DESIGN OF MANUFACTURING SYSTEMS WITH THE AID OF NEURAL NETWORK INVERSE MODELS

by

Sean Arnold

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

Bachelor of Science

at the

Massachusetts Institute of Technology

May 1993

Signature of Author [illegible]
Department of Mechanical Engineering
May 7,1993

Certified by [illegible]
Professor George Chryssolouris
Thesis Supervisor

Accepted by [illegible]
Professor Peter Griffith
Chairman, Department Committee

DESIGN OF MANUFACTURING SYSTEMS WITH THE AID OF NEURAL NETWORK INVERSE MODELS

by

SEAN ARNOLD

Submitted to the Department of Mechanical Engineering
on May 7, 1993 in partial fulfillment of the
requirements for the Degree of Bachelor of Science in
Mechanical Engineering

ABSTRACT

A manufacturing system design can be described by a vector of numerical decision variables $\mathbf{x}$. The system's performance may be characterized by a scalar y, which may be a function of one or more performance measures. The manufacturing system design problem may be viewed as finding a design $\mathbf{x}$ such that its performance $y(\mathbf{x})$ achieves a goal value y_{goal}. This problem may be solved via an inverse model f^{-1} such that $\mathbf{x} = f^{-1}(y_{goal})$. This paper deals with two issues: first, the establishment of suitable decision variables and performance measures; and second, an evaluation of neural networks as inverse models for manufacturing systems design.

The results show that it is possible for an inverse neural network to provide a design solution whose performance exceeds the best systems on which it is trained.

Thesis Supervisor: Professor George Chryssolouris

1. Introduction

There are a number of approaches to the difficult endeavor of designing manufacturing systems. In the academic literature, the overall manufacturing system design problem has usually been decomposed into sub-problems which are then treated independently (Chryssolouris 1992). One common 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 to minimize acquisition and operation costs or meet a given production rate requirement (Miller and Davis 1977, Behnezhad and Khoshnevis 1988, Lee *et al.* 1989). The resource layout problem is the problem of locating a set of resources in a constrained floor space. The objective is typically to minimize the costs of placing particular resources at particular locations, and the costs of locating resources with a high degree of interaction far away from each other (Carrie *et al.* 1978, Liggett 1981, Evans *et al.* 1987). In material flow problems, the objective is to determine the layout of a material handling system (e.g., an automated guided vehicle system with fixed travel paths) such that the total distance travelled by material per unit time is minimized (Gaskins and Tanchoco 1987, Rabeneck *et al.* 1989). 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 (Jafari and Shanthikumar 1989). The above sub-problems are interrelated in the sense that the optimal solution of any one depends upon the solution chosen for the others. For example, optimal buffer capacities depend on the number of resources of each type in the system. The sub-problems are usually separately treated, however, because important manufacturing system performance measures (e.g., production rate, work-in-process) are difficult to express analytically; separation of the sub-problems overcomes this difficulty in two ways: 1) it allows surrogate, more easily calculated performance measures (e.g., total material travel distance) to be used; 2) it may restrict the design of the system to configurations whose performance measures are easier to calculate (e.g., transfer lines with strictly serial material flow).

In industrial practice, trial and error remains the most frequently used design approach. In this approach, a suitable manufacturing system design (values for an appropriate collection of decision variables) is first guessed, and then the performance of the system design is evaluated, typically via discrete-event simulation software. If the performance is unsatisfactory, then the guess-and-evaluate design cycle is repeated. When designing a large and complicated manufacturing system, many cycles may be required. Often, however, the number of cycles is limited by the computational burden of the required simulations. The success of the trial and error approach relies heavily on the skill of the designer or "guesser". Intuition and rules of thumb derived from experience are often applied (Malde and Bafna 1986, Ballard *et al.* 1989).

The above approaches to manufacturing system design are characterized by the use of *forward* models which relate manufacturing system designs to their performance. This paper, in contrast, addresses the manufacturing system design problem with the help of *inverse* models which map a desired performance goal y_{goal} onto a system design (characterized by a vector of numerical decision variables $\mathbf{x}$) that achieves that goal. This paper seeks to: 1) provide a framework for evaluating the performance $y(\mathbf{x})$ of a manufacturing system; and 2) investigate methods of using neural networks to construct inverse models of the mapping $y(\mathbf{x})$ – that is, models which take desired y values as input and produce $\mathbf{x}$ values as output.

2. An Evaluation Framework for Manufacturing Systems Design

In this paper, an evaluation framework encompasses the definition of manufacturing system decision variables $\mathbf{x}$ 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 $\mathbf{x}$ 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 designing a manufacturing system for multi-stage, high-volume machining. The manufacturing system to be designed must take as input raw castings and output machined parts (Fig. 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.

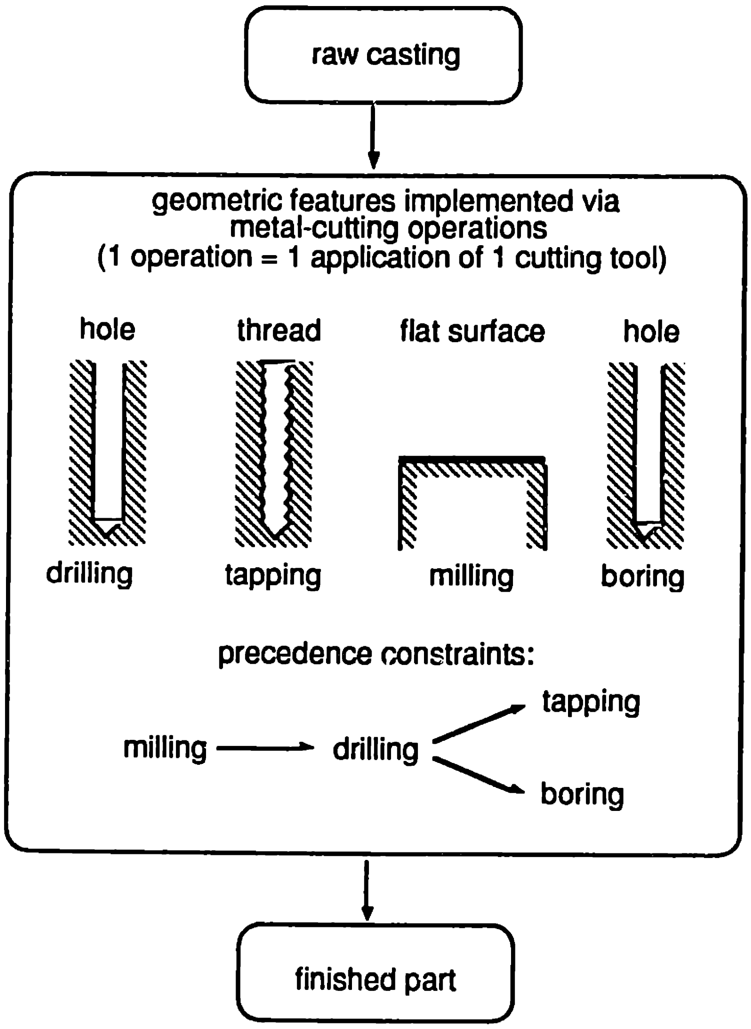

FIGURE 1. Required processing.

There are two generic machine classes for such a manufacturing system. A batch operation machine (BOM, Fig. 2a) has tool heads 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. 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.

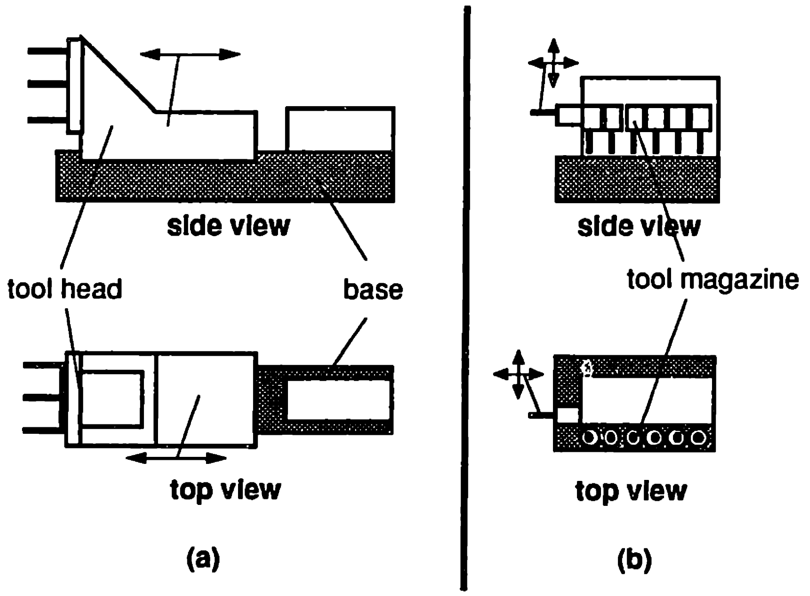

FIGURE 2. Schematics of the two generic machine classes: (a) batch operation machine, (b) sequential operation machine.

Batch operation machines (BOMs) are arranged serially in a transfer line (Fig. 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. 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. 3c).

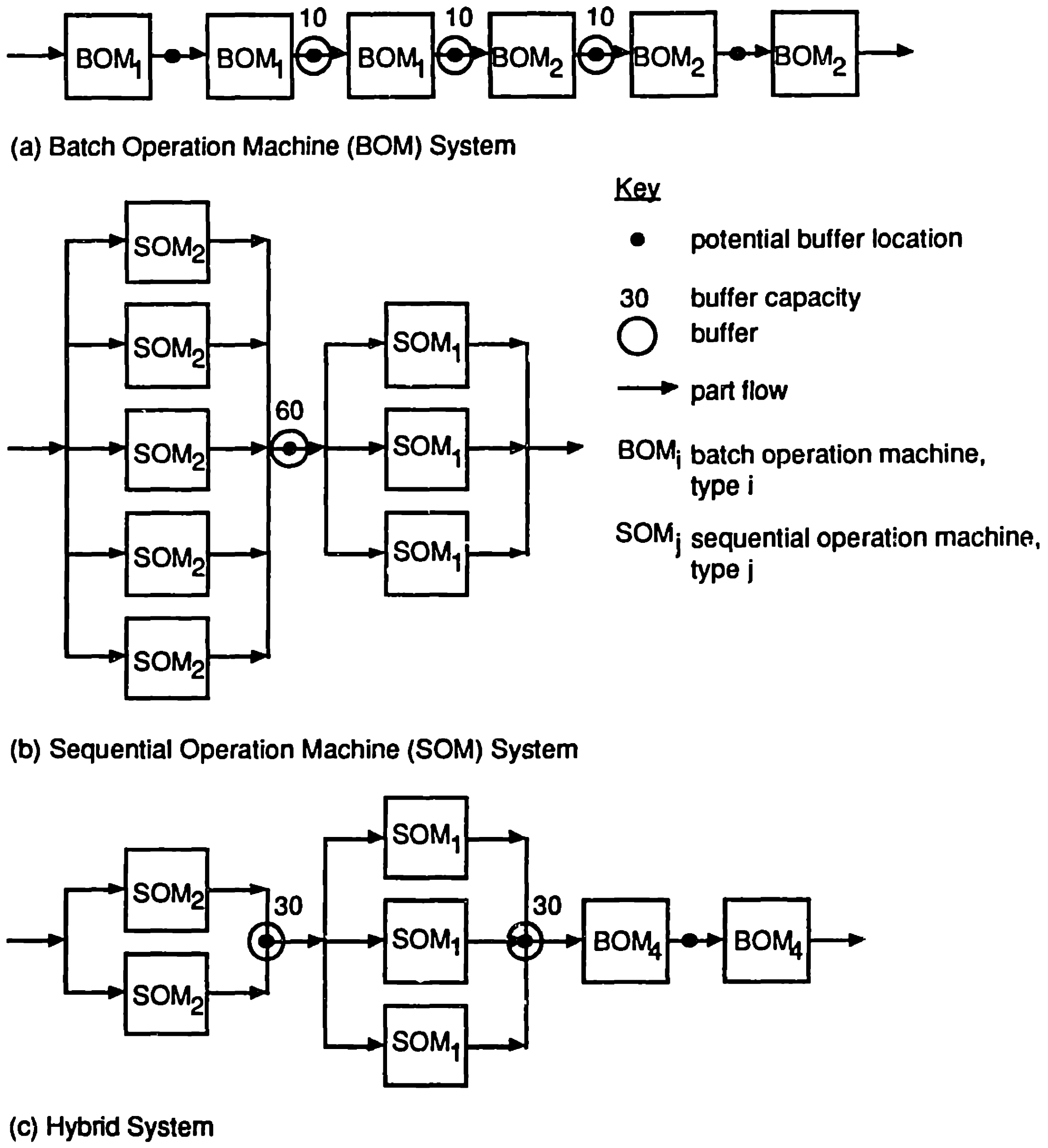
BOM1
BOM1
10
BOM1
10
BOM2
10
BOM2
BOM2
(a) Batch Operation Machine (BOM) System
Key
● potential buffer location
30 buffer capacity
buffer
part flow
BOMi batch operation machine, type i
SOMj sequential operation machine, type j
SOM2
SOM2
SOM2
SOM2
SOM2
60
SOM1
SOM1
SOM1
(b) Sequential Operation Machine (SOM) System
SOM2
SOM2
30
SOM1
SOM1
SOM1
30
BOM4
BOM4
(c) Hybrid System

FIGURE 3. Generic manufacturing system types.

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 ($j \in [1, M-1], s_j \in [1, M]$). 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 $s_{k+1}, s_{k+2}, \ldots s_{M-1}$ 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).

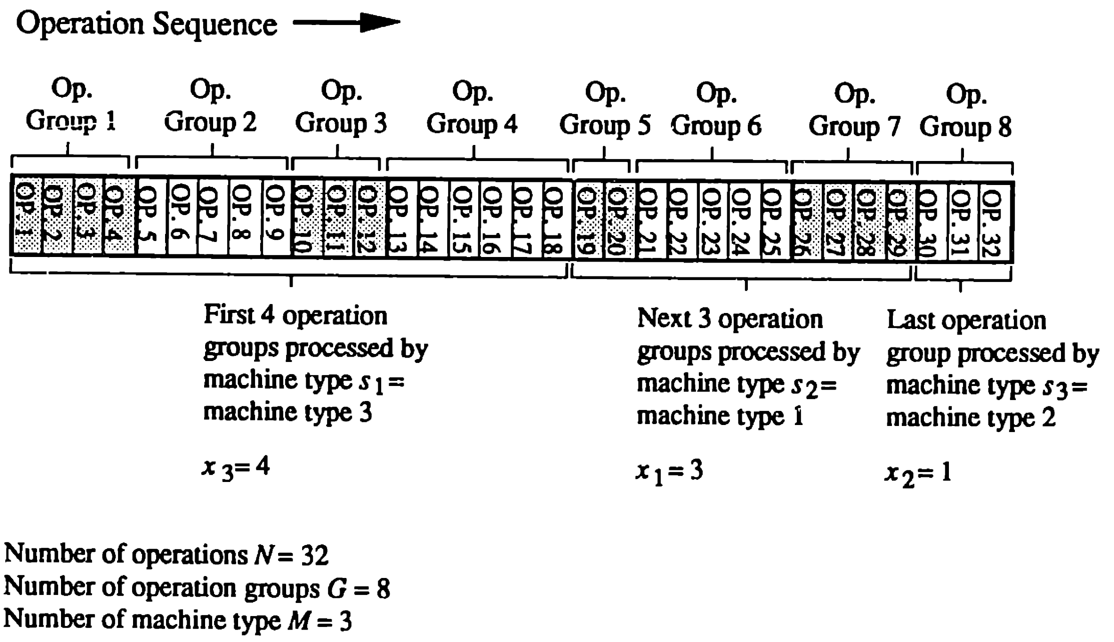

FIGURE 4. An illustration of the x_i and s_j decision variables for a hypothetical manufacturing system for a given operation sequence.

In order for the x_i and s_j decision variables to uniquely identify which operation groups are processed by which machine types, it is necessary to constrain the set of allowable manufacturing systems to those in which each machine type processes only a single block of operation groups. For example, in Figure 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.

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 design 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 design (Fig. 5).

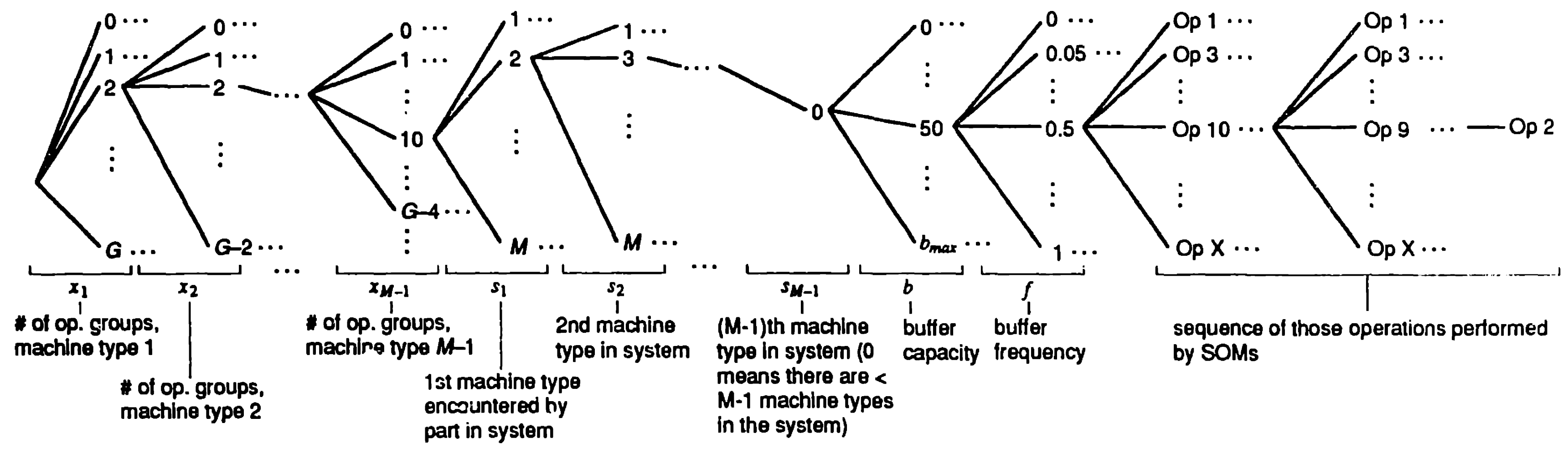
0 ···
1 ···
2
G ···
0 ···
1 ···
2
G–2 ···
0 ···
1 ···
10
G–4 ···
1 ···
2
M ···
1 ···
3
M ···
0
0 ···
50
b_{max} ···
0 ···
0.05 ···
0.5
1 ···
Op 1 ···
Op 3 ···
Op 10 ···
Op X ···
Op 1 ···
Op 3 ···
Op 9 ···
Op 2
Op X ···
x_1
# of op. groups, machine type 1
x_2
# of op. groups, machine type 2
x_{M-1}
# of op. groups, machine type M–1
s_1
1st machine type encountered by part in system
s_2
2nd machine type in system
s_{M-1}
(M-1)th machine type in system (0 means there are < M-1 machine types in the system)
b
buffer capacity
f
buffer frequency
sequence of those operations performed by SOMs

FIGURE 5. The space of manufacturing system designs **x**.

From this tree (Fig. 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 paper, the operation sequence is assumed to be given, an output of the process planning function.

As an example of the application of these decision variables, consider a specific case, the hybrid system (Fig. 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 $2M = 2 \bullet 4 = 8$ 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.

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 $\delta_b = <n_{bl}/n_b>^-$ spaces, where $<\bullet>^-$ denotes the greatest integer less than or equal to $\bullet$. In the hybrid system (Fig. 3c), $\delta_b = <3/2>^- = 1$. 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.

Once the decision variables $\mathbf{x}$ are defined, an evaluation framework requires the definition of a performance index $y(\mathbf{x})$. The performance index used in this paper 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*.

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

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.

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

$C_a(p)$	$\equiv$	acquisition cost for period p
M	$\equiv$	number of machine types
$c_m(i)$	$\equiv$	acquisition cost of one unit of machine type i
$n_{ma}(i, p)$	$\equiv$	number of machine type i acquired in period p

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

Labor Cost

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

$C_l(p)$	$\equiv$	labor cost for period p
c_l	$\equiv$	labor cost rate
t	$\equiv$	duration of each period
$n_l(p)$	$\equiv$	average number of workers at any given time during period p
$PR(p)$	$\equiv$	demanded production rate for period p
$PA(p)$	$\equiv$	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

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

$C_i(p)$	$\equiv$	inventory cost for period p
c_i	$\equiv$	inventory carrying cost per part per period
$\overline{WIP}(p)$	$\equiv$	average work in process for period p

Maintenance Cost

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

$C_m(p)$	$\equiv$	total maintenance cost in period p
M	$\equiv$	number of machine types
$n_m(i, t)$	$\equiv$	number of machines of type i in period p
$c_p(i)$	$\equiv$	preventive maintenance cost of machine type i per period
$c_r(i)$	$\equiv$	repair cost rate for machine type i
$\bar{t}_d(i)$	$\equiv$	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.

$$C_c(p, n_\Delta) = \delta(p, n_\Delta) \sum_{j=1}^{n_m(p)} \left[1 - \prod_{k=1}^{n_j(j)} (1 - P_c(j, k))\right] c_c(j) \tag{6}$$

$C_c(p, n_\Delta)$	$\equiv$	part design change cost for period p if the number of periods per part design change is n_Δ
$\delta(p, n_\Delta)$	$\equiv$	1 if a part design change occurs in period p, 0 otherwise
$n_m(p)$	$\equiv$	total number of machines in the system in period p, irrespective of type
$n_f(j)$	$\equiv$	number of features worked on by the jth machine in the system
$P_c(j, k)$	$\equiv$	probability that the kth feature worked on by the jth machine will be modified in a design change
$c_c(j)$	$\equiv$	cost of modifying/replacing the jth 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 jth 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 jth machine, $c_c(j)$, yields an expected modification/replacement cost for the jth 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:

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

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.

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

$C_T(n_\Delta)$	$\equiv$	present value of total life cycle cost
n_Δ	$\equiv$	number of periods per part design change
n_t	$\equiv$	number of periods in the life cycle of the system
$C_a(p)$	$\equiv$	total machine acquisition cost for period p
$C_l(p)$	$\equiv$	total labor cost for period p
$C_i(p)$	$\equiv$	total inventory cost per period p
$C_m(p)$	$\equiv$	total maintenance cost in period p
$C_c(p, n_\Delta)$	$\equiv$	part design change cost for period p
r	$\equiv$	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:

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

e	$\equiv$	efficiency
n_Δ	$\equiv$	number of periods between part design changes
n_t	$\equiv$	number of periods in the system's life cycle
PA	$\equiv$	production rate
$PA \bullet T$	$\equiv$	number of parts produced in life cycle of duration T
$C_T(n_\Delta)$	$\equiv$	total life cycle cost, given that the number of periods between part design changes is n_Δ
$P_\Delta(n_\Delta)$	$\equiv$	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. 6), which measure a system's ability to accommodate design changes, they ensure that the efficiency index (Eq. 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 9 is the final definition of the performance index $y(\mathbf{x})$ for the multi-stage, high-volume machining problem.

The example problem addressed in this paper 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 contains 59 geometric features (e.g., holes, slots) requiring $N = 126$ operations for implementation. 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 TLS ($0 \leq x \leq 27$). The number of operation groups implemented via CNC is $27-x$.

Systems with many TLSs (high x) are likely to be preferred if the design of the part to be machined changes very infrequently, duc 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.

s Machine type sequence ($s \in \{1, 2\}$). $s = 1$ means that TLSs process the first x operation groups, and CNCs process the remaining $27-x$. $s = 2$ means that CNCs process the first $27-x$ operation groups, and TLSs process the remaining x.

b The capacity of inventory storage buffers in the system ($0 \leq b \leq 150$).

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. 6). As shown, the solution space contains $28 \times 2 \times 151 \times 16 = 135{,}296$ solutions.

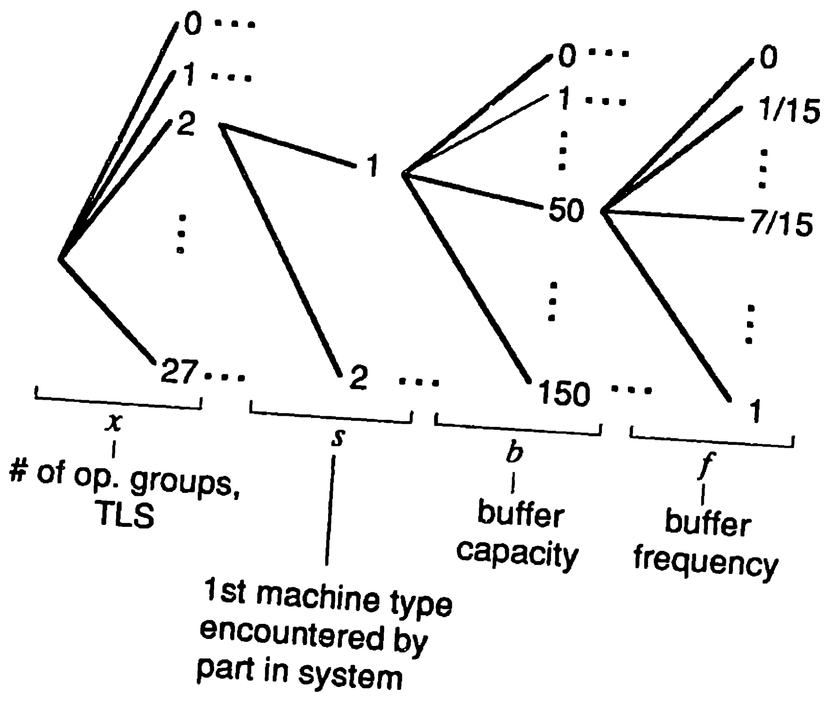

FIGURE 6. Solution space of the example problem.

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

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

TABLE 1. Machine processing times for the example problem.

Other parameters and their assumed values are summarized in Table 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
$P_\Delta(2)$	Probability that the number of periods between part design changes will be 2	0.5
$P_\Delta(3)$	Probability that the number of periods between part design changes will be 3	0.5
$P_\Delta(n_\Delta \neq 2, 3)$	Probability that the number of periods between part design changes is neither 2 nor 3	0.0

TABLE 2. Assumed parameter values for the example problem.

It is assumed that the manufacturing system to be designed will require a relatively high degree of flexibility. Part design changes are frequent: there is a 50% chance that the part design will change every two years ($P_\Delta(2) = 0.5$), and a 50% chance that the part design will chance every three years ($P_\Delta(3) = 0.5$). Furthermore, the extent of each design change is large: 90% of the geometric features in the part are altered with each design change.

3. Approaches for Formulating Inverse Simulation Models

The above evaluation framework defines the decision variables $\mathbf{x}$ and the performance index $y(\mathbf{x})$ for a manufacturing system design problem in high-volume machining. We now turn to the problem of constructing an inverse model f^{-1} such that $\mathbf{x} \approx f^{-1}(y(\mathbf{x}))$, so that given a desired performance index value y_{goal}, the design that will achieve that performance can be estimated by $\mathbf{x}_{goal} = f^{-1}(y_{goal})$.

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 design being $\mathbf{x}_i$. This approach has one principal difficulty. Certain performance measures comprising $f(\mathbf{x})$, such as the production rate (Eq. 3, Eq. 9) and the average work-in-process (Eq. 4), 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 Monte Carlo 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.

The remaining approaches to be investigated in this paper use discrete-event simulation in conjunction with a class of nonlinear regression models known as neural networks. The type of neural network models considered here are called multi-layer perceptrons (Rumelhart *et al.* 1986). These networks 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. 7). A network maps inputs $\mathbf{u}$ to outputs $\mathbf{v}$ in the following way. The output of the ith node in the input layer (layer 0) is clamped to the value of the ith component of the network input:

$$out_i^{[0]} = u_i \tag{10}$$

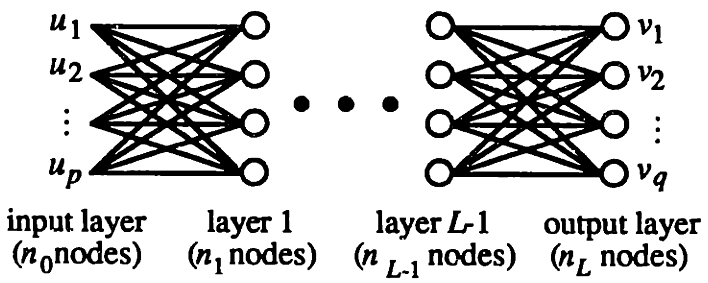

FIGURE 7. A multilayer perceptron neural network.

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

$$net_j^{[m]} = \sum_i w_{ji} in_i^{[m]} - w_{j0} \tag{11}$$

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, $out_j^{[m]}$:

$$out_j^{[m]} = \frac{1}{1 + \exp\left(-net_j^{[m]}\right)} \tag{12}$$

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

$$out_j^{[m]} = in_j^{[m+1]} \tag{13}$$

If the node j is in the output layer (layer L), then the node's output is taken to be the jth component of the network output **v**, and is simply a copy of its net input:

$$v_j = out_j^{[L]} = net_j^{[L]} \tag{14}$$

A network's weights w_{ji} are adjusted on the basis of a number of training pairs $\{(\mathbf{u}_1{}^*, \mathbf{v}_1{}^*), (\mathbf{u}_2{}^*, \mathbf{v}_2{}^*), \ldots, (\mathbf{u}_s{}^*, \mathbf{v}_s{}^*)\}$, 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 $(\mathbf{u}_k{}^*, \mathbf{v}_k{}^*)$, the vector $\mathbf{u}_k{}^*$ is input to the network, yielding the output vector $\mathbf{v}_k$. Each weight w_{ji} in the network is then adjusted according to the formula

$$\Delta w_{ji}[t+1] = -\eta \frac{\partial E_k}{\partial w_{ji}} + \alpha \Delta w_{ji}[t] \tag{15}$$

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_k is the squared error

$$E_k = \frac{1}{2}\left(\mathbf{v}_k^* - \mathbf{v}_k\right)^T \left(\mathbf{v}_k^* - \mathbf{v}_k\right) \tag{16}$$

In Equation 15, 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 total squared error

$$E = \sum_{k=1}^{s} E_k \tag{17}$$

Once trained (i.e., when the total squared error E falls below a predetermined level), the network may be used as a model of the mapping $\mathbf{v} = \mathbf{g}(\mathbf{u})$.

A second approach for constructing an inverse model $f^{-1}(y)$ is to generate via simulation a number of input-output *training pairs* of the form $\{(y(\mathbf{x}_1^*), \mathbf{x}_1^*), (y(\mathbf{x}_2^*), \mathbf{x}_2^*), \ldots, (y(\mathbf{x}_s^*), \mathbf{x}_s^*)\}$. 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 $y(\mathbf{x}_i^*)$ as the network inputs $\mathbf{u}_i$ and the designs $\mathbf{x}_i^*$'s as the network output targets $\mathbf{v}_i^*$. The model f^{-1} is used by supplying it with a desired performance goal y_{goal}, and recording the prescribed design $\mathbf{x}_{goal} = f^{-1}(y_{goal})$. 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 designs $\mathbf{x}_i^*$ have the same or very similar performance index values $y(\mathbf{x}_i^*)$. 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

$$E = \frac{1}{2}\left[(\mathbf{x}_1^* - \mathbf{x})^T(\mathbf{x}_1^* - \mathbf{x}) + (\mathbf{x}_2^* - \mathbf{x})^T(\mathbf{x}_2^* - \mathbf{x})\right] \tag{18}$$

where $\mathbf{x}$ is the network output given y^* as input. This is minimized by the average design $\mathbf{x} = (\mathbf{x}_1^* + \mathbf{x}_2^*)/2$. 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 $\{(\mathbf{x}_1^*, y(\mathbf{x}_1^*)), (\mathbf{x}_2^*, y(\mathbf{x}_2^*)), \ldots, (\mathbf{x}_s^*, y(\mathbf{x}_s^*))\}$ 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. 8).

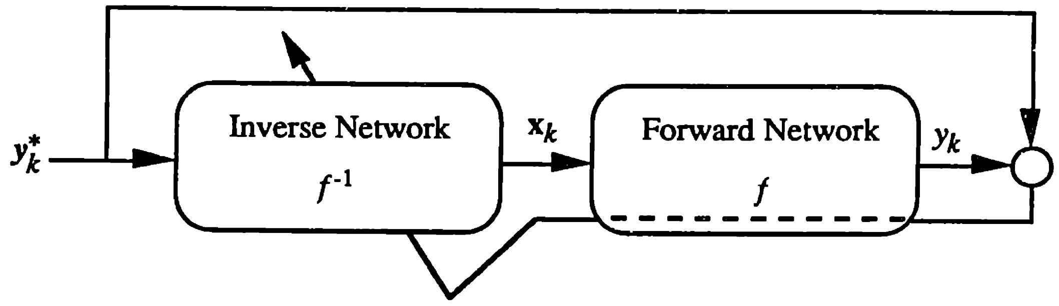

FIGURE 8. Distal supervised learning.

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

$$E_k = \frac{1}{2}(y_k^* - y_k)^T (y_k^* - y_k) \tag{19}$$

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 16. In this case, the derivative $\partial E/\partial w_{ji}$ can be evaluated using the chain rule as:

$$\frac{\partial E_k}{\partial w_{ji}} = -\frac{\partial \mathbf{x}_k^T}{\partial w_{ji}} \frac{\partial y_k}{\partial \mathbf{x}_k} (y_k^* - y_k) \tag{20}$$

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 $\mathbf{x}_i^*$. 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 design $\mathbf{x}_k^* = [x_{k1}^* \; x_{k2}^* \; \ldots \; x_{kq}^*]^T$ in which one or more of the decision variables are "out of bounds." For example, the meaning of the buffer frequency decision variable f described in Section 2 is defined only within the range [0, 1]; any value outside this range is out of bounds. Thus, even if the "out of bounds" design $\mathbf{x}_k$, 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 design 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):

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

y^*	$\equiv$	desired performance index value
$\mathbf{x}$	$\equiv$	design $[x_1 \; x_2 \; \ldots \; x_p]^T$ output by the inverse network given y^* as input
y	$\equiv$	output of the forward network given $\mathbf{x}$ as input
$\mathbf{x}^+$	$\equiv$	vector $[x_1^+ \; x_2^+ \; \ldots \; x_p^+]^T$ of upper bounds on the decision variables $[x_1 \; x_2 \; \ldots \; x_p]^T$
$\mathbf{x}^-$	$\equiv$	vector $[x_1^- \; x_2^- \; \ldots \; x_p^-]^T$ of lower bounds on the decision variables $[x_1 \; x_2 \; \ldots \; x_p]^T$
$\mathbf{H}^+$	$\equiv$	diagonal matrix given by $h_{ii}^+ = \begin{cases} 1, \text{ if } x_i \geq x_i^+ \\ 0, \text{ otherwise} \end{cases}$
$\mathbf{H}^-$	$\equiv$	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 subscript k from previous equations has been omitted for clarity. We will refer to the distal supervised learning approach with the modified square error (Eq. 21) as the *distal supervised learning with constraints*.

4. Application

The direct inverse, distal supervised learning and distal supervised learning with constraints methods were applied to the high-volume machining system design problem defined in Section 2.

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. 3).

# Samples	INPUT	OUTPUT			
	y	x	s	b	f
128		3, 10, 17, 24	1, 2	0, 50, 100, 150	0.25,0.50, 0.75, 1.00

TABLE 3. 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 cycles through the training data. The designs $\mathbf{x}_k$ prescribed by the trained network for a range of desired performance index values y_k^* are shown in Table 4.

k	INPUT	INVERSE NETWORK OUTPUT $\mathbf{x}_k$			
	y_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 4. Designs 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 4, 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 q (= 4) network outputs and all s (= 128) training pairs at the end of training, as defined by

$$RMSE = \left[\frac{1}{qs} \sum_{k=1}^{s} (\mathbf{v}_k^* - \mathbf{v}_k)^T (\mathbf{v}_k^* - \mathbf{v}_k) \right]^{\frac{1}{2}} \tag{22}$$

where $\mathbf{v}_k^*$ is a q-component vector specifying the desired network outputs for the k^{th} training pair and $\mathbf{v}_k$ 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 (Eq. 17) or equivalently the RMSE (Eq. 22) close to zero when one-to-many mappings are explicitly specified in the training data.

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. 5).

# Samples	INPUT				OUTPUT
	x	s	b	f	y
72	3, 10, 17, 24	1, 2	0, 100, 150	0.25, 0.75, 1.00	

TABLE 5. 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. 22, $q = 1$, $s = 72$) 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. Twenty-one performance index values y_k^* were provided to the inverse network during the training process, ranging from 0.0 to 0.4 in increments of 0.02. The designs $\mathbf{x}_k$ prescribed by the trained inverse network for a range of desired performance index values y_k^* are shown in Table 6.

k	INPUT y_k^*	INVERSE NETWORK OUTPUT 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 6. Designs 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 6 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.

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 Equation 21, with the upper and lower bounds of the decision variables set to 0 and 1 respectively. The designs x_k prescribed by the trained inverse network for a range of desired performance index values y_k^* are shown in Table 7.

	INPUT	INVERSE NETWORK OUTPUT x_k				FORWARD NET
k	y_k^*	x	s	b	f	OUTPUT y_k
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 7. Designs 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. The design solution prescribed for the highest desired performance index value y_k^*, 0.40, is, $x = 0$, $s = 1$, $b = 0$, $f = 0$. This is a coding for a system with all CNC machines and no buffers (Fig. 9). The efficiency of this system, as evaluated via simulation, is 0.2979, which exceeds the highest "known" efficiency of 0.2923 by 1.2%.

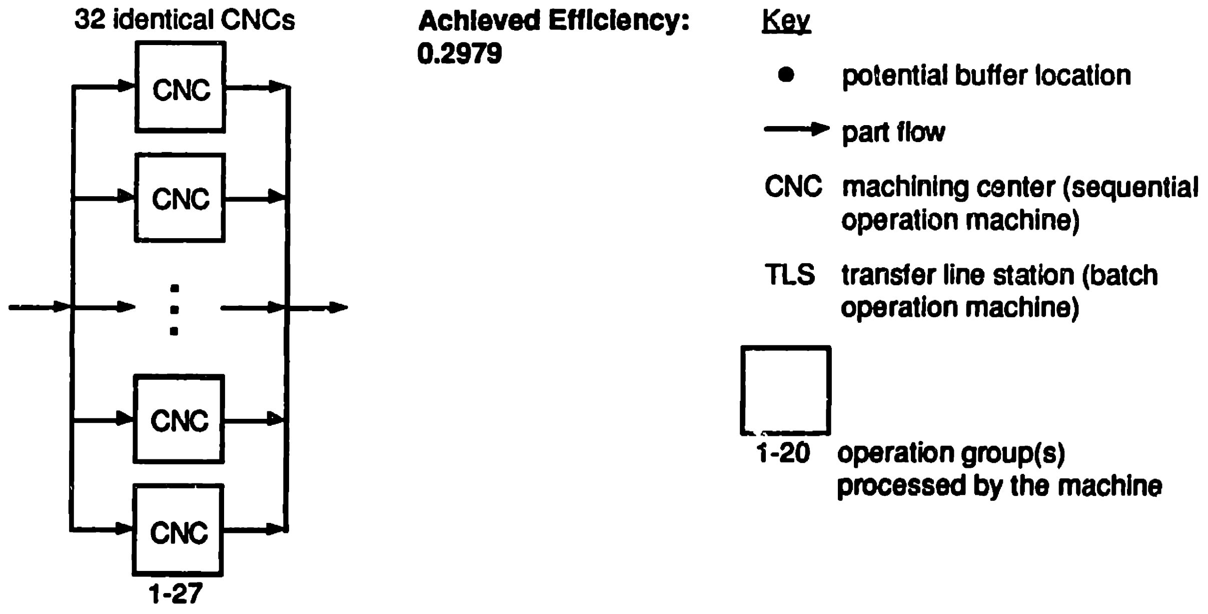
32 identical CNCs
CNC
CNC
CNC
CNC
1-27
Achieved Efficiency:
0.2979
Key
● potential buffer location
part flow
CNC machining center (sequential operation machine)
TLS transfer line station (batch operation machine)
1-20 operation group(s) processed by the machine

Figure 9. System prescribed by network trained via the distal supervised learning with constraints method.

5. Variable Part Change Frequency

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 methods discussed for predicting **x** for a desired performance, an evaluation of how the network's predictions change with respect to the assumptions in Table 2 is necessary. Because the performance evaluation used in this paper incorporates costs that occur over the life of the manufacturing system, including those incurred due to part design changes, it is of particular importance that the effect of design change frequency on the network output be examined.

Therefore two new sets of training pairs are generated with the same decision variables as found in Table 5, and modified efficiency values y. In the first training data set, the efficiency values y are modified such that $P_{\Delta}(5) = 1.0$ and $P_{\Delta}(n_{\Delta} \neq 5) = 0.0$ (see Table 2). All of the other parameters found in both tables 1 and 2 remain the same. Note that there is no need to simulate the training data again because the actual simulation run is independent of the design change frequency used in the calculation of the efficiency. This would not be the case if a parameter such as the mean time between failures, which is used in the simulation, was altered.

Again, all decision variables were normalized to fall within the range [0, 1]. The range of the efficiency values y were [0.0608, 0.4127]. The training procedure was the same as in the previous distal supervised learning with constraints method, with the exception of the y_k^* values used in the training of the inverse network. Because the range of efficiency values y exceeds 0.4000, the performance index values y_k^* used in the inverse network training were expanded to twenty-six values ranging from 0.0 to 0.5 in increments of 0.02. The forward training consisted of 50,000 cycles through the training data, with the final RMSE being 0.05. The designs $\mathbf{x}_k$ prescribed by the trained inverse network for a range of desired performance index values y_k^* are shown in Table 8.

k	INPUT	INVERSE NETWORK OUTPUT $\mathbf{x}_k$				FORWARD NET
	y_k^*	x	s	b	f	OUTPUT y_k
1	0.30	0.903	0.295	0.302	0.536	0.29
2	0.32	0.909	0.274	0.270	0.519	0.32
3	0.34	0.916	0.253	0.239	0.502	0.34
4	0.36	0.922	0.232	0.207	0.486	0.36
5	0.38	0.928	0.212	0.176	0.469	0.39
6	0.40	0.935	0.191	0.145	0.452	0.41
7	0.42	0.941	0.171	0.114	0.436	0.43
8	0.44	0.947	0.150	0.084	0.419	0.44
9	0.46	0.954	0.130	0.053	0.403	0.46
10	0.48	0.960	0.110	0.023	0.387	0.48
11	0.50	0.966	0.090	-0.007	0.371	0.49

TABLE 8. Designs output by a neural network trained via the distal supervised learning with constraints method. Part design change every five years.

The design solution prescribed for an efficiency of 0.5 is $x = 26$, $s = 0$, $b = 0$, $f = .371$. Note that x has been multiplied by 27, the maximum number of operations, in order to reverse the previous normalization. Since the buffer size is zero, the buffer frequency can be effectively interpreted as $f = 0$. This solution is implemented as a system with the first operation performed by the CNC, the remaining 26 performed by the TLS, and no buffers (Fig. 10)...As expected, the TLS becomes more desirable as the part design frequency decreases. The actual efficiency of the system, as determined by simulation, is 0.4282, which exceeds the previous highest efficiency by 3.8%.

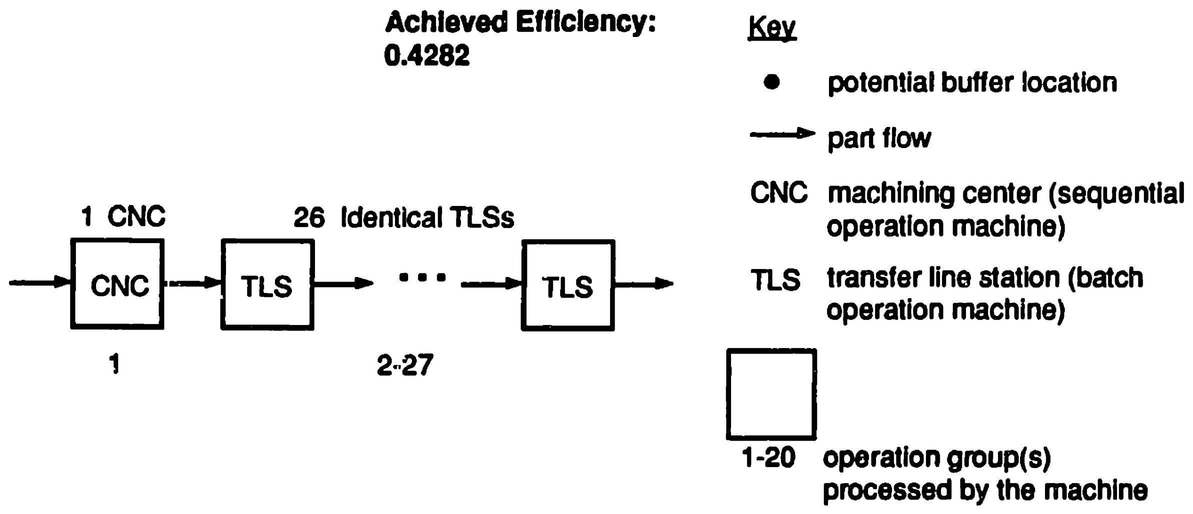
Achieved Efficiency:
0.4282
Key
● potential buffer location
part flow
CNC machining center (sequential operation machine)
TLS transfer line station (batch operation machine)
1-20 operation group(s) processed by the machine
1 CNC
26 Identical TLSs
CNC
TLS
TLS
1
2-27

Figure 10. System prescribed by network trained via the distal supervised learning with constraints method. Part design change every five years.

In the second set of data, efficiency values y were modified such that $P_\Delta(10) = 1.0$ and $P_\Delta(n_\Delta \neq 10) = 0.0$. The range of efficiency values y were [0.0629, 0.5575]. Using the previous training procedure, but again expanding the y_k^* values to accommodate the expanded range of y, the distal supervised learning with constraints method was applied. In this case, y_k^* was thirty-one values ranging from 0.0 to 0.6 in increments of 0.02. The forward training consisted of 50,000 cycles through the training data, with the final RMSE being 0.004. The designs x_k prescribed by the trained inverse network for a range of desired performance index values y_k^* are shown in Table 9.

k	INPUT y_k^*	INVERSE NETWORK OUTPUT x_k				FORWARD NET OUTPUT y_k
		x	s	b	f	
1	0.40	0.983	0.396	0.994	0.361	0.39
2	0.42	0.983	0.369	0.994	0.332	0.41
3	0.44	0.982	0.341	0.993	0.304	0.44
4	0.46	0.981	0.314	0.993	0.276	0.46
5	0.48	0.981	0.288	0.993	0.248	0.48
6	0.50	0.980	0.261	0.993	0.221	0.50
7	0.52	0.979	0.235	0.993	0.194	0.52
8	0.54	0.979	0.210	0.993	0.168	0.54
9	0.56	0.978	0.184	0.922	0.142	0.56
10	0.58	0.977	0.159	0.992	0.116	0.57
11	0.60	0.977	0.135	0.992	0.090	0.59

TABLE 9. Designs output by a neural network trained via the distal supervised learning with constraints method. Part design change every ten years.

The design solution prescribed by the network for an efficiency of 0.6 is $x = 26$, $s = 0$, $b = 0$, $f = 0$. The design variable s suggested by the network is not actually zero, but 0.135. Yet the only valid values for s are 0 and 1, so some interpretation must be employed much like the previous case when b was zero and f was not. Notice that this design solution is the same as the design solution proposed by the network trained with part design changes every five years (Fig. 10), the only difference being an achieved efficiency which is substantially higher (Fig. 11). The actual efficiency of the system is 0.5784, which exceeds the previous highest efficiency by 3.7%.

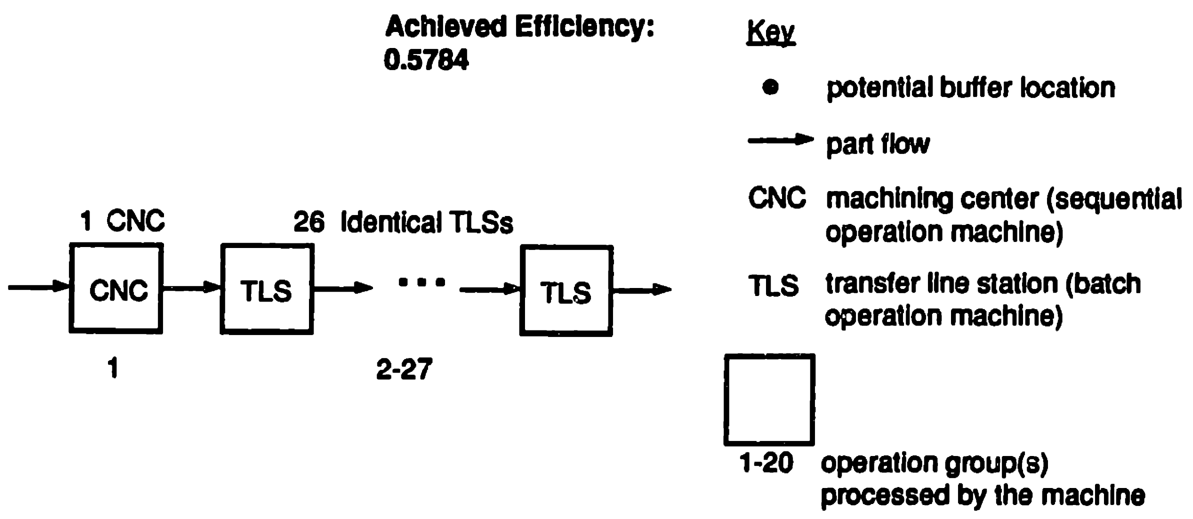
Achieved Efficiency:
0.5784
Key
● potential buffer location
part flow
CNC machining center (sequential operation machine)
TLS transfer line station (batch operation machine)
1-20 operation group(s) processed by the machine
1 CNC
26 Identical TLSs
CNC
TLS
TLS
1
2-27

Figure 11. System prescribed by network trained via the distal supervised learning with constraints method. Part design change every ten years.

6. Conclusions

The inverse mapping formulation of the manufacturing system design problem, which requires a model that takes as input a desired performance index value y_{goal} and outputs a suitable manufacturing system design $\mathbf{x}_{goal}$, 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 overcome this difficulty by selectively implementing one of the one-to-many mappings in accordance with given constraints on the mapping outputs. Systems prescribed by an inverse neural network model trained using this method are able to exceed the performance of the best systems with which they were trained.

An area of future research in this approach to manufacturing systems design is the establishment of a method for estimating the appropriate desired performance index value y_{goal}.

References

Ballard, J.L., S. Jagannathan, R. Hartline and B. Derksen, 1989, "Simulation Assisted Design and Installation of a Flexible Assembly System," *Proceedings of the 1989 Winter Simulation Conference*, pp. 904-907.

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.

Carrie, A.S., J.M. Moore, M. Roczniak, 1978, "Graph Theory and Computer Aided Facilities Design," *OMEGA, The International Journal of Management Science*, Vol. 6, No. 4, pp. 353-361.

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

Chryssolouris, G., 1992, *Manufacturing Systems: Theory and Practice*, Springer-Verlag, New York.

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

Draper, N.R. and H. Smith, 1981, *Applied Regression Analysis*, John Wiley & Sons, New York.

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.

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.

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.

Hogg, R.V. and J. Ledolter, 1987, *Engineering Statistics*, Macmillan, New York.

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.

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

Lee, H.F., M.M. Srinivasan and C.A. Yano, 1989, "An Algorithm for the Minimum Cost Configuration Problem in Flexible Manufacturing Systems," *Proceedings of the Third ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research and Applications*, Elsevier Science Publishers B.V., Amsterdam, pp. 85-90.

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

Malde, A.J. and K.M. Bafna, 1986, "Facilities Design Using a CAD System," *1986 International Industrial Engineering Conference Proceedings*, pp. 118-123.

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

Phadke, M.S., 1989, *Quality Engineering Using Robust Design*, Prentice Hall, Englewood Cliffs, New Jersey.

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

Ratkowsky, D.A., 1983, *Nonlinear Regression Analysis*, Marcel Dekker, New York.

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.

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

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.

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.

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