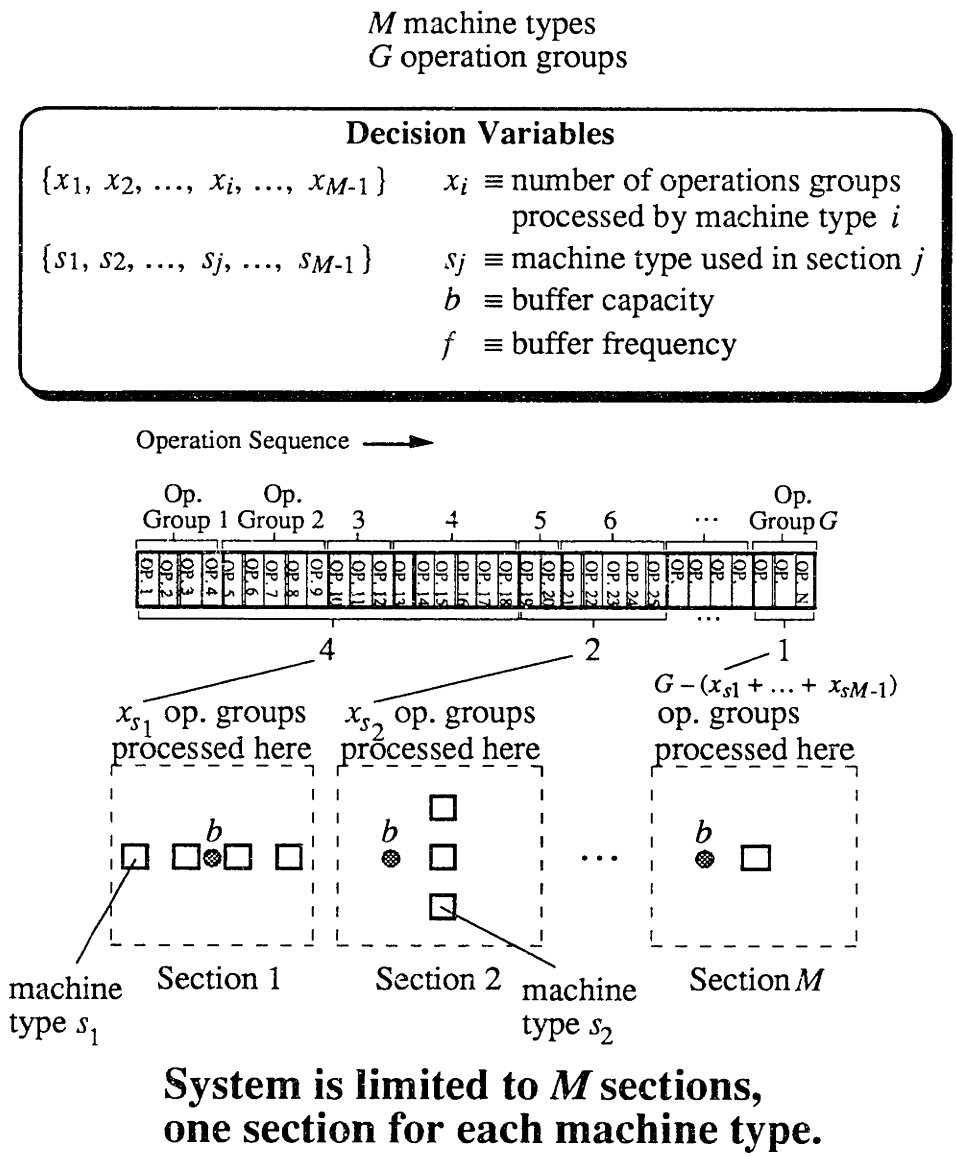


FIGURE 9-1. The existing decision variables can only represent configurations in which there is at most one section of each machine type.

This restriction can be lifted as follows. Let the number of machine sections in the system be an arbitary constant K. Then define decision variables as follows:

x_i The number of operation groups processed by whatever machine type is in machine section i ($i \in [1, K-1]$, $x_i \in [0, G]$). Only $K-1$ variables are necessary because the

number of operation groups processed by the machine type in section K is determined by default from the expression $G-(x_1+x_2+\ldots+x_{K-1})$.

s_j The machine type in section j ($j \in [1, K]$, $s_j \in [0, M]$). As a part travels through the system, the first machine type it encounters is s_1, the next type it encounters is s_2, and so on. If there are only r different sections in the system, then s_{r+1}, s_{r+2}, ... s_K are all set to 0.

The above redefinition of the x_i and s_j decision variables permits configurations with multiple sections of a single machine type to be represented (Fig. 9-2). The remaining decision variables remain unchanged.

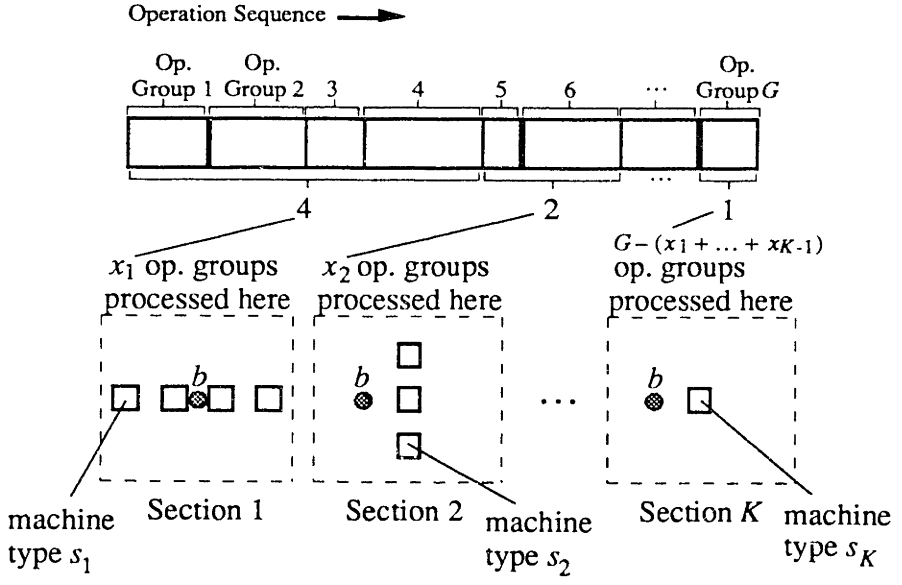
b The capacity of the buffers within the system ($b \in [0, b_{\max}]$). This quantity is assumed to be uniform throughout the system.

f The frequency with which buffers occur within the system. This is defined to be the number of buffers within the system divided by the number of potential buffer locations within the system. Within a system, buffers may be located between any two adjacent machines in the part flow.

M machine types
G operation groups
K machine sections

Decision Variables

$\{x_1, x_2, \ldots, x_i, \ldots, x_{K-1}\}$	$x_i \equiv$ number of operations groups processed by machine section i
$\{s_1, s_2, \ldots, s_j, \ldots, s_K\}$	$s_j \equiv$ machine type used in section j
	$b \equiv$ buffer capacity
	$f \equiv$ buffer frequency



Number of sections not longer limited to M, the number of machine types.

A given machine type may appear in more than one section.

FIGURE 9-2. Generalized decision variables.

The generalized decision variables permit consideration of a richer set of configurations, at a price of more decision variables: $2K+1$ variables versus the original $2M$, where K is the arbitrarily defined maximum number of machine sections, and M is the number of machine types. Increasing K increases the richness of the configuration space.

9.2. A Richer Solution Space for the Industrial Problem

The industrial problem of Chapter 5 was solved using the generalized decision variables. As before, the number of machine types $M = 2$, and the number of operation groups $G = 27$. However, the maximum number of machine sections K was increased from 2, the effective assumption given the previous set of decision variables, to a new value of 5. This increased the number of decision variables from 4 to 11, and the number of potential solution configurations by almost 4 orders of magnitude, from 9.0×10^4 to 4.2×10^8.

Two hundred sixteen training samples were generated via simulation. Blocks of 54 configurations containing 2, 3, 4 and 5 machine sections respectively were simulated (Tbl. 9-1). Efficiency values for the training samples were calculated using the parameter values in Table 5-2.

# Samples	INPUT $\mathbf{x}^{[s]}$				OUTPUT
	$(x_1\ x_2\ x_3\ x_4)$	$(s_1\ s_2\ s_3\ s_4\ s_5)$	b	f	$y^{[s]}$
27	(3 24 0 0) (14 13 0 0) (24 3 0 0)	(1 2 0 0 0)	10, 50, 90	0.0, 0.5, 1.0	
27	(3 24 0 0) (13 14 0 0) (24 3 0 0)	(2 1 0 0 0)	10, 50, 90	0.0, 0.5, 1.0	
27	(2 24 1 0) (7 13 7 0) (12 3 12 0)	(1 2 1 0 0)	10, 50, 90	0.0, 0.5, 1.0	
27	(12 3 12 0) (7 14 6 0) (2 24 1 0)	(2 1 2 0 0)	10, 50, 90	0.0, 0.5, 1.0	
27	(2 12 1 12) (7 7 7 6) (12 2 12 1)	(1 2 1 2 0)	10, 50, 90	0.0, 0.5, 1.0	
27	(12 2 12 1) (7 7 6 7) (2 12 1 12)	(2 1 2 1 0)	10, 50, 90	0.0, 0.5, 1.0	
27	(1 12 1 12) (5 7 5 6) (8 2 8 1)	(1 2 1 2 1)	10, 50, 90	0.0, 0.5, 1.0	
27	(8 2 8 1) (5 7 4 7) (1 12 1 12)	(2 1 2 1 2)	10, 50, 90	0.0, 0.5, 1.0	

TABLE 9-1. Training samples drawn from the rich configuration space.

Eight 11-30-1 neural networks (NET11-30-1) were trained using the first 27, 54, 81, 108, 135, 162, 189 and 216 training samples from Table 9-1 respectively. That is, the first two networks were trained using configurations containing 2 machine sections, the next two networks were trained using configurations containing up to 3 machine sections, the next two networks were trained using configurations containing up to 4 machine sections, and finally, the last two networks were trained using configurations containing up to 5 machine sections. The neural networks were trained for 7,000 iterations via backpropagation (Rumelhart *et al.* 1986, Section 3.1.3), with a gain α of 0.05 and a momentum η of 0.80.

The performance of NET11-30-1 was compared to that of NET4-10-1 (Section 7.3.1), which was trained via the original decision variables. The basis of comparison was the achieved efficiency of configurations prescribed by the two network models, for a given amount of simulation-generated training data (Fig. 9-3).

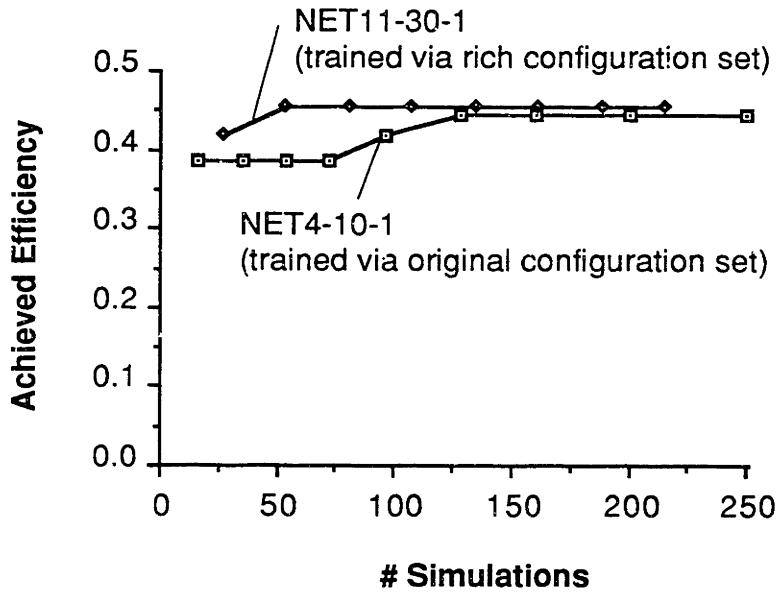


FIGURE 9-3. Achieved efficiency of the neural network-based forward model method for two different levels of training configuration richness.

Although the rich configuration set yields better prescribed configurations at lower numbers of training samples, it is important to note that the best achieved efficiency of NET11-30-1 is attained at 54 samples. At this point, the training data do not yet contain any configurations with more than 2 machine sections. The incorporation of a richer set of configurations into the training data thus did not help the performance of the neural network-based forward model method. This provides some justification for the formulation of the industrial problem of Chapter 5 as a 2-machine section problem.

9.3. Comparison of Configuration Approaches for a Larger Problem

The analytical forward model plus enumeration method (Section 3.2), the simulation plus hill-climbing method (Section 3.3), the linear regression forward model plus enumeration method (Section 3.4), the neural network forward model plus enumeration method (Section 3.5) and the neural network inverse model method (Section 3.6) were applied to a configuration problem consisting of $K = 6$ sections, $G = 28$ operation groups, and $M = 2$ machine types. This problem is distinct from the previously addressed industrial problem. It requires $2K+1 = 13$ decision variables and boasts a solution space of 4.6×10^9 configurations.

The linear regression model used was of the form given in Equation 3-1, which is linear with respect to the decison variables. A 13-16-1 neural network forward model and a 1-30-30-13 neural network inverse model were used.

The relative performance of the forward model methods from above for this large configuration problem, with a solution space approximately 5 orders of magnitude larger than the original industrial problem, is shown in Figure 9-4.

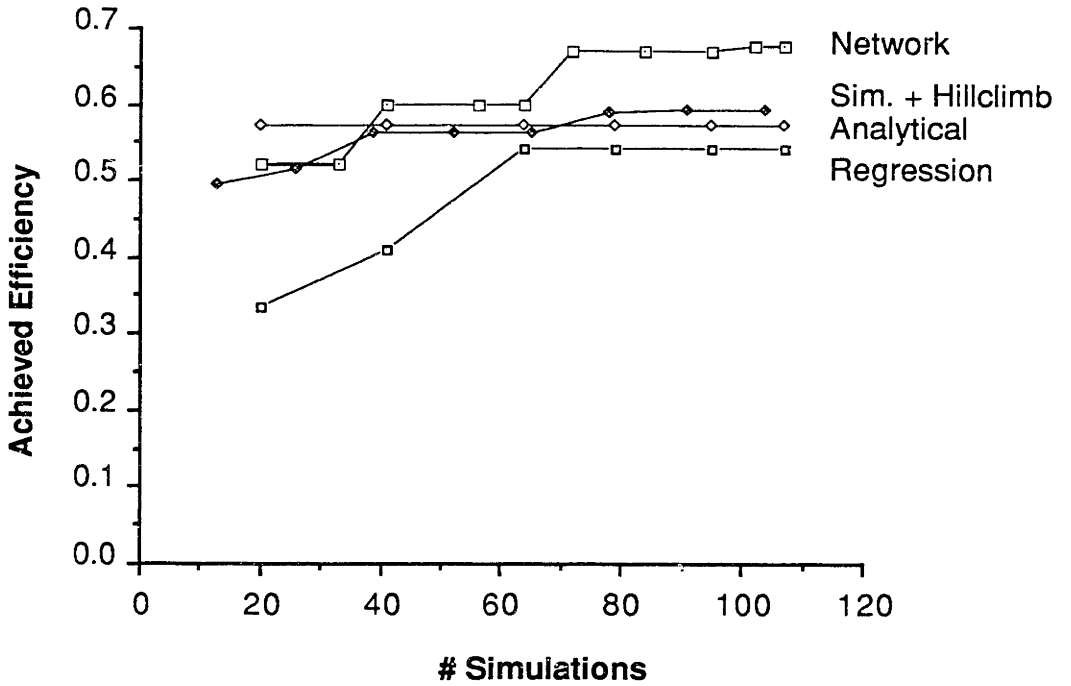


FIGURE 9-4. Achieved efficiency of different forward model configuration methods for a large configuration problem with 4.6×10^9 solutions.

As with the smaller, original configuration problem (Chapter 5), there is a window of computational effort (above approximately 40 training simulations) in which the network-based method prescribes better configurations than the other forward model methods. Scaling up the configuration problem did not change this result from the smaller problem.

The performance of the neural network inverse model method relative to some of the above forward methods is shown in Figure 9-5.

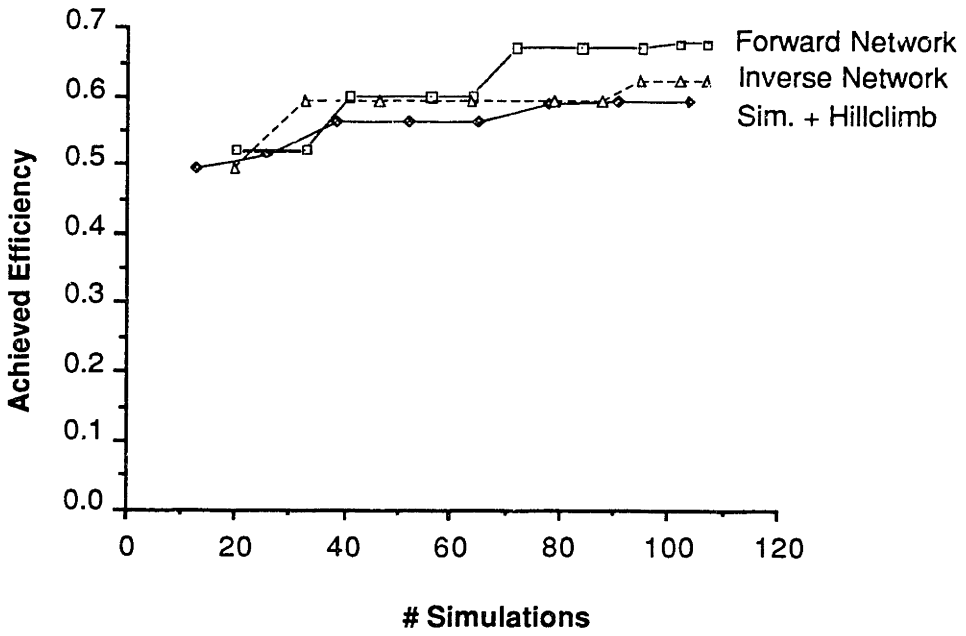


FIGURE 9-5. Performance of inverse method versus forward methods for a large configuration problem with 4.6×10^9 solutions.

For the large configuration problem, the performance of the inverse method falls off somewhat from that of the forward network method. Generally, it lies between that of the forward network method and that of simulation plus hill-climbing.

The advantage of the inverse method over the forward methods in a configuration problem of this size is computational effort. Exhaustive enumeration of the configuration space, even using a quickly-executed empirical model (such as a forward neural network) to evaluate each configuration, becomes very computationally expensive. On the other hand, an inverse network can be trained to successfully implement only a *single* efficiency to configuration mapping, where the single efficiency is the desired efficiency. It does not have to be trained to successfully map the entire range of efficiency values onto corresponding configurations. This can make inverse network training computationally efficient relative to the enumeration required for forward model methods.

10. Conclusions

The contributions of this research center around the following four areas.

Decision Variables

The use of empirical models for manufacturing system configuration requires the representation of alternative configurations in terms of quantitative decision variables. Since all of the empirical models investigated require a fixed number of inputs and outputs, the number of decision variables for representing all configurations must be a constant. A representation with these properties for the problem of designing multi-stage machining systems has been developed.

Although the industrial problem addressed involves only two machine types, the decision variables are general enough to handle a problem with an arbitrary number of machine types.

A further generalization permits configurations with an arbitrary number K of machine sections to be represented. Selection of K determines the richness of the set of configurations that can be drawn upon to solve the configuration problem. Large K means that configurations in which every few operations are performed by a different machine type are representable; small K means a coarser partitioning of the machining operations amongst the eligible machine types. Representational capability via large K is achieved at the expense of increasing the domain size of the configuration to performance index mapping; this is the mapping that must be approximated by a suitable forward model if the configuration problem is to be solved. A related experimental result is that for the given industrial problem, the consideration of configurations containing up to 5 machine sections

does not provide a significant benefit over considering just configurations with up to 2 machine sections.

Performance Index

The proposed performance index quantitatively assesses manufacturing flexibility. The guiding principle behind the quantification of flexibility is that manufacturing flexibility is a function of both the ability to adapt to change and the likelihood or demand for that change.

Decision Variables to Performance Index Mapping

For the given industrial problem, the roughness of the decision variables to performance index mapping was quantified via measures employing integrated first and second derivatives of the performance index. The mapping is relatively smooth. This motivates the use of empirical models to approximate the mapping and subsequently to prescribe manufacturing system configurations. If the mapping were too rough, an excessive number of computationally expensive mapping samples would be needed in order to employ empirical modeling methods.

Solution Methods

This research explored the use of computationally efficient empirical models fit via accurate but computationally expensive simulation models for manufacturing system configuration. Of the empirical models tested, neural network forward and inverse models generally prescribed better configurations than linear regression models. However, neural network models are more sensitive to training simulation accuracy than the linear regression models. For both the smaller original configuration problem, with 10^5 solutions, and a larger

problem with 10^9 solutions, there proved to be a window of computational effort (in the form of evaluation of configurations via simulation) within which neural network models prescribed better configurations with less computational effort than simulation plus hill-climbing. The neural network models also prescribed better configurations than a crude analytical model.

The above observations indicate that neural network models, as configuration tools, are best applied when evaluation of the performance index is both computationally expensive via simulation and difficult analytically.

The inverse model method for manufacturing system configuration requires a model that takes as input a desired performance index value y_{goal} and outputs a suitable manufacturing system configuration $\mathbf{x}^*$. This is difficult to address directly via empirical modeling tools because of the one-to-many nature of the mapping. The distal supervised learning with constraints method for training neural network models, however, is able to address this difficulty by selectively implementing one of the one-to-many mappings in accordance with given constraints on the mapping outputs.

The inverse neural network model method performed comparably to the forward neural network model method for the small configuration problem, and between the forward network method and simulation plus hill-climbing for the large configuration problem. It may be preferred to the forward network method in a situation in which enumeration of configurations, required by the forward network method, is computationally expensive.

Limitations

A number of issues related to this research remain open.

The proposed decision variables make use of so-called categorical variables to denote machine type (e.g., 1 represents CNC machining center, 2 represents transfer line station). In a configuration problem with more than two machine types, which machine types should be adjacent in value? The machines might be ordered by their degree of flexibility. The choice of machine order on the categorical variable scale may also affect the performance of the different configuration methods.

In this research, the valuation of inventory was set on the basis of the perceived importance of inventory reduction in industry. However, a model for arriving at this cost was not provided. The results of Section 7.3.2 suggest that the advantage of neural networks as configuration tools over more standard regression models will exist, but to a smaller degree, if inventory cost is significantly less than the value assumed in this research.

Results of the original, small configuration problem suggest that given enough simulation data, the performance of simulation plus hill-climbing eventually catches up to that of network-based methods. The computational effort window within which network-based methods perform better than simulation plus hill-climbing was observed in both the small and large configuration problems. However, whether this window exists for other configuration problems and how wide one could expect it to be in general has not been determined.

Finally, Figure 4-5 shows that the operation sequence of sequential operation machines (SOMs) such as CNC machining centers affects the performance of a given manufacturing system configuration. If operations are performed in a bad sequence, tool change times and tool spindle translation times will be excessive, resulting in a high processing time for each SOM and consequently the need for more SOMs to meet the demanded production

rate. For this research, the operation sequence was assumed to be given. However, better configurations could be prescribed by optimizing the operation sequence for each configuration, and then using the resulting optimal SOM processing times in the simulation of the configuration. Solution of the operation sequence problem would integrate seamlessly into the neural network-based methods for the configuration problem, because it would merely change the numerical values of their training data.

References

Apple, J.M., *Plant Layout and Material Handling*, 1950, The Ronald Press Company, New York.

Bazaraa, M.S. and H.D. Sherali, 1980, "Bender's Partitioning Scheme Applied to a New Formulation of the Quadratic Assignment Problem," *Naval Research Logistics Quarterly*, Vol. 27, p. 29.

Behnezhad, A.R. and B. Khoshnevis, 1988, "The Effects of Manufacturing Progress Function on Machine Requirements and Aggregate Planning Problems", *International Journal of Production Research*, Vol. 26, No. 2, pp. 309-326.

Bulgak, A.A. and J.L. Sanders, 1988, "Integrating a Modified Simulated Annealing Algorithm with the Simulation of a Manufacturing System to Optimize Buffer Sizes in Automatic Assembly Systems", *Proceedings of the 1988 Winter Simulation Conference*, pp. 684-690.

Caramanis, M., 1987, "Production System Design: A Discrete-Event Dynamic System and Generalized Benders' Decomposition Approach," *International Journal of Production Research*, Vol. 25, No. 8, pp. 1223-1234.

Castillo, E., 1988, *Extreme Value Theory in Engineering*, Academic Press, Inc., San Diego, California.

Chryssolouris, G. and M. Lee, 1992, "An Assessment of Flexibility in Manufacturing Systems," *Manufacturing Review*, Vol. 5, No. 2, pp. 105-116.

Chryssolouris, G., 1991, *Manufacturing Systems: Theory and Practice*, Springer-Verlag, New York.

Chryssolouris, G., M. Lee, J.E. Pierce, M.K. Domroese, 1990, "Use of Neural Networks for the Design of Manufacturing Systems", *Manufacturing Review*, Vol. 3, No. 3, pp. 187-194.

Dutta, S. and S. Shekhar, 1988, "Bond Rating: A Non-Conservative Application of Neural Networks", *1988 IEEE International Conference on Neural Networks*, II-443–II-450.

Evans, G.W., M.R. Wilhelm, and W. Karwowski, 1987, "A Layout Design Heuristic Employing the Theory of Fuzzy Sets," *International Journal of Production Research*, Vol. 25, No. 10, pp. 1431-1450.

Filley, R.D., 1983, "CAD for Facilities Planning: Survey Identifies Software, Systems Most Useful to IEs," *Industrial Engineering*, March 1983, p. 66.

Fogelman Soulie, F., P. Gallinari, Y. Le Cun and S. Thiria, 1987, "Evaluation of Network Architectures on Test Learning Tasks," IEEE First International Conference on Neural Networks, June 1987, San Diego, CA, pp. II-653-II-660.

Francis, R.L. and J.A. White, 1974, *Facility Layout and Location: An Analytical Approach*, Prentice-Hall, New York.

Gaskins, R.J. and J.M.A. Tanchoco, 1987, "Flow Path Design for Automated Guided Vehicle Systems," *International Journal of Production Research*, Vol. 25, No. 5, pp. 667-676.

Gavett, J.W. and N.V. Plyter, 1966, "The Optimal Assignment of Facilities to Locations by Branch and Bound," *Operations Research*, Vol. 14, p. 210.

Gershwin, S.B. and I.C. Schick, 1979, "Analytic Methods for Calculating Performance Measures of Production Lines with Buffer Storages," Laboratory for Information and Decision Systems, Working Paper LIDS–P–863.

Gershwin, S.B., R.R. Hildebrant, R. Suri and S.K. Mitter, 1986, "A Control Perspective on Recent Trends in Manufacturing Systems," *IEEE Control Systems Magazine*, April 1986.

Graves, G.W. and A.B. Whinston, 1970, "An Algorithm for the Quadratic Assignment Problem," *Management Science*, Vol. 17, p. 453.

Gumbel, E.J., 1954, "Statistical Theory of Extreme Values and Some Practical Applications," *National Technical Information Service Report PB-175 818*, U.S. Department of Commerce, Springfield, VA.

Gunn, E.A. and A. Kusiak, 1986, "The Procurement Problem: An Integer Programming Problem Well Suited to a Solution Using Duality", *Naval Research Logistics Quarterly*, Vol. 33, pp. 613-620.

Hall, A.R. *et al.*, 1991, "Aluminum Transmission Case Transfer Lines Process Principles," Ford Motor Company Transmission and Chassis Division Report, May 1, 1991.

Heragu, S. and A. Kusiak, 1988, "Knowledge Based System for Machine Layout (KBML)," *1988 International Industrial Engineering Conference Proceedings*, pp. 159-164.

Ho, Y.C., R. Suri, X.R. Cao, G.W. Diehl, J.W. Dille and M. Zazanis, 1984, "Optimization of Large Multiclass (Non-Product Form) Queueing Networks Using Perturbation Analysis," *Large Scale Systems*, Vol. 7, No. 2/3, Dec 1984, pp. 165-180.

Jafari, M. A. and J. G. Shanthikumar, 1989, "Determination of Optimal Buffer Storage Capacities and Optimal Allocation in Multistage Automatic Transfer Lines", *IIE Transactions*, Vol. 21, No. 2, pp. 130-135.

Jordan, M.I. and D.E. Rumelhart, 1992, "Forward Models: Supervised Learning with a Distal Teacher," *Cognitive Science*, Vol. 16, pp. 307-354.

Jordan, M.I., 1992, "Constrained Supervised Learning," *Journal of Mathematical Psychology*, Vol. 36, pp. 396-425.

Kaplan, R.S., 1983, "Measuring Manufacturing Performance: A New Challenge for Managerial Accounting Research," *The Accounting Review*, Vol. 58, No. 4, pp. 686-705.

Kaufmann, L. and F. Broeckx, 1978, "An Algorithm for the Quadratic Assignment Problem Using Benders' Decomposition," *European Journal of Operations Research*, Vol. 2, p. 204.

Kirkpatrick, S., C.D. Gelatt Jr., and M.P. Vecchi, 1983, "Optimization by Simulated Annealing," *Science*, Vol. 220, pp. 671-680.

Koulamas, C. P., 1989, "Optimal Buffer Space Size in Two-Stage Machining Systems with Markovian or Non-Markovian Tool Life Processes", *International Journal of Production Research*, Vol. 27, No. 7, pp. 1167-1178.

Kusiak, A., 1987, "The Production Equipment Requirements Problem," *International Journal of Production Research*, Vol. 25, No. 3, pp. 319-325.

Law, A.M. and W.D. Kelton, 1991, *Simulation Modeling and Analysis*, McGraw-Hill, Inc., New York.

Lawler, E.L., 1963, "The Quadratic Assignment Problem," *Management Science*, Vol. 9, p. 586.

Lee, R.C. and J.M. Moore, 1967, "CORELAP – Computerized Relationship Layout Planning," *Journal of Industrial Engineering*, Vol. 18, p. 194.

Liggett, R.S., 1981, "The Quadratic Assignment Problem: An Experimental Evaluation of Solution Strategies," *Management Science*, Vol. 27, No. 4, April 1981, pp. 442-458.

Lippmann, R.P., 1987, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine,* April 1987.

Miller, D.M. and R.P. Davis, 1977, "The Machine Requirements Problem", *International Journal of Production Research*, Vol. 15, No. 2, pp. 219-221.

Miller, D.M., and R.P. Davis, 1978, "A Dynamic Resource Allocation Model for a Machine Requirements Problem," *AIIE Transactions*, Vol. 10, No. 3, pp. 237-243.

Morris, W.T., 1958, "Facilities Planning," *Journal of Industrial Engineering*, Vol. 9.

Morris, W.T., 1962, *Plant Layout and Design*, Macmillan, New York.

Rabeneck, C.W., J.S. Usher, and G.W. Evans, 1989, "An Analytical Models for AGVS Design," *1989 International Industrial Engineering Conference & Societies' Manufacturing and Productivity Symposium Proceedings*, pp. 191-195.

Reed, R., Jr., 1961, *Plant Layout: Factors, Principles, and Techniques*, R.D. Irwin, Homewood, IL.

Rumelhart, D.E., G.E. Hinton and R.J. Williams, 1986, "Learning Internal Representation by Error Propagation", *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, Foundations*, D.E. Rumelhart and J.L. McClelland (eds.), MIT Press, Cambridge, MA.

Sahni, S. and T. Gonzalez, 1976, "P-Complete Approximation Problems," *Journal of the Association of Computing Machinery*, Vol. 23, No. 3, p. 555-565.

Scott, David W., 1992, *Multivariate Density Estimation*, Wiley, New York.

Seber, G.A.F., 1977, *Linear Regression Analysis*, Wiley, New York.

Seber, G.A.F., 1989, *Nonlinear Regression*, Wiley, New York.

Seehof, J.M. and W.O. Evans, 1967, "Automated Layout Design Program," *The Journal of Industrial Engineering*, Vol. 18, No. 12, pp. 690-695.

Shubin, J.A. and H. Madeheim, 1951, *Plant Layout*, Prentice-Hall, New York.

Son, Y.K., 1991, "A Cost Estimation Model for Advanced Manufacturing Systems," *International Journal of Production Research*, Vol. 29, No. 3, pp. 441-452.

Soulie, F.F., P. Gallinari, Y. Le Cun, S. Thiria, 1987, "Evaluation of Neural Network Architectures on Test Learning Tasks", *IEEE First International Conference on Neural Networks*, II-653-II-660.

Suh, N.P., 1990, *The Principles of Design*, Oxford University Press, New York.

Suresh, N.C. and J.R. Meredith, 1985, "Justifying Multimachine Systems: An Integrated Strategic Approach," *Journal of Manufacturing Systems*, Vol. 4, No. 2, pp. 117-134.

Suri, R. and R.R. Hildebrant, 1984, "Modelling Flexible Manufacturing Systems Using Mean Value Analysis," *Journal of Manufacturing Systems*, Vol. 3, No. 1, pp. 27-38.

Swamidass, P.M. and M.A. Waller, 1990, "A Classification of Approaches to Planning and Justifying New Manufacturing Technologies," *Journal of Manufacturing Systems*, Vol. 9, No. 3, pp. 181-193.

Ueno, N., S. Sotojima, and J. Takeda, 1991, "Simulation-Based Approach to Design a Multi-Stage Flow-Shop in Steel Works," *Proceedings of the 24th Annual Simulation Symposium*. IEEE Computer Society Press, Los Angeles, California. pp. 332-337.

Usher, J.S., G.W. Evans, and M.R. Wilhelm, 1988, "AGV Flow Path Design and Load Transfer Point Location," *1988 International Industrial Engineering Conference Proceedings*, pp. 174-179.

Wabalickis, R.N., 1988, "Justification of FMS with the Analytical Hierarchy Process," *Journal of Manufacturing Systems*, Vol. 7, No. 3, pp. 175-182.

Wilhelm, M.R. and G.W. Evans, 1987, "The State-of-the-Art in AGV System Analysis and Planning", *Proceeding of the AGVS '87*, October 1987, Pittsburgh, Pennsylvania.

THESIS PROCESSING SLIP

FIXED FIELD: ill _____ name _____

index _____ biblio _____

▶ COPIES (Archives) Aero Dewey (Eng) Hum

Lindgren Music Rotch Science

TITLE VARIES ▶☐ _____

_____

_____

NAME VARIES. ▶☐ _____

_____

IMPRINT (COPYRIGHT) _____

▶COLLATION: 159 p

_____

▶ADD. DEGREE: _____ ▶DEPT.: _____

SUPERVISORS: _____

_____

_____

_____

_____

_____

_____

_____

_____

NOTES.

cat'r: date:

page: 339

▶DEPT: M.E.

▶YEAR: 1993 ▶DEGREE: Ph.D.

▶NAME LEE, Moshin

_____